

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)

Д.А. ПОПОВА-КОВАРЦЕВА, Е.В. СОПЧЕНКО

ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ

Рекомендовано редакционно-издательским советом федерального государственного автономного образовательного учреждения высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева» в качестве учебного пособия для обучающихся по основной образовательной программе высшего образования по направлению подготовки 02.03.02. Фундаментальная информатика и информационные технологии

САМАРА
Издательство Самарского университета
2019

УДК 004(075)

ББК 32.97я7

П58

Рецензенты: д-р техн. наук, проф. С.А. Прохоров;

д-р техн. наук, проф. А.А. Тюгашев

Попова-Коварцева, Дарья Александровна

П58 **Основы проектирования баз данных:** учеб. пособие / *Д.А. Попова-Коварцева, Е.В. Сопченко.* – Самара: Изд-во Самарского университета, 2019. – 112 с.: ил.

ISBN 978-5-7883-1450-1

В предлагаемом пособии излагаются основы теории проектирования баз данных, рассматриваются следующие вопросы:

- преобразование информации о предметной области в данные. Уровни представления данных;

- характеристика моделей представления данных. Рассматриваются особенности проектирования реляционных баз данных, нормальные формы схем отношений;

- основные этапы проектирования баз данных: инфологическое, логическое и физическое проектирование. Рассматриваются графические нотации Питера Чена и нотация IDEF1X.

Теоретические сведения снабжены примерами. По каждой главе приводятся контрольные вопросы.

Подготовлено на кафедре программных систем Самарского университета и предназначено для студентов дневного отделения, обучающихся по направлению подготовки бакалавров «Фундаментальная информатика и информационные технологии», изучающих курсы «Базы данных» и «Проектирование баз данных». Может быть полезным для студентов других направлений подготовки, изучающих автоматизированные информационные системы и технологии баз данных.

УДК 004(075)

ББК 32.97я7

ISBN 978-5-7883-1450-1

© Самарский университет, 2019

ОГЛАВЛЕНИЕ

Список используемых сокращений	5
Введение	6
Глава 1. Введение в теорию баз данных.....	8
1.1. Автоматизированные информационные системы.....	8
1.2. Уровни представления данных	11
1.3. Основные модели данных.....	14
1.3.1. Иерархическая модель данных	15
1.3.2. Сетевая модель данных.....	18
1.3.3. Реляционная модель данных	18
1.3.4. Другие модели данных.....	19
1.4. Контрольные вопросы к главе 1.....	24
Глава 2. Реляционная модель данных.....	26
2.1. Базовые понятия реляционной модели данных.....	26
2.1.1. Понятия сущности, атрибута, домена	27
2.1.2. Ключи отношений	28
2.2. Отношение, схема отношения, схема БД.....	29
2.2.1. Свойства реляционных отношений	30
2.2.2. Виды реляционных отношений.....	32
2.2.3. Связывание таблиц.....	33
2.3. Целостность данных.....	34
2.4. Операции над отношениями.....	37
2.4.1. Теоретико–множественные операции над отношениями.....	38
2.4.2. Специальные реляционные операторы	41
2.5. Контрольные вопросы к главе 2.....	46
Глава 3. Проектирование базы данных	48
3.1. Инфологическое проектирование	51
3.1.1. Описание сущностей предметной области	51

3.1.2. Типы сущностей и иерархия наследования	53
3.1.3. Описание связей	56
3.1.4. Выбор ключа сущности	62
3.2. Логическое проектирование	63
3.2.1. Построение логической модели в нотации IDEF1X ...	64
3.2.2. Ограничения целостности	71
3.3. Физическое проектирование	73
3.4. Контрольные вопросы к главе 3.....	73
Глава 4. Особенности проектирования реляционных БД.....	75
4.1. Избыточное дублирование данных и аномалии	75
4.2. Правила преобразования ER–диаграммы в схему БД.....	78
4.3. Функциональные зависимости.....	84
4.3.1. Типы функциональных зависимостей.....	84
4.3.2. Аксиомы вывода функциональных зависимостей	87
4.3.3. Аксиомы вывода многозначных зависимостей	88
4.4. Нормальные формы схем отношений.....	91
4.4.1. Первая нормальная форма (1НФ)	91
4.4.2. Вторая нормальная форма (2НФ)	93
4.4.3. Третья нормальная форма (3НФ).....	95
4.4.4. Нормальная форма Бойса–Кодда (НФБК)	96
4.4.5. Четвертая нормальная форма (4НФ)	98
4.4.6. Денормализация	100
4.5. Пример нормализации до 3НФ	102
4.6. Контрольные вопросы к главе 4.....	107
Библиографический список	109

СПИСОК ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ

АИС – автоматизированная информационная система.

БД – база данных.

ДИСП – документальные информационно-поисковые системы.

ИС – информационная система.

ИМД – иерархическая модель данных.

КС – концептуальная схема.

МД – модель данных.

НФ – нормальная форма.

НФБК – нормальная форма Бойса-Кодда.

ООМД – объектно-ориентированная модель данных.

ОРМД – объектно-реляционная модель данных.

ПО – программное обеспечение.

ПрО – предметная область.

СУБД – система управления базами данных.

ФЗ – функциональная зависимость.

ФИПС – фактографические информационно-поисковые системы.

ER-диаграмма (от англ. entity-relation) – диаграмма «сущность-связь».

ВВЕДЕНИЕ

Информация и данные в современном обществе рассматриваются как жизненно важные, ключевые ресурсы. Человек создал естественные информационные системы, позволяющие поддерживать любые процессы исследования, планирования, производства, принятия решений.

Базы данных составляют основу информационных банковских систем, систем управления качеством, систем бронирования. Современное информационное общество немислимо без огромных массивов информации, которые хранят данные научных исследований, личную информацию миллионов пользователей интернет-пространства, корпоративную и прочую информацию.

Доступность баз данных, основанных на естественнo-языковых связях объектов реального мира, позволила расширить их применение от компьютеров и сложнейших технических систем до простейшей бытовой техники.

Накопление огромного объема информации и необходимость ее скоростной обработки сделали неизбежным появление суперкомпьютеров и параллельных вычислений, и, соответственно, развитие алгоритмов управления сверхбольшими базами данных.

Наконец, основанные на распределенных базах данных технологии блокчейн (от англ. blockchain–цепочка блоков) открыли новую эру в хранении и обработке информации, обеспечивая открытое децентрализованное хранение данных с многократным контролем целостности. С применением этих технологий становится возможной организация саморегулирующегося всемирного информационного пространства, в котором товары и услуги опла-

чиваются криптовалютами, проводятся свободные и открытые выборы лидеров государств, заключаются многомиллиардные контракты между корпорациями, совершаются покупки автомобилей и недвижимости обычными гражданами, поддерживаются авторские права на книги, музыку, другие произведения искусства, хранятся и широко используются данные медицинских исследований.

Учебное пособие охватывает основные темы, касающиеся проектирования баз данных. Книга ориентирована на использование в учебном процессе, поэтому теоретический материал дополнен примерами по каждой теме. Пособие состоит из глав, каждая из которых соответствует одной теме. В конце каждой главы приводятся контрольные вопросы.

В главе 1 рассмотрены основные понятия теории баз данных, трехуровневая архитектура ANSI/SPARC, описаны основные модели данных, их достоинства и недостатки. Глава 2 посвящена описанию реляционной модели данных. Рассмотрены базовые понятия реляционной модели, вопрос связывания таблиц и обеспечения целостности данных. Приводится описание операций реляционной алгебры с примерами их использования. В главе 3 подробно рассмотрены этапы проектирования баз данных, инфологическое, логическое и физическое проектирование. Приводится описание структурных нотаций отображения модели «сущность–связь» – нотация Питера Чена и нотация IDEF1X. Глава 4 посвящена вопросам проектирования баз данных методом нормальных форм. Приводится понятие функциональных зависимостей и их типов. Описаны нормальные формы схем отношений. В конце главы рассмотрен пример нормализации отношения до третьей нормальной формы.

Глава 1. ВВЕДЕНИЕ В ТЕОРИЮ БАЗ ДАННЫХ

1.1. Автоматизированные информационные системы

С самого начала развития вычислительной техники образовались два основных направления ее использования.

Первое направление – применение вычислительной техники для выполнения громоздких численных расчетов. Становление этого направления способствовало интенсификации методов численного решения сложных математических задач, развитию класса языков программирования, ориентированных на удобную запись численных алгоритмов.

Второе направление, появившееся несколько позже первого, – использование средств вычислительной техники в автоматических или автоматизированных информационных системах.

Информационная система (ИС) представляет собой программный комплекс, функции которого состоят в поддержке надежного хранения информации в памяти компьютера, выполнении специфических для данного приложения преобразований информации и/или вычислений, предоставлении пользователям удобного и легко осваиваемого интерфейса. Обычно объемы информации, с которыми приходится иметь дело таким системам, очень велики, а сама информация имеет довольно сложную структуру. Методы организации процессов обработки информации, реализуемые в концепции баз данных, позволили принципиально по-новому подойти к их воплощению в автоматизированных системах.

Автоматизированные информационные системы (АИС) – это системы, в которых представление, хранение и обработка информации осуществляются с помощью вычислительной техники. Центральным элементом любой АИС является база данных, которую необходимо постоянно поддерживать в актуальном состоянии.

Данные – представление объектов реального мира и их свойств в формализованном виде, пригодном для хранения, передачи, интерпретации или обработки. В случае использования данных для уменьшения неопределенности знаний о каком-либо объекте данные превращаются в информацию [6].

База данных (БД) – совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимая от прикладных программ [4].

Ведение базы данных – деятельность по обновлению, восстановлению и перестройке структуры базы данных с целью обеспечения ее целостности, сохранности и эффективности использования [4].

Система управления базами данных (СУБД) – совокупность программ и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных и обеспечения взаимодействия ее с прикладными программами [4].

В настоящее время АИС являются неотъемлемой частью современного инструментария информационного обеспечения различных видов деятельности и наиболее бурно развивающейся отраслью индустрии информационных технологий.

АИС можно классифицировать по типу хранимых данных:

1. Документальные информационно-поисковые системы (ДИСП) предназначены для хранения и обработки документальных данных: адресов хранения документов, наименований, описаний, а также текстов документов, графических изображений

(например, географических карт), звуковой информации (мелодии) и т.д. Такие данные представляются в неструктурированном виде.

2. Фактографические информационно-поисковые системы (ФИПС) хранят и обрабатывают структурированные данные в виде чисел и текстов. Над такими данными можно выполнять различные операции (нахождение суммы, минимума, максимума и т.п.).

Классическим примером АИС является банк данных.

Банк данных – это автоматизированная информационная система, состоящая из одной или нескольких баз данных и системы хранения, обработки и поиска информации в них [5].

Банк данных является сложной человеко-машинной системой, включающей в свой состав различные взаимосвязанные и взаимозависимые компоненты:

- *информационная компонента*: ядром банка данных является база данных;
- *программные средства* представляют собой сложный комплекс, обеспечивающий взаимодействие всех частей информационной системы при ее функционировании;
- *языковые средства различного назначения*: языки описания данных, языки манипулирования данными, языки запросов и другие языковые средства;
- *технические средства*: в качестве технических средств для банков данных чаще всего используются универсальные электронные вычислительные машины (ЭВМ), периферийные средства для ввода информации в базу данных и отображения выводимой информации. Если банк данных реализуется в сети, то необходимы соответствующие технические средства для обеспечения его сетевой работы;
- *организационно-методические средства* представляют собой различные инструкции, методические и регламенти-

рующие материалы, предназначенные для пользователей разных категорий, взаимодействующих с банком данных;

- *администраторы банка данных*, т.е. специалисты, обеспечивающие создание банка данных, его функционирование и развитие.

Основными функциями банка данных являются:

- хранение информации и организация ее защиты;
- периодическое изменение хранимых данных (добавление, удаление, обновление);
- поиск и отбор данных по запросам пользователей и прикладных программ;
- обработка найденных записей и вывод результатов в заданной форме.

1.2. Уровни представления данных

Понятия архитектуры и структуры являются одними из важнейших в теории БД и служат основой для понимания возможностей современных СУБД.

При описании предметной области, как правило, выделяют три уровня абстракции: *внешний*, *концептуальный* и *внутренний* (см. рис. 1.1).

Предложенная комитетом ANSI/SPARC (Комитет Планирования Стандартов и Норм Национального Института Стандартизации США) *трехуровневая архитектура* описания элементов хранимых данных обеспечивает их разделение и независимость.

Внутренний уровень наиболее приближен к физической системе непосредственного хранения данных. Он описывает, каким образом размещаются данные на устройствах хранения информации, и соответствует внутренней схеме БД.



Рис. 1.1. Трехуровневая архитектура ANSI/SPARC

Внутренняя схема базы данных – схема базы данных, определяющая представление данных в среде хранения и пути доступа к ним [4].

Для традиционного пользователя БД внутренний уровень, как правило, недоступен к просмотру и модификации [3].

Концептуальный уровень является переходным от внутреннего к внешнему уровням и, по сути, есть обобщенное представление данных для множества пользователей. На этом уровне содержание БД представляется в целом, в отличие от внешнего уровня, где конкретные данные представляются конкретному пользователю [3]. На концептуальном уровне важно описать предметную область, задав ее границы и выделив в ней основные объекты и их характеристики.

Предметная область информационной системы – это совокупность реальных процессов и объектов (сущностей), представляющих интерес для ее пользователей.

Для описания предметной области используют три основных конструктивных элемента – *сущность*, *атрибут* и *связь*.

Сущность – это представление набора реальных или абстрактных объектов (людей, вещей, мест, событий т. д.), которые имеют общие *атрибуты* или характеристики. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе.

Например, для предметной области УНИВЕРСИТЕТ в качестве типов сущностей могут рассматриваться СТУДЕНТ, ПРЕПОДАВАТЕЛЬ, ДИСЦИПЛИНА, КАФЕДРА и т.п.

Атрибут сущности – это именованная характеристика, являющаяся некоторым свойством сущности.

Для сущности СТУДЕНТ атрибутами могут являться номер студенческого билета, фамилия и имя, дата рождения и т.д.

Экземпляр сущности – это конкретный представитель данной сущности.

К примеру, экземпляром сущности СТУДЕНТ является студент Иванов Федор Степанович, номер зачетной книжки – 201734, год рождения – 2000, место проживания – город Самара, телефон – 2279090.

Связь – это некоторая ассоциация между двумя сущностями. Одна сущность может быть связана с другой сущностью или сама с собою. Связи позволяют по одной сущности находить другие сущности, связанные с ней.

Например, между сущностями СТУДЕНТ и ДИСЦИПЛИНА существует связь «изучает»: СТУДЕНТ *изучает* ДИСЦИПЛИНУ.

Концептуальная схема (КС) базы данных – схема базы данных, определяющая представление базы данных, единое для всех ее приложений и не зависящее от используемого в системе управления этой базой данных представления данных в среде хранения и путей доступа к ним [4].

Внешний уровень связан со способами представления данных непосредственно для пользователей. На внешнем уровне пользователю предоставляется возможность манипуляции данными в СУБД с помощью специального языка [3].

Внешняя схема базы данных – схема базы данных, поддерживаемая системой управления базы данных для приложений [4].

Трехуровневая модель предполагает, что полученная КС берется за основу для реализации специализированной СУБД. Такой подход в ряде случаев применяется в системах реального времени с целью достижения высокоскоростных характеристик доступа к реальным данным.

Однако чаще всего в качестве СУБД берется некая универсальная СУБД, ориентированная на поддержку той или иной модели данных.

1.3. Основные модели данных

Модель данных (МД) – совокупность правил порождения структур данных в базе данных, операций над ними, а также ограничений целостности, определяющих допустимые связи и значения данных, последовательность их изменения [4].

БД создается для достижения определенных целей исследования, и в зависимости от изменения или расширения целей модель БД может меняться.

Развитие теории и практики проектирования и эксплуатации баз данных сопровождается интенсивным развитием моделей данных. Самой первой МД, которая использовалась для построения концептуальных схем была *иерархическая модель*. Вслед за ней появились *сетевые модели*. Потом *ER-модели*, и, как итог развития моделей, возникли *реляционные* и *постреляционные модели*. Каждая из перечисленных моделей имеет свои достоинства и недо-

статки. Достоинство проявляется тогда, когда логика представления предметной области адекватно описывается выбранной моделью данных.

1.3.1. Иерархическая модель данных

В основу иерархической МД положен тот факт, что данные предметной области могут объединяться в группы по наличию у них тех или иных признаков или иных общих свойств. Эти свойства выделяются в предметной области в виде абстрактных данных (понятий), и между ними устанавливаются иерархические связи.

Иерархическая модель данных (ИМД) – это модель, в которой абстрактные понятия находятся в отношении предшествования таким образом, что каждому понятию соответствует только один предшественник (родитель). Только одна часть, называемая *корнем модели*, не имеет предшественника.

Структура данных представляет собой иерархическое дерево, концевые вершины которого задают конкретные данные, а все вершины более высокого уровня определяют признаки классификации конкретных данных.

Тип дерева состоит из одного «корневого» типа записи и упорядоченного набора из нуля или более типов поддеревьев (каждое из которых является некоторым типом дерева). Тип дерева в целом представляет собой иерархически организованный набор типов записи. *Иерархическая БД* состоит из упорядоченного набора деревьев; или, более точно, из упорядоченного набора нескольких экземпляров одного типа дерева (рис. 1.2).

Для такой базы данных определен полный порядок обхода: сверху вниз, слева направо. Особенностью реализации операций поиска в иерархической модели является то, что операция всегда начинается поиск с корневой вершины.

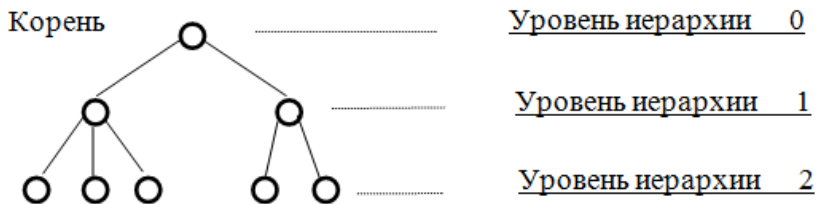


Рис. 1.2. Представление связей в иерархической модели

Примерами типичных операторов манипулирования иерархически организованными данными могут быть следующие:

- найти указанное дерево;
- перейти от одного дерева к другому;
- перейти от одной записи к другой внутри дерева;
- перейти от одной записи к другой в порядке обхода иерархии;
- вставить новую запись в указанную позицию;
- удалить текущую запись.

В иерархических БД автоматически поддерживается целостность ссылок между предками и потомками. Основное правило: никакой потомок не может существовать без своего родителя. Заметим, что аналогичное поддержание целостности по ссылкам между записями, не входящими в одну иерархию, не поддерживается. Кроме того, в иерархических системах поддерживается форма представлений БД на основе ограничения иерархии.

Рассмотрим пример иерархической базы данных для предметной области УНИВЕРСИТЕТ. В состав университета входят факультеты, которые, в свою очередь, *включают в себя* кафедры и *готовят* студентов *по направлениям*. Студенты зачислены в группы, соответствующие тому или иному направлению подготовки. Любой преподаватель является сотрудником одной кафедры.

На рис. 1.3 представлена иерархическая модель БД УНИВЕРСИТЕТ. Каждая вершина дерева соответствует одному типу объектов предметной области, который может характеризоваться произвольным количеством свойств (атрибутов).

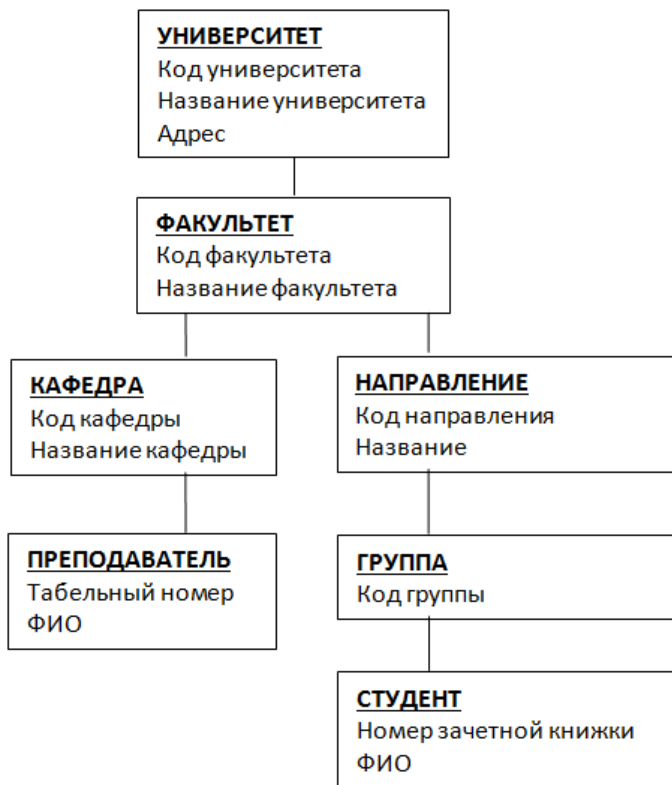


Рис. 1.3. Пример иерархической базы данных

Достоинство ИМД: данные, отражающие общие свойства совокупности конкретных данных не дублируются.

Недостаток ИМД связан с дублированием данных в случае их однотипности.

1.3.2. Сетевая модель данных

Сетевая модель так же, как и иерархическая, обладает весьма выразительными свойствами.

Необходимость сетевой модели проявляется тогда, когда одни и те же конкретные данные в рамках одной и той же предметной области классифицируются не одной, а несколькими системами классификации, то есть предметная область разбита на части, связанные между собой бинарными связями. Пример схемы простейшей сетевой БД показан на рис. 1.4.



Рис. 1.4. Представление связей в сетевой модели

Если в иерархических структурах запись-потомок должна иметь ровно одного предка, то в сетевых структурах данных потомок может иметь любое число предков. Связи между записями в сетевой МД выполняются в виде указателей, то есть каждая запись хранит ссылку на другую однотипную запись (или признак конца списка) и ссылки на списки подчиненных записей.

Достоинство сетевых МД: данные имеют четкую структуру.

Недостатком сетевых МД является большое количество дополнительной информации о связях.

1.3.3. Реляционная модель данных

Реляционная модель данных была предложена Э. Коддом и основана на понятии отношения (relation). Она является наиболее

распространенной и практически все современные СУБД ориентированы на такое представление данных.

Реляционная модель данных (РМД) – это модель, в которой данные можно представить в виде отношений, изменяющихся во времени. Традиционно в реляционных системах *отношением* называют таблицу, *кортежем* – строку таблицы, а *атрибутом* – столбец. При этом атрибуты имеют уникальные имена в рамках одного отношения.

Достоинство реляционной модели заключается в простоте для понимания, наглядности и удобстве физической реализации на ЭВМ.

Недостатки реляционной модели данных: модель не допускает представления объектов со сложной структурой, поскольку в ее рамках возможно моделирование лишь с помощью двумерных таблиц. Данные об объектах содержатся, как правило, во многих таблицах, что значительно замедляет обработку данных.

Более подробно реляционная модель будет рассмотрена в главе 2.

1.3.4. Другие модели данных

В последнее время при разработке БД активно используются такие модели, как постреляционная, объектно-ориентированная, объектно-реляционная и многомерная модели.

Постреляционная модель данных в общем случае представляет собой расширенную реляционную модель, снимающую ограничение неделимости значений полей. То есть, допускаются многозначные поля, значения которых состоят из подзначений. Набор значений многозначных полей считается самостоятельной таблицей, встроенной в основную.

Достоинство постреляционной модели данных: возможность представления связанных реляционных таблиц одной постреляционной таблицей.

Недостаток постреляционной модели: сложность в обеспечении целостности данных.

На рис. 1.5 приведен пример представления одной и той же информации с помощью реляционной (а) и постреляционной (б) модели. В первом случае данные хранятся в рамках двух таблиц: таблица ГРУППА_СТАРОСТА содержит данные о кодах групп и фамилиях кураторов в этих группах, таблица ГРУППА содержит информацию о кодах групп и фамилиях студентов, соответствующих группам. Таблицы связаны между собой по полю «Код_группы». В рамках постреляционной модели данные могут храниться в таблице, представленной на рис. 1.5,б).

Как видно из рис. 1.5, по сравнению с реляционной моделью в постреляционной модели данные хранятся более эффективно, а при обработке не требуется выполнять операцию соединения данных из двух таблиц.

а) ГРУППА_СТАРОСТА		б) ГРУППА		
Код группы	ФИО куратора	Код группы	ФИО куратора	ФИО студента
01	Антонов	01	Антонов	Павлова
02	Фурсов			Семенов
03	Иванов	02	Фурсов	Терехова
				Якимова
		03	Иванов	Лунев
				Мельников
				Окунев

ГРУППА	
Код группы	ФИО студента
01	Павлова
01	Семенов
02	Терехова
02	Якимова
03	Лунев
03	Мельников
03	Окунев

Рис. 1.5. Структура данных:

а – реляционная модель; б – постреляционная модель

Объектно-ориентированная модель данных (ООМД) представляет структуру, которую можно изобразить графически в виде дерева, узлами которого являются объекты.

Каждый объект характеризуется уникальным *идентификатором, состоянием и поведением*. Состояние объекта определяется множеством значений его свойств (атрибутов). Поведение объекта описывают методы, называемые процедурами. То есть, составной частью описания объекта являются процедуры, способные производить действия над атрибутами объекта в случае наступления тех или иных событий. Объекты могут объединяться в классы. Экземпляры одного класса отличаются лишь значениями своих свойств, но не своими методами. Методы устанавливаются при определении класса.

Для выполнения действий над объектами применяются объектно-ориентированные механизмы – *наследование, инкапсуляция, полиморфизм*.

Суть наследования состоит в том, что на основе существующего класса можно образовать новый класс объектов, который будет наследовать свойства родительского класса.

Доступ к данным осуществляется только лишь в соответствии с правилами поведения объекта, описываемыми методами (инкапсуляция).

Полиморфизм в объектно-ориентированных языках программирования означает способность одного и того же программного кода работать с разнотипными данными.

Основным достоинством ООМД является способность отображать информацию о сложных объектах. Эта модель обычно применяется для сложных предметных областей, при моделировании которых не хватает функциональности реляционной модели.

Недостаток ООМД: неудобство обработки больших массивов данных.

Объектно-реляционная модель данных (ОРМД) является гибридной моделью, сочетающей возможности реляционной модели с объектными свойствами данных. В ОРМД используются такие объектно-ориентированные компоненты, как инкапсуляция, полиморфизм, наследование и т.п. Отличительная особенность объектно-реляционной модели от ООМД состоит в том, что она основана на стратегии реляционной модели.

Многомерная модель предназначена для аналитической обработки информации. В источнике [1] приводится описание многомерной модели, в данной модели используются такие понятия, как агрегируемость, историчность, прогнозируемость данных.

Агрегируемость данных означает возможность их рассмотрения с различным уровнем обобщения.

Историчность обеспечивает высокий уровень статичности (неизменяемости) данных и их взаимосвязей, а также, в обязательном порядке, привязку данных к временным точкам.

Прогнозируемость данных подразумевает задание функций прогнозирования и применение их к различным интервалам времени.

Многомерная модель обладает высокой наглядностью. На рисунке 1.6 приведены реляционное (а) и многомерное (б) представление одних и тех же данных. В данных таблицах приведена информация о среднем балле студентов на момент наступления 11-ого числа в каждом из трех месяцев (октябрь, ноябрь, декабрь).

Основные понятия в многомерной модели – *измерение и ячейка*.

Измерение многомерной модели – это множество однотипных данных, образующих одну из граней многомерного гиперкуба. Примеры наиболее часто используемых временных измерений – дни, месяцы, кварталы и годы.

Ячейка многомерной модели – это поле, значение которого однозначно определяется фиксированным набором измерений. Тип поля чаще всего определен цифрой.

а)			б)			
Фамилия	Дата	Средний балл	Фамилия	11.10	11.11	11.12
Петров	11.10	4,34	Петров	4,34	4,67	4,78
Петров	11.11	4,67	Фадеев	4,7	4,8	5,0
Петров	11.12	4,78	Иванов	3,15	3,67	4,15
Фадеев	11.10	4,7				
Фадеев	11.11	4,8				
Фадеев	11.12	5,0				
Иванов	11.10	3,15				
Иванов	11.11	3,67				
Иванов	11.12	4,15				

Рис. 1.6 Реляционное(а) и многомерное(б) представление данных

Пример трехмерной модели данных приведен на рис. 1.7. В данном примере с помощью многомерной модели представлена информация о среднем балле студентов по определенному предмету на конкретную дату. Здесь используются следующие измерения:

- Фамилии студентов: Петров, Фадеев, Иванов;
- Название предмета: Алгебра, Физика, Программирование;
- Дата: 11.10, 11.11, 11.12.

На практике часто используют многомерную модель с размерностью больше, чем три. В этих случаях более удобно иметь дело с двумерными таблицами или графиками. Данные при этом представляют собой *срезы* из многомерного хранилища данных.

Срез представляет собой подмножество гиперкуба, полученного в результате фиксации одного или нескольких измерений [1].

Достоинством многомерной модели является удобство и эффективность анализа больших объемов данных, имеющих временную связь, а также быстрота реализации сложных запросов.



Рис. 1.7 Пример трехмерной модели данных

Недостаток этой модели состоит в громоздкости в случае ее использования для решения стандартных задач оперативной обработки.

1.4. Контрольные вопросы к главе 1

1. Дайте определения следующим понятиям: данные, база данных, СУБД, ведение базы данных.
2. В чем отличие данных от информации?
3. В чем отличие базы данных от банка данных и СУБД?
4. Назовите основные компоненты банка данных и их назначение.
5. Классифицируйте АИС по типу хранимых данных.
6. Что понимается под трехуровневой архитектурой ANSI/SPARC?
7. Дайте определения внешней схеме БД, концептуальной схеме БД, внутренней схеме БД.

8. Каковы особенности иерархической модели организации данных?

9. Каковы особенности сетевой модели организации данных?

10. Каковы особенности многомерной модели организации данных?

11. Каковы особенности постреляционной модели организации данных?

12. Многомерная модель данных. Достоинства и недостатки, пример реализации.

Глава 2. РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

2.1. Базовые понятия реляционной модели данных

Принципы реляционной модели были заложены в 1969–1970 гг. американским ученым Э. Коддом (E.F. Codd). 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.) [8].

Основными понятиями реляционных баз данных являются: тип данных, домен, атрибут, кортеж, отношение, первичный ключ. Покажем смысл этих понятий на примере отношения СТУДЕНТ, содержащего информацию о студентах некоторого университета (рис. 2.1).

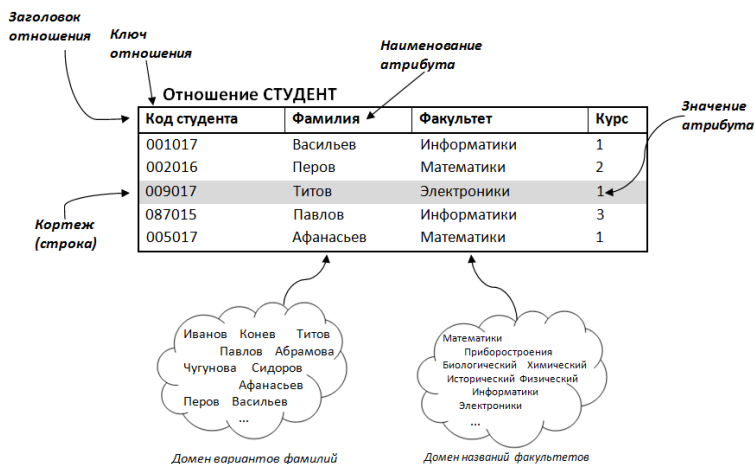


Рис. 2.1. Отношение и его компоненты

2.1.1. Понятия сущности, атрибута, домена

Коротко определим основные понятия реляционной модели следующим образом:

Сущность есть объект любой природы, данные о котором хранятся в базе данных. Данные о сущности хранятся в отношении.

Отношение представляет собой двумерную таблицу, содержащую некоторые данные.

Строки таблицы-отношения называются *кортежами*, а столбцы *атрибутами*.

Каждый атрибут в отношении имеет *наименование*, которое указывается в заголовочной части отношения (в именительном падеже единственного числа).

Степень отношения – это количество атрибутов отношения.

Кардинальное число отношения – количество кортежей в отношении.

Домен – множество допустимых значений атрибута. Все элементы домена относятся к одному типу данных и отвечают некоторому логическому условию (ограничению на диапазон значений). В самом общем виде домен определяется заданием некоторого базового типа данных, к которому относятся элементы домена, и произвольного логического выражения, применяемого к элементу типа данных. Если вычисление этого логического выражения дает результат «истина», то элемент данных является элементом домена.

Тип данных определяет множество значений и операций, которые могут быть применены к значениям. Понятие тип данных в реляционной модели полностью адекватно понятию типа данных в языках программирования. Обычно в современных реляционных БД допускается хранение символьных, числовых данных, битовых строк, специализированных числовых данных (таких как «день-

ги»), а также специальных временных данных (дата, время, временной интервал).

2.1.2. Ключи отношений

Ключ отношения – это такой атрибут (набор атрибутов) отношения, что в любой момент времени в отношении не существует строк, для которых значение или комбинация значений ключевых атрибутов являются одинаковыми. Ключ является уникальным идентификатором кортежей отношения.

Ключ, который состоит из двух и больше атрибутов, называется *составным ключом*.

Пусть R – отношение с атрибутами A_1, A_2, \dots, A_n . Говорят, что множество атрибутов $K=(A_i, A_j, \dots, A_k)$ отношения R является *возможным ключом* R тогда и только тогда, когда оно удовлетворяет двум независимым от времени условиям:

- уникальности: в произвольный заданный момент времени никакие два различные кортежа R не имеют одного и того же значения для атрибутов A_i, A_j, \dots, A_k .
- минимальности: ни один из атрибутов A_i, A_j, \dots, A_k 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 его возможных ключей (выбранный произвольным образом) принимается за *первичный ключ* (*primary key*). Остальные возможные ключи, если они есть, называются *альтернативными ключами*.

Например, для отношения СТУДЕНТ существует несколько возможных ключей, однозначно определяющих запись об одном студенте: номер зачетной книжки, номер паспорта, номер соци-

ального страхования. При выборе номера зачетной книжки в качестве первичного ключа два других номера становятся альтернативными ключами.

Также отношение может содержать *внешний ключ (foreign key)*. Внешний ключ предназначен для хранения значения первичного ключа другой таблицы с целью организации связи между этими таблицами.

2.2. Отношение, схема отношения, схема БД

Рассмотрим основные понятия реляционных баз данных более детально. Отношение является основной структурой, с помощью которой представляются данные в реляционной модели. На рис. 2.2 изображено множество элементов, из которых образуется отношение R , оно содержит две части: заголовок отношения и тело отношения.

$(A_1: D_1)$	$(A_2: D_2)$...	$(A_n: D_n)$
$(A_1: v_{11})$	$(A_2: v_{12})$		$(A_n: v_{1n})$
$(A_1: v_{21})$	$(A_2: v_{22})$		$(A_n: v_{2n})$
$(A_1: v_{m1})$	$(A_2: v_{m2})$		$(A_n: v_{mn})$

} **Заголовок отношения**
} **Тело отношения**

Рис. 2.2. Множество элементов отношения

Заголовок отношения состоит из такого фиксированного множества имен атрибутов A_1, A_2, \dots, A_n , что существует взаимно однозначное соответствие между этими атрибутами A_k и определяющими их доменами D_k ($k = 1, 2, \dots, n$, где n – степень отношения R).

Тело отношения – это структура, представляющая собой множество кортежей.

Кортеж – множество пар *<имя атрибута: значение атрибута>* следующего вида: $\{ \langle A_1:V_{i1} \rangle, \langle A_2:V_{i2} \rangle, \dots, \langle A_n:V_{in} \rangle \}$ ($i = 1, 2, \dots, t$, где t – кардинальное число отношения R).

Каждая такая пара $\langle A_k:V_{ik} \rangle$ соответствует паре $\langle A_k:D_k \rangle$ для каждого атрибута A_k в заголовке отношения. Для любой такой пары $\langle A_k:V_{ik} \rangle$ элемент V_{ik} является значением из домена D_k , который связан с атрибутом A_k .

Схема отношения – это именованное множество пар *<имя атрибута, имя домена (или типа, если понятие домена не подерживается)>*.

Степень или «арность» схемы отношения – мощность этого множества.

Схема БД – это набор именованных схем всех отношений, входящих в ее структуру.

Например, схема отношения СТУДЕНТ, приведенного на ис. 2.1, будет иметь вид: СТУДЕНТ (Код студента, Фамилия, Факультет, Курс).

Реляционной БД называют набор экземпляров конечных отношений, построенных в соответствии с логической схемой БД.

2.2.1. Свойства реляционных отношений

1. Отсутствие в отношении одинаковых кортежей

Это свойство следует из того факта, что тело отношения – это *математическое множество* (кортежей). В классической теории множеств по определению множество *не содержит одинаковых элементов*. Важным следствием того, что в отношении нет одинаковых строк (кортежей), является то, что в отношении *всегда существует, по крайней мере, один потенциальный ключ*. Действительно, так как кортежи уникальны, то обязательно комбинация

всех или части атрибутов будет обладать свойством уникальности, и, следовательно, может служить ключом отношения, однозначно идентифицирующим кортежи.

2. *Кортежи отношения никаким образом не упорядочены*

Это свойство следует из того, что тело отношения – это математическое множество, а простые множества в математике *не упорядочены*. Так в отношении, представленном на рис. 2.2, кортежи могли быть расположены в любом другом порядке, и, тем не менее, это все равно было бы то же самое отношение. Обращение к конкретному кортежу и его идентификация могут быть осуществлены *только по ключу отношения*.

3. *Атрибуты отношения никаким образом не упорядочены*

Это свойство следует из того факта, что заголовок отношения определен как простое математическое множество, а именно: множество пар *<имя атрибута: имя домена>*. Для ссылки на значение атрибута в кортеже отношения всегда используется имя атрибута.

4. *Значения всех атрибутов являются атомарными*

В реляционной модели домены, на которых определены атрибуты отношения и из которых «черпаются» фактические значения атрибутов, могут содержать только атомарные (неделимые, скалярные) значения. Другими словами, на пересечении столбца и строки таблицы, представляющей отношение, должно быть в точности *одно* значение, а не набор значений или какая-либо сложная (составная) структура значений.

Отношение, удовлетворяющее этому условию, называется *нормализованным*, или представленным в *первой нормальной форме* (другие нормальные формы будут рассмотрены позже). В качестве значений атрибутов ненормализованного отношения могут использоваться и более сложные структуры значений, например, другие отношения. Вся информация, представленная в ненормали-

зованном отношении ОТНОШЕНИЕ 1, может быть полностью представлена в виде нормализованного отношения ОТНОШЕНИЕ 2 (рис. 2.3) [9].

а) ОТНОШЕНИЕ 1			б) ОТНОШЕНИЕ 2		
Код студента	Дисциплина	Оценка	Код студента	Дисциплина	Оценка
001	Информатика	5	001	Информатика	5
	История	4	001	История	4
	Электротехника	5	001	Электротехника	5
	Физика	4	001	Физика	4
002	Информатика	3	002	Информатика	3
	Базы данных	4	002	Базы данных	4
	Физика	4	002	Физика	4
003	Математика	5	003	Математика	5
	Физика	5	003	Физика	5
	Электротехника	3	003	Электротехника	3

Рис. 2.3. Примеры ненормализованного(а)
и нормализованного(б) отношений

2.2.2. Виды реляционных отношений

В реляционных системах поддерживаются несколько *видов отношений*.

Именованное отношение – это отношение, определенное в СУБД с помощью специального оператора создания отношения, которому при этом присваивается имя, уникальное в конкретной базе данных [2].

Базовое отношение – именованное отношение, которое является автономным и не определяется или не выводится из других отношений, т. е. которое не является *производным*.

Производное отношение – отношение, которое определено через другие (как правило, базовые) отношения путем использования средств СУБД, то есть получается в результате преобразования каких-либо других отношений.

Представление – это именованное виртуальное производное отношение, которое существует в базе данных исключительно через свое определение в терминах других отношений. При изменении значений данных в исходных отношениях будут изменяться и данные, видимые через конкретное *представление*.

Результат запроса – это неименованное производное отношение, содержащее данные – результат конкретного запроса. Результат запроса в БД не хранится, а существует только до тех пор, пока он необходим пользователю.

Хранимое отношение – отношение, физически поддерживаемое в памяти компьютера. К хранимым, в большинстве случаев, относятся базовые отношения [3].

2.2.3. Связывание таблиц

В реляционных СУБД для указания связей таблиц производят операцию их связывания. Установление связи между таблицами облегчает доступ к данным. Связывание таблиц при выполнении таких операций, как поиск, просмотр, редактирование, выборка и подготовка отчетов, обычно обеспечивает возможность обращения к произвольным полям связанных записей. Это уменьшает количество явных обращений к таблицам данных и число манипуляций в каждой из них.

При связывании двух таблиц выделяют *основную* и *дополнительную (подчиненную)* таблицы. Логическое связывание таблиц производится с помощью ключа связи.

Ключ связи, по аналогии с обычным ключом таблицы, состоит из одного или нескольких полей, которые в данном случае называют *полями связи*. Суть связывания состоит в установлении соответствия полей связи основной и дополнительной таблиц. Поля связи основной таблицы могут быть обычными и ключевыми.

В зависимости от того, как определены поля связи основной и дополнительной таблиц (как соотносятся ключевые поля с полями связи), между двумя таблицами в общем случае могут устанавливаться следующие четыре основных вида связи:

- один – один (1:1);
- один – много (1:M);
- много – один (M:1);
- много – много (M:N).

Более подробно виды связей будут описаны ниже.

2.3. Целостность данных

Одним из базовых понятий теории баз данных является понятие целостности БД, т.е. соответствие данных, хранящихся в ней, ее структуре, логике функционирования и иным явно заданным правилам.

Ограничение целостности – это правило, налагающее некоторое ограничение на возможное состояние базы данных (значения данных и/или связей между ними).

Поддержка целостности в реляционных БД основана на выполнении следующих требований:

1. Первое требование называется *требованием целостности сущностей*. Объекту или сущности реального мира в реляционных БД соответствуют кортежи отношений. Конкретно требование состоит в том, что любой кортеж любого отношения отличим от любого другого кортежа этого отношения, т.е., другими словами, любое отношение должно обладать определенным первичным ключом. Это требование автоматически удовлетворяется, если в системе не нарушаются базовые свойства отношений.

2. Второе требование называется *требованием целостности по ссылкам*. Очевидно, что при соблюдении нормализованности

отношений сложные сущности реального мира представляются в реляционной БД в виде нескольких кортежей нескольких отношений. Связь между отношениями осуществляется с помощью миграции ключа. *Миграцией ключа* называется процесс переноса простого или составного первичного ключа одной сущности в другую. Внешний ключ существует для обеспечения непротиворечивости данных внутри БД, т.е. значение внешнего ключа не может быть таким, которого нет среди значений первичного ключа связанной таблицы.

В базах данных отношения, как правило, связаны друг с другом. Например, в базе данных ФАКУЛЬТЕТ отношение СТУДЕНТ (Код студента, Фамилия, Факультет, Курс, Код группы) связано с отношением ГРУППА (Код группы, Специальность, Староста). *Первичными ключами (primary key)* отношений СТУДЕНТ и ГРУППА являются атрибуты Код студента и Код группы соответственно. Значение атрибута Код группы в отношении СТУДЕНТ допустимо только в том случае, если такое значение имеется в качестве значения первичного ключа отношения ГРУППА. В этом случае атрибут Код группы в отношении СТУДЕНТ является *внешним ключом (foreign key)*, ссылающимся на первичный ключ – Код группы отношения ГРУППА (см. рис. 2.4).

СТУДЕНТ					ГРУППА		
<u>Код студента</u> (primary key)	Фамилия	Факультет	Курс	<u>Код группы</u> (foreign key)	<u>Код группы</u> (primary key)	Специальность	Староста
1016	Иванов	Математики	1	111	111	Прикладная математика	Козлова
2016	Векшин	Математики	1	111	112	Прикладная математика	Васильев
3016	Мальцев	Математики	1	112	211	Фундаментальная информатика	Савельева
4016	Чиклаев	Информатики	1	211	221	Информационная безопасность	Кузнецов
5016	Якушин	Информатики	1	211	231	Информационная безопасность	Конева
1017	Петров	Информатики	2	221			
1015	Демин	Информатики	3	231			

Рис. 2.4. Пример связывания отношений
(с указанием первичного и внешнего ключей)

Правило ссылочной целостности подразумевает состояние БД в конкретный момент времени. Но как избежать временных некорректных ситуаций, которые могут возникнуть при обновлении данных в БД? Необходимо заранее обдумать вопрос о том, что произойдет при попытке удаления кортежей из отношения, на которое ссылается внешний ключ. При этом существуют следующие вероятные возможности:

а) *ограничить*, т.е. не удалять, пока пользователь не удалит ссылающиеся кортежи, т.е. *отложить* удаление;

б) *каскадировать*, т.е. *удалить*, удаляя все соответствующие ссылающиеся кортежи.

Наконец нужно предусмотреть технологию того, что будет происходить при попытке обновления первичного ключа отношения, на которое ссылается некоторый внешний ключ. Здесь имеются те же возможности, как и при удалении:

а) *ограничить*, т.е. *отложить* до удаления значений ссылающихся кортежей;

б) *каскадировать*, т.е. *обновить* во всех ссылающихся кортежах.

Таким образом, для каждого внешнего ключа в БД должны предусматриваться не только атрибут или комбинация атрибутов, составляющих этот внешний ключ, но также и варианты поведения БД в рассмотренных выше случаях.

NULL-значения

Осложнения при обеспечении целостности данных могут быть вызваны неопределенными или отсутствующими значениями. Для решения проблем отсутствия значений Э. Кодд предложил ввести специальные метки, названные им NULL-значениями, которые определил так: если данный кортеж имеет NULL-значение данного атрибута, то это означает, что в нем значение атрибута отсутствует. Это не то же, что числовой ноль

или пробел, это вообще не значение, а только метка – обозначение отсутствия любого значения.

Большинство современных реляционных СУБД поддерживают NULL–значения.

При этом ни один элемент первичного ключа базового отношения не может быть NULL–значением. Это правило касается только *базовых* отношений (т.е. не вычисляемых, не производных), оно применимо только для *первичных* ключей, а для альтернативных ключей NULL–значения могут быть запрещены или разрешены [11].

2.4. Операции над отношениями

В реляционной модели для описания операций, которые можно осуществлять над данными, существует два эквивалентных математических аппарата. Это реляционная алгебра и реляционное исчисление.

Основы реляционной алгебры были заложены Э. Коддом. Реляционная алгебра задает набор операторов для выполнения операций над реляционными отношениями. Операции реляционной алгебры Кодда можно разделить на две группы: *базовые теоретико-множественные* и *специальные реляционные*.

1. Первая группа операций включает в себя операторы, представляющие собой традиционные операции над множествами, а именно:

- объединение отношений;
- пересечение отношений;
- разность (вычитание) отношений;
- декартово произведение отношений.

2. Вторая группа представляет собой специальные реляционные операторы:

- выборка (селекция);
- проекция;
- соединение отношений;
- деление отношений.

Операции реляционной алгебры могут выполняться над одним (унарная операция) или двумя отношениями (бинарная операция). При выполнении бинарной операции участвующие в операциях отношения должны быть совместимы по структуре. Отношения называются *совместимыми*, если число и состав их атрибутов совпадают, то есть схемы отношений одинаковые.

2.4.1. Теоретико–множественные операции над отношениями

1. *Объединение (UNION)* двух совместимых отношений $R1$ и $R2$ одинаковой размерности – это отношение R , содержащее все элементы исходных отношений. Повторяющиеся кортежи не включаются.

Пример 2.1. Объединение отношений (рисунок 2.5)



Рис. 2.5. Пример объединения отношений

2. *Пересечение (INTERSECT)* двух совместимых по типу отношений $R1$ и $R2$ одинаковой размерности – это отношение R с телом, включающим в себя кортежи, одновременно принадлежащие обоим отношениям.

Пример 2.2. Пересечение отношений (рис.2.6)

Пусть входные отношения $R1$ и $R2$ имеют тот же вид, что в Примере 2.1, тогда на выходе получим:

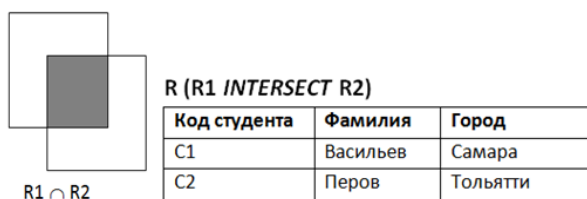


Рис. 2.6. Пример пересечения отношений

3. *Вычитание (MINUS)* двух совместимых по типу отношений $R1$ и $R2$ одинаковой размерности – это отношение R с телом, состоящим из множества всех кортежей, принадлежащих отношению $R1$ и не принадлежащих отношению $R2$.

Пример 2.3. Вычитание отношений (рис. 2.7)

Пусть входные отношения $R1$ и $R2$ опять имеют тот же вид, что в Примере 2.1, тогда на выходе получим:



Рис. 2.7. Пример вычитания отношений

4. *Декартово произведение отношений* (TIMES) – это бинарный оператор, предназначенный для комбинирования двух отношений.

Пусть $R1$ - отношение степени k_1 , а $R2$ - отношение степени k_2 .

$R = R1 \times R2$ будет представлять собой множество кортежей таких, что первые k_1 элементов кортежа отношения R составляют кортежи отношения $R1$, а последующие k_2 элементов кортежа из R составляются кортежами из $R2$. Сочетание кортежей из $R1$ и $R2$ при образовании кортежа R осуществляется по правилам декартова произведения. Здесь $R1$ и $R2$ могут иметь разные схемы. Степень отношения R равна сумме степеней отношений операндов $R1$ и $R2$. Мощность отношения R равна произведению мощностей $R1$ и $R2$.

Пример 2.4. Произведение отношений (рис. 2.8)

R1		R2	
Код студента	Фамилия	Код дисциплины	Название
С3	Титов	Д1	Информатика
С4	Павлов	Д2	Базы данных
		Д3	Физика

R (R1×R2)			
Код студента	Фамилия	Код дисциплины	Название
С3	Титов	Д1	Информатика
С3	Титов	Д2	Базы данных
С3	Титов	Д3	Физика
С4	Павлов	Д1	Информатика
С4	Павлов	Д2	Базы данных
С4	Павлов	Д3	Физика

Рис. 2.8. Пример произведения отношений

Следует заметить, что с практической точки зрения операция декартова произведения сама по себе не имеет большого значения, так как она не приводит к получению какой-либо новой информации.

2.4.2. Специальные реляционные операторы

1. *Выборка* ($R \text{ WHERE } f$) отношения R по формуле f представляет собой новое отношение, заголовок которого совпадает с заголовком отношения R , а тело содержит множество кортежей, являющихся подмножеством множества кортежей отношения R , для которых формула f принимает значение истина.

Здесь f – логическая формула (предикат), принимающая значение «истина» или «ложь», образованная операндами, представляющими собой имена атрибутов отношений R или литералы, а также скалярными операторами сравнения ($=, <>, >, <, >=, <=$) и логическими операторами И, ИЛИ, НЕ.

Выбор – унарная операция.

Пример 2.5. Выборка

На рис. 2.9 представлено исходное отношение R и два отношения, полученные с помощью операторов:

- 1) $R \text{ WHERE Курс} = 1$;
- 2) $R \text{ WHERE Факультет} = \text{«Информатики» AND Курс} = 3$.

R

Код студента	Фамилия	Факультет	Курс
001017	Васильев	Информатики	1
002016	Перов	Математики	2
009017	Титов	Электроники	1
087015	Павлов	Информатики	3
005017	Афанасьев	Математики	1

R WHERE Курс = 1

Код студента	Фамилия	Факультет	Курс
001017	Васильев	Информатики	1
009017	Титов	Электроники	1
005017	Афанасьев	Математики	1

R WHERE Факультет = «Информатики » AND Курс = 3

Код студента	Фамилия	Факультет	Курс
087015	Павлов	Информатики	3

Рис. 2.9. Пример выборки из отношения

2. *Проекция* отношения R по атрибутам X, Y, \dots, Z обозначается $R[X, Y, \dots, Z]$, где каждый из указанных атрибутов принадлежит отношению R , – это отношение с заголовком $\{X, Y, \dots, Z\}$ и телом, содержащим кортежи отношения R , за исключением повторяющихся кортежей. Повторение одинаковых атрибутов в списке X, Y, \dots, Z запрещается.

С помощью оператора проекции получается «вертикальное» подмножество атрибутов отношения R с последующим исключением дублирующих кортежей (если они возникают).

Пример 2.6. Проекция

Пусть исходным отношением будет отношение R из Примера 2.5. Тогда по атрибутам *Фамилия* и *Факультет* получим следующую проекцию (рис. 2.10):

R[Фамилия, Факультет]	
Фамилия	Факультет
Васильев	Информатики
Перов	Математики
Титов	Электроники
Павлов	Информатики
Афанасьев	Математики

Рис. 2.10. Пример операции проекции

3. *Соединение* – операция реляционной алгебры, связывающая таблицы. Операция соединения используется для связывания данных между таблицами. Это, возможно, наиболее важная функция любого языка баз данных.

Θ-соединение отношений. Пусть отношения $R1$ и $R2$ не имеют атрибутов с одинаковыми именами, а символ Θ (греческая бук-

ва тэта) обозначает один из операторов сравнения: =, <>, >, <, <=, >=. Тогда Θ -соединением отношения $R1$ по атрибуту X с отношением $R2$ по атрибуту Y является отношение, кортежи которого получаются в результате декартова произведения отношений $R1$ и $R2$ и для которых условие $X \Theta Y$ принимает значение истина.

Другими словами, ТЭТА-соединение это $(R1 \text{ TIMES } R2) \text{ WHERE } X \Theta Y$.

Очевидно, что сравниваемые атрибуты X и Y должны быть определены на общем домене для того, чтобы операция сравнения имела смысл.

Пример 2.7. Θ -соединение

Рассмотрим некоторую авиакомпанию, в которой хранятся данные о пилотах (отношение $R1$) и самолетах (отношение $R2$) (рисунок 2.11). Пусть пилотам и самолетам присвоен некий статус (атрибуты в соответствующих отношениях – 'Статус_П' и 'Статус_С'). Рабочий процесс организован таким образом, что пилоты имеют право управлять только теми самолетами, у которых статус не выше статуса пилота.

R1 (Пилоты)		R2 (Самолеты)	
Пилот	Статус_П (X)	Самолет	Статус_С (Y)
Иванов	4	Boeing 737-100	3
Федоров	3	ТУ-204	2
Игнатьев	2	Ан-140	1
Осипов	1	A380	4

Рис. 2.11. Исходные отношения $R1$ и $R2$ для операции Θ -соединения

Ответ на вопрос «какие пилоты могут управлять какими самолетами?» дает Θ -соединение отношений $R1$ и $R2$: $(R1 \text{ TIMES } R2) \text{ WHERE } X \geq Y$ (рис. 2.12).

В результате получим отношение R :

R

Пилот	Статус_П	Самолет	Статус_С
Иванов	4	Boeing 737-100	3
Иванов	4	ТУ-204	2
Иванов	4	Ан-140	1
Иванов	4	A380	4
Федоров	3	Boeing 737-100	3
Федоров	3	ТУ-204	2
Федоров	3	Ан-140	1
Игнатьев	2	ТУ-204	2
Игнатьев	2	Ан-140	1
Осипов	1	Ан-140	1

Рис. 2.12. Отношение R , полученное в результате Θ -соединения отношений R_1 и R_2

Частным случаем соединения являются *эквисоединение* и *естественное соединение*.

Операция *эквисоединения* характеризуется тем, что оператор сравнения в этом соединении – это знак равенства.

Операция *естественного соединения* (операция R_1 JOIN R_2) применяется к двум отношениям, имеющим общий атрибут (простой или составной). Этот атрибут в отношениях имеет одно и то же имя (совокупность имен) и определен в одном и том же домене (доменах).

Результатом операции естественного соединения является отношение R , которое представляет собой проекцию эквисоединения отношений R_1 и R_2 по общему атрибуту на объединенную совокупность атрибутов обоих отношений.

Пример 2.8. Естественное соединение

Пусть необходимо найти естественное соединение отношений R_1 (ПИЛОТЫ) и R_2 (САМОЛЕТЫ) из Примера 2.7, соединение

нужно произвести по общему атрибуту, характеризующему статус (в отношении $R1$ – это Статус_П, а в $R2$ – Статус_С). Поскольку условие операции требует одинаковости имен атрибутов, по которым выполняется соединение, то применяется операция RENAME переименования атрибутов.

Операция переименования позволяет изменить имя атрибута отношения и имеет вид:

RENAME<исходное отношение><старое имя атрибута> AS <новое имя атрибута>.

В результате естественного соединения отношений $R1$ и $R2$ получим (рисунок 2.13):

($R1$ RENAME Статус_П AS Статус) JOIN ($R2$ RENAME Статус_С AS Статус)

Пилот	Статус	Самолет
Иванов	4	A380
Федоров	3	Boeing 737-100
Игнатьев	2	ТУ-204
Осипов	1	Ан-140

Рис. 2.13. Результат естественного соединения отношений $R1$ и $R2$

4. Деление отношения.

Пусть $R1$ и $R2$ – два отношения.

Тогда отношение $R := R1/R2$ может быть получено следующим образом:

1. Результирующее отношение R будет содержать только те атрибуты делимого $R1$, которых нет в делителе $R2$.

2. В результирующее отношение R включаются только те недублирующиеся кортежи из $R1$, декартово произведение кото-

рых с делителем $R2$ содержится в делимом (является подмножеством делимого).

Пример 2.9. Деление

Пусть даны два отношения $R1$ и $R2$. Требуется получить список студентов, изучающих все дисциплины, приведённые во втором отношении. Эта операция может быть выполнена путем деления первого отношения на второе (рисунок 2.14).

R1		R2	R (R1 DIVIDE BY R2)
Студент	Дисциплина	Дисциплина	Студент
Иванов	Информатика	Информатика	Иванов
Иванов	Физика	Физика	Зубков
Иванов	Математика		
Иванов	Электротехника		
Иванов	История		
Федоров	Информатика		
Федоров	История		
Федоров	Математика		
Орлов	История		
Орлов	Математика		
Орлов	Физика		
Орлов	БД		
Зубков	Информатика		
Зубков	Математика		
Зубков	Физика		

Рис. 2.14. Пример операции деления

2.5. Контрольные вопросы к главе 2

1. Что понимается в реляционной алгебре под типом данных, доменом, атрибутом, схемой отношения, схемой БД, кортежем, отношением?
2. Ключ отношения: возможный, первичный, альтернативный.
3. Каковы основные свойства отношений?

4. Назовите виды реляционных отношений.
5. Что понимается в реляционной алгебре под степенью (рангом) отношения, θ -сравнимыми атрибутами, эквивалентными схемами отношений, основным и подчиненным отношением, первичным и внешним ключом?
6. Какие типы связей могут быть между двумя отношениями?
7. Понятие целостности БД. Что в реляционной алгебре понимается под целостностью сущностей и ссылок?
8. Возможные проблемы ограничения целостности и способы их решения. Для чего используют NULL-значения?
9. Какие основные операции в реляционной алгебре используются над отношениями?
10. Приведите примеры выборки и проекции.
11. Приведите примеры эквисоединения, деления отношений.

Глава 3. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Проектирование базы данных – одна из наиболее сложных и ответственных задач, связанных с созданием информационной системы. В результате её решения должны быть определены: содержание БД, эффективный для всех её будущих пользователей способ организации данных и инструментальные средства управления данными.

Проектируемая база данных должна обеспечивать:

- хранение всей необходимой информации;
- возможность получения данных по всем необходимым запросам;
- сокращение избыточности и дублирования данных;
- целостность данных (правильность их содержания): исключение противоречий в содержании данных, исключение их потери и т.д.

Основными задачами при проектировании баз данных являются:

1. Определение границ проектируемой предметной области (постановка задания), построение структуры данных предметной области – концептуальное проектирование (инфологическая модель).

2. Выбор оптимальной СУБД для заданной предметной области. Описание БД на языке СУБД – логическое проектирование (даталогическая модель).

3. Определение типов используемых данных и методов доступа к ним (запросы, отчёты, формы) – физическое проектирование (физическая модель).

Проектирование БД происходит в несколько этапов. На рис. 3.1 приведена схема этапов проектирования базы данных.



Рис. 3.1. Этапы проектирования БД

На *первом этапе* проектирования баз данных производится *системный анализ* предметной области. Эта задача предшествует проектированию базы данных, и решают ее аналитики. Однако проектировщик баз данных должен знать результаты выполнения этой задачи и уметь правильно интерпретировать их в ходе проектирования. Результаты анализа предметной области, а именно, модели данных на семантическом уровне, являются исходными данными для решения задач проектирования базы данных [10].

На данном этапе рассматриваются цели и задачи, анализируются информационные потребности всех категорий будущих пользователей базы данных. Рассматриваются формы входных и выходных потоков данных. Затем уточняются алгоритмы и процедуры обработки хранимых данных. Формируются требования, ко-

торым должна удовлетворять проектируемая база данных, и определяется список объектов предметной области, свойства которых будут использоваться при разработке базы данных.

На *второй стадии* проектирования выполняется моделирование данных. *Моделирование данных* – это процесс создания логической структуры данных.

На *третьем этапе* производится выбор СУБД. Теоретически при выборе СУБД нужно принимать во внимание десятки факторов. Но практически разработчики руководствуются лишь собственной интуицией и несколькими наиболее важными критериями, к которым, в частности, относятся:

- тип модели данных, которую поддерживает данная СУБД, её адекватность потребностям рассматриваемой предметной области;
- характеристики производительности системы;
- запас функциональных возможностей для дальнейшего развития ИС;
- степень оснащённости системы инструментарием для персонала администрирования данными;
- удобство и надежность СУБД в эксплуатации;
- стоимость СУБД и дополнительного программного обеспечения.

После того как выбор СУБД завершён, необходимо приступить к следующему, *четвертому этапу*, – проектированию даталогической модели базы данных. При формировании даталогической схемы каждая из определённых в концептуальной схеме сущностей отображается в таблицу, которая является одним отношением. При этом следует учитывать ограничения на размер таблиц, накладываемые конкретной СУБД.

На *пятом этапе* необходимо в конкретной СУБД, выбранной ранее, реализовать базу данных по той информации, которую со-

брали, обработали и подготовили (на предыдущих этапах проектирования базы данных). Описываются модули, их назначение, а также структура модулей.

Основные этапы проектирования рассмотрим более подробно.

3.1. Инфологическое проектирование

3.1.1. Описание сущностей предметной области

Цель инфологического проектирования состоит в получении моделей, отражающих предметную область и информационные потребности пользователей. В качестве инструмента для построения модели данных на этапе проектирования можно выбрать модель «сущность–связь» (Entity–Relationship).

Метод «сущность–связь» по-другому называют методом «ER–диаграмм» (ER –аббревиатура от слов *Entity* (сущность) и *Relation* (связь)).

Модель «сущность–связь» основывается на некоторой важной семантической информации о реальном мире и предназначена для логического представления данных. Она определяет значения данных в контексте их взаимосвязи с другими данными. Из модели «сущность–связь» могут быть порождены все существующие модели данных (иерархическая, сетевая, реляционная, объектная), поэтому она является наиболее общей.

Данная модель была предложена Питером Ченом в 1976 году. Важным свойством модели «сущность–связь» является то, что она может быть представлена в виде графической схемы. Это значительно облегчает анализ предметной области. Используются и другие структурные нотации отображения модели «сущность–связь» (нотация Ричарда Баркера, IDEF и другие).

Существуют различные варианты отображения ER–диаграмм, но все варианты диаграмм «сущность–связь» исходят из одной идеи – рисунок всегда нагляднее текстового описания. ER–диаграммы используют графическое изображение *сущностей* предметной области в виде прямоугольников, их *свойств (атрибутов)* – в виде овалов, *взаимосвязей* между сущностями – в виде ромбов.

В большинстве случаев прибегают к определению типов сущностей. Тип позволяет выделить из всего множества сущностей предметной области группу сущностей, однородных по структуре и поведению [7].

На рисунке 3.2 изображена сущность СТУДЕНТ и соответствующие ей атрибуты: Номер зачетной книжки, ФИО, Год рождения, Место рождения, Телефон.



Рис. 3.2. Пример типа сущности СТУДЕНТ с атрибутами

Атрибуты можно условно классифицировать следующим образом:

1. *Идентифицирующие и описательные атрибуты.* Идентифицирующие атрибуты имеют уникальное значение для сущностей данного типа, описательные атрибуты заключают в себе ин-

тересующие свойства сущности. Для каждой сущности должен быть указан первичный ключ.

2. *Составные и простые атрибуты.* Простой атрибут состоит из одного компонента, его значение неделимо; составной атрибут является комбинацией нескольких компонентов, возможно, принадлежащих разным типам данных.

3. *Однозначные и многозначные атрибуты* могут иметь одно или много значений для каждого экземпляра сущности соответственно.

4. *Основные и производные атрибуты.* Значение основного атрибута не зависит от других атрибутов. Значение производного атрибута вычисляется на основе значений других атрибутов.

5. *Обязательные и необязательные.* Значение обязательного атрибута всегда устанавливается при помещении данных в БД; значение необязательного атрибута может быть пропущено, т.е. ему может быть присвоено NULL-значение.

3.1.2. Типы сущностей и иерархия наследования

Сущность является *независимой*, если каждый ее экземпляр может быть однозначно идентифицирован без установления связей с другой сущностью.

В качестве иллюстрации можно привести сущности ОТДЕЛ и СОТРУДНИК, которые являются независимыми, если предположить, что сотрудник может работать в организации, не числясь ни в одном отделе (например, директор организации).

Сущность является *зависимой* (дочерней), если уникальная идентификация ее экземпляров возможна только путем установления связи с другой сущностью. Примером зависимой сущности может являться сущность ЗАКАЗ в паре сущностей КЛИЕНТ–ЗАКАЗ. Сущность КЛИЕНТ является независимой, т.к. в базе дан-

ных может храниться информация о клиентах, не сделавших заказ. Информация о заказе не имеет смысла без информации о клиенте, его разместившем.

Зависимая сущность на диаграмме помещается в прямоугольник с двойной рамкой (рис. 3.3).

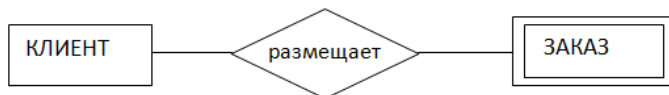


Рис. 3.3. Пример зависимой сущности

Различают несколько типов сущностей:

- стержневая;
- ассоциативная;
- именующая;
- характеристическая;
- категориальная.

Стержневая сущность является независимой. В любой БД должна присутствовать хотя бы одна стержневая сущность.

Ассоциативная сущность связана с несколькими родительскими сущностями. Такая сущность содержит информацию о связях сущностей. Они используются тогда, когда необходимо ликвидировать связь типа «многие ко многим».

Именующая сущность – это частный случай ассоциативной сущности, не имеющей собственных атрибутов (только атрибуты родительских сущностей, мигрировавших в качестве внешнего ключа).

Характеристическая сущность – это сущность, связанная только с одной родительской и хранящая информацию о характеристиках родительской сущности. В ней производится описание

или уточнение элемента другой сущности (обычно она связана со стержневой сущностью).

Категориальная сущность – это дочерняя сущность в иерархии наследования.

Иерархия наследования (или иерархия категорий) представляет собой особый тип объединения сущностей, которые разделяют общие характеристики.

Обычно иерархию наследования создают, либо когда несколько сущностей имеют общие по смыслу атрибуты, либо когда сущности имеют общие по смыслу связи (например, сущности ПОСТОЯННЫЙ СОТРУДНИК и СОВМЕСТИТЕЛЬ имеют сходную по смыслу связь «работает в» с сущностью ФИРМА), либо когда это диктуется бизнес-правилами.

Супертип содержит совместно используемые атрибуты, в то время как *подтип* содержит уникальные атрибуты.

Для каждой категории можно указать дискриминатор – атрибут родового предка, который показывает, как отличить одну категориальную сущность от другой. Супертип и его подтип(ы) поддерживают связь 1:1. Подтипы должны образовывать полное множество, т. е. любой экземпляр супертипа должен относиться к некоторому подтипу. Иногда для полноты множества надо определять дополнительный подтип, например, ПРОЧИЕ. Подтип наследует свойства и связи супертипа. Например, тип сущности ПРОГРАММИСТ является подтипом сущности СОТРУДНИК. Программисты обладают всеми свойствами сотрудников и участвуют во всех связях, однако обратные утверждения неверны.

Тип сущности, его подтипы, подтипы этих подтипов и т. д. образуют *иерархию наследования*.

На рис. 3.4 изображен пример иерархии категорий для описанного выше примера.

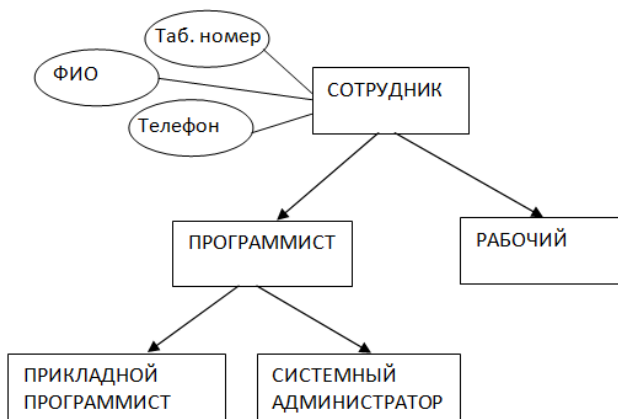


Рис. 3.4. Пример иерархии категорий

3.1.3. Описание связей

Между сущностями могут существовать *связи*, имеющие различный содержательный смысл (семантику).

Например:

- студент *учится* в группе;
- врач *лечит* пациента;
- летчик *управляет* самолетом.

Связи характеризуются тремя свойствами:

1. Мощность (1:1, 1:M, M:N);
2. Полнота: 1) полные; 2) неполные;
3. Степень: 1) унарные; 2) бинарные; 3) тернарные, ...,
n) n-арные.

Мощность (кардинальность) связи выражает определенное число экземпляров сущностей, связанных с одним экземпляром связанной сущности.

Мощность связи обозначается числами или буквами:

- 1 – один;

- N или M – многие;
- <число> – конкретное количество экземпляров.

Пример 3.1. Виды связей

1) Тип связи 1:1. Примером является связь между сущностями ГРАЖДАНИН РФ и ПАСПОРТ РФ, каждый гражданин Российской Федерации может иметь только один паспорт РФ (рисунок 3.5 а).

2) Тип связи 1:M. Примером является связь между сущностями ГРУППА и СТУДЕНТ (в одной группе обучается много студентов) (рисунок 3.5 б).

3) Тип связи M:N. Примером является связь между сущностями ПРЕПОДАВАТЕЛЬ и ДИСЦИПЛИНА. Каждый преподаватель может вести несколько дисциплин, и одну и ту же дисциплину могут вести несколько преподавателей (рис. 3.5 в).

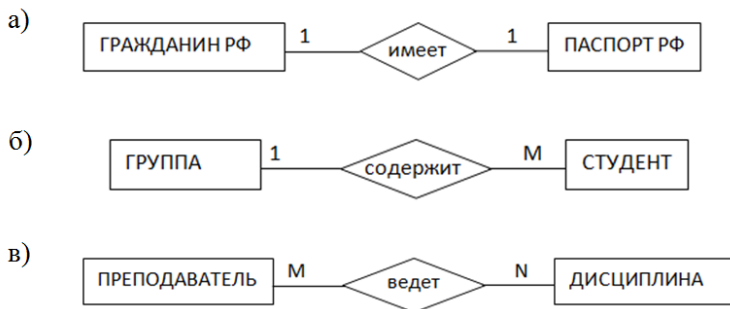


Рис. 3.5. Примеры различной мощности (кардинальности) связи:

а – мощность связи 1:1; б – мощность связи 1:M;

в – мощность связи M:N.

Полнота связи – возможность существования в сущности экземпляра, не связанного ни с одним экземпляром другой сущности.

Существуют градации полноты связи:

1. *Полная.* Ни в одной из связанных сущностей не может быть экземпляра, не связанного хотя бы с одним экземпляром другой сущности.

2. *Неполная.* В обеих сущностях могут существовать экземпляры, не связанные ни с одним экземпляром другой сущности.

Степень связи указывает на число ассоциированных сущностей или участников.

Унарная связь существует тогда, когда ассоциация поддерживается внутри единственной сущности.

Бинарная связь существует тогда, когда ассоциируются две сущности.

Тернарная связь имеет место тогда, когда связываются три сущности. Существуют и более высокие степени связи, они довольно редки и не имеют особых названий.

Рекурсивная связь имеет место, когда есть связь между экземплярами одного и того же набора сущностей (такое условие соблюдается в унарных связях).

На рис. 3.6 изображены три типа связей.



Рис. 3.6. Примеры типов связей

Метод «сущность–связь» основан на использовании диаграмм, называемых соответственно *диаграммами ER-экземпляров* и *диаграммами ER-типа*.

На рис. 3.7 приведена диаграмма ER-экземпляров для сущностей ПИЛОТ и САМОЛЕТ со связью «управляет».

ПИЛОТ	управляет	САМОЛЕТ
Сидоров	←	Ил-76
Данилов	→	Ту-134
Ефремов	←	А-320
Федоров	→	Boeing747
Никитин	←	МиГ-23Б
Ларионов	→	Ту-154
Карпов	←	Ан-26
Антонов	→	Су-17М

Рис. 3.7. Диаграмма ER-экземпляров

На рис. 3.8 изображена диаграмма ER-типа (ER-диаграмма), соответствующая диаграмме ER-экземпляров, изображенной на рис. 3.7.

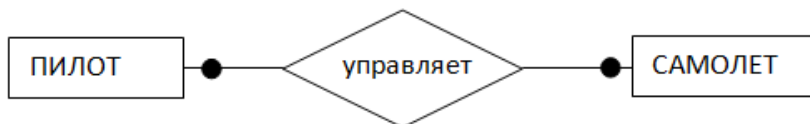


Рис. 3.8. Диаграмма ER-типа

На основе анализа диаграмм ER-типа формируются отношения проектируемой БД. При этом учитывается степень связи сущностей и класс их принадлежности, которые, в свою очередь, определяются на основе анализа диаграмм ER-экземпляров соответствующих сущностей.

Степень связи является характеристикой связи между сущностями (один из типов 1:1, 1:M, M:1, M:N).

Класс принадлежности (КП) характеризует участие сущности в рассматриваемой связи и может быть обязательным или необязательным.

Класс принадлежности сущности является обязательным, если все экземпляры этой сущности обязательно участвуют в рассматриваемой связи, в противном случае класс принадлежности сущности является необязательным.

В приведенной на рис. 3.7 диаграмме степень связи между сущностями 1:1, а класс принадлежности обеих сущностей необязательный. И действительно, из рисунка видно, что каждый пилот управляет только одним самолетом, а каждый самолет управляется только одним пилотом (связь 1:1). Еще из диаграммы ER-экземпляров видно, что есть пилоты, которые не управляют ни одним самолетом, и есть самолеты, не управляемые ни одним пилотом (необязательный КП обеих сущностей).

На рис. 3.9 изображена диаграмма, на которой сущности имеют обязательный и необязательный класс принадлежности. Каждый сотрудник в организации должен иметь какую-либо должность, т. е. нет сотрудника без должности. Это означает, что не существует объекта класса СЛУЖАЩИЙ, не связанного с каким-либо объектом класса ДОЛЖНОСТЬ. Но должность может быть и вакантной, т. е. быть не связанной ни с одним объектом класса СЛУЖАЩИЙ. Такая связь является обязательной по отношению к классу СЛУЖАЩИЙ и необязательной по отношению к классу ДОЛЖНОСТЬ.

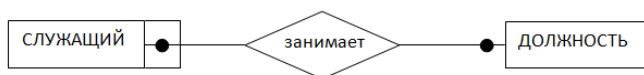


Рис. 3.9. Пример различных классов принадлежности

На диаграммах ER-типа *обязательное* участие в связи экземпляров сущности отмечается блоком с точкой внутри, смежным с

блоком этой сущности. При *необязательном* участии экземпляров сущности в связи дополнительный блок к блоку сущности не пристраивается, а точка размещается на линии связи. Символы на линии связи указывают на степень связи [1].

Пример 3.2. ER–диаграмма

На рис. 3.10 изображена ER–диаграмма в рамках нотации Питера Чена. Исходя из данной предметной области УЧЕБНЫЙ ПРОЦЕСС, выделены следующие сущности с соответствующими им атрибутами:

Сущность СТУДЕНТ с атрибутами: Номер студенческого билета, ФИО, Адрес проживания.

Сущность ПРЕПОДАВАТЕЛЬ с атрибутами: Табельный номер, ФИО, Должность, Кафедра, Телефон.

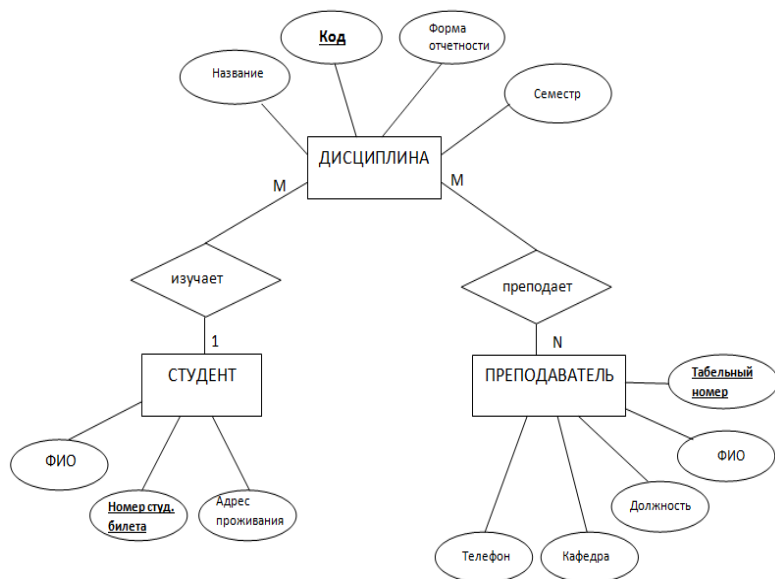


Рис. 3.10. Пример ER–диаграммы

Сущность ДИСЦИПЛИНА с атрибутами: Код дисциплины, Название, Форма отчетности, Семестр.

Ключевой атрибут на диаграмме выделяется подчеркиванием.

Между сущностями СТУДЕНТ и ДИСЦИПЛИНА имеется связь – «изучает». При этом подразумевается, что один студент может изучать несколько дисциплин (связь 1:М). Между сущностями ПРЕПОДАВАТЕЛЬ и ДИСЦИПЛИНА существует связь «преподают», причем преподаватель может вести несколько дисциплин и одну дисциплину могут преподавать несколько преподавателей (связь N:М).

3.1.4. Выбор ключа сущности

Под ключом подразумевается элемент данных, который позволяет уникально идентифицировать отдельные экземпляры некоторого типа сущности.

Тип сущности может иметь несколько потенциальных ключей. 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, несуществующий экземпляр стержневой сущности. По тем же причинам необходимо обеспечить уникальность *первичного ключа*.

При выборе первичного ключа следует отдавать предпочтение несоставным ключам или ключам, составленным из минимального числа атрибутов. Нецелесообразно 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 с одинаковыми фамилиями, именами и отчествами, при этом номер зачетной книжки является предпочтительным первичным ключом.

Потенциальный ключ сущности может оказаться большим по размеру (особенно, если он составной), и во всех сущностях, связанных с данной, потребуется создать поля такого же размера для хранения ссылок. В силу этих и других соображений в практике проектирования БД используют *суррогатный первичный ключ*, такой ключ не несет в себе информацию об объектах и служит для идентификации записей.

3.2. Логическое проектирование

Логическая структура БД должна соответствовать логической модели предметной области и учитывать связь модели данных с поддерживаемой СУБД. Поэтому этап начинается с выбора модели данных, где важно учесть её простоту и наглядность.

Предпочтительнее, когда естественная структура данных совпадает с представляющей её моделью. Так, например, если данные представлены в виде иерархической структуры, то и модель лучше выбирать иерархическую. Однако на практике такой выбор чаще определяется системой управления БД, а не моделью данных. Поэтому концептуальная модель фактически транслируется в такую модель данных, которая совместима с выбранной системой управления БД.

Для реляционной модели данных даталогическая модель – набор схем отношений, отражающих сущности и их связи.

Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам. Этот этап может быть в значительной степени автоматизирован.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Логическое проектирование заключается в определении числа и структуры таблиц, формировании запросов к БД, определении типов отчетных документов, разработке алгоритмов обработки информации, создании форм для ввода и редактирования данных в базе и решении ряда других задач.

Решение задач логического проектирования БД в основном определяется спецификой задач предметной области. Наиболее важной здесь является проблема структуризации данных, на ней мы сосредоточим основное внимание.

При проектировании структур данных для автоматизированных систем можно выделить три основных подхода:

1. Сбор информации об объектах решаемой задачи в рамках одной таблицы (отношения) и последующая декомпозиция ее на несколько взаимосвязанных таблиц на основе процедуры нормализации отношений.

2. Формулирование знаний о системе (определение типов исходных данных и их взаимосвязей) и требований к обработке данных, получение с помощью CASE–системы (системы автоматизации проектирования и разработки баз данных) готовой схемы БД или даже готовой прикладной информационной системы.

3. Структурирование информации для использования в информационной системе в процессе проведения системного анализа на основе совокупности правил и рекомендаций.

3.2.1. Построение логической модели в нотации IDEF1X

Логическая модель может быть создана с использованием CASE–средств, таких как ERWin, Design/IDEF и другие. Данные программные продукты используют для моделирования нотацию

IDEF1X, которая основана на подходе Питера Чена и является методологией построения реляционных структур, относящихся к типу методологий «сущность–связь». Информационная модель, построенная с помощью IDEF1X–методологии, отображает логическую структуру информации об объектах системы. Таким образом, концептуальная модель, представленная в соответствии со стандартом IDEF1X, является логической схемой базы данных для проектируемой системы.

В рамках стандарта IDEF1X сущность изображается в виде прямоугольника, разделенного внутри на две части. В верхней части записываются ключевые атрибуты, в нижней – неключевые. Название сущности записывается над прямоугольником (рис. 3.11).

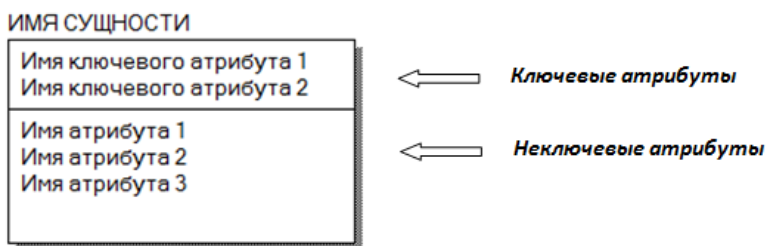




Рис. 3.11. Обозначение сущности в методологии IDEF1X

Данная методология описывает способы изображения двух типов сущностей – независимой и зависимой, и связей – идентифицирующих и неидентифицирующих. Связь изображается на ER–диаграмме линией, проводимой между сущностью–родителем и сущностью–потомком с точкой на конце линии у сущности–потомка. Идентифицирующая связь изображается сплошной линией, неидентифицирующая – пунктирной.

В таблице 3.1 приводятся обозначения элементов модели «сущность–связь» в рамках методологии IDEF1X.

Таблица 3.1. Обозначение элементов ER–диаграммы методологии IDEF1X

Элемент диаграммы	Обозначает
Обозначения сущностей	
	независимая сущность
	зависимая сущность
Обозначения связей	
	идентифицирующая связь
	неидентифицирующая связь
Обозначение кардинальности связей	
	1,1
	0,M
	0,1
	1,M
	точно N (N – произвольное число)

Каждая сущность может обладать любым количеством связей с другими сущностями. Связи дается имя, выражаемое грамматической формой глагола. Для связи дополнительно может присутствовать указание мощности: какое количество экземпляров сущности–потомка может существовать для сущности–родителя. Имя связи всегда формируется с точки зрения родителя. Сущность может обладать атрибутами, которые наследуются через связь с родительской сущностью. Последние обычно являются *внешними ключами* (FK на рис. 3.12) и служат для организации связей между сущностями.

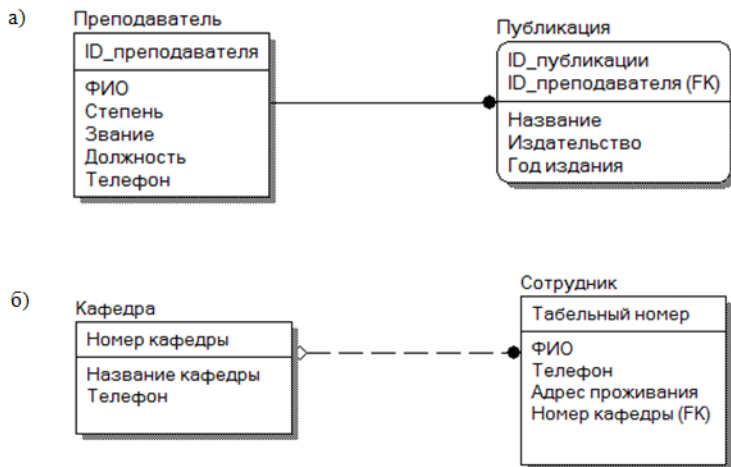


Рис. 3.12. Пример идентифицирующей (а) и неидентифицирующей (б) связи

Если внешний ключ сущности используется в качестве ее первичного ключа (РК) или как часть составного первичного ключа, то сущность является зависимой от родительской сущности. Если внешний ключ не является первичным и не входит в составной первичный ключ, то сущность является независимой от родительской сущности. Если сущность является зависимой, то связь ее с родительской сущностью называется идентифицирующей (рис. 3.12,а), в противном случае – неидентифицирующей (рис. 3.12,б).

При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности. В дочерней сущности новые атрибуты помечаются как внешний ключ – (FK). В примере на рисунке 3.12а ключевой атрибут *ID_преподавателя* родительской сущности ПРЕПОДАВАТЕЛЬ перешел в состав ключевого атрибута сущности ПУБЛИКАЦИЯ. При установлении

неидентифицирующей связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности. Неидентифицирующая связь служит для связывания независимых сущностей.

Имя связи – фраза, характеризующая отношение между родительской и дочерней сущностями. Для связи один–ко–многим идентифицирующей или неидентифицирующей достаточно указать имя, характеризующее отношение от родительской к дочерней сущности. Для связи многие–ко–многим следует указывать имена связи как от родительской к дочерней сущности, так и от дочерней к родительской.

Имя роли (функциональное имя) – это синоним атрибута внешнего ключа, который показывает, какую роль играет атрибут в дочерней сущности.

В примере на рис. 3.13 внешний ключ в сущности ПРОЕКТ имеет функциональное имя *Руководитель*, это имя показывает какую роль играет атрибут *ID_сотрудника* в данной сущности.

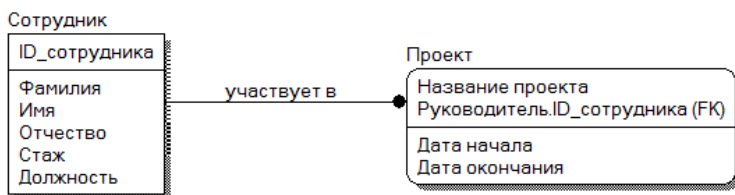


Рис. 3.13. Функциональное имя внешнего ключа

Обязательным является применение имен ролей в том случае, когда два или более атрибутов одной сущности определены на одной и той же области, то есть они имеют одинаковую область значений, но разный смысл (рис. 3.14).

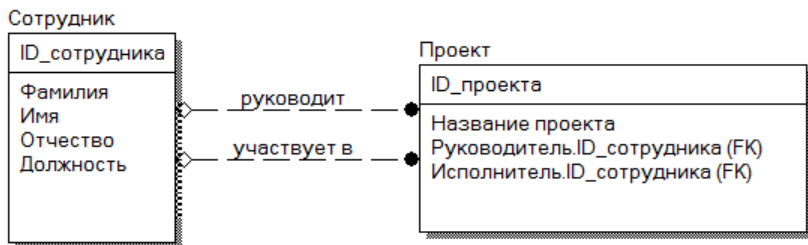


Рис. 3.14. Пример обязательного использования имен ролей

Кроме того, в IDEF1X вводится понятие «отношение категоризации». Некоторые объекты являются категориями других существующих объектов, поэтому сущности могут быть категориями других сущностей.

Отношение полной категоризации – отношение между двумя или более сущностями, в котором каждый экземпляр одной сущности, называемой *общей сущностью*, связан в точности с одним экземпляром одной и только одной из других сущностей, называемых *сущностями–категориями*.

Каждый экземпляр общей сущности и связанный с ним экземпляр одной из категориальных сущностей изображает один и тот же предмет реального мира и поэтому обладает одним и тем же уникальным идентификатором. Например, в базе данных университета требуется хранить информацию о сотрудниках, при этом должности работников образовательной организации высшего образования можно классифицировать таким образом, что они будут разделены на следующие категории:

- научно–педагогические (профессорско–преподавательский состав, научные работники);
- инженерно–технические;
- административно–хозяйственные.

Каждая из перечисленных категорий обладает набором уникальных атрибутов (рис. 3.15). Значение некоторого атрибута в экземпляре общей сущности определяет, с какими из возможных сущностей–категорий он связан. Этот атрибут называется *дискриминатором* отношения категоризации. В нашем примере дискриминатором может быть атрибут *Должность*.

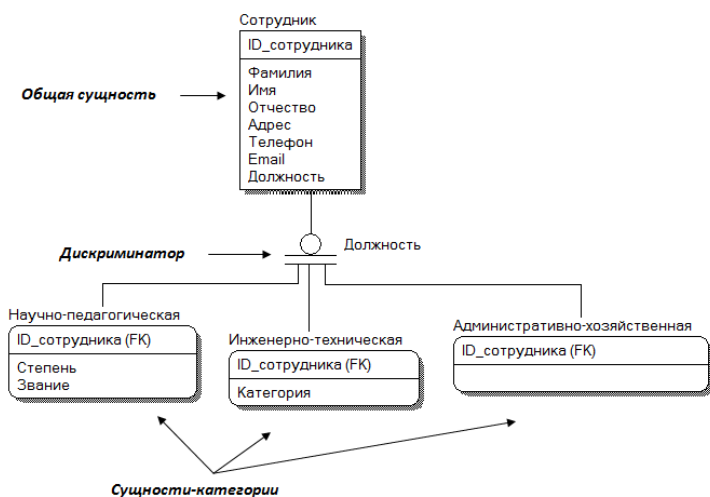


Рис. 3.15. Отношение полной категоризации в нотации IDEF1X

Сущности–категории, связанные с одной общей сущностью, всегда являются взаимоисключающими. То есть экземпляр общей сущности может соответствовать экземпляру только одной сущности–категории. Следовательно, сотрудник не может одновременно относиться и к научно–педагогическому, и к инженерно–техническому, и к административно–хозяйственному составу.

IDEF1X–синтаксис допускает, однако, существование неполного множества категорий. Если существует экземпляр общей сущности, не связанный ни с каким экземпляром из сущностей–

категорий, то такое отношение называется *отношением неполной категоризации*. На неполноту множества категорий на диаграмме указывает круг, подчеркнутый один раз (рис. 3.16).

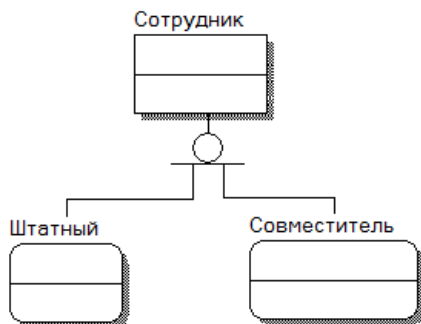


Рис. 3.16. Пример изображения неполной категории в нотации IDEF1X

3.2.2. Ограничения целостности

Этот компонент инфологической модели представляет собой совокупность правил, позволяющих обеспечить в любой момент времени правильность данных (непротиворечивость, удовлетворяемость, адекватность существующим знаниям о реальном мире). Эти правила устанавливаются исходя из семантики предметной области.

Обычно ограничения целостности формулируются на естественном языке и/или с использованием предикатов, хотя в некоторых современных системах автоматизированного проектирования для этого имеются специализированные языковые средства.

Ограничения могут быть *внутренними (неявными)* и *явными*.

Внутренние ограничения целостности предусмотрены самой моделью данных и тесно связаны со структурой данных. С ограничениями этого типа хорошо согласуются операции манипулиро-

вания данными, поэтому контроль за соблюдением внутренних ограничений обычно не вызывает трудностей. Как правило, внутренние ограничения целостности задаются вместе с определением структурных спецификаций средствами языка описания данных.

К внутренним ограничениям целостности в нотациях IDEF1X можно отнести:

- ограничения на значения ключевых атрибутов: уникальность значений атрибутов первичных и альтернативных ключей определяется понятием ключа. Эти ограничения задаются соответствующим выделением ключей в концептуальной схеме;
- ограничения на допустимые значения атрибутов: все допустимые значения атрибутов должны удовлетворять условию принадлежности соответствующему домену. Эти ограничения задаются соответствующим описанием доменов и атрибутов. Кроме того, в описании атрибутов указывается обязательность значений определенных атрибутов;
- ограничения на существующие значения (ссылочные ограничения): существование одних сущностей (дочерних, типа категория) ставится в зависимость от существования других (родительских, родовых). Эти ограничения представляются с помощью соответствующих связей между множествами сущностей.

Явные ограничения целостности задаются разработчиками. Для описания явных ограничений используются исчисление предикатов и утверждения на естественном языке. Обеспечение контроля явных ограничений целостности представляет собой серьезную проблему реализации информационной системы.

3.3. Физическое проектирование

Заключительным этапом проектирования базы данных является построение физической модели БД. Основная цель данного этапа – описание способа физической реализации логического проекта базы данных. Здесь детально расписывается схема данных с указанием всех типов, полей, размеров и ограничений. Для реляционной базы данных на этом этапе сущности преобразуются в таблицы, атрибуты – в столбцы. Связи между сущностями преобразуются в физическую схему БД согласно правилам преобразования ER–диаграмм (правила преобразования будут рассмотрены в главе 4).

На данном этапе решается задача грамотного размещения объектов базы данных в пространстве памяти, строятся индексы, определяется целесообразность использования хеширования и кластеризации [7]. На физическом этапе проектирования учитывается специфика выбранной СУБД.

Проектирование базы данных имеет итерационный характер. В процессе функционирования системы могут быть выявлены недостатки построенной модели БД, либо может возникнуть потребность во внесении изменений в модель. В этом случае потребуется модификация всего первоначального проекта БД.

3.4. Контрольные вопросы к главе 3

1. Назовите основные задачи, возникающие при проектировании баз данных.
2. Каковы основные этапы проектирования баз данных?
3. Что понимается под методами описания объектов предметной области и их характеристик?

4. Опишите типы сущностей и иерархию наследования сущностей.
5. Приведите примеры описания связей различных типов: по мощности, по полноте, по степени.
6. Опишите принципы выбора ключа отношения.
7. Назовите способы и средства построения логической модели предметной области.
8. Поясните понятия полной и неполной категоризации сущностей.
9. Что понимается под явными и неявными ограничениями целостности?
10. В чем заключается физическое проектирование базы данных?

Глава 4. ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ РЕЛЯЦИОННЫХ БД

4.1. Избыточное дублирование данных и аномалии

При одном и том же наборе атрибутов, описывающих предметную область, может быть предложено множество различных вариантов группировки этих атрибутов в отношения, то есть множество различных вариантов построения логической схемы БД. Удачность проекта в значительной степени зависит от того, насколько полно в нем были учтены естественные связи между данными, определяемые семантикой предметной области.

Охарактеризуем основные проблемы, имеющие место при определении структур данных в отношениях реляционной модели.

Существует два вида дублирования данных: простое (неизбыточное) и избыточное. Наличие первого из них допускается в базах данных, а избыточное дублирование данных может приводить к проблемам при обработке данных.

Пример 4.1. Избыточное дублирование данных

Пусть имеется отношение СТУДЕНТ (Фамилия, Группа). Для студентов, обучающихся в одной группе, номер группы совпадает (рис. 4.1). В то же время для каждого студента номер группы – уникальный атрибут, поэтому ни один из номеров групп не является избыточным.

Примером избыточного дублирования данных является отношение СТУДЕНТ (Фамилия, Группа, Куратор), которое в отличии

от предыдущего примера дополнено атрибутом Куратор (рис. 4.2). Естественно, что у студентов одной группы один и тот же куратор.

СТУДЕНТ

Фамилия	Группа
Иванов	601
Мурзабаева	601
Семенов	603
Быков	602
Денисова	601

Рис. 4.1. Пример избыточного дублирования данных

СТУДЕНТ

Фамилия	Группа	Куратор
Иванов	601	Чернов
Мурзабаева	601	Чернов
Семенов	603	Васильева
Быков	602	Мухина
Денисова	601	Чернов

Рис. 4.2. Пример избыточного дублирования данных

Выходом из данной ситуации является декомпозиция исходного отношения на два новых отношения СТУДЕНТ_ГРУППА(Фамилия, Группа) и ГРУППА_КУРАТОР (Группа, Куратор) (рис. 4.3).

Избыточное дублирование данных при обработке кортежей отношения создает проблемы, названные Э. Коддом *аномалиями обновления отношения*. Он показал, что для некоторых отношений проблемы возникают при попытке удаления, добавления или редактирования их кортежей.

СТУДЕНТ_ГРУППА		ГРУППА_КУРАТОР	
Фамилия	Группа	Группа	Куратор
Иванов	601	601	Чернов
Мурзабаева	601	603	Васильева
Семенов	603	602	Мухина
Быков	602		
Денисова	601		

Рис. 4.3. Отношения, полученные путем декомпозиции исходного отношения

Аномалией будем называть такую ситуацию в таблицах БД, которая приводит к противоречиям в БД, либо существенно усложняет обработку данных.

Выделяют три основных вида аномалий: аномалии модификации (или редактирования), аномалии удаления и аномалии добавления.

Аномалии модификации проявляются в том, что изменение значения одного данного может повлечь за собой просмотр всей таблицы и соответствующее изменение некоторых других записей таблицы.

Для отношения СТУДЕНТ (ФИО, Группа, Куратор), где в столбце Группа хранится полное название группы, а столбец Куратор содержит ФИО куратора группы, изменение значения Куратор (например, для устранения ошибки) может привести к существованию более одного куратора одной и той же группы.

Аномалии удаления состоят в том, что при удалении какого-либо данного из таблицы может пропасть и другая информация, которая не связана напрямую с удаляемым данным.

Для отношения СТУДЕНТ (ФИО, Группа, Куратор) удаление студента может привести к удалению из БД и ФИО куратора группы (в том случае, если для данной группы запись – единственная).

Аномалии добавления возникают в случаях, когда информацию в таблицу нельзя поместить до тех пор, пока она неполная, либо вставка новой записи требует дополнительного просмотра таблицы.

Для отношения СТУДЕНТ (ФИО, Группа, Куратор) добавление названия новой группы повлечет обязательное определение ФИО студента и куратора, в то время как эти данные могут быть пока не известны. В то же время, при добавлении нового студента значение поля Куратор в новой записи может не совпадать со значением данного поля для другого студента этой же группы.

Для рационального выбора логической схемы БД, минимизации проблем, связанных с избыточностью и аномалиями, необходимо учитывать зависимости, связывающие данные.

4.2. Правила преобразования ER–диаграммы в схему БД

Правила формирования отношений основываются на учете *степени связи* между сущностями (1:1, 1:M, M:1, M:N) и *классе принадлежности* экземпляров сущностей (обязательный, необязательный) [1].

1. ER–диаграмма двух сущностей, связанных между собой связью степени 1:1 с *обязательным классом* принадлежности с обеих сторон, трансформируется в *одно* реляционное отношение. Первичным ключом данного отношения может быть ключ любой сущности.

2. Связь 1:1 с *обязательным классом* принадлежности сущности с одной и *необязательным* с другой стороны в реляционной схеме реализуется следующим образом: под каждую сущность формируется по отношению с первичными ключами, являющимися ключами соответствующих сущностей. К отношению, сущность

которого имеет обязательный КП, добавляется в качестве атрибута ключ сущности с необязательным КП.

3. В случае, если степень связи 1:1 и класс принадлежности обеих сущностей является *необязательным*, то необходимо использовать три отношения. Два отношения соответствуют связываемым сущностям, ключи которых являются первичными в этих отношениях. Третье отношение является связным между первыми двумя, поэтому его ключ объединяет ключевые атрибуты связываемых отношений.

Пример 4.2. Применение правил 1-3 формирования отношений

Для того чтобы проиллюстрировать первые три правила, рассмотрим предметную область, в которой имеются сущности ПИЛОТ и САМОЛЕТ со связью «управляет» между ними.

Пусть сущность ПИЛОТ характеризуется атрибутами Код_П (код пилота), ФИО (фамилия, имя, отчество), ЧН (часы налета). Сущность САМОЛЕТ характеризуется атрибутами Код_С (код самолета), Название (название самолета). Класс принадлежности обеих сущностей – обязательный, т.е. все экземпляры сущностей обязательно участвуют в связи (рис. 4.4).

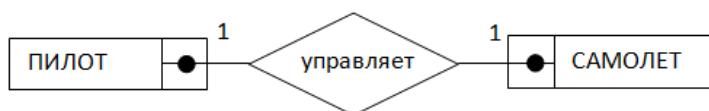


Рис. 4.4. ER-диаграмма для правила 1

Тогда схема отношения, полученного по правилу 1, будет выглядеть следующим образом:

ПИЛОТ_САМОЛЕТ (Код_П, ФИО, ЧН, Код_С, Название)

Сформированное отношение содержит полную информацию о пилотах и самолетах (рис. 4.5). Ключом итогового отношения вы-

бран атрибут Код_П – ключ отношения ПИЛОТ. Видно, что в этом отношении отсутствуют пустые поля (класс принадлежности обеих сущностей обязательный) и каждому пилоту соответствует ровно один самолет (связь 1:1). В данном случае одного отношения достаточно, а в качестве ключа может быть выбран либо код самолета, либо код пилота.

ПИЛОТ_САМОЛЕТ

Код_П	ФИО	ЧН	Код_С	Название
П1	Иванов	236	К1	ТУ154
П2	Климов	1044	К2	ТУ134
П3	Петров	870	К3	АН26
П4	Быков	500	К4	Ил76

Рис. 4.5. Отношение, полученное по правилу 1

Рассмотрим вариант, когда исходное отношение, содержащее информацию о пилотах и самолетах, выглядит следующим образом (рис. 4.6):

ПИЛОТ_САМОЛЕТ

Код_П	ФИО	ЧН	Код_С	Название
П1	Иванов	236	К1	ТУ154
П2	Климов	1044	К2	ТУ134
П3	Петров	870	К3	АН26
-	-	-	К4	Ил76

Рис. 4.6. Исходное отношение для правила 2

Из рисунка видно, что в отношении имеются пустые поля (обозначено «-»), т.е. имеются самолеты, которыми не управляет ни один пилот. Понятно, что такое отношение соответствует варианту связи 1:1 между сущностями ПИЛОТ и САМОЛЕТ и необязательному классу принадлежности со стороны сущности САМОЛЕТ.

Применим правило 2 и получим два отношения со следующими схемами:

ПИЛОТ (Код_П, ФИО, ЧН, Код_С)

САМОЛЕТ (Код_С, Название)

В результате применения данного правила удалось избежать пустых полей в отношениях (рис. 4.7).

ПИЛОТ				САМОЛЕТ	
Код_П	ФИО	ЧН	Код_С	Код_С	Название
П1	Иванов	236	К1	К1	ТУ154
П2	Климов	1044	К2	К2	ТУ134
П3	Петров	870	К3	К3	АН26
				К4	Ил76

Рис. 4.7. Отношения, полученные по правилу 2

Рассмотрим еще один вариант – когда связь между сущностями по-прежнему 1:1, а класс принадлежности обеих сущностей необязательный. При таком варианте исходная сущность будет выглядеть следующим образом (рис. 4.8):

ПИЛОТ_САМОЛЕТ				
Код_П	ФИО	ЧН	Код_С	Название
П1	Иванов	236	К1	ТУ154
П2	Климов	1044	-	-
П3	Петров	870	К3	АН26
-	-	-	К4	Ил76

Рис. 4.8. Исходное отношение для правила 3

Применив к данному отношению правило 3, получим набор из трех отношений (рис. 4.9). Первичными ключами данных отношений станут: Код_П – в отношении ПИЛОТ, Код_С – в отношении САМОЛЕТ, а ключом отношения ПИЛОТ_САМОЛЕТ станет составной ключ «Код_П, Код_С».

ПИЛОТ			САМОЛЕТ		ПИЛОТ_САМОЛЕТ	
Код_П	ФИО	ЧН	Код_С	Название	Код_П	Код_С
П1	Иванов	236	К1	ТУ154	П1	К1
П2	Климов	1044	К3	АН26	П3	К3
П3	Петров	870	К4	Ил76		

Рис. 4.9. Отношения, полученные по правилу 3

Определяющим фактором при формировании отношений для связи 1:М является класс принадлежности М-связной сущности. Класс принадлежности 1-связной сущности не влияет на результат.

4. Если связь типа 1:М и класс принадлежности сущности на стороне М является обязательным, то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ сущности на стороне 1 добавляется как атрибут в таблицу для сущности на стороне М.

5. Если связь типа 1:М и класс принадлежности сущности на стороне М является необязательным, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

Пример 4.3. Иллюстрация применения правила 5 для формирования отношений.

Связь между сущностями ПИЛОТ и САМОЛЕТ – 1:М, при этом класс принадлежности обеих сущностей необязательный (рис. 4.10).

У отношения, изображенного на рисунке 4.10 имеются следующие проблемы:

ПИЛОТ_САМОЛЕТ

Код_П	ФИО	ЧН	Код_С	Название
П1	Иванов	236	К1	ТУ154
П2	Климов	1044	К2	ТУ134
П2	Климов	1044	К3	АН26
-	-	-	К5	А320
П3	Петров	870	К3	АН26
П4	Быков	500	К4	Ил76
П4	Быков	500	К1	ТУ154
П5	Егоров	680	-	-

Рис. 4.10. Исходное отношение для правила 5

- дублирование данных: информация о пилотах повторяется столько раз, сколькими самолетами он управляет;
- пустые поля в кортежах: в тех случаях, когда пилот не управляет ни одним самолетом и есть самолеты, не управляемые ни одним пилотом.

Если бы класс принадлежности сущности ПИЛОТ был бы обязательным, то исчезла бы проблема, связанная с пустыми полями у пилотов, не управляющих ни одним самолетом. Однако это не решило бы все остальные проблемы.

Для устранения всех проблем, перечисленных выше, необходимо воспользоваться правилом 5 и в результате получить три отношения со следующими схемами:

ПИЛОТ (Код_П, ФИО, ЧН)

САМОЛЕТ (Код_С, Название)

ПИЛОТ_САМОЛЕТ (Код_П, Код_С)

При формировании отношений для связи N:M между сущностями необходимо будет построить три отношения, не зависимо от класса принадлежности сущностей.

6. Если связь типа M:N, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный

ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

4.3. Функциональные зависимости

4.3.1. Типы функциональных зависимостей

Функциональные зависимости (ФЗ) являются отражением семантики взаимосвязи данных в предметной области. С каждым построенным отношением в БД связывается определенная совокупность функциональных зависимостей, которые являются в ряде случаев источником аномалий данных.

Вспомним, что любое отношение рассматривается как переменная, принимающая определенные значения в определенные моменты времени.

Пусть R – переменная отношения, A, B – произвольные подмножества множества всех атрибутов R . $A \rightarrow B$, то есть B функционально зависит от A тогда и только тогда, когда для любого допустимого значения R каждое значение A связано только с одним значением B .

Это означает, что во всех кортежах с одинаковым значением атрибута A атрибут B будет иметь одно и то же значение. Отметим, что A и B могут быть составными – состоять из двух и более атрибутов. Говорится: « A функционально определяет B » или « B функционально зависит от A ».

Левая часть выражения называется *детерминантом* (*детерминантой*) функциональной зависимости (ФЗ), правая – *зависимой частью* ФЗ.

Наличие функциональной зависимости в отношении определяется природой вещей, информация о которых представлена кортежами отношения.

Пример 4.4. Примеры функциональных зависимостей

Пусть в отношении СТУДЕНТ (Номер зачетной книжки, Фамилия, Имя, Отчество, Адрес, Код группы) существуют такие ФЗ:

Номер зачетной книжки \rightarrow Фамилия, Имя, Отчество

Номер зачетной книжки \rightarrow Адрес, Код группы

Номер зачетной книжки, Фамилия, Имя, Отчество \rightarrow Адрес, Код группы

Это лишь некоторые ФЗ, из которых можно сделать вывод, что если детерминант содержит первичный ключ, то множество всех остальных атрибутов отношения функционально зависит от него. Аналогичный вывод можно сделать и для всех альтернативных ключей.

Множество атрибутов отношения, которое содержит в качестве подмножества потенциальный ключ, называется *суперключом* этого отношения.

Пример 4.5. Примеры функциональных взаимозависимостей

Если в то же отношение СТУДЕНТ добавить атрибут Куратор группы, то появятся такие ФЗ:

Код группы \rightarrow Куратор группы

Куратор группы \rightarrow Код группы

(причем, ни атрибут Код группы, ни атрибут Куратор группы не являются потенциальными ключами).

Если существует функциональная зависимость вида $A \rightarrow B$ и $B \rightarrow A$, то между A и B имеется взаимно однозначное соответствие, или *функциональная взаимозависимость*. Наличие функциональной взаимозависимости между атрибутами A и B обозначим как $A \leftrightarrow B$ или $B \leftrightarrow A$.

Существуют такие ФЗ, которые учитываются только формально, т.к. они всегда существуют и подразумеваются самим определением ФЗ. Это *тривиальные ФЗ*.

Тривиальная функциональная зависимость – это такая ФЗ, зависящая часть которой является подмножеством детерминанта.

Пример 4.6. Пример тривиальной функциональной зависимости

Номер зачетной книжки, Фамилия, Имя, Отчество \rightarrow Фамилия, Имя, Отчество

Код группы, Курс \rightarrow Курс

Наличие между атрибутами такого рода зависимостей никакой дополнительной смысловой нагрузки не несет и при нормализации не рассматривается, но все же такие зависимости существуют и всегда формально учитываются.

Множество всех функциональных зависимостей, которые задаются конкретным множеством функциональных зависимостей S , т. е. могут быть выведены из этих зависимостей, называется *замыканием* множества зависимостей S и обозначается S^+ .

Частичная функциональная зависимость – это зависимость неключевого атрибута от части составного ключа.

Пример 4.7. Пример частичной функциональной зависимости

Для отношения ПРЕПОДАВАТЕЛЬ–ДИСЦИПЛИНА (Фамилия преподавателя, Должность, Дисциплина, Часы) первичным ключом является составной ключ – {Фамилия преподавателя, Дисциплина}. Значение атрибута Часы зависит от атрибута Дисциплина, т.е. имеется частичная функциональная зависимость: Дисциплина \rightarrow Часы.

Атрибут C зависит от атрибута A *транзитивно* (существует транзитивная зависимость), если для атрибутов A, B, C выполняются условия $A \rightarrow B$ и $B \rightarrow C$, но обратная зависимость отсутствует.

Пример 4.8. Пример транзитивной функциональной зависимости

Пусть существует отношение СОТРУДНИК (ФИО, Должность, Оклад). В рамках одной организации для каждой должности установлен определенный оклад, поэтому можно сказать, что существует зависимость вида Должность \rightarrow Оклад. Также можно сказать, что существует зависимость ФИО \rightarrow Должность. В отношении СОТРУДНИК транзитивной зависимостью связаны атрибуты:

ФИО \rightarrow Должность \rightarrow Оклад

4.3.2. Аксиомы вывода функциональных зависимостей

Пусть имеется отношение r со схемой отношения R ; $X, Y, Z, W \subseteq R$.

1. Рефлексивность: $X \rightarrow X$.
2. Пополнение: если r удовлетворяет функциональной зависимости $X \rightarrow Y$, то оно удовлетворяет и функциональной зависимости $XZ \rightarrow Y$.
3. Аддитивность: если в отношении r заданы функциональные зависимости $X \rightarrow Y$ и $X \rightarrow Z$, то существует функциональная зависимость $X \rightarrow YZ$.
4. Проективность: если в отношении r задана функциональная зависимость $X \rightarrow YZ$, то существует и функциональная зависимость $X \rightarrow Y$.
5. Транзитивность: если в отношении r заданы функциональные зависимости $X \rightarrow Y$ и $Y \rightarrow Z$, то существует и функциональная зависимость $X \rightarrow Z$.
6. Псевдотранзитивность: если в отношении r заданы функциональные зависимости $X \rightarrow Y$ и $YZ \rightarrow W$, то существует и функциональная зависимость $XZ \rightarrow W$.

Данная система аксиом является полной и избыточной.

В отношении R атрибут B *многозначно зависит* от атрибута A , если каждому значению A соответствует множество значений B , не связанных с другими атрибутами из R .

Многозначные зависимости (МЗ) могут быть «один ко многим» (1:М), «многие к одному» (М:1) или «многие ко многим» (М:Н) и обозначаются соответственно: $A \twoheadrightarrow B$, $A \ll B$ и $A \ll B$.

Например, пусть преподаватель ведет несколько предметов, а каждый предмет может вестись несколькими преподавателями, тогда имеет место зависимость ФИО \ll Предмет.

4.3.3. Аксиомы вывода многозначных зависимостей

Пусть имеется отношение r со схемой отношения R ; $X, Y, Z, W \subseteq R$.

1. Рефлексивность: $X \twoheadrightarrow X$.
2. Пополнение: если r удовлетворяет многозначной зависимости $X \twoheadrightarrow Y$, то оно удовлетворяет и многозначной зависимости $XZ \twoheadrightarrow Y$.
3. Аддитивность: если в отношении r заданы многозначные зависимости $X \twoheadrightarrow Y$ и $X \twoheadrightarrow Z$, то существует и многозначная зависимость $X \twoheadrightarrow YZ$.
4. Проективность: если в отношении r задана многозначная зависимость $(X \twoheadrightarrow Y) \& (X \twoheadrightarrow Z)$, то существует и многозначная зависимость $X \twoheadrightarrow (Y \cap Z)$ или $X \twoheadrightarrow (Y - Z)$.
5. Транзитивность: если в отношении r заданы многозначные зависимости $X \twoheadrightarrow Y$ и $Y \twoheadrightarrow Z$, то существует и многозначная зависимость $X \twoheadrightarrow Z - Y$.
6. Псевдотранзитивность: если в отношении r заданы многозначные зависимости $X \twoheadrightarrow Y$ и $YW \twoheadrightarrow Z$, то существует и многозначная зависимость $XW \twoheadrightarrow Z - (YW)$.

7. Дополнение: если в отношении r заданы многозначные зависимости $X \twoheadrightarrow Y$ и $Z = R - XY$, то существует и многозначная зависимость $X \twoheadrightarrow Z$.

Замечание. В общем случае между двумя атрибутами одного отношения могут существовать зависимости: 1:1, 1:M, M:1 и M:N. Поскольку зависимость между атрибутами является причиной аномалий, стараются расчленивать отношения с зависимостями атрибутов на несколько отношений. В результате образуется совокупность связанных отношений (таблиц) со связями вида 1:1, 1:M, M:1 и M:N. Связи между таблицами отражают зависимости между атрибутами различных отношений.

Два или более атрибута называются *взаимно независимыми*, если ни один из этих атрибутов не является функционально зависимым от других атрибутов.

В случае двух атрибутов отсутствие зависимости атрибута A от атрибута B можно обозначить так: $A \not\rightarrow B$. Случай, когда $A \not\rightarrow B$ и $B \not\rightarrow A$, можно обозначить $A \not\rightarrow B$.

Зависимости между атрибутами отношения можно установить с помощью диаграммы зависимостей [11].

Пример 4.9. Примеры диаграмм зависимостей

Допустим в некоторой компании необходимо реализовать базу данных учета работы над проектами. При этом каждый проект имеет свой уникальный номер и название, над каждым проектом могут работать один или несколько сотрудников, один и тот же сотрудник может принимать участие в нескольких проектах. Стоимость оплаты часа работы сотрудника зависит от занимаемой должности. Руководителю фирмы необходимо знать, какую сумму по каждому проекту в итоге получает каждый сотрудник. Для данного примера диаграмма зависимостей будет выглядеть следующим образом (рис. 4.11):

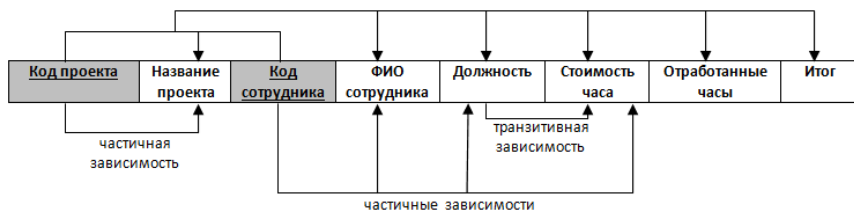


Рис. 4.11. Пример диаграммы зависимостей

Авторы из источника [12] предлагают следующее обозначение:

- атрибуты первичного ключа выделены жирным шрифтом и подчеркнуты, а также закрашены другим оттенком;
- линии со стрелками над сущностями указывают все возможные желательные зависимости, т. е. зависимости, основанные на первичном ключе;
- линии со стрелками в нижней части диаграммы зависимостей указывают на необязательные зависимости (частичная, транзитивная зависимость).

Возможны и другие варианты диаграмм зависимостей (рис. 4.12), но как бы ни выглядела такая диаграмма, она облегчает процесс анализа отношения и выделения всевозможных зависимостей между атрибутами, а в последствии и процесс нормализации.

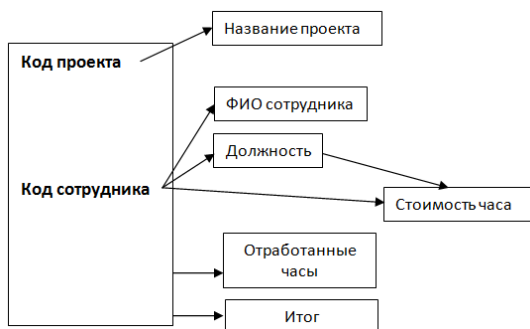


Рис. 4.12. Пример альтернативной диаграммы зависимостей

4.4. Нормальные формы схем отношений

Для того чтобы избежать информационной избыточности БД, а также трудностей сохранения целостности БД при выполнении операций включения, удаления и модификации, необходимо логическую схему БД выражать в виде совокупности схем отношений в так называемой *нормальной форме (НФ)*.

Каждая следующая нормальная форма ограничивает определенный тип функциональных зависимостей, устраняет соответствующие аномалии при выполнении операций над отношениями БД и сохраняет свойства предшествующих нормальных форм.

Выделяют следующую последовательность нормальных форм:

- первая нормальная форма (1НФ);
- вторая нормальная форма (2НФ);
- третья нормальная форма (3НФ);
- усиленная третья нормальная форма, или нормальная форма Бойса –Кодда (БКНФ);
- четвертая нормальная форма (4НФ);
- пятая нормальная форма (5НФ);
- шестая нормальная форма (6НФ).

В основе процесса проектирования лежит метод нормализации – декомпозиция отношения, находящегося в предыдущей НФ, в два и более отношения, удовлетворяющие требованиям следующей НФ.

4.4.1. Первая нормальная форма (1НФ)

Отношение находится в 1НФ, если значение, определяемое доменом каждого атрибута, является атомарным, то есть значения не являются ни списками, ни множествами простых или сложных

значений. Исходное отношение строится таким образом, чтобы оно было в 1НФ.

Пример 4.10. Приведение отношения к 1НФ

Допустим имеется отношение ЭКСПЕРИМЕНТ, содержащее информацию о шифре эксперимента и дате его проведения (рис. 4.13).

ЭКСПЕРИМЕНТ

Шифр	Дата
P2B	7 июня, 9 июня, 21 августа
P2Г	21 марта, 5 апреля
P2H	26 июня

Рис. 4.13. Отношение ЭКСПЕРИМЕНТ

Отношение ЭКСПЕРИМЕНТ содержит сложный атрибут Дата (данный атрибут состоит из множества значений). Для приведения к 1НФ требуется сделать данный атрибут атомарным. Исходное отношение, приведенное к 1НФ, выглядит следующим образом (рис. 4.14):

ЭКСПЕРИМЕНТ

Шифр	Дата
P2B	7 июня
P2B	9 июня
P2B	21 августа
P2Г	21 марта
P2Г	5 апреля
P2H	26 июня

Рис. 4.14. Отношение ЭКСПЕРИМЕНТ, приведенное к 1НФ

Проблема таблицы, приведенной к первой нормальной форме, состоит в том, что в ней имеются частичные зависимости – т. е. зависимости, основанные только на части первичного ключа.

Эта частичная зависимость от ключа приводит к следующему:

1. В отношении присутствует явное и неявное избыточное дублирование данных.

2. Следствием избыточного дублирования данных является проблема их редактирования.

Часть избыточности устраняется при переводе отношения в 2НФ.

4.4.2. Вторая нормальная форма (2НФ)

Отношение находится во 2НФ, если оно находится в 1НФ и каждый неключевой атрибут функционально полно зависит от первичного ключа (составного).

Пример 4.11. Приведение отношения ко 2НФ

Ключом отношения ЭКСПЕРИМЕНТ (рис. 4.15) является комбинация полей (Шифр, Дата). Этому отношению назначена функциональная зависимость Шифр→Номер стенда, это значит, что каждый эксперимент проводится только на одном стенде.

ЭКСПЕРИМЕНТ			
Шифр	Дата	Испытатель	Номер стенда
P21	6 июня	Иванов	1
P21	7 июня	Петров	1
P814	9 июня	Иванов	3

Рис. 4.15. Отношение ЭКСПЕРИМЕНТ (приведение ко 2НФ)

Допустим, мы хотим модифицировать кортеж: (P21, 6 июня; Испытатель=Иванов; Номер стенда=2). Выполнение этой модификации приводит к нарушению объявленной ФЗ Шифр→Номер стенда. Чтобы ее не нарушить, необходимо произвести замену для

данного шифра номера стенда на новый, хотя изначально предполагалось скорректировать только 1 кортеж. Эта проблема возникла из-за того, что схема отношений не находится во 2НФ.

Рассмотрим более внимательно исходное отношение и выделим имеющиеся функциональные зависимости. О данной предметной области известно, что в один день эксперимент может производиться только одним испытателем. Также известно, что один и тот же испытатель может проводить эксперименты в различные сроки с различными шифрами.

Перечислим функциональные зависимости:

Шифр → Номер стенда

Шифр, Дата → Испытатель

Шифр, Дата → Испытатель, Номер стенда

Атрибут Испытатель зависит от составного ключа (Шифр, Дата) полностью, а атрибут Номер стенда только частично зависит от ключа. Таким образом можно сделать вывод, что данное отношение не находится во 2-ой НФ, так как имеется частичная зависимость между непервичным атрибутом и составным ключом.

Для устранения частичной зависимости и перевода отношения во 2НФ необходимо, используя операцию проекции, разложить его на несколько отношений следующим образом:

1. Построить проекцию без атрибутов, находящихся в частичной функциональной зависимости от первичного ключа.

2. Построить проекции на части составного первичного ключа и атрибуты, зависящие от этих частей.

В итоге получим два отношения R1 и R2 (рис. 4.16):

R1			R2	
Шифр	Дата	Испытатель	Шифр	№ стенда
P21	6 июня	Иванов	P21	1
P21	7 июня	Петров	P814	3
P814	9 июня	Иванов		

Рис. 4.16. Отношения R1 и R2, полученные в результате приведения отношения ЭКСПЕРИМЕНТ во 2НФ

4.4.3. Третья нормальная форма (3НФ)

Отношение находится в 3НФ, если оно находится в 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Пример 4.12. Приведение отношения к 3НФ

Ключом отношения ЭКСПЕРИМЕНТ, приведенного на рис. 4.17, является составной атрибут (Шифр, Дата). Следует отметить, что у каждого испытателя есть свой личный код.

ЭКСПЕРИМЕНТ

Шифр	Дата	Код	Испытатель
P21	6 июня	311	Иванов
P21	7 июня	300	Петров
P814	9 июня	311	Иванов

Рис. 4.17. Отношение ЭКСПЕРИМЕНТ (приведение к 3НФ)

Выделим функциональные зависимости отношения ЭКСПЕРИМЕНТ:

Шифр, Дата \rightarrow Код, Испытатель

Код \rightarrow Испытатель

Испытатель \rightarrow Код

Атрибут Испытатель транзитивно зависит от (Шифр, Дата). Аналогично и атрибут Код транзитивно зависит от ключа.

Транзитивные зависимости также порождают избыточное дублирование информации в отношении.

Устраним их, используя операцию проекции на атрибуты. Получим два отношения R1 и R2, каждое из которых находится в 3НФ (рис. 4.18):

R1			R2	
Шифр	Дата	Код	Код	Испытатель
P21	6 июня	311	311	Иванов
P21	7 июня	300	300	Петров
П814	9 июня	311		

Рис. 4.18. Отношения R1 и R2, полученные в результате приведения отношения ЭКСПЕРИМЕНТ к ЗНФ

На практике построение ЗНФ схем отношений в большинстве случаев является достаточным, и приведением к ним процесс проектирования реляционной БД заканчивается.

Если в отношении имеется зависимость атрибутов составного ключа от неключевых атрибутов, то необходимо перейти к усиленной ЗНФ.

4.4.4. Нормальная форма Бойса–Кодда (НФБК)

Детерминант – это любой атрибут, от которого полностью функционально зависит некоторый другой атрибут. Другими словами, детерминантом функциональной зависимости называется ее левая (определяющая) часть.

Отношение находится в БКНФ, если каждый детерминант является потенциальным ключом.

Ситуация, когда отношение будет находится в ЗНФ, но не в БКНФ, возникает при условии, что отношение имеет два (или более) возможных ключа, которые являются составными и имеют общий атрибут.

Пример 4.13. Приведение отношения к НФБК

Имеется расписание производимых экспериментов (рисунок 4.19). Каждый эксперимент можно проводить один раз в день. Для

проведения эксперимента или экспериментов каждому испытателю в определенный день предоставляется определенный стенд. В течении дня данный стенд может использоваться разными испытателями.

ЭКСПЕРИМЕНТ

Шифр	Дата	Время	Испытатель	Стенд
P21	6 июня	10.00	Иванов	1
P22	6 июня	12.00	Иванов	1
П814	6 июня	12.00	Петров	2
P22	10 июня	10.00	Иванов	2

Рис. 4.19. Отношение ЭКСПЕРИМЕНТ (приведение к НФБК)

В качестве первичного ключа выберем (Шифр, Дата).

Потенциальные ключи:

(Шифр, Дата)

(Дата, Время, Испытатель)

(Дата, Время, Стенд)

Все атрибуты отношения зависят от всего ключа, нет частичных зависимостей, нет транзитивных зависимостей. Однако существует следующая зависимость: Испытатель, Дата → Стенд.

Нарушается определение нормальной формы Бойса–Кодда – данный составной атрибут является детерминантом, но не является потенциальным ключом. Данная ФЗ не нарушает ЗНФ, но порождает аномалии обновления. Выход – разбиение отношения на два:

R1 (Шифр, Дата, Время, Испытатель)

R2 (Испытатель, Дата, Стенд)

Заметим, что на практике такие ситуации встречаются достаточно редко, для всех прочих отношений ЗНФ и БКНФ эквивалентны.

4.4.5. Четвертая нормальная форма (4НФ)

Для отношений, в которых имеют место только функциональные зависимости между атрибутами, приведение их к нормальной форме Бойса–Кодда заканчивает процесс их нормализации. Однако в отношениях кроме *функциональных* могут присутствовать и другие, более сложные, виды зависимостей между атрибутами. Так, введение *четвертой* нормальной формы связано с необходимостью решения проблем, обусловленных наличием в отношении так называемой *многозначной* зависимости.

Пример 4.14. Приведение отношения к 4НФ

Допустим имеется следующее отношение, содержащее информацию о некоторой системе. В отношении содержится список подсистем, для каждой подсистемы – список подпрограмм, а также список параметров, вычисляемых в данной подсистеме. Подпрограммы могут быть использованы в разных подсистемах, и разные подпрограммы могут вычислять одинаковые параметры. Предполагается, что подпрограмма, используемая в подсистеме, вычисляет все параметры, относящиеся к данной подсистеме.

Пусть исходная информация в таком отношении выглядит следующим образом (рис. 4.20):

Недостатком данного отношения является то, что при добавлении/удалении подпрограммы приходится добавлять/удалять столько кортежей, сколько имеется параметров в подсистеме. Причиной данной аномалии является наличие некоторой зависимости между атрибутами отношений, а именно – многозначной зависимости.

Действительно, в исходном отношении существуют следующие многозначные зависимости:

Подсистема →→ Подпрограмма

Подсистема →→ Параметр

Подсистема	Подпрограмма	Параметр
A	SUB 1	P1
A	SUB 1	P2
A	SUB 1	P3
B	SUB 2	P1
B	SUB 2	P2
B	SUB 3	P1
B	SUB 3	P2
B	SUB 1	P1
B	SUB 1	P2
C	SUB 4	P1

Рис. 4.20. Исходное отношение для приведения его к 4НФ

В произвольном отношении $R(A,B,C)$ может одновременно существовать многозначная зависимость $A \twoheadrightarrow B$ и $A \twoheadrightarrow C$. Данную ситуацию обозначим как $A \twoheadrightarrow B/C$.

Дальнейшая нормализация отношения основана на следующей теореме.

Теорема Фейгина (Fagin R.). Отношение можно спроецировать без потерь в отношении $R1(A,B)$ и $R2(A,C)$ в том и только том случае, когда существует зависимость $A \twoheadrightarrow B/C$.

Под проецированием без потерь здесь понимается такой способ декомпозиции отношения, при котором исходное отношение полностью и без избыточности восстанавливается путем естественного соединения полученных отношений [1].

Отношение R находится в четвертой нормальной форме (4НФ) в том и только в том случае, когда существует многозначная зависимость $A \twoheadrightarrow B$, а все остальные атрибуты R функционально зависят от A .

Для преобразования отношения $R\{A, B, C\}$, находящегося в нормальной форме Бойса–Кодда и имеющего многозначные зави-

симости $A \twoheadrightarrow B|C$, в четвертую нормальную форму его следует разбивать на две проекции $\{A, B\}$ и $\{A, C\}$.

В результате разбиения исходного отношения, получим следующие два отношения (рис. 4.21):

Подсистема	Подпрограмма
A	SUB 1
B	SUB 2
B	SUB 3
B	SUB 1
C	SUB 4

Подсистема	Параметр
A	P1
A	P2
A	P3
B	P1
B	P2
C	P1

Рис. 4.21. Отношения R1 и R2, полученные в результате приведения исходного отношения к 4НФ

Первичные ключи данных отношений:

R1 – (Подсистема, Подпрограмма)

R2 – (Подсистема, Параметр)

Оба этих отношения находятся в 4НФ и свободны от замеченных недостатков.

Нормализация сокращает дублирование данных, но появление новых отношений усложняет схему базы данных.

Существует ряд дополнительных нормальных форм. Однако такие нормальные формы, как 5НФ и 6НФ (нормальная форма доменного ключа), вряд ли встретятся в реальных предметных областях. Они представляют, по большей части, теоретический интерес. Описание пятой нормальной формы можно получить в источниках [1, 11].

4.4.6. Денормализация

До сих пор речь шла только о недостатках ненормализованных (или в недостаточной мере нормализованных) баз данных.

Создание нормализованных отношений является важнейшей частью проектирования БД. В хорошем проекте БД должны учитываться различные требования к обработке информации. По мере того как таблицы преобразуются в соответствии с требованиями нормализации, их число растет. Большее число таблиц влечет за собой увеличение количества операций обмена данными с диском (операции ввода/вывода) и увеличение времени на обработку логических связей, что приводит к снижению скорости всей системы в целом. Следовательно, возможны обстоятельства, при которых потребуется некоторое снижение уровня нормализации, что позволит увеличить скорость обработки данных. Кроме того, некоторые аномалии могут иметь чисто теоретическое значение. В силу этих обстоятельств на практике обычно стараются найти разумный компромисс и редко доводят нормализацию до 5НФ. Практика показала, что большинство динамично обновляющихся БД обычно имеют 3НФ. Во многих случаях полезно избавление от многозначных зависимостей (4НФ).

Если количество запросов на выборку существенно превышает количество запросов на обновление, имеет смысл проанализировать возможности сознательной *денормализации* базы данных и поступиться даже требованиями 3НФ.

Денормализация – это процесс модификации структуры таблиц нормализованной базы данных с целью повышения производительности за счет допущения некоторой управляемой избыточности данных. Единственным оправданием денормализации является попытка повышения скорости работы приложений, работающих с базой данных.

Нормальные формы низких уровней имеют место (а иногда они просто необходимы) в специализированных БД, называемых *информационными хранилищами*. В информационных хранилищах, как правило, используются структуры 2НФ в условиях слож-

ной, многоуровневой информационной среды, питающейся от множества источников.

Однако за денормализацию нужно платить. В денормализованной базе данных повышается избыточность данных, что может повысить производительность, но потребует больше усилий для контроля ссылочной целостности. Усложнится процесс создания приложений, поскольку данные будут повторяться и их труднее будет отслеживать. Существует золотая середина между нормализацией и денормализацией, но, чтобы найти ее, требуются немалые усилия и хорошее знание предметной области.

Денормализация бывает нескольких видов [7]:

- *нисходящая денормализация* предлагает перенос атрибута из одной (родительской) сущности в подчиненную (дочернюю) сущность.
- *восходящая денормализация* предлагает перенос атрибута из подчиненной (дочерней) сущности в родительскую сущность, обычно в форме итоговых данных.
- *внутритабличная денормализация* выполняется в пределах одной таблицы, т.е. это процесс введения избыточных колонок в одной таблице с целью увеличения производительности запроса строки по производному значению.

4.5. Пример нормализации до 3НФ

Пусть требуется спроектировать базу данных заказов магазина–склада цифровой техники и товаров для офиса. Клиентами данного магазина являются фирмы, офисы, организации. Сам процесс сделки выглядит следующим образом: клиент звонит или приходит в магазин, беседует с сотрудником, который и принимает заказ.

Чтобы проиллюстрировать вышесказанное, покажем каким образом должна выглядеть таблица заказов (рис. 4.22):

ЗАКАЗЫ

№ заказа	Дата заказа	Код клиента	Имя клиента	Адрес клиента	Позиции заказа
100	05/05/17	545	ООО«Флаг»	Ленинская, 45	2199, Ноутбук ACER Aspire, 25500р - 2шт, итог 51000р; 1115, Проектор Optoma, 43200 - 1 шт, итог 43200р.; 1123, Проектор Epson, 21000 - 2 шт, итог 42000р.; 3312, МФУ HP, 25100р. - 3 шт., итог 75300р.
101	05/05/17	120	АО «Сервис»	Солнечная, 23	2224, Ноутбук Dell Inspiron, 18200р -1шт, итог 18200р 9978, Офис. кресло Rec, 5300р. - 2шт., итог 10600р.;
102	20/06/17	777	АО «Гарант»	Московское шоссе, 71	9978, Офис. кресло Rec, 5300р. - 5шт., итог 26500р.; 5556, Принтер Epson, 6700. - 2шт., итог 13400р.
103	22/06/17	545	ООО «Флаг»	Ленинская, 45	1166, Проектор INFOCUS, 16200р. - 3шт., итог 48600р.; 3344, МФУ Epson, 13000р. - 2 шт., итог 26000р.; 1123, Проектор Epson, 21000 - 1 шт, итог 21000р.; 6611, 3D ручка UNID, 2500р. - 3 шт., итог 8700р.

Рис. 4.22. Исходное отношение ЗАКАЗЫ

Столбцы данной таблицы содержат следующую информацию:

- номер и дату заказа;
- код, имя и адрес клиента;
- информацию о заказе, т.е. код и наименование товара, его стоимость, количество единиц товара и итоговую стоимость по каждой позиции.

Исходное отношение ЗАКАЗЫ содержит сложный атрибут Позиции заказа. Для приведения к 1НФ требуется сделать все атрибуты простыми (рис. 4.23) и ввести составной ключ отношения (№ заказа, Код товара).

В этой таблице содержится значительное количество повторяющейся информации. Например, сведения о каждом клиенте повторяются для каждого сделанного им заказа. Такая структура таблицы приведет к трате значительного времени на ввод повторяющихся данных. Наличие повторяющейся информации приведет к неограниченному увеличению объема базы данных, возрастает вероятность ошибок при вводе.

ЗАКАЗЫ

<u>№ заказа</u>	<u>Дата заказа</u>	<u>Код клиента</u>	<u>Имя клиента</u>	<u>Адрес клиента</u>	<u>Код товара</u>	<u>Описание товара</u>	<u>Цена</u>	<u>Количество</u>	<u>Итого</u>
100	05/05/17	545	ООО«Флаг»	Ленинская, 45	2199	Ноутбук ACER Aspire	25500	2	51000
100	05/05/17	545	ООО«Флаг»	Ленинская, 45	1115	Проектор Optoma	43200	1	43200
100	05/05/17	545	ООО«Флаг»	Ленинская, 45	1123	Проектор Epson	21000	2	42000
100	05/05/17	545	ООО«Флаг»	Ленинская, 45	3312	МФУ HP	25100	3	75300
101	05/05/17	120	АО «Сервис»	Солнечная, 23	2224	Ноутбук Dell Inspiron	18200	1	18200
101	05/05/17	120	АО «Сервис»	Солнечная, 23	9978	Офис. кресло Rec	5300	2	10600
102	20/06/17	777	АО «Гарант»	Московское шоссе, 71	9978	Офис. кресло Rec	5300	5	26500
102	20/06/17	777	АО «Гарант»	Московское шоссе, 71	5556	Принтер Epson	6700	2	13400
103	22/06/17	545	ООО «Флаг»	Ленинская, 45	1166	Проектор INFOCUS	16200	3	48600
103	22/06/17	545	ООО «Флаг»	Ленинская, 45	3344	МФУ Epson	13000	2	26000
103	22/06/17	545	ООО «Флаг»	Ленинская, 45	1123	Проектор Epson	21000	1	21000
103	22/06/17	545	ООО «Флаг»	Ленинская, 45	6611	3D ручка UNID	2900	3	8700

Рис. 4.23. Отношение ЗАКАЗЫ, приведенное к 1НФ

Следующий этап нормализации – приведение ко второй нормальной форме. 2НФ позволяет уменьшить количество повторяющихся данных. Вторая НФ предполагает, что каждый столбец зависит от всего первичного ключа. Необходимо проанализировать, какие ФЗ имеются в отношении ЗАКАЗЫ. Нагляднее всего использовать диаграмму функциональных зависимостей (рис. 4.24).

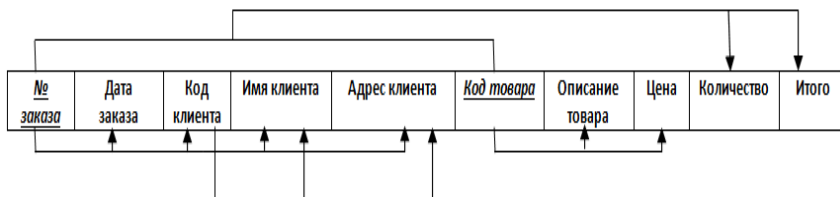


Рис. 4.24. Диаграмма функциональных зависимостей отношения ЗАКАЗЫ

К выделению этих ФЗ для рассматриваемого Примера приводят следующие соображения.

1. Каждому коду товара однозначно соответствует его описание и цена, т.е. имеют место функциональные зависимости:

Код товара → Описание товара

Код товара → Цена

2. Каждому номеру заказа однозначно соответствуют дата заказа, код клиента, его имя и адрес.

№ заказа → Дата заказа

№ заказа → Код клиента

№ заказа → Имя клиента

№ заказа → Адрес клиента

3. Каждому коду клиента однозначно соответствует его имя и адрес.

Код клиента → Имя клиента

Код клиента → Адрес клиента

4. Количество заказываемого товара и итоговая стоимость однозначно идентифицируется только составным первичным ключом отношения.

№ заказа, Код товара → Количество

№ заказа, Код товара → Итого

Таким образом, можно сделать вывод, что в отношении ЗАКАЗЫ имеются частичные функциональные зависимости, а также зависимости неключевых атрибутов от других неключевых атрибутов (транзитивные зависимости).

Для перевода отношения ЗАКАЗЫ во 2НФ произведем операцию проекции. В результате получим следующий набор отношений: ЗАКАЗ_КЛИЕНТ, ТОВАР, ЗАКАЗЫ (рис. 4.25):

ЗАКАЗ_КЛИЕНТ

<i>№ заказа</i>	<i>Дата заказа</i>	<i>Код клиента</i>	<i>Имя клиента</i>	<i>Адрес клиента</i>
100	05/05/17	545	ООО«Флаг»	Ленинская, 45
101	05/05/17	120	АО «Сервис»	Солнечная, 23
102	20/06/17	777	АО «Гарант»	Московское шоссе, 71
103	22/06/17	545	ООО «Флаг»	Ленинская, 45

ТОВАР

<i>Код товара</i>	<i>Описание товара</i>	<i>Цена</i>
2199	Ноутбук ACER Aspire	25500
1115	Проектор Optoma	43200
1123	Проектор Epson	21000
3312	МФУ HP	25100
2224	Ноутбук Dell Inspiron	18200
9978	Офис. кресло Rec	5300
5556	Принтер Epson	6700
1166	Проектор INFOCUS	16200
3344	МФУ Epson	13000
6611	3D ручка UNID	2900

ЗАКАЗЫ

<i>№ заказа</i>	<i>Код товара</i>	<i>Количество</i>	<i>Итого</i>
100	2199	2	51000
100	1115	1	43200
100	1123	2	42000
100	3312	3	75300
101	2224	1	18200
101	9978	2	10600
102	9978	5	26500
102	5556	2	13400
103	1166	3	48600
103	3344	2	26000
103	1123	1	21000
103	6611	3	8700

Рис. 4.25. Набор отношений, полученных в результате декомпозиции исходного отношения ЗАКАЗЫ

Внимательно изучив полученные в результате декомпозиции отношения, можно сказать, что переход во 2 НФ позволил исключить явную избыточность данных в таблице ТОВАР – повторение строк со сведениями об определенном товаре. В таблице ЗАКАЗ_КЛИЕНТ исключили повторение информации о клиенте, однако в этой таблице по-прежнему имеет место неявное дублирование данных. В данном отношении присутствуют транзитивные зависимости:

№ заказа → Код клиента

№ заказа → Имя клиента

№ заказа → Адрес клиента

Код клиента → Имя клиента

Код клиента → Адрес клиента

Для дальнейшего совершенствования отношения необходимо преобразовать его в 3НФ. Устраним транзитивные зависимости, ис-

пользуя операцию проекции на атрибуты, являющиеся причиной транзитивных зависимостей. Преобразуем отношение ЗАКАЗ_КЛИЕНТ, получив при этом обновленное отношение ЗАКАЗ_КЛИЕНТ и новое отношение КЛИЕНТ (рис. 4.26).

ЗАКАЗ_КЛИЕНТ			КЛИЕНТ		
<u>№ заказа</u>	<u>Дата заказа</u>	<u>Код клиента</u>	<u>Код клиента</u>	<u>Имя клиента</u>	<u>Адрес клиента</u>
100	05/05/17	545	545	ООО «Флаг»	Ленинская, 45
101	05/05/17	120	120	АО «Сервис»	Солнечная, 23
102	20/06/17	777	777	АО «Гарант»	Московское шоссе, 71
103	22/06/17	545			

Рис. 4.26. Отношения, полученные в результате декомпозиции первоначального отношения ЗАКАЗ_КЛИЕНТ

Все полученные отношения находятся в третьей нормальной форме, так как в них нет частичных функциональных зависимостей. Транзитивных зависимостей в данных отношениях тоже нет.

Таким образом, мы провели процесс нормализации. В результате получили четыре отношения:

- ЗАКАЗЫ (№ заказа, Код товара, Количество, Итого)
- ТОВАР (Код товара, Описание товара, Цена)
- ЗАКАЗ–КЛИЕНТ (№ заказа, Дата заказа, Код клиента)
- КЛИЕНТ (Код клиента, Имя клиента, Адрес клиента)

4.6. Контрольные вопросы к главе 4

1. Приведите примеры избыточного и неизбыточного дублирования данных. Какие способы существуют для решения этой проблемы. Приведите примеры.

2. Приведите примеры аномалий обновления отношений.

3. Назовите правила преобразования ER–диаграмм в схему базы данных.
4. Дайте определение функциональной зависимости, функциональной взаимозависимости, частичной зависимости и транзитивной зависимости.
5. Аксиомы вывода функциональных зависимостей.
6. Аксиомы вывода многозначных функциональных зависимостей.
7. Дайте определения 1НФ, 2 НФ, 3НФ. Приведите примеры.
8. Дайте определения НФ Бойса–Кодда, 4 НФ. Приведите примеры.
9. Другие виды нормальных форм отношений.
10. Какие задачи решает денормализация? Назовите основные виды денормализации.
11. Для произвольной предметной области приведите пример нормализации до 3 НФ.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Хомоненко, А.Д. Базы данных [Текст]: учебник для высших учебных заведений / А.Д. Хомоненко, В.М. Цыганков, М.Г. Мальцев; под ред. проф. А.Д. Хомоненко. – 6-е изд., доп. – СПб.: КОРОНА–Век, 2009. – 736 с.
2. Вендров, А.М. CASE–технологии. Современные методы и средства проектирования информационных систем [Текст] / А.М. Вендров. – М.: Финансы и статистика, 2005. – 176 с.
3. Глушаков, С.В. Базы данных [Текст]: учеб. издание / С.В. Глушаков, Д.В. Ломотько. – Харьков: Фолио; М.: ООО «Издательство АСТ», 2002. – 504 с.
4. ГОСТ 20886–85. Организация данных в системах обработки данных. Термины и определения [Текст]. – Взамен ГОСТ 20886-75; введ. 1986-07-01. – М.: Стандартиформ, 2005.
5. ГОСТ 7.73–96. Система стандартов по информации, библиотечному и издательскому делу. Поиск и распространение информации. Термины и определения [Текст]. – Взамен ГОСТ 7.27-80; введ. 1998-01-01. – Минск: Межгос. совет по стандартизации, метрологии и сертификации; М.: ИПК Издательство стандартов, 1997.
6. ГОСТ Р 52653–2006. Информационно–коммуникационные технологии в образовании. Термины и определения [Текст]. – Введ. 2006-12-27. – М.: Стандартиформ, 2007.
7. Карпова, И.П. Базы данных [Текст]: учеб. пособие / И.П. Карпова. – СПб.: Питер, 2013.– 240 с.
8. Кириллов, В.В. Основы проектирования реляционных баз данных [Текст]: учеб. пособие / В.В. Кириллов. – Санкт–

- Петербургский гос. институт точной механики и оптики (техн. ун-т), кафедра вычислительной техники. URL: <http://khpri-iiip.mipk.kharkiv.edu/library/dbms/kir1/> (дата обращения 12.08.2019).
9. Образовательный портал «Электронный университет ВГУ». Курс «Базы данных». URL: <https://edu.vsu.ru/mod/book/view.php?id=52629&chapterid=14291> (дата обращения 19.02.2019).
 10. Проектирование баз данных. Основы проектирования реляционных баз данных. URL: http://www.intuit.ru/studies/professional_retraining/953/courses/191/info (дата обращения 19.02.2019).
 11. Проектирование реляционных баз данных // Учебные материалы ОКСО 210000. Электронная техника, радиотехника и связь. Лекции для преподавателей и студентов ВУЗ. URL: <http://siblec.ru/index.php?dn=html&way=bW9kL2h0bWwvY29udGVudC84c2VtLzA3Mi9tYWluLmh0bQ==> (дата обращения 12.03.2019).
 12. Роб, П. Системы баз данных: проектирование, реализация и управление [Текст] / П. Роб, К. Коронел. – 5-е изд., перераб. и доп., пер. с англ. – СПб.: БХВ-Петербург, 2004. – 1040 с.

Учебное издание

*Попова-Коварцева Дарья Александровна,
Сопченко Елена Вильевна*

ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ

Учебное пособие

Текст печатается в авторской редакции
Техническое редактирование Т.К. К р е т и н и н о й
Подготовка оригинал-макета Л.Р. Д м и т р и е н к о

Подписано в печать 27.11.2019. Формат 60x84 1/16.

Бумага офсетная. Печ. л. 7,0.

Тираж 120 экз. (1 з-д 1-30). Заказ .

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)
443086, Самара, Московское шоссе, 34.

Изд-во Самарского университета.
443086, Самара, Московское шоссе, 34.

