

2. Этот запрос выбирает из таблицы «Билеты» (tickets) всех пассажиров с именами, состоящими из трех букв (в шаблоне присутствуют три символа «_»):

```
SELECT passenger_name
FROM tickets
WHERE passenger_name LIKE '___ %';
```

Предложите шаблон поиска в операторе LIKE для выбора из этой таблицы всех пассажиров с фамилиями, состоящими из пяти букв.

```
demo=# SELECT passenger_name
FROM tickets
WHERE passenger_name LIKE '% _____' LIMIT 3;
 passenger_name
-----
 ILYA POPOV
VLADIMIR POPOV
PAVEL GUSEV
(3 rows)
```

7. Самые крупные самолеты в нашей авиакомпании — это Boeing 777-300. Выяснить, между какими парами городов они летают, поможет запрос:

```
SELECT DISTINCT departure_city, arrival_city
FROM routes r
JOIN aircrafts a ON r.aircraft_code = a.aircraft_code
WHERE a.model = 'Boeing 777-300'
ORDER BY 1;
```

departure_city	arrival_city
Екатеринбург	Москва
Москва	Екатеринбург
Москва	Новосибирск
Москва	Пермь
Москва	Сочи
Новосибирск	Москва
Пермь	Москва
Сочи	Москва

(8 строк)

К сожалению, в этой выборке информация дублируется. Пары городов приведены по два раза: для рейса «туда» и для рейса «обратно». Модифицируйте запрос таким образом, чтобы каждая пара городов была выведена только один раз:

departure_city	arrival_city
Москва	Екатеринбург
Новосибирск	Москва
Пермь	Москва
Сочи	Москва

(4 строки)

```
demo=# SELECT DISTINCT departure_city, arrival_city
FROM routes r
JOIN aircrafts a ON r.aircraft_code = a.aircraft_code
WHERE a.model = 'Boeing 777-300' and departure_city>arrival_city
ORDER BY 1;
```

departure_city	arrival_city
Москва	Екатеринбург
Новосибирск	Москва
Пермь	Москва
Сочи	Москва

(4 rows)

9. Для ответа на вопрос, сколько рейсов выполняется из Москвы в Санкт-Петербург, можно написать совсем простой запрос:

```
SELECT count( * )
FROM routes
WHERE departure_city = 'Москва'
AND arrival_city = 'Санкт-Петербург';
```

count
12

(1 строка)

А с помощью какого запроса можно получить результат в таком виде?

departure_city	arrival_city	count
Москва	Санкт-Петербург	12

(1 строка)

```
demo=# SELECT departure_city, arrival_city, count( * )
FROM routes
WHERE departure_city = 'Москва'
AND arrival_city = 'Санкт-Петербург' GROUP BY departure_city, arrival_city;
```

departure_city	arrival_city	count
Москва	Санкт-Петербург	12

(1 row)

13. Ответить на вопрос о том, каковы максимальные и минимальные цены билетов на все направления, может такой запрос:

```
SELECT f.departure_city, f.arrival_city,
       max( tf.amount ), min( tf.amount )
FROM flights_v f
JOIN ticket_flights tf ON f.flight_id = tf.flight_id
GROUP BY 1, 2
ORDER BY 1, 2;
```

departure_city	arrival_city	max	min
Абакан	Москва	101000.00	33700.00
Абакан	Новосибирск	5800.00	5800.00
Абакан	Томск	4900.00	4900.00
Анадырь	Москва	185300.00	61800.00
Анадырь	Хабаровск	92200.00	30700.00
...			
Якутск	Мирный	8900.00	8100.00
Якутск	Санкт-Петербург	145300.00	48400.00

(367 строк)

А как выявить те направления, на которые не было продано ни одного билета? Один из вариантов решения такой: если на рейсы, отправляющиеся по какому-то направлению, не было продано ни одного билета, то максимальная и минимальная цены будут равны NULL. Нужно получить выборку в таком виде:

departure_city	arrival_city	max	min
Абакан	Архангельск		
Абакан	Грозный		
Абакан	Кызыл		
Абакан	Москва	101000.00	33700.00
Абакан	Новосибирск	5800.00	5800.00
...			

Модифицируйте запрос, приведенный выше.

```
demo=# SELECT f.departure_city, f.arrival_city,
max( tf.amount ), min( tf.amount )
FROM flights_v f
FULL OUTER JOIN ticket_flights tf ON f.flight_id = tf.flight_id
GROUP BY 1, 2
ORDER BY 1, 2 NULLS FIRST
demo=# LIMIT 10;
```

departure_city	arrival_city	max	min
Абакан	Архангельск		
Абакан	Грозный		
Абакан	Кызыл		
Абакан	Москва	101000.00	33700.00
Абакан	Новосибирск	5800.00	5800.00
Абакан	Томск	4900.00	4900.00
Анадырь	Москва	185300.00	61800.00
Анадырь	Хабаровск	92200.00	30700.00
Анапа	Белгород	18900.00	6300.00
Анапа	Москва	36600.00	12200.00

(10 rows)

- 19.* В разделе 6.4 мы использовали рекурсивный алгоритм в общем табличном выражении. Изучите этот пример, чтобы лучше понять работу рекурсивного алгоритма:

```
WITH RECURSIVE ranges ( min_sum, max_sum )
AS (
    VALUES( 0,      100000 ),
           ( 100000, 200000 ),
           ( 200000, 300000 )
    UNION ALL
    SELECT min_sum + 100000, max_sum + 100000
    FROM ranges
    WHERE max_sum < ( SELECT max( total_amount ) FROM bookings )
)
SELECT * FROM ranges;
```

min_sum	max_sum	
-----+-----		
0	100000	исходные строки
100000	200000	
200000	300000	
100000	200000	результат первой итерации
200000	300000	
300000	400000	
200000	300000	результат второй итерации
300000	400000	
400000	500000	
300000	400000	
400000	500000	
500000	600000	
...		
1000000	1100000	результат (n-3)-й итерации
1100000	1200000	
1200000	1300000	
1100000	1200000	результат (n-2)-й итерации
1200000	1300000	
1200000	1300000	результат (n-1)-й итерации (предпоследней)

(36 строк)

Здесь мы с помощью предложения VALUES специально создали виртуальную таблицу из трех строк, хотя для получения требуемого результата достаточно только одной строки (0, 100000). Еще важно то, что предложение UNION ALL не удаляет строки-дубликаты, поэтому мы можем видеть весь рекурсивный процесс порождения новых строк.

При рекурсивном выполнении запроса

```
SELECT min_sum + 100000, max_sum + 100000
FROM ranges
WHERE max_sum < ( SELECT max( total_amount ) FROM bookings )
```

каждый раз выполняется проверка в условии WHERE. И на $(n - 2)$ -й итерации это условие отсеивает одну строку, т. к. после $(n - 3)$ -й итерации значение атрибута max_sum в третьей строке было равно 1 300 000.

Ведь запрос

```
SELECT max( total_amount ) FROM bookings;
```

выдаст значение

```
      max
-----
1204500.00
(1 строка)
```

Таким образом, после $(n - 2)$ -й итерации во временной области остается всего две строки, после $(n - 1)$ -й итерации во временной области остается только одна строка.

Заключительная итерация уже не добавляет строк в результирующую таблицу, поскольку единственная строка, поданная на вход команде SELECT, будет отклонена условием WHERE. Работа алгоритма завершается.

Задание 1. Модифицируйте запрос, добавив в него столбец level (можно назвать его и iteration). Этот столбец должен содержать номер текущей итерации, поэтому нужно увеличивать его значение на единицу на каждом шаге. Не забудьте задать начальное значение для добавленного столбца в предложении VALUES.

Задание 2. Для завершения экспериментов замените UNION ALL на UNION и выполните запрос. Сравните этот результат с предыдущим, когда мы использовали UNION ALL.

```

demo=# WITH RECURSIVE ranges ( iter, min_sum, max_sum )
AS (
VALUES(1, 0, 100000 ),
(2, 100000, 200000 ),
(3, 200000, 300000 )
UNION ALL
SELECT iter+1, min_sum + 100000, max_sum + 100000
FROM ranges
WHERE max_sum < ( SELECT max( total_amount ) FROM bookings )
)
SELECT * FROM ranges LIMIT 12;
  iter | min_sum | max_sum
-----+-----+-----
    1 |         0 | 100000
    2 |    100000 | 200000
    3 |    200000 | 300000
    2 |    100000 | 200000
    3 |    200000 | 300000
    4 |    300000 | 400000
    3 |    200000 | 300000
    4 |    300000 | 400000
    5 |    400000 | 500000
    4 |    300000 | 400000
    5 |    400000 | 500000
    6 |    500000 | 600000
(12 rows)

```



```

demo=# WITH RECURSIVE ranges ( iter, min_sum, max_sum )
AS (
VALUES(1, 0, 100000 ),
(2, 100000, 200000 ),
(3, 200000, 300000 )
UNION
SELECT iter+1, min_sum + 100000, max_sum + 100000
FROM ranges
WHERE max_sum < ( SELECT max( total_amount ) FROM bookings )
)
SELECT * FROM ranges LIMIT 12;
 iter | min_sum | max_sum
-----+-----+-----
   1  |      0  | 100000
   2  | 100000  | 200000
   3  | 200000  | 300000
   4  | 300000  | 400000
   5  | 400000  | 500000
   6  | 500000  | 600000
   7  | 600000  | 700000
   8  | 700000  | 800000
   9  | 800000  | 900000
  10  | 900000  | 1000000
  11  | 1000000 | 1100000
  12  | 1100000 | 1200000
(12 rows)

```

Для данных запросов UNION ALL на каждой итерации добавляла по 3 строки, а UNION по 1.

21. В тексте главы был приведен запрос, выводящий список городов, в которые нет рейсов из Москвы.

```

SELECT DISTINCT a.city
FROM airports a
WHERE NOT EXISTS (
    SELECT * FROM routes r
    WHERE r.departure_city = 'Москва'
    AND r.arrival_city = a.city
)
AND a.city <> 'Москва'
ORDER BY city;

```

Можно предложить другой вариант, в котором используется одна из операций над множествами строк: объединение, пересечение или разность.

Вместо знака «?» поставьте в приведенном ниже запросе нужное ключевое слово — UNION, INTERSECT или EXCEPT — и обоснуйте ваше решение.


```
SELECT city
  FROM airports
 WHERE city <> 'Москва'
?
SELECT arrival_city
  FROM routes
 WHERE departure_city = 'Москва'
ORDER BY city;
```

```
demo=# SELECT city
FROM airports
WHERE city <> 'Москва'
EXCEPT
SELECT arrival_city
FROM routes
WHERE departure_city = 'Москва'
ORDER BY city;
      city
-----
Благовещенск
Иваново
Иркутск
Калуга
Когалым
Комсомольск-на-Амуре
Кызыл
Магадан
Нижекамск
Новокузнецк
Стрежевой
Сургут
Удачный
Усть-Илимск
Усть-Кут
Ухта
Череповец
Чита
Якутск
Ярославль
(20 rows)
```

23. Предположим, что департамент развития нашей авиакомпании задался вопросом: каким будет общее число различных маршрутов, которые теоретически можно проложить между всеми городами?

Если в каком-то городе имеется более одного аэропорта, то это учитывать не будем, т. е. маршрутом будем считать путь между *городами*, а не между *аэропортами*. Здесь мы используем соединение таблицы с самой собой на основе неравенства значений атрибутов.

```
SELECT count( * )
  FROM ( SELECT DISTINCT city FROM airports ) AS a1
  JOIN ( SELECT DISTINCT city FROM airports ) AS a2
    ON a1.city <> a2.city;
```

204

Контрольные вопросы и задания

```
count
-----
10100
(1 строка)
```

Задание. Перепишите этот запрос с общим табличным выражением.

```
demo=# WITH ct AS (SELECT DISTINCT city FROM airports) SELECT count( * )
FROM ct AS a1
JOIN ct AS a2
ON a1.city <> a2.city;
count
-----
10100
(1 row)
```