ГЛАВА 1

Обзор оператора SELECT

Эта глава содержит краткий обзор элементов SQL, используемых в книге. В частности, мы рассмотрим следующие темы:

- Простые запросы `select` (см. следующий раздел)
- Обобщенные табличные выражения (см. с. 34)
- Случаи использования оператора `case` (см. с. 38)
- Соединения (см. с. 43)

Все эти элементы будут встречаться в примерах из остальной части книги, поэтому стоит потратить немного времени и освоить их применение.

Простые запросы SELECT

В начале было слово, и слово это было `select`. Чтобы получить данные из таблицы реляционной БД, нужен `select`. Вот его простейший вывод (для экономии места приведем здесь только первые 10 строк из набора результатов, содержащих обозначение штата и количество клиентов в нем; округ Колумбия (DC) будем считать штатом):

```
SELECT * FROM crm.CustomerCountByState;
```

	State	Total
0	AK	6
1	AL	0
2	AR	1
3	AZ	9
4	CA	72

	State	Total
5	CO	9
6	CT	5
7	DC	1
8	DE	0
9	FL	28

Так можно получить «сырой» дамп всех столбцов таблицы (*), не зависящий от порядка строк в таблице (*порядок базы данных*).

Перейдем к «расширенному» набору данных с бóльшим количеством столбцов. Выведем первые 10 результатов по штату, который в наборе данных клиентов будет указан первым. Так же оставим только нужные столбцы:

```
SELECT TOP 10 /* Потому что нам нужно только 10 */
    LastName,
    FirstName,
    City,
    State
FROM crm.NormalizedCustomer
ORDER BY State; -- Это фильтр для TOP 10
```

	LastName	FirstName	City	State
0	Campain	Roxane	Fairbanks	AK
1	Ferencz	Erick	Fairbanks	AK
2	Giguere	Wilda	Anchorage	AK
3	Kitty	Gail	Anchorage	AK
4	Paprocki	Lenna	Anchorage	AK
5	Weight	Penney	Anchorage	AK
6	Deleo	Carin	Little Rock	AR
7	Borgman	Keneth	Phoenix	AZ
8	Eschberger	Christiane	Phoenix	AZ
9	Kannady	Regenia	Scottsdale	AZ

Разберемся подробнее. Сначала рассмотрим сам запрос `select`. Я стараюсь всегда разбивать свои SQL-запросы именно так, потому что так их проще понять. Такой «шаблон» можно быстро пробежать глазами. Теперь посмотрим на таблицу, в которой 10 строк и 4 столбца, как мы запрашивали.

Сколько клиентов в каждом штате? Следующий код представляет собой распространенный шаблон — я постоянно использую его для разведочного анализа данных (exploratory data analysis, EDA) (https://oreil.ly/gm1Br). Этот пример может не учитывать истинных уникальных клиентов, если в данных есть дубликаты (что часто бывает, даже в рабочей CRM-системе), но для наших целей этого будет достаточно:

```
SELECT TOP 10
    State,
    COUNT(*) Total /* "Total" — это имя, присвоенное столбцу */
FROM crm.NormalizedCustomer
GROUP BY State /* Требуется для агрегатных функций */
ORDER BY 2 DESC, 1; -- Штаты с наибольшим количеством клиентов выводятся первыми
```

	State	Total
0	CA	72
1	NJ	52
2	NY	46
3	TX	32
4	PA	29
5	FL	28
6	OH	22
7	MD	17
8	IL	15
9	MI	14

В большинстве примеров я буду использовать запрос вида `select top 10`, чтобы ограничить результирующий вывод, как в примере выше.

В этом типе запроса распределения мы рассматриваем один (или несколько) столбцов и выполняем операцию подсчета `count (*)` количества строк для каждой группы (мы рассмотрим оператор `count` в главе 2, посвященной полезным функциям SQL). Обратите внимание, что результаты выполнения `count` находятся в именованном столбце, в данном случае *Total*.

Результаты показывают хорошее распределение с «длинным хвостом» единиц в *Total*. Группировка `group by` является обязательной и должна содержать все столбцы, которые не являются частью агрегации (подробнее об этом чуть позже).

Интерес представляет инструкция `order by`. В ней я сначала задаю сортировку по самым часто встречающимся штатам (`2 desc` обозначает второй столбец в результирующем наборе, в порядке убывания), а затем первый столбец в результирующем наборе (`1`). Обычно в рабочем (продакшен) коде нежелательно использовать порядковые номера столбцов в подобных предложениях, поскольку если список столбцов изменится, придется переписывать инструкцию `order by`. Удалите столбец, и внезапно его порядковый номер окажется за пределами допустимого диапазона, что приведет к ошибке. Но в этой книге мы не пишем рабочий код. Я использую эту сокращенную форму, чтобы вводить меньше символов при использовании шаблона:

```
SELECT Foo, COUNT(*) Total FROM Bar GROUP BY Foo ORDER BY 2 DESC, 1;
```

Многие диалекты также поддерживают использование порядковых чисел в `group by`. И вновь я предупреждаю, что для рабочего кода это не всегда подходит.

Если бы мне было нельзя использовать порядковые числа, я бы сделал так. Вот этот же запрос, но в нем не используются порядковые числа и столбцам результатов даны осмысленные имена:

```
SELECT TOP 10
    State [Топ 10 штатов], /* В именах столбцов можно ставить пробелы, чтобы */
    COUNT(*) [Всего клиентов] /* результат было проще читать (и если вы так
делаете, */
                              /* заключайте имена в квадратные скобки).*/
FROM crm.NormalizedCustomer
GROUP BY State
ORDER BY COUNT(*) DESC, State; -- Обратите внимание на повторное использование
функции COUNT
```

	Топ 10 штатов	Всего клиентов
0	CA	72
1	NJ	52
2	NY	46
3	TX	32
4	PA	29
5	FL	28
6	OH	22
7	MD	17
8	IL	15
9	MI	14

Обобщенные табличные выражения

Второй оператор `count(*)` поднимает еще один вопрос. Мы можем использовать функции и в других местах, например в выражении `where` или `order by`. Что, если нам нужны все штаты с *10 и более* клиентами, а не 10 первых? Мы не будем знать их количество заранее, поэтому не сможем использовать его в выражении `select top 10` (или `limit 10`). Вот один из способов получить нужный результат, используя обобщенные табличные выражения (common table expressions, CTE) или инструкцию `with`:

```
/*
   Считайте, что CTE — это временное "представление",
   хотя и не очень производительное.
   В данном случае CTE именуется CustomerCountByState.
*/
WITH CustomerCountByState
AS
(
    SELECT
        State,
        COUNT(*) Total
    FROM crm.NormalizedCustomer
    GROUP BY State
)
/*
   Теперь можно использовать CTE.
*/
SELECT
    State [Штаты с количеством клиентов, большим 10],
    Total [Клиентов в штате]
FROM CustomerCountByState
WHERE Total > 9 /* Или >= 10, на ваш выбор */
ORDER BY Total DESC, State;
```

	Штаты с количеством клиентов, большим 10	Клиентов в штате
0	CA	72
1	NJ	52
2	NY	46
3	TX	32
4	PA	29
5	FL	28
6	OH	22
7	MD	17
8	IL	15

	Штаты с количеством клиентов, большим 10	Клиентов в штате
9	MI	14
10	MA	12
11	WI	11
12	TN	10

CTE не очень эффективны. Если они используются в запросе несколько раз, они работают скорее как макрорасширение, а не как настоящее представление. Но мне они нравятся, потому что позволяют упорядочивать запросы, чтобы они были понятны другим читателям. В SQL есть *множество* способов решения предыдущей задачи. Подзапрос даст тот же результат, только, как мне кажется, менее удобочитаемый. В принципе, CTE можно рассматривать как подзапрос в следующем выражении `select`, если предопределить его и присвоить ему имя. Я назвал подзапрос *CustomerCountByState*, чтобы подчеркнуть сходство с предыдущим CTE:

```
SELECT
    State [Штаты с количеством клиентов, большим 10],
    Total [Клиентов в штате]
FROM
(
    SELECT
        State,
        COUNT(*) Total
    FROM crm.NormalizedCustomer
    GROUP BY State
) CustomerCountByState /* Нужно присвоить подзапросу имя.
                          Оно может быть любым. Я часто использую "A"
                          для таких простых запросов, как этот, но
                          сейчас я назвал его так же, как CTE, для ясности.*/
WHERE Total > 9
ORDER BY Total DESC, State;
```

	Штаты с количеством клиентов, большим 10	Клиентов в штате
0	CA	72
1	NJ	52
2	NY	46
3	TX	32
4	PA	29
5	FL	28
6	OH	22
7	MD	17

	Штаты с количеством клиентов, большим 10	Клиентов в штате
8	IL	15
9	MI	14
10	MA	12
11	WI	11
12	TN	10

Я часто использую CTE для анализа EDA, а затем, когда приходит время выпуска на продакшен, меняю их на необходимые представления с теми же именами. Следовательно:

```
-- Этот код:
WITH Foo
AS
(
    SELECT * FROM Bar WHERE Status = 'Foo'
)
SELECT * FROM Foo;
-- Меняется на:

CREATE VIEW Foo
AS
(
    SELECT * FROM Bar WHERE Status = 'Foo'
)
```

Запрос может оставаться неизменным (предполагается, что он выполняется в схеме по умолчанию), а представление может использоваться и в других запросах:

```
SELECT * FROM Foo;
```

Интересно, что выражения CTE можно записывать в строку, выстраивая и фильтруя каждое по предыдущему:

```
WITH CustomerCountByState
AS
(
    SELECT
        State,
        COUNT(*) Total
    FROM crm.NormalizedCustomer
    GROUP BY State
),
CitiesInTopStates
AS
(
    SELECT
```

```
        City,
        State,
        COUNT(*) Total
    FROM crm.NormalizedCustomer
    WHERE
        /*
           Штаты, где клиентов больше 4 (или >= 5).
        */
        State IN (SELECT State FROM CustomerCountByState WHERE Total > 4)
    GROUP BY City, State
)
SELECT
    City,
    State,
    Total
FROM CitiesInTopStates
/*
   Штаты, где клиентов 5 или больше (или > 4).
*/
WHERE Total >= 5
ORDER BY Total DESC, City, State;
```

	City	State	Total
0	New York	NY	14
1	Philadelphia	PA	8
2	Chicago	IL	7
3	Miami	FL	6
4	Raton	NM	6
5	Baltimore	MD	5
6	Gardena	CA	5
7	Milwaukee	WI	5
8	Orlando	FL	5
9	Phoenix	AZ	5
10	San Francisco	CA	5

Обратите внимание, что в следующем примере CTE может стать хорошим способом уйти от использования порядковых номеров или SQL-функций в предложении `order by`, но я бы не рекомендовал вводить их только для этого:

```
WITH TopTenStates
AS
(
    SELECT
```

```
        State,
        COUNT(*) Total
    FROM crm.NormalizedCustomer
    GROUP BY State
)
SELECT TOP 10
    State [Топ 10 штатов],
    Total [Всего клиентов]
FROM TopTenStates
ORDER BY Total DESC, State;
```

	Топ 10 штатов	Всего клиентов
0	CA	72
1	NJ	52
2	NY	46
3	TX	32
4	PA	29
5	FL	28
6	OH	22
7	MD	17
8	IL	15
9	MI	14

Случаи использования оператора CASE

Я часто использую выражение `case`. По сути, оно представляет собой разновидность оператора `switch` в языках с синтаксисом C. Вспомните старые времена, как в этом примере:

```
/* Старый добрый оператор переключения в синтаксисе C в стиле K&R. */
switch(foo) {
    case 0:
        bar = "Это был ноль.";
        break;
    case 1:
        bar = "Это был один.";
        break;
    default:
        bar = "Не знаю, что это было.";
        break;;
}
```