1. **Использование оператора EXISTS**

Нужно 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 В

WHERE MARK < 3

AND B.STUDENT\_ID = A.STUDENT\_ID);

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 В

WHERE A.SUBJ\_\_ID = B.SUBJ\_ID

AND A.STUDENT\_ID < > В.STUDENT\_ID);

Из таблицы STUDENT требуется извлечь строки для каждого студента, сдавшего более одного предмета.

SELECT \*

FROM STUDENT FIRST

WHERE EXISTS

(SELECT SUBJ\_ID

FROM EXAM\_MARKS SECOND

GROUP BY SUBJ\_ID

HAVING COUNT (SUBJ\_ID) > 1

WHERE FIRST.STUDENT ID = SECOND.STUDENT ID);

**2.Операторы сравнения с множеством значений**

**IN, ANY, All**

Операторы сравнения с множеством значений имеют следующий смысл.

|  |  |
| --- | --- |
| IN | Равно любому из значений, полученных во внутреннем запросе. |
| NOT IN | Не равно ни одному из значений, полученных во внутреннем запросе. |
| = ANY | То же, что и IN, соответствует логическому оператору OR. |
| > ANY, > = ANY | Больше, чем (либо больше или равно) любое полученное число. Эквивалентно > или > = для самого меньшего полученного числа. |
| < ANY, < = ANY | Меньше, чем (либо меньше или равно) любое полученное число. |
| =ALL | Равно всем полученным значениям. Эквивалентно логическому оператору AND |
| >ALL, >=ALL | Больше, чем (либо больше или равно) все полученные числа. Эквивалент > или > = для самого  большего полученного числа. |
| <ALL, <=ALL | Меньше, чем (либо меньше или равно) все полученные числа. Эквивалентно < или < = самого  меньшего полученного числа. |

*Следует иметь в виду, что в некоторых СУБД поддерживаются не все из этих операторов!*

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);

Оператор ALL, как правило, эффективно используется с неравенствами, а не с равенствами, поскольку значение равно всем, которое должно получиться в этом случае в результате выполнения подзапроса, может иметь место, только если все результаты идентичны.

Такая ситуация практически не может быть реализована,

*так как если подзапрос генерирует множество различных значений, то никакое одно значение не может быть равно сразу всем значениям в обычном смысле.*

В SQL выражение < > ALL реально означает не равно ни одному из результатов подзапроса.

Подзапрос, выбирающий данные о названиях всех университетов с рейтингом более высоким, чем рейтинг любого университета Воронежа:

SELECT \*

FROM UNIVERSITY

WHERE RATING > ALL

(SELECT RATING

FROM UNIVERSITY

WHERE CITY = 'Воронеж');

В этом запросе вместо ALL можно использовать ANY (проанализируйте, как в этом случае изменится смысл приведенного запроса):

SELECT \*

FROM UNIVERSITY

WHERE NOT RATING <= ANY

(SELECT RATING

FROM UNIVERSITY

WHERE CITY = 'Воронеж');

**3.Особенности применения операторов ANY, ALL, EXISTS для NULL-значений**

Необходимо иметь в виду, что при обработке NULL-значений следует учитывать различие реакции на них операторов EXISTS, ANY и ALL.

Когда правильный подзапрос не генерирует никаких выходных данных, оператор ALL автоматически принимает значение истина, а оператор ANY — значение ложь.

Запрос

SELECT \*

FROM UNIVERSITY

WHERE RATING > ANY

(SELECT RATING

FROM UNIVERSITY

WHERE CITY = 'New York');

не генерирует выходных данных, в то время как запрос

SELECT \*

FROM UNIVERSITY

WHERE RATING > ALL

(SELECT RATING

FROM UNIVERSITY

WHERE CITY = 'New York');

полностью воспроизведет таблицу UNIVERSITY.

Использование NULL-значений создает определенные проблемы для рассматриваемых операторов. Когда в SQL сравниваются два значения, одно из которых NULL-значение, результат принимает значение UNKNOWN (неизвестно).

Предикат UNKNOWN, так же, как и FALSE-предикат, создает ситуацию, когда строка не включается в состав выходных данных, но результат при этом будет различен для разных типов запросов, в зависимости от использования в них ALL или ANY вместо EXISTS. Рассмотрим в качестве примера две реализации запроса:

Найти все данные об университетах, рейтинг которых меньше рейтинга любого университета в Москве.

1) SELECT \*

FROM UNIVERSITY

WHERE RATING < ANY

(SELECT RATING

FROM UNIVERSITY

WHERE CITY = 'Москва');

2) SELECT \*

FROM UNIVERSITY A

WHERE NOT EXISTS

(SELECT \*

FROM UNIVERSITY В

WHERE A.RATING >= B.RATING

AND B.CITY = 'Москва');

При отсутствии в таблицах NULL оба эти запроса ведут себя совершенно одинаково. Пусть теперь в таблице UNIVERSITY есть строка с NULL-значениями в столбце RATING. В версии запроса с ANY в основном запросе, когда выбирается поле RATING с NULL, предикат принимает значение UNKNOWN и строка не включается в состав выходных данных. Во втором же варианте запроса, когда NOT EXISTS выбирает эту строку в основном запросе, NULL-значение используется в предикате подзапроса, присваивая ему значение UNKNOWN. Поэтому в результате выполнения подзапроса не будет получено ни одного значения, и подзапрос примет значение ложь. Это в свою очередь сделает NOT EXISTS истинным, и, следовательно, строка с NULL-значением в поле RATING попадет в выходные данные. По смыслу запроса такой результат является неправильным, так как на самом деле рейтинг университета, описываемого данной строкой, может

быть и больше рейтинга какого-либо московского университета (он просто неизвестен). Указанная проблема связана с тем, что значение EXISTS всегда принимает значения истина или ложь, и никогда — UNKNOWN .

Это является доводом для использования в таких случаях оператора ANY вместо EXISTS.