ГЛАВА 4. ФОРМАЛИЗАЦИЯ РЕЛЯЦИОННОЙ МОДЕЛИ

4.1. Формализованное описание отношений и схемы отношений

Как уже отмечалось в п. 3.4, реляционная модель описывает представление данных в виде двумерной таблицы, называемой отношением. Наименованиями столбцов этой таблицы служат имена атрибутов. Рассмотрим формализованное описание соответствующих понятий.

Схемой отношений $R(A_1, A_2, ..., A_n)$ называется конечное множество имен атрибутов $\{A_1, A_2, ..., A_n\}$, где n – арность схемы отношения. Понятию «схема отношения» соответствует описание структуры двумерной таблицы (имена столбцов). Каждому имени атрибута A_i соответствует допустимое множество значений, которые может принимать атрибут A_i. Это множество значений D_i называется доменом атрибута A_i, $i = \overline{1,n}$. По определению, домены являются непустыми конечными или счетными множествами.

Уточним, что в теории реляционных баз данных домен рассматривается как множество значений одного (причем простого) типа данных. Понятию домена D_i соответствует множество значений, стоящих в столбце A_i рассматриваемой таблицы.

Пусть $D = D_1 \cup D_2 \cup \ldots \cup D_n$.

Отношением r со схемой R называется конечное множество отображений $\{t_1, t_2, ..., t_p\}$ из множества R: $\{A_1, A_2, ..., A_n\}$ в множество D: $\{ D_1 \cup D_2 \cup \ldots \cup D_n \}$, таких, что $t_k(A_i) \in D_i$, $k = \overline{1,p}$; $i = \overline{1,n}$.

Отображение t_k называется k-м кортежем, n – размерность кортежа.

Понятию k-го кортежа соответствует множество значений, стоящих в k-й строке рассматриваемой таблицы.

Понятию отношения r соответствует множество значений, стоящих во всех строках рассматриваемой таблицы.

Ключом (первичным ключом) отношения r со схемой R называется подмножество $K = \{A_{i1}, A_{i2}, ..., A_{im}\} \subseteq \{A_1, A_2, ..., A_n\}$, где $\{i1, i2, ..., im\} \subseteq \{1, 2, \ldots, n\}$, такое, что любые два различных кортежа $t_1, t_2 \in r$ $(t_1 \neq t_2)$ не совпадают по значениям множества $K = \{A_{i1}, A_{i2}, ..., A_{im}\}$.

Таким образом, достаточно знать значение кортежа на множестве K, чтобы однозначно его идентифицировать.

Совокупность схем отношений, используемых для представления концептуальной модели, называется схемой реляционной базы данных (реляционной моделью данных). Текущие значения соответствующих отношений называются реляционной базой данных.

Отметим следующие свойства отношения:

1. Порядок рассмотрения атрибутов в схеме отношения (отношении) не имеет значения, т.к. для ссылки на значение атрибута в кортеже отношения всегда используется имя атрибута.
2. Значения всех атрибутов являются атомарными (неделимыми). Это следует из определения домена как множества значений простого типа данных, т.е. среди значений домена не могут содержаться множества.
3. Порядок рассмотрения кортежей в отношении не имеет значения, т.к. отношение представляет собой множество кортежей, а элементы множества, по определению теории множеств, неупорядочены.

4.2. Манипулирование данными в реляционной модели

Для манипулирования данными в реляционной модели используются два формальных аппарата:

- реляционная алгебра, основанная на теории множеств;
- реляционное исчисление, базирующееся на исчислении предикатов первого порядка.

Операции, реализуемые с помощью указанных аппаратов, обладают важным свойством: они замкнуты на множестве отношений. Это означает, что выражения реляционной алгебры и формулы реляционного исчисления определяются над отношениями реляционных БД и результатом вычисления также являются отношения. В результате любое выражение или формула могут интерпретироваться как отношение, что позволяет использовать их в других выражениях или формулах.

Как мы увидим, алгебра и исчисление обладают большой выразительной мощностью, очень сложные запросы к базе данных могут быть выражены с помощью одного выражения реляционной алгебры или одной формулы реляционного исчисления. Именно по этой причине такие механизмы включены в реляционную модель данных. Конкретный язык манипулирования реляционными БД называется *реляционно полным,* если любой запрос, выражаемый с помощью одной операции реляционной алгебры или одной формулы реляционного исчисления, может быть выражен с помощью одного оператора этого языка.

Механизмы реляционной алгебры и реляционного исчисления эквивалентны, т.е. для любого допустимого выражения реляционной алгебры можно построить эквивалентную формулу реляционного исчисления и наоборот. Почему же в реляционной модели данных присутствуют оба эти механизма?

Дело в том, что они различаются уровнем процедурности. Выражения реляционной алгебры строятся на основе алгебраических операций (высокого уровня), и подобно тому, как интерпретируются арифметические и логические выражения, выражение реляционной алгебры также имеет процедурную интерпретацию. Другими словами, запрос, представленный на языке реляционной алгебры, может быть реализован на основе вычисления элементарных алгебраических операций с учетом их старшинства и возможного наличия скобок. Для формулы реляционного исчисления однозначная интерпретация, вообще говоря, отсутствует. Формула только устанавливает условия, которым должны удовлетворять кортежи результирующего отношения. Поэтому языки реляционного исчисления являются более непроцедурными или декларативными.

Поскольку механизмы реляционной алгебры и реляционного исчисления эквивалентны, то в конкретной ситуации для проверки степени реляционности некоторого языка БД можно пользоваться любым из этих механизмов.

Заметим, что крайне редко алгебра или исчисление принимаются в качестве полной основы какого-либо языка БД. Обычно (как, например, в случае языка SQL) язык основывается на некоторой смеси алгебраических и логических конструкций. Тем не менее знание алгебраических и логических основ языков баз данных часто бывает полезно на практике.

4.3. Операции реляционной алгебры

Операции реляционной алгебры определены на множестве отношений и являются замкнутыми относительно этого множества (образуют алгебру). Оказывается, что любой произвольный запрос к БД можно представить в виде последовательности, составленной из пяти основных операций реляционной алгебры. Рассмотрим эти операции.

Объединение $r \cup s$

Объединением отношений r и s называется множество кортежей, которые принадлежат или r, или s, или им обоим. Для операции объединения требуется одинаковая арность отношений.

Для примера, пусть

r

a	b	a
d	a	f
c	b	d

s

b	g	a
d	a	f

тогда

$r \cup s$

a	b	a
d	a	f
c	b	d
b	g	a

Заметим, что с помощью операции объединения может быть реализовано добавление нового кортежа к имеющемуся отношению. В этом случае *r* – исходное отношение, *s* – отношение, содержащее один добавляемый кортеж.

Разность *r – s*

Разностью отношений *r* и *s* называется множество кортежей, принадлежащих *r*, но не принадлежащих *s*. Для этой операции также требуется одинаковая арность отношений.

$r - s$

a	b	a
c	b	d

Заметим, что с помощью операции разности может быть реализовано удаление кортежа из имеющегося отношения. В этом случае *r* – исходное отношение, *s* – отношение, содержащее один удаляемый кортеж.

Декартово произведение *r* ×*s*

Пусть *r* и *s* – отношения арности $k1$ и $k2$ соответственно. Декартовым произведением $r \times s$ называется множество кортежей длины $k1+k2$, первые $k1$ компонентов которых образуют кортежи, принадлежащие *r*, а последние $k2$ – кортежи, принадлежащие *s*.

$r \times s$

a	b	a	b	g	a
a	b	a	d	a	f
d	a	f	b	g	a
d	a	f	d	a	f
c	b	d	b	g	a
c	b	d	d	a	f

Проекция $\pi_{A_{i1}, A_{i2},..., A_{im}}(r)$

Проекция $\pi_{A_{i1}, A_{i2},..., A_{im}}(r)$ есть множество кортежей, получаемых из кортежей отношения *r* выбором столбцов с именами A_{i1}, A_{i2}, …, A_{im}. Другими словами, это операция построения «вертикального» подмножества, получаемого путем выбора определенных атрибутов и исключения остальных. Повторяющиеся кортежи исключаются.

$\pi_{1,3}(r)$

a	a
d	f
c	d

Выбор (селекция) $\sigma_F(r)$

Пусть F – формула, образованная: операндами, являющимися константами или именами атрибутов, арифметическими операторами сравнения, логическими операторами (и, или, не), тогда выбором (селекцией) σ_F называется множество кортежей, компоненты которого удовлетворяют условию, заданному формулой F.

$\sigma_{(1)=(3)}(r)=$

a	b	a

Здесь F:(1)=(3) – содержимое первого столбца равно содержимому третьего столбца.

Приведем ряд примеров представления запросов в задаче зачисления абитуриентов с помощью формальных операций.

Пример 1.

Сформировать список абитуриентов (фамилия, имя, отчество).

Рассмотрим сущность АБИТУРИЕНТ (*entrant*), представленную в ER-диаграмме рис. 23. Атрибут «Фамилия» обозначен здесь «*sur-*

name». Для ответа на запрос необходимо взять проекцию отношения entrant на столбец *surname*:

$$\pi_{surname}(entrant).$$

Пример 2.

Выдать список фамилий и дат рождений абитуриентов, которым на текущую дату (*tek_date*) больше 35 лет.

Рассмотрим то же отношение entrant. Сначала выбираем абитуриентов, которым больше 35 лет: $\sigma_{birth\_date+35<tek\_date}(entrant)$.

Затем берем проекцию полученного отношения на столбцы *surname* и *birth_date*:

$$\pi_{surname,\, birth\_date}(\sigma_{birth\_date+35<tek\_date}(entrant)).$$

Заметим, что можно было бы выполнить эти две операции в другой последовательности – сначала проекция, а затем селекция. Предлагается оценить, какой из этих вариантов лучше по оценке числа выполняемых элементарных действий и объему требуемой памяти.

Пример 3.

Выдать список фамилий абитуриентов, получивших оценку «неудовлетворительно». Рассматриваем ту же ER-диаграмму на рис. 23.

Оценка абитуриента является атрибутом отношения *mark*. Если бы в этом отношении присутствовал атрибут «фамилия», то задача решалась бы аналогично примеру 2. В отношении mark присутствует атрибут «идентификатор абитуриента» – *entrant_id*, а фамилия присутствует в отношении *entrant*. Для ответа на этот запрос необходимо связывать по *entrant_id* отношение *mark* и отношение *entrant*.

Сначала выберем из отношения *mark* кортежи с оценкой «*неудовлетворительно*». Обозначим полученное отношение *R*1. (Дальнейшие промежуточные отношения будем обозначать последовательно *R*2, *R*3 и т.д.).

$$R1 = \sigma_{mark=«неудовлетворительно»}(mark).$$

Далее нас будет интересовать только атрибуты *entrant_id* и *mark*. Поэтому возьмем проекцию на эти столбцы.

$$R2 = \pi_{entrant\_id,\, mark}(R1).$$

Далее необходимо связать отношения *entrant* и *R*2 (склеить таблицы). Для склейки таблиц используется операция «декартово произведение»:

$$R3 = entrant \times R2.$$

В отношении $R3$ присутствуют два одинаковых столбца *entrant_id* из отношений *entrant* и $R2$. Выбирая из отношения $R3$ строки, в которых значения *entrant_id* в соответствующих столбцах совпадают, получим сведения об абитуриентах, получивших оценку «неудовлетворительно»:

$$R4 = \sigma_{entrant.\ entrant\_id\ =\ R2.\ entrant\_id}(R3),$$

где *entrant. entrant_id* и *R2. entrant_id* обозначают соответственно столбец *entrant_id* соответствующей первой и второй составной части декартова произведения. Теперь осталось только выбрать фамилии соответствующих студентов

$$R5 = \pi_{surname}(R4).$$

Получаем требуемый результат. Заметим, что для экономии действий и памяти, перед тем как склеивать таблицы, целесообразно было сделать операцию проекции отношения *entrant* на столбцы *entrant_id, surname* (чтобы не включать в декартово произведение лишние столбцы).

Введенные пять основных операций реляционной алгебры позволяют реализовать любой запрос к реляционной базе данных. Однако наряду с основными операциями достаточно часто удобно использовать так называемые дополнительные операции реляционной алгебры (которые могут быть выражены через основные).

Пересечение $r \cap s$

Пересечением отношений r и s называется множество кортежей, принадлежащих как r, так и s. Пересечение может быть выражено через операции разности

$$r \cap s = r - (r - s).$$

θ-соединение $r \underset{i\,\theta\,j}{\triangleright\!\triangleleft} s$

θ-соединение r и s по столбцам A_i и A_j представляет собой множество таких кортежей в декартовом произведении r и s, что i-й компонент r находится в отношении θ с j-м компонентом s, где θ– арифметический оператор сравнения. Если θ является оператором равенства, то эта операция называется эквисоединением.

$$r \underset{i\,\theta\,j}{\triangleright\!\triangleleft} s = \sigma_{i\theta(l+j)}(r \times s),$$

где l – арность отношения r.

Пример.

r

1	2	3
4	5	6
7	8	9

s

3	1
6	2

$r \triangleright\triangleleft s$
(2)<(1)

1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

Заметим, что в примере 3 две последовательно идущие операции (декартово произведение и селекция) вместе как раз представляют операцию соединения. Причем использование декартова произведения для соединения таблиц обязательно обусловливает использование селекции как следующей операции для установления связи между таблицами. Поэтому целесообразно использовать такую объединенную операцию и программно реализовывать в СУБД именно операцию соединения.

Естественное соединение $r \triangleright\triangleleft s$

Операция применима тогда и только тогда, когда столбцы имеют имена (являются атрибутами). Операция применима к отношениям, у которых есть одинаковые атрибуты.

Пусть

$$r = (A1, ..., Ak, B1,..., Bn),\ s = (A1, ..., Ak, C1,..., Cm),$$

имена $A1, ..., Ak$ совпадают.

Тогда $r \triangleright\triangleleft s$ определяется следующим образом

$$r \triangleright\triangleleft s = \pi_{B1,..., Bn,\ A1,..., Ak,\ C1,..., Cm}(\sigma_{r.A1=s.A1\&r.A2=s.A2\&...r.Ak=s.Ak}(r \times s)).$$

Для подчеркивания важности приведенных операций реляционной алгебры, а также для уточнения понятия реляционной СУБД приведем следующее определение одного из ведущих специалистов в области реляционных баз данных К.Дж. Дейта: «Будем называть систему реляционной, если она поддерживает, по крайней мере, реляционные базы данных, т.е. базы данных, которые могут восприниматься пользователем как таблицы и только как таблицы, операции селекции, проекции

и соединения реляционной алгебры, не требуя при этом, чтобы каким-то образом были предопределены физические пути доступа для поддержки этих операций».

4.4. Использование формального аппарата для оптимизации схем отношений

4.4.1. Проблема выбора рациональных схем отношений

При представлении концептуальной схемы в виде реляционной модели возможны различные варианты выбора схем отношений. Одни варианты выбора рассматривались в предыдущих разделах (п. 3.4), другие получаются объединением (или разбиением) некоторых схем отношений. От правильного выбора схем отношений, представляющих концептуальную схему, в значительной степени будет зависеть эффективность функционирования базы данных.

Рассмотрим для примера конкретную схему отношений и проанализируем её недостатки. Предположим, что данные об абитуриентах, специальностях, формах обучения включены в таблицу, содержащую данные об атрибутах. Схема отношения имеет следующий вид:
АБИТУРИЕНТ (код абитуриента, ФИО, факультет, специальность, форма обучения).

Эта схема отношений обусловливает следующие недостатки соответствующей базы данных:

- Дублирование информации (избыточность).

  У абитуриентов, поступающих на один факультет, будет повторяться название факультета. Для разных факультетов будут повторяться одинаковые формы обучения, специальности.
- Потенциальная противоречивость (аномалии обновления).

  Если, например, изменится название специальности, то изменяя её в одном кортеже (у одного абитуриента), необходимо изменять и во всех других кортежах, где она присутствует.
- Потенциальная возможность потери сведений (аномалии удаления).

  При удалении информации о всех абитуриентах, поступающих на определенную специальность, мы теряем все сведения об этой специальности.
- Потенциальная возможность невключения информации в базу данных (аномалии включения).

  В базе данных будут отсутствовать сведения о специальности, если на нее абитуриенты не подали заявлений.

В теории реляционных баз данных существуют формальные методы построения реляционной модели базы данных, в которой отсутствует избыточность и аномалии обновления, удаления и включения.

Построение рационального варианта схем отношений (обладающего лучшими свойствами при операциях включения, модификации и удаления данных, чем все остальные наборы схем) осуществляется путем так называемой нормализации схем отношений. Нормализация производится в несколько этапов. На начальном этапе схема отношений должна находиться в первой нормальной форме (1НФ). Отношение находится в первой нормальной форме, если все атрибуты отношения принимают простые значения (атомарные или неделимые), не являющиеся множеством или кортежем из более элементарных составляющих. Далее отношение, представленное в первой нормальной форме, последовательно преобразуется во вторую и третью нормальные формы. Процесс построения второй и третьей нормальных форм будет описан в следующих подразделах. При некоторых предположениях о данных третья нормальная форма является искомым наилучшим вариантом.

Если эти предположения не выполняются, то процесс нормализации продолжается и отношение преобразуется в четвертую и пятую нормальные формы. Построение соответствующих форм описано в [24, 28 и др.] и в данной книге не рассматривается.

Прежде чем перейти к построению второй нормальной формы, необходимо определить ряд формальных понятий.

4.4.2. Функциональные зависимости (зависимости между атрибутами отношения)

Пусть $R(A_1, A_2, \ldots, A_n)$ – схема отношения, а X и Y – подмножества $\{A_1, A_2, \ldots, A_n\}$. Тогда X функционально определяет Y или Y функционально зависит от X (обозначается $X \rightarrow Y$) тогда и только тогда, когда каждое значение множества X отношения R связано с одним значением множества Y отношения R. Иначе говоря, если два кортежа R совпадают по значению X, они совпадают и по значению Y.

Замечание. Функциональные зависимости характеризуют все отношения, которые могут быть значениями схемы отношения R в принципе. Поэтому единственный способ определить функциональные зависимости – внимательно проанализировать семантику (смысл) атрибутов.

Функциональные зависимости являются, в частности, ограничениями целостности, поэтому целесообразно проверять их при каждом обновлении базы данных.

Пример функциональной зависимости:

Код абитуриента → Фамилия, Имя, Отчество.

Полное множество функциональных зависимостей

Для каждого отношения существует вполне определенное множество функциональных зависимостей между атрибутами данного отношения. Причем из одной или более функциональных зависимостей, присущих рассматриваемому отношению, можно вывести другие функциональные зависимости, также присущие этому отношению. Заданное множество функциональных зависимостей для отношения R обозначим F, полное множество функциональных зависимостей, которые логически можно получить из F, называется замыканием F и обозначается F^+.

Если множество функциональных зависимостей совпадает с замыканием данного множества, то такое множество функциональных зависимостей называется полным.

Введенные понятия позволяют формально определить понятие ключа.

Пусть существует некоторая схема R с атрибутами $A_1A_2\ldots A_n$, F – некоторое множество функциональных зависимостей и X – некоторое подмножество R. Тогда X называется ключом, если, во-первых, в F^+ существует зависимость $X \rightarrow A_1A_2...A_n$ и, во-вторых, ни для какого подмножества Y, входящего в X, зависимость $Y \rightarrow A_1A_2...A_n$ не принадлежит F^+.

Полной функциональной зависимостью называется зависимость неключевого атрибута от всего составного ключа. Частичной функциональной зависимостью будем называть зависимость неключевого атрибута от части составного ключа.

Для вычисления замыкания множества функциональных зависимостей используются следующие правила вывода (аксиомы Армстронга):

Пусть известна некоторая схема отношения R $\{A_1, A_2,\ldots, A_n\}$ с множеством атрибутов $U=\{A_1, A_2,\ldots, A_n\}$ и множество функциональных зависимостей F, заданных на множестве U.

Аксиома рефлективности. Если Y входит в X, а X входит в U ($Y \subseteq X \subseteq U$), то $X \rightarrow Y$ логически следует из F. Это правило дает тривиальные зависимости, так как в них правая часть содержится в левой части.

Аксиома пополнения. Если $X \rightarrow Y$ и Z есть подмножество U, то $XZ \rightarrow YZ$. В данном случае функциональная зависимость $X \rightarrow Y$ либо содержалась в исходном множестве F, либо может быть выведена из F с использованием описываемых аксиом.

Аксиома транзитивности. Если $X \rightarrow Y$ и $Y \rightarrow Z$, то $X \rightarrow Z$.

Справедлива следующая **теорема**. Аксиомы Армстронга являются полными и надежными.

Это значит, что используя их мы выведем все возможные функциональные зависимости, логически следующие из F, и не выведем никаких лишних зависимостей.

Существует несколько других правил вывода, которые следуют из аксиом Армстронга.

Правило самоопределения. $X \rightarrow X$.

Правило объединения. Если $X \rightarrow Y$ и $X \rightarrow Z$, то $X \rightarrow Y \cup Z$.

Правило псевдотранзитивности. Если $X \rightarrow Y$ и $W \cup Y \rightarrow Z$, то $X \cup W \rightarrow Z$.

Правило композиции. Если $X \rightarrow Y$ и $Z \rightarrow W$, то $X \cup W \rightarrow Y \cup W$.

Правило декомпозиции. Если $X \rightarrow Y$ и Z входит в Y, то $X \rightarrow Z$.

Надо отметить, что вычисление замыкания множества функциональных зависимостей является трудоемкой задачей при достаточно большом количестве атрибутов (за счет выписывания большого количества тривиальных зависимостей).

4.4.3. Декомпозиция схемы отношения

Последовательный переход от одной нормальной формы к другой при нормализации схем отношений реализуется через декомпозицию. Основной операцией, с помощью которой осуществляется декомпозиция, является проекция.

Декомпозицией схемы отношения $R = \{A_1, A_2, \ldots A_n\}$ называется замена ее совокупностью подмножеств R, таких, что их объединение дает R. При этом допускается, чтобы подмножества были пересекающимися.

Декомпозиция должна обеспечить то, что запросы (выборка данных по условию) к исходному отношению и отношениям, получаемым в результате декомпозиции, дадут одинаковый результат. Соответствующее условие будет выполняться, если каждый кортеж отношения R может быть представлен как естественное соединение его проекций на каждое из подмножеств. В этом случае говорят, что декомпозиция обладает свойством соединения без потерь.

Алгоритм декомпозиции основан на следующей теореме.

Теорема. Пусть $R(A, B, C)$ – отношение, A, B, C – атрибуты.

Если R удовлетворяет зависимости $A \rightarrow B$, то R равно соединению его проекций A, B и A, C

$$R(A, B, C) \qquad \begin{matrix} R(A, B) \\ R(A, C) \end{matrix}$$

При нормализации необходимо выбирать такие декомпозиции, которые обладают свойством соединения без потерь. Для проверки, обладает ли декомпозиция данным свойством, используется специальный алгоритм [28, 40]. Алгоритм состоит в следующем.

Пусть есть некоторая схема отношения $R = A_1 \ldots A_n$, множество функциональных зависимостей F и некоторая декомпозиция $(R_1, \ldots R_k)$ исходной схемы, состоящая из k подсхем.

Необходимо построить таблицу с n столбцами и k строками. Столбец j соответствует атрибуту A_j, строка k – схеме отношения R_k. На пересечении строки i и столбца j поместим символ a_j, если A_j принадлежит R_i. В противном случае поместим туда символ b_{ij}.

Рассматриваем каждую зависимость из множества F до тех пор, пока в таблице невозможно сделать какие-либо изменения. Всякий раз, рассматривая зависимость $X \rightarrow Y$, мы ищем строки, которые совпадают по всем столбцам, соответствующим атрибутам из X. При обнаружении таких строк отождествляем символы в столбцах, соответствующих атрибутам из Y. Если при этом один из отождествляемых символов равен a_j, то приравниваем и другой a_j. В том случае когда они равны b_{ij} и b_{lj}, делаем их оба равными b_{ij} или b_{lj} по своему усмотрению.

После модификации строк таблицы указанным выше способом может обнаружиться, что некоторая строка стала равной $a_1 \ldots a_k$. Тогда декомпозиция обладает свойством соединения без потерь. Если такой строки не получается, то декомпозиция не обладает таким свойством.

Пример.

Декомпозиция схемы $ABCD$ на AB и ACD.

$A \rightarrow B$, $AC \rightarrow D$ (AC – сокращенная запись $A \cup C$).

A	B	C	D
a1	a2	b13	b14
a1	b22	a3	a4

Поскольку $A \rightarrow B$ и две строки совпадают по A, то

A	B	C	D
a1	a2	b13	b14
a1	a2	a3	a4

Так как одна строка состоит из всех «а», то наша декомпозиция обладает свойством соединения без потерь.

Вторым важнейшим желательным свойством декомпозиции является свойство сохранения функциональных зависимостей.

Стремление к тому, чтобы декомпозиция сохраняла зависимости, естественно. Функциональные зависимости являются некоторыми ограничениями на данные. Если декомпозиция не обладает этим свойством, то для того чтобы проверить, не нарушают ли введенные данные условия целостности (функциональные зависимости), нам приходится соединять все проекции.

Таким образом, для правильно построенного проекта базы данных необходимо, чтобы декомпозиции обладали свойством соединения без потерь, и желательно, чтобы они обладали свойством сохранения функциональных зависимостей.

4.4.4. Выбор рационального набора схем отношений путем нормализации

Вторая нормальная форма (2НФ)

Отношение находится в 2НФ, если оно находится в 1НФ и каждый неключевой атрибут зависит от первичного ключа (не зависит от части ключа).

Для перевода отношения в 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 ключа и атрибуты, зависящие от этих частей.

Третья нормальная форма (3НФ)

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

Отношение находится в 3НФ в том и только том случае, если все неключевые атрибуты отношения взаимно независимы и полностью зависят от первичного ключа.

Оказывается, что любая схема отношений может быть приведена к 3НФ декомпозицией, обладающей свойствами соединения без потерь и сохраняющей зависимости.

Мотивировка третьей нормальной формы

Третья нормальная форма исключает избыточность и аномалии включения и удаления. К сожалению, 3НФ не предотвращает все возможные аномалии.

Нормальная форма Бойса-Кодда (НФБК)

Если в R для каждой зависимости $X \rightarrow A$, где A не принадлежит X, X включает в себя некоторый ключ, то говорят, что данное отношение находится в нормальной форме Бойса-Кодда.

Детерминантом функциональной зависимости называется минимальная группа атрибутов, от которой зависит некоторый другой атрибут или группа атрибутов, причем эта зависимость нетривиальная.

Отношение находится в НФБК тогда и только тогда, когда каждый его детерминант является потенциальным ключом.

НФБК является более строгой версией 3НФ. Иными словами, любое отношение, находящееся в НФБК, находится в 3НФ. Обратное неверно.

Пример. Расписание консультаций. Каждая группа может прийти на консультацию один раз в день. Для проведения консультации или консультаций преподавателю на определенный день предоставляется аудитория. В течение дня данная аудитория может использоваться разными преподавателями. Введем отношение:

РАСПИСАНИЕ КОНСУЛЬТАЦИЙ (Группа, Дата, Время, Преподаватель, Аудитория)

В качестве первичного ключа выбрали «Группа, Дата».

Потенциальные ключи: «Группа, Дата»; «Преподаватель, Дата, Время»; «Аудитория, Дата, Время».

Группа	Дата	Время	Преподаватель	Аудитория
721	13 мая	10.30	Визгунов	401
722	13 мая	12.00	Визгунов	401
723	13 мая	12.00	Трифонов	402
722	1 июня	10.30	Визгунов	402

Все атрибуты зависят от всего ключа. Нет неполных зависимостей. Нет транзитивных зависимостей.

Однако существует следующая зависимость – Преподаватель, Дата определяют ($\rightarrow$) Аудитория.

Нарушается определение формы БК – данный составной атрибут является детерминантом, но не является потенциальным ключом.

Данная функциональная зависимость не нарушает третьей нормальной формы, однако порождает аномалии обновления.

Преподавателю Визгунову на 13 мая выделена аудитория 401. Если по каким-то причинам администрация выделяет преподавателю другую аудиторию, то информация об этом должна быть внесена два раза, что может быть не сделано и, соответственно, стать причиной приведения базы данных в противоречивое состояние.

Выход – разбиение исходного отношения на два:

- РАСПИСАНИЕ КОНСУЛЬТАЦИЙ (Группа, Дата, Время, Преподаватель);
- АУДИТОРИИ (Преподаватель, Дата, Аудитория).

Важно отметить, что хотя такое разбиение (декомпозиция) приводит к устранению избыточности данных, оно также приводит к потере функциональной зависимости: Аудитория, Дата, Время перестают быть потенциальным ключом, так как теперь эти атрибуты находятся в разных отношениях.

Мотивировка нормальной формы Бойса-Кодда

В нормальной форме Бойса-Кодда не существует избыточности и аномалий включения, удаления и модификации. Оказывается, что любая схема отношения может быть приведена в нормальную форму Бойса-Кодда таким образом, чтобы декомпозиция обладала свойством соединения без потерь. Однако схема отношения может быть неприводимой в НФБК с сохранением зависимостей. В этом случае приходится довольствоваться третьей нормальной формой.

4.4.5. Пример нормализации до 3НФ

В настоящее время можно выделить два способа применения нормализации.

Во-первых, можно рассмотреть исходные документы предметной области, выписать все атрибуты, которые в них используются, свести их в одну таблицу и применить процедуру нормализации, основываясь на функциональных зависимостях. Полученное множество отношений будет являться структурой создаваемой базы данных.

Данный подход широко использовался на начальном этапе развития баз данных. Его основным недостатком является большая трудоемкость при выписывании всех функциональных зависимостей. В на-

стоящее время в базах данных хранятся сотни и тысячи таблиц. Применение данного подхода в этом случае становится нерациональным, а иногда и невозможным.

Второй способ, который в настоящее время широко используется на практике, – проверка построенной диаграммы «сущность – связь». Из рассмотренной ранее теории следует, что данные должны храниться в таблицах, находящихся в третьей нормальной форме или в более высоких формах. Каждая сущность будет представлена в базе данных в виде таблицы или представления. Представления, в свою очередь, также будут выбираться из таблиц.

Таким образом, задача сводится к проверке нормализации всех сущностей, отображающихся в таблицы базы данных. Если таблица, получающаяся из некоторой сущности, не является таблицей в третьей нормальной форме, то она должна быть заменена на несколько таблиц, находящихся в третьей нормальной форме.

Рассмотрим пример из предметной области зачисления абитуриентов.

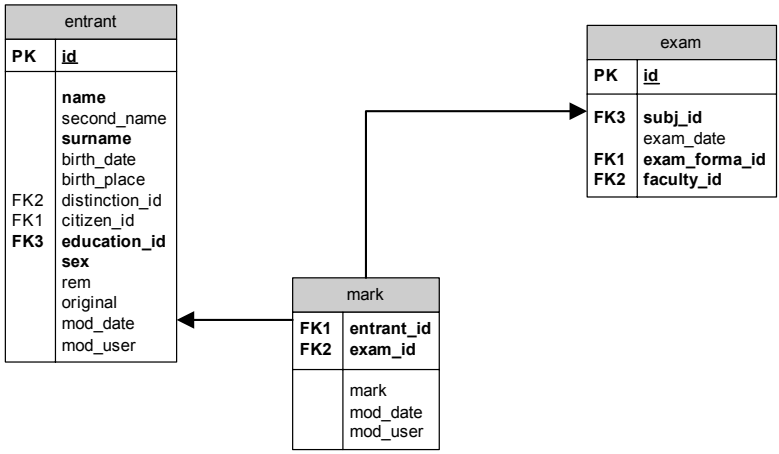

Рис. 31. Фрагмент ER-диаграммы

Таблица, соответствующая сущности entrant, находится в третьей нормальной форме. Действительно, все атрибуты зависят от ключа (от поля id) и эта зависимость полная. Также в таблице отсутствуют транзитивные зависимости.

То же самое можно сказать о сущностях mark и exam. Проверка на соответствие третьей нормальной форме остальной части модели предоставляется в качестве упражнения читателю.

Рассмотрим пример применения первого подхода.

Пусть мы имеем набор ведомостей сданных экзаменов и хотим построить структуру таблиц для хранения этой информации.

Сведем все данные, которые у нас есть, в одну таблицу. Она будет выглядеть примерно так (для упрощения мы не рассматриваем часть атрибутов):

Код абитуриента	Фамилия	Имя	Код экзамена	Предмет и дата	Оценка
1	Сергеев	Сергей	1	Математика 1.08.03	4
2	Иванов	Иван	1		5
1	Сергеев	Сергей	2	Физика 5.08.03	5
2	Иванов	Иван	2		5

Данная таблица ненормализованная, так как на пересечении строки и столбца располагается не одно значение, а несколько.

Перейдем к первой нормальной форме. Для этого разнесем значения предмета и даты в разные столбцы и запишем для каждой строчки информацию по экзамену.

Код абитуриента	Фамилия	Имя	Код экзамена	Предмет	Дата	Оценка
1	Сергеев	Сергей	1	Математика	1.08.03	4
2	Иванов	Иван	1	Математика	1.08.03	5
1	Сергеев	Сергей	2	Физика	5.08.03	5
2	Иванов	Иван	2	Физика	5.08.03	5

Теперь на пересечении любой сроки и любого столбца находится одно значение и, следовательно, данная таблица находится в первой нормальной форме.

Ключом данного отношения будет совокупность атрибутов – Код абитуриента и Код экзамена.

Действительно, эта совокупность атрибутов функционально определяет имя абитуриента, фамилию абитуриента, предмет, дату экзамена и оценку. Ни код абитуриента, ни код экзамена по отдельности не

определяют всех атрибутов. Например, код абитуриента не определяет даты экзамена, а код экзамена не определяет фамилии абитуриента.

В данной таблице существуют неполные зависимости.

Код абитуриента функционально определяет имя и фамилию.

Код экзамена функционально определяет предмет и дату.

Выделим из нашей таблицы новые таблицы, соответствующие неполным зависимостям. Получаем следующие отношения:

- АБИТУРИЕНТЫ (Код абитуриента, Имя, Фамилия). Ключ – Код абитуриента.
- ЭКЗАМЕНЫ (Код экзамена, Предмет, Дата). Ключ – Код экзамена.

  В исходном отношении остаются атрибуты Код абитуриента, Код экзамена, Оценка.
- ОЦЕНКИ (Код абитуриента, Код экзамена, Оценка). Ключ, по-прежнему, – совокупность атрибутов Код абитуриента и Код экзамена.

Полученные отношения находятся в третьей нормальной форме, так как в них нет неполных функциональных зависимостей.

Транзитивных зависимостей в данных отношениях тоже нет, следовательно, они находятся в третьей нормальной форме.

Таким образом, мы провели процесс нормализации для ведомостей экзаменов. В результате были получены три отношения в третьей нормальной форме.

4.4.6. Целостная часть реляционной модели. Реализация условия целостности данных в современных СУБД

Напомним, что под целостностью базы данных понимается то, что в ней содержится полная, непротиворечивая и адекватно отражающая предметную часть (правильная) информация. 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.

Пример внешнего ключа.

СТУДЕНТ (Код студента, ФИО) сдает ЭКЗАМЕН (Код студента, Предмет, Оценка).

Атрибут Код студента сущности ЭКЗАМЕН называется *внешним ключом,* поскольку его значения однозначно характеризуют сущности, представленные кортежами некоторого другого отношения – отношения Студент (мы предполагаем, что поле Код студента является ключом отношения Студент).

Говорят, что отношение, в котором определен внешний ключ, ссылается на соответствующее отношение, в котором такой же атрибут является первичным ключом.

Требование целостности по ссылкам или требование внешнего ключа состоит в том, что для каждого значения внешнего ключа в ссылающемся отношении в отношении, на которое ведет ссылка, должен найтись кортеж с таким же значением первичного ключа либо значение внешнего ключа должно быть неопределенным (т.е. ни на что не указывать).

Ограничения целостности сущности и по ссылкам должны поддерживаться СУБД. Для соблюдения целостности сущности достаточно гарантировать отсутствие в любом отношении кортежей с одним и тем же значением первичного ключа. (В Access для этого предназначена специальная реализация целочисленного поля – поле типа «Счетчик».) С целостностью по ссылкам дела обстоят несколько более сложно.

Понятно, что при обновлении ссылающегося отношения (вставке новых кортежей или модификации значения внешнего ключа в существующих кортежах) достаточно следить за тем, чтобы не появлялись некорректные значения внешнего ключа.

Но как быть при удалении кортежа из отношения, на которое ведет ссылка?

Здесь существуют три подхода, каждый из которых поддерживает целостность по ссылкам. Первый подход заключается в том, что запрещается производить удаление кортежа, на который существуют ссылки (т.е. сначала нужно либо удалить ссылающиеся кортежи, либо

соответствующим образом изменить значения их внешнего ключа). При втором подходе при удалении кортежа, на который имеются ссылки, во всех ссылающихся кортежах значение внешнего ключа автоматически становится неопределенным. Наконец, третий подход (каскадное удаление) состоит в том, что при удалении кортежа из отношения, на которое ведет ссылка, из ссылающегося отношения автоматически удаляются все ссылающиеся кортежи.

В развитых реляционных СУБД обычно можно выбрать способ поддержания целостности по ссылкам для каждой отдельной ситуации определения внешнего ключа. Конечно, для принятия такого решения необходимо анализировать требования конкретной прикладной области.

Заметим, что все современные СУБД поддерживают и целостность сущностей, и целостность по ссылкам, но позволяют пользователям выключать данные ограничения и, таким образом, строить базы данных, не соответствующие реляционной модели. Опыт показывает, что отход от основных положений реляционной модели приводит к краткосрочному выигрышу – алгоритмы становятся проще, но впоследствии серьезно усложняют задачу, особенно ее сопровождение.