

104. Укажите основные достоинства реляционного подхода к моделированию данных

- 1) Наличие небольшого набора абстракций которые позволяют просто моделировать большую часть распространенных предметных областей и допускают точные формальные определения оставаясь интуитивно понятными;
- 2) Наличие простого и в то же время мощного математического аппарата опирающегося главным образом на теорию множеств и математическую логику и обеспечивающего теоретический базис реляционного подхода к организации баз данных;
- 3) Возможность ненавигационного (спецификационного) манипулирования данными без необходимости знания конкретной физической организации баз данных во внешней памяти.

105. Перечислите и дайте определения основных структурных понятий реляционной модели.

- 1) **Тип данных**
- 2) **Домен**
- 3) **Атрибут**
- 4) **Кортеж**
- 5) **Отношение**

(+ определения)

- 1) **Тип данных** множество значений и операций над ними.
(аналогично понятию типа данных в ЯП)
- 2) **Домен** множество допустимых значений.
- 3) **Атрибут** именованный домен представляющий семантически значимые объекты
- 4) **Кортеж** это множество пар имя атрибута значение с одним вхождением каждого имени атрибута принадлежащего схеме отношения.
- 5) **Отношение** – это множество кортежей соответствующих одной схеме отношения.

106. Какие свойства характерны для отношений реляционной модели?

- 1) Отсутствие кортежей-дубликатов
- 2) Отсутствие упорядоченности атрибутов
- 3) Атомарность значений атрибутов.
- 4) Отсутствие упорядоченности кортежей

(+ пояснения из методички)

- 1) Отсутствие кортежей-дубликатов

Отношения не содержат кортежей-дубликатов следует из определения отношения как множества кортежей. В классической теории множеств по определению каждое множество состоит из различных элементов. Из этого свойства вытекает наличие у каждого отношения так называемого первичного ключа – набора атрибутов значения которых однозначно определяют кортеж отношения.

- 2) Отсутствие упорядоченности атрибутов

Атрибуты отношений не упорядочены т.к. схема отношения есть множество пар $\langle \text{имя атрибута}, \text{имя домена} \rangle$.

- 3) Атомарность значений атрибутов

Значения всех атрибутов являются атомарными. Это следует из определения домена как потенциального множества значений простого типа данных т.е. среди значений домена не могут содержаться множества значений и агрегаты значений.

- 4) Отсутствие упорядоченности кортежей

Свойство отсутствия упорядоченности кортежей отношения также является следствием определения отношения-экземпляра как множества кортежей.

107. Сформулируйте простейшие правила перехода от ER-схемы Чена к реляционной схеме БД.

1. Каждое множество сущностей представляются самостоятельными отношениями их однозначные атрибуты становятся атрибутами отношения *ключи множества сущностей являются возможными ключами отношения*; при необходимости в качестве первичного ключа используется суррогатный атрибут.

2. Бинарные множества связей типа (1:1) или 1:M без атрибутов представляются дублированием первичного ключа 1 отношения в M отношения. (В случае типа связи 1:1 выбор M отношения осуществляется произвольно.)

3. Бинарные множества связей типа M:N без атрибутов представляются самостоятельными отношениями куда дублируются первичные ключи отношений построенных для множества сущностей.

4. Множества связей с атрибутами или степенью больше 2х представляются самостоятельными отношениями куда дублируются первичные ключи отношений

построенных для множеств сущностей. Однозначные атрибуты множества связей становятся атрибутами этого отношения.

5. Каждый многозначный атрибут множества сущностей или связей представляется отдельным отношением куда дублируется первичный ключ отношения а второй атрибут предназначен для значения.

108. Что такое представление и для чего они предназначены? Какой командой SQL они создаются?

Представление создается с помощью команды **CREATE VIEW**.

Представление это отношение которое в базе данных не существует но создается по требованию пользователя в момент обращения.

Представление – динамический результат одной или нескольких реляционных операций над базовыми отношениями с целью создания некоторого иного отношения. Представление является виртуальным отношением которое реально в базе данных не существует но создается по требованию пользователя в момент обращения к представлению.

Предназначены:

- 1) Для сокрытия части БД от определенных пользователей. Пользователь не будет иметь сведений о существовании атрибутов или кортежей отсутствующих в доступных ему представлениях.
- 2) Для упрощения сложных операций (с базовыми отношениями). Например если представление будет определено на основе соединения двух отношений то пользователь сможет выполнять над ним простые унарные операции выборки и проекции которые будут автоматически преобразованы средствами СУБД в эквивалентные операции с выполнением соединения базовых отношений.
- 3) Для удобства доступа к данным. Одни и те же данные в одно и то же время могут рассматриваться разными пользователями совершенно различными способами.

109. Какие типы ограничений целостности можно декларативно задать в командах языка SQL?

- Ограничения на значения атрибутов
- Ограничения на отображения между атрибутами одного отношения
- Ограничения на отображения между отношениями (под вопросом)

Пока схема такая: называем первые два если скажет что ответ неполный называем третий.

Вариант ответа 2:

- задание обязательности/необязательности значений (NULL/NOT NULL)
- условия проверки значения (CHECK)
- задание уникальности столбца (UNIQUE PRIMARY KEY)

110. Перечислите конструкции языка SQL связанные с ограничениями целостности.

- Тип данных атрибута
- Описатель атрибута или группы атрибутов PRIMARY KEY определяющий первичный ключ отношения
- Описатель атрибута или группы атрибутов UNIQUE определяющий возможный ключ отношения
- Описатели атрибута NULL и NOT NULL позволяющие определить соответственно может или нет атрибут иметь неопределенное значение в кортежах отношения
- Конструкция FOREIGN KEY (REFERENCES) определяющая внешний ключ отношения
- Конструкция CHECK определяющая условие на значение атрибута(ов) которому должны удовлетворять все кортежи отношения

111. Что такое неопределенное значение и логическое значение unknown? Какими свойствами они обладают?

Неопределенное значение (NULL) означает отсутствие значения или его неприемлемость для данного кортежа.

Неопределенное значение не принадлежит никакому типу данных и может присутствовать среди значений любого атрибута определенного на любом типе данных (если это явно незапрещено при определении атрибута).

NULL не следует понимать как нулевое численное значение или заполненную пробелами текстовую строку.

Пусть

a – это значение некоторого типа данных или NULL

op – любая двухместная операция этого типа данных (например +)

lop – операция сравнения значений этого типа (например =) тогда

свойства NULL:

- 1) a op NULL = NULL и наоборот
- 2) NULL op a = NULL
- 3) a lop NULL = unknown и наоборот
- 4) NULL lop a = unknown

Unknown (Неизвестное значение) – это третье значение булевого типа обладающее следующими **свойствами:**

- 1) NOT unknown = unknown
- 2) true AND unknown = unknown

- 3) true OR unknown = true
- 4) false AND unknown = false
- 5) false OR unknown = unknown

112. Укажите два основных правила целостности реляционной модели. Как они обеспечиваются?

1) **Требованием целостности сущностей.**

(любой кортеж любого отношения должен быть отличим от любого другого кортежа любого отношения т. е. любое отношение должно обладать первичным ключом)

Обеспечивается:

Гарантированием отсутствия отношений содержащих кортежидубликаты и что значение первичного ключа не является NULL.

2) **Требованием целостности по ссылкам**

Существует ТРИ подхода обеспечения:

- 1. Запрещается удалять кортеж на который существуют ссылки (т.е. сначала нужно либо удалить ссылающиеся кортежи либо соответствующим образом изменить их значения внешнего ключа).
- 2. При удалении кортежа на который имеются ссылки во всех ссылающихся кортежах значение внешнего ключа автоматически становится неопределенным.
- 3. При удалении кортежа из отношения на которое ведет ссылка из ссылающегося отношения автоматически удаляются все ссылающиеся кортежи.

113. Дайте определения суперключа потенциального ключа составного ключа первичного ключа альтернативного ключа суррогатного ключа. Как они соотносятся друг с другом?

Суперключ атрибут или их множество которое единственным образом идентифицирует кортеж данного отношения.

Потенциальный ключ суперключ который не содержит подмножества также являющегося суперключом данного отношения.

Составной ключ ключ состоящий из нескольких атрибутов.

Первичный ключ – потенциальный ключ который выбран для уникальной идентификации кортежей отношения.

Альтернативный ключ – потенциальный ключ который не выбран в качестве первичного.

Суррогатный ключ – искусственный атрибут отношения не имеющий связей с какойлибо естественной характеристикой явлений ПрО и вводимый в схему отношения исключительно для организации связей кортежей этого отношения с кортежами других отношений.

Все эти ключи исходят из суперключа они входят в множество “суперключа”.

114. Что такое суррогатный первичный ключ? Почему в последнее время проектировщики предпочитают использовать только их?

Суррогатный первичный ключ – искусственный атрибут отношения который так же как и естественный обеспечивает уникальную идентификацию кортежей отношения.

Преимущества первичных суррогатных ключей:

- 1) Компактность
- 2) Единообразие (обычно их имя составляется из стандартного префикса или суффикса и имени отношения – Код сотрудника или СТУДЕНТ_ID и все они имеют один и тот же тип значений – INTEGER со стандартной максимальной длиной)
- 3) Отсутствие потребности в изменениях которые могут затронуть много отношений схемы

115. Что такое внешний ключ? Должен ли он обладать свойством уникальности? Для чего и как он используется?

1. **Внешний ключ** – атрибут или множество атрибутов внутри отношения которые соответствуют потенциальному ключу (как правило первичному) некоторого отношения (может быть того же самого) и является его подмножеством.
2. Внешний ключ **НЕ** обязан быть уникальным. Если он неуникален то связь 1:M иначе 1:1.
3. Используется для связи с первичным ключом в реляционной модели является атрибутом нужен для обеспечения связи между кортежами.

116. Укажите основные компоненты команды SQL CREATE TABLE. Приведите примеры. [ПОЛНОЕ ОПИСАНИЕ СИНТАКСИСА](#)

- 1) **schema** – имя схемы в которой создается таблица (по умолчанию используется схема пользователя в сеансе которого выполняется команда).
- 2) **table** – имя создаваемой таблицы.
- 3) **column** – имя столбца.
- 4) **datatype** – тип данных столбца (CHAR NUMBER DATE BLOB...)
- 5) **CONSTRAINT** ограничение целостности. **constraint_name** – уникальное имя ограничения целостности. (NULL NOT NULL UNIQUE PRIMARY KEY references_clause CHECK)
- 6) **references_clause** определяет ограничение ссылочной целостности для внешнего ключа таблицы.

Команды CREATE TABLE для медицинской БД

```

CREATE TABLE БОЛЬНИЦА (
  К/Б          NUMBER (6)          PRIMARY KEY,
  Название     VARCHAR2 (30)        NOT NULL,
  Адрес        VARCHAR2 (50),
  Ч/К          NUMBER (4))

CREATE TABLE ПАЛАТА (
  К/Б          NUMBER (6)          NOT NULL
                                REFERENCES БОЛЬНИЦА (К/Б) ON DELETE CASCADE,
  Н/П          NUMBER (6)          NOT NULL,
  Название     VARCHAR2 (30),
  Ч/К          NUMBER (2),
  PRIMARY KEY (К/Б,Н/П))

CREATE TABLE ВРАЧ (
  К/В          NUMBER (6)          PRIMARY KEY,
  К/Б          NUMBER (6)          REFERENCES БОЛЬНИЦА (К/Б) ON DELETE SET NULL,
  Фамилия      VARCHAR2 (30)        NOT NULL,
  Специальность VARCHAR2 (30))

CREATE TABLE ПАЦИЕНТ (
  Р/Н          NUMBER (6)          PRIMARY KEY,
  Фамилия      VARCHAR2 (30)        NOT NULL,
  Адрес        VARCHAR2 (50),
  Д/Р          DATE,
  Пол          CHAR (1)            CHECK (Пол IN ('М','Ж')),
  НМП          VARCHAR2 (20)        UNIQUE)

CREATE TABLE ВРАЧ-ПАЦИЕНТ (
  К/В          NUMBER (6)          NOT NULL
                                REFERENCES ВРАЧ (К/В) ON DELETE CASCADE,
  Р/Н          NUMBER (6)          NOT NULL
                                REFERENCES ПАЦИЕНТ (Р/Н) ON DELETE CASCADE,
  PRIMARY KEY (К/В,Р/Н))

```

117. Что такое триггер? Для чего они предназначены?

Триггер – это программа на языке программирования сервера которая автоматически выполняется СУБД в момент наступления какого-то определенного события (Для Oracle PL/SQL).

Используются:

- 1) Для проверки правильности введенных данных и наложения сложных ОЦ на данные.
- 2) Для осуществления косвенных модификаций данных сопутствующих действиям пользователей (например в случае денормализации схемы);
- 3) Для выдачи предупреждений напоминающих о необходимости выполнить какиелибо действия (например с помощью электронной почты)
- 4) Для накопления сведений об изменениях и лицах которые их совершили.

118. При каких событиях в системе БД могут запускаться триггеры? Какие факторы влияют на запуск триггеров обновления данных?

События при которых могут запускаться триггеры:

- 1) Операторы INSERT UPDATE DELETE применяемые к таблице или представлению
- 2) Операторы CREATE ALTER или DROP применяемые к любому объекту схемы
- 3) Регистрация пользователя в системе или выход из нее
- 4) Запуск базы данных или остановка экземпляра Oracle
- 5) Сообщение об ошибке

Время запуска триггера определяется с помощью ключевых слов BEFORE и AFTER: при указании BEFORE триггер запускается до возникновения связанных с ним событий а AFTER – после.

119. Чем отличаются триггеры для таблиц от триггеров для представлений?

Триггеры для таблиц отличаются от триггеров для представлений использованием основных команд SQL CREATE TRIGGER.

Есть два типа триггеров:

строковые триггеры (для таблицы) которые выполняются для каждой затронутой активизирующим событием строки таблицы

операторные триггеры выполняющиеся только один раз для всего события даже если активизирующее событие затрагивает множество строк (триггер для представлений)

BEFORE AFTER. нельзя использовать в триггерах для представлений;
(*Before:* в теле триггера можно изменять NEWзначения и нельзя изменять OLDзначения столбцов;
After: в теле триггера нельзя изменять ни NEWзначения ни OLDзначения столбцов.)

INSTEAD OF нельзя использовать в триггерах для таблиц
(в теле триггера можно читать NEWзначения и OLDзначения и нельзя изменять ни NEWзначения ни OLDзначения столбцов.)

Ключевое слово NEW в теле триггера используется для ссылки на новое значение столбца ключевое слово OLD может быть использовано для ссылки на старое значение столбца.

120. Как в коде триггера можно ссылаться на значения столбцов модифицируемых строк?

С помощью ключевых слов **NEW** и **OLD**.

NEW в теле триггера используется для ссылки на новое значение столбца

OLD на старое значение столбца.

(Ссылки на OLD неприменимы для событий удаления а на NEW для событий вставки)

Синтаксис обращений – { :NEW | :OLD }.<имя столбца>.

121. Какова последовательность выполнения триггеров и основного действия с данными?

Запуск триггеров по мере возникновения триггерных событий (INSERT UPDATE DELETE) происходит в следующем порядке:

1. Выполнение табличного триггера BEFORE на уровне оператора.
2. Для каждой строки охваченной данным оператором:
 - а) Выполнение любого триггера BEFORE на уровне строки
 - б) Блокировка данных и выполнение основного действия предписанного командой SQL (типа mutex'а xD)
 - в) Выполнение проверок ОЦ
 - г) Выполнение любого триггера AFTER на уровне строки
3. Выполнение табличного триггера AFTER на уровне оператора.

При разработке триггеров BEFORE необходимо руководствоваться правилом что они не должны вносить дополнительных изменений в базу данных т.к. они исполняются до выполнения проверок ОЦ.

122. Назовите основные компоненты команды SQL CREATE TRIGGER. Приведите примеры.

ПОЛНЫЙ СИНТАКСИС CREATE TRIGGER

Убраны элементы которые по словам Бабанова на лекции мы в принципе писать не будем. Если доебется смотрите файл по ссылке сверху

1. **schema** – имя схемы в которой создается триггер (по умолчанию используется схема пользователя в сеансе которого выполняется команда).
2. **trigger** – имя триггера.
3. **время выполнения триггера:** **BEFORE** (до выполнения основного действия) **AFTER** (после выполнения основного действия) **INSTEAD OF** (вместо выполнения основного действия).
4. **ddl_event** определяет грамматику триггеров для команд языка определения данных.

5. *dml_event_clause* определяет грамматику триггеров для команд языка манипулирования данными. Команды указывающие на то для чего будет выполняться триггер: **DELETE** (для удаления строк таблицы) **INSERT** (для добавления строк в таблицу) **UPDATE** (для модификации строк таблицы) **FOR EACH ROW** (определяет является ли триггер строковым. Если эта фраза опущена триггер является операторным). **ON** указывает на имя таблицы на изменения в которой будет реагировать триггер.
6. **WHEN** определяет дополнительное условие при истинности которого будет рl выполняться триггер.
7. *pl/sql_block* определяет блок PL/SQL который выполняется когда триггер активизируется

123. Каковы основные особенности навигационного стиля манипулирования реляционными данными? Когда используется навигационный стиль манипулирования реляционными данными?

- Особенность построчное считывание информации при помощи циклов
- Используется: в интерфейсах с реляционными СУБД
- Все эти интерфейсы строятся по принципу:
 - 1) СУБД передается запрос на выборку данных
 - 2) В памяти программы отводится место для размещения значений который считываются из текущей строки
 - 3) Затем с помощью операторов **цикла ЯП** происходит построчное сканирование результирующей выборки с чтением и обработкой следующих данных одной строки

(***Для обеспечения произвольных алгоритмов манипулирования данными в навигационном языке должна быть предусмотрена возможность параллельного независимого сканирования одной и той же или различных таблиц.***)

124. Что собой представляют курсоры PL/SQL?

Курсор это механизм обработки селектируемых строк.

PL/SQL использует два типа курсоров – **неявные** и **явные**:

- 1) **Неявные** организуются системой неявно для всех команд SQL если они селектируют и выполняют некоторые действия с одной строкой.
- 2) **Явные** объявляются в области объявлений если селектируют и выполняют действия над множеством строк. Необходимо объявить явный курсор открыть его в цикле этого курсора просканировать строки результата выполняя требуемые действия с данными и закрыть курсор

Аппарат курсоров языка PL/SQL позволяет добавить к функциональности спецификационных команд SQL возможность построчного сканирования результирующих таблиц с данными.

125. Какие команды предусмотрены в языке PL/SQL для объявления и обращения к курсорам?

CURSOR объявление курсора

SELECT определяется по указанному имени курсора

OPEN открытие курсора

FETCH чтение строк

CLOSE закрытие курсора

Конструкция BULK COLLECT команды *FETCH* позволяет за одно обращение к ней прочитать целиком все столбцы результирующей таблицы в соответствующие коллекции типы данных которых соответствуют типам данных столбцов.

126. Как управлять процессом обращений к курсору с помощью атрибутов курсора?

Каждый явный курсор имеет четыре атрибута – *%ISOPEN* *%FOUND* *%NOTFOUND* *%ROWCOUNT*. Синтаксис обращения – имя курсора атрибут

Эти атрибуты возвращают полезную информацию о выполнении обращений к данным результирующей таблицы. Их следует использовать для адекватной обработки ситуаций связанных с курсором.

- 1) ***%ISOPEN*** возвращает значение TRUE если курсор открыт FALSE – в противном случае.
- 2) ***%FOUND*** принимает значение:
 - а) NULL если курсор открыт но команды *FETCH* для него не выполнялись
 - б) TRUE если последняя команда *FETCH* прочитала данные из текущей строки
 - в) FALSE если очередной *FETCH* вышел за границу курсора.

До открытия курсора и после его закрытия обращение к атрибуту *%FOUND* вызывает исключительную ситуацию *INVALID_CURSOR*.

- 3) ***%NOTFOUND*** возвращает TRUE когда *%FOUND* дает FALSE и наоборот. В остальных случаях реакция аналогична работе *%FOUND*.
- 4) ***%ROWCOUNT*** равен нулю сразу после открытия курсора и числу строк результирующей таблицы прочитанных до этого командами *FETCH* в процессе сканирования.

До открытия курсора и после его закрытия обращение к атрибуту %ROWCOUNT вызывает исключительную ситуацию INVALID_CURSOR.

ПРИМЕР:

```
DECLARE
  p_name ПЕРСОНАЛ.Фамилия%TYPE;
  p_sal ПЕРСОНАЛ.З/П%TYPE;
  CURSOR cursor1 IS
    SELECT Фамилия, З/П FROM ПЕРСОНАЛ
    WHERE Должность = 'СИДЕЛКА';
BEGIN
  ***
  OPEN cursor1;
  LOOP
    FETCH cursor1 INTO p_name, p_sal;
    EXIT WHEN cursor1%NOTFOUND;
    -- обработка одной записи
  END LOOP;
  CLOSE cursor1;
  ***
END;
```

127. Укажите и охарактеризуйте классы спецификационных языков реляционной модели.

1. **Алгебраические языки** позволяют выражать запросы средствами специализированных операторов применяемых к отношениям.
2. **Языки исчисления предикатов** в которых запросы описывают требуемое множество кортежей путем указания логического выражения которому должны удовлетворять эти кортежи. В зависимости от вида объектов которые обозначают переменные языки этого класса делятся на:
 - a. языки реляционного исчисления с переменными кортежами (ALPHA)
 - b. языки реляционного исчисления с переменными на доменах. (QBE)
3. **SQLподобные языки** относятся к классу реляционных языков основанных на отображениях. Каждая конструкция SELECT этих языков задает отображение строк исходных таблиц в строки результирующей таблицы.

128. Поясните деление языков на процедурные и декларативные
Запрос в **процедурных языках** представляет алгоритм получения результата (требуют полного описания последовательности шагов) а в **декларативных языках** представляет собой некоторый образец или условие по которым необходимо найти требуемые данные (нужно описать все факты и правила данной ситуации).

Примеры:

Декларативные: SQL язык запросов по образцу QBE

Процедурные: реляционная алгебра

129. Дайте определение основных и дополнительных операций реляционной алгебры Кодда. Поясните на примерах их работу.

Основные:

1) **Объединение (UNION).**

Объединение двух отношений представляет собой множество кортежей которые принадлежат либо только одному отношению либо им обоим.

Операнды теоретикомножественных операций UNION MINUS и INTERSECT должны иметь совпадающие с точностью до имен атрибутов заголовки.

2) **Разность (MINUS).**

Разностью отношений называется множество кортежей принадлежащих первому отношению но не принадлежащих второму отношению

$$R - S = \{t \mid t \in R \wedge t \notin S\}.$$

3) **Декартово произведение (TIMES).**

Декартовым произведением отношений называется множество всех кортежей степени k_1+k_2 где первые k_1 компонентов которых образуют кортежи принадлежащие первому отношению а последние k_2 – кортежи принадлежащие второму отношению.

4) **Проекция (PROJECT).**

Существо этой операции заключается в том что берется отношение удаляются некоторые из его атрибутов и (или) переупорядочиваются оставшиеся атрибуты.

5) **Селекция (SELECT).**

В этом случае результат операции селекции есть множество кортежей принадлежащих отношению таких что при подстановке i го компонента кортежа вместо всех вхождений номера i в формулу F для всех i она станет истинной.

Наряду с номерами в операциях селекции можно использовать имена атрибутов.

Дополнительные:

1) **Пересечение (INTERSECT).**

Пересечение отношений R и S представляет собой множество кортежей которые одновременно принадлежат и R и S

2) **Частное (DIVIDE).**

Множество кортежей t длины $(r - s)$ таких что для всех кортежей u длины s принадлежащих S кортеж $t||u$ принадлежит R

(Θ читается как 'тета')

3) **Соединение (JOIN). Θ соединение (Θ JOIN)**

Θ соединение R и S представляет собой множество кортежей их Декартова произведения что i ый компонент R находится в отношении Θ с j ым компонентом S. Если Θ является оператором равно (=) эта операция часто называется эквисоединением (EQUIJOIN)

Реляционная алгебра Кодда

Совместимые отношения* – это отношения, у которых совпадают заголовки (количество и имена атрибутов)

Основные операции

1. **ОБЪЕДИНЕНИЕ*** $R \cup S = \{t \mid t \in R \vee t \in S\}$
2. **РАЗНОСТЬ*** $R - S = \{t \mid t \in R \wedge t \notin S\}$
3. **ДЕКАРТОВО ПРОИЗВЕДЕНИЕ** $R \times S = \{r \parallel s \mid r \in R \wedge s \in S\}$
4. **ПРОЕКЦИЯ**
 $\pi_{i_1, \dots, i_m}(R) = \{ \langle a_1, \dots, a_m \rangle \mid \exists \langle b_1, \dots, b_k \rangle (\langle b_1, \dots, b_k \rangle \in R \wedge \forall j = \overline{1, m} (a_j = b_{i_j})) \}$
5. **СЕЛЕКЦИЯ** $\sigma_F(R) = \{r \mid r \in R \wedge F\}$

Дополнительные операции

1. **ПЕРЕСЕЧЕНИЕ*** $R \cap S = \{t \mid t \in R \wedge t \in S\} = R - (R - S)$
2. **ЧАСТНОЕ** $R \div S = \{t^{(r-s)} \mid \forall u^{(s)} (u \in S \rightarrow t \parallel u \in R)\}$
 $R \div S = \pi_{1, \dots, r-s}(R) - \pi_{1, \dots, r-s}((\pi_{1, \dots, r-s}(R) \times S) - R)$
3. **СОЕДИНЕНИЕ**
 $R \bowtie_j S = \sigma_{i\theta_j(r+j)}(R \times S)$
 $R \bowtie_{X\theta Y} S = \{r \parallel s \mid r \in R \wedge s \in S \wedge (\pi_X(r) \theta \pi_Y(s))\}$

130. Проведите на конкретном примере сравнительный анализ всех разновидностей операции соединения

Объединим группы X и группы Y

При естественном соединении (или эквисоединении (это inner join на диаграмме)) в результате будут исключены по одному экземпляру из каждой пары совпадающих атрибутов (атрибуты группы X или Y)

При композиционном соединении (это естественное соединение (это outer excluding join на диаграмме как я понял)) атрибуты соединения не включаются в результат (атрибуты групп X и Y).
(такое соединение уместно при использовании для связей кортежей суррогатных ключей)

При левом внешнем соединении к результату естественного соединения добавляются не вошедшие кортежи левого операнда конкатенированные кортежами заполненными

NULL'ами справа степень этих кортежей равна степени правого операнда. Будут включены (атрибуты группы X).

| a1 | b1 | | b2 | c2 |
|----|----|--|----|----|
| a | 1 | | 1 | x |
| b | 1 | | 2 | y |
| c | 3 | | 3 | z |
| d | 4 | | | |

Левое внешнее соединение

| a1 | b1 | b2 | c2 |
|----|----|------|------|
| a | 1 | 1 | x |
| b | 1 | 1 | x |
| c | 3 | 3 | z |
| d | 4 | Null | Null |

(Проще сказать что все ровно наоборот)

При правом внешнем соединении к результату естественного соединения добавляются не вошедшие кортежи правого операнда конкатенированные кортежами заполненными NULL'ами слева степень этих кортежей равна степени левого операнда. Будут включены (атрибуты группы Y).

Правое внешнее соединение

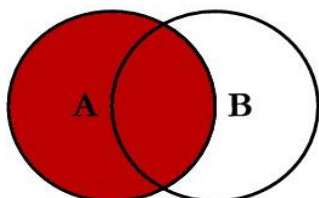
| a1 | b1 | b2 | c2 |
|------|------|----|----|
| a | 1 | 1 | x |
| b | 1 | 1 | x |
| c | 3 | 3 | z |
| Null | Null | 2 | y |

При полном внешнем соединении результат совпадает с объединением левого и правого внешних соединений. Ничего не исключается из результата.

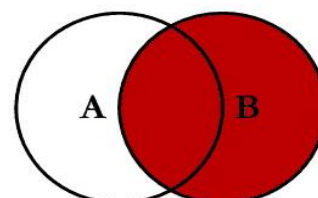
Полное внешнее соединение

| a1 | b1 | b2 | c2 |
|------|------|------|------|
| a | 1 | 1 | x |
| b | 1 | 1 | x |
| c | 3 | 3 | z |
| d | 4 | Null | Null |
| Null | Null | 2 | y |

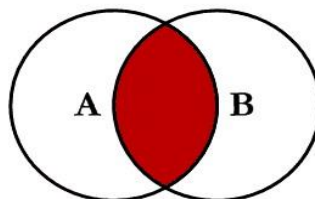
SQL JOINS



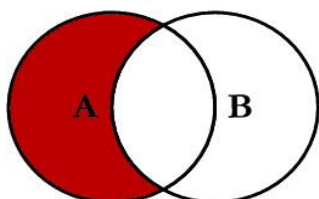
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



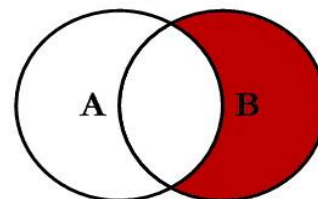
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



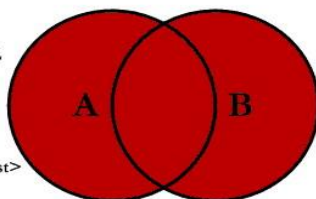
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



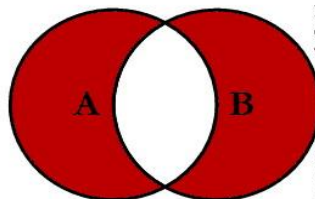
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

131. Как в языке реляционной алгебры выполняются действия изменяющие состояние БД?

Язык реляционной алгебры *обычно не используют* для изменения состояния БД однако действия включения модификации и удаления кортежей можно выполнить и в этом языке

- 1) действие включения выполняется операцией UNION;
- 2) действие удаления выполняется операцией MINUS;
- 3) действие модификации сводится к действию удаления и последующего включения;

Во всех этих случаях первым операндом и результатом является изменяемое базовое отношение;

Вторым операндом этих операций является либо производное отношение полученное запросом выборки либо отношение кортежи которого заданы литерально (явным указанием их компонентов).

132. Какой вид имеют запросы в реляционном исчислении с переменными кортежами?

(ψ пси)

Запросы в реляционном исчислении с переменными кортежами имеют вид $\{t \mid \psi(t)\}$ (t такое что пси от t) где t – переменная кортеж т.е. переменная обозначающая кортеж некоторой фиксированной длины а ψ – формула построенная из атомов и

совокупности операторов при этом t – единственная свободная переменная кортеж в формуле ψ .

133. Укажите разновидности атомов формул реляционного исчисления с переменными кортежами.

Атомы формул $\psi(\text{пси})$ могут быть двух типов.

1. $R(s)$ где R – имя отношения а s – переменная кортеж. Этот атом принимает значение истина когда s есть кортеж отношения R .
2. $s[i] \Theta(\text{тета}) u[j]$ где s и u являются переменными кортежами или константами а $\Theta(\text{тета})$ – оператор сравнения. Этот атом принимает значение истина когда i ый компонент s находится в отношении $\Theta(\text{тета})$ с j ым компонентом u .
3. Сравнение элемента кортежа с константой. $s[i] \Theta a$ или $a \Theta s[i]$ где Θ и $s[i]$ имеют тот же смысл а a – это константа. Этот атом принимает значение истина когда i ый компонент s находится в отношении Θ с константой a .

134. Перечислите правила построения формул реляционного исчисления с переменными кортежами.

(ψ пси)

Формулы а также свободные и связанные вхождения переменных кортежей в эти формулы определяются рекурсивно следующим образом:

1. Каждый атом есть формула. Все вхождения переменных кортежей упомянутые в атоме являются свободными в этой формуле.
2. Если ψ_1 и ψ_2 – формулы то $\psi_1 \wedge \psi_2$ $\psi_1 \vee \psi_2$ и $\neg \psi_1$ – тоже формулы (ψ_1 или ψ_2 не ψ_1)
3. Если ψ – формула в которой есть свободная переменная s то $\exists s (\psi)$ – также формула. (существует s от ψ)
4. Если ψ – формула в которой есть свободная переменная s то $\forall s (\psi)$ – также формула. (любое s от ψ)
5. Формулы могут заключаться в круглые скобки для изменения приоритета операторов.
6. Ничто иное не является формулой.

Реляционное исчисление с переменными-кортежами

Запрос – $\{t \mid \psi(t)\}$

Атомы:

1. $R(s)$
2. $s[i] \Theta u[j]$
3. $s[i] \Theta a$ и $a \Theta s[i]$

Правила образования формул:

1. Атом – формула.
2. Если ψ_1 и ψ_2 – формулы, то $\psi_1 \wedge \psi_2, \psi_1 \vee \psi_2, \neg \psi_1$ – тоже формулы.
3. Если ψ – формула, то $\exists s(\psi)$ – тоже формула.
4. Если ψ – формула, то $\forall s(\psi)$ – тоже формула.
5. Приоритет операций – $\Theta, \left\{ \begin{matrix} \exists \\ \forall \end{matrix} \right\}, \neg, \wedge, \vee$
6. Круглые скобки для изменения приоритета.
6. Ничто иное не является формулой.

135. Как определяется статус связанасвободна переменныхкортежей?

Вхождение переменной в формулу является связанным если этой переменной предшествует квантор для всех или существует. В противном случае мы называем переменную свободной.

136. Какой вид имеют запросы в реляционном исчислении с переменными на доменах?

Запросы в реляционном исчислении с переменными на доменах имеют вид

$$\{x_1 x_2 \dots x_k \mid \psi(x_1, x_2, \dots, x_k)\}, \text{ где } x_1, x_2, \dots, x_k$$

($x_1 x_2 \dots x_k$ такие что ψ от $x_1 x_2 \dots x_k$) где $x_1 x_2 \dots x_k$ – переменные на доменах т.е. переменные обозначающие скалярные значения взятые из определенных доменов а ψ – формула построенная из атомов и совокупности операторов.

137. Укажите разновидности атомов формул реляционного исчисления с переменными на доменах.

Атомы формул ψ (пси) могут быть двух типов.

1. $R(x_1 x_2 \dots x_k)$ где R – имя отношения степени k а каждое x_i есть константа или переменная на домене. Этот атом принимает значение истина когда $x_1 x_2 \dots x_k$ есть кортеж отношения R .
2. $x \Theta u$ где x и u являются константами или переменными на доменах а Θ – оператор сравнения. Этот атом принимает значение истина когда x находится в отношении Θ с u .

138. Перечислите правила построения формул реляционного исчисления с переменными на доменах.

1. Каждый атом есть формула. Все вхождения переменных на доменах упомянутые в атоме являются свободными в этой формуле.
2. Если ψ_1 и ψ_2 – формулы то $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$ и $\neg \psi_1$ – тоже формулы (ψ_1 или ψ_2 не ψ_1)
3. Если ψ – формула в которой есть свободная переменная x то $\exists x (\psi)$ – также формула. (существует x от ψ)
4. Если ψ – формула в которой есть свободная переменная x то $\forall x (\psi)$ – также формула. (любое x от ψ)
5. Формулы могут заключаться в круглые скобки для изменения приоритета операторов. Предполагается следующий порядок старшинства операторов –

$$\Theta, \left\{ \begin{matrix} \exists \\ \forall \end{matrix} \right\}, \neg, \wedge, \vee.$$

(оператор сравнения существует/для любого отрицание и или)

6. Ничто иное не является формулой.

139. Как определяется статус связанасвободна переменных на доменах?

То же самое что и на кортежах получается

Вхождение переменной в формулу является связанным если этой переменной предшествует квантор для всех или существует. В противном случае мы называем переменную свободной.

может это

(Запрос реляционного исчисления с переменными на доменах есть выражение вида $(x_1 x_2 \dots x_k \text{ такие что } \psi \text{ от } x_1 x_2 \dots x_k)$

$\{x_1 x_2 \dots x_k \mid \psi(x_1, x_2, \dots, x_k)\}$, где x_1, x_2, \dots, x_k – **свободные переменные** на доменах в формуле ψ и **других** свободных переменных в этой формуле **нет**)

140. Перечислите основные отличительные особенности языка QBE.

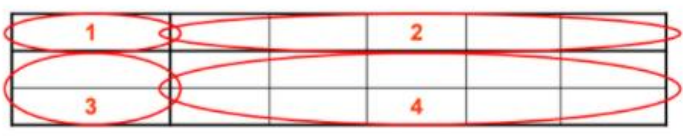
- Ориентация на диалоговое взаимодействие человека с системой БД.
- Двумерный синтаксис команд. (Поскольку операции задаются в табличной форме говорят что QBE имеет двумерный синтаксис)
- Минимум вводимых человеком символов.
- Объединение в одном и том же синтаксисе функциональности всех языков системы БД (языка определения данных языка манипулирования данными языка определения ограничений целостности языка безопасности данных).

- Мощная и в то же время бескванторная человеческая логика.

141. Какие группы полей выделяются в таблице-шаблоне QBE? К каким элементам БД они относятся?

| Группы полей: | К каким элементам БД относятся: |
|--|---|
| 1) Левое верхнее поле таблицы-шаблона | Идентификации отношения и действия с ним |
| 2) Остальные поля заголовка таблицы-шаблона | Идентификация атрибутов отношения и действия с ними |
| 3) Поля сбоку таблицы-шаблона (за исключением верхнего) | Действия с кортежами отношения |
| 4) Поля тела таблицы-шаблона (за исключением левого столбца) | Действия с компонентами кортежа |

Язык QBE



Области таблицы-шаблона запроса:

1. Действия с отношениями.
2. Действия с атрибутами.
3. Действия с кортежами.
4. Действия с компонентами кортежей.

Основная грамматическая конструкция
 [<действие>.] [<операция сравнения>]
 [{<элемент-константа>|<элемент-пример>}]

<действие> ::= {I|U|D|P}

142. Опишите на примере последовательность совместных действий пользователя и системы по формулированию запроса QBE.

- 1) Предлагается пустой шаблон таблицы.
- 2) Пользователь заполняет поле в левом верхнем углу именем таблицы
- 3) Поля шапки таблицы за исключением левого могут заполниться автоматически

- 4) После этого пользователь формулирует запрос ($x_1 x_2 \dots x_k$ такие что больница от $x_1 x_2 \dots x_k$)
 $\{x_1 x_2 x_3 x_4 \mid БОЛЬНИЦА(x_1 x_2 x_3 x_4)\}$ или $\{f \mid \exists x \exists y (ВРАЧ(x) f' ХИРУРГ(y))\}$
- 5) После того как запрос сформулирован пользователь нажимает клавишу Enter для получения ответа.
- 6) Если необходимы две или более таблиц можно сформировать дополнительные пустые шаблоны (используя специальные функциональные клавиши) и затем ввести информацию в их заголовки.
- 7) Условия заданные в одной строке связываются конъюнкцией. Для дизъюнкции условий их нужно указать в разных строках.
- 8) Порядок строк в запросах несущественен.

**Реляционное исчисление с переменными на доменах.
Язык QBE. Примеры запросов**

Получить полные сведения о больницах
 $\{x_1 x_2 x_3 x_4 \mid БОЛЬНИЦА(x_1 x_2 x_3 x_4)\}$

| БОЛЬНИЦА | К/Б | Название | Адрес | Ч/К |
|----------|------|----------|-------|------|
| | P.X1 | P.X2 | P.X3 | P.X4 |

| БОЛЬНИЦА | К/Б | Название | Адрес | Ч/К |
|----------|-----|----------|-------|-----|
| P. | | | | |

Получить фамилии хирургов
 $\{f \mid \exists x \exists y (ВРАЧ(x) f' ХИРУРГ(y))\}$

| ВРАЧ | К/В | К/Б | Фамилия | Специальность |
|------|-----|-----|---------|---------------|
| | | | P.E | ХИРУРГ |

Язык QBE. Примеры запросов (продолжение)

Выдать названия палат больницы с кодом 5, имеющих более 10 коек

| ПАЛАТА | К/Б | Н/П | Название | Ч/К |
|--------|-----|-----|----------|-----|
| | 5 | | P.X | >10 |

Выдать фамилии пациентов – женщин до 30 лет или мужчин после 50 лет

| ПАЦИЕНТ | Р/Н | Фамилия | Адрес | Д/Р | Пол | НМП |
|---------|-----|---------|-------|-------|-----|-----|
| | | P.X | | >1976 | Ж | |
| | | P.Y | | <1956 | М | |

Выдать фамилии пациентов от 30 до 50 лет

| ПАЦИЕНТ | Р/Н | Фамилия | Адрес | Д/Р | Пол | НМП |
|---------|-----|---------|-------|-------|-----|-----|
| | | P.X | | <1976 | | |
| | | X | | >1956 | | |

Язык QBE. Примеры запросов (продолжение)

Выдать фамилии врачей, лечащих пациента с Р/Н 111111

| ВРАЧ | К/В | К/Б | Фамилия | Специальность |
|------|-----|-----|---------|---------------|
| | Y | | P.X | |

| ВРАЧ-ПАЦИЕНТ | К/В | Р/Н |
|--------------|-----|--------|
| | Y | 111111 |

Выдать фамилии врачей, не лечащих пациента с Р/Н 111111

| ВРАЧ | К/В | К/Б | Фамилия | Специальность |
|------|-----|-----|---------|---------------|
| | Y | | P.X | |

| ВРАЧ-ПАЦИЕНТ | К/В | Р/Н |
|--------------|-----|----------|
| | 1 | Y 111111 |

Выдать фамилии врачей, лечащих кого-нибудь, кроме пациента с Р/Н 111111

| ВРАЧ | К/В | К/Б | Фамилия | Специальность |
|------|-----|-----|---------|---------------|
| | Y | | P.X | |

| ВРАЧ-ПАЦИЕНТ | К/В | Р/Н |
|--------------|-----|----------|
| | Y | <>111111 |

Если пользователь введет P. TAB P. (или P. P.) в поле имени таблицы система выведет структуру БД т. е. имена таблиц и имена соответствующих этим таблицам столбцов.

P. TAB показывает все доступные таблицы БД

143. Какие различные синтаксические конструкции с ключевым словом SELECT предусмотрены в стандарте SQL? Для каких ситуаций использования они предназначены? В чем особенности каждой конструкции?

Язык допускает три типа синтаксических конструкций начинающихся с ключевого слова SELECT: спецификация курсора (cursor specification) оператор выборки (select statement) и подзапрос (subquery).

Курсор – это средство языка SQL позволяющее с помощью набора специальных операторов получить построчный доступ к результату запроса к БД.

В основе каждой из них лежит синтаксическая конструкция табличное выражение.

Особенность:

К табличным выражениям не предъявляются какие-либо ограничения.

Оператор выборки – это отдельный оператор языка SQL/89 позволяющий получить результат запроса в прикладной программе без использования курсора.

Особенность:

Ограничение результирующая таблица должна содержать не более одной строки.

Подзапрос запрос который может входить в предикат условия выборки оператора SQL.

Особенность:

- + Ограничение результирующая таблица должна содержать в точности один столбец.
- + Вместо констант разделов WHERE и HAVING можно использовать значения столбцов текущих строк таблиц внешних запросов.

144. В чем заключается основная семантика табличного выражения команды SELECT? Из каких разделов оно состоит и для чего предназначен каждый раздел?

Семантика табличного выражения состоит в том что на основе последовательного применения разделов from where group by и having из заданных в разделе from таблиц строится некоторая новая результирующая таблица порядок следования строк которой не определен и среди строк которой могут находиться дубликаты (т.е. в общем случае таблица-результат табличного выражения является мультимножеством строк).

1) **FROM**

Результатом выполнения раздела FROM является расширенное декартово произведение таблиц заданных списком таблиц раздела FROM.

2) **WHERE**

Результатом раздела WHERE является таблица состоящая из тех строк R для которых результатом вычисления условия поиска является true.

3) **GROUP BY**

сгруппированная таблица

Результатом раздела GROUP BY является разбиение R на множество групп строк в которых для каждого столбца из списка столбцов раздела GROUP BY во всех строках каждой группы значения этого столбца совпадают.

4) **HAVING**

Результатом выполнения раздела HAVING является сгруппированная таблица содержащая только те группы строк для которых результат вычисления условия поиска есть true.

145. Опишите в целом алгоритм вычисления табличного выражения команды SELECT.

Если табличное выражение содержит только раздел FROM (это единственный обязательный раздел табличного выражения) то результат табличного выражения совпадает с результатом раздела FROM.

Если в табличном выражении присутствует раздел WHERE то далее вычисляется он.

Если в табличном выражении присутствует раздел GROUP BY то далее выполняется он.

Наконец последним при вычислении табличного выражения выполняется раздел HAVING (если он присутствует).

146. Каковы особенности использования неопределенных значений атрибутов и логического значения unknown в запросах SQL?

(“Вычисление условия поиска должно производиться не в булевой а в трехзначной логике со значениями true false и unknown (неизвестно)”.)

true AND unknown = unknown
false AND unknown = false
unknown AND unknown = unknown
true OR unknown = true
false OR unknown = unknown
unknown OR unknown = unknown
NOT unknown = unknown.

Для использования NULL (неопределенного значения) должны выполняться:

- требование целостности сущностей любое отношение должно обладать первичным ключом.
- требование целостности по ссылкам в отношении не должно быть повторяющихся кортежей.

Логическое значение unknown используется только в том случае если используется NULL.

Вариант 2:

Неопределённое значение NULL определяет обязательность (NOT) и необязательность (YES) значений столбца в строках таблицы.

Для проверки наличия значения NULL в предложении WHERE запроса используются ключевые слова IS NULL или IS NOT NULL.

Логические операторы включающие значения UNKNOWN будут возвращать UNKNOWN за исключением случаев когда результат оператора не зависит от выражения UNKNOWN

- Значение выражения **не определено** если в его вычислении участвует хотя бы одно неопределенное значение.
- В контексте GROUP BY DISTINCT и ORDER BY неопределенное значение выступает как специальный **вид определенного значения** т.е. возможно например образование группы строк значение указанного столбца которых является неопределенным.

147. Какие виды атомов (предикатов) предусмотрены в стандарте SQL для логического выражения условия поиска?

- 1) предикат сравнения
- 2) предикат between (между)
- 3) предикат exists
- 4) предикат in
- 5) предикат like
- 6) предикат с квантором
- 7) предикат null

148. В каких случаях атомы (предикаты) принимают значения true false и unknown?

1. предикат сравнения

Если хотя бы один из операндов операции сравнения имеет неопределенное значение или если правый операнд является подзапросом с пустым результатом то значение предиката сравнения равняется **unknown**.

2. предикат between (между)

Результат тот же самый что и для предикатов сравнения поэтому значение предиката равно **true** если значения обоих предикатов сравнения равны true **false** если хотя бы одно значение предиката сравнения равно false. **unknown** если значение хотя бы одного предиката сравнения равно unknown.

3. предикат in

Значение предиката равно **true** в том и только том случае когда *значение левого операнда совпадает хотя бы с одним значением списка правого операнда.*

Если список правого операнда пуст или значение подразумеваемого предиката сравнения $x = y$ равно false для каждого элемента y списка правого операнда то значение предиката in равно false.

В противном случае значение предиката in равно **unknown**.

4. предикат like

Значение предиката равно **true** если значение указанного столбца удовлетворяет заданному шаблону. (**false** если не удовлетворяет)

Значение предиката like есть **unknown** если значение столбца либо шаблона не определено.

5. предикат null

Этот предикат всегда принимает значения *true* или *false*. Значение предиката равно **true** тогда и только тогда когда значение указанного столбца не определено.

6. предикат с квантором

Для **ALL**: значение предиката равно **true** если результат вычисления подзапроса пуст или значение предиката равно *true* для каждого элемента входящего в подзапрос.

Значение предиката равно **false** если хотя бы для одного элемента входящего в подзапрос значение предиката равно *false*. В остальных случаях значение предиката равно **unknown**.

Для **SOME**: значение предиката равно **false** если результат вычисления подзапроса пуст или значение предиката равно *false* для каждого элемента в подзапросе. Значение предиката равно **true** если значение предиката равно *true* хотя бы для одного элемента входящего в подзапрос. В остальных случаях значение предиката равно **unknown**.

7. предикат exists.

Значением этого предиката всегда является *true* или *false* и это значение равно **true** тогда и только тогда когда результат вычисления подзапроса не пуст.

| Предикат | true | false | unknown |
|-----------|---|-------|---|
| Сравнения | Стандартно для операций $< > = <>$ и т.д. | | <ul style="list-style-type: none">Если хотя бы один из операндов операции сравнения имеет неопределенное значениеесли правый операнд является подзапросом с пустым результатом |

| | | | |
|----------------|--|--|--|
| <i>between</i> | <p>Т.к <i>x BETWEEN y AND z</i> эквивалентно $x \geq y \text{ AND } x \leq z$ то значения <i>true false and unknown</i> предикат принимает как и предикат сравнения</p> | | |
| <i>in</i> | <p>в том и только том случае когда значение левого операнда совпадает хотя бы с одним значением списка правого операнда</p> | <p>Если список правого операнда пуст (так может быть если правый операнд задается подзапросом) или значение подразумеваемого предиката сравнения $x = y$ (где x – значение выражения левого операнда) равно <i>false</i> для каждого элемента y списка правого операнда</p> | <p>В противном случае (например так может быть если значение левого операнда есть <i>NULL</i>)</p> |
| <i>like</i> | <p>если значение указанного столбца (<i>column specification</i>) удовлетворяет заданному шаблону (<i>pattern</i>)</p> | <p>Если значение не удовлетворяет условию</p> | <p>если значение столбца либо шаблона не определено.</p> |
| <i>null</i> | <p>значение <i>x IS NULL</i> равно <i>true</i> тогда и только тогда когда значение x не определено</p> | | <p>предикат всегда принимает значения <i>true</i> или <i>false</i>.</p> |

| | | | |
|--|---|--|--|
| <p><i>C</i> квантором</p> <p><i>A) x <comp op> ALL S</i></p> <p><i>Б) x <comp op> SOME S</i></p> | <p><i>А) если S пусто значение предиката x <comp op> s равно true для каждого s входящего в S.</i></p> <p><i>Б) если значение предиката x <comp op> s равно true хотя бы для одного s входящего в S</i></p> | <p><i>А) если значение предиката x <comp op> s равно false хотя бы для одного s входящего в S.</i></p> <p><i>Б) если S пусто или значение предиката x <comp op> s равно false для каждого s входящего в S.</i></p> | <p><i>В остальных случаях</i></p> |
| <p><i>exists</i></p> | <p><i>тогда и только тогда когда результат вычисления подзапроса не пуст.</i></p> | <p><i>Когда результат пуст</i></p> | <p><i>Значением этого предиката всегда является true или false</i></p> |

Логические операторы в логическом выражении включающие значения UNKNOWN будут возвращать UNKNOWN за исключением случаев когда результат оператора не зависит от выражения UNKNOWN.

149. Перечислите и поясните все случаи при которых вычисление табличного выражения приведет к сгруппированной таблице. Приведите примеры.

1) С использованием раздела GROUP BY

Результатом этого раздела является разбиение таблицы на множество групп строк в которых для каждого столбца из списка столбцов раздела GROUP BY во всех строках каждой группы значения этого столбца совпадают.

2) Без раздела GROUP BY но с HAVING

В частности если раздел HAVING присутствует в табличном выражении не содержащем GROUP BY то результатом его выполнения будет либо пустая таблица либо результат выполнения предыдущих разделов табличного выражения рассматриваемый как одна группа без столбцов группирования.

3) Без раздела GROUP BY

Результат вычисления разделов представляет собой сгруппированную таблицу (правильнее сказать псевдосгруппированную) состоящую из одной группы без выделенных столбцов группирования.

150. Какие дополнительные ограничения накладываются на условие поиска раздела HAVING по сравнению с условием поиска раздела WHERE?

Эти ограничения следуют из того что условие поиска раздела HAVING задает условие на целую группу а не на индивидуальные строки.

В выражениях предикатов входящих в условие выборки раздела HAVING прямо можно использовать только спецификации столбцов указанных в качестве группирования в разделе GROUP BY. Остальные столбцы можно использовать только внутри спец агрегатных функций COUNT SUM AVG MIN MAX вычисляющих в данном случае некоторое агрегатное значение для всей группы строк.

Аналогично обстоит дело с подзапросами входящими в предикаты условия выборки раздела HAVING: если в подзапросе используется характеристика текущей группы то она может задаваться только путем ссылки на столбцы группирования.

151. Укажите различные случаи применения агрегатных функций в списке выборки в зависимости от вида табличного выражения. Приведите примеры.

Наиболее часто применяются агрегатные функции SQL SUM MIN MAX AVG COUNT. Следует различать два случая применения агрегатных функций:

1) Агрегатные функции используются сами по себе и возвращают одно результирующее значение.

- Функция SQL SUM возвращает сумму значений столбца таблицы базы данных. Она может применяться только к столбцам значениями которых являются числа. (Пример: SELECT SUM(З/П) FROM(СОТРУДНИК) WHERE(Job='Clerk'). Почитаем зарплату сотрудников с должностью Clerk).

- Функция SQL MIN также действует в отношении столбцов значениями которых являются числа и возвращает минимальное среди всех значений столбца. (Пример: SELECT MIN(З/П) FROM(СОТРУДНИК) WHERE(Dept=42). Узнаем минимальную зарплату сотрудников отдела с номером 42).

- Функция SQL MAX применяется когда требуется определить максимальное значение среди всех значений столбца. (Пример: SELECT MAX(З/П) FROM(СОТРУДНИК) WHERE(Dept=42). Узнаем максимальную зарплату сотрудников отдела с номером 42).

- Функция SQL AVG возвращает среднее значение среди всех значений столбца. (Пример: SELECT AVG(Years) FROM(СОТРУДНИК) WHERE(Dept=42). Можем узнать средний трудовой стаж сотрудников отдела с номером 42).

· Функция SQL COUNT возвращает количество записей таблицы базы данных. (Если в запросе указать SELECT COUNT(ИМЯ_СТОЛБЦА) ... то результатом будет количество записей без учёта тех записей в которых значением столбца является NULL (неопределённое). Если использовать в качестве аргумента звёздочку и начать запрос SELECT COUNT(*) ... то результатом будет количество всех записей (строк) таблицы.). (Пример: SELECT COUNT(Надбавки) FROM(СОТРУДНИК) можем узнать число всех сотрудников которые получают надбавки к зп (вернет Число сотрудников у которых значения столбца Надбавки не NULL) а с запросом SELECT COUNT(*) (Надбавки) FROM(СОТРУДНИК) можем узнать общее количество записей в таблице.)

2) Агрегатные функции используются с оператором SQL GROUP BY то есть с группировкой по полям (столбцам) для получения результирующих значений в каждой группе.

Оператор SQL GROUP BY служит для группировки результирующих значений по столбцам таблицы базы данных. (Пример: можем найти сумму продаж альбомов (Sale) всех исполнителей (Singer): SELECT Singer SUM(Sale) AS AllSales FROM Artists GROUP BY Singer. Оператор SQL AS позволяет задать новое имя столбца на выходе.)