

Реляционная база
данных

База данных

В узком смысле слова, база данных — это некоторый набор данных, необходимых для работы.

Однако данные — это абстракция; никто никогда не видел "просто данные"; они не возникают и не существуют сами по себе. Данные суть отражение объектов реального мира.

В широком смысле слова база данных — это совокупность описаний объектов реального мира и связей между ними, актуальных для конкретной прикладной области.

Классификация СУБД от модели данных

Традиционно все СУБД классифицируются в зависимости от модели данных, которая лежит в их основе. Принято выделять:

Иерархическую
модель данных

Сетевую модель
данных

Реляционную
модель данных

Иногда к ним добавляют модель данных на основе инвертированных списков.

Реляционная модель данных

Реляционной считается такая база данных, в которой все данные представлены для пользователя в виде прямоугольных таблиц значений данных, и все операции над базой данных сводятся к манипуляциям с таблицами. Таблица состоит из строк и столбцов и имеет имя, уникальное внутри базы данных. Таблица отражает тип объекта реального мира, а каждая ее строка — конкретный объект.

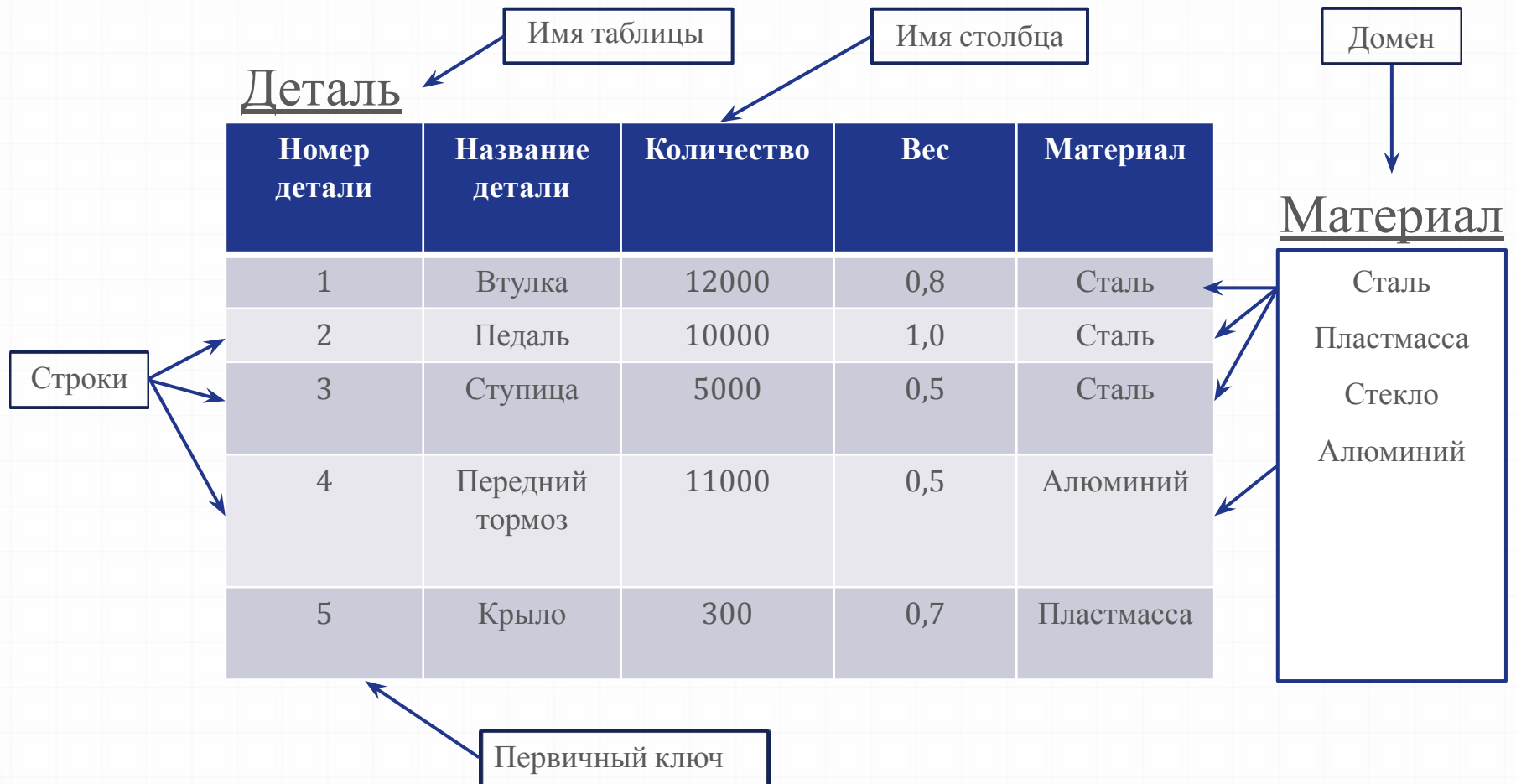
Основные понятия базы данных

Так, таблица Деталь содержит сведения о всех деталях, хранящихся на складе, а ее строки являются наборами значений атрибутов конкретных деталей. Каждый столбец таблицы — это совокупность значений конкретного атрибута объекта. Так, столбец Материал представляет собой множество значений "Сталь", "Олово", "Цинк", "Никель". В столбце Количество содержатся целые неотрицательные числа. Значения в столбце Вес — вещественные числа, равные весу детали в килограммах.

Эти значения не появляются из воздуха. Они выбираются из множества всех возможных значений атрибута объекта, которое называется доменом. Так, значения в столбце материал выбираются из множества имен всех возможных материалов — пластмасс, древесины, металлов и т.д. Следовательно, в столбце Материал принципиально невозможно появление значения, которого нет в соответствующем домене, например, "вода" или "песок".

Каждый столбец имеет имя, которое обычно записывается в верхней части таблицы. Оно должно быть уникальным в таблице, однако различные таблицы могут иметь столбцы с одинаковыми именами. Любая таблица должна иметь по крайней мере один столбец; столбцы расположены в таблице в соответствии с порядком следования их имен при ее создании. В отличие от столбцов, строки не имеют имен; порядок их следования в таблице не определен, а количество логически не ограничено.

Основные понятия базы данных



Взаимосвязь таблиц базы данных

Взаимосвязь таблиц является важнейшим элементом реляционной модели данных. Она поддерживается внешними ключами.

Рассмотрим пример, в котором база данных хранит информацию о рядовых служащих (таблица Служащий) и руководителях (таблица Руководитель) в некоторой организации. Первичный ключ таблицы Руководитель — столбец Номер. Столбец Фамилия не может выполнять роль первичного ключа, так как в одной организации могут работать два руководителя с одинаковыми фамилиями. Любой служащий подчинен единственному руководителю, что должно быть отражено в базе данных. Таблица Служащий содержит столбец Номер руководителя, и значения в этом столбце выбираются из столбца Номер таблицы Руководитель. Столбец Номер Руководителя является внешним ключом в таблице Служащий.

Взаимосвязь таблиц базы данных



Таблицы невозможно хранить и обрабатывать, если в базе данных отсутствуют "данные о данных", например, описатели таблиц, столбцов и т.д. Их называют обычно метаданными. Метаданные также представлены в табличной форме и хранятся в словаре данных.

Помимо таблиц, в базе данных могут храниться и другие объекты, такие как экранные формы, отчеты, представления и даже прикладные программы, работающие с базой данных.

Для пользователей информационной системы недостаточно, чтобы база данных просто отражала объекты реального мира. Важно, чтобы такое отражение было однозначным и непротиворечивым. В этом случае говорят, что база данных удовлетворяет условию целостности.

Для того, чтобы гарантировать корректность и взаимную непротиворечивость данных, на базу данных накладываются некоторые ограничения, которые называют ограничениями целостности.

Ограничительные условия, поддерживающие целостность

В реляционной модели Кодда есть несколько ограничительных условий, используемых для проверки данных в базе данных, а также для придания осмысленности структуре данных. Принято выделять следующие ограничения:

Категорная
целостность

Целостность на
уровне ссылок

Функциональные
зависимости

Целостность категории и ССЫЛОК

В целостной части реляционной модели данных фиксируются два базовых требования целостности, которые должны поддерживаться в любой реляционной СУБД.

Первое требование называется требованием целостности сущности.

Второе требование называется требованием целостности по ссылкам, является более сложным

Операции над реляционными данными

Множество операций над реляционными данными образуют реляционную алгебру. Каждая операция использует одну или две таблицы. Основных операций восемь, которые разбиты на две группы.

Традиционные операции

- 1) Объединение двух отношений ($C1 = A \cup B$) предполагает, что на входе задано два односхемных отношения A и B . Результат объединения есть построенное по той же схеме отношение C , содержащее все кортежи A и все кортежи отношения B .
- 2) Пересечение двух отношений ($C2=A \cap B$) предполагает на входе два односхемных отношения A и B . На выходе создается отношение по той же схеме, содержащее только те кортежи отношения A , которые есть в отношении B .
- 3) Вычитание двух отношений ($C3=A-B$). Все три отношения строятся по одной схеме. В результирующее отношение $C3$ включаются только те кортежи из A , которых нет в отношении B .
- 4) Декартово произведение ($C4=A \times B$). Ее важное отличие от предшествующих состоит в том, что отношения A и B могут быть построены по разным схемам, а схема отношения $C4$ включает все атрибуты отношения A и B .

Специальные операции

- 1) Операция селекция выполняется по строкам. На входе операции используется одно отношение. Результат выборки есть новое отношение, построенное по той же схеме, содержащее подмножество кортежей исходного отношения, удовлетворяющих условию выборки.
- 2) Операция проекция. На входе операции используется одно отношение. Результирующее отношение включает подмножество атрибутов исходного. Каждому кортежу исходного отношения соответствует такой кортеж в результирующем отношении, что значения одинаковых атрибутов этих двух кортежей совпадают. Но при этом в результирующем отношении кортежи-дубликаты устраняются, в связи с чем мощность результирующего отношения может быть меньше мощности исходного.
- 3) Операция соединение естественное. На входе операции используется два отношения. В каждом из отношений выделен атрибут, по которому будет осуществляться соединение. Оба атрибута должны быть определены на одном и том же домене. Схема результирующего отношения включает все атрибуты двух отношений. Допускается, чтобы в схеме результирующего отношения вместо двух атрибутов, по которым выполняется соединение, был представлен только один. Операция соединение похожа на декартово произведение.
- 4) Операция деление. На входе операции используется два отношения A и B . Пусть отношение A , называемое делимым, содержит атрибуты (A_1, A_2, \dots, A_n) . Отношение B – делитель – содержит подмножество атрибутов A ; положим, (A_1, A_2, \dots, A_k) , где $(k < n)$. Результирующее отношение C определено на атрибутах отношения A , которых нет в B , т.е. $A_{k+1}, A_{k+2}, \dots, A_n$. Кортеж включается в результирующее отношение только, если его декартово произведение с отношением B содержится в делимом-отношении A .

Операции реляционной модели данных предоставляют возможность произвольно манипулировать отношениями, позволяя обновлять БД, а также выбирать подмножества хранимых данных и представлять их в нужном виде.

Рассмотренные нами операции реляционной алгебры или алгебры отношений, позволяют пошагово описать процесс получения результирующего отношения.

Нормализация отношений

Одна из важнейших проблем проектирования схемы БД заключается в выделении типов записей, определении состава их атрибутов. Группировка атрибутов должна быть рациональной, т.е. минимизирующей дублирование данных и упрощающей процедуры их обработки и обновления.

Сначала эти вопросы решались интуитивно. Однако интуиция может подвести даже опытного специалиста, поэтому Коддом был разработан в рамках реляционной модели данных аппарат, называемый нормализацией отношений. И хотя идеи нормализации сформулированы в терминологии реляционной модели данных, они в равной степени применимы и для других моделей данных.

Коддом выделено три нормальных формы отношений. Самая совершенная из них - третья. Предложен механизм, позволяющий любое отношение преобразовать к третьей нормальной форме. В процессе таких преобразований могут выделяться новые отношения.

Первая нормальная форма

Отношение называется нормализованным или приведенным к первой нормальной форме (1НФ), если все его атрибуты простые.

Ненормализованное отношение легко сделать нормализованным. Такое преобразование может привести к увеличению мощности отношения и изменению ключа.

Функциональная зависимость. Пусть X и Y - два атрибута некоторого отношения, Говорят, что Y функционально зависит от X , если в любой момент времени каждому значению X соответствует не более чем одно значение атрибута Y . Функциональную зависимость можно обозначить так: $X > Y$.

Полная функциональная зависимость. Говорят, что не ключевой атрибут функционально полно зависит от составного ключа, если он функционально зависит от ключа, но не находится в функциональной зависимости ни от какой части составного ключа.

Вторая нормальная форма

Отношение находится во второй нормальной форме, если оно находится в первой нормальной форме и каждый не ключевой атрибут функционально полно зависит от составного ключа.

Чтобы отношение привести ко второй нормальной форме, необходимо:

- 0 построить его проекцию, исключив атрибуты, которые не находятся в полной функциональной зависимости от составного ключа;
- 0 построить дополнительно одну или несколько проекций на часть составного ключа и атрибуты, функционально зависящие от этой части ключа.

Транзитивная зависимость. Пусть X, Y, Z - три атрибута некоторого отношения. При этом $X \rightarrow Y$ и $Y \rightarrow Z$, но обратное соответствие отсутствует, т. е. $Z \not\rightarrow Y$ или $Y \not\rightarrow X$. Тогда говорят, что Z транзитивно зависит от X .

Третья нормальная форма

Отношение находится в третьей нормальной форме, если оно находится во второй нормальной форме и каждый неключевой атрибут не транзитивно зависит от первичного ключа. Рассматриваемая версия третьей нормальной формы часто называется нормальной формой Бойса-Кодда (НФБК).

Другие нормальные формы

Первая нормальная форма запрещает таблицам иметь неатомарные, или многозначные атрибуты. Однако существует множество ситуаций моделирования, требующих многозначных атрибутов. Например, преподаватель в вузе отвечает за несколько дисциплин. Существует несколько решений, каждое из которых имеет определенные недостатки. Все они требуют лишней памяти из-за наличия пустых значений, либо из-за необходимости вводить избыточные данные. Те из них, в которых есть пустые значения, нарушают категорийную целостность, поскольку все атрибуты вместе составляют ключ таблицы. Эти кажущиеся связи между независимыми атрибутами можно исключить, потребовав, чтобы каждое значение атрибута сочеталось с каждым значением другого атрибута как минимум в одной строке. Условие, обеспечивающее независимость атрибутов путем обязательного повторения значений, называется многозначной зависимостью. Многозначная зависимость является таким же ограничительным условием, как функциональная зависимость. Очевидно, что поскольку они требуют огромного числа повторений значений данных, важный этап процесса нормализации состоит в избавлении от многозначных зависимостей.

Таблица имеет четвертую нормальную форму (4НФ), если она имеет 3НФ и не содержит многозначных зависимостей.

Для избавления от некоторых других аномалий были предложены еще несколько нормальных форм: пятая нормальная форма (5НФ), нормальная форма область/ключ (НФОК) и т.д. Однако они имеют очень ограниченное практическое использование.

Заключение

Необходимо подчеркнуть, что настоящая работа не дает рецепта построения хорошей схемы базы данных. Она, скорее, обозначает проблему и объясняет, как ее можно решить в общем виде. Для того чтобы дать практические рекомендации необходимо выполнить следующие шаги:

1. Выбрать концептуальную модель, с помощью которой будет построена концептуальная схема;
2. Построить точное описание семантических ограничений, поддерживаемых выбранной СУБД;
3. Построить отображение выбранной концептуальной модели в модель данных, поддерживаемую СУБД.
4. Определить, что такое хорошая схема и описать методику ее построения.