УРОК 10

Группировка данных

На этом уроке вы узнаете, как группировать данные таким образом, чтобы можно было подводить итоги по подмножеству записей таблицы. Для этого предназначены два предложения инструкции SELECT: GROUP BY и HAVING.

Принципы группировки данных

На предыдущем уроке вы узнали, что итоговые функции SQL можно применять для получения статистических показателей. Это позволяет подсчитывать число строк, вычислять суммы и средние значения, а также определять наибольшее и наименьшее значения, не прибегая к извлечению всех данных.

Прежде все итоговые вычисления выполнялись над всеми данными таблицы или над данными, которые соответствовали условию WHERE. В качестве напоминания приведем пример, в котором возвращается количество товаров, предлагаемых поставщиком DLL01.

Ввод ▼

```
SELECT COUNT(*) AS num_prods
FROM Products
WHERE vend_id = 'DLL01';
```

Вывод ▼

```
num_prods
-----------
4
```

Но что если вы хотите узнать количество товаров, предлагаемых каждым поставщиком? Или выяснить, какие поставщики предлагают только один товар или, наоборот, несколько товаров?

Именно в таких случаях нужно использовать *группы*. Группировка дает возможность разделить все данные на логические наборы, благодаря чему становится возможным выполнение статистических вычислений отдельно по каждой группе.

Создание групп

Группы создаются с помощью предложения GROUP BY инструкции SELECT.

Лучше всего это можно продемонстрировать на конкретном примере.

Ввод ▼

```
SELECT vend_id, COUNT(*) AS num_prods
FROM Products
GROUP BY vend_id;
```

Вывод ▼

```
vend_id     num_prods
---------   ---------
BRS01       3
DLL01       4
FNG01       2
```

Анализ ▼

Данная инструкция SELECT выводит два столбца: vend_id, содержащий идентификатор поставщика товара, и num_prods, содержащий вычисляемые поля (он создается с помощью функции COUNT(*)). Предложение GROUP BY заставляет СУБД отсортировать данные и сгруппировать их по столбцу vend_id. В результате значение num_prods будет вычисляться по одному разу для каждой группы записей vend_id, а не один раз для всей таблицы products. Как видите, в результатах указывается, что поставщик BRS01 предлагает три товара, поставщик DLL01 — четыре, а поставщик FNG01 — два.

Поскольку было использовано предложение GROUP BY, не пришлось указывать каждую группу, для которой должны быть выполнены вычисления. Это было сделано автоматически. Предложение GROUP BY заставляет СУБД сначала группировать данные, а затем выполнять вычисления по каждой группе, а не по всему набору результатов.

Прежде чем применять предложение GROUP BY, ознакомьтесь с важными правилами, которыми необходимо руководствоваться.

- В предложениях GROUP BY можно указывать произвольное число столбцов. Это позволяет вкладывать группы одна в другую, благодаря чему обеспечивается тщательный контроль над тем, какие данные подлежат группировке.
- Если в предложении GROUP BY используются вложенные группы, данные подытоживаются для последней указанной группы. Другими словами, если задана группировка, вычисления осуществляются для всех указанных столбцов (вы не сможете получить данные для каждого отдельного столбца).
- Каждый столбец, указанный в предложении GROUP BY, должен быть извлекаемым столбцом или выражением (но не итоговой функцией). Если в инструкции SELECT используется какое-то выражение, то же самое выражение должно быть указано в предложении GROUP BY. Псевдонимы применять нельзя.
- В большинстве реализаций SQL нельзя указывать в предложении GROUP BY столбцы, в которых содержатся данные переменной длины (например, текстовые поля или поля комментариев).
- За исключением инструкций, связанных с итоговыми вычислениями, каждый столбец, упомянутый в инструкции SELECT, должен быть представлен в предложении GROUP BY.
- Если столбец, по которому выполняется группировка, содержит строку со значением NULL, оно будет трактоваться как отдельная группа. Если имеется несколько строк со значениями NULL, они будут сгруппированы вместе.
- Предложение GROUP BY должно стоять после предложения WHERE и перед предложением ORDER BY.

> СОВЕТ: **ключевое слово ALL**
>
> В некоторых реализациях SQL (например, в Microsoft SQL Server) поддерживается необязательное ключевое слово ALL в предложении GROUP BY. Его можно применять для извлечения всех групп, даже тех, которые не имеют соответствующих строк (в таком случае итоговая функция возвращает значение NULL). Обратитесь к документации своей СУБД, чтобы узнать, поддерживает ли она ключевое слово ALL.

ПРЕДУПРЕЖДЕНИЕ: **указание столбцов по их относительному положению**

Некоторые реализации SQL позволяют указывать столбцы в предложении GROUP BY по их положению в списке инструкции SELECT. Например, выражение GROUP BY 2, 1 может означать группировку по второму извлекаемому столбцу, а затем — по первому. И хотя такой сокращенный синтаксис довольно удобен, он поддерживается не всеми реализациями SQL. Его применение также является рискованным в том смысле, что весьма высока вероятность возникновения ошибок при редактировании инструкций SQL.

Фильтрация по группам

SQL позволяет не только группировать данные с помощью предложения GROUP BY, но и осуществлять их фильтрацию, т.е. указывать, какие группы должны быть включены в результаты запроса, а какие — исключены из них. Например, вам может понадобиться список клиентов, которые сделали хотя бы два заказа. Чтобы получить такие данные, необходим фильтр, относящийся к целой группе, а не к отдельным строкам.

Вы уже знаете, как работает предложение WHERE (см. урок 4). Однако в данном случае его нельзя использовать, поскольку условия WHERE касаются строк, а не групп. Собственно говоря, предложение WHERE "не знает", что такое группы.

Но что тогда следует применить вместо предложения WHERE? В SQL предусмотрено другое предложение, подходящее для этих целей: HAVING. Оно очень напоминает предложение WHERE. И действительно, все типы выражений в предложении WHERE, с которыми вы уже знакомы, допустимы и в предложении HAVING. Единственная разница состоит в том, что WHERE фильтрует строки, а HAVING — группы.

СОВЕТ: **предложение HAVING поддерживает все операторы предложения WHERE**

На уроках 4 и 5 было показано, как применять предложение WHERE (включая использование метасимволов и логических операторов). Все эти метасимволы и операторы поддерживаются и в предложении HAVING. Синтаксис точно такой же, отличаются только начальные слова.

Как же осуществляется фильтрация по группам? Рассмотрим следующий пример.

Ввод ▼

```
SELECT cust_id, COUNT(*) AS orders
FROM Orders
GROUP BY cust_id
HAVING COUNT(*) >= 2;
```

Вывод ▼

```
cust_id       orders
----------    -----------
1000000001    2
```

Анализ ▼

Первые три строки этого запроса напоминают инструкцию SELECT, рассмотренную ранее. Однако в последней строке появляется предложение HAVING, которое фильтрует группы с помощью выражения COUNT(*) >= 2 — два или больше заказов.

Как видите, предложение WHERE здесь не работает, поскольку фильтрация основана на итоговом значении группы, а не на значениях отобранных строк.

> ПРИМЕЧАНИЕ: **разница между предложениями HAVING и WHERE**
>
> Вот как это можно объяснить: предложение WHERE фильтрует строки до того, как данные будут сгруппированы, а предложение HAVING — после того, как данные были сгруппированы. Это важное различие. Строки, которые были исключены по условию WHERE, не войдут в группу, иначе это могло бы изменить вычисляемые значения, которые, в свою очередь, могли бы повлиять на фильтрацию групп в предложении HAVING.

А теперь подумаем: возникает ли необходимость в использовании как предложения WHERE, так и предложения HAVING в одной инструкции? Конечно, возникает. Предположим, вы хотите усовершенствовать фильтр предыдущей инструкции таким образом, чтобы возвращались имена всех клиентов, которые сделали два или более заказа за последние 12 месяцев. Чтобы добиться этого, можно

добавить предложение WHERE, которое учитывает только заказы, сделанные за последние 12 месяцев. Затем вы добавляете предложение HAVING, чтобы отфильтровать только те группы, в которых имеются минимум две строки.

Чтобы лучше разобраться в этом, рассмотрим следующий пример, в котором перечисляются все поставщики, предлагающие не менее двух товаров по цене 4 доллара и более за единицу.

Ввод ▼

```
SELECT vend_id, COUNT(*) AS num_prods
FROM Products
WHERE prod_price >= 4
GROUP BY vend_id
HAVING COUNT(*) >= 2;
```

Вывод ▼

```
vend_id       num_prods
----------    -----------
BRS01          3
FNG01          2
```

Анализ ▼

Данный пример нуждается в пояснении. Первая строка представляет собой основную инструкцию SELECT, использующую итоговую функцию, — точно так же, как и в предыдущих примерах. Предложение WHERE фильтрует все строки со значениями в столбце prod_price не менее 4. Затем данные группируются по столбцу vend_id, после чего предложение HAVING фильтрует только группы, содержащие не менее двух членов. При отсутствии предложения WHERE была бы получена лишняя строка (поставщик, предлагающий четыре товара, каждый из которых дешевле 4 долларов), как показано ниже.

Ввод ▼

```
SELECT vend_id, COUNT(*) AS num_prods
FROM Products
GROUP BY vend_id
HAVING COUNT(*) >= 2;
```

Вывод ▼

```
vend_id      num_prods
----------   -----------
BRS01        3
DLL01        4
FNG01        2
```

> ПРИМЕЧАНИЕ: **использование предложений HAVING и WHERE**
>
> Предложение HAVING столь сильно напоминает предложение WHERE, что в большинстве СУБД оно трактуется точно так же, если только не указано предложение GROUP BY. Тем не менее следует знать, что между ними существует разница. Используйте предложение HAVING только вместе с предложением GROUP BY, а предложение WHERE — для стандартной фильтрации на уровне строк.

Группировка и сортировка

Важно понимать, что предложения GROUP BY и ORDER BY существенно различаются, хотя с их помощью иногда можно добиться одинаковых результатов. Разобраться в этом поможет табл. 10.1.

ТАБЛИЦА 10.1. **Сравнение предложений ORDER BY и GROUP BY**

ORDER BY	GROUP BY
Сортирует полученные результаты	Группирует строки. Однако отображаемый результат может не соответствовать порядку группировки
Могут быть использованы любые столбцы (даже не указанные в предложении SELECT)	Могут быть использованы только извлекаемые столбцы или выражения; должно быть указано каждое выражение из предложения SELECT
Не является необходимым	Требуется, если используются столбцы (или выражения) с итоговыми функциями

Первое из отличий, перечисленных в табл. 10.1, является очень важным. Чаще всего вы обнаружите, что данные, сгруппированные

с помощью предложения GROUP BY, будут отображаться в порядке группировки. Но так будет не всегда, и в действительности этого не требуется в спецификациях SQL. Более того, даже если СУБД сортирует данные так, как указано в предложении GROUP BY, вам может понадобиться отсортировать их по-другому. То, что вы группируете данные определенным способом (чтобы получить для группы необходимые итоговые значения), не означает, что требуемый результат должен быть отсортирован именно так. Следует явным образом указывать предложение ORDER BY, даже если оно совпадает с предложением GROUP BY.

> СОВЕТ: **не забывайте использовать предложение ORDER BY**
>
> Как правило, всякий раз, когда вы используете предложение GROUP BY, приходится указывать и предложение ORDER BY. Это единственный способ, гарантирующий, что данные будут отсортированы правильно. Не следует надеяться на то, что данные будут отсортированы предложением GROUP BY.

Чтобы продемонстрировать совместное использование предложений GROUP BY и ORDER BY, рассмотрим пример. Следующая инструкция SELECT аналогична тем, которые использовались ранее: она выводит номер заказа и количество товаров для всех заказов, которые содержат три товара или больше.

Ввод ▼

```
SELECT order_num, COUNT(*) AS items
FROM OrderItems
GROUP BY order_num
HAVING COUNT(*) >= 3;
```

Вывод ▼

```
order_num      items
----------     -----
20006          3
20007          5
20008          5
20009          3
```

Чтобы отсортировать результат по количеству заказанных товаров, все, что необходимо сделать, — это добавить предложение ORDER BY, как показано ниже.

Ввод ▼

```
SELECT order_num, COUNT(*) AS items
FROM OrderItems
GROUP BY order_num
HAVING COUNT(*) >= 3;
ORDER BY items, order_num;
```

> ПРИМЕЧАНИЕ: **несовместимость с Access**
>
> Microsoft Access не позволяет осуществлять сортировку по псевдонимам, и для данной СУБД этот пример неприменим. Выход состоит в замене столбца items (в предложении ORDER BY) вычисляемым выражением или номером поля. По существу, будут работать оба варианта: ORDER BY COUNT(*), order_num и ORDER BY 1, order_num.

Вывод ▼

```
order_num      items
----------     -----
20006          3
20009          3
20007          5
20008          5
```

Анализ ▼

В этом примере предложение GROUP BY используется для группировки данных по номеру заказа (столбец order_num), благодаря чему функция COUNT(*) может вернуть количество товаров в каждом заказе. Предложение HAVING фильтрует данные таким образом, что возвращаются только заказы с тремя и более товарами. Наконец, результат сортируется за счет использования предложения ORDER BY.

Порядок предложений в инструкции SELECT

Предложения инструкции SELECT должны указываться в определенном порядке. В табл. 10.2 перечислены все предложения, которые мы изучили до сих пор, в порядке, в котором они должны следовать.

ТАБЛИЦА 10.2. **Предложения инструкции SELECT и порядок их следования**

Предложение	Описание	Необходимость
SELECT	Столбцы или выражения, которые должны быть получены	Да
FROM	Таблица для извлечения данных	Только если извлекаются данные из таблицы
WHERE	Фильтрация на уровне строк	Нет
GROUP BY	Определение группы	Только если выполняются итоговые вычисления по группам
HAVING	Фильтрация на уровне групп	Нет
ORDER BY	Порядок сортировки результатов	Нет

Резюме

На предыдущем уроке вы узнали, как применять итоговые функции SQL для выполнения сводных вычислений. На этом уроке рассказывалось о том, как использовать предложение GROUP BY для выполнения аналогичных вычислений по отношению к группам данных и получения отдельных результатов для каждой группы. Было показано, как с помощью предложения HAVING осуществлять фильтрацию на уровне групп. Кроме того, объяснялось, какова разница между предложениями ORDER BY и GROUP BY, а также между предложениями WHERE и HAVING.