университеты, в которых он не учится.

36. Напишите запрос, выбирающий из таблицы SUBJECT данные о названиях предметов обучения, экзамены по которым *сданы* более чем одним студентом.

2.12. 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. Эквивалент < или < = для самого большего полученного числа.
= 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.

Примеры запросов с использованием приведенных 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);
```

Выборка данных об идентификаторах студентов, у которых оценки превосходят величину, по крайней мере, одной из оценок, полученных ими же 6 октября 1999 года.

```
SELECT DISTINCT STUDENT_ID
    FROM EXAM_MARKS
    WHERE MARK > ANY
        (SELECT MARK
              FROM EXAM_MARKS
              WHERE EXAM_DATE = '06/10/1999');
```

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 в Воронеже:

```
SELECT *
    FROM UNIVERSITY
    WHERE RATING > ALL
```

```
        ( SELECT RATING
          FROM UNIVERSITY
          WHERE CITY = ‘Воронеж’);
```

В этом запросе вместо **ALL** можно также 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’);
```

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

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 (подразумевается, что в базе нет данных об университетах из города New York), 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**, также 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 B
          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**.

2.14. Использование COUNT вместо EXISTS

При отсутствии **NULL**-значений оператор **EXISTS** может быть использован вместо **ANY** и **ALL**. Также вместо **EXISTS** и **NOT EXISTS** могут быть использованы те же самые подзапросы, но с использованием **COUNT**(*) в предложении **SELECT**. Например, запрос

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

может быть представлен и в следующем виде

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

УПРАЖНЕНИЯ

37. Напишите запрос, выбирающий данные о названиях университетов, рейтинг которых равен или превосходит рейтинг Воронежского государственного университета.
38. Напишите запрос, использующий **ANY** или **ALL**, выполняющий выборку данных о студентах, у которых в городе их постоянного местожительства нет университета.
39. Напишите запрос, выбирающий из таблицы EXAM_MARKS данные о названиях предметов обучения, для которых значение полученных на экзамене оценок (поле MARK) превышает любое значение оценки для предмета, имеющего идентификатор равный 105.
40. Напишите этот же запрос с использованием **MAX**.

2.15. Оператор объединения UNION

Оператор **UNION** используется для объединения выходных данных двух или более SQL-запросов в единое множество строк и столбцов. Например, для того, чтобы получить в одной таблице фамилии и идентификаторы студентов и преподавателей из Москвы, можно использовать следующий запрос.

```
SELECT 'Студент_____', SURNAME, STUDENT_ID
    FROM STUDENT
    WHERE CITY = 'Москва'
UNION
SELECT 'Преподаватель', SURNAME, LECTURER_ID
    FROM LECTURER
    WHERE CITY = 'Москва';
```

Обратите внимание на то, что символом ";" (точка с запятой) оканчивается только последний запрос. Отсутствие этого символа в конце **SELECT-**запроса означает, что следующий за ним запрос также, как и он сам, является частью общего запроса с **UNION**.

Использование оператора **UNION** возможно только при объединении