СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ
лекция 06, 2021/2022 учебный год

Гранков М.В.
Моб. +7 919 887 20 96 (БиЛайн)
E-mail: mv_2@mail.ru

Требования к реляционным БД (Правила Кодда)

1.Явное представление данных.
Все данные должны быть представлены явно и их значения не должны рассчитываться косвенными алгоритмами (исключение – однозначные отображения).
Пример: если, явно не указан пол сотрудника, то его нельзя (ошибочно) получать из фамилии, т.к. различные алгоритмы интерпретации фамилии в различных приложениях могут вызвать противоречия (нарушить целостность) в БД. Для явного представления данных используются типы: *числа, строки, даты, время, логический*.

Требования к реляционным БД (Правила Кодда)

2.Гарантированный доступ к данным.

Вся информация в БД должна быть доступной для приложения. Выделение любого значения в РБД выполняется при указании:

- имени отношения
- указателя на кортеж
- имени атрибута

Тройка («имя отношения», «первичный ключ», «имя атрибута») однозначно указывает на значение атрибута.

3.Полная обработка неопределенных значений.

Неопределенные значения, отличные от любого определенного значения, должны поддерживаться для всех типов данных, при выполнении всех операций. Это правило относится, прежде всего, к атрибутам.

Требования к реляционным БД (Правила Кодда)

4.Доступ к базе данных в терминах реляционной модели.

Описание БД (перечень отношений, определения схем отношений и ключей, статистическая информация и т.д.) должно быть выполнено на реляционном языке.

Пользователь должен иметь доступ к этой информации с помощью реляционного языка.

Т.е. должна быть возможность администрирования баз данных независимо от приложений.

Требования к языкам реляционным БД (правило 5)

5.Полнота подмножества реляционного языка.

Реляционная схема может поддерживать несколько языков, по крайней мере, языки DDL и DML. Однако хотя бы один из языков должен иметь синтаксис предложений, поддерживающий следующие понятия:

- определение данных (отношения, атрибуты, домены, ключи, ограничения целостности);
- определение виртуальных (мнимых) отношений образованных с помощью реляционных выражений на основе одного или нескольких отношений (определение представлений);
- манипулирование данными (интерактивное или программное);
- ограничения целостности;
- санкционированный доступ;
- управление транзакциями (начало транзакции, фиксация выполнения, отказ от выполнения).

Требования к языкам реляционным БД (правило 5)

Язык определения данных должен обеспечить выполнение первого и второго правила в самых сложных ситуациях.

Пример: если некоторое значение вычисляется на основе значений нескольких атрибутов, то оно может быть явно представлено в одном из виртуальных представлений. Виртуальное отношение формируется на основе некоторого выражения реляционного языка. Виртуальное представление может использоваться для доступа как обычное отношение БД. Корректность информации, доступной через виртуальное представление обуславливается следующим (шестым) правилом

Требования к реляционным БД (Правила Кодда)

6. Обновляемость представлений(виртуальных отношений).
Все представления (виртуальные отношения) должны автоматически обновляться при модификации данных в базовых отношениях. Например, $A = R \cup S$. A это результат вычисления выражения, т.е. точного объекта в предметной области, соответствующего A, нет. Реализация такого объекта может быть сделано с помощью механизма «виртуальных» отношений. Представление A, согласно шестого правила Кодда, должно обновляться, как только меняется R или S

7.Наличие высокоуровнего языка манипулирования данными.
Операции вставки, обновления и удаления должны применяться к отношению в целом, обеспечивая контроль над целостностью базы данных при модификации отношений. При выполнении модификации отношения в целом легко проверить ограничения: уникальности ключа, ограничения на значения и пр.

Требования к реляционным БД (Правила Кодда)

8.Физическая независимость данных.

Прикладные программы не должны зависеть от используемых способов хранения данных на носителях информации и методов доступа к ним. Физическая независимость обеспечивает работоспособность приложений при изменении расположения данных в сети.

9.Логическая независимость данных.

Прикладные программы не должны зависеть от реализации любого из используемых представлений (виртуальных отношений). Определив виртуальные отношение в рамках БД, можно разрабатывать приложения, использующие это отношение, не беспокоясь о том, что структура БД изменится и виртуальные отношение будет строиться на основе другого реляционных выражения.

Требования к реляционным БД (Правила Кодда)

10. Согласования языковых уровней.

Все ограничения целостности и внешние представления (виртуальные отношения, отчеты) должны определяться не в приложениях, а должны быть описаны с помощью языка определения данных (DDL) и сохранены в каталоге (словаре) базы данных.

11. Дистрибутивная независимость.

Реляционная система должна быть распространяема и переносима. Создание разнородных компьютерных систем требует обеспечения доступа к базам данных в различных OS и на различных платформах. Дистрибутивная независимость предполагает полную реализацию СУБД для различных платформ или реализацию коммуникационных блоков в составе СУБД, позволяющих обмениваться данными различным СУБД

Требования к реляционным БД (Правила Кодда)

12. Согласования языковых уровней.

Все ограничения целостности и внешнее представления (виртуальные отношения, отчеты) должны определяться не в приложениях, а должны быть определены с помощью языка определения данных и сохранения в каталоге (словаре) базы данных.

Отношения и реляционная алгебра

В реляционной модели База Данных рассматривается как множество отношений, а отношение – как множество кортежей. На множестве отношений заданы реляционные операции. В результате получена алгебра, т.е. система, позволяющая производить вычисления на множестве отношений. Данная система называется *реляционной алгеброй*. В реляционной алгебре операндами и результатом каждой операции являются отношения. Следовательно, полученная система является замкнутой (операции не выводят из множества отношений). Как и во всякой алгебре, мы приходим к понятиям «переменная» и «значение переменной» (в нашем случае – «*переменная отношения*» и «*значение переменной отношения*»).

Переменная отношения – это абстрактное понятие, под которым мы будем понимать произвольное отношение. Для некоторых операций – произвольное отношение с определенной схемой (заголовком отношения).

Отношения и реляционная алгебра

Для того чтобы задать значение отношения необходимо задать:

1. Схему отношения (заголовок отношения), как множество атрибутов, точнее множество пар вида (< имя_ атрибута >, <имя_ домена >).
2. Множество кортежей (тело отношения).

Каждый кортеж – упорядоченное множество. С каждым значением в кортеже связано имя атрибута, т.е. кортеж можно понимать как множество пар (< имя _ атрибута>, < значение._ атрибута>).

В нашем курсе рассматриваются отношения, обладающие следующими свойствами:

1. В отношениях нет одинаковых кортежей.
2. Кортежи не упорядочены.
3. Атрибуты именованы и, таким образом, могут рассматриваться неупорядоченными.
4. Все значения атрибутов – атомарные.

Отношения и реляционная алгебра

Пояснения к свойствам отношений:

- Отношение это множество кортежей, следовательно, кортежи не упорядочены и в отношении не существует двух одинаковых кортежей, следовательно, в отношении существует хотя бы один первичный ключ. Поиск нужного кортежа можно реализовать с помощью ключей.
- Каждому атрибуту в кортеже соответствует уникальное имя из схемы отношения. При этом каждый атрибут является атомарным, в том смысле, что его невозможно разбить на более простые части, которые соответствуют каким-либо параметрам объекта из предметной области.

Очень часто требования к свойства отношений в конкретных языках СУБД нарушаются (свойства 1-3). Требование к атомарности атрибутов часто нарушаются разработчиками БД.

Отношения и реляционная алгебра

Примеры нарушений требований к отношениям:

В некоторых СУБД допускается, чтобы кортежи повторялись, вводится упорядоченность кортежей и атрибутов в кортеже. В языках СУБД есть операции: перейти на первый кортеж; на следующий кортеж; прочитать атрибут № 5.

Проблема нарушения требования атомарности атрибутов.

Теория множеств допускает, что элемент кортежа был бы отношением. В некоторых СУБД такое определение допускается.

Пример1*:* База данных сотрудников. У объекта «Сотрудник» есть атрибут *Дети*. Значение атрибута - список: пар (*имя ребенка*, *год рождения*). Данное отношение не удовлетворяет требованию атомарности атрибутов, но достаточно точно описывает предметную область

Пример2. Атрибут ФИО можно иногда рассматривать как атомарный атрибут, иногда как множество их 3-х атрибутов: «Фамилия», «Имя», «Отчество».

Нормализация отношений. 1-я нормальная форма (1НФ)

Отношения, у которых все атрибуты атомарны, называются *нормализованными* (находящимися в 1НФ). Реляционная теория БД рассматривает только нормализованные отношения. Математическое отношение необязательно нормализовано.

Алгоритм нормализации отношения до 1НФ

Дано: $R_1 = \{A, R_2^A\}, R_2^A = \{C, D\}$ - ненормализованное отношение.

1. $\pi_A(R_1) = \{A\} = \{\{a_i\}\}, \{a_i\}$ – множество значений атрибута A отношения R_1.

2. $\pi_{R_2^A}(R_1) = \{R_2^A\} = \{\{C, D\}\}$ - множество значений атрибута R_2^A отношения R_1.

3. $\sigma_{A=a_i}(R_1)$ – кортеж из отношения R_1, в котором атрибут $A=a_i$

4. $\pi_{R_2^A}(\sigma_{A=a_i}(R_1))$ – отношение с атрибутами, $\{C, D\}$ соответствующее значению a_i атрибута A

5. $R = \pi_A(R_1) \times \pi_{R_2^A}(\sigma_{A=a_i}(R_1))$ – искомое нормализованное отношение со схемой $\{A, C, D\}$

Пример нормализация отношения до 1НФ

Пусть: R_1 – Сотрудники, R_2^A – Дети, A – Фамилия, C – Имя, D – Год (рождения).

	Фамилия	**Имя**	**Год**
1	Иванов	Коля Саша	2008 2010
2	Петров	Вова Маша	2010 2011
3	Сидоров	Наташа	2012
4	Николаев	Null	Null

$\pi_A(R_1)$ ={Иванов, Петров, Сидоров, Николаев}.

$\pi_{R_2^A}(R_1)$ = {{(Коля, 2008), (Саша, 2010)}, {(Вова, 2010), (Маша, 2011)}, {(Наташа, 2012)}, {(Null,Null)}}.

$\sigma_{A=a_2}(R_1)$ ={(Петров, {(Вова, 2010), (Маша, 2011)})}.

$\pi_{R_2^A}(\sigma_{A=a_2}(R_1))$ ={(Вова, 2010), (Маша, 2011)}.

$\pi_A(\sigma_{A=a_2}(R_1)) \times \pi_{R_2^A}(\sigma_{A=a_2}(R_1))$ = {(Петров, Вова, 2010), (Петров, Маша, 2011)}

$\pi_A(R_1) \times \pi_{R_2^A}(\sigma_{A=a_i}(R_1))$ = {{(Иванов, Коля, 2008), (Иванов, Саша, 2010)}, {(Петров, Вова, 2010), (Петров, Маша, 2011)}, {(Сидоров, Наташа, 2012)}, {(Николаев, Null,Null)}}

Виды отношений, используемых в реляционных системах

- *Именованное отношение* – это переменная типа отношение, у которой есть имя.
- *Базовое отношение* – это именованное отношение, которое не является производным от других отношений.
- *Производное отношение* – это отношение, определённое через другие именованные отношения (посредством реляционного выражения) и, в конечном итоге, через базовые отношения.
- *Выражаемое отношение* – это отношение, которое можно получить из набора именованных отношений посредством некоторого реляционного выражения. Каждое именованное отношение является выражаемым отношением, но необязательно, что выражаемое отношение имеет имя.
- *Множество всех выражаемых отношений* – это объединение множества всех базовых отношений и множества всех производных отношений.

Виды отношений, используемых в реляционных системах

- *Представление* - это именованное производное отношение. Представление, как и базовое отношение, является переменной отношения.
- *Снимки* – это именованные производные отношения (являются переменными отношения). Создание снимка похоже на выполнение запроса, за исключением того, что результат такого запроса сохраняется в базе данных под некоторым именем как отношение, доступное только для чтения. Периодически (например: раз в сутки) снимок обновляется.
- *Результаты запроса* – неименованное производное отношение, является результатом вычисления некоторого реляционного выражения. База данных не обеспечивает постоянное хранение результатов запроса. Для этого результат запроса необходимо присвоить некоторому именованному отношению.

Виды отношений, используемых в реляционных системах

- *Промежуточным результатом* называется неименованное производное отношение, являющееся результатом некоторого реляционного выражения.
- *Хранимое отношение* – это отношение, которое поддерживается во внешней памяти.

Исчисление предикатов в реляционных системах

Результатом реляционного выражения является отношение, т.е. множество. Как известно, множество может быть задано с помощью предиката: $A = \{a \mid P(a) \equiv 1\}$, где:

$P(a)$ - неопределенное высказывание (предикат).

Можно показать, что реляционному выражению можно поставить в соответствие некоторый предикат, который определит некоторое отношение, совпадающее с результатом вычисления данного реляционного выражения.

Понятие реляционного исчисления, т.е. специального применения исчисления предикатов в реляционных базах данных, было впервые предложено Коддом. Исчисление предикатов необходимо производить на множестве некоторых переменных. Первоначально реляционное исчисление было основано на переменных кортежа.

Исчисление предикатов в реляционных системах

- *Переменная кортежа* – это переменная, которая изменяется на некотором отношении, т.е. это переменная, допустимые значения которой – это кортежи данного отношения. Если переменная кортежа T изменяется в пределах отношения R, то в любой данное время значения переменой T это некоторый кортеж t из отношения R.
- Реляционное исчисление, основанное на переменных кортежа, называется *исчислением кортежей*.
- Альтернативное исчисление, определённое на переменных доменов (т.е. переменных, которые изменяются не на отношениях, а на доменах) называется *исчислением доменов*.
- Наиболее известен язык исчисления доменов это Query – By – Example (QBE) - язык, реляционной СУБД PARADOX фирмы BORLAND.
- Реляционная алгебра, исчисление кортежей и исчисление доменов являются эквивалентными вариантами языков реляционных СУБД.

Целостность БД и используемые ключи

- **Целостность БД** – свойство БД, при наличии которого БД содержит полную и непротиворечивую информацию, необходимую и достаточную для корректного функционирования приложений.
- Некоторые условия целостности проверяются с помощью типов (например, даты). В сложных случаях целостность проверяется вычислением логических выражений. Разработка таких выражений существенно опирается на предметную область.
- Большинство правил целостности являются специфическими для данной БД.
- Имеется два правила целостности, применимые к любой базе данных. Эти два правила относятся к так называемым потенциальным и внешним ключам.

Потенциальные ключи

- *Потенциальный ключ* – это обобщение понятия первичного ключа. Первичный ключ уникальным образом идентифицирует кортеж в отношении.
- *Потенциальные ключи* обладают свойством уникальности; в отношении их может быть несколько; первичный ключ только один и выбирается из потенциальных ключей.

Определение потенциального ключа:

- Пусть R – некоторая переменная отношения, тогда потенциальный ключ K для R это подмножество атрибутов R, всегда обладающее следующими свойствами:

1. Свойство уникальности: нет двух различных кортежей в текущем значении переменной R с одинаковыми значениями K.
2. Свойство не избыточности: никакое из подмножеств K не обладает свойством уникальности.

Потенциальные ключи

- На практике в отношениях чаще всего встречается один потенциальный ключ, который выбирается первичным.

Пример исключения: дана БД элементов таблицы Менделеева: порядковый номер элемента, атомная масса, название элемента и т.д. – все это потенциальные ключи.

Из определения следует, что *потенциальный ключ* – множество атрибутов.

- *Потенциальный ключ* из 1-го атрибута – это *простой ключ*.
- *Потенциальный ключ* из нескольких атрибутов – это *составной ключ*.
- Если *потенциальных ключей* несколько, то один должен быть выбран в качестве первичного ключа, а остальные будем называть *альтернативными ключами*.

Внешние ключи

- Пусть R2 – базовое отношение, тогда *внешний ключ*, например, FK в отношении R2 – это подмножество множества атрибутов R2, такое что:

1. Существует базовое отношение R1 (R1 и R2 не обязательно различны) с потенциальным ключом СК;
2. Каждое значение FK в текущем значении R2 всегда совпадает со значением СК некоторого кортежа в текущем значении R1.

Пример: Даны два отношения: AUTO={COLORE,A,…} (R2) – сведения об автомобилях некоторой организации и отношение MENU={C_COLORE,NAME_COLORE} справочник возможных цветов автомобилей (R1).

Замечания:

1. внешний ключ – это множество (вводят через {}), но если внешний ключ простой, то скобки могут опускаться;
2. каждому значению внешнего ключа соответствует значение потенциального ключа (обратное не справедливо).

Требования к ключам в реляционных БД

- Если внешний ключ составной, то соответствующий ему потенциальный ключ тоже составной. Если внешний ключ простой, то потенциальный ключ тоже простой.
- Каждый атрибут, входящий в данный внешний ключ, должен быть определен на том же домене, что и соответствующий атрибут, соответствующего потенциального ключа.
- Для внешнего ключа не требуется, чтобы он был компонентой первичного ключа данного отношения.
- Значение внешнего ключа является ссылкой на кортеж, содержащий соответствующее значение потенциального ключа (так называемый ссылочный или целевой кортеж). Проблема обеспечения того, что база данных не должна включать никаких неверных значений внешних ключей называемой проблемой *ссылочной целостности*.

Требования к ключам в реляционных БД

- Ограничение, по которому каждое значение внешнего ключа должно соответствовать значению потенциального ключа, называется *ссылочным ограничением.*
- Отношение, которое содержит внешнюю ссылку, называется *ссылающимся отношением*, а отношение, которое содержит потенциальный ключ – *ссылочным или целевым отношением*.
- Ссылочные ограничения представляются ссылочными диаграммами. Иногда над стрелкой пишут внешний ключ:

$$AUTO \xrightarrow{COLORE} MENU$$

- Отношение может быть одновременно и ссылочным и ссылающимся ($R3 \rightarrow R2 \rightarrow R1$).
- Отношение может ссылаться само на себя ($R1 \rightarrow R1$).

Пример: Список сотрудников, у которых есть атрибут – код сотрудника, которому данный сотрудник подчиняется.

- Ссылки могут образовывать цикл (ссылочный цикл):

$Rn \rightarrow Rn\text{-}1 \ldots R1 \rightarrow Rn$.

Ссылочная целостность

- База данных обладает ссылочной целостностью, если она не содержит значений внешних ключей, которым не соответствует значение первичного ключа. Удаление в ссылающемся отношении никогда не нарушает ссылочную целостность.
- Проверить ссылочную целостность в момент добавления или коррекции ссылающегося отношения не является сложным. Вся сложность лишь в предоставлении возможности добавить кортеж в ссылочное отношение из среды работы с ссылающимся отношением.
- Добавление в ссылочное отношение никогда не нарушает ссылочную целостность.
- Проблемы проверки ссылочной целостности возникают лишь при удалении и корректировки ссылочного отношения. Если при выполнении этих операций нарушается ссылочная целостность, то используют две основные стратегии:

Стратегии поддержки ссылочной целостность

1. операции, нарушающие целостность, отвергнуть;
2. изменить ссылающееся отношение, чтобы не нарушить целостность. В ссылочных цепочках, возможно, потребуется изменить несколько отношений (так называемое каскадирование!).

Пример: удалить или изменить кортежи ссылочного отношения (MENU).

- Второй вариант реализации возможен при организации диалога с пользователем (предложить изменить ссылающиеся отношения, удаляемую запись (кортеж) перенести в архив и т. д.).
- Особая сложность возникает при наличии ссылочных циклов. Возможно, проверку ссылочной целостности следует отложить до момента фиксации транзакции.

Правило целостности объекта

- Вместе с понятием первичного ключа реляционная модель включает правило целостности объекта, которое заключается в следующем:

Ни один элемент первичного ключа базового отношения не может иметь Null – значение.

Предпосылки введения такого правила следующие:

1. кортежи базового отношения соответствуют объектам реального мира;
2. объекты реального мира различны (т.е. некоторым образом опознаваемы);
3. значит, должны быть различимы представления реального мира, т.е. кортежи базового отношения.

- Если первичный ключ или его часть имеет Null – значение, то идентичность кортежа объекту теряется.

В реляционной БД никогда не следует записываем информацию о чём – то, чего не возможно идентифицировать.

Проблема использования Null – значений

- Проблема использования Null – значений до конца ещё не решена. Введение и поддержка этого правила имеет несколько противоречий.

Пример: допустим, что в базе AUTO мы ввели понятие Null – значение для атрибута COLORE. Составим список цветов автомобилей. В этом списке (отношении) возможно, будет значение – цвет не определён. Наш запрос не является базовым отношением. И для него требования целостности объекта может не применяться. Но если результат запроса сохранить как базовое отношение, то это отношение буде состоять только из одного атрибута – цвет, которое должно быть первичным ключом. Одно из значений ключа – Null.

- Альтернативной стратегией Null – значениям является использование значений по – умолчанию.

Внешние ключи и Null – значения

- Внешние ключи могут принимать Null – значения.

Пример: если рассмотреть самоссылающееся отношение <сотрудники>, в котором есть атрибут <код_начальника>, то у президента компании это атрибут содержит Null – значение. Т.о. понятие внешнего ключа должно быть дополнено возможностью принимать значение Null.

- Определение внешнего ключа должно быть расширено:

Пусть R2 – базовое отношение. Тогда внешний ключ, например, FK в отношении R2 – это подмножество множества атрибутов R2, такое что:

1. Существует базовое отношение R1 (R1 и R2 не обязательно различны) с потенциальным ключом СК.
2. Всегда каждое значение FK в текущем значении R2 или является Null – значением, или совпадает со значением СК некоторого кортежа в текущем значении R1.

Внешние ключи и Null – значения

- Возможность включения или не включения Null – значений в первичные и внешние ключи регулируются соответствующими опциями операторов языка определения данных БД.
- Замечание: Допустимость принятия Null – значения для внешнего ключа в некоторой степени решает проблему ссылочной целостности.
- **Пример:** при удалении некоторого цвета из справочника цветов достаточно в записях об автомобилях (отношение AUTO), у которых атрибут <Цвет> совпадал с удаляемым из отношения <MENU> цветом, установить цвет в Null.