Содержание:

Реляционные системы

Реляционные системы далеко не сразу получили широкое распространение. В то время как основные теоретические результаты в этой области были получены еще в 70-х годах и тогда же появились первые прототипы реляционных СУБД, долгое время считалось невозможным добиться эффективной реализации таких систем. Однако постепенное накопление методов и алгоритмов организации реляционных баз данных и управления ими привели к тому, что уже в середине 80-х годов реляционные системы практически вытеснили с мирового рынка ранние СУБД.

Реляционная модель данных основывается на математических принципах, вытекающих непосредственно из теории множеств и логики предикатов. Эти принципы впервые были применены в области моделирования данных в конце 1960-х гг. доктором ***Е.Ф. Коддом****, в то время работавшим в IBM, а впервые опубликованы - в 1970 г.

Эдгар Франк «Тед» Кодд — британский учёный, работы которого заложили основы теории реляционных баз данных.

Работая в компании IBM, он создал реляционную модель данных. Он также внёс существенный вклад в другие области информатики.

Биография

- Родился в Портланде (Дорсет) в Англии. Его отец был производителем кожи, а его мама была учительницей. Обучался математике и химии в Оксфордском университете (Exeter College).
- Во время Второй мировой войны служил пилотом в военно-воздушных силах.
- В 1948 переехал в Нью-Йорк, чтобы работать в IBM как математик-программист.

- В 1953, из-за преследований со стороны сенатора Джозефа Маккарти (Joseph McCarthy), Кодд переехал в Оттаву (Канада).
- В 1963 он вернулся в США и получил докторскую степень по информатике и вычислительной технике в Университете Мичигана (University of Michigan, Ann Arbor). В 1965 он переехал в Сан-Хосе (Калифорния), чтобы работать в Альмаденском Исследовательском Центре IBM.
- В 60-х — 70-х годах он работал над своими теориями хранения данных. В 1970 издал работу «A Relational Model of Data for Large Shared Data Banks», которая считается первой работой по реляционной модели данных.
- Кодд продолжил разрабатывать и расширять реляционную модель. Одна из нормальных форм названа в его честь (Нормальная форма Бойса — Кодда).
- В начале 80-х реляционная модель начала входить в моду. Борясь с недобросовестными поставщиками СУБД, которые утверждали, что их устаревшие продукты поддерживают реляционную технологию, Кодд опубликовал «12 правил Кодда», описывающие, что должна содержать реляционная СУБД. Его борьба коснулась языка SQL, который Кодд считал неправильной реализацией теории. Это делало его положение в IBM достаточно тяжелым, так как та поставляла продукты, основанные на SQL. Он покинул IBM и организовал вместе с Кристофером Дейтом и несколькими другими людьми собственную консалтинговую компанию.
- Кодд ввёл в оборот термин OLAP и написал 12 законов аналитической обработки данных. Он также занимался клеточными автоматами.
- В 1976 Кодд получил почетное звание IBM Fellow. В 1981 он получил премию Тьюринга.
- В 2002 журнал Forbes поместил реляционную модель данных в список важнейших инноваций последних 85 лет.
- Эдгар Ф. Кодд умер от сердечного приступа у себя дома во Флориде на острове Вильямс в возрасте 79 лет в пятницу 18 апреля 2003. У него было четверо детей и шесть внуков.

12 правил Кодда (англ. *Codd’s 12 rules*) — 13 правил (в данном случае исчисление начинается с 0), которым должна удовлетворять каждая система управления реляционными базами данных.

Предложены английским математиком Эдгаром Коддом (Edgar Codd) в 1985 году в статьях в журнале *ComputerWorld*.

Правило 0: Основное правило (Foundation Rule):

Система, которая рекламируется или позиционируется как реляционная система управления базами данных, должна быть способна управлять базами данных, используя исключительно свои реляционные возможности.

Правило 1: Информационное правило (The Information Rule):

Вся информация в реляционной базе данных на логическом уровне должна быть явно представлена единственным способом: значениями в таблицах.

Правило 2: Гарантированный доступ к данным (Guaranteed Access Rule):

В реляционной базе данных каждое отдельное (атомарное) значение данных должно быть логически доступно с помощью комбинации имени таблицы, значения первичного ключа и имени столбца.

Правило 3: Систематическая поддержка отсутствующих значений (Systematic Treatment of Null Values):

Неизвестные, или отсутствующие значения NULL, отличные от любого известного значения, должны поддерживаться для всех типов данных при выполнении любых операций. Например, для числовых данных неизвестные значения не должны рассматриваться как нули, а для символьных данных — как пустые строки.

Правило 4: Доступ к словарю данных в терминах реляционной модели (Active On-Line Catalog Based on the Relational Model):

Словарь данных должен сохраняться в форме реляционных таблиц, и СУБД должна поддерживать доступ к нему при помощи стандартных языковых средств, тех же самых, которые используются для работы с реляционными таблицами, содержащими пользовательские данные.

Правило 5: Полнота подмножества языка (Comprehensive Data Sublanguage Rule):

Система управления реляционными базами данных должна поддерживать хотя бы один реляционный язык, который

(а) имеет линейный синтаксис,

(б) может использоваться как интерактивно, так и в прикладных программах,

(в) поддерживает операции определения данных, определения представлений, манипулирования данными (интерактивные и программные), ограничители целостности, управления доступом и операции управления транзакциями(begin, commit и rollback).

Правило 6: Возможность изменения представлений (View Updating Rule):

Каждое представление должно поддерживать все операции манипулирования данными, которые поддерживают реляционные таблицы: операции выборки, вставки, изменения и удаления данных.

Правило 7: Наличие высокоуровневых операций управления данными (High-Level Insert, Update, and Delete):

Операции вставки, изменения и удаления данных должны поддерживаться не только по отношению к одной строке реляционной таблицы, но и по отношению к любому множеству строк.

Правило 8: Физическая независимость данных (Physical Data Independence):

Приложения не должны зависеть от используемых способов хранения данных на носителях, от аппаратного обеспечения компьютеров, на которых находится реляционная база данных.

Правило 9: Логическая независимость данных (Logical Data Independence):

Представление данных в приложении не должно зависеть от структуры реляционных таблиц. Если в процессе нормализациидна реляционная таблица разделяется на две, представление должно обеспечить объединение этих данных, чтобы изменение структуры реляционных таблиц не сказывалось на работе приложений.

Правило 10: Независимость контроля целостности (Integrity Independence):

Вся информация, необходимая для поддержания целостности, должна находиться в словаре данных. Язык для работы с данными должен выполнять проверку входных данных и автоматически поддерживать целостность данных.

Правило 11: Независимость от расположения (Distribution Independence):

База данных может быть распределённой, может находиться на нескольких компьютерах, и это не должно оказывать влияния на приложения. Перенос базы данных на другой компьютер не должен оказывать влияния на приложения.

Правило 12: Согласование языковых уровней (The Nonsubversion Rule):

Если используется низкоуровневый язык доступа к данным, он не должен игнорировать правила безопасности и правила целостности, которые поддерживаются языком более высокого уровня.

Кодд предложил применение реляционной алгебры в СУРБД, для расчленения данных в связанные наборы. Он организовал свою систему БД вокруг концепции, основанной на наборах данных.

В реляционной модели данные разбиваются на наборы, которые составляют табличную структуру. Эта структура таблиц состоит из индивидуальных элементов данных, называемых полями. Одиночный набор или группа полей известна как запись.

Формулируя принципы реляционной модели, доктор *Кодд* выбрал термин "отношение" (relation), потому что, по его мнению, этот термин однозначен (в то время как, например, термин "таблица" имеет множество различных видов - таблица в тексте, электронная таблица и пр.). Весьма распространено следующее заблуждение: *реляционная модель* названа так потому, что она определяет связи между таблицами. На самом деле, название этой модели происходит от отношений (таблиц базы данных), лежащих в ее основе.

Каждая строка, содержащая данные, называется ***кортежем***, каждый столбец отношения называется ***атрибутом*** (на уровне практической работы с современными реляционными БД используются термины "запись" и "поле").

Элементами описания реляционной модели данных на концептуальном уровне являются *сущности*, *атрибуты*, *домены* и связи

Сущность

Сущность - некоторый обособленный объект или событие, информацию о котором необходимо сохранять в базе данных, имеющий определенный набор свойств - *атрибутов*. Сущности могут быть как физические, так и абстрактные. Для *сущностей* различают ее тип и экземпляр. Тип характеризуется именем и списком свойств, а экземпляр - конкретными значениями свойств.

Атрибуты сущности бывают:

1. Идентифицирующие и описательные. Идентифицирующие *атрибуты* имеют уникальное значение для *сущностей* данного типа и являются *потенциальными ключами*. Они позволяют однозначно распознавать экземпляры *сущности*. Из *потенциальных ключей* выбирается один **первичный ключ** (ПК). В качестве ПК обычно выбирается потенциальный ключ, по которому чаще происходит обращение к *экземплярам записи*. ПК должен включать в свой состав минимально необходимое для идентификации количество *атрибутов*. Остальные ***атрибуты*** называются описательными.
2. Простые и составные. Простой *атрибут* состоит из одного компонента, его значение неделимо. Составной *атрибут* является комбинацией нескольких компонентов, возможно, принадлежащих разным типам данных (например, адрес). Решение о том, использовать составной *атрибут* или разбивать его на компоненты, зависит от особенностей процессов его использования и может быть связано с обеспечением высокой скорости работы с большими базами данных.
3. Однозначные и многозначные - могут иметь соответственно одно или много значений для каждого экземпляра сущности.
4. Основные и производные. Значение основного *атрибута* не зависит от других *атрибутов*. Значение производного *атрибута* вычисляется на основе значений других *атрибутов* (например, возраст человека вычисляется на основе даты его рождения и текущей даты

Спецификация *атрибута* состоит из его названия, указания типа данных и описания ограничений целостности - множества значений (или домена), которые может принимать данный *атрибут*.

Домен

Домен - это набор всех допустимых значений, которые может содержать *атрибут*. Понятие **"*домен*"** часто путают с понятием "тип данных". Необходимо различать эти два понятия. Тип данных - это физическая концепция, а домен - логическая. Например, "целое число" - это тип данных, а "возраст" - это домен.

Связи

Связи - на концептуальном уровне представляют собой простые ассоциации между *сущностями*. Например, утверждение "Покупатели приобретают продукты" указывает, что между *сущностями* "Покупатели" и "Продукты" существует связь, и такие *сущности* называются **участниками** этой связи.

Существует несколько типов связей между двумя *сущностями*: это связи " **один к одному** ", " **один ко многим** " и " **многие ко многим** ".

Диаграмма "сущности-связи" (*Entity-Relationship diagrams*, или E/R diagram) служит для описания схемы базы на концептуальном уровне проектирования. Метод был предложен в 1976 г. Питером Пин Шань Ченом (Peter Pin Shan Chen). На диаграммах "сущности-связи" *сущности*изображаются в виде прямоугольников, *атрибуты* - в виде эллипсов, а связи - в виде ромбов (см. рис. 2.6).

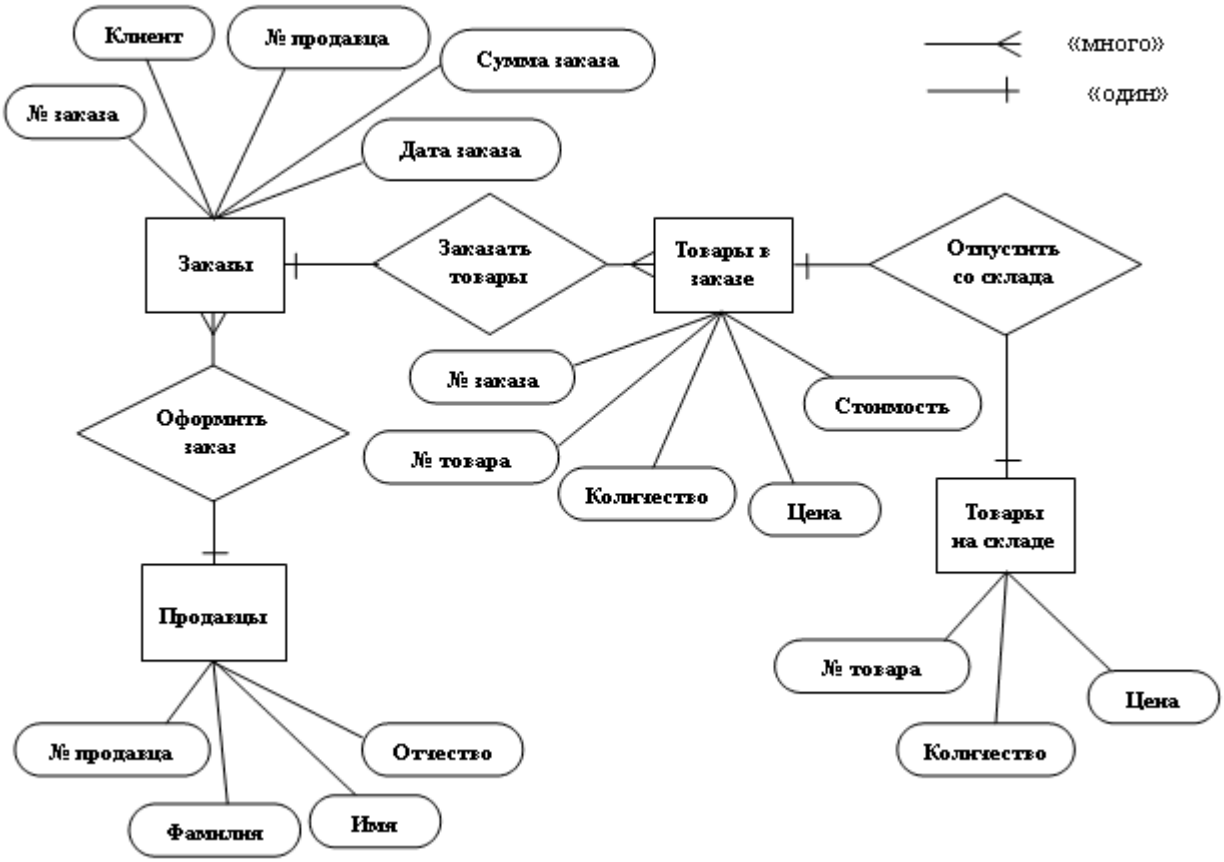

Рис. 2.6. Диаграмма "сущности-связи"

В дальнейшем многими авторами были разработаны свои варианты подобных моделей (нотация Мартина, нотация *IDEF1X*, нотация Баркера и др.).

В рамках реляционной модели данных *Э.Ф. Коддом* были разработаны принципы нормализации отношений и предложен механизм, позволяющий любое *отношение* преобразовать к *третьей нормальной форме*.

Нормализация - это формальный метод анализа *отношений* на основе их первичного ключа и существующих связей. Ее задача - это замена одной схемы (или совокупности *отношений*) БД другой схемой, в которой *отношения* имеют более простую и регулярную структуру.

Первая нормальная форма (1НФ) связана с понятиями простого и сложного *атрибутов*. Простой ***атрибут*** - это *атрибут*, значения которого атомарны (т.е. неделимы). Сложный *атрибут* может иметь значение, представляющее собой объединение нескольких значений одного или разных доменов. В первой нормальной форме устраняются повторяющиеся *атрибуты* или группы *атрибутов*, т.е. производится выявление неявных *сущностей*, "замаскированных" под *атрибуты* .

Отношение приведено к 1НФ, если все его атрибуты - простые, т.е. значение *атрибута* не должно быть множеством или повторяющейся группой.

Для приведения таблиц к *1НФ* необходимо разбить сложные *атрибуты* на простые, а многозначные *атрибуты* вынести в отдельные *отношения*.

Вторая нормальная форма (2НФ) применяется к *отношениям* с составными ключами (состоящими из двух и более *атрибутов*) и связана с понятиями функциональной зависимости.

Если в любой момент времени каждому значению *атрибута* A соответствует единственное значение *атрибута* B, то B функционально зависит от A ($A \longrightarrow B$). Атрибут (группа ***атрибутов***) A называется *детерминантом*.

Во второй нормальной форме устраняются *атрибуты*, зависящие только от части уникального ключа. Эта часть уникального ключа определяет отдельную *сущность*.

Отношение находится во 2НФ, если оно приведено к 1НФ и каждый неключевой атрибут функционально полно зависит от составного первичного ключа.

Третья нормальная форма (3НФ) связана с понятием транзитивной зависимости. Пусть A, B, C - *атрибуты* некоторого *отношения*. При этом $A \longrightarrow B$ и $B \longrightarrow C$, но обратное соответствие отсутствует, т.е. C не зависит от B или B не зависит от A.

Тогда говорят, что C транзитивно зависит от A (A $\longrightarrow\longrightarrow$ C).

В *третьей нормальной форме* устраняются *атрибуты*, которые зависят от *атрибутов* , не входящих в уникальный ключ. Эти *атрибуты* являются основой отдельной *сущности*.

Отношение находится в 3НФ, если оно находится во 2НФ и не имеет атрибутов, не входящих в первичный ключ и находящихся в транзитивной зависимости от первичного ключа.

Существуют также *нормальная форма Бойса-Кодда* (НФБК), 4НФ и 5НФ. ***Однако наибольшее значение имеет 1НФ***, т.к. последующие НФ связаны с понятиями о составных ключах и сложных зависимостях от ключей, а на практике встречаются обычно более простые случаи.

Моделирование структуры базы данных при помощи алгоритма *нормализации* имеет серьезные недостатки:

1. Методика *нормализации* предполагает *первоначальное размещение* всех *атрибутов* проектируемой предметной области в одном *отношении*, что является очень неестественной операцией. Интуитивно разработчик сразу проектирует несколько *отношений* в соответствии с обнаруженными *сущностями*. Даже если совершить насилие над собой и создать одно или несколько *отношений*, включив в них все предполагаемые *атрибуты*, то совершенно неясен смысл полученного *отношения*.
2. Невозможно сразу определить полный список *атрибутов*. Пользователи имеют привычку называть разными именами одни и те же вещи или наоборот, называть одними именами разные вещи.
3. Для проведения процедуры *нормализации* необходимо выделить зависимости *атрибутов*, что тоже очень нелегко.

Ссылки

https://ru.wikipedia.org/wiki/12_правил_Кодда

https://www.intuit.ru/studies/courses/93/93/lecture/28077?page=4

http://mediaknowledge.ru/c736d15c99e5c4c3.html