

Лекция 8. Группировка данных (продолжение)

В рамках данной лекции мы рассмотрим группировку данных по одному или более столбцам, группировку данных с помощью выражений и формирование обобщений в рамках группы.

Группировка по одному столбцу

Формирование группы по одному столбцу – самый простой и наиболее распространенный тип группировки. Например, если требуется найти общие остатки (*total balance*) для всех типов счетов, нужно всего лишь провести группировку по столбцу *account.product_cd*:

```
SELECT product_cd, SUM(avail_balance) prod_balance
FROM account
GROUP BY product_cd;
```

product_cd	prod_balance
BUS	9345.55
CD	19500.00
CHK	73008.01
MM	17045.14
SAV	1855.76
SBL	50000.00

Этот запрос формирует 6 групп, по одной для каждого типа счетов, и затем суммирует доступные остатки по всем строкам в каждой группе.

Группировка по нескольким столбцам

В некоторых случаях может понадобиться сформировать группы, охватывающие более одного столбца. Развивая предыдущий пример, представим, что требуется найти общие остатки не только по каждому типу счетов, но и по отделениям:

```
SELECT product_cd, open_branch_id,
       SUM(avail_balance) tot_balance
FROM account
GROUP BY product_cd, open_branch_id;
```

product_cd	open_branch_id	tot_balance
BUS	2	9345.55
BUS	4	0.00
CD	1	11500.00

CD	2	8000.00
CHK	1	782.16
CHK	2	3315.77
CHK	3	1057.75
CHK	4	67852.33
MM	1	14832.64
MM	3	2212.50
SAV	1	767.77
SAV	2	700.00
SAV	4	387.99
SBL	3	50000.00

Этот вариант запроса формирует 14 групп, по одной для каждого обнаруженного в таблице **account** сочетания типа счетов и отделения.

Группировка посредством выражений

Кроме столбцов группировку данных можно выполнить на основании значений, сгенерированных выражениями. Рассмотрим запрос, который группирует сотрудников по году начала их работы в банке:

```
SELECT EXTRACT(YEAR FROM start_date) year,
       COUNT(*) how_many
FROM employee
GROUP BY EXTRACT(YEAR FROM start_date);
```

year	how_many
2004	2
2005	3
2006	8
2007	3
2008	2

Формирование обобщений

Выше мы рассмотрели пример группировки по нескольким столбцам, когда выводили общие остатки счетов по каждому типу счетов и отделению. Допустим теперь, что кроме общих остатков для каждого сочетания тип счетов/отделение требуется получить и общие остатки по каждому отдельному типу счетов. Чтобы это сделать, можно использовать оператор **with rollup** (с обобщением):

```
SELECT product_cd, open_branch_id,
       SUM(avail_balance) tot_balance
FROM account
GROUP BY product_cd, open_branch_id WITH ROLLUP;
```

product_cd	open_branch_id	tot_balance
BUS	2	9345.55
BUS	4	0.00
BUS	NULL	9345.55
CD	1	11500.00
CD	2	8000.00
CD	NULL	19500.00
CHK	1	782.16
CHK	2	3315.77
CHK	3	1057.75
CHK	4	67852.33
CHK	NULL	73008.01
MM	1	14832.64
MM	3	2212.50
MM	NULL	17045.14
SAV	1	767.77
SAV	2	700.00
SAV	4	387.99
SAV	NULL	1855.76
SBL	3	50000.00
SBL	NULL	50000.00
NULL	NULL	170754.46

21 строка

Теперь имеется 7 дополнительных результатов, по одному для каждого из шести разных типов счетов, и один – общая сумма (для всех типов счетов). Для 6 обобщений по типам счетов столбец *open_branch_id* содержит значение **null**, поскольку обобщение осуществляется по всем отделениям. Например, взглянув на строку #3 результата, можно заметить, что всего по счетам *BUS* во всех отделениях внесено \$9345,55. Строка итоговой суммы в обоих столбцах, *product_cd* и *open_branch_id*, также содержит значение **null**. Последняя строка выходных данных показывает общую сумму \$170 754,46 для всех типов счетов и всех отделений.

При работе с **Oracle Database** для выполнения обобщения применяется немного отличающийся синтаксис:

GROUP BY ROLLUP(product_cd, open_branch_id)

Преимущество этого синтаксиса в том, что он позволяет выполнять обобщения для подмножества столбцов в блоке *group by*. Например, если группировка осуществляется по столбцам *a*, *b* и *c*, можно было бы указать, что сервер должен проводить обобщения только для *b* и *c*:

GROUP BY a, ROLLUP(b, c)

Если кроме суммы по типам счетов требуется подсчитать сумму по каждому отделению, можно использовать вариант **with cube**, который

формирует строки суммы для всех возможных сочетаний группирующих столбцов:

```
SELECT product_cd, open_branch_id,
       SUM(avail_balance) tot_balance
FROM account
GROUP BY product_cd, open_branch_id WITH CUBE;
```

product_cd	open_branch_id	tot_balance
NULL	NULL	170754.46
NULL	1	27882.57
NULL	2	21361.32
NULL	3	53270.25
NULL	4	68240.32
BUS	2	9345.55
BUS	4	0.00
BUS	NULL	9345.55
CD	1	11500.00
CD	2	8000.00
CD	NULL	19500.00
CHK	1	782.16
CHK	2	3315.77
CHK	3	1057.75
CHK	4	67852.33
CHK	NULL	73008.01
MM	1	14832.64
MM	3	2212.50
MM	NULL	17045.14
SAV	1	767.77
SAV	2	700.00
SAV	4	387.99
SAV	NULL	1855.76
SBL	3	50000.00
SBL	NULL	50000.00

25 строк

Применение **with cube** дает на 4 строки больше, чем версия запроса с **with rollup**, по одной для каждого из четырех ID отделений. Как и в случае с **with rollup**, значения **null** в столбце **product_cd** обозначают то, что производится суммирование по отделениям.

При работе с **Oracle Database** для указания на операцию **cube** также применяется немного отличающийся синтаксис:

```
GROUP BY CUBE(product_cd, open_branch_id)
```

Условия групповой фильтрации

В лекции 4 были представлены различные типы условий фильтрации и показано, как их можно использовать в блоке **where**. При группировке данных

тоже можно применять условия фильтрации к данным после формирования групп. Этот тип условий фильтрации должен располагаться в блоке **having**:

```
SELECT product_cd, SUM(avail_balance) prod_balance
FROM account
WHERE status = 'ACTIVE'
GROUP BY product_cd
HAVING SUM(avail_balance) >= 10000;
```

Условия фильтрации

product_cd	prod_balance
CD	19500.00
CHK	73008.01
MM	17045.14
SBL	50000.00

Данный запрос содержит **два** условия фильтрации:

- 1) в блоке **where**, которое отсеивает неактивные счета (воздействует на данные *до* группировки);
- 2) в блоке **having**, которое отсеивает счета всех типов с общим доступным остатком меньше \$10 000 (воздействует на данные *после* создания групп).

Если по ошибке оба фильтра помещены в блок **where**, возникает следующая ошибка:

```
SELECT product_cd, SUM(avail_balance) prod_balance
FROM account
WHERE status = 'ACTIVE'
      AND SUM(avail_balance) > 10000
GROUP BY product_cd;
```

ERROR 1111 (HY000): Invalid use of group function

Данный запрос дал сбой, потому что агрегатную функцию нельзя включать в блок **where**.

! При введении фильтров в запрос, включающий блок **group by**, необходимо тщательно продумать, к чему применяется фильтр – к необработанным данным (тогда он относится к блоку **where**) или к сгруппированным данным (в этом случае он относится к блоку **having**).

В блок **having**, однако, можно включить агрегатные функции, *не* перечисленные в блоке **select**, например:

```

SELECT product_cd, SUM(avail_balance) prod_balance
FROM account
WHERE status = 'ACTIVE'
GROUP BY product_cd
HAVING MIN(avail_balance) >= 1000
      AND MAX(avail_balance) <= 10000;

```

product_cd	prod_balance
CD	19500.00
MM	17045.14

Этот запрос формирует общие остатки для каждого типа счетов, но условие фильтрации блока **having** исключает все группы, минимальный остаток которых меньше \$1000 или максимальный остаток которых больше \$10 000.

Литература:

1. Алан Бьюли. Изучаем SQL: пер. с англ. – СПб-М.: Символ, O'Reilly, 2007. – 310 с.
2. Alan Beaulieu. Learning SQL, 2nd Edition. – O'Reilly Media, 2009. – 337 p.