Подготовка к магистратуре МГТУ ИУ7 by Winterpuma

Направление: 09.04.04

Дисциплина 1.

Базы данных

Оглавление

1. Реляционная модель. Реляционные объекты данных. Целостность реляционных данных.

Реляционная модель - совокупность данных, состоящая из набора двумерных таблиц.

Согласно Дейту реляционная модель состоит из трех частей, описывающих разные аспекты реляционного подхода:

- **структурная часть** - отвечает за принцип построения структуры реляционной БД на нормализованном наборе n-арных отношений, в форме таблиц. Реляционная БД структурно может представляться только в виде отношений.
- **манипуляционная часть** - утверждаются операторы манипулирования отношениями:
    - реляционная алгебра (база - теория множеств),
    - реляционное исчисление (база - логический аппарат исчисления предикатов первого порядка).
- **целостная часть** - фиксируются два базовые требования целостности:
    - требование целостности сущностей (первичного ключа) - любой кортеж любого отношения отличим от любого другого кортежа этого отношения (т.е. любое отношение должно обладать первичным ключом).
    - требование целостности по ссылкам (внешнего ключа) - для каждого значения внешнего ключа, появляющегося в ссылающемся отношении, в отношении на которое ведет ссылка, должен найтись кортеж с таким же значением первичного ключа, либо значение внешнего ключа должно быть неопределенным.

Реляционные объекты данных:

- **отношение** - двумерная таблица, состоящая из столбцов и строк
- **атрибут** - поименованный столбец отношения

- **домен** - набор допустимых значений для одного или нескольких атрибутов
- **кортеж** - строка отношения
- **степень отношения** - количество его атрибутов
- **кардинальность** - количество кортежей в отношении
- **первичный ключ** - атрибут (или множество атрибутов), значения которого уникально идентифицируют кортежи.

2. Реляционная алгебра.

Реляционная алгебра — замкнутая система операций над отношениями в реляционной модели данных.

Основные операции реляционной алгебры (все операции перечислить невозможно, поскольку любая операция, удовлетворяющая определению реляционной, является частью реляционной алгебры):

- **Переименование** (R RENAME Atr1, Atr2, … AS NewAtr1, NewAtr2, …)
  В результате применения операции переименования получаем новое отношение, с измененными именами атрибутов.
- **Объединение** (A UNION B)
  Отношение с тем же заголовком, что и у совместимых по типу отношений A и B, и телом, состоящим из кортежей, принадлежащих или A, или B, или обоим отношениям.
- **Пересечение** (A INTERSECT B)
  Отношение с тем же заголовком, что и у отношений A и B, и телом, состоящим из кортежей, принадлежащих одновременно обоим отношениям A и B.
- **Вычитание** (A MINUS B)
  Отношение с тем же заголовком, что и у совместимых по типу отношений A и B, и телом, состоящим из кортежей, принадлежащих отношению A и не принадлежащих отношению B.

- **Декартово произведение** (A TIMES B)

  Отношение (A1, A2, …, Am, B1, B2, …, Bm), заголовок которого является сцеплением заголовков отношений A(A1, A2, …, Am) и B(B1, B2, …, Bm), а тело состоит из кортежей, являющихся сцеплением кортежей отношений A и B: (a1, a2, …, am, b1, b2, …, bm) таких, что (a1, a2, …, am)∈ A, (b1, b2, …, bm)∈ B.

- **Выборка** (ограничение) (A WHERE d)

  Отношение с тем же заголовком, что и у отношения A, и телом, состоящим из кортежей, значения атрибутов которых при подстановке в условие c дают значение ИСТИНА. d представляет собой логическое выражение, в которое могут входить атрибуты отношения A и/или скалярные выражения.

- **Проекция** (PROJECT A {x, y, …, z})

  Отношение с заголовком (X, Y, …, Z) и телом, содержащим множество кортежей вида (x, y, …, z), таких, для которых в отношении A найдутся кортежи со значением атрибута X равным x, значением атрибута Y равным y, …, значением атрибута Z равным z. При выполнении проекции выделяется «вертикальная» вырезка отношения-операнда с естественным уничтожением потенциально возникающих кортежей-дубликатов.

- **Соединение** (A JOIN c)

  Операция соединения есть результат последовательного применения операций декартового произведения и выборки. Если в отношениях имеются атрибуты с одинаковыми наименованиями, то перед выполнением соединения такие атрибуты необходимо переименовать.

- **Деление** (A DIVIDEBY B)

  Отношение с заголовком (X1, X2, …, Xn) и телом, содержащим множество кортежей (x1, x2, …, xn), таких, что для всех кортежей (y1, y2, …, ym) ∈ B в отношении A(X1, X2, …, Xn, Y1, Y2, …, Ym) найдется кортеж (x1, x2, …, xn, y1, y2, …, ym).

3. Реляционное исчисление (вариант кортежей).

Формула реляционного исчисления только устанавливает условия, которым должны удовлетворять кортежи результирующего отношения. Поэтому языки реляционного исчисления являются более непроцедурными или декларативными.

RANGE - оператор, который используется для определения кортежной переменной.

RANGE СОТРУДНИК IS СОТРУДНИКИ - определение переменной СОТРУДНИК, областью определения которой является отношение СОТРУДНИКИ.

Атрибуты переменной - для того, чтобы сослаться на значение атрибута СОТР_ИМЯ переменной СОТРУДНИК, нужно употребить конструкцию СОТРУДНИК.СОТР_ИМЯ.

Правильно построенные формулы (WFF - Well-Formed Formula) служат для выражения условий, накладываемых на кортежные переменные.

- **простые сравнения** - операции сравнения скалярных значений. Например "СОТРУДНИК.СОТР_НОМ = 140".
- **логические связки** NOT, AND, OR и IF ... THEN.
- **кванторы** EXISTS var (form) и FORALL var (form)

Переменные, входящие в WFF, могут быть **свободными или связанными**.

- Все переменные, входящие в WFF, при построении которой не использовались кванторы, являются **свободными**. Фактически, это означает, что если для какого-то набора значений свободных кортежных переменных при вычислении WFF получено значение true, то эти значения кортежных переменных могут входить в результирующее отношение.
- Если же имя переменной использовано сразу после квантора при построении WFF вида EXISTS var (form) или FORALL var (form), то в

этой WFF и во всех WFF, построенных с ее участием, var - это **связанная** переменная. Это означает, что такая переменная не видна за пределами минимальной WFF, связавшей эту переменную. При вычислении значения такой WFF используется не одно значение связанной переменной, а вся ее область определения.

Целевой список - компонент, который определяет набор и имена столбцов результирующего отношения.

- var.attr, где var - имя свободной переменной соответствующей WFF, а attr - имя атрибута отношения, на котором определена переменная var;
- var, что эквивалентно наличию подсписка var.attr1, var.attr2, ..., var.attrn, где attr1, attr2, ..., attrn включает имена всех атрибутов определяющего отношения;
- new_name = var.attr; new_name - новое имя соответствующего атрибута результирующего отношения.

4. Условные выражения и предикаты языка SQL.

Условные выражения.

- **IF-THEN-ELSIF-ELSE -** оператор используется для выполнения кода при условие TRUE (истинно), или выполнения другого кода, если условие принимает значение FALSE (ложь). Выражение ELSE является необязательным, и его можно опускать.

```
IF УСЛОВИЕ1 THEN
        {... выполняется, когда УСЛОВИЕ1 истинно (TRUE)...}
ELSIF УСЛОВИЕ2 THEN
        {... выполняется, когда УСЛОВИЕ2 истинно (TRUE)...}
ELSE
        {... выполняется, когда оба: УСЛОВИЕ1 и УСЛОВИЕ2 ложно (FALSE)...}
END IF
```

- **GOTO -** вызывает код для перехода к метке после оператора GOTO.

```
GOTO label_name;
```

- **CASE -** имеет функциональность IF-THEN-ELSE

```
CASE [ expression ]
  WHEN condition_1 THEN result_1
  ...
  WHEN condition_n THEN result_n
  ELSE result
END
```

Предикаты языка SQL

Предикат в языке SQL может принимать одно из трех значений **TRUE** (истина), **FALSE** (ложь) или **UNKNOWN** (неизвестно).

Исключение составляют следующие предикаты: **NULL** (отсутствие значения), **EXISTS** (существование), **UNIQUE** (уникальность) и **MATCH** (совпадение), которые не могут принимать значение UNKNOWN.

- **Предикаты сравнения** - представляет собой два выражения, соединяемых оператором сравнения. Имеется шесть традиционных операторов сравнения: =, >, <, >=, <=, <>.
  - Данные типа NUMERIC (числа) сравниваются в соответствии с их алгебраическим значением.
  - Данные типа CHARACTER STRING (символьные строки) сравниваются в соответствии с их алфавитной последовательностью.
- **Предикат BETWEEN** - проверяет, попадают ли значения проверяемого выражения в диапазон, задаваемый пограничными выражениями, соединяемыми служебным словом AND.

  <Проверяемое выражение> [NOT] BETWEEN <Начальное выражение> AND <Конечное выражение>
- **Предикат IN** - определяет, будет ли значение проверяемого выражения обнаружено в наборе значений, который либо явно определен, либо получен с помощью табличного подзапроса.

5. Функциональные зависимости. Правила вывода Армстронга и Дарвена. Замыкание множества атрибутов.

Функциональная зависимость - связь типа многие к одному между двумя множествами атрибутов заданной переменной-отношения. Для заданной переменной-отношения R зависимость A -> B (где A и B являются подмножествами множества атрибутов переменной-отношения R) выполняется для переменной-отношения R тогда и только тогда, когда любые два кортежа переменной-отношения R с одинаковыми значениями атрибутов множества A имеют одинаковые значения атрибутов множества B.

Армстронг предложил набор **правил вывода** новых функциональных зависимостей на основе данных: (знак "→" это функциональная зависимость)

- 1: **правило рефлексивности**. Если $B \subseteq A \Rightarrow A \rightarrow B$.
- 2: **дополнение**. $A \rightarrow B \Rightarrow AC \rightarrow BC$, где C – новый атрибут в пределах данной схемы. AC, BC – объединение множеств.
- 3: **транзитивность**. $A \rightarrow B, B \rightarrow C \Rightarrow A \rightarrow C$.

Набор правил Армстронга является **полным** – для заданного множества ФЗ минимальный набор может быть выведен только с помощью этих трех правил; является **исчерпывающим** – в результате применений этих ФЗ никакие дополнительные ФЗ не могут быть получены.

Дополнительные правила, упрощающие задачу вывода функциональных зависимостей:

- 4: **самоопределение**. $A \rightarrow A$
- 5: **декомпозиция**. $A \rightarrow BC \Rightarrow A \rightarrow B, A \rightarrow C$
- 6: **объединение**. $A \rightarrow B, A \rightarrow C \Rightarrow A \rightarrow BC$
- 7: **композиция**. $A \rightarrow B, C \rightarrow D \Rightarrow AC \rightarrow BD$
- 8: **правило унификации** (общая теорема определения Дарвена).
  $A \rightarrow B, C \rightarrow D \Rightarrow A \cup (C-B) \rightarrow BD$

Применяя правила из предыдущего раздела до тех пор, пока создание новых функциональных зависимостей не прекратится само собой, мы получим **замыкание** для заданного множества функциональных зависимостей.

Замыкание множества атрибутов X над множеством ФЗ S — максимальное по включению множество атрибутов, обозначаемое X_S^+, функционально зависящих от S.

Множество ФЗ S **неприводимо**, если:

- Каждая правая часть ФЗ содержит ровно один атрибут
- Каждая левая часть ФЗ минимальна по включению
- S минимально по включению

Множество ФЗ S **минимально по включению**, если ни одна функциональная зависимость из множества S не может быть удалена из множества S без изменения его замыкания S+.

6. Нормализация отношений. Первая, вторая и третья нормальные формы. Нормальная форма Бойса-Кодда.

Процесс **нормализации** заключается в разложении (декомпозиции) исходных отношений БД на более простые отношения. Каждая ступень этого процесса приводит схему отношений в последовательные нормальные формы. Для каждой ступени нормализации имеются наборы ограничений, которым должны удовлетворять отношения БД.

1NF - первая нормальная форма.

Простой атрибут - атрибут, значения которого атомарны (неделимы).

Сложный атрибут - получается соединением нескольких атомарных атрибутов, которые могут быть определены на одном или разных доменах. (его также называют вектор или агрегат данных).

Отношение находится в **1NF** если значения всех его атрибутов атомарны.

Алгоритм нормализации описан следующим образом:

1. Начиная с отношения, находящегося на верху дерева, берется его первичный ключ, и каждое непосредственно подчиненное отношение расширяется путем вставки домена или комбинации доменов этого первичного ключа.
2. Первичный Ключ каждого расширенного таким образом отношения состоит из Первичного Ключа, который был у этого отношения до расширения и добавленного Первичного Ключа родительского отношения.
3. После этого из родительского отношения вычеркиваются все непростые домены, удаляется верхний узел дерева, и эта же процедура повторяется для каждого из оставшихся поддеревьев.

2NF - вторая нормальная форма.

Отношение находится во **2НФ**, если оно находится в 1НФ и каждый неключевой атрибут функционально полно зависит от ключа.

Неключевой атрибут **функционально полно зависит** от составного ключа если он функционально зависит от всего ключа в целом, но не находится в функциональной зависимости от какого-либо из входящих в него атрибутов.

Пример приведения в 2НФ:

> ПОСТАВКИ (N_ПОСТАВЩИКА, ТОВАР, ЦЕНА).
>
> * Пусть все поставщики поставляют товар по одной и той же цене, тогда:
>
> (функциональные зависимости: N_поставщика, товар -> цена; товар -> цена)

Аномалия: при изменении цены товара необходим полный просмотр отношения для того, чтобы изменить все записи о его поставщиках (т.к. в одной структуре данных объединены два семантических факта).

Разложение во 2НФ:

> ПОСТАВКИ (N_ПОСТАВЩИКА, ТОВАР)
>
> ЦЕНА_ТОВАРА (ТОВАР, ЦЕНА)

3NF - третья нормальная форма.

Пусть X, Y, Z - три атрибута некоторого отношения. При этом X --> Y и Y --> Z, но обратное соответствие отсутствует, т.е. Z -/-> Y и Y -/-> X. Тогда Z **транзитивно зависит** от X.

Отношение находится в **3НФ**, если оно находится во 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Пример приведения в 3НФ:

> ХРАНЕНИЕ (ФИРМА, СКЛАД, ОБЪЕМ)
> (функциональные зависимости: фирма -> склад; склад -> объем)

Аномалии:

- если в данный момент ни одна фирма не получает товар со склада, то в базу данных нельзя ввести данные о его объеме (т.к. не определен ключевой атрибут)
- если объем склада изменяется, необходим просмотр всего отношения и изменение кортежей для всех фирм, связанных с данным складом.

Для этого заменяем на два отношения (3НФ):

> ХРАНЕНИЕ (ФИРМА, СКЛАД)
>
> ОБЪЕМ_СКЛАДА (СКЛАД, ОБЪЕМ)

BCNF - нормальная форма Бойса-Кодда.

Отношение находится в **BCNF**, если оно находится во 3НФ и в ней отсутствуют зависимости атрибутов первичного ключа от неключевых атрибутов.

Ситуация, когда отношение будет находится в 3NF, но не в BCNF, возникает при условии, что отношение имеет два (или более) возможных ключа, которые являются составными и имеют общий атрибут. На практике такая ситуация встречается достаточно редко, для всех прочих отношений 3NF и BCNF эквивалентны.

7. Нормализация отношений. Многозначные зависимости и четвертая нормальная форма. Зависимости соединения и пятая нормальная форма. Процесс **нормализации** заключается в разложении (декомпозиции) исходных отношений БД на более простые отношения. Каждая ступень этого процесса приводит схему отношений в последовательные нормальные формы. Для каждой ступени нормализации имеются наборы ограничений, которым должны удовлетворять отношения БД.

Четвертая нормальная форма касается отношений, в которых имеются повторяющиеся наборы данных. Декомпозиция, основанная на функциональных зависимостях, не приводит к исключению такой избыточности. В этом случае используют декомпозицию, основанную на многозначных зависимостях.

Многозначная зависимость является обобщением функциональной зависимости и рассматривает соответствия между множествами значений атрибутов.

Отношение находится в **4NF** если оно находится в BCNF и в нем отсутствуют многозначные зависимости, не являющиеся функциональными зависимостями.

Пример: отношение ПРЕПОДАВАТЕЛЬ (ИМЯ, КУРС, УЧЕБНОЕ_ПОСОБИЕ)

ИМЯ	КУРС	УЧЕБНОЕ_ПОСОБИЕ
N	Теория упругости	Теория упругости
N	Теория колебаний	Теория упругости
N	Теория упругости	Теория колебаний
N	Теория колебаний	Теория колебаний
K	Теория удара	Теория удара
K	Теория удара	Теоретическая механика

добавляем:

K	Теория упругости	Теория удара
K	Теория упругости	Теоретическая механика

Это отношение имеет значительную избыточность и его использование приводит к возникновению **аномалии обновления**.

Аномалия: добавление информации о том, что профессор К будет также читать лекции по курсу "Теория упругости" приводит к необходимости добавить два кортежа вместо одного.

Аномалия обновления возникает в данном случае потому, что в отношении ПРЕПОДАВАТЕЛЬ имеются многозначные зависимости:

1. зависимость *множества* значений атрибута КУРС от *множества* значений атрибута ИМЯ
2. зависимость *множества значений атрибута УЧЕБНОЕ_ПОСОБИЕ* от *множества значений атрибута ИМЯ.*

Указанные аномалии исчезают при замене отношения ПРЕПОДАВАТЕЛЬ его проекциями:

ИМЯ	КУРС
N	Теория упругости
N	Теория колебаний
K	Теория удара
K	Теория упругости

ИМЯ	УЧЕБНОЕ_ПОСОБИЕ
N	Теория упругости
N	Теория колебаний
K	Теоретическая механика
K	Теория удара

Зависимости по соединению и пятая нормальная форма (5NF).

Существуют отношения, для которых нельзя выполнить декомпозицию без потерь на две проекции, но которые можно подвергнуть декомпозиции без потерь на три (или более) проекций. Этот факт получил название **зависимости по соединению**, а такие отношения называют 3-декомпозируемые отношения (ясно, что любое отношение можно назвать "n-декомпозируемым", где n >= 2).

Зависимость по соединению является обобщением многозначной зависимости. Отношения, в которых имеются зависимости по соединению, не являющиеся одновременно ни многозначными, ни функциональными, также характеризуются аномалиями обновления. Поэтому, вводится понятие пятой нормальной формы.

Отношение находится в **5НФ** тогда и только тогда, когда любая зависимость по соединению в нем определяется только его возможными ключами.

Другими словами, каждая проекция такого отношения содержит не менее одного возможного ключа и не менее одного неключевого атрибута.

8. Управление транзакциями. Типы транзакций. Свойства транзакций.

Транзакция является логической единицей обработки в СУБД.

- Если транзакция выполнена успешно, все модификации данных, сделанные в течение транзакции, принимаются и становятся постоянной частью базы данных.
- Если в результате выполнения транзакции происходят ошибки и должна быть произведена отмена или выполнен откат, все модификации данных будут отменены.

Управление транзакциями в приложениях реализуется, главным образом, путем указания того, когда транзакция начинается и заканчивается. В системе должна быть возможность правильной обработки ошибок, прерывающих транзакцию до ее окончания.

Типы транзакций:

- **Явные транзакции**. Каждая транзакция явно начинается с инструкции BEGIN TRANSACTION и явно заканчивается инструкцией COMMIT или ROLLBACK.
- **Неявные транзакции**. Новая транзакция неявно начинается, когда предыдущая транзакция завершена, но каждая транзакция явно завершается инструкцией COMMIT или ROLLBACK.

Инструкция COMMIT: если транзакция выполнена успешно, ее следует зафиксировать. COMMIT гарантирует, что все изменения в пределах данной транзакции стали постоянной частью базы данных.

Инструкция ROLLBACK: если в транзакции произойдет ошибка или пользователь решит отменить транзакцию, следует выполнить ее откат. ROLLBACK отменяет все изменения, сделанные в пределах транзакции, возвращая данные в то состояние, в котором они находились на начало транзакции. Инструкция ROLLBACK также освобождает удерживаемые транзакцией ресурсы.

Свойства ACID используются для поддержания целостности базы данных во время обработки транзакций:

- **Атомарность**: транзакция — это единица операции. Вы либо выполняете его полностью, либо не выполняете вообще. Не может быть частичного исполнения.
- **Согласованность**: после выполнения транзакции она должна перейти из одного согласованного состояния в другое.
- **Изоляция**: транзакция должна выполняться изолированно от других транзакций (без блокировок). Во время одновременного выполнения транзакции промежуточные результаты транзакций, выполняемые одновременно выполняемыми транзакциями, не должны быть доступны друг другу. (Уровень 0,1,2,3)
- **Долговечность**: · После успешного завершения транзакции изменения в базе данных должны сохраняться. Даже в случае системных сбоев.

9. Управление транзакциями. Тупиковые ситуации и способы их обнаружения. Уровни изоляции. Поддержка блокировок в стандарте языка SQL.

Транзакция является логической единицей обработки в СУБД.

- Если транзакция выполнена успешно, все модификации данных, сделанные в течение транзакции, принимаются и становятся постоянной частью базы данных.
- Если в результате выполнения транзакции происходят ошибки и должна быть произведена отмена или выполнен откат, все модификации данных будут отменены.

Управление транзакциями в приложениях реализуется, главным образом, путем указания того, когда транзакция начинается и заканчивается. В системе должна быть возможность правильной обработки ошибок, прерывающих транзакцию до ее окончания.

Тупиковые ситуации и способы их обнаружения.

Тупик — это состояние системы базы данных, имеющей две или более транзакций, когда каждая транзакция ожидает элемент данных, который блокируется какой-либо другой транзакцией.

О тупике можно указать циклом в **графике ожидания**. Это ориентированный граф, в котором вершины обозначают транзакции, а ребра — ожидания элементов данных.

Алгоритмы обнаружения тупиков могут использовать **таймеры**. Каждая транзакция связана с таймером, который установлен на период времени, в течение которого ожидается завершение транзакции. Если транзакция не завершается в течение этого периода времени, таймер отключается, указывая на возможную тупиковую ситуацию.

Уровни изолированности:

- **Read uncommitted** - самая плохая согласованность, самая высокая скорость выполнения транзакций. Каждая транзакция видит незафиксированные изменения другой транзакции.
- **Read committed** - параллельно исполняющиеся транзакции видят только зафиксированные изменения из других транзакций. Таким образом, данный уровень обеспечивает защиту от грязного чтения.
- **Repeatable read** - удаленные и измененные записи другой транзакцией не видны.
- **Serializable** - уровень, при котором транзакции ведут себя как будто ничего более не существует, никакого влияния друг на друга нет.

Принудительное упорядочение транзакций обеспечивается с помощью **механизма блокировок**. Суть этого механизма в следующем: если для выполнения некоторой транзакции необходимо, чтобы некоторый объект базы данных (кортеж, набор кортежей, отношение, набор отношений,..) не

изменялся непредсказуемо и без ведома этой транзакции, такой объект блокируется.

Основными **видами блокировок** являются:

- **блокировка со взаимным доступом** / S-блокировка / блокировкой по чтению.
- **монопольная блокировка** (без взаимного доступа) / X-блокировка / блокировка по записи. Этот режим используется при операциях изменения, добавления и удаления объектов.

10. Безопасность данных в базах данных. Поддержка мер обеспечения безопасности в стандарте языка SQL: механизм представлений и подсистема полномочий.

В действующем стандарте языка SQL предусматривается поддержка только избирательного управления доступом.

- **Механизм представлений** - может быть использован для скрытия очень важных данных от несанкционированных пользователей.
- **Подсистема полномочий** - наделяет одних пользователей правом избирательно и динамично задавать различные полномочия другим пользователям, а также отбирать такие полномочия в случае необходимости.

Предоставление привилегий задается с помощью директивы **GRANT**.

```
GRANT { ALL [ PRIVILEGES ] | список_разрешений }
ON список_объектов
TO список_принципалов
[ WITH GRANT OPTION ] [ AS принципал ]
```

Принципалом может быть:

- пользователь базы данных,
- роль базы данных,
- роль приложения.

Если задана директива WITH GRANT OPTION, это значит, что указанные пользо-ватели наделены особыми **полномочиями** для заданного объекта — правом предоставления полномочий. Это, в свою очередь, означает, что для работы с данным объектом они могут наделять полномочиями других пользователей.

Если пользователь A наделяет некоторыми полномочиями другого пользователя B, то впоследствии он может отменить эти полномочия для пользователя B.

Отмена полномочий выполняется с помощью директивы **REVOKE**.

```
REVOKE [ GRANT OPTION FOR ] список_разрешений ON список_объектов
{ FROM | TO } список_принципалов
[ CASCADE ]
[ AS принципал ]
```

Инструкция **DENY** запрещает разрешения на члены класса OBJECT защищаемых объектов.

```
DENY список_разрешений
ON список_объектов
TO список_принципалов [ CASCADE ]
[ AS принципал ]
```

11. Декларативная и процедурная поддержка ограничений целостности. Поддержка ограничений целостности в стандарте языка SQL.

Каждая система обладает своими средствами поддержки ограничений целостности. Различают два способа реализации:

- Декларативная поддержка ограничений целостности.
- Процедурная поддержка ограничений целостности.

Декларативная поддержка ограничений целостности заключается в определении ограничений средствами языка определения данных (DDL - Data Definition Language). Обычно средства декларативной поддержки целостности определяют ограничения на значения доменов и атрибутов, целостность

сущностей (потенциальные ключи отношений) и ссылочную целостность (целостность внешних ключей). Декларативные ограничения целостности можно использовать при создании и модификации таблиц средствами языка DDL или в виде отдельных утверждений (ASSERTION).

Например, следующий оператор создает таблицу PERSON и определяет для нее некоторые ограничения целостности:

```
CREATE TABLE PERSON
 (Pers_Id INTEGER PRIMARY KEY,
 Pers_Name CHAR(30) NOT NULL,
 Dept_Id REFERENCES DEPART(Dept_Id) ON UPDATE CASCADE ON DELETE CASCADE);
```

После выполнения оператора для таблицы PERSON будут объявлены следующие ограничения целостности:

- Поле Pers_Id образует потенциальный ключ отношения.
- Поле Pers_Name не может содержать null-значений.
- Поле Dept_Id является внешней ссылкой на родительскую таблицу DEPART, причем, при изменении или удалении строки в родительской таблице каскадно должны быть внесены соответствующие изменения в дочернюю таблицу.

Процедурная поддержка ограничений целостности заключается в использовании триггеров и хранимых процедур.

Не все ограничения целостности можно реализовать декларативно. Примером такого ограничения может служить требование из примера 1, утверждающее, что поле Dept_Kol таблицы DEPART должно содержать количество сотрудников, реально числящихся в подразделении. Для реализации этого ограничения необходимо создать триггер, запускающийся при вставке, модификации и удалении записей в таблице PERSON, который корректно изменяет значение поля Dept_Kol.

Поддержка ограничений целостности в стандарте языка SQL.

Стандарт SQL позволяет задавать **декларативные** ограничения следующими способами:

- Как ограничения домена.
- Как ограничения, входящие в определение таблицы.
- Как ограничения, хранящиеся в базе данных в виде независимых утверждений (assertion).

Стандарт SQL не предусматривает процедурных ограничений целостности, реализуемых при помощи триггеров и хранимых процедур, однако существуют **действия, исполняемые по ссылке**. Эти действия определяют, что будет происходить при изменении значения родительского ключа, на который ссылается некоторый внешний ключ. Эти действия можно задавать независимо для операций обновления (ON UPDATE) или для операций удаления (ON DELETE) записей в родительском отношении.

Стандартом SQL определяется 4 типа действий, исполняемых по ссылке:

- **CASCADE**. Изменения значения родительского ключа автоматически приводят к таким же изменениям связанного с ним значения внешнего ключа. Удаление кортежа в родительском отношении приводит к удалению связанных с ним кортежей в дочернем отношении.
- **SET NULL**. Все внешние ключи, которые ссылаются на обновленный или удаленный родительский ключ получают значения NULL.
- **SET DEFAULT**. Все внешние ключи, которые ссылаются на обновленный или удаленный родительский ключ получают значения, принятые по умолчанию для этих ключей.
- **NO ACTION**. Значения внешнего ключа не изменяются. Если операция приводит к нарушению ссылочной целостности (появляются "висящие" ссылки), то такая операция не выполняется.

Источники вдохновения

6, 7 вопросы: Теория нормальных форм (mstu.edu.ru) (божественные примеры нормальных форм)

8 вопрос: 7) Управление транзакциями СУБД - CoderLessons.com

9: Уровни изолированности транзакций для самых маленьких / Хабр (habr.com)

9: Распределенная СУБД - обработка тупиковых ситуаций - CoderLessons.com

9: Многопользовательский доступ к данным (mstu.edu.ru)

10: 15.7. Поддержка мер обеспечения безопасности в языке sql (studfile.net)

11: Введение в системы управления базами данных. Глава 9. Транзакции и целостность баз данных (citforum.ru)