Лекция 11. Условная логика. Выражение case

В определенных ситуациях может потребоваться, чтобы SQL-выражения вели себя так или иначе в зависимости от значений определенных столбцов или выражений. ***Условная логика*** дает возможность выбирать одно из направлений выполнения программы.

Выражение *case*

Все основные серверы БД включают встроенные функции, имитирующие выражение *if_then_else*, которое есть в большинстве языков программирования (например, функция **decode()** Oracle, функция **if()** MySQL и функция **coalesce()** SQL Server). Выражения ***case*** тоже разработаны для поддержки логики *if_then_else*, но в сравнении со встроенными функциями обладают двумя преимуществами:

- Выражение *case* является частью стандарта SQL (версия SQL92) и реализовано в Oracle Database, SQL Server и MySQL.
- Выражения *case* встроены в грамматику SQL и могут быть включены в выражения *select, insert, update* и *delete*.

Далее будут представлены выражения *case* двух разных типов.

Выражения *case* с перебором вариантов

Синтаксис данного выражения:

```
CASE
   WHEN C1 THEN E1
   WHEN C2 THEN E2
   ...
   WHEN CN THEN EN
   [ELSE ED]
END
```

Выражение по умолчанию

В этом описании C1, C2, …, CN обозначают условия, а E1, E2, …, EN – выражения, которые должны быть возвращены выражением *case*. Если условие в блоке *when* выполняется, выражение *case* возвращает соответствующее выражение. Кроме того, символ ED представляет применяемое по умолчанию выражение, возвращаемое выражением *case*, если не выполнено *ни одно* из условий C1, C2, …, CN (блок *else* является необязательным, поэтому он заключен в квадратные скобки). Все выражения, возвращаемые различными блоками *when*, должны обеспечивать результаты одного типа (например, *date*, *number*, *varchar*).

Рассмотрим пример выражения *case с перебором вариантов* (*searched case*):

```
SELECT title,
  CASE
     WHEN employee.title = 'Head Teller'
        THEN 'Head Teller'
     WHEN employee.title = 'Teller'
     AND YEAR(employee.start_date) > 2007
        THEN 'Teller Trainee'
     WHEN employee.title = 'Teller'
     AND YEAR(employee.start_date) < 2006
        THEN 'Experienced Teller'
     WHEN employee.title = 'Teller'
        THEN 'Teller'
     ELSE 'Non-Teller'
  END qualification
FROM employee;
```

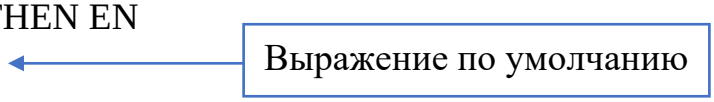
При вычислении выражения *case* блоки *when* обрабатываются сверху вниз. Как только одно из условий блока *when* принимает значение *true*, возвращается соответствующее выражение, а все остальные блоки *when* игнорируются. Если ни одно из условий блока *when* не выполняется, возвращается выражение блока *else*.

Хотя предыдущий пример возвращает строковые выражения, помните, что выражения *case* могут возвращать выражения любого типа, включая подзапросы.

Простые выражения *case*

Простое выражение case (*simple case expression*) очень похоже на выражение *case с перебором вариантов*, но несколько менее функционально. Синтаксис:

```
CASE V0
   WHEN V1 THEN E1
   WHEN V2 THEN E2
   ...
   WHEN VN THEN EN
   [ELSE ED]
END
```



В этом описании V0 представляет значение, а символы V1, V2, …, VN – значения, сравниваемые с V0. Символы E1, E2, …, EN представляют выражения, возвращаемые выражением *case*, а ED – выражение, которое должно быть возвращено, если ни одно из значений набора V1, V2, …, VN не соответствует значению V0.

Вот пример *простого выражения case*:

```
SELECT cust_type_cd,
  CASE customer.cust_type_cd
     WHEN 'I' THEN
        (SELECT CONCAT(i.first_name, ' ', i.last_name)
        FROM individual i
        WHERE i.cust_id = customer.cust_id)
     WHEN 'B' THEN
        (SELECT b.name
        FROM business b
        WHERE b.cust_id = customer.cust_id)
     ELSE 'Unknown Customer Type'
  END customer_name
FROM customer;
```

Простые выражения case менее функциональны, чем выражения *case* с *перебором вариантов*, потому что в них нельзя задать собственные условия; в них просто используются условия равенства. Ниже показана аналогичная логика, использующая выражение *case с перебором вариантов*:

```
SELECT cust_type_cd,
  CASE
     WHEN customer.cust_type_cd = 'I' THEN
        (SELECT CONCAT(i.first_name, ' ', i.last_name)
        FROM individual I
        WHERE i.cust_id = customer.cust_id)
     WHEN customer.cust_type_cd = 'B' THEN
        (SELECT b.name FROM business b
        WHERE b.cust_id = customer.cust_id)
     ELSE 'Unknown Customer Type'
  END customer_name
FROM customer;
```

Выражения *case с перебором вариантов* позволяют создавать условия вхождения в диапазон, условия неравенства и составные условия,

использующие *and/or/not*, поэтому рекомендуется применять выражения *case с перебором вариантов* во всех случаях, кроме самых простых.

Ошибки деления на ноль

Проводя вычисления, включающие деление, нужно все время заботиться о том, чтобы знаменатель никогда не был равен нулю. Некоторые серверы БД, такие как Oracle Database, встретив нулевой знаменатель, формируют ошибку, а MySQL просто присваивает результату вычисления значение *null*, как показывает следующий пример:

```
mysql> SELECT 100 / 0;
+-----------+
| 100 / 0 |
+-----------+
|    NULL |
+-----------+
1 row in set (0.00 sec)
```

Чтобы защитить вычисления от ошибок или, еще хуже, от загадочного получения *null*, следует ко всем знаменателям применять условную логику.

Рассмотрим запрос, который вычисляет отношение остатка на счете к общему остатку для всех счетов одного типа. Поскольку для некоторых типов счетов, таких как ссуды коммерческим предприятиям, общий остаток может равняться нулю, если на текущий момент все ссуды полностью выплачены, лучше всего включить выражение *case*, гарантирующее, что знаменатель никогда не будет равен нулю.

```
SELECT a.cust_id, a.product_cd, a.avail_balance /
  CASE
      WHEN prod_tots.tot_balance = 0 THEN 1
      ELSE prod_tots.tot_balance
  END percent_of_total
FROM account a INNER JOIN
   (SELECT a.product_cd, SUM(a.avail_balance) tot_balance
    FROM account a
    GROUP BY a.product_cd) prod_tots
    ON a.product_cd = prod_tots.product_cd;
```

```
+---------+------------+------------------+
| cust_id | product_cd | percent_of_total |
+---------+------------+------------------+
|      10 | BUS        |         0.000000 |
|      11 | BUS        |         1.000000 |
|       1 | CD         |         0.153846 |
|       6 | CD         |         0.512821 |
|       7 | CD         |         0.256410 |
|       9 | CD         |         0.076923 |
|       1 | CHK        |         0.014488 |
|       2 | CHK        |         0.030928 |
|       3 | CHK        |         0.014488 |
|       4 | CHK        |         0.007316 |
|       5 | CHK        |         0.030654 |
|       6 | CHK        |         0.001676 |
|       8 | CHK        |         0.047764 |
|       9 | CHK        |         0.001721 |
|      10 | CHK        |         0.322911 |
|      12 | CHK        |         0.528052 |
|       3 | MM         |         0.129802 |
|       4 | MM         |         0.321915 |
|       9 | MM         |         0.548282 |
|       1 | SAV        |         0.269431 |
|       2 | SAV        |         0.107773 |
|       4 | SAV        |         0.413723 |
|       8 | SAV        |         0.209073 |
|      13 | SBL        |         1.000000 |
+---------+------------+------------------+
24 rows in set (0.13 sec)
```

Обработка значений Null

Хотя значения *null* удобны для хранения в таблицах неизвестных значений столбцов, они не всегда подходят для отображения или использования в выражениях. Например, в окне ввода данных вы, скорее всего, предпочтете отображать слово «*unknown*», а не оставлять пустое поле. При извлечении данных выражение *case* позволяет вместо значения *null* подставлять строку:

```
SELECT emp_id, fname, lname,
  CASE
     WHEN title IS NULL THEN 'Unknown'
     ELSE title
  END
FROM employee;
```

Значения *null* в вычислениях часто являются причиной результата *null*, как показывает следующий пример:

```
mysql> SELECT (7 * 5) / ((3 + 14) * null);
```

```
+---------------------------------+
| (7 * 5) / ((3 + 14) * null)   |
+---------------------------------+
|                            NULL |
+---------------------------------+
1 row in set (0.08 sec)
```

Проводя вычисления, полезно преобразовать значения *null* в число (обычно 0 или 1) с помощью выражения *case*, чтобы обеспечить результат вычисления, отличный от *null*.

Также условная логика часто применяется при **обновлении** или **удалении** данных таблиц.

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.