Advanced SQL Tutorial

Stephen CUI

November 4, 2022

目录

Chapter 1

神奇的SQL

1.1 CASE表达式

1.1.1 写在前面

因为 CASE 表达式是不依赖于具体数据库的技术，所以可以提高 SQL 代码的可移植性。

1.1.2 CASE 表达式概述

CASE 表达式有简单 CASE 表达式（simple case expression）和搜索 CASE 表达式（searched case expression）两种写法

Examples 1.1.1 CASE表达式概述

```
-- 简单CASE表达式
CASE sex
WHEN '1' THEN '男'
WHEN '2' THEN '女'
ELSE '其他' END;

-- 搜索CASE表达式
CASE
WHEN sex='1' THEN '男'
WHEN sex='2' THEN '女'
ELSE '其他' END;
```

简单 CASE 表达式正如其名，写法简单，但能实现的事情比较有限。简单 CASE 表达式能写的条件，搜索 CASE 表达式也能写。

在编写 SQL 语句的时候需要注意，在发现为真的 WHEN 子句时， CASE 表达式的真假值判断就会

中止，而剩余的 WHEN 子句会被忽略。为了避免引起不必要的混乱，使用 WHEN 子句时要注意条件的排他性。

Examples 1.1.2 WHEN子句的排他性

```
-- 剩余的WHEN子句被忽略的写法示例
CASE
WHEN COL_1 IN {'A', 'B'} THEN 'FIRSTT'
WHEN COL_1 IN {'A'} THEN 'SECOND'
     ELSE 'OTHERS' END;
```

注意事项：

1. 统一各分支返回的数据类型

   一定要注意 CASE 表达式里各个分支返回的数据类型是否一致。某个分支返回字符型，而其他分支返回数值型的写法是不正确的。

2. 不要忘了写END

   使用 CASE 表达式的时候，最容易出现的语法错误是忘记写 END。

3. 养成写ELSE子句的习惯

   与 END 不同，ELSE 子句是可选的，不写也不会出错。不写 ELSE 子句时， CASE 表达式的执行结果是 NULL。但是不写可能会造成“语法没有错误，结果却不对”这种不易追查原因的麻烦，所以最好明确地写上 ELSE 子句（即便是在结果可以为 NULL 的情况下）。

1.1.3 将已有编号方式转化为新的方式并统计

建表的命令点这里

Examples 1.1.3

```
SELECT
    CASE pref_name
        WHEN '德岛' THEN '四国'
        WHEN '香川' THEN '四国'
        WHEN '爱媛' THEN '四国'
        WHEN '高知' THEN '四国'
        WHEN '福冈' THEN '九州'
        WHEN '佐贺' THEN '九州'
        WHEN '长崎' THEN '九州'
        ELSE '其他'
```

```
    END AS district,
    SUM(population)
FROM
    poptbl
GROUP BY CASE pref_name
    WHEN '德岛' THEN '四国'
    WHEN '香川' THEN '四国'
    WHEN '爱媛' THEN '四国'
    WHEN '高知' THEN '四国'
    WHEN '福冈' THEN '九州'
    WHEN '佐贺' THEN '九州'
    WHEN '长崎' THEN '九州'
    ELSE '其他'
END;
```

这里的关键在于将 SELECT 子句里的 CASE 表达式复制到 GROUP BY 子句里。需要注意的是，如果对转换前的列pref_name进行 GROUP BY，就得不到正确的结果（因为这并不会引起语法错误，所以容易被忽视）。

Examples 1.1.4 确实有些麻烦，但是这样是符合SQL标准的写法

```
SELECT
    CASE
        WHEN population < 100 THEN '01'
        WHEN population >= 100 AND population < 200 THEN '02'
        WHEN population >= 200 AND population < 300 THEN '03'
        WHEN population >= 300 THEN '04'
        ELSE NULL
    END AS pop_class,
    COUNT(*) AS cnt
FROM
    poptbl
GROUP BY CASE
    WHEN population < 100 THEN '01'
    WHEN population >= 100 AND population < 200 THEN '02'
    WHEN population >= 200 AND population < 300 THEN '03'
    WHEN population >= 300 THEN '04'
    ELSE NULL
```

```
END;
```

这个技巧非常好用。不过，必须在 SELECT 子句和 GROUP BY 子句这两处写一样的 CASE 表达式，这有点儿麻烦。后期需要修改的时候，很容易发生只改了这一处而忘掉改另一处的失误。所以，如果我们可以像下面这样写，那就方便多了。

Examples 1.1.5 这要这样写

```
-- 不建议使用这种写法
SELECT
    CASE pref_name
        WHEN '德岛' THEN '四国'
        WHEN '香川' THEN '四国'
        WHEN '爱媛' THEN '四国'
        WHEN '高知' THEN '四国'
        WHEN '福冈' THEN '九州'
        WHEN '佐贺' THEN '九州'
        WHEN '长崎' THEN '九州'
        ELSE '其他'
    END AS district,
    SUM(population)
FROM
    poptbl
GROUP BY district;
```

没错，这里的 GROUP BY 子句使用的正是 SELECT 子句里定义的列的别称——district。但是严格来说，这种写法是违反标准 SQL 的规则的。因为 GROUP BY 子句比 SELECT 语句先执行，所以在 GROUP BY 子句中引用在 SELECT 子句里定义的别称是不被允许的。

1.1.4 用一条SQL语句进行不同条件的统计

进行不同条件的统计是 CASE 表达式的著名用法之一。（建表的命令点这里）

Examples 1.1.6 典型的将一维表格转换为二维表格

```
SELECT
    pref_name,
    SUM(CASE WHEN sex = '1' THEN population ELSE 0 END) AS cnt_m,
    SUM(CASE WHEN sex = '2' THEN population ELSE 0 END) AS cnt_f
```

```
5 FROM poptbl2
6 GROUP BY pref_name;
```

这里是将“行结构”的数据转换成了“列结构”的数据。除了 SUM，COUNT、AVG 等聚合函数也都可以用于将行结构的数据转换成列结构的数据。

这个技巧可贵的地方在于，它能将 SQL 的查询结果转换为二维表的格式。

1.1.5 用CHECK约束定义多个列的条件关系

1.1.6 在UPDATE语句里进行条件分支