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

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.

2.16. 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 'макс оц', A.STUDENT_ID, SURNAME, MARK, EXAM_DATE
```

```
    FROM STUDENT A, EXAM_MARKS B
    WHERE (A.STUDENT_ID = B.STUDENT_ID
        AND B.MARK =
            (SELECT MAX(MARK)
            FROM EXAM_MARKS C
            WHERE C.EXAM_DATE = B.EXAM_DATE))
UNION ALL
SELECT 'мин оц ', A.STUDENT_ID, SURNAME, MARK, EXAM_DATE
    FROM STUDENT A, EXAM_MARKS B
        WHERE (A.STUDENT_ID = B.STUDENT_ID
            AND B.MARK =
                (SELECT MIN(MARK)
                FROM EXAM_MARKS C
                WHERE C.EXAM_DATE = B.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 (см. раздел 2.9). Более эффективна форма запроса, возвращающего аналогичный результат:

```
SELECT 'макс оц', 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 C
                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 'мин оц ', 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 C
                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
```

2.17. 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 так:

```
SELECT 'макс оц', 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 C
                    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 'мин оц ', 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 C
                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
ORDER BY SURNAME, E.EXAM_DATE;
```

2.18. Внешнее объединение

Часто полезна операция объединения двух запросов, в которой второй запрос выбирает строки, исключенные первым. Такая операция называется внешним объединением.

Рассмотрим пример. Пусть в таблице STUDENT имеются записи о студентах, в которых не указан идентификатор университета. Требуется составить список студентов с указанием наименования университета для тех студентов, у которых эти данные есть, но при этом не отбрасывая и студентов, у которых университет не указан. Можно получить желаемые сведения, сформировав объединение двух запросов, один из которых выполняет выборку студентов с названиями их университетов, а второй выбирает студентов с **NULL**-значениями в поле UNIV_ID. В данном случае оказывается полезной возможность вставки в запрос констант, в нашем случае текстовой константы ‘не известен’, чтобы отметить в списке тех студентов, у которых отсутствует информация об университете.

```
SELECT SURNAME, NAME, UNIV_NAME
    FROM STUDENT, UNIVERSITY
    WHERE STUDENT.UNIV_ID = UNIVERSITY.UNIV_ID
UNION
SELECT SURNAME, NAME, ‘не известен    ’
    FROM STUDENT
    WHERE UNIV_ID IS NULL
ORDER BY 1;
```

Для совместимости столбцов объединяемых запросов константу ‘не известен’ во втором запросе следует дополнить пробелами так, чтобы ее длина соответствовала длине поля UNIV_NAME или использовать для согласования типов функцию **CAST**. В некоторых СУБД согласование типов поля и замещающей его текстовой константы осуществляется автоматически.

УПРАЖНЕНИЯ

41. Создайте объединение двух запросов, которые выдают значения полей UNIV_NAME, CITY, RATING для всех университетов. Те из них, у которых рейтинг равен или выше 300, должны иметь комментарий ‘Высокий’, все остальные – ‘Низкий’.

42. Напишите команду, которая выдает список фамилий студентов, с комментарием ‘успевает’ у студентов, имеющих все положительные оценки, комментарием ‘не успевает’ для сдававших экзамены, но имеющих хотя бы одну неудовлетворительную оценку, и комментарием ‘не сдавал’ – для всех остальных. В выводимом результате фамилии студентов упорядочить по алфавиту.

43. Выведите объединенный список студентов и преподавателей, живущих в Москве, с соответствующими комментариями ‘студент’ или ‘преподаватель’.

44. Выведите объединенный список студентов и преподавателей Воронежского государственного университета с соответствующими комментариями ‘студент’ или ‘преподаватель’.

2.19. Соединение таблиц с использованием оператора JOIN

Если в операторе **SELECT** после ключевого слова **FROM** указывается не одна, а две таблицы, то в результате выполнения запроса, в котором отсутствует предложение **WHERE**, каждая строка одной таблицы будет соединена с каждой строкой второй таблицы. Такая операция называется *декартовым произведением* или *полным* (**CROSS**) *соединением* таблиц базы данных. Сама по себе эта операция не имеет практического значения, более того, при ошибочном использовании она может привести к неожиданным нештатным ситуациям, так как в этом случае в ответе на запрос количество записей будет равно произведению числа записей в соединяемых таблицах, то есть может оказаться чрезвычайно большим. Соединение таблиц имеет смысл тогда, когда соединяются *не все* строки исходных таблиц, а только те, которые интересуют пользователя. Такое ограничение может быть осуществлено с помощью использования в запросе соответствующего условия в предложении **WHERE**. Таким образом, SQL позволяет выводить информацию из нескольких таблиц, связывая их по значениям определенных полей.

Например, если необходимо получить фамилии студентов (таблица STUDENT) и для каждого студента – названия университетов (таблица