4.2.5. Теория реляционных БД и классическая методика проектирования реляционных схем БД

Задачу проектирования реляционной базы данных (РБД) Дейт сформулировал следующим образом: «выбрать подходящую логическую структуру для заданного массива данных, которые требуется поместить в базу данных. Иначе говоря, нужно решить вопрос, какие необходимы базовые отношения и какой набор атрибутов они должны включать». Данное определение нуждается в дополнении. Хотя упомянутые структурные аспекты играют существенную роль для БД, другой важной задачей является обеспечение целостности данных, которая решается введением в схему БД ограничений целостности.

Таким образом, логическое проектирование реляционной базы данных для некоторой предметной области заключается в создании реляционной схемы БД, включающей определения базовых отношений и ограничений целостности, отражающих семантику предметной области.

В этой и последующей главах будет представлен ряд методик проектирования РБД. Каждая методика представляет собой структурированный подход, предусматривающий использование специализированных процедур, технических приемов, инструментов, документации и нацеленный на поддержку и упрощение процесса проектирования. В этом пункте кратко описан классический реляционный подход, основанный на процедуре нормализации универсального отношения, особое внимание уделяется его проблемам, связанным с ограниченностью использования, чрезмерной сложностью, наличием нерешенных вопросов и противоречий. В следующей главе будут представлены так называемые семантические методики моделирования с использованием ER-модели, EER-модели и ERM-модели, которые могут рассматриваться как процедуры получения проекта предварительных отношений.

Классическая методика признается ее апологетами единственной строгой теорией проектирования реляционных схем БД (известны даже попытки распространения ее идей на проектирование ER-схем). Хотя и эти авторы признаются в сложности ее понятий, возможности конфликтных ситуаций, неоднозначности результата и наличии других нерешенных проблем. Это дает повод для следующих утверждений.

«В моделировании данных используется ряд достаточно сильных предположений, которые делают теорию (часто называемую теорией баз данных) весьма интересной, но непосредственно неприменимой на практике. Вместе с тем теория баз данных позволяет глубже понять модели данных, схемы баз данных и их свойства, и в этом смысле ее в первую очередь следовало бы рассматривать как средство понимания, а не как средство проектирования. Даже наиболее «благополучные» формальные методы проектирования БД при ближайшем рассмотрении обнаруживают пробелы, которые можно восполнить только интуицией проектировщика».

Таким образом, многие считают, что теория БД пока далека от совершенства, а «проектирование БД – это скорее искусство, чем наука».

Классический подход к проектированию реляционных схем БД можно использовать как автономно, так и в сочетании с той или иной методикой семантического моделирования, задачей которой является получение набора предварительных отношений. В первом случае задается множество зависимостей между данными и неявно схема универсального отношения U. Требуется получить декомпозицию (разбиение) универсального отношения – набор отношений, обеспечивающих ту же информативность, но обладающих лучшими эксплуатационными характеристиками. Во втором случае нам дан результат декомпозиции универсального отношения – множество отношений R_1, …, R_n и множество зависимостей, а задача состоит в том, чтобы определить, эффективна ли декомпозиция. Оценка эффективности базируется на тех же критериях, что и задача корректной декомпозиции U.

Предположим, нам дана схема БД, описанная в терминах схем отношений, атрибутов и зависимостей, и мы желаем установить, является ли эта схема «хорошей». Говоря конкретней, мы хотели бы иметь гарантии адекватности представления схемой всех известных нам зависимостей или, по крайней мере, всех зависимостей, которые мы желали бы представить. Необходимо также устранить избыточность схемы. Кроме того, хотелось бы надеяться, что различные информационные единицы схемы сформированы правильно (т.е. атрибуты правильно кластеризованы). Ответы на эти вопросы и призвана дать теория БД.

Отметим, что при использовании классической методики семантика реального мира описывается в терминах атрибутов (т.е. свойств объектов), собранных в одно «бессмысленное» универсальное отношение, и неких зависимостей между ними (мы ограничимся рассмотрением лишь функциональных зависимостей – наиболее простой формы зависимостей), смысл которых с трудом поддается пониманию даже профессионалами. Обратите внимание на то, как далеко подобное восприятие реального мира от традиционной онтологической картины мира, включающей объекты, их свойства и связи между ними.

Прежде чем приступать к рассмотрению классической методики и сопутствующей ей теории реляционных БД, имеет смысл наметить некоторые цели проектирования. Среди множества целей, стоящих перед проектированием, следующие представляются наиболее важными.

1. Возможность хранения всех необходимых данных в БД.

Эта цель кажется очевидной, тем не менее, она очень важна. Предполагается, что БД должна содержать все данные, представляющие интерес для предприятия, так что при проектировании следует предусмотреть возможность размещения в БД всех необходимых данных.

2. Исключение избыточного дублирования данных.

Как мы уже знаем, в реляционной БД не обойтись без необходимого дублирования данных ключевых атрибутов для представления связей между кортежами отношений. А вот все другие случаи дублирования данных должны быть исключены.

3. Сведение числа хранимых в БД отношений к минимуму.

Не следует увлекаться декомпозицией отношений – возможно получение большого числа излишне «мелких» отношений. При этом в запросах потребуется выполнение дополнительных дорогостоящих операций соединения этих отношений для получения всей требуемой информации.

4. Нормализация отношений для упрощения решения проблем, связанных с вставкой, обновлением и удалением данных.

Некоторым отношениям сопутствуют проблемы изменения их данных. Проектировщик должен уметь обнаруживать эти потенциально опасные отношения и «нормализовать их» посредством разбиения на два или более отношений.

Третья и четвертая цели взаимно противоречивы, и проектировщику требуется достигнуть разумного компромисса.

Поскольку задачей нашего учебного пособия является знакомство читателя прежде всего с практически полезными идеями, в своем изложении теории реляционных БД мы ограничимся лишь рассказом о той ее части, которая необходима для усвоения одной из самых простых и конструктивных методик проектирования реляционных схем БД. Она заключается в приведении отношений схемы в нормальную форму Бойса-Кодда (НФБК), с одной стороны, минуя получение вторых и третьих нормальных форм, с другой стороны, не достигая четвертой и пятой нормальных форм.

Аргументация этой методики достаточно убедительна. Во-первых, НФБК представляет собой наивысшую степень качества отношений, которой можно достигнуть, анализируя лишь относительно простые, функциональные зависимости. Четвертая и пятая нормальные формы требуют рассмотрения несомненно более трудных для понимания

многозначных зависимостей и зависимостей соединения. Во-вторых, доведение отношений схемы сразу до НФБК конструктивно проще процедуры последовательного преобразования отношений схемы сначала во вторую, затем в третью нормальные формы и, наконец, в НФБК.

Универсальное отношение

КОНСУЛЬТАНТ (Номер студента (СНОМ), Курс, Семестр, Фамилия студента (СФАМ), Номер комнаты (КНОМ), Номер телефона (ТНОМ), Оценка)

СНОМ	СФАМ	КНОМ	ТНОМ	КУРС	СЕМЕСТР	ОЦЕНКА
3215	Джонс Г.	120DH	2136	MTH122	F84	1.6
				SCI120	F84	2.4
				PHY230	W85	2.1
				MTH122	W85	2.3
3462	Смит А.	238VH	2344	MTH122	W84	2.3
				MTH123	W85	3.5
				PSY220	W85	3.7
3567	Хауз Дж.	120DH	2136	SCI239	W84	3.3
				EGR171	F84	3.5
				PHY141	F84	1.8
4756	Алекс К.	345VH	3321	MUS389	F83	4.0

На протяжении этого пункта пособия нас будет сопровождать академическая демонстрационная ПрО.

Предположим, что требуется разработать небольшую БД для академического консультанта университета. У консультанта имеется много консультируемых им студентов, живущих на территории университетского городка. Задачей консультанта является помощь студенту при построении индивидуальной образовательной траектории – графика освоения дисциплин образовательной программы. Для этого ему необходимо иметь сведения о ранее освоенных студентом дисциплинах.

Первый шаг процесса проектирования состоит в определении как всех атрибутов, наличие которых в БД ожидает консультант, так и связей между атрибутами. Эта информация получается проектировщиком от консультанта в ходе ряда детальных обсуждений, не оставляющих сомнений в том, что он знает, какие данные должны быть в БД, каким образом БД будет использоваться, и какую информацию консультант ожидает получать от БД.

После нескольких бесед с консультантом имена и условия, связанные с атрибутами, хранение которых предполагается, были определены следующим образом.

СНОМ (номер студента): целое значение, уникальное для каждого студента университета.

СФАМ (фамилия студента): каждый студент имеет только одну фамилию, но возможно, что одну и ту же фамилию носят несколько студентов.

КНОМ (номер комнаты в общежитии городка): каждый студент живет на территории городка и имеет комнату; в одной комнате может проживать более одного студента.

ТНОМ (номер телефона студента): каждая комната общежития имеет один телефон, и им пользуются все студенты, проживающие в этой комнате.

Курс (номер курса): это идентификационный номер курса (дисциплины), посещаемого студентом; консультант будет сохранять данные только о курсах, завершенных студентом.

Семестр (университетский семестр): представляет собой семестр, в котором данный курс был завершен студентом; возможно, что студент изучал один и тот же курс в различных семестрах.

Оценка (оценка за курс): оценка, полученная студентом за определенный курс в данном семестре.

На слайде представлены схема и экстенсионал таблицы *КОНСУЛЬТАНТ*, предоставленной консультантом в качестве примера представления необходимых ему данных. Хотя данные и представлены в виде таблицы, она не является отношением.

Действительно, рассмотрим одну строку таблицы (всего в ней четыре строки – по количеству представленных в ней студентов). Значения четырех полей (*СНОМ*, *СФАМ*, *КНОМ* и *ТНОМ*) – атомарные, в то время как значения в полях *Курс*, *Семестр* и *Оценка* – множественные. Данная «строка» очевидным образом отличается по форме от кортежей, рассматривавшихся нами до сих пор. Отличие в том, что не все поля строки содержат атомарные значения.

Приведение универсального отношения в первую нормальную форму

СНОМ	СФАМ	КНОМ	ТНОМ	КУРС	СЕМЕСТР	ОЦЕНКА
3215	Джонс Г.	120DH	2136	MTH122	F84	1.6
3215	Джонс Г.	120DH	2136	SCI120	F84	2.4
3215	Джонс Г.	120DH	2136	PHY230	W85	2.1
3215	Джонс Г.	120DH	2136	MTH122	W85	2.3
3462	Смит А.	238VH	2344	MTH122	W84	2.3
3462	Смит А.	238VH	2344	MTH123	W85	3.5
3462	Смит А.	238VH	2344	PSY220	W85	3.7
3567	Хауз Дж.	120DH	2136	SCI239	W84	3.3
3567	Хауз Дж.	120DH	2136	EGR171	F84	3.5
3567	Хауз Дж.	120DH	2136	PHY141	F84	1.8
4756	Алекс К.	345VH	3321	MUS389	F83	4.0

Для придания приведенным на предыдущем слайде данным формы отношения необходимо реконструировать их таким образом, чтобы каждый элемент кортежа имел атомарное значение. Обычно это удается сделать с помощью простого процесса вставки (результат представлен на слайде). В результате этого процесса добавляется большой объем избыточных данных, исключение которых достигается последующими шагами проектирования. Полученное отношение называют универсальным отношением.

Определение 4.2.5.1. Универсальное отношение – это отношение, в которое включаются все представляющие интерес атрибуты ПрО и которое может таким образом содержать все данные, предполагающиеся к хранению в БД. Для малых БД (включающих не более 15-20 атрибутов) универсальное отношение может использоваться в качестве отправной точки при проектировании РБД.

Начинающий проектировщик, возможно, решит использовать отношение *КОНСУЛЬТАНТ* в качестве окончательного варианта схемы БД. Это выглядит достаточно логичным. Зачем разбивать отношение *КОНСУЛЬТАНТ* на несколько более мелких отношений, если оно способно включить в себя все данные? Существует несколько причин, почему не следует использовать данное отношение, в качестве единственного в БД. Это обусловлено тем, как будет использоваться БД, и какое воздействие на данные в отношении *КОНСУЛЬТАНТ* будут оказывать определенные операции. Различаются три специфичные проблемы: проблема, сопутствующая включению новых кортежей; проблема, связанная с удалением кортежей; проблема, обусловленная необходимостью обновления (модификации) данных в БД. Выделенные проблемы обычно называют аномалиями вставки, удаления и обновления, понимая под аномалией отклонение от нормы.

Аномалия вставки

Если у консультанта появляется новый консультируемый им студент, еще не закончивший ни одного курса, для него необходимо включить в БД кортеж с неопределенными значениями атрибутов *Курс*, *Семестр* и *Оценка*. В таком случае для добавления информации об очередном освоенном студентом курсе должен быть предусмотрен алгоритм, анализирующий наличие неопределенных значений в этих атрибутах. Если это так, задача решается с помощью команды *UPDATE*, в противном случае необходимо выполнить команду *INSERT*. В идеале действие вставки новых данных всегда должно выполняться командой *INSERT*.

Аномалия удаления

В экстенсионале нашего отношения *КОНСУЛЬТАНТ* присутствует только один кортеж, в котором *СНОМ* = 4756. Этот кортеж соответствует студентке с именем *Алекс К*.

Предположим, что консультант узнает, что эта студентка не закончила курс *MUS389*, как это отмечено в БД, и удаляет этот кортеж из отношения. Поскольку это единственный кортеж с информацией об этой студентке, его удаление приведет к исключению студентки из БД. Для устранения подобных случаев алгоритм удаления информации о сданных студентами курсах должен предусматривать анализ того, а не последний ли это курс у студента, и в этом случае вместо команды *DELETE* выполнять команду *UPDATE* с «обнулением» значений атрибутов *Курс*, *Семестр* и *Оценка*. В идеале действие удаления данных из БД всегда должно выполняться командой *DELETE*.

Аномалия обновления

В экстенсионале отношения *КОНСУЛЬТАНТ* большое число избыточных данных. Избыточность данных всегда свидетельствует о возможности модификации только части требуемых данных с помощью операции обновления (известная проблема синхронизации модификаций). Отношение *КОНСУЛЬТАНТ* характеризуется как явной, так и неявной избыточностью. Явная избыточность заключается в том, что фамилия данного студента, номер комнаты и номер телефона могут появиться в отношении несколько раз. В экстенсионале нашего отношения *КОНСУЛЬТАНТ* номер комнаты *Джонс Г.* появляется четырежды. Если она обратится к своему консультанту и сообщит ему об изменении номера ее комнаты, то консультант будет вынужден проследить изменение этого номера во всех четырех кортежах во избежание противоречивости данных.

Неявная избыточность обнаруживается в том, что один и тот же номер телефона имеют все студенты, живущие в одной комнате. В нашем случае телефонный номер для комнаты *120DH* появляется в сочетании с именами *Джонс Г.* и *Хауз Дж*. Допустим, *Джонс Г.* известила своего консультанта о том, что ее номер телефона изменен на 7777, забыв при этом сообщить о подруге по комнате. Если консультант изменит телефонный номер только в тех кортежах, которые относятся к *Джонс Г.*, то определить правильный номер телефона, расположенного в комнате *120DH*, будет затруднительно, поскольку в отношении будут присутствовать два различных телефонных номера для этой комнаты.

Как видим, многие отношения обладают неудовлетворительными качествами при выполнении операций над данными. Эти их свойства и позволяет устранить предлагаемая далее методика проектирования РБД.

4.2.5.1. Функциональные зависимости и нормальные формы

Ранее отмечалось, что составной частью проектирования РБД является процесс разбиения отношений с неудовлетворительными свойствами (аномалиями) на новые отношения. В связи с этим процессом следует поставить несколько вопросов.

1. Из какого источника вы получаете отношение (или отношения) перед началом процесса?
2. Каким образом вы можете распознать отношения, нуждающиеся в разбиении?
3. Каким образом вы будете осуществлять разбиение?
4. Что является признаком завершения процесса разбиения?

Ответы на эти вопросы нам предстоит дать в этом пункте.

Для БД с числом атрибутов меньшим 20 начальной точкой указанной процедуры может служить универсальное отношение. Это отношение содержит все представляющие интерес атрибуты и имеет структуру, в которой каждый кортеж состоит из атомарных элементов.

Как догадывается читатель, с такими «смешными» по семантической сложности БД давно уже не имеют дело проектировщики систем БД. Как поступать в случае, когда в ПрО одних только типов объектов больше 20, не говоря уж о суммарном количестве их атрибутов? Классическая методика на этот счет рекомендаций не дает. И тут ей на помощь приходит семантическая методика, позволяющая построить множество предварительных отношений. Предварительными эти отношения называют потому, что окончательно их судьба будет решена в ходе применения для них классической методики. Поскольку в каждом из них редко встречается более 20 атрибутов, предварительные отношения прекрасно подходят в качестве универсальных отношений, используемых классической методикой.

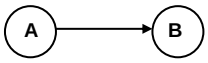
Так сочетанием двух методик снимается проблема применения классической методики в семантически сложных ПрО.

Функциональные зависимости и нормальные формы

Говорят, что отношение находится в первой нормальной форме (1НФ), если каждый его элемент имеет и всегда будет иметь атомарное значение.

Если даны два атрибута A и B, то говорят, что B функционально зависит (ФЗ) от A, если для каждого значения A существует не более одного связанного с ним значения B. A и B могут быть составными. Говорят еще, что A функционально определяет B.

A -> B



Если A -> B и B не зависит функционально от любого подмножества A, то говорят, что A представляет собой детерминант B.

Коддом доказано утверждение о том, что большинство потенциальных аномалий в БД устраняется декомпозицией каждого отношения в нормальную форму Бойса-Кодда (НФБК).

Отношение находится в нормальной форме Бойса-Кодда (НФБК), если и только если каждый детерминант атрибутов отношения является возможным ключом отношения.

Процесс нормализации отношений основывается на концепции нормальных форм. Говорят, что отношение находится в определенной нормальной форме, если оно удовлетворяет заданному для этой формы набору условий. В таком случае процедуру нормализации можно охарактеризовать как последовательное приведение данного набора отношений к некоторой более желательной форме. Процедура нормализации включает разбиение, или декомпозицию, отношения, не удовлетворяющего условиям нормальной формы, на другие отношения. Процесс декомпозиции на самом деле является операцией проекции, т.е. каждое из получаемых отношений в действительности является проекцией исходного отношения. Таким образом, оператор декомпозиции в процедуре нормализации фактически является оператором проекции.

Определение 4.2.5.2. Говорят, что отношение находится в **первой нормальной форме (1НФ)**, если каждый его элемент имеет и всегда будет иметь атомарное значение. Отношение должно быть в 1НФ даже прежде постановки вопроса о его разбиении на два или более отношения. По большому счету реляционная модель и теория имеют дело только с такими отношениями.

Определение 4.2.5.3. Процесс разбиения отношения с целью уменьшения вероятности возникновения аномалий называется **декомпозицией**. Ключевой для осуществления декомпозиции логическим методическим путем является концепция функциональных зависимостей между атрибутами в рассматриваемом отношении (при анализе отношения с точки зрения повышенных нормальных форм в расчет берутся другие виды зависимостей в данных).

Определение 4.2.5.4. Функциональная зависимость определяется следующим образом: если даны два атрибута *A* и *B*, то говорят, что *B* **функционально зависит (ФЗ)** от *A*, если для каждого значения *A* в любой момент времени существует не более одного связанного с ним значения *B*. *A* и *B* могут быть составными. Говорят еще, что *A* **функционально определяет** *B*. Надеюсь, все увидели, что в понятии «функциональная зависимость» скрыто функциональное отображение между атрибутами *A* и *B*.

С практической точки зрения смысл данного определения состоит в том, что если *B* функционально зависит от *A*, то каждый из кортежей, имеющих одно и то же значение *A*, должен иметь также одно и то же значение *B*. Значения *A* и *B* могут изменяться время от времени, но при этом они должны изменяться так, чтобы каждое уникальное значение *A* имело только одно значение *B*, связанное с ним. ФЗ описываются с помощью нескольких различных нотаций. Две наиболее часто используемых – символическая и графическая – показаны на слайде.

Единственный способ определения функциональных зависимостей для схемы отношения заключается в том, чтобы внимательно проанализировать семантику

атрибутов. В этом смысле зависимости являются фактически высказываниями о закономерностях реального мира.

Функциональные зависимости – это особый вид ограничений целостности, а потому они, несомненно, являются понятием семантическим. Распознавание ФЗ представляет собой часть процесса выяснения смысла тех или иных данных.

Предположим, что в реальном мире существует некоторое функциональное отображение. Поскольку это ограничение является частью семантического описания предметной области, оно должно быть каким-то образом представлено в БД. Для этого ограничение необходимо описать в определении БД таким образом, чтобы оно могло быть приведено в действие средствами СУБД. Способ представления функционального отображения в определении реляционной БД состоит в объявлении соответствующей ФЗ.

ФЗ не могут быть доказаны путем простого просмотра отдельного экстенсионала отношения и нахождения двух атрибутов, имеющих те же значения в более чем одном кортеже. Такой анализ может служить ключом к тому, в каком направлении следует вести поиск ФЗ, но не доказательством.

Определение 4.2.5.5. Если *A* -> *B* и *B* не зависит функционально от любого подмножества *A*, то говорят, что *A* представляет собой **детерминант** *B*.

Одним из самых первых, но и одним из самых важных результатов в области реляционных БД стало доказанное Коддом утверждение о том, что большинство потенциальных аномалий в БД устраняется декомпозицией каждого отношения в нормальную форму Бойса-Кодда (НФБК).

Определение 4.2.5.6. Отношение находится в **нормальной форме Бойса-Кодда (НФБК)**, если и только если каждый детерминант атрибутов отношения является возможным ключом отношения.

Хотя существуют нормальные формы более высокого уровня, которые накладывают более сильные ограничения на разрабатываемые отношения, на практике большинство проектировщиков стараются получить отношения в НФБК.

Универсальное отношение КОНСУЛЬТАНТ не находится в НФБК

КОНСУЛЬТАНТ (Номер студента (СНОМ), Курс, Семестр, Фамилия студента (СФАМ), Номер комнаты (КНОМ), Номер телефона (ТНОМ), Оценка)

Функциональные зависимости:
СНОМ -> СФАМ
СНОМ -> КНОМ
КНОМ -> ТНОМ
ТНОМ -> КНОМ
СНОМ -> ТНОМ
СНОМ, Курс, Семестр -> Оценка

Детерминанты:
СНОМ, КНОМ, ТНОМ, {СНОМ, Курс, Семестр}

Возможный ключ – {СНОМ, Курс, Семестр}

В качестве примера обратимся к отношению *КОНСУЛЬТАНТ*, схема которого приведена на слайде, там же выписаны все функциональные зависимости, имеющиеся между атрибутами этого отношения. Укажем соображения, послужившие основой выделения этих ФЗ.

Номера студентов являются уникальными. Каждому студенту назначается номер *СНОМ*, причем все номера различны. Таким образом, если вы имеете значение *СНОМ* студента, вы знаете, что с ним может быть связана только одна фамилия *СФАМ*: *СНОМ -> СФАМ*. Обратное не является верным – *СФАМ -> СНОМ* не является правильной ФЗ, поскольку несколько студентов могут иметь одну и ту же фамилию.

Каждый студент прикреплен к одной комнате общежития, но в одной комнате может проживать более чем один студент. Таким образом, *СНОМ -> КНОМ* является верной ФЗ, а *КНОМ -> СНОМ* – нет.

Поскольку в каждой комнате установлен только один телефон, номер которого уникален, и каждый телефон, в свою очередь, принадлежит жильцам только одной комнаты, получаем *КНОМ -> ТНОМ* и *ТНОМ -> КНОМ.*

Поскольку в каждой комнате установлен только один телефон, и этот телефон имеет уникальный номер, следовательно, только один телефонный номер может быть связан с данным студентом, или иначе *СНОМ -> ТНОМ.*

Последняя ФЗ представляет собой пример сложного набора атрибутов, входящих в ФЗ. Зависимость *СНОМ, Курс, Семестр -> Оценка* означает, что оценка однозначно определяется только в том случае, если известен *СНОМ* студента, изучавшего данный *Курс* в данном *Семестре*. Необходимо помнить, что студент может повторить прохождение учебного курса в другом семестре и получить в общем случае другую оценку.

Опытному проектировщику БД достаточно беглого взгляда на ФЗ отношения, чтобы сделать вывод о том, что отношение *КОНСУЛЬТАНТ* нельзя считать «хорошим».

Детерминанты в отношении *КОНСУЛЬТАНТ* определить легко: ими являются левые части всех ФЗ в отношении.

Отношение *КОНСУЛЬТАНТ* имеет только один возможный ключ, а именно набор атрибутов {*СНОМ, Курс, Семестр*}. Этот вывод получен путем нахождения минимального набора атрибутов, таких что, если известны их значения, то можно однозначно определить и значения остальных атрибутов отношения. С помощью ФЗ легко видеть, что один номер *СНОМ* определяет *СФАМ, КНОМ* и *ТНОМ*, а для определения значения *Оценки* должны быть известны значения всего набора атрибутов {*СНОМ, Курс, Семестр*}, составляющих возможный ключ отношения. Таким образом, если известны

значения атрибутов возможного ключа, то значения всех других атрибутов могут быть однозначно определены.

Отношение *КОНСУЛЬТАНТ* не находится в НФБК, поскольку не каждый детерминант в отношении является возможным ключом. А значит, для более качественной работы с БД его необходимо подвергнуть декомпозиции.

4.2.5.2. Декомпозиционный алгоритм проектирования

К этому моменту все готово для общего описания одного метода, определяющего, как следует осуществлять с помощью декомпозиции проектирование реляционных БД. Далее будет показано, что этот метод не свободен от недостатков и поэтому требует усовершенствования, которое мы также проведем.

Декомпозиционный алгоритм проектирования

1. **Разработка универсального отношения для БД и преобразование его в 1НФ. Оно пока является единственным в очереди еще не проверенных на НФБК отношений.**
2. **Определение всех ФЗ очередного отношения.**
3. **Определение того, находится ли очередное отношение в НФБК. Если да, то проектирование для него завершается, если нет – это отношение декомпозируется на два новых, которые помещаются в очередь еще не проверенных на НФБК отношений.**
4. **Повторение шагов 2 и 3 для каждого очередного отношения. Проектирование завершается, когда очередь еще не проверенных на НФБК отношений опустеет, а, значит, все полученные отношения находятся в НФБК.**

Предложенный выше метод не определяет, как осуществлять декомпозицию отношения, не приведенного к НФБК, на два отношения.

Декомпозиция осуществляется с помощью ФЗ следующим образом.

Пусть отношение R(A, B, C, D, E, …) не находится в НФБК. Определяется ФЗ (например, C -> D), которая является причиной этого (C является детерминантом, но не является возможным ключом).

Создаются два новых отношения

R1(A, B, C, E, …) и R2(C, D),

где зависимая часть ФЗ (D) была выделена из R и опущена при формировании R1. ФЗ была полностью использована при формировании R2.

Отсутствие потерь информации при декомпозиции отношения R(X, Y, Z) на R1(X, Y) и R2(Y, Z) гарантируется при условии, если от общего атрибута (общих атрибутов) двух полученных отношений (Y) зависят все атрибуты одного из отношений.

Простым правилом выбора ФЗ для декомпозиции может служить поиск цепочек ФЗ вида A -> B -> C с последующим использованием крайней правой зависимости.

Способ декомпозиции, приведенный на слайде, обеспечивает декомпозицию без потерь при естественном соединении. Вообще процесс декомпозиции имеет два важных свойства, которые следует учитывать. Первое из них – это свойство **соединения без потерь** (англ. lossless-join), которое позволяет восстановить любой кортеж исходного отношения, используя соответствующие кортежи меньших отношений, полученных в результате декомпозиции. Второе – свойство **сохранения зависимостей** (англ. dependency preservation), которое позволяет сохранить все ограничения, наложенные на исходное

отношение, посредством наложения некоторых ограничений на каждое из меньших отношений, полученных после декомпозиции.

Более формально эти свойства декомпозиции определяются следующим образом.

Определение 4.2.5.7. Рассмотрим схему отношения R, на атрибутах которого задано множество функциональных зависимостей F. Заменим R множеством схем $\rho = \{R_1, \ldots, R_k\}$, таким, что $\bigcup_{i=1}^{k} R_i = R$. В данном случае эта запись означает, что множество атрибутов R есть объединение множеств атрибутов всех R_i. Декомпозиция **эффективна**, если схема ρ эквивалентна R и удаляет некоторые аномалии. Схемы ρ и R **эквивалентны**, если удовлетворяются два условия:

1) соединения без потерь информации;
2) сохранения зависимостей.

Определение 4.2.5.8. Декомпозиция ρ обладает **свойством соединимости без потерь информации** по отношению к множеству функциональных зависимостей F, если для каждой реализации (экстенсионала) r отношения R, удовлетворяющей F, $r = *_{i=1}^{k} \pi_{R_i}(r)$, где «*» – операция естественного соединения, а R_i в данном случае представляет атрибуты отношения R_i. Свойство соединимости без потерь гарантирует восстановление по расширениям отношений-проекций $\{R_1, \ldots, R_k\}$ любого расширения r отношения R. Если при соединении получается большее, чем в исходном расширении r, число кортежей, часть информации теряется, а декомпозиция R в ρ не будет декомпозицией без потерь информации.

Необходимое и достаточное условие получения соединения без потерь информации двух отношений R_1 и R_2 сводится к тому, что пересечение их атрибутов должно функционально определять R_1 и/или R_2:

$R_1 \cap R_2 \rightarrow R_1$ или $R_1 \cap R_2 \rightarrow R_2$.

Определение 4.2.5.9. Декомпозиция называется **сохраняющей зависимости**, если зависимости R сохранены в новой схеме отношений ρ. Формально это свойство можно выразить следующим образом. Рассмотрим подмножество ФЗ F_i, содержащее все такие ФЗ $X \rightarrow Y$, что $XY \subseteq R_i$. Декомпозиция ρ сохраняет множество ФЗ F, если $\left(\bigcup_{i=1}^{k} F_i\right)^{+} = F^{+}$. Операция «+» образует множество всех ФЗ, выводимых из исходного множества ФЗ (т.е. являющихся их следствиями).

Если оператором декомпозиции в процедуре нормализации является операция проекции, то обратной операцией служит операция соединения. «Обратимость» означает, что исходное отношение равно соединению его проекций, полученных в ходе декомпозиции.

Определение 4.2.5.10. Классической теорией нормализации (normalization) называют теорию декомпозиции без потерь, основанной на проекции как операторе декомпозиции и на естественном соединении как соответствующем операторе композиции.

В процессе нормализации часто возникает ситуация, когда отношение может быть подвергнуто декомпозиции без потерь несколькими разными способами. Это связано, в частности, с порядком применения зависимостей для декомпозиции и является одной из причин неоднозначности проекта схемы РБД.

Концепция независимых проекций предоставляет критерий выбора одного из возможных вариантов декомпозиции. В частности, вариант декомпозиции,

обеспечивающий независимость проекций, в общем случае предпочтительнее вариантов, в которых проекции будут зависимы.

Определение 4.2.5.11. Риссанен показал, что проекции R_1 и R_2 отношения R будут **независимы** тогда и только тогда, когда:

а) каждая ФЗ в отношении R является логическим следствием ФЗ в его проекциях R_1 и R_2;

б) общие атрибуты проекций R_1 и R_2 образуют возможный ключ, по крайней мере, для одной из этих проекций.

Как видим, независимые проекции сочетают в себе оба свойства – и соединимость без потерь, и сохранение зависимостей.

Определение 4.2.5.12. Отношение, которое не может быть подвергнуто декомпозиции с получением независимых проекций, называется **атомарным**. Однако это вовсе не означает, что каждое неатомарное отношение следует непременно разбить на атомарные компоненты. Таким образом, можно дойти до бинарных отношений, что грозит неэффективностью при реализации БД.

Пример декомпозиции отношения КОНСУЛЬТАНТ

КОНСУЛЬТАНТ (СНОМ, Курс, Семестр, СФАМ, КНОМ, ТНОМ, Оценка)
Шаг 1. КНОМ -> ТНОМ
R1 (СНОМ, Курс, Семестр, СФАМ, КНОМ, Оценка)
СНОМ -> СФАМ
СНОМ -> КНОМ
СНОМ, Курс, Семестр -> Оценка
R2 (КНОМ, ТНОМ)
КНОМ -> ТНОМ
ТНОМ -> КНОМ

Шаг 2. СНОМ -> СФАМ, КНОМ
R3 (СНОМ, Курс, Семестр, Оценка)
СНОМ, Курс, Семестр -> Оценка
R4 (СНОМ, СФАМ, КНОМ)
СНОМ -> СФАМ, КНОМ

В качестве примера использования приведенного алгоритма выполним декомпозицию отношения *КОНСУЛЬТАНТ*.

Обращаясь вновь к детерминантам и возможным ключам этого отношения, видим, что имеются три детерминанта, не являющиеся возможными ключами – *СНОМ, ТНОМ* и *КНОМ*. Кандидатами среди ФЗ для осуществления проекции являются *СНОМ -> СФАМ, СНОМ -> КНОМ, КНОМ -> ТНОМ, ТНОМ -> КНОМ* и *СНОМ -> ТНОМ.*

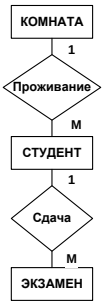
На этом этапе должно быть принято решение, какую ФЗ следует выбрать для проведения первой проекции. Не исключено, что в итоге выбора той или иной начальной проекции будут получены различные схемы БД. Если это именно так, то каждая из получившихся схем, вообще говоря, должна быть подвергнута анализу с целью выбора наиболее полно отвечающей требованиям и условиям предприятия.

Простым правилом выбора ФЗ для проекции может служить поиск «цепочки ФЗ» вида *A -> B -> C* с последующим использованием для проекции крайней правой зависимости. В нашем случае такой «цепочкой» является *СНОМ -> КНОМ -> ТНОМ* и «конец цепочки» *КНОМ -> ТНОМ* порождает первую проекцию. Полученные на первом шаге отношения *R1* и *R2* вместе с присущими им ФЗ приведены на слайде.

Отношение *R2* находится в НФБК, поскольку все его детерминанты являются возможными ключами. Отношение *R2* не нуждается более в декомпозиции. Однако отношение *R1* не находится в НФБК, так как детерминант *СНОМ* не является возможным ключом. Следовательно, отношение *R1* необходимо подвергнуть дальнейшему разложению. Детерминант *СНОМ*, из-за которого возникло затруднение, имеет два зависимых от него атрибута *СФАМ* и *КНОМ*. Как будет показано далее, две ФЗ *СНОМ -> СФАМ* и *СНОМ -> КНОМ* эквивалентны одной ФЗ с составной правой частью *СНОМ -> СФАМ, КНОМ.*

Проекция отношения *R1*, порождаемая этой ФЗ, приводит к получению отношений *R3* и *R4*. Эти два отношения находятся в НФБК и вместе с отношением *R2* могли бы использоваться при формировании БД для консультанта.

Результат декомпозиции



R3 (ЭКЗАМЕН)

СНОМ	КУРС	СЕМЕСТР	ОЦЕНКА
3215	MTH122	F84	1.6
3215	SCI120	F84	2.4
3215	PHY230	W85	2.1
3215	MTH122	W85	2.3
3462	MTH122	W84	2.3
3462	MTH123	W85	3.5
3462	PSY220	W85	3.7
3567	SCI239	W84	3.3
3567	EGR171	F84	3.5
3567	PHY141	F84	1.8
4756	MUS389	F83	4.0

R2 (КОМНАТА)

КНОМ	ТНОМ
120DH	2136
238VH	2344
345VH	3321

R4 (СТУДЕНТ)

СНОМ	СФАМ	КНОМ
3215	Джонс Г.	120DH
3462	Смит А.	238VH
3567	Хауз Дж.	120DH
4756	Алекс К.	345VH

На слайде представлены экстенсионалы хранимых отношений *R2*, *R3* и *R4*, эквивалентные исходному экстенсионалу отношения *КОНСУЛЬТАНТ*. Заметим, что в процессе декомпозиции автоматически произошло разбиение исходного отношения *КОНСУЛЬТАНТ* на три логических компонента: *R2*, в котором содержится информация о комнатах и их телефонах; *R3*, в котором содержится информация об освоенных студентами учебных курсах и сданных по ним экзаменам, и *R4*, в котором содержится информация о студентах. Такое логическое разбиение является прямым результатом использования в процессе декомпозиции заложенной в ФЗ информации, детализирующей то, как различные атрибуты в исходном отношении соотносятся друг с другом.

Сейчас самое время задаться вопросом: присутствуют ли в нашей БД аномалии, которые характеризовали отношение *КОНСУЛЬТАНТ*, или декомпозиция автоматически привела к их устранению?

Вставка

Когда отношение *КОНСУЛЬТАНТ* служило основой БД, новый студент не мог быть в нее включен до действительного завершения им какого-либо курса. В окончательной БД информация общего характера о консультируемых студентах хранится в отношении *R4*. Как только новый студент принимается в университет, и ему отводится комната, так сразу кортеж этого студента может быть включен в отношение *R4*. Студенту нет необходимости даже приступать к посещению учебного курса для того, чтобы быть включенным в БД. С другой стороны, всякое добавление информации о сданном студентом курсе всегда сводится к вставке нового кортежа в отношение *R3*. Таким образом, исходная аномалия вставки исключена с помощью декомпозиции.

Обновление

При использовании отношения *КОНСУЛЬТАНТ* проблема возникла на этапе изменения консультантом номера телефона мисс *Джонс Г.* на 7777. Результатом этого изменения явилось появление в БД двух различных телефонных номеров для телефона, установленного в комнате *120DH*. В новой схеме БД телефонные номера располагаются в отношении *R2*, и каждая комната может иметь только один телефонный номер, назначенный расположенному в этой комнате телефонному аппарату. Следует помнить, что номер комнаты *КНОМ* является первичным ключом отношения *R2*, а значения первичного ключа по определению должны быть уникальными. Для модификации телефонного номера *Джонс Г.* следует в кортеже отношения *R2*, в котором *КНОМ* равен *120DH*, изменить номер *ТНОМ* на 7777. Обратите внимание на то обстоятельство, что это действие заключается в изменении телефонного номера для комнаты, так что для всех

живущих в этой комнате студентов телефонный номер также будет изменен. Таким образом, исходная аномалия обновления устраняется с помощью НФБК-проектирования.

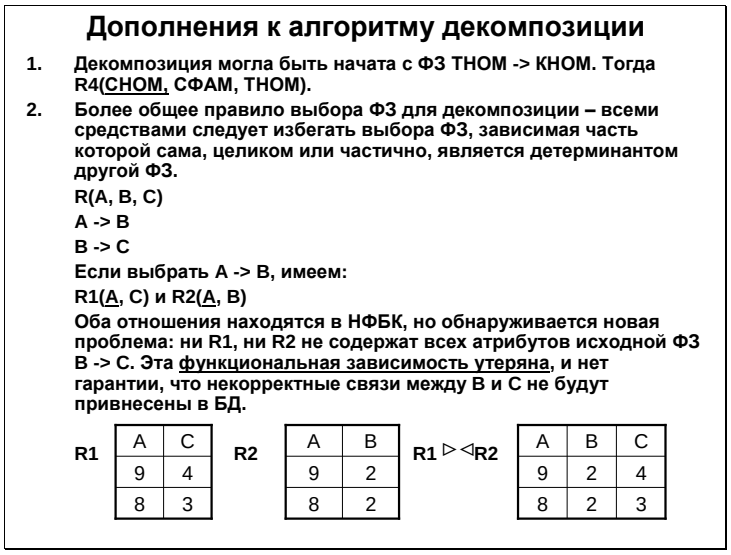
Удаление

Когда отношение *КОНСУЛЬТАНТ* использовалось в качестве единственного отношения БД, удаление информации о последнем освоенном студенткой *Алекс К.* курсе приводило к исчезновению из БД всей информации об этой студентке. Этого не может произойти в случае использования новой схемы с тремя отношениями, поскольку информация об оценках и информация о студентах общего характера разнесена в два различных отношения *R3* и *R4*. При исключении факта, что студент с номером 4756 освоил курс *MUS389* в семестре *F83*, из отношения *R3* будет удален кортеж <4756, *MUS389, F83,* 4.0>. Это действие никак не скажется на общей информации о студенте, хранимой в *R4*.

Итак, все три аномалии, присутствовавшие в БД, состоящей из одного отношения, устраняются при таком проектировании. Результат устранения аномалий – появление вместо одного трех требующих хранения отношений. Это означает, что запросы, которые должны быть составлены для получения информации из БД, возможно, станут более сложными, поскольку в них понадобится выполнять соединения двух или трех отношений. Но эта дань должна быть принесена в жертву для создания эффективной и удобной в работе схемы.

Справа на слайде приведена ER-диаграмма академической предметной области. Если перевести ее в реляционную модель, используя ранее приведенные правила преобразования, мы получим абсолютно тот же результат – отношения в НФБК. Этот факт говорит о том, что семантическая методика проектирования РБД, как правило, также обеспечивает эффективные реляционные схемы, но в применении она несомненно проще, почему сейчас чаще и используют именно ее.

4.2.5.3. Дополнения к алгоритму декомпозиции



Следует сделать несколько дополнительных комментариев к предложенному алгоритму декомпозиции.

Первое замечание касается того факта, что с помощью этого алгоритма может быть получено несколько вариантов схем БД. Этому способствует несколько факторов. Один из них – выбираемые для декомпозиции отношений ФЗ.

Декомпозиция отношения *КОНСУЛЬТАНТ* была начата проекцией, порождаемой ФЗ *КНОМ -> ТНОМ*. Указанная ФЗ была выбрана потому, что являлась последней в цепочке ФЗ *СНОМ -> КНОМ -> ТНОМ*.

На самом деле в этом отношении присутствует и другая цепочка с таким же числом ФЗ – *СНОМ -> ТНОМ -> КНОМ*. Здесь правой ФЗ является *ТНОМ -> КНОМ*. Если эта ФЗ выделяется первой из отношения *КОНСУЛЬТАНТ*, то в итоге будет получена схема БД со следующей совокупностью отношений в НФБК:

R2 (КНОМ, ТНОМ),

R3 (СНОМ, Курс, Семестр, Оценка),

R4 (СНОМ, СФАМ, ТНОМ).

Эта схема БД столь же корректна, как и предыдущая. Единственное отличие состоит в том, что атрибут *ТНОМ* стал играть роль более важную, нежели *КНОМ*. *ТНОМ* в данной ситуации используется в качестве первичного ключа для отношения *R2* и атрибута, связывающего *R4* и *R2*. Два различных решения одной проблемы являются прямым результатом взаимной зависимости, существующей между атрибутами *ТНОМ* и *КНОМ*. Какое из двух решений является «лучшим» определяется выбором проектировщика и зависит до некоторой степени от того, как консультант планирует использовать БД.

Второе замечание касается правила выбора ФЗ. Ранее было сказано, что в процессе проектирования с помощью проекции декомпозицию следует осуществлять путем поиска цепочек ФЗ с последующим проецированием, порождаемым ФЗ, замыкающей цепочку. Более обобщенное правило можно сформулировать следующим образом: всеми средствами следует избегать выбора ФЗ, зависимая часть которой сама – целиком или частично – является детерминантом другой ФЗ. Понятно, что предыдущее правило представляет собой частный случай последнего, при котором зависимая часть целиком является детерминантом.

На слайде приведен пример декомпозиции, нарушающей правило выбора ФЗ. Если вопреки правилу для декомпозиции отношения *R(A, B, C)* с цепочкой ФЗ *A -> B -> C*

выбрана зависимость *A -> B*, будут получены отношения *R1(A, C)* и *R2(A, B)*. Хотя оба эти отношения находятся в НФБК, со всей отчетливостью обнаруживается следующая проблема: ни отношение *R1*, ни отношение *R2* не содержат все атрибуты ФЗ *B -> C*, которая является ФЗ исходного отношения. Эта зависимость утеряна в процессе проектирования. С практической точки зрения это означает, что если приведенные отношения *R1* и *R2* будут использованы для создания БД, то нельзя будет утверждать, что некорректные связи между *B* и *C* не будут привнесены в БД. В нижней части слайда приведен пример экстенсионалов этих отношений, иллюстрирующий выявленную проблему. При соединении *R1* и *R2* по атрибуту *A* два значения *C* (3 и 4) могут быть соотнесены с одним и тем же значением *B* (2), что противоречит ФЗ, утерянной в процессе проектирования.

Проблема в данном примере возникает из-за проекции, порожденной ФЗ, зависимая часть которой сама является детерминантом другой ФЗ. При использовании правила цепочки (или обобщенного правила) эта проблема не возникает.

3. Другим случаем возможной потери ФЗ в процессе проектирования является ситуация, в которой один атрибут зависит от двух различных детерминантов.

R(A, C , B)
A -> B
C -> B

Отношение R не находится в НФБК. Если выбрать одну из ФЗ для декомпозиции, вторая теряется.

Если выбрать A -> B, имеем:

R1(A, C) и R2(A, B) => C -> B утеряна

Если выбрать C -> B, имеем:

R1(A, C) и R2(C, B) => A -> B утеряна

Остается разбить R на R1(A, B) и R2(C, B). Они находятся в НФБК, и исходные ФЗ сохранены.

Метод синтеза: необходимо все ФЗ с одинаковыми детерминантами выделить в группы и для каждой группы построить свое отношение, куда включить все атрибуты всех ФЗ этой группы.

Третье замечание касается того, что метод декомпозиции не является единственным возможным методом проектирования реляционных схем БД, более того в некоторых случаях он дает неудовлетворительные результаты.

На слайде представлен другой случай потери ФЗ в процессе проектирования. Он возникает в ситуации, когда один атрибут зависит от двух различных детерминантов. Пусть для отношения *R(A, B, C)* определены ФЗ *A -> B* и *C -> B*. Это отношение не находится в НФБК, так как имеется только один возможный ключ – {*A, C*}, в то время как детерминантами являются атрибуты *A* и *C*. Правило цепочек здесь неприменимо по причине их отсутствия. Кроме того, если одна из ФЗ будет выделена и использована для декомпозиции, то другая будет неизбежно утеряна (оба варианта приведены на слайде). Возникла ситуация, обязывающая проектировщика найти способ разбиения исходного отношения на *R1(A, B)* и *R2(C, B)*, не приводящий к утере ни одной ФЗ.

Этот путь не соответствует стандартному методу декомпозиции, но может привести к лучшему результату. Единственное, что может сделать проектировщик, столкнувшись с указанной ситуацией, это проверить три возможных набора отношений и оценить, какой из трех наиболее соответствует нуждам предприятия. В частности, полученные в последнем случае отношения необходимо проверить на предмет возникновения проблем в случае соединения двух итоговых отношений при поиске данных на этапе эксплуатации окончательного варианта БД.

Второй метод разбиения отношения, используемый в последнем примере, хотя и основан на подходе к проектированию, отличном от декомпозиции, тем не менее, используется многими проектировщиками. В основе подхода, называемого некоторыми **методом синтеза**, лежит утверждение, что необходимо все ФЗ с одинаковыми детерминантами выделять в группы, и каждой группе отводить свое собственное отношение. Получаемые отношения проверяются на их соответствие НФБК. В последнем примере были две зависимости с различными детерминантами. Согласно методу синтеза каждой ФЗ следует выделить ее собственное отношение – *R1(A, B)* и *R2(C, B)*.

Метод синтеза может быть использован как самостоятельно, так и в сочетании с методом декомпозиции. Обычно акцент делается на метод декомпозиции, а метод синтеза используется как альтернатива при поиске выхода из затруднительных ситуаций, подобной разобранной выше. Тот факт, что существуют различные методы проектирования, которые могут быть использованы как порознь, так и до некоторой степени смешанным образом, свидетельствует о том, что проектирование БД является частично наукой, а частично искусством. Возможность развития нескольких вполне «законных» проектов схемы БД, имеющих общую исходную точку – универсальное

отношение, является неотъемлемым фактором проектирования БД. Часть процесса проектирования заключается в оценке нескольких альтернатив с целью выявления наиболее полно отвечающей нуждам предприятия.

4.2.5.4. Избыточные ФЗ

Алгоритм проектирования реляционных БД, рассмотренный ранее, кажется на первый взгляд довольно естественным, однако он не свободен от некоторых внутренних проблем. Одна проблема заключается в том, что процесс декомпозиции может осложниться в результате присутствия так называемых избыточных ФЗ.

Определение 4.2.5.13. Зависимость, не заключающая в себе такой информации, которая не могла бы быть получена на основе других зависимостей из числа использованных при проектировании БД, называется **избыточной ФЗ.**

Поскольку избыточная ФЗ не содержит уникальной информации, она может быть удалена из набора ФЗ без отрицательного воздействия на результаты. Избыточные ФЗ удаляются на начальном этапе проектирования до применения алгоритма декомпозиции.

Избыточные ФЗ

Пусть F – множество ФЗ для схемы отношения R и X->Y – некоторая ФЗ. Говорят, что зависимость X->Y логически следует из F, если для каждого экстенсионала отношения R, удовлетворяющего зависимостям из F, удовлетворяется также X->Y. В таком случае зависимость X->Y является избыточной.

Существуют правила вывода новых ФЗ из уже имеющихся. Эти правила могут использоваться для определения избыточных ФЗ: если ФЗ может быть выведена из других ФЗ с помощью этих правил, она избыточна.

Дадим более формальное определение избыточных ФЗ.

Определение 4.2.5.14. Пусть *F* – множество ФЗ для схемы отношения *R* и *X -> Y* – некоторая ФЗ. Говорят, что зависимость *X -> Y* **логически следует** из *F*, если для каждого экстенсионала отношения *R*, удовлетворяющего зависимостям из *F*, удовлетворяется также *X -> Y*. В таком случае зависимость *X -> Y* является **избыточной.**

Существуют правила вывода новых ФЗ из уже имеющихся. Эти правила могут использоваться для определения избыточных ФЗ: если ФЗ может быть выведена из других ФЗ с помощью этих правил, она избыточна.

Правила вывода функциональных зависимостей

А1 (РЕФЛЕКТИВНОСТЬ): A,X -> X

А2 (ПОПОЛНЕНИЕ): A -> B => A,Z -> B,Z и A,Z -> B

А3 (ТРАНЗИТИВНОСТЬ): A -> B и B -> C => A -> C

П4 (ОБЪЕДИНЕНИЕ): A -> B и A -> C => A -> B,C

П5 (ДЕКОМПОЗИЦИЯ): A -> B,C => A -> B и A -> C

П6 (ПСЕВДОТРАНЗИТИВНОСТЬ): A -> B и B,Z -> C => A,Z -> C

Первые три правила – аксиомы Армстронга. Они являются надежными и полными.

Надежность: если зависимость X -> Y выведена из F с помощью этих аксиом, то она справедлива в любом отношении, в котором справедливы зависимости из F.

Полнота: всякий раз, когда зависимость X -> Y не может быть выведена из F с помощью этих аксиом, она логически не следует из F.

На самом деле основное использование правил вывода ФЗ связано с получением всех ФЗ, присущих БД. Как правило, этот процесс начинается с определения функциональных зависимостей, которые представляются очевидными с точки зрения их семантики. Но в отношениях может также существовать целый ряд других полезных функциональных зависимостей. Знание полной картины ФЗ позволяет, в частности, обоснованно определить возможные ключи отношений.

Определение 4.2.5.15. Множество всех функциональных зависимостей, которые могут быть выведены из заданного множества функциональных зависимостей *F*, называется **замыканием** *F* и записывается как F^+.

Необходимо определить правила, позволяющие вычислить F^+ из *F*. Набор правил вывода, называемый **аксиомами Армстронга**, показывает способы вывода новых функциональных зависимостей из заданных (на слайде они обозначены именами А1, А2 и А3).

Следует отметить, что каждое из этих трех правил можно обосновать исходя непосредственно из определения понятия функциональной зависимости. Этот набор правил является **полным**; это означает, что если задано множество *F* функциональных зависимостей, то все функциональные зависимости, производные от *F*, можно вывести из *F* с помощью только этих правил. Такие правила являются также **непротиворечивыми**, поскольку они не позволяют вывести какие-либо дополнительные функциональные зависимости, которые не следовали бы из *F*. Иными словами, эти правила могут применяться для получения замыкания F^+.

На основе трех правил, приведенных выше, можно вывести несколько дополнительных правил, позволяющих упростить практическую задачу вычисления F^+ (на слайде они обозначены именами П4, П5 и П6).

Правило 1 (рефлективность) указывает, что множество атрибутов всегда функционально определяет любое из своих подмножеств.

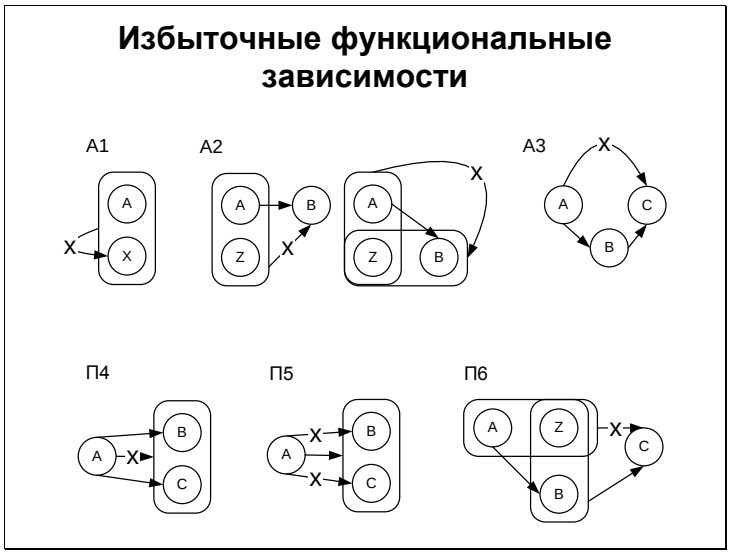
Правило 2 (пополнение) указывает, что добавление одного и того же множества атрибутов и к левой, и к правой частям функциональной зависимости (или только к левой) приводит к получению еще одной действительной зависимости.

Правило 3 (транзитивность) указывает, что функциональные зависимости являются транзитивными.

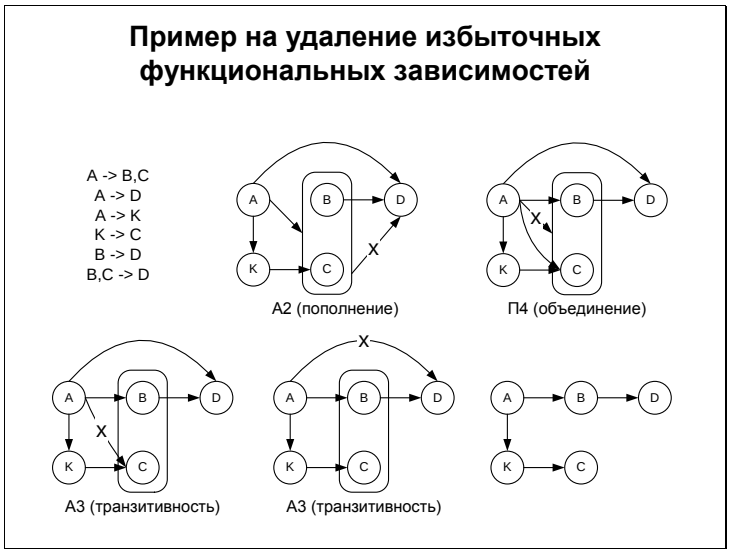
Правило 5 (декомпозиция) определяет, что можно удалять атрибуты из правой части зависимости. Повторное применение этого правила позволяет разложить функциональную зависимость *A -> B, C, D* на ряд функциональных зависимостей *A -> B*, *A -> C* и *A -> D*.

Правило 4 (объединение) указывает, что в процессе проектирования может быть выполнена обратная операция, при которой ряд зависимостей с одинаковыми левыми частями *A -> B*, *A -> C* и *A -> D* объединяется в одну функциональную зависимость *A -> B, C, D*.

Как уже отмечалось, эти же правила вывода (А1, А2, А3, П4, П5 и П6) можно использовать для исключения избыточных ФЗ.



На слайде показаны графические диаграммы ФЗ, демонстрирующие применение правил вывода для удаления избыточных ФЗ. На дуге избыточной ФЗ имеется знак «X». Он сопутствует ФЗ, являющейся следствием соответствующего правила.



В качестве примера на удаление избыточных ФЗ рассмотрим набор ФЗ, показанный на слайде.

Первый рисунок представляет собой графическую диаграмму этого исходного набора ФЗ.

На первом шаге избавляемся от ФЗ *B,C -> D* так как по аксиоме А2 она является следствием *B -> D*.

На следующем шаге заменяем ФЗ *A -> B,C* на пару эквивалентных ей ФЗ *A -> B* и *A -> C* (эту эквивалентность обеспечивают правила П4 и П5).

Вторая из вновь образованных ФЗ (*A -> C*) удаляется, поскольку она является следствием ФЗ *A -> K* и *K -> C* (аксиома А3).

Благодаря применению этой же аксиомы, на последнем шаге удалена ФЗ *A -> D*.

В результате образовался следующий набор ФЗ, неизбыточно характеризующий все зависимости, представленные в исходном наборе:

A -> B,

B -> D,

A -> K,

K -> C.

Минимальное покрытие ФЗ отношения

Набор неизбыточных ФЗ, полученный путем удаления всех избыточных ФЗ из исходного набора с помощью 6 правил вывода, называется минимальным покрытием ФЗ отношения.

Избыточные ФЗ следует удалять по одной, каждый раз заново анализируя новый набор на предмет присутствия в нем избыточных ФЗ. Эта процедура завершается, как только не останется ни одной избыточной ФЗ. Оставшийся набор является минимальным покрытием ФЗ.

Минимальное покрытие не всегда является уникальным, порядок удаления ФЗ может оказать влияние на полученный результат.

Определение 4.2.5.16. Набор неизбыточных ФЗ, полученный путем удаления всех избыточных ФЗ из исходного набора с помощью 6 правил вывода, называется **минимальным покрытием** ФЗ отношения.

Избыточные ФЗ следует удалять по одной, каждый раз заново анализируя новый набор на предмет присутствия в нем избыточных ФЗ. Эта процедура завершается, как только не останется ни одной избыточной ФЗ. Оставшийся набор является минимальным покрытием ФЗ.

Минимальное покрытие не всегда является уникальным, порядок удаления ФЗ может оказать влияние на полученный результат.

И это положение дел является еще одним фактором, который может привести к получению различных схем БД в ходе проектирования.

4.2.5.5. Модификация декомпозиционного алгоритма проектирования

Как вы понимаете, обсуждение избыточных ФЗ вовсе неспроста сопутствует нашему изложению. Анализ и удаление избыточных ФЗ является важным этапом классической методики проектирования РБД, предшествующим собственно декомпозиции.

Поскольку основой декомпозиции являются именно ФЗ, естественно использовать только те из них, которые образуют минимальное покрытие. Это, по крайней мере, исключит заведомо неэффективные варианты схемы, которые в противном случае могут быть рассмотрены.

Для обсуждения второго отрицательного воздействия избыточных ФЗ на процесс проектирования обратимся к примеру декомпозиции отношения *КОНСУЛЬТАНТ*. Вас не смущает тот факт, что в результирующей схеме отсутствует ФЗ *СНОМ -> ТНОМ*? Неужели такое, на первый взгляд безупречное проектирование породило схему, даже не эквивалентную исходной?

Ничего подобного, по той простой причине, что эта ФЗ изначально была избыточной (кстати, единственной). Действительно, в силу аксиомы транзитивности она является следствием зависимостей *СНОМ -> КНОМ* и *КНОМ -> ТНОМ*.

Никуда эта «якобы утерянная» ФЗ из нашей схемы не делась, поскольку в ней явно присутствуют ФЗ *СНОМ -> КНОМ* и *КНОМ -> ТНОМ*, и, благодаря им, СУБД сама позаботится о выполнении их следствия. Но, как видим, не зная про избыточные ФЗ, можно было сделать неправильный вывод о качестве нашей декомпозиции и «сдуру наломать дров».

Таким образом, представим окончательный вид декомпозиционного алгоритма проектирования и заключительные проверки полученной схемы.

Модификация декомпозиционного алгоритма проектирования

1. Построение универсального отношения. Оно пока является единственным в очереди еще не проверенных на НФБК отношений.

2. Определение всех ФЗ, существующих между атрибутами этого отношения.

3. Удаление всех избыточных ФЗ из исходного набора ФЗ с целью получения минимального покрытия.

4. Использование ФЗ из минимального покрытия для декомпозиции универсального отношения в набор НФБК-отношений.

4.1. Определение ФЗ минимального покрытия для очередного отношения.

4.2. Определение того, находится ли очередное отношение в НФБК. Если да, то проектирование для него завершается, если нет – это отношение декомпозируется на два новых, которые помещаются в очередь еще не проверенных на НФБК отношений.

4.3. Повторение шагов 4.1 и 4.2 для каждого очередного отношения. Проектирование завершается, когда очередь еще не проверенных на НФБК отношений опустеет, а, значит, все полученные отношения находятся в НФБК.

Если может быть построено более чем одно минимальное покрытие, осуществляется сравнение результатов, полученных на основе этих минимальных покрытий, с целью определения варианта, лучше других отвечающего требованиям ПрО.

При использовании алгоритма декомпозиции следует помнить о нежелательности проекции, порождаемой ФЗ, у которой зависимая часть целиком или частично является детерминантом другой ФЗ; также повышенное внимание требуется в тех случаях, когда зависимая часть ФЗ зависит более чем от одного детерминанта. В любом из этих случаев может быть утеряна ФЗ из БД. Если в процессе декомпозиции достигнуто состояние, в котором проецирование, не влекущее за собой потерь ФЗ, становится невозможным, проектировщик должен будет сделать выбор из двух альтернатив:

- выбор оставшихся ФЗ и создание одного отношения для каждых детерминанта и набора зависящих от него атрибутов (метод синтеза);
- изменение порядка ранее проведенных декомпозиций, ведь алгоритм проектирования не ведет к единственному решению.

Проверка отношений на завершающей фазе проектирования

1.Составляются списки ФЗ для каждого отношения. Эти списки ФЗ проверяются по двум направлениям:

1.1. Одна и та же ФЗ не должна появляться более чем в одном отношении.

1.2. Набор ФЗ, полученный в результате проектирования, должен в точности совпадать с набором, присутствующем в минимальном покрытии, полученном перед началом проектирования. В противном случае, необходимо показать возможность получения итогового набора ФЗ из минимального покрытия с помощью правил вывода и наоборот.

2.Осуществляется проверка на присутствие избыточных отношений. Если устанавливается избыточность отношения, его следует исключить из проектного набора. Отношение является избыточным, если:

2.1) все атрибуты этого отношения могут быть найдены в одном другом отношении проектного набора;

2.2) все атрибуты этого отношения могут быть найдены в отношении, которое может быть получено из других отношений проектного набора с помощью серии операций соединения над этими отношениями.

3. Отношения рассматриваются с практической точки зрения. Изучается характер использования отношений в конструируемой БД и определяется, будут ли они обеспечивать те типы запросов и операций обновления, которые предполагается выполнять.

Если хотя бы одна из этих проверок окажется недостоверной, следует проанализировать процесс проектирования для выявления ошибок и/или рассмотреть другие варианты проектирования.

После завершения разработки НФБК-отношений, рассматриваемых уже в качестве окончательного проекта, полученный набор необходимо проконтролировать на предмет наличия невыявленных проблем.

Вопросы и задания к пункту 4.2.5

1. Как формулируется задача проектирования реляционной базы данных? Какие цели при этом преследуются?
2. Что такое универсальное отношение?
3. Какие аномалии могут обнаруживаться в неудачно спроектированной базе данных?
4. Что такое первая нормальная форма отношения?
5. Что такое функциональная зависимость?
6. Что такое детерминант атрибута?
7. Что такое нормальная форма Бойса-Кодда?
8. Что такое декомпозиция?
9. Объясните декомпозиционный алгоритм проектирования реляционной схемы.
10. Какими качествами должна обладать декомпозиция?
11. Расскажите о правилах выбора функциональной зависимости на очередном шаге декомпозиции.
12. Что такое метод синтеза отношений?
13. Что такое избыточная функциональная зависимость?
14. Назовите правила вывода функциональных зависимостей.
15. Что такое минимальное покрытие функциональных зависимостей отношения? Как его построить с использованием правил вывода функциональных зависимостей?
16. Как окончательно выглядит декомпозиционный алгоритм проектирования реляционных схем баз данных?
17. Какие проверки отношений следует провести на завершающей фазе проектирования?

Вопросы и задания к параграфу 4.2

1. Укажите основные достоинства реляционного подхода к моделированию данных.
2. Перечислите и дайте определения основных структурных понятий реляционной модели.
3. Какие свойства характерны для отношений реляционной модели?
4. Сформулируйте простейшие правила перехода от ER-схемы Чена к реляционной схеме БД.
5. Что такое представление, и для чего они предназначены?
6. Какие типы ограничений целостности можно декларативно задать в командах языка SQL?
7. Что такое неопределенное значение? Какими свойствами оно обладает?
8. Укажите два основных правила целостности реляционной модели. Как они обеспечиваются?
9. Дайте определения суперключа, потенциального ключа, составного ключа, первичного ключа, альтернативного ключа, суррогатного ключа. Как они соотносятся друг с другом?
10. Почему в последнее время проектировщики предпочитают использовать только суррогатные первичные ключи?
11. Что такое внешний ключ? Должен ли он обладать свойством уникальности? Для чего и как он используется?
12. Что такое триггер? Для чего они предназначены?
13. При каких событиях в системе БД могут запускаться триггеры?
14. Какие факторы влияют на запуск триггеров обновления данных?
15. Чем отличаются триггеры для таблиц от триггеров для представлений?
16. Как в коде триггера можно ссылаться на значения столбцов модифицируемых строк?

17. Какова последовательность выполнения триггеров и основного действия с данными?

18. Когда используется навигационный стиль манипулирования реляционными данными?

19. Что собой представляют курсоры PL/SQL?

20. Какие команды предусмотрены в языке PL/SQL для объявления и обращения к курсорам?

21. Как управлять процессом обращений к курсору с помощью атрибутов курсора?

22. Укажите и охарактеризуйте классы спецификационных языков реляционной модели.

23. Поясните деление языков на процедурные и декларативные.

24. Дайте определение всех операций реляционной алгебры Кодда.

25. Проведите на конкретном примере сравнительный анализ разновидностей операции соединения.

26. Как в языке реляционной алгебры выполняются действия, изменяющие состояние БД?

27. Какой вид имеют запросы в реляционном исчислении с переменными-кортежами?

28. Укажите разновидности атомов формул реляционного исчисления с переменными-кортежами.

29. Перечислите правила построения формул реляционного исчисления с переменными-кортежами.

30. Как определяется статус «связана-свободна» переменных-кортежей?

31. Какой вид имеют запросы в реляционном исчислении с переменными на доменах?

32. Укажите разновидности атомов формул реляционного исчисления с переменными на доменах.

33. Перечислите правила построения формул реляционного исчисления с переменными на доменах.

34. Как определяется статус «связана-свободна» переменных на доменах?

35. Перечислите основные отличительные особенности языка QBE.

36. Каковы грамматические особенности языка QBE?

37. Какие группы полей выделяются в таблице-шаблоне QBE? К каким элементам БД они относятся?

38. Опишите последовательность совместных действий пользователя и системы по формулированию запроса QBE.

39. Какие различные синтаксические конструкции с ключевым словом *SELECT* предусмотрены в стандарте SQL? Для каких ситуаций использования они предназначены?

40. Охарактеризуйте отличительные особенности грамматики каждой из этих *SELECT*-конструкций.

41. В чем заключается основная семантика табличного выражения команды *SELECT*? Из каких разделов оно состоит, и для чего предназначен каждый раздел?

42. Опишите в целом алгоритм вычисления табличного выражения команды *SELECT*.

43. Каковы особенности использования неопределенных значений атрибутов и логического значения *unknown* в запросах SQL?

44. Какие виды атомов (предикатов) предусмотрены в стандарте SQL для логического выражения условия поиска? В каких случаях они принимают значения *true*, *false* и *unknown*?

45. Перечислите и поясните все случаи, при которых вычисление табличного выражения приведет к сгруппированной таблице.

46. Какие дополнительные ограничения накладываются на условие поиска раздела *HAVING* по сравнению с условием поиска раздела *WHERE*?

47. Укажите различные случаи применения агрегатных функций в списке выборки в зависимости от вида табличного выражения.

48. Для решения каких двух задач применим классический подход к проектированию реляционных БД? Охарактеризуйте эти задачи.

49. Какие цели должен преследовать проектировщик реляционной БД?

50. Что такое универсальное отношение?

51. Какие аномалии могут возникать при использовании «некачественных» отношений?

52. Укажите условие первой нормальной формы отношений.

53. Что такое декомпозиция отношения? Для чего она используется? С помощью какой операции над отношениями она осуществляется?

54. Как в теории реляционных БД определяется функциональная зависимость (ФЗ)? Какое отображение стоит за этим понятием?

55. Что является источником информации о ФЗ?

56. Что такое детерминант атрибута?

57. Определите условие нормальной формы Бойса-Кодда (НФБК).

58. Приведите первоначальный алгоритм нормализации отношений до НФБК.

59. Как осуществляется декомпозиция отношения?

60. Укажите желательные свойства декомпозиции. Дайте им определения.

61. Каких правил следует придерживаться при выборе ФЗ для очередной декомпозиции?

62. В чем заключается метод синтеза?

63. Что такое избыточная ФЗ?

64. Перечислите правила вывода ФЗ.

65. Какими свойствами обладают аксиомы Армстронга?

66. Как определять избыточные ФЗ с использованием правил вывода ФЗ?

67. Что такое минимальное покрытие ФЗ отношения?

68. Как окончательно выглядит декомпозиционный алгоритм проектирования реляционных схем БД?

69. Какие проверки отношений следует провести на завершающей фазе проектирования?