**1.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 В

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 В

WHERE A.RATING > = B.RATING AND B.CITY = 'Москва');

**2.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 «;» оканчивается только последний запрос. Отсутствие этого символа в конце SELECT-запроса означает, что следующий за ним запрос так же, как и он сам, является частью общего запроса с UNION.

Использование оператора UNION возможно только при объединении запросов, соответствующие столбцы которых *совместимы по объединению*, то есть соответствующие числовые поля должны *иметь полностью совпадающие тип и размер,*

символьные поля должны иметь точно *совпадающее количество символов*. Если NULL-значения запрещены для столбца хотя бы одного любого подзапроса объединения, то они должны быть запрещены и для всех соответствующих столбцов в других подзапросах объединения.

**3.Устранение дублирования в UNION**

В отличие от обычных запросов UNION автоматически исключает из выходных данных дубликаты строк, например, в запросе

SELECT CITY

FROM STUDENT

UNION

SELECT CITY

FROM LECTURER;

совпадающие наименования городов будут исключены.

Если все же необходимо в каждом запросе вывести все строки независимо от того, имеются ли такие же строки в других объединяемых запросах, то следует использовать во множественном запросе конструкцию с оператором UNION ALL.

Так, в запросе

SELECT CITY

FROM STUDENT

UNION ALL

SELECT CITY

FROM LECTURER;

дубликаты значений городов, выводимые второй частью запроса, не будут исключаться.

Приведем еще один пример использования оператора UNION.

Пусть необходимо составить отчет, содержащий для каждой даты сдачи экзаменов сведения по каждому студенту, получившему

максимальную или минимальную оценки.

SELECT 'МАКСОЦ', A.STUDENT\_ID, SURNAME, MARK, EXAM\_DATE

FROM STUDENT A, EXAM\_MARKS В

WHERE (A.STUDENT\_ID = B.STUDENT\_ID

AND B.MARK =

(SELECT MAX (MARK)

FROM EXAM\_MARKS С

WHERE C.EXAM\_DATE = B.EXAM\_DATE))

UNION ALL

SELECT 'МИНОЦ’, A.STUDENT\_ID, SURNAME, MARK, EXAM\_DATE

FROM STUDENT A, EXAM\_MARKS В

WHERE (A.STUDENT\_ID = B.STUDENT\_ID

AND B.MARK =

(SELECT MIN (MARK)

FROM EXAM\_MARKS С

WHERE C.EXAM\_DATE = В.EXAM\_DATE));

Для отличия строк, выводимых первой и второй частями запроса, в них вставлены текстовые константы 'Макс оц' и 'Мин оц'.

В приведенном запросе агрегирующие функции используются в подзапросах. Это является нерациональным с точки зрения времени, затрачиваемого на выполнение запроса. Более эффективна форма запроса, возвращающего аналогичный результат:

SELECT 'МаКСОЦ', A.STUDENT\_ID, SURNAME, E.MARK, E.EXAM\_DATE

FROM STUDENT A,

(SELECT B.STUDENT\_ID, B.MARK, B.EXAM\_DATE

FROM EXAM\_MARKS B,

(SELECT MAX (MARK) AS MAX\_MARK, C.EXAM\_DATE

FROM EXAM\_\_MARKS С

GROUP BY C.EXAM\_DATE) D

WHERE В.EXAM\_DATE=D.EXAM\_DATE

AND B.MARK=MAX\_MARK) E

WHERE A.STUDENT\_ID=E.STUDENT\_ID

UNION ALL

SELECT 'МИНОЦ', A.STUDENT\_\_ID, SURNAME, E.MARK, E .EXAM\_DATE

FROM STUDENT A,

(SELECT B.STUDENT\_ID, B.MARK, B.EXAM\_DATE

FROM EXAM\_MARKS B,

(SELECT MIN (MARK) AS MIN\_MARK, C.EXAM\_DATE

FROM EXAM\_MARKS С

GROUP BY C.EXAM\_DATE) D

WHERE B.EXAM\_DATE=D.EXAM\_DATE

AND B.MARK=MIN\_MARK) E

WHERE A.STUDENT\_ID=E.STUDENT\_ID;

**3.Использование UNION с ORDER BY**

Предложение ORDER BY применяется для упорядочения выходных данных объединения запросов так же, как и для отдельных запросов. Последний пример, при необходимости упорядочения выходных данных запроса по фамилиям студентов и датам

экзаменов, может выглядеть следующим образом:

SELECT 'МАКСОЦ', A.STUDENT\_ID, SURNAME, E.MARK, E.EXAM\_DATE

FROM STUDENT A,

(SELECT B.STUDENT\_ID, B.MARK, B.EXAM\_DATE

FROM EXAM\_MARKS B,

(SELECT MAX (MARK) AS MAX\_MARK, C.EXAM\_DATE

FROM EXAM\_MARKS С

GROUP BY C.EXAM\_DATE) D

WHERE B.EXAM\_DATE=D.EXAM\_DATE

AND B.MARK=MAX\_MARK) E

WHERE A.STUDENT\_ID=E.STUDENT\_ID

UNION ALL

SELECT 'МИНОЦ', A.STUDENT\_ID, SURNAME, E.MARK, E.EXAM\_DATE

FROM STUDENT A,

(SELECT B.STUDENT\_ID, B.MARK, B.EXAM\_DATE

FROM EXAM\_MARKS B,

(SELECT MIN (MARK) AS MIN\_MARK, C.EXAM\_DATE

FROM EXAM\_MARKS С

GROUP BY C.EXAM\_DATE) D

WHERE В.EXAM\_DATE=D.EXAM\_DATE

AND B.MARK=MIN\_MARK) E

WHERE A.STUDENT\_ID=E.STUDENT\_ID

ORDER BY SURNAME, E.EXAM\_DATE;