Лекция 7

Нормальные формы.
Декомпозиция без потерь

При проектировании базы данных решаются две основные проблемы.

- Проблема логического проектирования баз данных.
  Необходимо отобразить объекты предметной области в абстрактные объекты модели данных таким образом, чтобы это отображение не противоречило семантике предметной области и было, по возможности, лучшим (эффективным, удобным и т. д.).
- Проблема физического проектирования баз данных.
  Необходимо обеспечить эффективность выполнения запросов к базе данных, т. е. расположить данные во внешней памяти, создать дополнительные структуры (например, индексы и т.п.), учитывая особенности конкретной СУБД.

В случае реляционных баз данных трудно предложить какие-либо общие рецепты по части физического проектирования. Здесь слишком многое зависит от используемой СУБД. Поэтому ограничимся вопросами логического проектирования реляционных баз данных, которые существенны при использовании любой реляционной СУБД.

Будем считать, что проблема проектирования реляционной базы данных состоит в обоснованном принятии решений о том, из каких отношений должна состоять БД и какие атрибуты должны быть у этих отношений.

Декомпозиция без потерь и функциональные зависимости

Рассмотрим подход к *проектированию реляционных БД* на основе нормализации.

Процедура нормализации основывается на декомпозиции исходной переменной-отношения на другие переменные-отношения. Эта декомпозиция позволяет более эффективное выполнение операций обновления базы данных, поскольку сокращается число проверок и вспомогательных действий, поддерживающих целостность БД.

Процесс декомпозиции заключается в замене данной переменной-отношения некоторым набором ее проекций. Декомпозиция должна быть обратимой, т. е. должна имеется возможность собрать исходное отношение из декомпозированных отношений без потери информации (**декомпозиция без потерь**).

Корректные и некорректные декомпозиции отношений.

Теорема Хита

Пример 7.1 Дана переменная-отношение S с заголовком {S#, STATUS, CITY} (атрибут SNAME для простоты исключили)

S поставщики		
S#	STATUS	CITY
S_1	20	London
S_2	30	Paris
S_4	20	London
S_5	30	Athens

Таблица 7.1 Отношение S

Рассмотрим два возможных варианта декомпозиции отношения S.

SSt	
S#	STATUS
S_1	20
S_2	30
S_4	20
S_5	30

SC	
S#	CITY
S_1	London
S_2	Paris
S_4	London
S_5	Athens

а)

SSt	
S#	STATUS
S_1	20
S_2	30
S_4	20
S_5	30

StC	
STATUS	CITY
20	London
30	Paris
20	London
30	Athens

б)

Таблица 7.2. Декомпозиции отношения S.

При проведении декомпозиции мы использовали операцию взятия проекции. Каждое из отношений SSt и SC является проекцией исходного отношения S

В случае декомпозиции (а) информация не утрачивается, из отношений SSt и SC можно узнать, что поставщик с номером S_2 имеет статус 30 и находится в Париже, а поставщик с номером S_5 имеет статус 30 и находится в Афинах.

Вторая декомпозиция (б) не дает возможности получить данные о местонахождении поставщиков S_2 и S_5, оба поставщика имеют одинаковый статус, и нельзя сказать какой из них находится в Париже, а какой в Афинах. Следовательно, эта декомпозиция приводит к потере информации.

В случае декомпозиции (а) отсутствие потери информации означает, что в результате естественного соединения отношений SSt и SC мы гарантированно получим отношение, заголовок и тело которого совпадают с заголовком и телом отношения S.

Это произойдет для любых допустимых (и согласованных) значений переменных отношений S, SSt и SC, поскольку у всех этих переменных атрибут S# является возможным ключом. В исходном отношении S существует неприводимое множество функциональных зависимостей {S#→STATUS, S#→CITY}. В отношениях SSt и SC также существуют функциональные зависимости S#→STATUS и S#→CITY соответственно.

В случае (б) при обратном соединении переменных-отношений SSt и StC исходная переменная-отношение S получена не будет, т.е. будет утрачена информация. Это произойдет потому, что в исходном отношении нет функциональной зависимости STATUS→CITY (и в отношении StC тоже).

«Обратимость» означает, что исходная переменная-отношение равна соединению ее проекций.

Теорема Хита.

Пусть задано отношение R {A, B, C} , где A, B и C, — множество атрибутов этого отношения. Если R удовлетворяет ФЗ A→B, тогда R равно соединению ее проекций PROJECT[A,B] и PROJECT[A,C]. R = (R PROJECT[A,B]) NATURAL JOIN (R PROJECT[A,C]).

Доказательство.

Пусть R1 — результат естественного соединения R PROJECT[A, B] и R PROJECT[A,C]. Докажем, что множество кортежей отношения R1 равно множеству кортежей отношения R. (A=B ⇔ A⊆B & B⊆A ;A⊆B ⇔ ∀a∈A ⇒ a∈B).

1. Докажем, что в теле R1 содержатся все кортежи тела отношения R (*R⊆R1*).
Пусть произвольный кортеж (a, b, c) ∈R. Тогда по определению операции взятия проекции (a,b)∈(R PROJECT [A, B]) и (a, c) ∈ (R PROJECT [A, C]). Следовательно, по определению естественного соединения (a, b, c)∈R1

Естественным соединением отношений A с заголовком {X,Y} и B с заголовком {Y,Z} называется отношение с заголовком {X, Y, Z} и телом, содержащим множество всех кортежей вида (x,y,z), таких, для которых в отношении A значение атрибута X равно x, значение атрибута Y равно y и в отношении B значение атрибута Y равно y, а значение атрибута Z равно *z.*

2. Докажем, что в теле R содержатся все кортежи тела отношения R1 (*R1⊆R*). Докажем от противного. Пусть в R1 появился набор (a,b,c'), которого не было в исходном отношении. Это означает, что в проекции на [A,C] есть пара (a,c'),и существует такой элемент (или элементы) b^*, что кортеж (a,b^*,c') присутствуют в исходном отношении (условие (a,c')∈(R PROJECT[A,C]) выполняется только в этом случае) следовательно пара (a,b^*)∈(R PROJECT[A, B]). Т.к. (a,b^*)∈(R PROJECT [A,B]) и (a,b)∈(R PROJECT[A,B]) ((a,b,c')∈R1 ⇒ (a,b)∈(R PROJECT[A, B])) и выполняется ФЗ A→B, то a=a ⇒ b^*=b, и следовательно (a,b^*,c')=(a,b,c').

Пусть в R1 появился набор (a,b',c) которого не было в исходном отношении. Это означает, что в проекции на [A,C] есть пара (a,c),и существует такой b, что кортеж (a,b,c) присутствуют в исходном отношении. Следовательно, пара (a,b)∈(R PROJECT[A, B]). Т.к. (a,b)∈(R PROJECT[A,B]) и (a,b')∈(R PROJECT[A,B]) и выполняется ФЗ A→B, то b'=b, и следовательно (a,b,c)=(a,b',c). **#**

Рассмотрим отношение Assignment_Department (clientID, employeelD, DepartmentID) (Таблица 7.3 (а)). Атрибут clientID содержит номера клиентов, с которыми работают служащие, employeelD – номера служащие которые работают с клиентами, DepartmentID. содержит номера отделов,в которых работают служащие. Каждый слу-

жащий работает только в одном отделе, т. е. имеется ФЗ employeeID→DepartmentID, но один служащий может работать с несколькими клиентами.

Assignment_Department		
clientID	employeeID	DepartmentID
1	2211	12
2	2213	10
3	2211	12

(а) Исходное отношение

ED	
employeeID	DepartmentID
2211	12
2213	10

CE	
clientID	employeeID
1	2211
2	2213
3	2211

(б) Декомпозиция без потерь по теореме Хита

Таблица 7.3.

В отношении Assignment_Department атрибут employeeID не является возможным ключом, показано наличия *ФЗ* employeeID→DepartmentID оказывается достаточно для *декомпозиции этого отношения без потерь*.

Определение. Атрибут B неприводимо (минимально) зависит от атрибута A*,* если выполняется неприводимая (минимальная) слева *ФЗ* A→B.

ФЗ называется **неприводимой слева,** если левая часть (детерминант) каждой функциональной зависимости из множества S является неприводимой, т.е. ни один атрибут из детерминанта не может быть опущен без изменения замыкания S^+.

Например, в отношении СТУДЕНТ(StudentID, Name, Groupe, Speciality) выполняются ФЗ StudentID →Groupe и {StudentID, Name}→ Groupe. Первая ФЗ является неприводимой (минимальной) слева, а вторая — нет. Поэтому Groupe неприводимо (минимально) зависит от StudentID, а для {StudentID, Name} свойство неприводимой зависимости не выполняется.

Диаграммы функциональных зависимостей

Пусть дана переменная отношение R и пусть к ней применимо некоторое неприводимое множество ФЗ I. Удобно представить это множество ФЗ в виде диаграммы *функциональных зависимостей (диаграммы ФЗ)*.

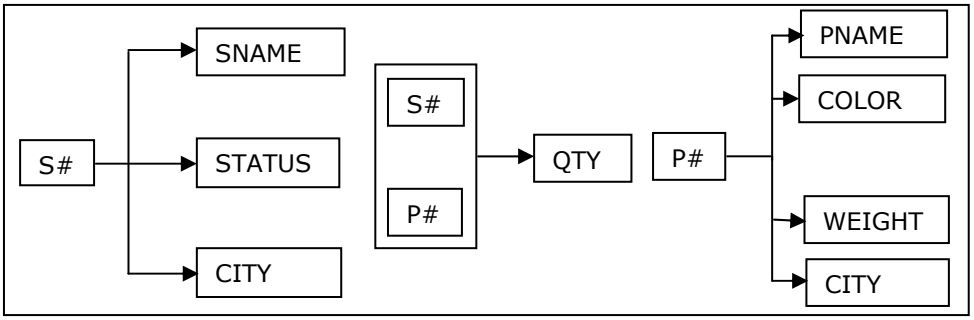

Рисунок 7.1. Диаграмма ФЗ для переменных отношения S#, P# и SP#.

На рисунке 7.1 все стрелки начинаются с первичного ключа, поэтому на данной диаграмме никакие стрелки не могут быть удалены. В общем случае, в множество ФЗ могут входить ФЗ, в которых детерминантом является не потенциальный ключ отношения, т.е. ФЗ атрибутов не являются неприводимыми и стрелки на диаграмме не начинаются с возможного ключа.

Процедуру нормализации можно неформально представить как процедуру удаления стрелок, не начинающихся с потенциальных ключей.

Нормальные формы.

Процесс проектирования БД с использованием метода нормальных форм является итерационным и заключается в последовательном переводе отношений из первой нормальной формы в нормальные формы более высокого порядка по определенным прави-

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

Исходной точкой является представление предметной области в виде одного или нескольких отношений, и на каждом шаге проектирования производится некоторый набор схем отношений, обладающих «улучшенными» свойствами. Процесс проектирования представляет собой процесс нормализации схем отношений, причем каждая следующая нормальная форма обладает свойствами «лучшими», чем предыдущая.

Каждой нормальной форме соответствует определенный набор ограничений, и отношение находится в некоторой нормальной форме, если удовлетворяет свойственному ей набору ограничений.

В теории реляционных баз данных обычно выделяется следующая последовательность нормальных форм:
первая нормальная форма (1NF)*;*
вторая нормальная форма (2NF)*;*
третья нормальная форма (3NF)*;*
нормальная форма Бойса-Кодда (BCNF)*;*
четвертая нормальная форма (4NF);
пятая нормальная форма, или нормальная форма проекции-соединения (5NF или PJ/NF).

Каждая последующая нормальная форма наследует свойства предыдущей и удовлетворяет некоторым дополнительным требованиям.
В основе процесса проектирования лежит метод нормализации, т. е. декомпозиции отношения, находящегося в предыдущей нормальной форме, на два или более отношений, которые удовлетворяют требованиям следующей нормальной формы.

Первая нормальная форма

Таблица находится в первой **нормальной форме**, если каждый её атрибут атомарен и все кортежи различны.

Выражение "атрибут атомарен" означает, что, в любом допустимом значении переменной-отношения каждый ее кортеж содержит только одно значение для каждого из атрибутов.

Например, не соответствуют 1НФ таблицы, в полях которых могут храниться списки значений.

Требование *первой нормальной формы* является базовым требованием классической реляционной модели данных. Будем считать, что исходный набор отношений уже соответствует этому требованию.

Неприводимые ФЗ и вторая нормальная форма

Множество ФЗ S называется **неприводимым** тогда и только тогда, когда оно обладает следующими свойствами.

1. Правая (зависимая) часть каждой ФЗ из множества S содержит только один атрибут (т.е. является одноэлементным множеством).
2. ФЗ является **неприводимой слева.**
3. Ни одна ФЗ из множества S не может быть удалена из множества S без изменения его замыкания S^+.

Пусть задана переменная-отношение FIRST с атрибутами {S#,STATUS, CITY, P#, QTY}
Пусть атрибуты {S#, P#} — первичный ключ. Переменная-отношение FIRST находится в первой нормальной форме (1НФ), т.е. каждый атрибут атомарен.

Диаграмма множества ФЗ показана на рис. 7.2, возможное тело отношения - в табл.7.4.

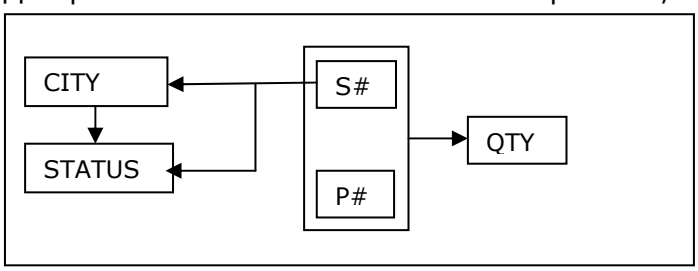

Рис. 7.2. Диаграмма множества ФЗ переменной-отношения FIRST.

FIRST				
S#	STATUS	CITY	P#	QTY
S_1	20	London	P_1	300
S_1	20	London	P_2	200
S_1	20	London	P_3	400
S_1	20	London	P_4	200
S_1	20	London	P_5	100
S_1	20	London	P_6	100
S_2	10	Paris	P_1	300
S_2	10	Paris	P_2	400
S_3	10	Paris	P_2	200
S_4	20	London	P_2	200
S_4	20	London	P_4	300
S_4	20	London	P_5	400

Таблица 7.4 Возможное тело значения переменной-отношения FIRST

Аномалии обновления, возникающие из-за наличия ФЗ, не являющихся неприводимыми

Во множество ФЗ отношения FIRST входят ФЗ, в которых детерминантом является не возможный ключ отношения. В диаграмме есть стрелки, начинающиеся не с {S#, P#}, т. е. некоторые ФЗ атрибутов от возможного ключа не являются неприводимыми.

В табл. 7.4 видно избыточное дублирование данных или избыточность данных. В каждом кортеже для поставщика с номером S_1 атрибут CITY имеет значение London и атрибут STATUS имеет значение 20.

Избыточность переменной отношения приводит к аномалиям обновления. Под аномалиями обновления понимаются трудности, с которыми приходится сталкиваться при выполнении операций добавления кортежей в отношение (INSERT), удаления кортежей (DELETE) и модификации кортежей (UPDATE). Рассмотрим сначала аномалии обновления, вызываемые наличием ФЗ S#→CITY. Эти аномалии связаны с избыточностью хранения значений атрибутов CITY и STATUS в каждом кортеже отношения.

Добавление кортежей (операция INSERT). Нельзя поместить в переменную- отношение FIRST информацию о том, что некий поставщик находится в определенном городе, не указав сведений хотя бы об одной детали, им поставляемой.

Удаление кортежей (операция DELETE). Если из переменной-отношения FIRST удалить данные о единственной поставке некоторого поставщика, то будет утеряна информация о данном поставщике вообще (т.е. информация о городе, где расположен поставщик и о его статусе).

Модификация кортежей (операция UPDATE). Название города для каждого поставщика повторяется несколько раз в переменной- отношения FIRST, поэтому чтобы изменить значение CITY, например, у поставщика S_1 вместо значения London записать значение Rome, необходимо найти в переменной- отношения FIRST все кортежи, где значение атрибута S#=S_1 и заменить значение London атрибута CITY на Rome. Если в переменной- отношении FIRST заменить не все значения London атрибута CITY на Rome, то база данных окажется в противоречивом состоянии.

Для преодоления этих трудностей можно произвести декомпозицию отношения FIRST на два отношения — SECOND{S#, STATUS, CITY} и SP{S#, P#, QTY}. На основании теоремы Хита эта декомпозиция является декомпозицией без потерь, поскольку в исходном отношении имелась ФЗ{S#, P#}→QTY.

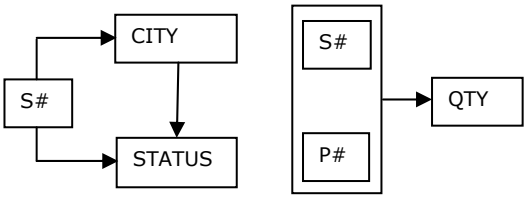

Рис. 7.3**.** Диаграммы множеств ФЗ в переменных- отношениях SECOND и SP.

SECOND		
S#	STATUS	CITY
S_1	20	London
S_2	10	Paris
S_3	10	Paris
S_4	20	London

SP		
S#	P#	QTY
S_1	P_1	300
S_1	P_2	200
S_1	P_3	400
S_1	P_4	200
S_1	P_5	100
S_1	P_6	100
S_2	P_1	300
S_2	P_2	400
S_3	P_2	200
S_4	P_2	200
S_4	P_4	300
S_4	P_5	400

Таблица 7.4 Возможные тела значений переменных- отношений SECOND и SP.

Измененная структура данных позволяет преодолеть все трудности, связанные с операциями с операциями обновления.

Добавление кортежей (операция INSERT).В отношение SECOND можно поместить информацию о поставщиках, которые не производят поставок деталей в настоящее время. Например, добавить в отношение SECOND информацию о поставщике с номером S_5, находящимся в Афинах и имеющим статус 30.

Удаление кортежей (операция DELETE). Если кто-то из поставщиков, поставляющих только один тип деталей, прекращает их поставлять, кортеж о поставке можно убрать из отношения SP. Информация о самом поставщике (номер, статус, город) не утратится. Уберем запись о поставке детали с номером P_2 из переменный - отношения SP.

Модификация кортежей(операция UPDATE). Если у какого- либо поставщика изменилось его месторасположение, достаточно модифицировать один кортеж в отношении SECOND. Пусть поставщик S_1 переместился из Лондона в Рим. Достаточно один раз изменить значение атрибута CITY в отношении SECOND.

SECOND		
S#	STATUS	CITY
S_1	20	London
S_2	10	Paris
S_3	10	Paris
S_4	20	London
S_5	30	Athens

Таблица 7.5 Тело значения переменной-отношения SECOND после операции INSERT

SP		
S#	P#	QTY
S_1	P1	300
S_1	P3	400
S_1	P4	200
S_1	P5	100
S_1	P6	100
S_2	P1	300
S_4	P4	300
S_4	P5	400

Таблица 7.6 Тело значения переменной-отношения SP после операции DELETE

SECOND		
S#	STATUS	CITY
S_1	20	Rome
S_2	10	Paris
S_3	10	Paris
S_4	20	London
S_5	30	Athens

Таблица 7.7 Тело значения переменной-отношения SECOND после операции UPDATE.

Вторая нормальная форма

Суть произведенной операции декомпозиции переменной-отношения FIRST на переменные-отношения SECOND и SP состояла в исключении зависимостей, которые не являлись неприводимыми.

Переменная отношения находится во ***второй нормальной форме (2NF)*** тогда и только тогда, когда она находится в *первой нормальной форме*, и каждый неключевой атрибут неприводимо функционально зависит от первичного ключа.

Переменные-отношенийя SECOND и SP находятся в *2*НФ, все неключевые атрибуты отношений минимально зависят от первичных ключей S# и {S#, P#} соответственно. Переменная-отношение FIRST не находится в 2НФ (например, ФЗ {S#, P#}→CITY не является неприводимой). Любая переменная-отношение, находящаяся в 1НФ, но не находящаяся в 2 НФ, может быть приведена к набору переменных отношений, находящихся в 2НФ. В результате декомпозиции мы получаем набор проекций исходной переменной отношения, естественное соединение значений которых воспроизводит значение исходной переменной отношения (т. е. это декомпозиция без потерь). Для переменных отношений SECOND и SP исходное отношение FIRST воспроизводится их естественным соединением по общему атрибуту S#.

Заметим, что допустимое значение переменной отношения SECOND может содержать кортежи, информационное наполнение которых выходит за пределы допустимых значений переменной отношения FIRST. Например, в теле отношения SECOND может находиться кортеж с данными о поставщике с номером S_5, который еще не участвует ни в одном поставке. Наличие такого кортежа не влияет на результат естественного соединения, тело которого все равно будет совпадать с телом допустимого значения переменной отношения FIRST.

Таким образом, первый этап процедуры нормализации состоит в создании проекций, которые позволяют исключить функциональные зависимости, не являющиеся неприводимыми.
Пусть дана переменная-отношение R, имеющая следующий вид.

R{A, B, C, D}
PRIMARY KEY{A,B} *(т.е. {A,B}→ C, {A,B}→ D)*

Предполагается ФЗ A→D

Процедура нормализации предусматривает замену переменной-отношения R следующими двумя проекциями R1 и R2.

R{A, D}
PRIMARY KEY{A}
R{A, B, C}
PRIMARY KEY{A,B}
FORIGN KEY {A} REFERENCES R1

Переменная- отношение R всегда может быть восстановлена посредством соединения переменных-отношений R1 и R2 по внешнему и соответствующему ему первичному ключу этих переменных- отношений.

Нетранзитивные ФЗ и третья нормальная форма

В произведенной декомпозиции переменной отношения FIRST множество ФЗ переменной отношения SP задан предельно просто – в единственной нетривиальной ФЗ детерминантом является возможный ключ {S#, P#}. При использовании этой переменной отношения какие-либо *аномалии обновления* не возникают. Однако выбранная структура переменной- отношения SECOND может вызвать некоторые проблемы.

Аномалии обновлений, возникающие из-за наличия транзитивных ФЗ

ФЗ переменной отношения SECOND по-прежнему порождают некоторые *аномалии обновления*. Они вызываются наличием **транзитивной** ФЗ S#→STATUS, через ФЗ S#→CITY и CITY→ STATUS.

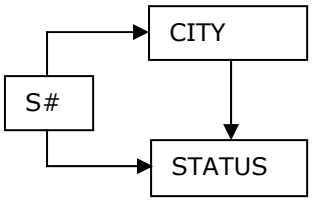

Рис. 7.4**.** Диаграмма ФЗ в переменной-отношения SECOND.

Эти *аномалии* связаны с избыточностью данных типа «город-статус», отвечающей ФЗ CITY→ STATUS.

- **Добавление кортежей** (операция INSERT). Нельзя поместить в базу данных сведения об определенном городе, обладающим некоторым статусом, до тех пор, пока в этом городе не появится конкретный поставщик. (Первичный ключ не может содержать неопределенные значения.)
- **Удаление кортежей** (операция DELETE). При удалении из переменной-отношения SECOND кортежа с данными о единственном поставщике некоторого города, будет утеряна информация о том каким статусе обладал данным город. Например, удалении из переменной-отношения SECOND кортежа (S_5, 30, Athens) будет утрачена информация о статусе Афин.
- **Модификация кортежей** (операция UPDATE). В переменной-отношения SECOND информация о статусе города повторяется несколько раз, т.е. некоторая избыточность данных все еще остается. Для изменения статус определенного города надо изменить значения атрибута STATUS во всех кортежах для всех поставщиков расположенных в данном городе.

Возможная декомпозиция

Для преодоления этих трудностей произведем декомпозицию переменной-отношения SECOND на две переменные- отношения – SC{S#, CITY} и CS{CITY, STATUS}. По теореме Хита, это снова декомпозиция без потерь по причине наличия ФЗ S#→CITY и CITY→STATUS. На рис. 7.5 показаны диаграммы ФЗ этих переменных отношений, а на таблице 7.8 (операция декомпозиции применялась к таблица 7.5) – их возможные значения.

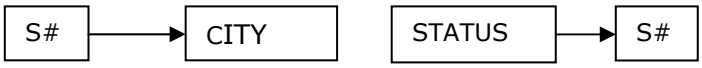

Рис. 7.5. Диаграммы ФЗ в отношениях SC и CS.

SC	
S#	CITY
S1	London
S2	Paris
S3	Paris
S4	London
S5	Athens

CS`	
CITY	STATUS
London	20
Paris	10
Athens	30

Таблица 7.8. Возможные значения переменных- отношений SC и CS.

Данное преобразование обратимо, т.к. переменная-отношения SECOND является естественным соединением значений отношений SC и CS. Такое изменение структуры переменных- отношений позволяет устранить проблемы, возникающие при выполнении операций обновления.

- **Добавление кортежей.** Чтобы сохранить информацию о статусе нового города достаточно добавить соответствующий кортеж к отношению CS.
- **Удаление кортежей.** При увольнении единственного поставщика из данного города, удаляется соответствующий кортеж из отношения SC, данные о статусе города остаются в отношении CS.
- **Модификация кортежей.** При изменении статуса города, изменяется значение атрибута STATUS ровно в одном кортеже отношения CS.

Проблемы, возникающие при выполнении операций обновления, были связаны с наличием транзитивной функциональной зависимости. Наличие этой ФЗ не означало, что

атрибут STATUS характеризовал не сущность, которая идентифицировалась первичным ключом S# (номер поставщика), а сущность город CITY. Смешивание этих двух типов информации приводило к возникновению проблем.

Третья нормальная форма

Переменная отношения находится в ***третьей нормальной форме (3НФ)*** тогда и только тогда, когда она находится во *второй нормальной форме*, и каждый неключевой атрибут нетранзитивно функционально зависит от первичного ключа.

Переменные- отношения SC и CS находятся в *3НФ*, все неключевые атрибуты нетранзитивно зависят от первичных ключей S# и CITY соответственно. Переменная- отношение SECOND не находится в *3НФ* (ФЗ S#→STATUS является транзитивной). Любое отношение, находящееся в *2 НФ*, но не находящееся в *3 НФ*, может быть приведено к набору отношений, находящихся в *3 НФ*. При этом получается набор проекций исходного отношения, естественное соединение которых воспроизводит исходное отношение (т. е. это декомпозиция без потерь). Для отношений SC и CS исходное отношение SECOND воспроизводится их естественным соединением по общему атрибуту CITY.

Заметим, что допустимые значения отношения CS могут содержать кортежи, информационное наполнение которых выходит за пределы тела отношения SECOND. Например, в теле отношения CS может находиться кортеж с данными о городе Вена, в котором нет еще поставщиков деталей. Наличие такого кортежа не влияет на результат естественного соединения, который все равно будет являться допустимым значением отношения SECOND.

Второй этап нормализации состоит в создании проекций для исключения транзитивных зависимостей.
Пусть дана переменная-отношение R с атрибутами A, B, C.

R {A, B, C}
PRIMARY KEY{A}
Имеется ФЗ B→C

Процедура нормализации предусматривает декомпозицию этой переменной отношения на переменные-отношения R1 и R2.

R1 {B, C}
PRIMARY KEY{B}
R2 {A, B}
PRIMARY KEY{A}
FORIGN KEY {B} REFERENCES R1

Переменная- отношение R всегда может быть восстановлена посредством соединения переменных-отношений R1 и R2 по внешнему и соответствующему ему первичному ключу этих переменных- отношений.

Уровень нормализации переменной-отношения определяется семантикой данных, а не их конкретными реализациями в некоторое время. Для того, чтобы однозначно определить находится ли данная переменная-отношение в 3НФ необходимо представлять себе сущность этих данных, т.е. существующие между ними зависимости.

Сохранение зависимостей.

Независимые проекции отношений. Теорема Риссанена

В процессе нормализации часто возникает ситуация, когда переменная-отношение может быть подвергнута декомпозиции без потерь несколькими разными способами. Рассмотрим переменную-отношение SECOND с ФЗ S#→CITY и CITY→STATUS, и транзитивной зависимостью S#→STATUS (на рис. 7.6 эта транзитивная зависимость показана пунктирной стрелкой).

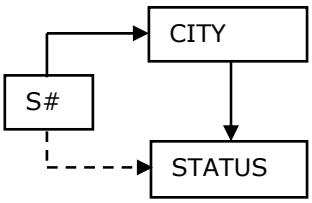

Рисунок 7.6 переменная-отношение SECOND

Аномалии обновления, сопровождающие переменную-отношение SECOND можно преодолеть посредством ее декомпозиции с последующей заменой двумя проекциями в ЗНФ. Существуют три варианта декомпозиции

Вариант 1. Декомпозиция **A**
SC{S#, CITY}
CS{CITY, STATUS}
Вариант 2. декомпозиция **B**.
SC {S#, CITY}
SS {S#, STATUS}
Вариант 3. декомпозиция **C**.
{S#, STATUS}
{CITY, STATUS}
Третий вариант декомпозиции переменной-отношения SECOND не является допустимой декомпозицией, поскольку сопровождается потерей информации.

SSt	
S#	STATUS
S_1	20
S_2	10
S_3	10
S_4	20
S_5	30

CSt	
CITY	STATUS
Rome	20
Paris	10
Paris	10
London	20
Athens	30

Переменная- отношение SECOND не является естественным соединением значений отношений SSt и CSt по атрибуту STATUS. В переменной- отношении SECOND существует ФЗ CITY→STATUS, но нет ФЗ STATUS→CITY, т.е. одному значению атрибута STATUS могут соответствовать несколько значений атрибута CITY. Например, из таблицы 7.9 видно, что значение атрибута STATUS равное 20 соответствует двум городам.

Таблица 7.9 (Тело переменной-отношения SECOND— из таблицы 7.7)

Рассмотрим подробнее варианты декомпозиции **A** и **B**.
Проекции SC одинаковы и для варианта **A**, и для варианта **B**. Декомпозиция **A** выполняется без потери информации (мы подробно ее рассматривали при определении ЗНФ). Декомпозиция **B** также выполняется без потери информации, и обе ее проекции находятся в ЗНФ. Однако декомпозиция **B** в отличие от декомпозиции **A**, не устраняет проблемы, связанные с обновлением отношения. Например, после выполнения декомпозиции **B** по-прежнему будет невозможно ввести информацию о том, что некоторый город имеет определенный статус, не указав конкретного поставщика из этого города.
В декомпозиции **A** обе проекции независимы одна от другой в том смысле, что обновления в каждой из них могут выполняться совершенно независимо. Если гарантируется, что выполняемые обновления будут допустимы, т.е. уникальность первичного ключа данной проекции не нарушается, то *соединение этих двух проекций после обновления всегда будет иметь результатом допустимое значение переменной-отношения* SECOND. Т.е. при соединении не будут нарушаться ограничения, наложенные на функциональные зависимости в переменной-отношении SECOND.
Однако в случае декомпозиции **B** вносимые в любую из двух проекций обновления должны тщательно контролироваться, чтобы исключить возможные нарушения функциональной зависимости CITY→STATUS. Нарушения могут иметь место, если два и более поставщиков находятся в одном и том же городе; в этом случае они должны иметь *один* статус. например, в декомпозиции **B** поставщик с номером 'S_1' перемещается из Лондона в Париж. Иначе говоря, две проекции декомпозиции B не являются независимыми одна от другой. Заметим, что зависимости, использованные для создания проекций в декомпозиции A, соответствуют *сплошными* стрелкам (см. рис. 7.6), тогда как одна из зависимостей, использованная для создания проекций в декомпозиции B, отмечена *пунктирной* стрелкой.
Основная проблема заключается в том, что в декомпозиции **B** ФЗ CITY→STATUS превращается в ***ограничение базы данных***, охватывающее две переменные-отношения. В декомпозиции **A** ограничением базы данных является *транзитивная* зависимость S#→STATUS, которая автоматически выполняется в случае выполнения двух ограничений *переменных-отношений:* S#→ CITY и CITY→STATUS. Реализовать эти ограничения очень просто, поскольку по сути они представляют собой требования поддержки уникальности значений первичных ключей в соответствующих переменных-отношениях.

Таким образом, концепция независимых проекций предоставляет критерий выбора одного из возможных вариантов декомпозиции. В частности, вариант декомпозиции,

обеспечивающий *независимость* проекций в приведенном выше смысле, в общем случае предпочтительнее вариантов, в которых проекции будут зависимы.

Теорема Риссанена

Проекции R1 и R2 переменной-отношения R будут **независимы** тогда и только тогда, когда:

- каждая ФЗ в переменной-отношении R является логическим следствием ФЗ в ее проекциях R1 и R2;
- общие атрибуты проекций R1 и R2 образуют потенциальный ключ по хотя бы для одной из этих проекций.♦ (Без доказательства.)

Рассмотрим заданные выше декомпозиции **A** и **B**. В декомпозиции **A** обе проекции независимы, поскольку их общий атрибут CITY является первичным ключом для переменной-отношения CS и каждая ФЗ переменной-отношения SECOND либо представлена в одной из проекций, либо является логическим следствием других имеющихся в них ФЗ. В декомпозиции **B**, наоборот, две составляющие ее проекции не являются независимыми, поскольку ФЗ CITY→STATUS не может быть выведена из ФЗ, существующих в этих проекциях, даже несмотря на то, что их общий атрибут S# является потенциальным ключом для обеих проекций.

Определение Переменная-отношение называется **атомарной**, если она не может быть подвергнута декомпозиции с получением независимых проекций.

Это не означает, что каждую неатомарную переменную-отношение следует непременно разбить на атомарные компоненты. Например, переменные-отношения S и P из базы данных поставщиков и деталей не являются атомарными, однако дальнейшая их декомпозиция имела бы мало смысла. Переменная-отношение SP, наоборот, *является* атомарной.

Требование сохранения зависимостей. Нормализация всегда должна предусматривать декомпозицию переменных-отношений на независимые проекции.

Замечания.

1. Пусть дана переменная-отношение R, которая после выполнения всех этапов нормализации заменяется множеством переменных-отношений R1, R2,..., Rn (проекциями переменной-отношения R).
2. Пусть также задано множество ФЗ S, имеющих место в исходной переменной-отношении R, и множество функциональных зависимостей S1, S2, ..., Sn, выполняющихся в переменных-отношениях R1, R2, ..., Rn.
3. Каждая функциональная зависимость в множестве Si будет иметь отношение только к атрибутам проекции Ri (где i=1, 2, 3, ..., n). В результате реализация ограничений (устанавливаемых существующими ФЗ) для любого данного множества Si представляется достаточно простой задачей. Однако в действительности необходимо реализовать все ограничения, определяемые исходным множеством ФЗ S. Следовательно, целесообразно выбрать такой вариант декомпозиции исходной переменной-отношения на проекции R1, R2, ..., Rn, при котором совместный эффект от реализации ограничений для отдельных множеств S1, S2, ..., Sn будет эквивалентен реализации всех ограничений для исходного множества функциональных зависимостей S. Иначе говоря, декомпозиция должна выполняться с сохранением зависимостей.
4. Пусть S' является объединением множеств зависимостей S1, S2, ..., Sn. В общем случае равенство S'=S не выполняется. Для декомпозиции с сохранением зависимостей достаточно, чтобы были равны замыкания множеств S и S' ($S^+ = S'^+$).

В общем случае не существует эффективного метода вычисления замыкания S^+ для заданного множества ФЗ, поэтому на практике вычисление и сравнение двух необходимых замыканий осуществить сложно.