MySQL

Занятие 4

План занятия

- **EXISTS**
- **ANY, ALL**
- **Особенности NULL значений**

EXISTS

- TRUE в случае *существования (возврата) хотя бы* одного найденного значения.
- В противном случае результат подзапроса – FALSE
- **EXISTS не может принимать значение unknown** (неизвестно).
- в подзапросе, указываемом в операторе **EXISTS, *нельзя использовать агрегирующие функции (они всегда что-то возвращают).***

EXISTS

- Пример: извлечь из таблицы EXAM_MARKS данные о студентах, получивших хотя бы одну неудовлетворительную оценку.

```
SELECT DISTINCT STUDENT_ID
FROM EXAM_MARKS A
WHERE EXISTS
    ( SELECT *
      FROM EXAM_MARKS B
      WHERE MARK < 3
      AND B.STUDENT_ID=A.STUDENT_ID);
```

Почему нельзя использовать агрегирующие функции.

```
SELECT count(DISTINCT STUDENT_ID)
FROM EXAM_MARKS A
WHERE EXISTS
   ( SELECT count(*)
   FROM EXAM_MARKS B
   WHERE MARK < 3        AND
   B.STUDENT_ID=A.STUDENT_ID);
```

Вернет количество всех строк в student

```
SELECT count(*) FROM student;
```

EXISTS

- Пример: требуется получить идентификаторы предметов обучения, экзамены по которым сдавались не одним, а несколькими студентами:

```
SELECT DISTINCT SUBJ_ID
FROM EXAM_MARKS A
WHERE EXISTS
    ( SELECT *
      FROM EXAM_MARKS B
      WHERE A.SUBJ_ID = B.SUBJ_ID
      AND A.STUDENT_ID <> B.STUDENT_ID);
```

EXISTS

- Пример: из таблицы STUDENT требуется извлечь строки для каждого студента, сдавшего более одного предмета.

```
SELECT * FROM student
WHERE STUDENT_ID IN
    (SELECT distinct(STUDENT_ID)
    FROM EXAM_MARKS FIRST
    WHERE EXISTS
        (SELECT SUBJ_ID
        FROM EXAM_MARKS SECOND
        WHERE FIRST.STUDENT_ID = SECOND.STUDENT_ID
            AND FIRST.SUBJ_ID <> SECOND.SUBJ_ID));
```

Задание 1

- Напишите запрос с **EXISTS**, выбирающий сведения обо всех студентах, для которых в том же городе, где живет студент, существуют университеты, в которых он не учится.

Задание 2

- Напишите запрос, выбирающий из таблицы SUBJECT названия предметов обучения, экзамены по которым сданы более чем одним студентом.

Операторы сравнения с множеством значений IN, ANY(SOME), ALL

- *operand comparison_operator* ANY (*subquery*)
- *operand* IN (*subquery*)
- *operand comparison_operator* ALL (*subquery*)

- *comparison_operator:* = > < >= <= <> !=

IN, ANY

Выбрать сведения о студентах, проживающих в городе, где расположен университет, в котором они учатся.

```
SELECT *
FROM STUDENT S
WHERE CITY = ANY
      ( SELECT CITY
      FROM UNIVERSITY U
      WHERE U.UNIV_ID= S.UNIV_ID);
```

Другой вариант этого запроса

```
SELECT *
FROM STUDENT S
WHERE CITY IN
      (SELECT CITY
      FROM UNIVERSITY U
      WHERE U.UNIV_ID= S.UNIV_ID);
```

IN, ANY

Выборка данных об идентификаторах студентов , у которых оценки превосходят величину, по крайней мере, одной из оценок, полученных ими же 10 января 2000 года.

```
SELECT DISTINCT STUDENT_ID
FROM EXAM_MARKS
WHERE MARK > ANY
    (SELECT MARK
    FROM EXAM_MARKS
    WHERE EXAM_DATE = "2000-01-10");
```

Задание 3

- Напишите запрос, выбирающий названия университетов , рейтинг которых равен или превосходит рейтинг Воронежского государственного университета .

ALL

Оператор **ALL** эффективно используется с неравенствами

- Подзапрос, выбирающий названия всех университетов с рейтингом более высоким, чем рейтинг любого университета в Воронеже:

```
SELECT *
FROM UNIVERSITY
WHERE RATING > ALL
    ( SELECT RATING
    FROM UNIVERSITY
    WHERE CITY = "Воронеж");
```

?Как в этом запросе в место ALL можно использовать ANY?

Задание 4

- Напишите запрос, использующий ALL, выполняющий выборку данных о студентах, у которых в городе их постоянного местожительства нет университета.
- Напишите такой же запрос с использованием ANY.

Особенности применения ANY, ALL, EXISTS при обработке пустых значений (NULL)

- Когда подзапрос возвращает NULL:
  - **ALL автоматически принимает TRUE**
  - **ANY автоматически принимает FALSE**

ANY

Запрос

```
SELECT *
FROM UNIVERSITY
WHERE RATING > ANY
    ( SELECT RATING
    FROM UNIVERSITY
    WHERE CITY = "New York");
```

Вернет пусто

ALL

```
SELECT *
FROM UNIVERSITY
WHERE RATING > ALL
   ( SELECT RATING
   FROM UNIVERSITY
   WHERE CITY = "New York");
```

полностью воспроизведет таблицу UNIVERSITY.

Особенности использования NULL значений

1) **SELECT ***
FROM UNIVERSITY
WHERE NOT RATING >= ANY
(SELECT RATING
FROM UNIVERSITY
WHERE CITY = "Москва");

2) **SELECT ***
FROM UNIVERSITY A
WHERE NOT EXISTS
(SELECT *
FROM UNIVERSITY B
WHERE A.RATING >= B.RATING
AND B.CITY = "Москва");

EXISTS всегда принимает значения истина или ложь, и никогда – UNKNOWN. Это является доводом для использования в таких случаях оператора **ANY вместо EXISTS**.

использование COUNT вместо EXISTS

```
SELECT *
FROM UNIVERSITY A
WHERE NOT EXISTS
    (SELECT *
    FROM UNIVERSITY B
    WHERE A.RATING > =
    B.RATING
          AND B.CITY =
"Москва");
```

```
SELECT *
FROM UNIVERSITY A
WHERE 1 >
          (SELECT
          COUNT(*)
          FROM UNIVERSITY B
          WHERE A.RATING >=
B.RATING AND B.CITY =
"Москва");
```

Задание 5

- Напишите запрос, выбирающий из таблицы EXAM_MARKS названия предметов обучения, для которых значение полученных на экзамене 10 января 2000 года оценок (поле MARK) превышает любое значение оценки (в тот же день) для предмета, имеющего идентификатор равный 5.
- Напишите этот же запрос с использованием **MAX.**