Реляционные модели

# Неформальное введение в реляционную модель

Формальное определение реляционной модели основано на теории множеств (математической моделью таблицы является отношение - relation, отсюда и произошел термин реляционная база данных). Е.Ф. Кодд определил систему операций над отношениями (реляционную алгебру) и сформулировал основные правила поддержки целостности реляционной базы данных. Им же был предложен и язык для манипулирования реляционной базой данных, который тогда получил название SEQUEL, а впоследствии превратился в язык SQL. Изучение языка SQL невозможно без знания основ реляционной модели, которую он полностью поддерживает, поэтому далее приведено ее формальное описание, основанное на работах Е.Ф.Кодда и К.Дж.Дейта. В качестве введения в реляционную модель приведем неформальные, но достаточно строгие определения.

## Таблицы и связи

Кодд выделил 12 принципов (правил) реляционных баз данных. Начнем с первого правила, которое позволит понять суть реляционной модели:

*• вся информация логически представлена в виде таблиц.*

Таблица состоит из строк и столбцов (в реляционной теории строке соответствует кортеж, а столбцу — атрибут, однако в стандарте SQL используются общепринятые термины «строка» и «столбец»).

Поскольку вся информация, хранимая в базе данных, не может быть размещена в одной таблице, возникает вопрос, как организовать связи между таблицами. Согласно первому правилу Кодда, вся информация, в том числе и информация о связях между данными, должна содержаться в самих таблицах. В реляционной модели для этого используются *общие столбцы таблиц*.

Например, пусть в базе данных коммерческой фирмы необходимо хранить ряд сведений о клиентах фирмы:

* порядковый номер (личный код) клиента;
* фамилия, имя, отчество;
* контактные телефоны.

При попытке разместить все эти данные в одной таблице возникает серьезная проблема со столбцом «контактные телефоны». Обычно клиенты оставляют несколько номеров - рабочий, домашний, мобильный телефон и т.д., причем количество номеров может быть различным (допустима даже такая ситуация, когда клиент вообще не предоставляет ни одного номера). Наилучшим способом для хранения контактных телефонов будет отдельная (детальная или подчиненная) таблица, связанная с основной таблицей данных о клиентах при помощи общего столбца «личный код клиента». Возможный вариант заполнения таблиц показан на рис. 1

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Рис. 1. Информация о клиентах с указанием контактных телефонов

В данном примере клиенту с кодом 1 соответствуют три связанных строки в таблице Контактные телефоны (у них всех в столбце код стоит значение 1), а клиенту с кодом 2 - только одна связанная строка. Получилась очень стройная и логичная структура из двух связанных таблиц, с помощью которой одинаково легко найти как все номера телефонов конкретного клиента, так и определить, какому клиенту принадлежит тот или иной телефон.

Рассмотрим еще один пример организации связи между таблицами в реляционной базе данных, например, товары и поставщики. Пусть необходимо хранить сведения о товарах и их поставщиках. В данном случае связь между товарами и поставщиками несколько сложнее, чем между клиентами и их телефонами. Каждый товар может поставлять, как правило, несколько фирм-поставщиков, но и каждая такая фирма обычно поставляет несколько товаров. В этом случае наилучшим решением будет создание еще одной таблицы, которую называют таблицей-связкой, для хранения информации о связях между товарами и их поставщиками. Пример заполнения такой базы данных показан на рис. 2, здесь таблица-связка названа, как обычно принято. Товары-поставщики и содержит всего два столбца - код товара и код поставщика. В таблицах-связках могут быть и дополнительные, уточняющие, столбцы, например цена данного товара у данного поставщика и т.д.

Изображение выглядит как текст, Шрифт, диаграмма, число

Автоматически созданное описание

Рис. 2. Информация о товарах и их поставщиках

## Первичные, альтернативные и внешние ключи

Обобщим материал приведенных примеров. Рассмотрим, например, связь между таблицами Клиенты и Контактные телефоны. В таблице Клиенты столбец «код» является уникальной характеристикой каждого клиента, такой столбец называется *первичным ключом (Primary Key - РК)*. Прилагательное «первичный» добавляется в связи с тем, что уникальных столбцов или уникальных сочетаний нескольких столбцов в реальных таблицах может быть несколько. Например, в таблицу Клиенты можно было бы добавить столбец серия и номер паспорта (или два столбца - серия и номер отдельно), и это был бы *альтернативный ключ* таблицы (если уникальный ключ состоит из нескольких столбцов, то он называется составным). Заметим, что сочетание фамилии, имени и отчества нельзя считать альтернативным ключом ввиду наличия полных однофамильцев Сочетания двух столбцов «код» и «фамилия» или «код» и «имя», конечно, обладают свойством уникальности, но это явно избыточные сочетания (фамилию или имя можно отбросить без потери уникальности). Таким образом, ключ таблицы должен удовлетворять двум признакам — уникальности и безизбыточности.

Второе правило Кодда гласит — *логический доступ к данным осуществляется по имени таблицы, имени столбца и значению первичного ключа*. Эти три составляющих позволяют однозначно получить любой элемент реляционной базы данных. Например, по таблице клиенты, столбцу фамилия и значению кода 2 легко получить фамилию Петров.

Связь главной и подчиненной таблиц обычно осуществляется с помощью первичного ключа главной таблицы, который помещается (экспортируется) в подчиненную таблицу и становится там *внешним ключом (Foreign Key - FK)*. Внешний ключ не обладает свойством уникальности (каждому клиенту может соответствовать несколько номеров телефонов, каждому поставщику - несколько товаров), обычно он является частью составного первичного ключа или неключевым столбцом. В примере с клиентами и телефонами подчиненная таблица имеет составной первичный ключ «код» и «номер телефона» – вместе они образуют уникальное и безизбыточное сочетание. Связь между таблицами Клиенты и Контактные телефоны называют связью «один ко многим» (тут прямая аналогия с иерархической и сетевой базами данных). В примере с товарами и поставщиками, как правило, наименования всех товаров и названия всех фирм-поставщиков сами по себе уникальны. Однако введение дополнительных ключевых столбцов, иначе называемых *суррогатными ключами*, является повсеместной практикой. Допустим, если какая-либо фирма-поставщик изменила название, чтобы отразить это изменение в базе данных, достаточно исправить всего одно значение в таблице Поставщики. Изменять суррогатный ключ при этом не нужно, поэтому информация в таблице-связке останется прежней.

## NULL-значения

*NULL-значения— это неопределенные или пустые значения данных, которые в реляционной теории трактуются как отсутствие информации* (правило Кодда №3). Их нельзя рассматривать как нулевые значения числовых полей или пустые строки в текстовых полях. Допустимость пустых значений в том или ином столбце необходимо указывать при определении таблиц. В отношении ключевых столбцов справедливы следующие правила:

* в первичном и альтернативном ключах NULL-значения не допускаются;
* во внешнем ключе наличие NULL-значений допустимо, однако при определении таблицы их можно запретить.

Существуют определенные правила обработки NULL-значений. Два NULL-значения никогда не равны друг другу.

## Метаданные. Схема базы данных

Описание структуры базы данных называется метаданными (данные о данных). *Метаданные должны храниться в базе данных, как и все прочие значения, а не в прикладной программе* (правило Кодда №4). Для наглядного представления структуры базы данных, удобного для пользователей, используется ее графическое изображение, которое называется схемой. Изобразим логическую схему базы данных Сведения о клиентах, воспользовавшись международным стандартом IDEF1X рис. 3.

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рис. 3. Логическая схема базы данных Сведения о клиентах

Как видно из схемы, главная и подчиненная таблицы неразрывно связаны друг с другом, и эта связь имеет логический характер, не зависящий от физического способа реализации таблиц.

## Правила ссылочной целостности

Внешний ключ подчиненной таблицы можно рассматривать как ссылку на строку главной таблицы с таким же значением первичного ключа. Отсюда вытекает *основное правило ссылочной целостности – все значения внешнего ключа должны быть согласованы с соответствующими значениями первичного ключа*. Применительно к нашему примеру это означает, что в таблице контактные телефоны не должно быть кодов клиентов, которых нет в основной таблице клиенты. *Ссылочная целостность должна жестко контролироваться при выполнении любых операций с данными таблиц* (правило Кодда №12). Правило Кодда №7 устанавливает 4 таких операции - *извлечение, вставка новых строк, удаление строк и изменение (обновление) существующих строк (операции select, insert, delete, update)*. Операция извлечения не может нарушить целостности, т.к. не изменяет данные. Рассмотрим основные стратегии поддержки ссылочной целостности для остальных операций. Операция вставки критична только для подчиненной таблицы (можно добавить нового клиента, у которого нет телефона, но нельзя добавить телефон несуществующего клиента), поэтому вставка новых строк в подчиненную таблицу должна проверяться на согласованность значений внешнего ключа. Удаление строк подчиненной таблицы, наоборот, абсолютно безопасно (можно удалить любой телефон, не нарушив ссылочной целостности). Однако удаление строк главной таблицы при наличии связанных строк в подчиненной таблице непременно приведет к нарушению ссылочной целостности, чего допустить нельзя ни в коем случае. Здесь имеется две основных стратегии удаления:

• запрет удаления таких строк (ограничение удаления - restrict),

• каскадное удаление строки главной таблицы вместе со всеми связанными строками подчиненной таблицы (удаление каскадом - cascade).

В рассмотренном примере с телефонами разумным решением будет каскадное удаление (при удалении клиента автоматически должны удаляться и все строки с его телефонами). В реальной практике чаще применятся запрет удаления строк при наличии на них хотя бы одной ссылки из других таблиц. Отметим, что ни одна СУБД не позволит удалить всю таблицу целиком, если имеется хотя бы одна другая таблица, которая на нее ссылается. Изменение (обновление) значений внешнего, а особенно первичного ключа заполненной базы данных обычно не происходит, однако стратегия каскадного обновления некоторыми СУБД поддерживается. Далее реляционная модель данных будет определена более формально. Наличие формального математического описания считается одним из достоинств реляционной модели.

# Реляционная модель. Структурная и целостная части

## Структурная часть

Реляционная модель данных является математической основой языка SQL, в которой приведено строгое формальное описание всех реляционных объектов, и ее изучение интересно именно с этой точки зрения.

Такие понятия, как таблица, строка, столбец и т.д., являются точными, но недостаточно формальными терминами для строгого математического описания реляционных объектов. При разработке реляционной модели за основу была принята теория множеств, а в качестве математической модели таблицы используется отношение (relation), которое определяется в терминах теории множеств. Собственно, и термин «реляционная» произошел от английского термина relation. Соответствие между реляционными и общепринятыми терминами:

|  |  |
| --- | --- |
| **Общепринятый термин** | **Реляционный термин** |
| Таблица | Отношение |
| Строка | Кортеж |
| Столбец | Атрибут |
| Множество допустимых значений столбца | Домен, на котором определен атрибут |
| Количество столбцов | Степень или арность |
| Количество строк | Кардинальное число |

## Атрибуты и домены. Схема отношения

**Атрибут** *определяется как именованный атомарный (неделимый) элемент данных.* *Множество допустимых значений атрибута называется* ***доменом****, на котором определен данный атрибут*. Каждый атрибут определен на своем домене, при этом допустимы одинаковые домены для различных атрибутов. Сравнивать между собой можно только значения атрибутов, определенных на одинаковых доменах.

Определение домена очень близко к определению типа данных в языках программирования. В самом общем виде домен определяется заданием некоторого базового типа данных и произвольного логического выражения, применяемого к каждому элементу данных. Если вычисление этого логического выражения дает результат "истина", то элемент данных является элементом домена.

В большинстве реляционных СУБД объект «домен» не используется, но для каждого атрибута задается базовый тип и ряд ограничений (constraints), которые проверяются для каждого значения атрибута. В стандарте SQL специфицирована команда create type... , которая позволяет конструировать произвольные типы данных, что еще точнее соответствует понятию домена, введенного Коддом для реляционных баз данных.

***Схема отношения*** *– это именованное множество упорядоченных пар (имя атрибута : имя домена).*

***Степень или "арность" схемы отношения*** *– мощность этого множества (количество элементов, входящих в множество).* Например, если в схему отношения входит всего два атрибута, то отношение называется бинарным, если три - тернарным.

***Схема БД*** *– это набор именованных схем отношений.*

## Кортежи. Отношение

***Кортеж****, соответствующий данной схеме отношения – это множество упорядоченных пар (имя атрибута : значение атрибута), которое содержит по одному вхождению каждого имени атрибута, принадлежащего схеме отношения.*

Значение атрибута является допустимым значением из домена данного атрибута. Арность кортежа, т.е. число элементов в нем, совпадает с арностью соответствующей схемы отношения. Таким образом, можно считать кортеж математической моделью одной (любой) строки таблицы.

***Отношение*** *– это множество кортежей, соответствующих одной схеме отношения.* В определении Дейта схема отношения называется заголовком отношения, а множество кортежей - телом отношения. Заголовок отношения соответствует заголовку («шапке») таблицы, тело отношения соответствует всей совокупности данных, содержащихся в таблице. Из определения отношения следуют его основные свойства:

* в отношении не может быть двух одинаковых кортежей (согласно определению множества, все его элементы уникальны),
* кортежи не упорядочены, атрибуты также не упорядочены (это свойство также является неотъемлемым свойством любого множества).

Добавим к этому, что имена всех атрибутов в пределах одного отношения должны быть уникальны.

***Реляционная база данных*** *– это набор отношений, имена которых совпадают с именами схем отношений в схеме БД.*

## Потенциальные ключи. Первичный ключ

Согласно свойствам отношений, каждый кортеж в целом уникален, однако отдельные атрибуты могут содержать и повторяющиеся значения. Тем не менее, в каждом отношении можно выделить хотя бы одну группу (подмножество) атрибутов (возможно, один атрибут), содержащих гарантированно уникальные значения.

***Потенциальным ключом отношения*** *(Candidate Key - СК) называют подмножество атрибутов отношения, которое удовлетворяет двум свойствам:*

*1. Уникальность (не существует двух одинаковых значений);*

*2. Безизбыточность (никакое подмножество потенциального ключа не является потенциальным ключом).*

Различают простые и составные потенциальные ключи (например, серия и номер паспорта - составной потенциальный ключ, а ИНН - простой).

В каждом отношении можно выделить один или несколько потенциальных ключей. Если таких ключей несколько, один из них выбирается в качестве первичного ключа (Primary Key - РК).

*Первичные ключи используются в качестве общих столбцов для связывания таблиц.* Поэтому в качестве первичного ключа обычно выбирают самый короткий, чаще всего, числовой атрибут, для которого гарантируется не только уникальность, но и неизменность значений. На практике часто невозможно выделить такой атрибут, в этом случае используют дополнительный атрибут, называемый *суррогатным ключом*, который автоматически заполняется уникальными значениями и никогда не изменяется.

*Все потенциальные ключи отношения, которые не являются первичным ключом, называются альтернативными ключами. Ни в одном из потенциальных ключей NULL-значения недопустимы.*

## Внешние ключи

Первичные ключи отношений используются в качестве общих атрибутов при связывании отношений. Для этой цели вводятся понятия родительского (главного) отношения и дочернего (подчиненного) отношения. Первичный ключ родительского отношения, экспортированный в дочернее отношение в качестве связующего атрибута, называется внешним ключом дочернего отношения. Посредством внешнего ключа кортежи дочернего отношения ссылаются на соответствующие им кортежи родительского отношения.

Определим данное понятие более формально. Назовем ***внешним ключом*** *(foreign key - FK) такое подмножество атрибутов дочернего отношения, что для любого его непустого значения обязательно найдется равное значение первичного ключа главного отношения.*

Обратное утверждение несправедливо, т.е. в родительском отношении могут найтись такие кортежи, на которые не ссылаются никакие кортежи дочернего отношения. Например, в отношении Сотрудники могут быть кортежи, для которых нет ни одного связанного кортежа в отношении Дети\_Сотрудников.

Значения внешних ключей не обязаны быть (и обычно не бывают) уникальными в своем отношении (продолжая пример с сотрудниками и детьми - у сотрудников может быть несколько детей, все они содержит одинаковые значения внешнего ключа).

Во внешнем ключе NULL-значения допустимы, в отличие от первичного ключа, однако на практике такая необходимость встречается крайне редко.

## Целостная часть реляционной модели

*Целостность данных – это механизм поддержания базы данных в непротиворечивом состоянии, соответствующем динамично изменяющейся предметной области.*

Угроза нарушения целостности данных возникает при выполнении операций манипулирования данными. Поэтому все СУБД должны контролировать операции вставки (Insert), удаления (Delete) и обновления (Update) и отказывать в выполнении операции, если в ней проводится попытка нарушить целостность базы данных. Эта проблема решается путем введения специальной системы мер, не позволяющих, например, вводить в БД данные заведомо неверного типа, дублирующиеся значения первичных ключей и т.п. Набор определенных правил, устанавливающих допустимость значений данных и их связей, называют правилами или ограничениями целостности (constraints).

Ограничения целостности задаются и хранятся в словаре данных БД как один из элементов определения таблицы, к которой они относятся. Тем самым любое приложение, обращающееся к этой таблице, необходимым образом должно придерживаться заданных правил. Изменения правил целостности может быть произведено на уровне базы данных в целом, а не для отдельного приложения. Это еще один из примеров воплощения принципа независимости данных и прикладного ПО.

Большинство БД подчиняется очень многим правилам поддержки целостности. Есть специфические (корпоративные) правила, которые характерны только для конкретной предметной области и применяется только к одной БД). Есть два общих особых правила, они применяются к любой БД и относятся к потенциальным (и первичным) ключам и ко внешним ключам. В реляционной модели данных определены два базовых универсальных требования обеспечения целостности:

* целостность сущностей,
* целостность ссылок.

## Целостность сущностей

Объект реального мира (сущность) представляется в реляционной базе данных как кортеж некоторого отношения. Требование целостности сущностей заключается в следующем:

каждый объект реального мира должен быть четко идентифицирован, т.е. любое отношение должно обладать первичным ключом.

Вполне очевидно, если данное требование не соблюдается, то в базе данных может храниться противоречивая информация об одном и том же объекте. Поддержание целостности сущностей обеспечивается средствами СУБД. Это осуществляется с помощью двух ограничений:

* при добавлении кортежей проверяется уникальность их первичных ключей и отсутствие в них NULL-значений,
* не позволяется изменение значений атрибутов, входящих в первичный ключ.

При попытке внести любое изменение в базу данных, нарушающее целостность сущностей, операция прерывается, а база данных остается в исходном, согласованном, состоянии.

## Целостность ссылок (ссылочная целостность)

Сложные объекты реального мира представляются в реляционной базе данных в виде кортежей нескольких отношений, связанных между собой с помощью внешних ключей.

Требование целостности по ссылкам состоит в следующем:

*все значения внешних ключей должны быть согласованы*.

Выполнение этого требования любая СУБД контролирует автоматически, при этом для каждой из операций манипулирования данными выполняются свои специфические действия. Безусловно, поддержка целостности отнимает много ресурсов, при этом существенно замедляется выполнение операций манипулирования данными, однако гарантированное обеспечение целостности ссылок стоит всех этих затрат.

Правила поддержки ссылочной целостности зависят от выполняемой операции манипулирования данными.

При выполнении *операции вставки* ссылочная целостность контролируется только в случае наличия внешних ключей, ссылающихся на другие отношения. В этом случае проверяется существование соответствующих значений первичных ключей, в случае их отсутствия операция вставки отменяется.

Наиболее остро стоит проблема обеспечения ссылочной целостности при выполнении *операции удаления* кортежей родительского отношения, на которые есть ссылки в дочерних отношениях (возможно, сразу в нескольких). В этом случае простое удаление кортежа приведет к наличию несогласованных значений внешних ключей во всех дочерних отношениях. Обеспечить ссылочную целостность при удалении можно несколькими способами:

1. Запретить удаление кортежей в родительском отношении при наличии хотя бы одного ссылающегося кортежа (restrict - ограничить удаления);
2. При удалении кортежа родительского отношения каскадом удалять все ссылающиеся на него кортежи дочернего отношения (cascade - каскадное удаление);
3. При удалении кортежа родительского отношения установить во всех ссылающихся кортежах NULL-значения во внешних ключах (set null). Этот способ можно применять только в случае, если NULL-значения в соответствующем внешнем ключе разрешены;
4. При удалении кортежа родительского отношения установить во всех ссылающихся кортежах значения по умолчанию во внешних ключах (set default). Значения по умолчанию задаются при создании базы данных.

Из предложенных способов наиболее безопасным является первый - запрет удаления, он применяется чаще всего. Способ каскадного удаления следует применять очень осторожно. При выполнении операций обновления во внешних ключах ссылочная целостность обеспечивается так же, как и при добавлении нового кортежа. Обновление первичных ключей при наличии ссылающихся внешних ключей категорически не рекомендуется. Вообще выполнять обновление первичных ключей нет никакой необходимости, даже если СУБД разрешает такую возможность.