Лекция 3. Предикаты раздела WHERE оператора SELECT

- Введение
- Логические выражения раздела WHERE
  - Предикат сравнения
  - Предикат between
  - Предикат is null
  - Предикат in
  - Предикат like
  - Предикат similar
  - Предикат exists
  - Предикат unique
  - Предикат overlaps
  - Предикат сравнения с квантором
  - Предикат match
  - Предикат is distinct

Введение (1)

- Продолжим рассматривать механизм выборки данных языка SQL – оператор SELECT
- Лекция целиком посвящена видам условных выражений, которые могут содержаться в разделе WHERE оператора выборки
- Определяются и иллюстрируются на примерах запросов все виды предикатов, специфицированных в стандарте SQL:1999.
- Конструкции оператора SELECT языка SQL в значительной степени ортогональны
- В частности, выбор способа указания ссылки на таблицы в разделе FROM никак не влияет на выбор варианта формирования условия выборки в разделе WHERE
- Это полезное свойство языка позволяет нам абстрагироваться от обсуждавшегося в предыдущей лекции многообразия способов указания ссылки на таблицу и
  - сосредоточиться на возможностях формирования запросов при использовании различных предикатов, допускаемых стандартом SQL:1999 в логических выражениях раздела WHERE

Введение (2)

- В стандарте SQL:1999 специфицированы 12 разновидностей предикатов, причем некоторые из них в действительности представляют собой семейства
  - например, под общим названием предиката сравнения скрываются шесть видов предикатов
- Набор допустимых предикатов в SQL явно избыточен, но тем не менее в языке SQL имеется явная тенденция расширения этого набора
- В частности, в SQL:2003 в связи с введением генератора типов мультимножеств в дополнение ко всем разновидностям предикатов SQL:1999 появилось три новых вида предикатов:
  - для проверки того, что заданное значение является элементом мультимножества (MEMBER);
  - что одно мультимножество входит в другое мультимножество (SUBMULTISET) и
  - что мультимножество не содержит дубликаты (IS A SET)

Введение (3)

- В этом курсе мы не приводим подробного описания этих видов предикатов по нескольким причинам:
  - введение конструктора типов мультимножеств в стандарте SQL:2003 не означает, что достигнута общая цель разработчиков стандарта SQL по обеспечению полного набора типов коллекций;
    - по всей видимости, в будущих версиях стандарта появятся дополнительные конструкторы типов коллекций, и набор видов предикатов изменится;
  - предикаты с мультимножествами трудно пояснять и иллюстрировать в отрыве от других объектно-реляционных средств языка SQL;
  - включение подобного материала в данную лекцию заметно увеличило бы ее объем и затруднило понимание более традиционных конструкций

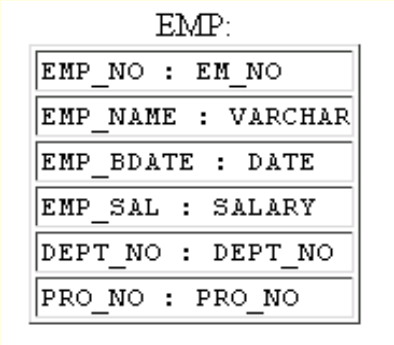
Логические выражения раздела WHERE (1)

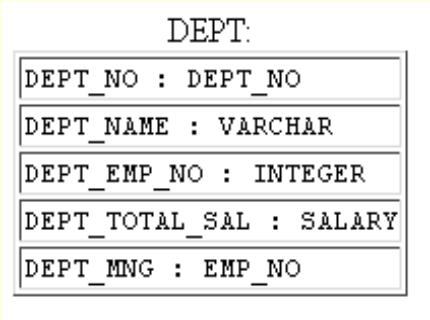
- Синтаксически логическое выражение раздела WHERE определяется как булевское выражение (boolean_value_expression), правила построения которого обсуждались в предыдущей лекции
- Основой логического выражения являются предикаты
- Предикат позволяет специфицировать условие, результатом вычисления которого может быть true, false или unknown
- В языке SQL:1999 допустимы следующие предикаты:

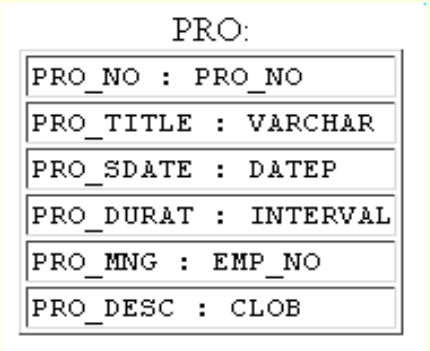
```
predicate ::= comparison_predicate
    | between_predicate
    | null_predicate
    | in_predicate
    | like_predicate
    | similar_predicate
    | exists_predicate
    | unique_predicate
    | overlaps_predicate
    | quantified_comparison_predicate
    | match_predicate
    | distinct_predicate
```

Логические выражения раздела WHERE (2)

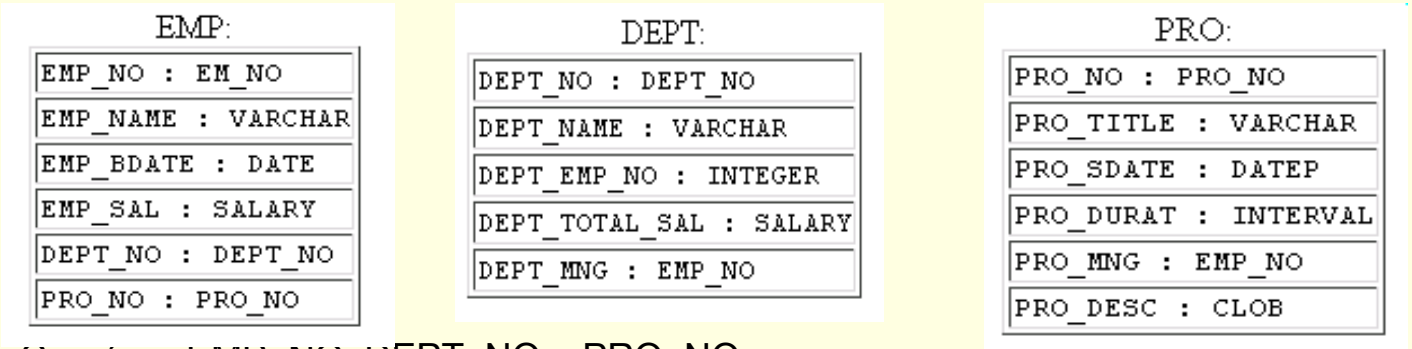
- Далее мы будем последовательно обсуждать разные виды предикатов и приводить примеры запросов с использованием базы данных СЛУЖАЩИЕ-ОТДЕЛЫ-ПРОЕКТЫ, определения таблиц которой на языке SQL были приведены раньше
- Для удобства повторим структуру таблиц







Логические выражения раздела WHERE (3)



- Столбцы EMP_NO, DEPT_NO и PRO_NO являются первичными ключами таблиц EMP, DEPT и PRO соответственно
- Столбцы DEPT_NO и PRO_NO таблицы EMP являются внешними ключами, ссылающимися на таблицы DEPT и PRO соответственно
  - DEPT_NO указывает на отделы, в которых работают служащие, а PRO_NO – на проекты, в которых они участвуют; оба столбца могут принимать неопределенные значения
- Столбец DEPT_MNG является внешним ключом таблицы DEPT
  - DEPT_MNG указывает на служащих, которые исполняют обязанности руководителей отделов; у отдела может не быть руководителя, и один служащий не может быть руководителем двух или более отделов
- Столбец PRO_MNG является внешним ключом таблицы PRO
  - PRO_MNG указывает на служащих, которые являются менеджерами проектов, у проекта всегда есть менеджер, и один служащий не может быть менеджером двух или более проектов

Логические выражения раздела WHERE (4)

- **Предикат сравнения**
- Этот предикат предназначен для спецификации сравнения двух строчных значений
- Синтаксис предиката следующий:

```
comparison_predicate ::=
    row_value_constructor comp_op row_value_constructor
comp_op ::= = | <> («не равно»)| < | >
    | <= («меньше или равно») | >= («больше или равно»)
```

- Строки, являющиеся операндами операции сравнения, должны быть одинаковой степени
- Типы данных соответствующих значений строк-операндов должны быть совместимы

Логические выражения раздела WHERE (5)

- Пусть X и Y обозначают соответствующие элементы строк-операндов, а xv и yv – их значения
- Тогда:
  - если xv и/или yv являются неопределенными значениями, то значение условия X comp_op Y - unknown;
  - в противном случае значением условия X comp_op Y является true или false в соответствии с естественными правилами применения операции сравнения.
- При этом:
  - Числа сравниваются в соответствии с правилами алгебры.
  - Сравнение двух символьных строк производится следующим образом:
    - если длина строки X не равна длине строки Y, то для выравнивания длин строк более короткая строка расширяется *символами набивки* (*pad symbol*);
    - если для используемого набора символов порядок сортировки явным образом не специфицирован, то в качестве символа набивки используется пробел;
    - далее производится лексикографическое сравнение строк в соответствии с предопределенным или явно определенным порядком сортировки символов

Логические выражения раздела WHERE (6)

- Сравнение двух битовых строк X и Y основано на сравнении соответствующих бит
- Если X*i* и Y*i* – значения *i*-тых бит X и Y соответственно и если lx и ly обозначает длину в битах X и Y соответственно, то:
  - X равно Y тогда и только тогда, когда lx = ly и X*i* = Y*i* для всех *i*;
  - X меньше Y тогда и только тогда, когда (a) lx < ly и X*i* = Y*i* для всех *i* меньших или равных lx, или (b) X*i* = Y*i* для всех *i* < *n* и X*n* = 0, а Y*n* =1 для некоторого *n* меньшего или равного min (lx, ly).
- Сравнение двух значений типа дата-время производится в соответствии с видом интервала, который получается при вычитании второго значения из первого
- Пусть X и Y – сравниваемые значения, а H – наименее значимое поле даты-времени X и Y
- Результат сравнения X comp_op Y определяется как (X – Y) H comp_ op INTERVAL (0) H
  - Два значения типа дата-время сравнимы только в том случае, если они содержат одинаковый набор полей даты-времени

Логические выражения раздела WHERE (7)

- Сравнение двух значений анонимного строкового типа производится следующим образом
- Пусть Rx и Ry обозначают строки-операнды, а Rx*i* и Ry*i* – i-тые элементы Rx и Ry соответственно
- Вот как определяется результат сравнения Rx comp_op Ry:
  - Rx = Ry есть true тогда и только тогда, когда Rx*i* = Ry*i* есть true для всех *i*;
  - Rx <> Ry есть true тогда и только тогда, когда Rx*i* <> Ry*i* есть true для некоторого *i*;
  - Rx < Ry есть true тогда и только тогда, когда Rx*i* = Ry*i* есть true для всех *i* < *n*, и Rx*n* < Ry*n* есть true для некоторого *n*;
  - Rx > Ry есть true тогда и только тогда, когда Rx*i* = Ry*i* есть true для всех *i* < *n*, и Rx*n* > Ry*n* есть true для некоторого *n*;
  - Rx <= Ry есть true тогда и только тогда, когда Rx = Ry есть true или Rx < Ry есть true;
  - Rx >= Ry есть true тогда и только тогда, когда Rx = Ry есть true или Rx > Ry есть true;

Логические выражения раздела WHERE (8)

- Rx = Ry есть false тогда и только тогда, когда Rx <> Ry есть true;
- Rx <> Ry есть false тогда и только тогда, когда Rx = Ry есть true;
- Rx < Ry есть false тогда и только тогда, когда Rx >= Ry есть true;
- Rx > Ry есть false тогда и только тогда, когда Rx <= Ry есть true;
- Rx <= Ry есть false тогда и только тогда, когда Rx > Ry есть true;
- Rx >= Ry есть false тогда и только тогда, когда Rx < Ry есть true;
- Rx comp_op Ry есть unknown тогда и только тогда, когда Rx comp_op Ry не есть true или false

Логические выражения раздела WHERE (9)

- **Примеры запросов с использованием предиката сравнения**
- Найти номера отделов, в которых работают служащие с фамилией 'Smith‘

```
SELECT DISTINCT EMP.DEPT_NO
FROM EMP
WHERE EMP.EMP_NAME = 'Smith';
```

- Добавили спецификацию DISTINCT в раздел SELECT, потому что в одном отделе могут работать несколько служащих с фамилией 'Smith', а их число нас в данном случае не интересует

Логические выражения раздела WHERE (10)

- Если бы нас интересовало число служащих с фамилией 'Smith' в каждом отделе, где такие служащие работают, то следовало бы, например, написать такой запрос:

```
SELECT EMP.DEPT_NO, COUNT(*)
FROM EMP
WHERE EMP.NAME = 'Smith'
GROUP BY EMP.DEPT_NO;
```

- В этом варианте запроса спецификация DISTINCT не требуется, поскольку
  - в запросе содержится раздел GROUP BY,
  - группировка производится в соответствии со значениями столбца EMP.DEPT_NO, и
  - строка результата соответствует одной группе