Реляционная модель данных. Понятия и определения. Основные свойства отношений. Целостность сущности и ссылок

С.Д. Кузнецов. Базы данных. Тема 3

План (1)

- Введение
- Базовые понятия реляционных баз данных
  - Тип данных
  - Домен
  - Заголовок отношения, кортеж, тело отношения, значение отношения, переменная отношения
  - Первичный ключ и интуитивная интерпретация реляционных понятий

План (2)

- Фундаментальные свойства отношений
  - Отсутствие кортежей-дубликатов, первичный и возможные ключи отношений
  - Отсутствие упорядоченности кортежей
  - Отсутствие упорядоченности атрибутов
  - Атомарность значений атрибутов, первая нормальная форма отношения
- Реляционная модель данных
  - Общая характеристика
  - Целостность сущности и ссылок

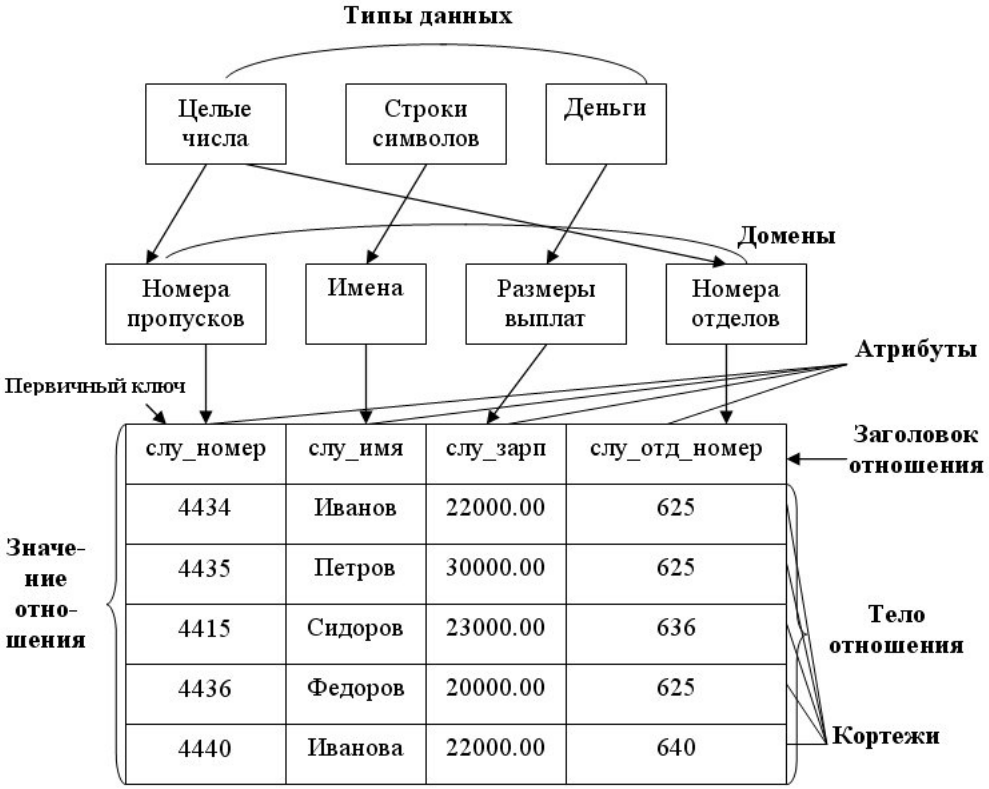
Введение

- Признанными достоинствами реляционного подхода принято считать следующие свойства:
  - реляционный подход основывается на небольшом числе интуитивно понятных абстракций, на основе которых возможно простое моделирование наиболее распространенных предметных областей; эти абстракции могут быть точно и формально определены;
  - теоретическим базисом реляционного подхода к организации баз данных служит простой и мощный математический аппарат теории множеств и математической логики;
  - реляционный подход обеспечивает возможность ненавигационного манипулирования данными без необходимости знания конкретной физической организации баз данных во внешней памяти

Базовые понятия реляционных баз данных (1)

- Основные понятия реляционных баз данных: *тип данных, домен, атрибут, кортеж, отношение*, *первичный ключ*
- Покажем смысл этих понятий на примере отношения

  СЛУЖАЩИЕ



слу_номер	слу_имя	слу_зарп	слу_отд_номер
4434	Иванов	22000.00	625
4435	Петров	30000.00	625
4415	Сидоров	23000.00	636
4436	Федоров	20000.00	625
4440	Иванова	22000.00	640

Базовые понятия реляционных баз данных (2) Тип данных (1)

- Значения данных, хранимые в реляционной базе данных, являются типизированными, т.е. известен тип каждого хранимого значения
- Понятие *типа данных* в реляционной модели данных полностью соответствует понятию типа данных в языках программирования
- Определение типа данных состоит из трех основных компонентов:
  - множества значений данного типа;
  - набор операций, применимых к значениям типа;
  - способ внешнего представления значений типа (литералов)

Базовые понятия реляционных баз данных (3) Тип данных (2)

- Обычно в современных реляционных базах данных допускается хранение
  - символьных, числовых данных (точных и приблизительных),
  - специализированных числовых данных (таких как «деньги»),
  - а также специальных «темпоральных» данных (дата, время, временной интервал)
- Кроме того, в реляционных системах поддерживается возможность определения пользователями собственных типов данных
- В примере мы имеем дело с данными трех типов: строки символов, целые числа и «деньги».

Базовые понятия реляционных баз данных (4) Домен (1)

- Понятие *домена* более специфично для баз данных, хотя и имеются аналогии с подтипами в некоторых языках программирования
- В общем виде домен определяется путем задания некоторого базового типа данных, к которому относятся элементы домена, и произвольного логического выражения, применяемого к элементу этого типа данных (*ограничения домена*)
- Элемент данных является элементом домена в том и только в том случае, если вычисление этого логического выражения дает результат *истина*
- С каждым доменом связывается имя, уникальное среди имен всех доменов соответствующей базы данных.

Базовые понятия реляционных баз данных (5) Домен (2)

- Наиболее правильной интуитивной трактовкой понятия домена является его восприятие как допустимого потенциального, ограниченного подмножества значений данного типа
- Например, домен ИМЕНА в примере определен на базовом типе символьных строк, но в число его значений могут входить только те строки, которые могут изображать имя
- Если некоторый атрибут отношения определяется на некотором домене (как, например, атрибут СЛУ_ИМЯ определяется на домене ИМЕНА), то в дальнейшем ограничение домена играет роль ограничения целостности, накладываемого на значения этого атрибута

Базовые понятия реляционных баз данных (6) Домен (3)

- Следует отметить также семантическую нагрузку понятия домена: данные считаются сравнимыми только в том случае, когда они относятся к одному домену.
- В нашем примере значения доменов НОМЕРА ПРОПУСКОВ и НОМЕРА ОТДЕЛОВ относятся к типу целых чисел, но не являются сравнимыми (допускать их сравнение было бы бессмысленно)

Базовые понятия реляционных баз данных (7) Отношение (1)

- Понятие отношения является наиболее фундаментальным в реляционном подходе к организации баз данные, поскольку *n*-арное отношение является единственной родовой структурой данных, хранящихся в реляционной базе данных
- Это отражено и в общем названии подхода – термин *реляционный (relational)* происходит от *relation (отношение)*
- Однако сам термин *отношение* является исключительно неточным, поскольку, говоря про любые сохраняемые данные, мы должны иметь в виду *тип* этих данных, *значения* этого типа и *переменные*, в которых сохраняются значения
- Соответственно, для уточнения термина *отношение* выделяются понятия *заголовка отношения*, *значения отношения* и *переменной отношения*
- Кроме того, потребуется вспомогательное понятие *кортежа*

Базовые понятия реляционных баз данных (8) Отношение (2)

- *Заголовком* (или *схемой*) *отношения R* (*HR*) называется конечное множество упорядоченных пар вида <*A*, *T*>, где *A* называется именем *атрибута*, а *T* – имя некоторого базового типа или ранее определенного домена
- По определению требуется, чтобы все имена атрибутов в заголовке отношения были различны.
- В примере заголовком отношения СЛУЖАЩИЕ является множество пар {<СЛУ_НОМЕР, НОМЕРА_ПРОПУСКОВ>, <СЛУ_ИМЯ, ИМЕНА>, <СЛУ_ЗАРП, РАЗМЕРЫ_ВЫПЛАТ>, <СЛУ_ОТД_НОМЕР, НОМЕРА_ОТДЕЛОВ>}.
- Если все атрибуты заголовка отношения определены на разных доменах, то, чтобы не плодить лишних имен, разумно использовать для именования атрибутов имена соответствующих доменов

Базовые понятия реляционных баз данных (9) Отношение (3)

- *Кортежем tR*, соответствующим заголовку *HR*, называется множество упорядоченных триплетов вида <*A*, *T*, *v*>, по одному такому триплету для каждого атрибута в *HR*. Третий элемент – *v* – триплета < *A*, *T*, *v* > должен являться допустимым значением типа данных или домена *T*
- Заголовку отношения СЛУЖАЩИЕ соответствуют, например, следующие кортежи:
  - {<СЛУ_НОМЕР, НОМЕРА_ПРОПУСКОВ, 4434>, <СЛУ_ИМЯ, ИМЕНА, Иванов>, <СЛУ_ЗАРП, РАЗМЕРЫ_ВЫПЛАТ, 22.000>, <СЛУ_ОТД_НОМЕР, НОМЕРА_ОТДЕЛОВ, 625>},
  - {<СЛУ_НОМЕР, НОМЕРА_ПРОПУСКОВ, 4455>, <СЛУ_ИМЯ, ИМЕНА, Кузнецов>, <СЛУ_ЗАРП, РАЗМЕРЫ_ВЫПЛАТ, 35.000>, <СЛУ_ОТД_НОМЕР, НОМЕРА_ОТДЕЛОВ, 320>}.

Базовые понятия реляционных баз данных (10) Отношение (4)

- *Телом BR* отношения *R* называется произвольное множество кортежей *tR*
- Одно из возможных тел отношения СЛУЖАЩИЕ показано на рисунке
- Заметим, что в общем случае могут существовать такие кортежи *tR*, которые соответствуют *HR*, но не входят в *BR*
- *Значением VR* отношения *R* называется пара множеств *HR* и *BR*
- Одно из допустимых значений отношения СЛУЖАЩИЕ показано на рисунке

Базовые понятия реляционных баз данных (11) Отношение (5)

- *Телом BR* отношения *R* называется произвольное множество кортежей *tR*
- Одно из возможных тел отношения СЛУЖАЩИЕ показано на рисунке
- Заметим, что в общем случае могут существовать такие кортежи *tR*, которые соответствуют *HR*, но не входят в *BR*
- *Значением VR* отношения *R* называется пара множеств *HR* и *BR*
- Одно из допустимых значений отношения СЛУЖАЩИЕ показано на рисунке
- В изменчивой реляционной базе данных хранятся отношения, значения которых изменяются во времени
- *Переменной VARR* называется именованный контейнер, который может содержать любое допустимое значение *VR*
- Естественно, что при определении любой *VARR* требуется указывать соответствующий заголовок отношения *HR*

Базовые понятия реляционных баз данных (12) Отношение (6)

- Любая принятая на практике операция обновления базы данных – INSERT (вставка кортежа в переменную отношения), DELETE (удаление кортежа из значения-отношения переменой отношения) и UPDATE (модификация кортежа значения-отношения переменной отношения) – с модельной точки зрения является операцией присваивания переменной отношения некоторого нового значения-отношения
- Это совсем не означает, что перечисленные операции должны выполняться именно таким образом в СУБД: главное, чтобы результата операций соответствовал этой модельной семантике
- Заметим, что в дальнейшем в тех случаях, когда точный смысл термина понятен по контексту, мы будем использовать не уточненный термин ***отношение***, как в смысле ***значение отношения***, так и в смысле ***переменная отношения***

Базовые понятия реляционных баз данных (13) Отношение (7)

- По определению *степенью*, или “*арностью*” заголовка отношения, кортежа, соответствующего этому заголовку, тела отношения, значения отношения и переменной отношения является мощность заголовка отношения
- Например, степень отношения СЛУЖАЩИЕ равна четырем, то есть оно является 4-арным (*кватернарным)*.
- Разумно считать *схемой реляционной базы данных* набор пар <имя_*VARR*, *HR* >, включающий имена и заголовки всех переменных отношения, которые определены в базе данных
- *Реляционная база данных* – это набор *VARR* (конечно, каждая переменная отношения в любой момент времени содержит некоторое значение-отношение, в частности, пустое)

Базовые понятия реляционных баз данных (14) Отношение (8)

- Заметим, что в классических реляционных базах данных после определения схемы базы данных могли изменяться только значения переменных отношений
- Однако теперь в большинстве реализаций допускается и изменение схемы базы данных
  - определение новых и
  - изменение заголовков существующих переменных отношений
- Это принято называть *эволюцией схемы базы данных*

Базовые понятия реляционных баз данных (15) Первичный ключ (1)

- По определению, первичным ключом переменной отношения является подмножество *S* множества атрибутов ее заголовка такое, что в любой момент времени
  - значение первичного ключа
    - (составное, если в состав первичного ключа входит более одного атрибута)

    в любом кортеже тела отношения отличается от значения первичного ключа в любом другом кортеже тела этого отношения, а
  - никакое собственное подмножество *S* этим свойством не обладает
- Позже мы покажем, что существование первичного ключа у любого значения отношения является следствием одного из фундаментальных свойств отношений, а именно, того свойства, что тело отношения является множеством кортежей

Базовые понятия реляционных баз данных (16) Первичный ключ (2)

- Обычным житейским представлением отношения является *таблица*, *заголовком* которой является схема отношения, а *строками* – кортежи отношения-экземпляра; в этом случае имена атрибутов именуют *столбцы* этой таблицы
- Поэтому иногда говорят про «столбцы таблицы», имея в виду «атрибуты отношения».
- Это достаточно грубая терминология, поскольку у обычных таблиц и строки, и столбцы упорядочены, тогда как атрибуты и кортежи отношений являются элементами неупорядоченных множеств
- Этой терминологии придерживаются в большинстве коммерческих реляционных СУБД
- Иногда также используются термины *файл* как аналог таблицы, *запись* как аналог строки и *поле* как аналог столбца

Фундаментальные свойства отношений (1)

Отсутствие кортежей-дубликатов (1)

- То свойство, что тело любого отношения никогда не содержит кортежей-дубликатов, следует из определения тела отношения как множества кортежей
  - В классической теории множеств по определению любое множество состоит из различных элементов
- Именно из этого свойства вытекает наличие у каждого значения отношения *первичного ключа* – минимального подмножества заголовка данного отношения, составное значение которого уникально определяет кортеж отношения
- Действительно, поскольку в любое время все кортежи тела любого отношения различны, у любого значения отношения свойством уникальности обладает, по крайней мере, полный набор его атрибутов

Фундаментальные свойства отношений (2)

Отсутствие кортежей-дубликатов (2)

- Однако в формальном определении первичного ключа требуется обеспечение его «минимальности», т.е. в набор атрибутов первичного ключа не должны входить такие атрибуты, которые можно отбросить без ущерба для основного свойства – однозначного определения кортежа
- Немного позже мы покажем, почему свойство минимальности первичного ключа является критически важным
- Понятно, что если у любого отношения существует набор атрибутов, обладающий свойством уникальности, то существует и минимальный набор атрибутов, обладающий свойством уникальности

Фундаментальные свойства отношений (3)

Отсутствие кортежей-дубликатов (3)

- Конечно, могут существовать значения отношения с несколькими несовпадающими минимальными наборами атрибутов, обладающими свойствами уникальности
- Например, если вернуться к предположениям темы 1 об уникальности значений атрибутов СЛУ_НОМЕР и СЛУ_ИМЯ отношения СЛУЖАЩИЕ, то для каждого значения этого отношения мы имеем два множества атрибутов, претендующих на звание первичного ключа – {СЛУ_НОМЕР} и {СЛУ_ИМЯ}
- В этом случае проектировщик базы данных должен решить, какое из альтернативных множеств атрибутов назвать первичным ключом, а остальные минимальные наборы атрибутов, обладающие свойством уникальности, называются *возможными ключами*

Фундаментальные свойства отношений (4)

Отсутствие кортежей-дубликатов (4)

- Понятие первичного ключа является исключительно важным в связи с понятием целостности баз данных
- Заметим, что хотя формально существование первичного ключа значения отношения является следствием того, что тело отношения – это множество, на практике первичные (и возможные) ключи *переменных отношений* появляются в результате явных указаний проектировщика отношения
- Определяя переменную отношения, проектировщик моделирует часть предметной области, данные из которой будет в будущем содержать база данных
- И конечно, проектировщик должен знать природу этих данных

Фундаментальные свойства отношений (5)

Отсутствие кортежей-дубликатов (5)

- Например, ему должно быть известно, что никакие два служащих ни в какой времени не могут иметь удостоверение с одним и тем же номером
- Поэтому он может (и даже должен, как будет показано ниже) явно объявить {СЛУ_НОМЕР} возможным ключом
- Если на предприятии установлено, что у всех сотрудников должны быть разные полные имена, то проектировщик может (и опять же должен) объявить возможным ключом и {СЛУ_ИМЯ}
- Потом проектировщик должен оценить, какой из возможных ключей является более надежным (свойство его уникальности никогда не будет отменено), и выбрать наиболее надежный возможный ключ в качестве первичного

Фундаментальные свойства отношений (6)

Отсутствие кортежей-дубликатов (6)

- Теперь поясним, почему проектировщику следует явно объявлять первичный и возможные ключи переменных отношений
- Дело в том, в результате этого объявления СУБД получает информацию, которая в дальнейшем будет использоваться как ограничение целостности
- СУБД никогда не допустит появления в переменной отношения значения-отношения, содержащего два кортежа с одинаковым значением атрибута СЛУ_НОМЕР
  - определение первичного ключа для данной переменной отношения нельзя отменить
- Появление двух кортежей с одинаковым значением атрибута СЛУ_ИМЯ будет также невозможно до тех пор, пока остается в силе определение {СЛУ_ИМЯ} как возможного ключа
- Тем самым, объявления первичного и возможных ключей дают СУБД возможность поддерживать целостность базы данных при попытках занесения в нее некорректных данных

Фундаментальные свойства отношений (7)

Отсутствие кортежей-дубликатов (7)

- Наконец, вернемся к свойству минимальности первичного (и возможных ключей)
- Это свойство является критически важным, и важность проявляется именно при трактовке первичного и возможных ключей как ограничений целостности
- К примеру, в нашем примере с отношением СЛУЖАЩИЕ свойством уникальности будет обладать не только множество атрибутов {СЛУ_НОМЕР}, но и, например, множество {СЛУ_НОМЕР, СЛУ_ОТД_НОМЕР}
- Но если бы мы выставили в качестве ограничения целостности требование уникальности {СЛУ_НОМЕР, СЛУ_ОТД_НОМЕР}, то СУБД гарантировала бы отсутствие кортежей с одинаковым значением атрибута СЛУ_НОМЕР не во всем значении отношения СЛУЖАЩИЕ, а только в группах кортежей с одним и тем же значением атрибута СЛУ_ОТД_НОМЕР
- Понятно, что это не соответствует смыслу моделируемой предметной области

Фундаментальные свойства отношений (8)

Отсутствие упорядоченности кортежей (1)

- Формально свойство отсутствия упорядоченности кортежей в значении отношения также является следствием определения тела отношения как множества кортежей
- Однако на это свойство можно взглянуть и с другой стороны
  - Да, то свойство, что тело отношения является множеством кортежей, облегчает построение полного механизма реляционной модели данных, включая базовые средства манипулирования данными – реляционные алгебру и исчисление
  - Но основная причина не в этом
- Можно было бы разработать другую теорию, в которой допускались бы упорядоченные «отношения»
- Однако хранить упорядоченные списки кортежей в условиях интенсивно обновляемой базы данных гораздо сложнее технически, а поддержка упорядоченности вызывает серьезные накладные расходы

Фундаментальные свойства отношений (9)

Отсутствие упорядоченности кортежей (2)

- Отсутствие требования к поддержанию порядка на множестве кортежей отношения придает СУБД дополнительную гибкость при хранении баз данных во внешней памяти и при выполнении запросов к базе данных
- Это не противоречит тому, что при формулировании запроса к БД, например, на языке SQL можно потребовать сортировку результирующей таблицы в соответствии со значениями некоторых столбцов
- Такой результат, вообще говоря, является не отношением, а некоторым упорядоченным списком кортежей, и он может быть только окончательным результатом, к которому уже нельзя адресовать запросы

Фундаментальные свойства отношений (10)

Отсутствие упорядоченности атрибутов (1)

- Атрибуты отношений не упорядочены, поскольку по определению заголовок отношения есть множество пар <имя_атрибута, имя_типа_или_домена>
- Для ссылки на значение атрибута в кортеже отношения всегда используется имя атрибута
- Легко заметить явную аналогию между заголовками отношений и структурными типами в языках программирования
- Даже в языке программирования С с его практически неограниченными возможностями работы с указателями настойчиво рекомендуется обращаться к полям структур только по их именам.

Фундаментальные свойства отношений (11)

Отсутствие упорядоченности атрибутов (2)

- Если, например, на языке С определена структурная переменная
  `STRUCT {integer a; char b; integer c} d;`
  то в стандарте языка решительно *не* рекомендуется использовать для доступу к символьному полю *b* конструкцию *(&d + sizeof(integer))
- Это объясняется тем, что при реальном расположении в памяти полей этой структурной переменной в том порядке, как они определены, во многих компьютерах потребуется выровнять поле *c* по байту с четным адресом
- Поэтому один байт просто пропадет
- При расположении структурной переменной в памяти экономный компилятор (вернее, оптимизатор) переставит местами поля *b* и *c*, и указанная выше конструкция не обеспечит доступа к полю *b*
- Для корректного обращения к полю *b* переменной *d* нужно использовать конструкции d.b или &d $\rightarrow$ b, т.е. явно указывать имя поля

Фундаментальные свойства отношений (12)

Отсутствие упорядоченности атрибутов (3)

- Аналогичными практическими соображениями оправдывается и отсутствие упорядоченности атрибутов в заголовке отношения
- В этом случае СУБД сама принимает решение о том, в каком физическом порядке следует хранить значения атрибутов кортежей (хотя обычно один и тот же физический порядок поддерживается для всех кортежей каждого отношения)
- Кроме того, это свойство облегчает выполнение операции модификации схем существующих отношений не только путем добавления новых атрибутов, но и путем удаления существующих атрибутов

Фундаментальные свойства отношений (13)

Первая нормальная форма (1)

- Эдгар Кодд справедливо утверждал, что для моделирования большинства предметных областей можно обойтись отношениями, атрибуты которых определены на простых доменах, элементы которых являются атомарными, не декомпозируемыми
- Он, в частности, говорил следующее: «Отношение, все домены которого являются простыми, может быть представлено двухмерным массивом … с однородными столбцами. Для отношения с одним или более непростыми доменами требуются несколько более сложные структуры данных.»
- Более того, он предлагал простую процедуру нормализации, приводящую отношение, значениями одного из атрибутов которых являются отношения, к нескольким отношениям над простыми доменами

Фундаментальные свойства отношений (14)

Первая нормальная форма (2)

- Однако и в истинной реляционной модели данных, и в модели данных SQL теперь можно определять отношения (таблицы), значениями атрибутов (столбцов) которых являются отношения (мультимножества строк)
- Скорее всего, это произошло по двум причинам:
  - Во-первых, в течение многих лет реляционную модель данных (и модель SQL) многие люди критиковали за сложность представления иерархически организованных данных. Понятно, что при наличии механизма вложенных отношений иерархические данные представляются вполне естественно.
  - Во-вторых, оказалось, что при тщательном определении системы типов, наличие атрибутов (столбцов) со значениями-отношениями никак не влияет на методы, разработанные в исходной теории реляционных БД

Фундаментальные свойства отношений (15)

Первая нормальная форма (3)

- Тем не менее, доводы Кодда, которые привели к появлению первой нормальной формы отношений, не утратили смысла

НОМЕР_ОТДЕЛА	ОТДЕЛ		
	СЛУ_НОМЕР	СЛУ_ИМЯ	СЛУ_ЗАРП
625	4434	Иванов	22000.00
	4435	Петров	30000.00
	4436	Федоров	20000.00
636	4415	Сидоров	23000.00
640	4440	Иванова	22000.00

Ненормализованное отношение ОТДЕЛЫ-СЛУЖАЩИЕ

СЛУ_НОМЕР	СЛУ_ИМЯ	СЛУ_ЗАРП	СЛУ_ОТД_НОМЕР
4434	Иванов	22000.00	625
4435	Петров	30000.00	625
4415	Сидоров	23000.00	636
4436	Федоров	20000.00	625
4440	Иванова	22000.00	640

Отношение СЛУЖАЩИЕ: нормализованный вариант отношения ОТДЕЛЫ-СЛУЖАЩИЕ

Фундаментальные свойства отношений (16)

Первая нормальная форма (4)

- Нормализованные по Кодду отношения составляют основу классического реляционного подхода к организации баз данных
- Они обладают некоторыми ограничениями (не любую информацию удобно представлять в виде плоских таблиц), но существенно упрощают манипулирование данными
- Рассмотрим, например, два идентичных оператора занесения кортежа:
  - Зачислить служащего Кузнецова (пропуск номер 3000, зарплата 115,000) в отдел номер 320 и
  - Зачислить служащего Кузнецова (пропуск номер 3000, зарплата 115,000) в отдел номер 310

Фундаментальные свойства отношений (17)

Первая нормальная форма (5)

- Если информация о сотрудниках представлена в виде отношения СЛУЖАЩИЕ, оба оператора будут выполняться одинаково:
  - вставить кортеж в отношение СЛУЖАЩИЕ
- Если же работать с ненормализованным отношением ОТДЕЛЫ-СЛУЖАЩИЕ, то:
  - первый оператор выразится в занесение кортежа,
  - а второй - в добавление кортежа с первичным ключом 310 в значение-отношение атрибута ОТДЕЛ
- При работе с ненормализованными отношениями аналогичные затруднения возникают при выполнении операций удаления и модификации кортежей

Реляционная модель данных (1)

Общая характеристика (1)

- В структурной части реляционной модели данных фиксируется, что единственной родовой структурой данных, используемой в реляционных БД, является нормализованное *n*-арное отношение
- Определяются понятия доменов, атрибутов, кортежей, заголовка, тела и переменной отношения
- По сути дела, до сих пор мы рассматривали именно понятия и свойства структурной составляющей реляционной модели

Реляционная модель данных (2)

Общая характеристика (2)

- В манипуляционной части реляционной модели определяются два фундаментальных механизма манипулирования реляционными БД – реляционная алгебра и реляционное исчисление
- Первый механизм базируется в основном на классической теории множеств (с некоторыми уточнениями и добавлениями), а второй – на классическом логическом аппарате исчисления предикатов первого порядка
- Мы кратко рассмотрели алгебру Кодда ранее и обсудим все эти механизмы более подробно в следующей теме, а пока лишь заметим, что основной функцией манипуляционной части реляционной модели является обеспечение меры реляционности любого конкретного языка реляционных БД:
  - язык называется реляционным, если он обладает не меньшей выразительностью и мощностью, чем реляционная алгебра или реляционное исчисление

Реляционная модель данных (3)

Целостность сущности и ссылок (1)

- В целостной части реляционной модели данных фиксируются два базовых требования целостности, которые должны поддерживаться в любой реляционной СУБД
- Первое требование называется *требованием целостности сущности* (*entity integrity*)
- Объекту или сущности реального мира в реляционных БД соответствуют кортежи отношений
- Конкретно, требование состоит в том, что любой кортеж любого значения-отношения любой переменной отношения должен быть отличим от любого другого кортежа этого значения отношения по составным значениям заранее определенного множества атрибутов переменной отношения, т.е., другими словами,
  - любая переменная отношения должно обладать первичным ключом
- Как мы видели, это требование автоматически удовлетворяется, если в системе не нарушаются базовые свойства отношений

Реляционная модель данных (4)

Целостность сущности и ссылок (2)

- На самом деле, требование целостности сущности полностью звучит следующим образом:
  - *у любой переменной отношения должен существовать первичный ключ, и никакое значение первичного ключа в кортежах значения-отношения переменной отношения не должно содержать неопределенных значений*
- Чтобы эта формулировка была полностью понятна, мы должны хотя бы кратко обсудить понятие *неопределенного значения* (NULL)
- Конечно, теоретически любой кортеж, заносимый в сохраняемое отношение, должен содержать все характеристики моделируемой им сущности реального мира, которые мы хотим сохранить в базе данных

Реляционная модель данных (5)

Целостность сущности и ссылок (3)

- Однако на практике не все эти характеристики могут быть известны к тому моменту, когда требуется зафиксировать сущность в базе данных
- Простым примером может быть процедура принятия на работу человека, размер заработной платы которого еще не определен
- В этом случае работник отдела кадров, который заносит в отношение СЛУЖАЩИЕ кортеж, описывающий нового служащего, просто не может обеспечить значение атрибута СЛУ_ЗАРП
  - любое значение домена РАЗМЕРЫ_ВЫПЛАТ будет неверно характеризовать зарплату нового сотрудника
- Эдгар Кодд предложил использовать в таких случаях неопределенные значения

Реляционная модель данных (6)

Целостность сущности и ссылок (4)

- Неопределенное значение не принадлежит никакому типу данных и может присутствовать среди значений любого атрибута, определенного на любом типе данных (если это явно не запрещено при определении атрибута)
- Если *a* – это значение некоторого типа данных или NULL,
  op – любая двуместная “арифметическая” операция этого типа данных (например, “+”), а
  lop – операция сравнения значений этого типа (например, “=”), то по определению:
  - *a op* NULL = NULL
    NULL *op a* = NULL
    a lop NULL = *unknown*
    NULL *lop a* = *unknown*

Реляционная модель данных (7)

Целостность сущности и ссылок (5)

- Здесь *unknown* – это третье значение логического, или булевского типа, обладающее следующими свойствами:
  - NOT *unknown* = *unknown*
    true AND *unknown* = *unknown*
    true OR *unknown* = *true*
    false AND *unknown* = *false*
    false OR *unknown* = *unknown*
- Напомним, что операции AND и OR являются коммутативными
- Требование целостности сущности означает, что первичный ключ должен полностью идентифицировать каждую сущность, а поэтому в составе любого значения первичного ключа не допускается наличие неопределенных значений

Реляционная модель данных (8)

Целостность сущности и ссылок (6)

- Второе требование называется *требованием целостности по ссылкам* (*referential integrity*) и является несколько более сложным
- Очевидно, что при соблюдении нормализованности по Кодду отношений сложные сущности реального мира представляются в реляционной БД в виде нескольких кортежей нескольких отношений
- Например, представим, что требуется представить в реляционной базе данных сущность ОТДЕЛ с атрибутами ОТД_НОМЕР (номер отдела), ОТД_РАЗМ (количество служащих) и ОТД_СЛУ (набор сотрудников отдела). Для каждого служащего нужно хранить СЛУ_НОМЕР (номер сотрудника), СЛУ_ИМЯ (имя сотрудника) и СЛУ_ЗАРП (заработная плата сотрудника)

Реляционная модель данных (9)

Целостность сущности и ссылок (7)

- Как мы увидим позже, при правильном проектировании соответствующей БД в ней появятся два отношения: ОТДЕЛЫ { ОТД_НОМЕР, ОТД_РАЗМ } (первичный ключ – {ОТД_НОМЕР}) и СОТРУДНИКИ { СЛУ_НОМЕР, СЛУ_ИМЯ, СЛУ_ЗАРП, СЛУ_ОТД_НОМ) (первичный ключ – {СЛУ_НОМЕР})
- Как видно, атрибут СЛУ_ОТД_НОМ появляется в отношении СЛУЖАЩИЕ не потому, что номер отдела является собственным свойством сотрудника, а лишь для того, чтобы иметь возможность восстановить при необходимости полную сущность ОТДЕЛ
- Значение атрибута СЛУ_ОТД_НОМ в любом кортеже отношения СЛУЖАЩИЕ должно соответствовать значению атрибута ОТД_НОМ в некотором кортеже отношения ОТДЕЛЫ

Реляционная модель данных (10)

Целостность сущности и ссылок (8)

- Атрибут такого рода (возможно, составной) называется *внешним ключом (foreign key)*, поскольку его значения однозначно характеризуют сущности, представленные кортежами некоторого другого отношения
  - т.е. задают значения их первичного ключа
- Конечно, внешний ключ может быть составным, т.е. состоять из нескольких атрибутов
- Говорят, что отношение, в котором определен внешний ключ, ссылается на соответствующее отношение, в котором такой же атрибут является первичным ключом

Реляционная модель данных (11)

Целостность сущности и ссылок (9)

- В полной форме требование целостности по ссылкам, или требование целостности внешнего ключа состоит в том, что для каждого значения внешнего ключа, появляющегося в кортеже значения-отношения ссылающейся переменной отношения,
  - либо в значении-отношении переменной отношения, на которую ведет ссылка, должен найтись кортеж с таким же значением первичного ключа,
  - либо значение внешнего ключа должно быть полностью неопределенным (т.е. ни на что не указывать)
- Для нашего примера это означает, что если для сотрудника указан номер отдела, то этот отдел должен существовать

Реляционная модель данных (12)

Целостность сущности и ссылок (10)

- Заметим, что, как и первичный ключ, внешний ключ должен специфицироваться при определении переменной отношения и представляет собой ограничение на допустимые значения-отношения этой переменной
- Другими словами, определение внешнего ключа представляет собой определение ограничения целостности базы данных
- Ограничения целостности сущности и по ссылкам должны поддерживаться СУБД
- Для соблюдения целостности сущности достаточно гарантировать отсутствие в любой переменной отношения значений-отношений, содержащих кортежи с одним и тем же значением первичного ключа (и запрещать вхождение в значение первичного ключа неопределенных значений)
- С целостностью по ссылкам дело обстоят несколько сложнее

Реляционная модель данных (13)

Целостность сущности и ссылок (11)

- Понятно, что при обновлении ссылающегося отношения достаточно следить за тем, чтобы не появлялись некорректные значения внешнего ключа
- Но как быть при удалении кортежа из отношения, на которое ведет ссылка?
- Здесь существуют три подхода, каждый из которых поддерживает целостность по ссылкам
- Первый подход заключается в том, что вообще запрещается производить удаление кортежа, для которого существуют ссылки
  - т.е. сначала нужно либо удалить ссылающиеся кортежи, либо соответствующим образом изменить значения их внешнего ключа
- При втором подходе при удалении кортежа, на который имеются ссылки, во всех ссылающихся кортежах значение внешнего ключа автоматически становится полностью неопределенным
- Наконец, третий подход (каскадное удаление) состоит в том, что при удалении кортежа из отношения, на которое ведет ссылка, из ссылающегося отношения автоматически удаляются все ссылающиеся кортежи

Заключение

- С очень большой вероятностью вам придется работать с какой-либо SQL-ориентированной СУБД
- Любая компания, производящая подобные СУБД, называет их реляционными системами
- Очень важно отчетливо понимать, какие свойства таких систем действительно являются реляционными, а что в них в действительности отходит от исходных, ясных и строгих идей реляционного подхода и даже противоречит этим идеям
- При наличии такого понимания можно более правильно организовывать базы данных и строить приложения в среде SQL-ориентированной СУБД