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

Базовые понятия реляционной модели данных

Общая характеристика реляционной модели данных

Основы реляционной модели данных были впервые изложены в статье Е.Кодда [43] в 1970 г. Эта работа послужила стимулом для большого количества статей и книг, в которых реляционная модель получила дальнейшее развитие. Наиболее распространенная трактовка реляционной модели данных принадлежит К.Дейту [11]. Согласно Дейту, реляционная модель состоит из трех частей:

- Структурной части.
- Целостной части.
- Манипуляционной части.

Структурная часть описывает, какие объекты рассматриваются реляционной моделью. Постулируется, что единственной структурой данных, используемой в реляционной модели, являются ***нормализованные n-арные отношения***.

Целостная часть описывает ограничения специального вида, которые должны выполняться для любых отношений в любых реляционных базах данных. Это ***целостность сущностей*** и ***целостность внешних ключей***.

Манипуляционная часть описывает два эквивалентных способа манипулирования реляционными данными - ***реляционную алгебру*** и ***реляционное исчисление***.

В данной главе рассматривается структурная часть реляционной модели.

Типы данных

Любые данные, используемые в программировании, имеют свои типы данных.

Важно! Реляционная модель требует, чтобы типы используемых данных были *простыми*.

Для уточнения этого утверждения рассмотрим, какие вообще типы данных обычно рассматриваются в программировании. Как правило, типы данных делятся на три группы:

- Простые типы данных.
- Структурированные типы данных.
- Ссылочные типы данных.

Простые типы данных

Простые, или атомарные, типы данных не обладают внутренней структурой. Данные такого типа называют ***скалярами***. К простым типам данных относятся следующие типы:

- Логический.
- Строковый.
- Численный.

Различные языки программирования могут расширять и уточнять этот список, добавляя такие типы как:

- Целый.
- Вещественный.
- Дата.
- Время.
- Денежный.
- Перечислимый.
- Интервальный.
- И т.д.…

Конечно, понятие атомарности довольно относительно. Так, строковый тип данных можно рассматривать как одномерный массив символов, а целый тип данных - как набор битов. Важно лишь то, что при переходе на такой низкий уровень теряется ***семантика (смысл) данных***. Если строку, выражающую, например, фамилию сотрудника, разложить в массив символов, то при этом теряется смысл такой строки как единого целого.

Структурированные типы данных

Структурированные типы данных предназначены для задания сложных структур данных. Структурированные типы данных конструируются из составляющих элементов, называемых компонентами, которые, в свою очередь, могут обладать структурой. В качестве структурированных типов данных можно привести следующие типы данных:

- Массивы
- Записи (Структуры)

Запись (или структура) представляет собой кортеж из некоторого декартового произведения множеств. Действительно, запись представляет собой именованный упорядоченный набор элементов r_i, каждый из которых принадлежит типу T_i. Таким образом, запись $r = (r_1, r_2, \ldots, r_n)$ есть элемент множества $T = T_1 \times T_2 \times \ldots \times T_n$. Объявляя новые типы записей на основе уже имеющихся типов, пользователь может конструировать сколь угодно сложные типы данных.

Общим для структурированных типов данных является то, что они *имеют внутреннюю структуру*, используемую *на том же уровне абстракции*, что и сами типы данных.

Поясним это следующим образом. При работе с массивами или записями можно манипулировать массивом или записью и как с единым целым (создавать, удалять, копировать целые массивы или записи), так и поэлементно. Для структурированных типов

данных есть специальные функции - конструкторы типов, позволяющие создавать массивы или записи из элементов более простых типов.

Работая же с простыми типами данных, например с числовыми, мы манипулируем ими как неделимыми целыми объектами. Чтобы "увидеть", что числовой тип данных на самом деле сложен (является набором битов), нужно перейти на более низкий уровень абстракции. На уровне программного кода это будет выглядеть как ассемблерные вставки в код на языке высокого уровня или использование специальных побитных операций.

Ссылочные типы данных

Ссылочный тип данных (***указатели***) предназначен для обеспечения возможности указания на другие данные. Указатели характерны для языков процедурного типа, в которых есть понятие области памяти для хранения данных. Ссылочный тип данных предназначен для обработки сложных изменяющихся структур, например деревьев, графов, рекурсивных структур.

Типы данных, используемые в реляционной модели

Собственно, для реляционной модели данных тип используемых данных не важен. Требование, чтобы тип данных был *простым*, нужно понимать так, что *в реляционных операциях не должна учитываться внутренняя структура данных*. Конечно, должны быть описаны действия, которые можно производить с данными как с единым целым, например, данные числового типа можно складывать, для строк возможна операция конкатенации и т.д.

С этой точки зрения, если рассматривать массив, например, как единое целое и не использовать поэлементных операций, то массив можно считать простым типом данных. Более того, можно создать свой, сколь угодно сложных тип данных, описать возможные действия с этим типом данных, и, если в операциях не требуется знание внутренней структуры данных, то такой тип данных также будет простым с точки зрения реляционной теории. Например, можно создать новый тип - комплексные числа как запись вида $z = (x,y)$, где $x \in \mathbb{R}, y \in \mathbb{R}$. Можно описать функции сложения, умножения, вычитания и деления, и *все действия с компонентами* x и y выполнять только *внутри* этих операций. Тогда, если в действиях с этим типом использовать *только* описанные операции, то внутренняя структура не играет роли, и тип данных извне выглядит как атомарный.

Именно так в некоторых пост-реляционных СУБД реализована работа со сколь угодно сложными типами данных, создаваемых пользователями.

Домены

В реляционной модели данных с понятием тип данных тесно связано понятие домена, которое можно считать уточнением типа данных.

Домен — это семантическое понятие. Домен можно рассматривать как подмножество значений некоторого типа данных, имеющих определенный смысл. Домен характеризуется следующими свойствами:

- Домен имеет *уникальное имя* (в пределах базы данных).
- Домен определен на некотором *простом* типе данных или на другом домене.
- Домен может иметь некоторое *логическое условие*, позволяющее описать подмножество данных, допустимых для данного домена.
- Домен несет определенную *смысловую нагрузку*.

Например, домен D, имеющий смысл "возраст сотрудника" можно описать как следующее подмножество множества натуральных чисел:

Если тип данных можно считать множеством всех возможных значений данного типа, то домен напоминает подмножество в этом множестве. Отличие домена от понятия подмножества состоит именно в том, что *домен отражает семантику*, определенную предметной областью. Может быть несколько доменов, совпадающих как подмножества, но несущие различный смысл. Например, домены "Вес детали" и "Имеющееся количество" можно одинаково описать как множество неотрицательных целых чисел, но смысл этих доменов будет различным, и это будут *различные* домены.

Основное значение доменов состоит в том, что *домены ограничивают сравнения*. Некорректно, с логической точки зрения, сравнивать значения из различных доменов, даже если они имеют одинаковый тип. В этом проявляется смысловое ограничение доменов. Синтаксически правильный запрос "выдать список всех деталей, у которых вес детали больше имеющегося количества" не соответствует смыслу понятий "количество" и "вес".

Замечание. Понятие домена помогает правильно *моделировать* предметную область. При работе с реальной системой в принципе возможна ситуация когда требуется ответить на запрос, приведенный выше. Система даст ответ, но, вероятно, он будет бессмысленным.

Замечание. Не все домены обладают логическим условием, ограничивающим возможные значения домена. В таком случае множество возможных значений домена совпадает с множеством возможных значений типа данных.

Замечание. Не всегда очевидно, как задать логическое условие, ограничивающее возможные значения домена. Я буду благодарен тому, кто приведет мне условие на строковый тип данных, задающий домен "Фамилия сотрудника". Ясно, что строки, являющиеся фамилиями, не должны начинаться с цифр, служебных символов, с мягкого знака и т.д. Но вот является ли допустимой фамилия "Гггггыыыыы"? Почему бы нет? Очевидно, нет! А может кто-то назло так себя назовет. Трудности такого рода возникают потому, что смысл реальных явлений далеко не всегда можно формально описать. Просто мы, как все люди, интуитивно понимаем, что такое фамилия, но никто не может дать такое формальное определение, которое отличало бы фамилии от строк, фамилиями не являющимися. Выход из этой ситуации простой - положиться на разум сотрудника, вводящего фамилии в компьютер.

Отношения, атрибуты, кортежи отношения

Определения и примеры

Фундаментальным понятием реляционной модели данных является понятие ***отношения***. В определении понятия отношения будем следовать книге К. Дейта [11].

Определение 1. ***Атрибут отношения*** *есть пара вида <Имя_атрибута : Имя_домена>.*

Имена атрибутов должны быть уникальны в пределах отношения. Часто имена атрибутов отношения совпадают с именами соответствующих доменов.

Определение 2. ***Отношение*** R, определенное на множестве доменов $D_1, D_2, \ldots, D_n$ (не обязательно различных), содержит две части: заголовок и тело.

Заголовок отношения содержит фиксированное количество атрибутов отношения:

$$(< A_1 : D_1 >, < A_2 : D_2 >, \ldots, < A_n : D_n >)$$

Тело отношения содержит множество кортежей отношения. Каждый ***кортеж отношения*** представляет собой множество пар вида <Имя_атрибута : Значение_атрибута>:

$$(< A_1 : Val_1 >, < A_2 : Val_2 >, \ldots, < A_n : Val_n >)$$

таких что значение Val_i атрибута A_i принадлежит домену D_i

Отношение обычно записывается в виде:

$$R(< A_1 : D_1 >, < A_2 : D_2 >, \ldots, < A_n : D_n >),$$

или короче

$$R(A_1, A_2, \ldots, A_n),$$

или просто

$$R.$$

Число атрибутов в отношении называют ***степенью*** (или ***-арностью***) отношения.

Мощность множества кортежей отношения называют ***мощностью*** отношения.

Возвращаясь к математическому понятию отношения, введенному в предыдущей главе, можно сделать следующие выводы:

Вывод 1. Заголовок отношения описывает декартово произведение доменов, на котором задано отношение. Заголовок статичен, он не меняется во время работы с базой данных.

Если в отношении изменены, добавлены или удалены атрибуты, то в результате получим уже *другое* отношение (пусть даже с прежним именем).

Вывод 2. Тело отношения представляет собой набор кортежей, т.е. подмножество декартового произведения доменов. Таким образом, тело отношения собственно и является отношением в математическом смысле слова. Тело отношения может изменяться во время работы с базой данных - кортежи могут изменяться, добавляться и удаляться.

Пример 1. Рассмотрим отношение "Сотрудники" заданное на доменах "Номер_сотрудника", "Фамилия", "Зарплата", "Номер_отдела". Т.к. все домены различны, то имена атрибутов отношения удобно назвать так же, как и соответствующие домены. Заголовок отношения имеет вид:

Сотрудники (Номер_сотрудника, Фамилия, Зарплата, Номер_отдела)

Пусть в данный момент отношение содержит три кортежа:

(1,Иванов, 1000, 1)

(2, Петров, 2000, 2)

(3, Сидоров, 3000, 1)

такое отношение естественным образом представляется в виде таблицы:

Номер_сотрудника	Фамилия	Зарплата	Номер_отдела
1	Иванов	1000	1
2	Петров	2000	2
3	Сидоров	3000	1

Таблица 1 Отношение "Сотрудники"

Определение 3. ***Реляционной базой данных*** называется набор отношений.

Определение 4. ***Схемой реляционной базы*** данных называется набор заголовков отношений, входящих в базу данных.

Хотя любое отношение можно изобразить в виде таблицы, нужно четко понимать, что *отношения не являются таблицами*. Это близкие, но не совпадающие понятия. Различия между отношениями и таблицами будут рассмотрены ниже.

Термины, которыми оперирует реляционная модель данных, имеют соответствующие "табличные" синонимы:

Реляционный термин	Соответствующий "табличный" термин
База данных	Набор таблиц
Схема базы данных	Набор заголовков таблиц
Отношение	Таблица
Заголовок отношения	Заголовок таблицы
Тело отношения	Тело таблицы
Атрибут отношения	Наименование столбца таблицы
Кортеж отношения	Строка таблицы
Степень (-арность) отношения	Количество столбцов таблицы
Мощность отношения	Количество строк таблицы
Домены и типы данных	Типы данные в ячейках таблицы

Свойства отношений

Свойства отношений непосредственно следуют из приведенного выше определения отношения. В этих свойствах в основном и состоят различия между отношениями и таблицами.

1. *В отношении нет одинаковых кортежей*. Действительно, тело отношения есть *множество* кортежей и, как всякое множество, не может содержать неразличимые элементы (см. понятие множества в гл.1.). Таблицы в отличие от отношений могут содержать одинаковые строки.
2. *Кортежи не упорядочены (сверху вниз)*. Действительно несмотря на то, что мы изобразили отношение "Сотрудники" в виде таблицы, нельзя сказать, что сотрудник Иванов "предшествует" сотруднику Петрову. Причина та же - тело отношения есть множество, а множество не упорядочено. Это вторая причина, по которой нельзя отождествить отношения и таблицы - строки в таблицах упорядочены. Одно и то же отношение может быть *изображено* разными таблицами, в которых *строки идут в различном порядке*.
3. *Атрибуты не упорядочены (слева направо)*. Т. к. каждый атрибут имеет уникальное имя в пределах отношения, то порядок атрибутов не имеет значения. Это свойство несколько отличает отношение от математического определения отношения (см. гл.1 - компоненты кортежей там *упорядочены*). Это также третья причина, по которой нельзя отождествить отношения и таблицы - столбцы в таблице упорядочены. Одно и то же отношение может быть *изображено* разными таблицами, в которых *столбцы идут в различном порядке*.
4. *Все значения атрибутов атомарны*. Это следует из того, что лежащие в их основе атрибуты имеют атомарные значения. Это четвертое отличие отношений от таблиц - в ячейки таблиц можно поместить что угодно - массивы, структуры, и даже другие таблицы.

Замечание. Из свойств отношения следует, что *не каждая* таблица может задавать отношение. Для того, чтобы некоторая таблица задавала отношение, необходимо, чтобы таблица имела простую структуру (содержала бы только строки и столбцы, причем, в

каждой строке было бы одинаковое количество полей), в таблице не должно быть одинаковых строк, любой столбец таблицы должен содержать данные только одного типа, все используемые типы данных должны быть простыми.

Замечание. Каждое отношение можно считать ***классом эквивалентности таблиц***, для которых выполняются следующие условия:

- Таблицы имеют одинаковое количество столбцов.
- Таблицы содержат столбцы с одинаковыми наименованиями.
- Столбцы с одинаковыми наименованиями содержат данные из одних и тех же доменов.
- Таблицы имеют одинаковые строки с учетом того, что порядок столбцов может различаться.

Все такие таблицы есть различные *изображения* одного и того же отношения.

Первая нормальная форма

Труднее всего дать определение вещей, которые всем понятны. Если давать не строгое, описательное определение, то всегда остается возможность неправильной его трактовки. Если дать строгое формальное определение, то оно, как правило, или тривиально, или слишком громоздко. Именно такая ситуация с определением отношения в ***Первой Нормальной Форме*** (***1НФ***). Совсем не говорить об этом нельзя, т.к. на основе 1НФ строятся более высокие нормальные формы, которые рассматриваются далее в гл. 6 и 7. Дать определение 1НФ сложно ввиду его тривиальности. Поэтому, дадим просто несколько объяснений.

Объяснение 1. Говорят, что отношение R находится в 1НФ, если оно удовлетворяет определению 2.

Это, собственно, тавтология, ведь из определения 2 следует, что других отношений не бывает. Действительно, определение 2 описывает, что является отношением, а что - нет, следовательно, отношений в непервой нормальной форме просто нет.

Объяснение 2. Говорят, что отношение R находится в 1НФ, если его атрибуты содержат только скалярные (атомарные) значения.

Опять же, определение 2 опирается на понятие домена, а домены определены на простых типах данных.

Непервую нормальную форму можно получить, если допустить, что атрибуты отношения могут быть определены на сложных типах данных - массивах, структурах, или даже на других отношениях. Легко себе представить таблицу, у которой в некоторых ячейках содержатся массивы, в других ячейках - определенные пользователями сложные структуры, а в третьих ячейках - целые реляционные таблицы, которые в свою очередь могут содержать такие же сложные объекты. Именно такие возможности предоставляются некоторыми современными пост-реляционными и объектными СУБД.

Требование, что отношения должны содержать только данные простых типов, объясняет, почему отношения иногда называют ***плоскими таблицами*** (***plain table***). Действительно, таблицы, задающие отношения двумерны. Одно измерение задается списком столбцов, второе измерение задается списком строк. Пара координат (Номер строки, Номер столбца)

однозначно идентифицирует ячейку таблицы и содержащееся в ней значение. Если же допустить, что в ячейке таблицы могут содержаться данные сложных типов (массивы, структуры, другие таблицы), то такая таблица будет уже не плоской. Например, если в ячейке таблицы содержится массив, то для обращения к элементу массива нужно знать *три* параметра (Номер строки, Номер столбца, номер элемента в массиве).

Таким образом появляется третье объяснение Первой Нормальной Формы:

Объяснение 3. Отношение R находится в 1НФ, если оно является плоской таблицей.

Мы сознательно ограничиваемся рассмотрением только классической реляционной теории, в которой все отношения имеют только атомарные атрибуты и заведомо находятся в 1НФ.

Выводы

Реляционная модель данных состоит из трех частей:

- Структурной части.
- Целостной части.
- Манипуляционной части.

В классической реляционной модели используются только ***простые (атомарные) типы данных***. Простые типы данных не обладают внутренней структурой.

Домены - это типы данных, имеющие некоторый смысл (семантику). Домены ограничивают сравнения - некорректно, хотя и возможно, сравнивать значения из различных доменов.

Отношение состоит из двух частей - ***заголовка отношения*** и ***тела отношения***. Заголовок отношения - это аналог заголовка таблицы. Заголовок отношения состоит из атрибутов. Количество атрибутов называется ***степенью отношения***. Тело отношения - это аналог тела таблицы. Тело отношения состоит из ***кортежей***. Кортеж отношения является аналогом строки таблицы. Количество кортежей отношения называется ***мощностью отношения***.

Отношение обладает следующими свойствами:

- В отношении нет одинаковых кортежей.
- Кортежи не упорядочены (сверху вниз).
- Атрибуты не упорядочены (слева направо).
- Все значения атрибутов атомарны.

Реляционной базой данных называется набор отношений.

Схемой реляционной базы данных называется набор заголовков отношений, входящих в базу данных.

Отношение находится в ***Первой Нормальной Форме*** (***1НФ***), если оно содержит только скалярные (атомарные) значения.