ВВЕДЕНИЕ В СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ (СУБД)

РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ

Неотъемлемой частью повседневной жизни стали базы данных, для поддержки которых требуется некоторый организационный метод, или механизм.

Такой механизм называется системой управления базами данных **(СУБД).**

РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ

База данных (БД) – это организованная в соответствии с определенными правилами совокупность данных, характеризующая состояние некоторой предметной области и используемая для удовлетворения информационных потребностей пользователей.

РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ

Система управления базами данных (СУБД) – это программная система, предназначенная для создания и хранения базы данных, предоставления к ней санкционированного доступа, а также поддержки функций администратора базы данных.

РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ

Ключевую роль при обеспечении эффективного хранения данных играют методы поддержания логических связей между данными. По способам организации связей выделяют различные модели данных. Большинство современных СУБД используют реляционную модель данных.

РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ

Реляционная модель данных впервые была предложена американским математиком Коддом в 1970 г. Фундаментальным понятием реляционной базы данных является отношение. Это отражено и в общем названии подхода – термин реляционный (relational) происходит от relation (отношение). На физическом уровне отношения представляют собой таблицы. В реляционной модели все данные представлены в виде простых таблиц, разбитых на строки и столбцы.

РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ

В 1985 г. Коддом были сформулированы правила, которым должна удовлетворять любая реляционная база данных. Приведенные ниже правила Кодда могут являться определением реляционной СУБД.

ПРАВИЛА КОДДА

1. **Правило информации**. Вся информация в базе данных должна быть представлена исключительно на логическом уровне и только одним способом – в виде значений, содержащихся в таблицах.
2. **Правило гарантированного доступа**. Логический доступ ко всем и к каждому элементу данных (атомарному значению) в реляционной базе данных должен обеспечиваться путем использования комбинации имени таблицы, первичного ключа и имени столбца.

   Данное правило указывает на роль первичных ключей при поиске информации в базе данных. Имя таблицы позволяет найти требуемую таблицу, имя столбца позволяет найти требуемый столбец, а первичный ключ позволяет найти строку, содержащую искомый элемент данных.

ПРАВИЛА КОДДА

3. **Правило поддержки недействительных значений**. В настоящей реляционной базе данных должна быть реализована поддержка недействительных значений, которые отличаются от строки символов нулевой длины, строки пробельных символов и от нуля или любого другого числа и используются для представления отсутствующих данных независимо от типа этих данных.

   Это правило требует, чтобы отсутствующие данные можно было представить с помощью недействительных значений, то есть NULL. Если в каком либо поле содержится значение NULL, то оно указывает на отсутствие каких-либо данных в этом поле.

ПРАВИЛА КОДДА

4. Правило динамического каталога, основанного на реляционной модели. Описание базы данных на логическом уровне должно быть представлено в том же виде, что и основные данные, чтобы пользователи, обладающие соответствующими правами, могли работать с ним с помощью того же реляционного языка, который они применяют для работы с основными данными.

Реляционная база данных должна сама себя описывать, то есть должна содержать набор системных таблиц, описывающих структуру самой базы данных. Данные о данных, содержащихся в базе данных, называются метаданными.

ПРАВИЛА КОДДА

5. Правило исчерпывающего подъязыка данных. Реляционная система может поддерживать различные языки и режимы взаимодействия с пользователем (например, режим вопросов и ответов). Однако, должен существовать, по крайней мере, один язык, операторы которого можно представить в виде строк символов, в соответствии с некоторым четко определенным синтаксисом и который в полной мере поддерживает следующие элементы:

ПРАВИЛА КОДДА

- определение данных;
- определение представлений;
- обработку данных (интерактивную и программную);
- условия целостности;
- идентификацию прав доступа;
- границы транзакций (начало, завершение и отмену).

СУБД должна использовать язык реляционной базы данных, например SQL. Такой язык должен поддерживать все основные функции СУБД: создание базы данных, чтение и ввод данных, реализацию защиты базы данных, построение запросов и т.д.

ПРАВИЛА КОДДА

6. Правило обновления представлений. Все представления, которые теоретически можно обновить, должны быть доступны для обновления.

Представления являются виртуальными таблицами, позволяющими показывать различным пользователям различные фрагменты структуры базы данных.

7. Правило добавления, обновления, и удаления. Возможность работать с отношением (таблицей) как с одним операндом должна существовать не только при чтении данных, но и при добавлении, обновлении и удалении данных.

Данное правило требует, чтобы операции добавления удаления и обновления можно было выполнять над множествами строк. Это правило предназначено для того, чтобы запретить реализации, в которых поддерживаются только операции над одной строкой.

ПРАВИЛА КОДДА

8. Правило независимости физических данных. Прикладные программы и утилиты для работы с данными должны на логическом уровне оставаться нетронутыми при любых изменениях способов хранения данных или методов доступа к ним.

9. Правило независимости логических данных. Прикладные программы и утилиты для работы с данными должны на логическом уровне оставаться нетронутыми при внесении в базовые таблицы любых изменений, которые теоретически позволяют сохранить нетронутыми содержащиеся в этих таблицах данные.

Правила 8 и 9 означают отделение пользователя и прикладной программы от низкоуровневой реализации базы данных. Они утверждают, что конкретные способы реализации и хранения или доступа, используемые в СУБД , и даже изменения структуры таблиц базы данных не должны влиять на возможность пользователя работать с данными.

ПРАВИЛА КОДДА

10. **Правило независимости условий целостности**. Должна существовать возможность определить условия целостности, специфические для конкретной реляционной базы данных, на подъязыке реляционной базы данных и хранить их в каталоге, а не в прикладной программе.

    Язык базы данных должен поддерживать ограничительные условия, налагаемые на вводимые данные и действия, которые могут быть выполнены над данными.
11. **Правило независимости распространения**. Реляционная СУБД не должна зависеть от потребностей конкретного пользователя.

    Язык базы данных должен обеспечивать возможность работы с распределенными данными, расположенными на других компьютерных системах.

ПРАВИЛА КОДДА

12. **Правило единственности**. Если в реляционной системе есть низкоуровневый язык (обрабатывающий одну запись за одно действие), то должна отсутствовать возможность использования его для того, чтобы обойти правила и условия целостности, выраженные на реляционном языке высокого уровня (обрабатывающем несколько записей за одно действие).

    Данное правило предотвращает использование других возможностей для работы с базой данных, помимо языка базы данных, поскольку это может нарушить ее целостность.

Таким образом, можно сформулировать следующее определение реляционной базы данных.

Реляционной называется база данных, в которой все данные, доступные пользователю, организованы в виде прямоугольных таблиц, а все операции над данными сводятся к операциям над этими таблицами.

Таблицы

В реляционной базе данных информация содержится в виде реляционных таблиц, разделенных на строки и столбцы, на пересечении которых содержатся значения данных.

Таблица – это некоторая регулярная структура, состоящая из конечного набора однотипных записей.

Таблица отражает тип объекта реального мира (сущность). Строки таблицы соответствуют экземпляру объекта, конкретному событию или явлению. Столбцы соответствуют атрибутам (признакам, характеристикам, параметрам) объекта, события, явления. У каждой таблицы имеется уникальное имя внутри базы данных, описывающее ее содержимое.

Таблицы

У каждого столбца в таблице есть свое имя, которое обычно служит заголовком столбца. Все столбцы в одной таблице должны иметь уникальные имена, однако разрешается присваивать одинаковые имена столбцам, расположенным в различных таблицах. В реляционной модели данных атрибуты отношений не упорядочены, т.е обращение к полям всегда происходит по именам, а не по расположению. В языке SQL допускается индексное указание столбцов таблиц, при этом столбцы рассматриваются в порядке слева направо (их порядок определяется при создании таблицы). В любой таблице всегда есть как минимум один столбец. Как правило, существует предел на количество столбцов в таблице. В большинстве СУБД в таблицах может содержаться более 30 000 столбцов

Таблицы

В реляционной модели данных для обозначения строки отношения используется понятие **кортеж**. Представлением кортежа на физическом уровне является строка таблицы базы данных. Строки таблицы не имеют имен и определенного порядка. В таблице может содержаться любое количество строк. Допустимым является и существование таблицы с нулевым количеством строк. Такая таблица называется пустой. Пустая таблица сохраняет структуру, определенную ее столбцами, просто в ней не содержатся данные. В ряде СУБД для количества строк в таблицах существует максимальный предел. Как правило, он составляет около 2 миллиардов строк.

Каждая горизонтальная строка таблицы является записью и представляет отдельную физическую сущность. Каждый вертикальный столбец таблицы представляет совокупность значений конкретного атрибута объекта. На пересечении каждой строки с каждым столбцом таблицы может содержаться в точности только одно значение данных.

Все значения, содержащиеся в одном и том же столбце, являются данными одного типа. В реляционной модели данных общая совокупность значений, из которой берутся действительные значения для определенных атрибутов (столбцов), называется **доменом**. Каждый столбец всегда определяется на одном домене. В реляционных базах данных домен определяется путем задания как минимум некоторого базового типа данных, к которому относятся элементы домена, а часто также и произвольного логического выражения, применяемого к элементам этого типа данных (ограничения домена).

Первичные и внешние ключи

Строки в реляционной базе данных не упорядочены, поэтому нельзя выбрать строку по ее номеру в таблице. В таблице нет “первой”, “второй”, ”девятой” или “последней” строки. Конкретная строка в таблице может быть определена по ключевому элементу.

Ключевым элементом данных называется такой элемент, по которому можно определить значения других элементов данных.

В реляционной базе данных в каждой таблице есть один или несколько столбцов, значения которых во всех строках разные. Этот столбец (столбцы) называется первичным ключом таблицы.

Первичный ключ – это атрибут или группа атрибутов, которые единственным образом идентифицируют каждую строку таблицы.

Если в таблице нет полей, значения которых во всех строках (записях, кортежах) уникальны, то для создания первичного ключа в нее обычно вводят дополнительное поле, значениями которого СУБД может распоряжаться по своему усмотрению.

Если первичный ключ представляет собой комбинацию столбцов, то такой первичный ключ называется **составным**.

Первичные и внешние ключи

Вторичные ключи устанавливаются по полям, которые часто используются при поиске или сортировке данных. В отличие от первичных ключей поля для вторичных ключей могут содержать не уникальные значения. Использование вторичных ключей, в большинстве случаев, увеличивает производительность СУБД.

Столбец одной таблицы, значения в котором совпадают со значениями столбца, являющегося первичным ключом другой таблицы, называется **внешним ключом.** Обычно в качестве первичного и внешнего ключей используются столбцы с одинаковыми именами из двух различных таблиц.

Внешний ключ, как и первичный ключ, тоже может представлять собой комбинацию столбцов. На практике внешний ключ всегда будет составным (состоящим из нескольких столбцов), если он ссылается на составной первичный ключ другой таблицы. Очевидно, что количество столбцов и их типы данных в первичном и внешнем ключах совпадают.

Если таблица связана с несколькими другими таблицами, она может иметь несколько внешних ключей. Внешние ключи реализуют отношения (связи) между таблицами базы данных.

Типы связей между таблицами

В реляционных базах данных между таблицами существуют связи (отношения). Если между некоторыми сущностями существует связь, то факты из одной сущности ссылаются или некоторым образом связаны с фактами другой сущности. Связь работает путем сопоставления первичного ключа одной таблицы (родительской сущности) с элементом внешнего ключа другой таблицы (дочерней сущности). Первичный и соответствующий ему внешний ключ помогают реализовать отношение родитель потомок между таблицами. В базе данных нужно хранить только актуальные, значимые связи.

Связи могут отличаться по типу связи (идентифицирующая, не идентифицирующая, полная и неполная категории, неспецифическая связь), по мощности связи, допустимости пустых (NULL) значений.

Типы связей между таблицами

Связь называется **идентифицирующей**, если экземпляр дочерней сущности идентифицируется (однозначно определяется) через ее связь с родительской сущностью. Атрибуты, составляющие первичный ключ родительской сущности, при этом входят в первичный ключ дочерней сущности. Дочерняя сущность при идентифицирующей связи всегда является зависимой.

Связь называется **не идентифицирующей**, если экземпляр дочерней сущности идентифицируется иначе, чем через связь с родительской сущностью. Атрибуты, составляющие первичный ключ родительской сущности, при этом входят в состав не ключевых атрибутов дочерней сущности.

Типы связей между таблицами

Мощность связи представляет собой отношение количества экземпляров родительской сущности к соответствующему количеству дочерней сущности. По мощности связи выделяют отношения:

- “один к одному”;
- “один ко многим”;
- “многие ко многим”.

Типы связей между таблицами

При связи **“один к одному”** одной строке родительской таблицы может соответствовать не более одной строки дочерней таблицы (и наоборот). Такая связь создается, если оба связанных столбца являются первичными ключами или имеют ограничение, обеспечивающее их уникальность. Связи этого типа встречаются редко, поскольку связанную подобным образом информацию обычно удается поместить в одной таблице.

Типы связей между таблицами

“Один ко многим” – наиболее распространенный тип связи. При этом типе связи одной строке родительской таблицы может соответствовать множество строк дочерней таблицы, но любой строке дочерней таблицы может соответствовать только одна строка родительской таблицы. Отношение “один ко многим”, существующее между атрибутами, в реляционной модели реализовано в виде одинаковых значений данных, хранящихся в двух таблицах. Все отношения, существующие между таблицами реляционной базы данных, реализуются в таком виде.

Типы связей между таблицами

При связи “**многие ко многим**” (неспецифическое отношение) одной строке родительской таблицы может соответствовать множество строк дочерней таблицы (и наоборот). Такая связь создается с помощью третьей таблицы, первичный ключ которой состоит из внешних ключей таблиц, связанных отношением “многие ко многим”.

Нормализация таблиц

Процесс нормализации баз данных был впервые предложен Коддом в 1970 г. Этот процесс основан на понятии функциональной зависимости.

Функциональная зависимость– это такая связь между двумя столбцами одной таблицы, когда каждому значению одного столбца соответствует только одно значение другого столбца.

Нормализация обычно приводит к разделению одной таблицы на две или более таблиц, соответствующих требованиям нормальных форм.

Нормализация таблиц

В 1970 г. Коддом было предложено только три вида нормальных форм: первая нормальная форма (1НФ), вторая нормальная форма (2НФ) и третья нормальная форма (3НФ).

Затем Бойсом и Коддом в 1974 г. Было сформировано более строгое определение третьей нормальной формы, которое получило название нормальной формы Бойса –Кодда. (НФБК).

Вслед за НФБК появились определения четвертой (4НФ) и пятой (5НФ) нормальных форм в 1977 и в 1979 г.

В 1981 г. Р. Фагин ввел новую нормальную форму, названную доменно-ключевой (ДКНФ).

Нормализация таблиц

На практике при проектировании таблиц, как правило использую первые три нормальные формы.

Список всех нормальных форм представлен ниже:

- Первая нормальная форма (1НФ)
- Вторая нормальная форма (2НФ)
- Третья нормальная форма (3НФ)
- Нормальная форма Бойса-Кодда (НФБК)
- Четвертая нормальная форма (4НФ)
- Пятая нормальная форма (5НФ)
- Доменно-ключевая нормальная форма (ДКНФ)

Все нормальные формы вложены друг в друга , то есть каждая последующая форма удовлетворяет требованиям предыдущей. Таблица в 2НФ является также и таблицей в 1НФ, таблица в 3НФ является таблицей и в 2НФ и в 1НФ, и т.д.

Нормализация таблиц

Первая нормальная форма (1НФ). Требует, чтобы на пересечении строки и столбца находилось единственное значение, которое должно быть атомарным (неделимым). В таблице, удовлетворяющей 1НФ, не должно быть повторяющихся групп.

Основные характеристики таблицы в 1НФ:

- в каждой строке таблицы должны содержаться данные, соответствующие некоторому объекту или его части
- в каждом столбце должны находиться данные, соответствующие одному из атрибутов отношения
- в каждой ячейке таблицы должно находиться только единственное значение
- у каждого столбца должно быть уникальное имя
- все строки (записи) в таблице должны быть различными
- порядок расположения столбцов и строк в таблице не имеет значения.

Нормализация таблиц

Вторая нормальная форма (2НФ) основана на понятии полной функциональной зависимости. Каждая таблица в 1НФ должна иметь первичный ключ. Он может состоять из одного или более столбцов (атрибутов). В последнем случае ключ называется составным. Таблица находится в 2НФ, если она находится в 1НФ и каждый ее столбец, не входящий в состав первичного ключа, функционально полно зависит от первичного ключа. Чтобы таблица была в 2НФ, все ее неключевые столбцы должны однозначно определяться всем ключом, то есть всеми его компонентами, а не некоторыми из них. Таким образом, второе правило нормализации требует, чтобы любой неключевой столбец зависел от всего первичного ключа, а не от его отдельных компонентов. Если первичный ключ является простым (состоит из одного столбца), то таблица автоматически находится в 2НФ.

Нормализация таблиц

Третья нормальная форма (3НФ) основана на понятии транзитивной зависимости. В таблицах могут быть так называемые транзитивные зависимости, которые являются источником аномалий модификации данных., против которых 2НФ бессильна. Транзитивная зависимость имеет место тогда, когда один атрибут однозначно определяет второй, второй определяет третий и т.д. Если для столбцов A, B и C некоторого отношения существуют зависимости C от B и B от A, то говорят, что атрибут C транзитивно зависит от атрибута A через атрибут B. Отношение находится в 3НФ, если оно находится в 1НФ и 2НФ и в нем не существует транзитивных зависимостей неключевых атрибутов от первичного ключа. Таким образом, третья нормальная форма требует, чтобы ни один неключевой столбец не зависел бы от другого неключевого столбца. Любой неключевой столбец должен зависеть только от столбца первичного ключа.

Нормализация таблиц

Нормальная форма Бойса-Кодда (НФБК) учитывает функциональные зависимости, в которых участвуют все потенциальные ключи отношения, а не только его первичный ключ. Для отношения с единственным потенциальным ключом 3НФ и НФБК эквивалентны. Отношение находится в НФБК тогда и только тогда, когда каждый его детерминант является потенциальным ключом. Функциональная зависимость –это такая связь между атрибутами B и A одного и того же отношения, когда каждому значению A соответствует только одно значение B. Атрибут A при этом называют детерминантом.

Нормализация таблиц

Четвертая нормальная форма (4НФ) связана с понятием **многозначной зависимости**. В случае многозначной зависимости, существующей между атрибутами A,B и C некоторого отношения, для каждого значения A имеются набор значений атрибута B и набор значений атрибута C. Однако входящие в эти наборы значения атрибутов B и C не зависят друг от друга. Отношение находится в 4НФ, если оно находится в НФБК и не содержит многозначных зависимостей.

Нормализация таблиц

Пятой нормальной формой (5НФ) называется отношение, которое не содержит зависимостей соединения. **Зависимость соединения** – это такая ситуация, при которой декомпозиция отношения может сопровождаться генерацией ложных строк при обратном соединении декомпозированных отношений с помощью операции естественного соединения.

Нормализация таблиц

Доменно-ключевая нормальная форма (ДКНФ). Если таблица находится в 3НФ или БКНФ, то остается довольно мало шансов для возникновения аномалий модификации данных, но они все равно есть. Чтобы исключить все виды возможных аномалий, таблица должна находиться в ДКНФ. Отношение находится в ДКНФ, если каждое ограничение, накладываемое на него, является логическим следствием определения доменов и ключей. Ограничение представляет собой любое правило, регулирующее возможные статические значения атрибутов, достаточно точное, чтобы можно было проверить его выполнимость. Правила редактирования, ограничения взаимосвязей и структуры отношений, функциональные и многозначные зависимости являются примерами таких ограничений.. Отсюда исключаются ограничения, связанные с изменением данных (ограничения, зависящие от времени). Таким образом, отношение находится в ДКНФ, если выполнение ограничений на домены и ключи влечет за собой выполнение всех ограничений.

В настоящее время не известен алгоритм преобразования таблицы в ДКНФ. Неизвестно также, какие таблицы могут быть приведены к ДКНФ. Поиск и создание таблиц (отношений) в ДКНФ сейчас является искусством, а не наукой.

Понятие денормализации

Чтобы исключить как можно больше аномалий модификации данных, необходимо как можно больше нормализовать таблицы базы данных. Идеальный случай, когда они доведены до ДКНФ, который на практике очень редко может быть осуществим. Чаще всего ограничиваются второй или третьей нормальными формами. Дело в том, что при нормализации обычно увеличивается количество таблиц в базе данных, и при достижении ими определенного количества, эффективность работы базы данных может оказаться слишком низкой. Кроме того, формулировать SQL запросы к базе данных тем легче, чем меньше в ней таблиц. Так что на любом этапе своего развития база данных может быть в какой-то степени денормализованной.