Введение в системы управления базами данных
А.Ю.Пушников

Данное учебное пособие было опубликовано в 1999 г. в двух частях издательством Башкирского государственного университета. Выходные данные: УДК 519.6, ББК 32.973-018.2 Пушников А.Ю. Введение в системы управления базами данных. Часть 1. Реляционная модель данных: Учебное пособие/Изд-е Башкирского ун-та. - Уфа, 1999. - 108 с. - ISBN 5-7477-0350-1. В электронной версии - полный текст книги

Web-страница: http://www.citforum.ru/database/dblearn/index.shtml

Целостность реляционных данных

Во второй части реляционной модели данных определяются два ограничения, которые должны выполняться в *любой* реляционной базе данных. Это:

- Целостность сущностей.
- Целостность внешних ключей.

Прежде, чем говорить о целостности сущностей, опишем использование null-значений в реляционных базах данных.

Null-значения

Основное назначение баз данных состоит в том, чтобы хранить и предоставлять информацию о реальном мире. Для представления этой информации в базе данных используются привычные для программистов типы данных - строковые, численные, логические и т.п. Однако в реальном мире часто встречается ситуация, когда данные неизвестны или не полны. Например, место жительства или дата рождения человека могут быть неизвестны (база данных разыскиваемых преступников). Если вместо неизвестного адреса уместно было бы вводить пустую строку, то что вводить вместо неизвестной даты? Ответ - пустую дату - не вполне удовлетворителен, т.к. простейший запрос "выдать список людей в порядке возрастания дат рождения" даст заведомо неправильных ответ.

Для того чтобы обойти проблему неполных или неизвестных данных, в базах данных могут использоваться типы данных, пополненные так называемым ***null-значением***. Null-значение - это, собственно, не значение, а некий маркер, показывающий, что значение неизвестно.

Таким образом, в ситуации, когда возможно появление неизвестных или неполных данных, разработчик имеет на выбор два варианта.

Первый вариант состоит в том, чтобы ограничиться использованием обычных типов данных и не использовать null-значения, а вместо неизвестных данных вводить либо нулевые значения, либо значения специального вида - например, договориться, что строка "АДРЕС НЕИЗВЕСТЕН" и есть те данные, которые нужно вводить вместо неизвестного адреса. В любом случае на пользователя (или на разработчика) ложится ответственность на правильную трактовку таких данных. В частности, может потребоваться написание специального программного кода, который в нужных случаях "вылавливал" бы такие данные. Проблемы, возникающие при этом очевидны - не все данные становятся равноправны, требуется дополнительный программный код, "отслеживающий" эту неравноправность, в результате чего усложняется разработка и сопровождение приложений.

Второй вариант состоит в использовании null-значений вместо неизвестных данных. За кажущейся естественностью такого подхода скрываются менее очевидные и более глубокие проблемы. Наиболее бросающейся в глаза проблемой является необходимость использования трехзначной логики при оперировании с данными, которые могут содержать null-значения. В этом случае при неаккуратном формулировании запросов, даже самые естественные запросы могут давать неправильные ответы. Есть более фундаментальные проблемы, связанные с теоретическим обоснованием корректности введения null-значений, например, непонятно вообще, входят ли null-значения в домены или нет.

Подробное обсуждение проблем использования null-значений выходит за пределы данной работы. Можно только сказать о том, что этот вопрос в теории реляционных баз данных окончательно не решен. Основоположник реляционного подхода Кодд считал null-значения неотъемлемой частью реляционной модели. К.Дейт, один из крупнейших теоретиков реляционной модели выступает категорически против null-значений (подробное обсуждение проблем, возникающих при использовании null-значений приведено в книге [11].

Практически все реализации современных реляционных СУБД позволяют использовать null-значения, несмотря на их недостаточную теоретическую обоснованность. Такую ситуацию можно сравнить с ситуацией, сложившейся в начале века с теорией множеств. Почти сразу после создания Кантором теории множеств, в ней были обнаружены внутренние противоречия (антиномии). Были разработаны более строгие теории, позволяющие избежать этих противоречий (конструктивная теория множеств). Однако в реальной работе большинство математиков пользуется классической теорией множеств, т.к. более строгие теории более ограничены и негибки в применении именно в силу своей большей строгости.

Мнение автора (очень скромное по сравнению с мнением корифеев реляционной теории) состоит в том, что желательно избегать null-значений. Тем не менее, приведем здесь описание трехзначной логики, необходимой для работы с null-значениями.

Трехзначная логика (3VL)

Т.к. null-значение обозначает на самом деле тот факт, что значение неизвестно, то любые алгебраические операции (сложение, умножение, конкатенация строк и т.д.) должны давать также неизвестное значение, т.е. null. Действительно, если, например, вес детали неизвестен, то неизвестно также, сколько весят 10 таких деталей.

При сравнении выражений, содержащих null-значения, результат также может быть неизвестен, например, значение истинности для выражения $a = b$ есть null, если один или оба аргумента есть null. Таким образом, определение истинности логических выражений базируется на ***трехзначной логике*** (***three-valued logic***, ***3VL***), в которой кроме значений T - ИСТИНА и F - ЛОЖЬ, введено значение U - НЕИЗВЕСТНО. Логическое значение U - это то же самое, что и null-значение. Трехзначная логика базируется на следующих таблицах истинности:

AND	**F**	**T**	**U**
F	F	F	F
T	F	T	U
U	F	U	U

Таблица 1 Таблица истинности AND

OR	**F**	**T**	**U**
F	F	T	U
T	T	T	T
U	U	T	U

Таблица 2 Таблица истинности OR

NOT	
F	T
T	F
U	U

Таблица 3 Таблица истинности NOT

Имеется несколько парадоксальных следствий применения трехзначной логики.

Парадокс 1. Null-значение не равно самому себе. Действительно, выражение null = null дает значение не ИСТИНА, а НЕИЗВЕСТНО. Значит выражение $x = x$ не обязательно ИСТИНА!

Парадокс 2. Неверно также, что null-значение не равно самому себе! Действительно, выражение null $\neq$ null также принимает значение не ИСТИНА, а НЕИЗВЕСТНО! Значит также, что и выражение $x \neq x$ тоже не обязательно ЛОЖЬ!

Парадокс 3. $a \text{ or } \text{not}(a)$ не обязательно ИСТИНА. Значит, в трехзначной логике не работает принцип исключенного третьего (любое высказывание либо истинно, либо ложно).

Таких парадоксов можно построить сколько угодно. Конечно, это на самом деле не парадоксы, а просто следствия из аксиом трехзначной логики.

Потенциальные ключи

По определению, тело отношения есть *множество* кортежей, поэтому отношения не могут содержать одинаковые кортежи. Это значит, что каждый кортеж должен обладать *свойством уникальности*. На самом деле, свойством уникальности в пределах отношения могут обладать отдельные атрибуты кортежей или группы атрибутов. Такие уникальные атрибуты удобно использовать для идентификации кортежей.

Определение 1. Пусть дано отношение R. Подмножество атрибутов K отношения R будем называть ***потенциальным ключом***, если K обладает следующими свойствами:

1. ***Свойством уникальности*** - в отношении не может быть двух различных кортежей, с одинаковым значением K.
2. ***Свойством неизбыточности*** - никакое подмножество в K не обладает свойством уникальности.

Любое отношение имеет по крайней мере один потенциальный ключ. Действительно, если никакой атрибут или группа атрибутов не являются потенциальным ключом, то, в силу уникальности кортежей, все атрибуты вместе образуют потенциальный ключ.

Потенциальный ключ, состоящий из одного атрибута, называется ***простым***. Потенциальный ключ, состоящий из нескольких атрибутов, называется ***составным***.

Отношение может иметь несколько потенциальных ключей. Традиционно, один из потенциальных ключей объявляется ***первичным***, а остальные - ***альтернативными***. Различия между первичным и альтернативными ключами могут быть важны в конкретной реализации реляционной СУБД, но с точки зрения реляционной модели данных, нет оснований выделять таким образом один из потенциальных ключей.

Замечание. Понятие потенциального ключа является *семантическим* понятием и отражает некоторый смысл (трактовку) понятий из конкретной предметной области. Для того чтобы проиллюстрировать этот факт рассмотрим следующее отношение "Сотрудники":

Табельный номер	Фамилия	Зарплата
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000

Таблица 4 Отношение "Сотрудники"

При первом взгляде на таблицу, изображающую это отношение, может показаться, что в таблице имеется три потенциальных ключа - в каждой колонке таблицы содержатся

уникальные данные. Однако среди сотрудников могут быть однофамильцы и сотрудники с одинаковой зарплатой. Табельный же номер по сути свой уникален для каждого сотрудника. Какие же соображения привели нас к пониманию того, что в данном отношении только один потенциальный ключ - "Табельный номер"? Именно *понимание смысла данных*, содержащихся в отношении.

Попробуем представить это отношение в другом виде, изменив наименования атрибутов:

A	**B**	**C**
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000

Предъявим кому-нибудь эту таблицу и не сообщим смысл наименований атрибутов. Очевидно, что невозможно судить, не понимая смысла данных, может или не может в этом отношении появиться, например, кортеж (1, Петров, 3000). Если бы, кстати, такой кортеж появился (что, на первый взгляд, вполне возможно, т.к. не нарушается уникальность кортежей), то мы точно смогли бы сказать, что *не является* альтернативным ключом - ни один из атрибутов по отдельности. Но мы не сможем сказать, что же *является* первичным ключом.

Замечание. Потенциальные ключи служат ***средством идентификации*** объектов предметной области, данные о которых хранятся в отношении. Объекты предметной области должны быть различимы.

Замечание. Потенциальные ключи служат единственным ***средством адресации на уровне кортежей*** в отношении. Точно указать какой-нибудь кортеж можно только зная значение его потенциального ключа.

Целостность сущностей

Т.к. потенциальные ключи фактически служат идентификаторами объектов предметной области (т.е. предназначены для *различения* объектов), то значения этих идентификаторов не могут содержать неизвестные значения. Действительно, если бы идентификаторы могли содержать null-значения, то мы не могли бы дать ответ "да" или "нет" на вопрос, совпадают или нет два идентификатора.

Это определяет следующее ***правило целостности сущностей***:

Правило целостности сущностей. Атрибуты, входящие в состав некоторого потенциального ключа не могут принимать null-значений.

Внешние ключи

Различные объекты предметной области, информация о которых хранится в базе данных, всегда взаимосвязаны друг с другом. Например, накладная на поставку товара *содержит* список товаров с количествами и ценами, сотрудник предприятия *имеет* детей, *числится* в подразделении и т.д. Термины "содержит", "имеет", "числится" отражают взаимосвязи между понятиями "накладная" и "список товаров", "сотрудник" и "дети", "сотрудник" и

"подразделение". Такие взаимосвязи отражаются в реляционных базах данных при помощи ***внешних ключей***, связывающих несколько отношений.

Рассмотрим пример с поставщиками и поставками деталей. Предположим, что нам требуется хранить информацию о наименовании поставщиков, наименовании и количестве поставляемых ими деталей, причем каждый поставщик может поставлять несколько деталей и каждая деталь может поставляться несколькими поставщиками. Можно предложить хранить данные в следующем отношении:

Номер поставщика	**Наименование поставщика**	***Номер детали***	**Наименование детали**	**Поставляемое количество**
1	Иванов	*1*	Болт	100
1	Иванов	*2*	Гайка	200
1	Иванов	*3*	Винт	300
2	Петров	*1*	Болт	150
2	Петров	*2*	Гайка	250
3	Сидоров	*3*	Винт	1000

Таблица 5 Отношение "Поставщики и поставляемые детали"

Потенциальным ключом этого отношения может выступать пара атрибутов {"Номер поставщика", "Номер детали"} - в таблице они выделены курсивом.

Приведенный способ хранения данных обладает рядом недостатков.

Что произойдет, если изменилось наименование поставщика? Т.к. наименование поставщика повторяется во многих кортежах отношения, то это наименование нужно *одновременно* изменить *во всех* кортежах, где оно встречается, иначе данные станут противоречивыми. То же самое с наименованиями деталей. Значит, данные хранятся в нашем отношении с большой избыточностью.

Далее, как отразить факт, что некоторый поставщик, например Петров, временно прекратил поставки деталей? Если мы удалим все кортежи, в которых хранится информация о поставках этого поставщика, то мы *потеряем данные* о самом Петрове как потенциальном поставщике. Выйти из этого положения, оставив в отношении кортеж типа (2, Петров, NULL, NULL, NULL) мы не можем, т.к. атрибут "Номер детали" входит в состав потенциального ключа и не может содержать null-значений. То же самое произойдет, если некоторая деталь временно не поставляется никаким поставщиком. Получается, что мы не можем хранить информацию о том, что есть некий поставщик, если он не поставляет хотя бы одну деталь, и не можем хранить информацию о том, что есть некоторая деталь, если она никем не поставляется.

Подобные проблемы возникают потому, что мы смешали в одном отношении различные объекты предметной области - и данные о поставщиках, и данные о деталях, и данные о поставках деталей. Говорят, что это отношение *плохо нормализовано* (просто нормализованным оно является хотя бы потому, что оно есть отношение и, следовательно, автоматически находится в 1НФ).

О том, как правильно нормализовать отношения, будет сказано в следующих главах, сейчас же предложим разнести данные по трем отношениям - "Поставщики", "Детали", "Поставки". Для нас важно выяснить, каким образом данные, хранящиеся в этих отношениях взаимосвязаны друг с другом. Эта связь определяется семантикой предметной области и описывается фразами: "Поставщики *выполняют* Поставки", "Детали *поставляются через* Поставки". Эти две взаимосвязи косвенно определяют новую взаимосвязь между "Поставщиками" и "Деталями": "Детали *поставляются* Поставщиками".

Эти фразы отражают различные типы взаимосвязей. Чтобы более точно отразить предметную область, можно иначе переформулировать фразы: "*Один* Поставщик может выполнять *несколько* Поставок", "*Одна* Деталь может поставляться *несколькими* Поставками". Это пример взаимосвязи типа "***один-ко-многим***".

Взаимосвязь между "Поставщиками" и "Деталями" можно переформулировать так: "*Несколько* Деталей может поставляться *несколькими* Поставщиками". Это пример взаимосвязи типа "***много-ко-многим***".

В реляционных базах данных основными являются взаимосвязи типа "один-ко-многим". Взаимосвязи типа "много-ко-многим" реализуются использованием нескольких взаимосвязей типа "один-ко-многим". Отношение, входящее в связь со стороны "один" (например, "Поставщики"), называют ***родительским отношением***. Отношение, входящее в связь со стороны "много" (например, "Поставки"), называется ***дочернем отношением***.

Механизм реализации взаимосвязи "один-ко-многим" состоит в том, что в дочернее отношение добавляются атрибуты, являющиеся ссылками на ключевые атрибуты родительского отношения. Эти атрибуты и являются внешними ключами, определяющими, с какими кортежами родительского отношения связаны кортежи дочернего отношения. Такие атрибуты еще называют ***мигрирующими*** из родительского отношения.

Таким образом, наш пример с поставщиками и поставляемыми деталями должен выглядеть следующим образом:

Номер поставщика	**Наименование поставщика**
1	Иванов
2	Петров
3	Сидоров

Таблица 6 Отношение "Поставщики"

Номер детали	**Наименование детали**
1	Болт
2	Гайка
3	Винт

Таблица 7 Отношение "Детали"

Номер поставщика	*Номер детали*	Поставляемое количество
1	*1*	100
1	*2*	200
1	*3*	300
2	*1*	150
2	*2*	250
3	*3*	1000

Таблица 8 Отношение "Поставки"

В отношении "Поставки" атрибуты "Номер поставщика" и "Номер детали" являются ссылками на ключевые атрибуты отношений "Поставщики" и "Детали", и, следовательно, являются внешними ключами. Заметим, что данные отношения свободны от недостатков, описанных выше, когда все данные предлагалось хранить в одном отношении. Действительно, при изменении наименования поставщика или детали, это изменение происходит только в *одном месте*. Если поставщик прекратил поставки всех деталей, то удаляются соответствующие кортежи в отношении "Поставки", данные же о самом поставщике остаются *без изменений*.

Дадим точное определение.

Определение 2. Пусть дано отношение R. Подмножество атрибутов FK отношения R будем называть ***внешним ключом***, если:

1. Существует отношение S (R и S не обязательно различны) с потенциальным ключом K.
2. Каждое значение FK в отношении R всегда совпадает со значением K для некоторого кортежа из S, либо является null-значением.

Отношение S называется ***родительским отношением***, отношение R называется ***дочерним отношением***.

Замечание. Внешний ключ, также как и потенциальный, может быть простым и составным.

Замечание. Внешний ключ должен быть определен на тех же доменах, что и соответствующий первичный ключ родительского отношения.

Замечание. Внешний ключ, как правило, *не обладает свойством уникальности*. Так и должно быть, т.к. в дочернем отношении может быть несколько кортежей, ссылающихся на один и тот же кортеж родительского отношения. Это, собственно, и дает тип отношения "один-ко-многим".

Замечание. Если внешний ключ все-таки обладает свойством уникальности, то связь между отношениями имеет тип "***один-к-одному***". Чаще всего такие отношения объединяются в одно отношение, хотя это и не обязательно.

Замечание. Хотя каждое значение внешнего ключа обязано совпадать со значениями потенциального ключа в некотором кортеже родительского отношения, то обратное, вообще говоря, неверно. Например, могут существовать поставщики, не поставляющие никаких деталей.

Замечание. Для внешнего ключа не требуется, чтобы он был компонентом некоторого потенциального ключа (как получилось в примере с поставщиками и деталями).

Замечание. Null-значения для атрибутов внешнего ключа допустимы только в том случае, когда атрибуты внешнего ключа не входят в состав никакого потенциального ключа

Целостность внешних ключей

Т.к. внешние ключи фактически служат ссылками на кортежи в другом (или в том же самом) отношении, то эти ссылки не должны указывать на несуществующие объекты. Это определяет следующее ***правило целостности внешних ключей***:

Правило целостности внешних ключей. Внешние ключи не должны быть несогласованными, т.е. для каждого значения внешнего ключа должно существовать соответствующее значение первичного ключа в родительском отношении.

Замечания к правилам целостности сущностей и внешних ключей

На самом деле приведенные правила целостности сущностей и внешних ключей прямо следуют из определений понятий "потенциальный ключ" и "внешний ключ".

Действительно, в определении потенциального ключа требуется, чтобы потенциальный ключ обладал свойством уникальности. Это фактически означает, что мы должны уметь различать значения потенциальных ключей, т.е. при сравнении двух значений потенциального ключа мы *всегда должны* получать значения либо ИСТИНА, либо ЛОЖЬ. Но любое сравнение, в которое входит null-значение, принимает значение U - НЕИЗВЕСТНО, откуда следует, что атрибуты потенциального ключа не могут содержать null-значений.

Для внешних ключей правило целостности фактически входит в определение (п. 2 определения 2).

Таким образом, с точки зрения реляционной теории, явная формулировка правил целостности является излишней - они автоматически вытекают из определений понятий ключа и внешнего ключа.

Тем не менее, явная формулировка правил целостности имеет определенный практический смысл. В большинстве серьезных СУБД за выполнением этих ограничений следит сама СУБД, если, конечно, пользователь явно объявил потенциальные и внешние ключи. Но, во-первых, для некоторых систем можно допустить, чтобы эти ограничения не выполнялись, а во-вторых, некоторые системы просто не поддерживают понятия целостности. В этих случаях за целостностью данных должен следить сам пользователь, или программист, разрабатывающий приложение для пользователя.

Явная формулировка правил целостности помогает четко понять, какие опасности несет в себе пренебрежение этими правилами.

Операции, могущие нарушить ссылочную целостность

Ссылочная целостность может нарушиться в результате операций, изменяющих состояние базы данных. Таких операций три - вставка, обновление и удаление кортежей в отношениях. Т.к. в определении ссылочной целостности участвуют два отношения - родительское и дочернее, а в каждом из них возможны три операции - вставка, обновление, удаление, то нужно рассмотреть шесть различных вариантов.

Для родительского отношения

- *Вставка кортежа в родительском отношении*. При вставке кортежа в родительское отношение возникает новое значение потенциального ключа. Т.к. допустимо существование кортежей в родительском отношении, на которые нет ссылок из дочернего отношения, то вставка кортежей в родительское отношение *не нарушает ссылочной целостности*.
- *Обновление кортежа в родительском отношении*. При обновлении кортежа в родительском отношении может измениться значение потенциального ключа. Если есть кортежи в дочернем отношении, ссылающиеся на обновляемый кортеж, то значения их внешних ключей станут некорректными. Обновление кортежа в родительском отношении *может привести к нарушению ссылочной целостности*, если это обновление затрагивает значение потенциального ключа.
- *Удаление кортежа в родительском отношении*. При удалении кортежа в родительском отношении удаляется значение потенциального ключа. Если есть кортежи в дочернем отношении, ссылающиеся на удаляемый кортеж, то значения их внешних ключей станут некорректными. Удаление кортежей в родительском отношении *может привести к нарушению ссылочной целостности*.

Для дочернего отношения

- *Вставка кортежа в дочернее отношение*. Нельзя вставить кортеж в дочернее отношение, если вставляемое значение внешнего ключа некорректно. Вставка кортежа в дочернее отношение *привести к нарушению ссылочной целостности*.
- *Обновление кортежа в дочернем отношении*. При обновлении кортежа в дочернем отношении можно попытаться некорректно изменить значение внешнего ключа. Обновление кортежа в дочернем отношении *может привести к нарушению ссылочной целостности*.
- *Удаление кортежа в дочернем отношении*. При удалении кортежа в дочернем отношении *ссылочная целостность не нарушается*.

Таким образом, ссылочная целостность в принципе может быть нарушена при выполнении одной из четырех операций:

- Обновление кортежа в родительском отношении.
- Удаление кортежа в родительском отношении.
- Вставка кортежа в дочернее отношение.
- Обновление кортежа в дочернем отношении.

Стратегии поддержания ссылочной целостности

Существуют ***две основные стратегии поддержания ссылочной целостности***:

- ***RESTRICT (ОГРАНИЧИТЬ)***- не разрешать выполнение операции, приводящей к нарушению ссылочной целостности. Это самая простая стратегия, требующая только проверки, имеются ли кортежи в дочернем отношении, связанные с некоторым кортежем в родительском отношении.
- ***CASCADE (КАСКАДИРОВАТЬ)***- разрешить выполнение требуемой операции, но внести при этом необходимые поправки в других отношениях так, чтобы не допустить нарушения ссылочной целостности и сохранить все имеющиеся связи. Изменение начинается в родительском отношении и каскадно выполняется в дочернем отношении. В реализации этой стратегии имеется одна тонкость, заключающаяся в том, что дочернее отношение само может быть родительским для некоторого третьего отношения. При этом может дополнительно потребоваться выполнение какой-либо стратегии и для этой связи и т.д. Если при этом какая-либо из каскадных операций (любого уровня) не может быть выполнена, то необходимо отказаться от первоначальной операции и вернуть базу данных в исходное состояние. Это самая сложная стратегия, но она хороша тем, что при этом не нарушается связь между кортежами родительского и дочернего отношений.

Эти стратегии являются стандартными и присутствуют во всех СУБД, в которых имеется поддержка ссылочной целостности.

Можно рассмотреть ***дополнительные стратегии поддержания ссылочной целостности***:

- ***SET NULL (УСТАНОВИТЬ В NULL)*** - разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей изменять на null-значения. Эта стратегия имеет два недостатка. Во-первых, для нее требуется допустить использование null-значений. Во-вторых, кортежи дочернего отношения теряют всякую связь с кортежами родительского отношения. Установить, с каким кортежем родительского отношения были связаны измененные кортежи дочернего отношения, после выполнения операции уже нельзя.
- ***SET DEFAULT (УСТАНОВИТЬ ПО УМОЛЧАНИЮ)*** - разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей изменять на некоторое значение, принятое по умолчанию. Достоинство этой стратегии по сравнению с предыдущей в том, что она позволяет не пользоваться null-значениями. Недостатки заключаются в следующем. Во-первых, в родительском отношении должен быть некий кортеж, потенциальный ключ которого принят как значение по умолчанию для внешних ключей. В качестве такого "кортежа по умолчанию" обычно принимают специальный кортеж, заполненный нулевыми значениями (не null-значениями!). Этот кортеж *нельзя удалять* из родительского отношения, и в этом кортеже *нельзя изменять* значение потенциального ключа. Таким образом, не все кортежи родительского отношения становятся равнозначными, поэтому приходится прилагать дополнительные усилия для отслеживания этой неравнозначности. Это плата за отказ от использования null-значений. Во-вторых, как и в предыдущем случае, кортежи дочернего отношения теряют всякую связь с кортежами родительского отношения. Установить, с каким кортежем родительского отношения были связаны измененные кортежи дочернего отношения, после выполнения операции уже нельзя.

В некоторых реализация СУБД рассматривается ***еще одна стратегия поддержания ссылочной целостности***:

- ***IGNORE (ИГНОРИРОВАТЬ)*** - выполнять операции, не обращая внимания на нарушения ссылочной целостности.

Конечно, это не стратегия, а отказ от поддержки ссылочной целостности. В этом случае в дочернем отношении могут появляться некорректные значения внешних ключей, и вся ответственность за целостность базы данных ложится на пользователя.

В дополнение к приведенным стратегиям пользователь может придумать свою уникальную стратегию поддержания ссылочной целостности.

Применение стратегий поддержания ссылочной целостности

Рассмотрим, как применяются стратегии поддержания ссылочной целостности при выполнении операций модификации базы данных.

При обновлении кортежа в родительском отношении

Допустимые стратегии:

RESTRICT (ОГРАНИЧИТЬ) - не разрешать обновление, если имеется хотя бы один кортеж в дочернем отношении, ссылающийся на обновляемый кортеж.

CASCADE (КАСКАДИРОВАТЬ) - выполнить обновление и каскадно изменить значения внешних ключей во всех кортежах дочернего отношения, ссылающихся на обновляемый кортеж.

SET NULL (УСТАНОВИТЬ В NULL) - выполнить обновление и во всех кортежах дочернего отношения, ссылающихся на обновляемый кортеж, изменить значения внешних ключей на null-значение.

SET DEFAULT (УСТАНОВИТЬ ПО УМОЛЧАНИЮ) - выполнить обновление и во всех кортежах дочернего отношения, ссылающихся на обновляемый кортеж, изменить значения внешних ключей на некоторое значение, принятое по умолчанию.

IGNORE (ИГНОРИРОВАТЬ) - выполнить обновление, не обращая внимания на нарушения ссылочной целостности.

При удалении кортежа в родительском отношении

Допустимые стратегии:

RESTRICT (ОГРАНИЧИТЬ) - не разрешать удаление, если имеется хотя бы один кортеж в дочернем отношении, ссылающийся на удаляемый кортеж.

CASCADE (КАСКАДИРОВАТЬ) - выполнить удаление и каскадно удалить кортежи в дочернем отношении, ссылающиеся на удаляемый кортеж.

SET NULL (УСТАНОВИТЬ В NULL) - выполнить удаление и во всех кортежах дочернего отношения, ссылающихся на удаляемый кортеж, изменить значения внешних ключей на null-значение.

SET DEFAULT (УСТАНОВИТЬ ПО УМОЛЧАНИЮ) - выполнить удаление и во всех кортежах дочернего отношения, ссылающихся на удаляемый кортеж, изменить значения внешних ключей на некоторое значение, принятое по умолчанию.

IGNORE (ИГНОРИРОВАТЬ) - выполнить удаление, не обращая внимания на нарушения ссылочной целостности.

При вставке кортежа в дочернее отношение

Допустимые стратегии:

RESTRICT (ОГРАНИЧИТЬ) - не разрешать вставку, если внешний ключ во вставляемом кортеже не соответствует ни одному значению потенциального ключа родительского отношения.

SET NULL (УСТАНОВИТЬ В NULL) - вставить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а null-значение.

SET DEFAULT (УСТАНОВИТЬ ПО УМОЛЧАНИЮ) - вставить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а некоторое значение, принятое по умолчанию.

IGNORE (ИГНОРИРОВАТЬ) - вставить кортеж, не обращая внимания на нарушения ссылочной целостности.

При обновлении кортежа в дочернем отношении

Допустимые стратегии:

RESTRICT (ОГРАНИЧИТЬ) - не разрешать обновление, если внешний ключ в обновляемом кортеже становится не соответствующим ни одному значению потенциального ключа родительского отношения.

SET NULL (УСТАНОВИТЬ В NULL) - обновить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а null-значение.

SET DEFAULT (УСТАНОВИТЬ ПО УМОЛЧАНИЮ) - обновить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а некоторое значение, принятое по умолчанию.

IGNORE (ИГНОРИРОВАТЬ) - обновить кортеж, не обращая внимания на нарушения ссылочной целостности.

Выводы

Современные СУБД допускают использование ***null-значений***, т.к. данные часто бывают неполными или неизвестными. Споры о допустимости использования null-значений ведутся до сих пор. Использование null-значения связано с применением ***трехзначной логики*** (***three-valued logic***, ***3VL***).

Средством, позволяющим однозначно идентифицировать кортежи отношения, являются ***потенциальные ключи отношения***.

Потенциальный ключ отношения - это набор атрибутов отношения, обладающий свойствами ***уникальности*** и ***неизбыточности***. Доступ к конкретному кортежу можно получить, лишь зная значение потенциального ключа для этого кортежа.

Традиционно один из потенциальных ключей объявляется ***первичным ключом***, остальные - ***альтернативными ключами***.

Потенциальный ключ, состоящий из одного атрибута, называется ***простым***.
Потенциальный ключ, состоящий из нескольких атрибутов, называется ***составным***.

Отношения связываются друг с другом при помощи ***внешних ключей***.

Внешний ключ отношения - это набор атрибутов отношения, содержащий ссылки на потенциальный ключ другого (или того же самого) отношения. Отношение, содержащее потенциальный ключ, на который ссылается некоторый внешний ключ, называется ***родительским отношением***. Отношение, содержащее внешний ключ, называется ***дочерним отношением***.

Внешний ключ не обязан обладать свойством уникальности. Поэтому, одному кортежу родительского отношения может соответствовать несколько кортежей дочернего отношения. Такой тип связи между отношениями называется "***один-ко-многим***".

Связи типа "***много-ко-многим***" реализуются использованием нескольких отношений типа "один-ко-многим".

В любой реляционной базе данных должны выполняться два ограничения:

- Целостность сущностей
- Целостность внешних ключей

Правило ***целостности сущностей*** *состоит в том, что а*трибуты, входящие в состав некоторого потенциального ключа не могут принимать null-значений.

Правило ***целостности внешних ключей*** состоит в том, что внешние ключи не должны ссылаться на отсутствующие в родительском отношении кортежи, т.е. внешние ключи должны быть корректны.

Ссылочную целостность могут нарушить операции, изменяющие состояние базы данных. Такими операциями являются операции вставки, обновления и удаления кортежей.

Для поддержания ссылочной целостности обычно используются две ***основные стратегии***:

- ***RESTRICT (ОГРАНИЧИТЬ)*** - не разрешать выполнение операции, приводящей к нарушению ссылочной целостности.
- ***CASCADE (КАСКАДИРОВАТЬ)*** - разрешить выполнение требуемой операции, но внести каскадные изменения в другие отношения так, чтобы не допустить нарушения ссылочной целостности.

Дополнительными стратегиями поддержания ссылочной целостности являются:

- ***SET NULL (УСТАНОВИТЬ В NULL)*** - все некорректные значения внешних ключей изменять на null-значения.
- ***SET DEFAULT (УСТАНОВИТЬ ПО УМОЛЧАНИЮ)*** - все некорректные значения внешних ключей изменять на некоторое значение, принятое по умолчанию.

В реальных СУБД можно также отказаться от использования какой-либо стратегии поддержания ссылочной целостности:

- ***IGNORE (ИГНОРИРОВАТЬ)*** - выполнять операции, не обращая внимания на нарушения ссылочной целостности.

Пользователь может разработать свою уникальную стратегию поддержания ссылочной целостности.