SQL进阶教程

Advanced SQL Tutorial

Stephen CUI[1]

November 4, 2022

[1]cuixuanStephen@gmail.com

目录

1 神奇的SQL **1**

1.1 CASE表达式 1

1.1.1 写在前面 1

1.1.2 CASE 表达式概述 1

1.1.3 将已有编号方式转化为新的方式并统计 2

1.1.4 用一条SQL语句进行不同条件的统计 4

1.1.5 用CHECK约束定义多个列的条件关系 5

1.1.6 在UPDATE语句里进行条件分支 5

1.1.7 表之间的数据匹配 6

1.1.8 在 CASE 表达式中使用聚合函数 7

1.1.9 本节小结 7

1.2 用SQL进行集合运算 8

1.2.1 写在前面 8

1.2.2 导入篇：集合运算的几个注意事项 8

1.2.3 比较表和表：检查集合相等性之基础篇 8

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
```

```
18 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

```
1  -- 不建议使用这种写法
2  SELECT
3      CASE pref_name
4          WHEN '德岛' THEN '四国'
5          WHEN '香川' THEN '四国'
6          WHEN '爱媛' THEN '四国'
7          WHEN '高知' THEN '四国'
8          WHEN '福冈' THEN '九州'
9          WHEN '佐贺' THEN '九州'
10         WHEN '长崎' THEN '九州'
11         ELSE '其他'
12     END AS district,
13     SUM(population)
14 FROM
15     poptbl
16 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

```
1  SELECT
2      pref_name,
3      SUM(CASE WHEN sex = '1' THEN population ELSE 0 END) AS cnt_m,
4      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

如果 CASE 表达式里没有明确指定 ELSE 子句，执行结果会被默认地处理成 ELSE NULL。

这个技巧的应用范围很广。例如，可以用它简单地完成主键值调换这种繁重的工作。通常，当我们想调换主键值 a 和 b 时，需要将主键值临时转换成某个中间值。使用这种方法时需要执行 3 次 UPDATE 操作，但是如果使用 CASE 表达式，1 次就可以做到。

```
update sometable
        set p_key = 'd'
        where p_key = 'a';

update sometable
        set p_key = 'a'
    where p_key = 'b';

update sometable
        set p_key = 'b'
    where p_key = 'd';
```

这里没有必要执行 3 次 UPDATE 操作，而且中间值 d 是否总能使用也是问题。而如果使用 CASE 表达式，就不必担心这些，1 次就可以完成调换[1]。

```
update sometable
        set p_key = case
    when p_key = 'a' then 'b'
    when p_key = 'b' then 'a'
    else p_key end
where p_key in ('a', 'b');
```

[1]如果在 PostgreSQL 和 MySQL 数据库执行这条 SQL 语句，会因主键重复而出现错误。但是，约束的检查本来就发生在更新完成后，因此更新途中主键一时出现重复也没有问题。事实上，在 Oracle、 DB2、SQL Server 数据库执行都没有问题。

1.1.7 表之间的数据匹配

与 DECODE 函数等相比，CASE 表达式的一大优势在于能够判断表达式。也就是说，在 CASE 表达式里，我们可以使用 BETWEEN、LIKE 和 >、 < 等便利的谓词组合，以及能嵌套子查询的 IN 和 EXISTS 谓词。因此，CASE 表达式具有非常强大的表达能力。

建表命令点击这里Course

我们需要做的是，检查表 OpenCourses 中的各月里有表 CourseMaster 中的哪些课程。这个匹配条件可以用 CASE 表达式来写。

```
select course_name,
        case when course_id in
        (select course_id from opencourses where month=200706)
    then 'yes' else 'no' end as 'June',
    case when course_id in
        (select course_id from opencourses where month=200707)
    then 'yes' else 'no' end 'July',
    case when course_id in
        (select course_id from opencourses where month=200708)
    then 'yes' else 'no' end 'August'
    from coursemaster;
```

```
select cm.course_name,
        case when exists
                (select course_id from opencourses oc
         where month=200706 and oc.course_id = cm.course_id)
                then 'Yes' else 'No' end 'June',
        case when exists
                (select course_id from opencourses oc
         where month=200707 and oc.course_id = cm.course_id)
                then 'Yes' else 'No' end 'July',
        case when exists
                (select course_id from opencourses oc
         where month=200708 and oc.course_id = cm.course_id)
                then 'Yes' else 'No' end 'August'
        from coursemaster cm;
```

这样的查询没有进行聚合，因此也不需要排序，月份增加的时候仅修改 SELECT 子句就可以了，扩展性比较好。

无论使用 IN 还是 EXISTS，得到的结果是一样的，但从性能方面来说， EXISTS 更好。

1.1.8 在 CASE 表达式中使用聚合函数

接下来介绍一下稍微高级的用法。这个用法乍一看可能让人觉得像是语法错误，实际上却并非如此。

建表命令点击这里StudentClub

```
select std_id, max(club_id) as main_club
        from studentclub
    group by std_id
    having count(*) = 1
union all
select std_id, club_id as main_club
        from studentclub
    where main_club_flg = 'Y';
```

这样做也能得到正确的结果，但需要写多条 SQL 语句。而如果使用 CASE 表达式，下面这一条 SQL 语句就可以了。

```
select std_id,
        case when count(*) = 1
                then max(club_id)
                else max(case when main_club_flg = 'Y' then club_id else null end)
    end 'main_club'
        from studentclub
    group by std_i
```

这条 SQL 语句在 CASE 表达式里使用了聚合函数，又在聚合函数里使用了 CASE 表达式。

1.1.9 本节小结

1. 在 GROUP BY 子句里使用 CASE 表达式，可以灵活地选择作为聚合的单位的编号或等级。这一点在进行非定制化统计时能发挥巨大的威力。

2. 在聚合函数中使用 CASE 表达式，可以轻松地将行结构的数据转换成列结构的数据。

3. 相反，聚合函数也可以嵌套进 CASE 表达式里使用。

4. 相比依赖于具体数据库的函数，CASE 表达式有更强大的表达能力和更好的可移植性。

5. 正因为 CASE 表达式是一种表达式而不是语句，才有了这诸多优点。

1.2 用SQL进行集合运算

SQL 语言的基础之一是集合论。但是，很长一段时间内，由于 SQL 没能很好地支持集合运算，所以相关功能并没有被人们充分地利用。过去这些年，SQL 凑齐了大部分基础的集合运算，人们终于可以真正地使用它了。

1.2.1 写在前面

集合论是 SQL 语言的根基——这是贯穿全书的主题之一。因为它的这个特性，SQL 也被称为面向集合语言。

1.2.2 导入篇：集合运算的几个注意事项

顾名思义，集合运算符的参数是集合，从数据库实现层面上来说就是表或者视图。因为和高中学过的集合代数很像，所以理解起来相对比较容易。但是，SQL 还是有几个特别的地方需要注意一下。

注意事项：

1. SQL 能操作具有重复行的集合，可以通过可选项 ALL来支持

   一般的集合论是不允许集合里存在重复元素的，因此集合 {1, 1, 2, 3, 3, 3} 和集合 {1, 2, 3} 被视为相同的集合。但是关系数据库里的表允许存在重复的行，称为多重集合（multiset, bag）。

   因此，SQL 的集合运算符也提供了允许重复和不允许重复的两种用法。如果直接使用 UNION 或 INTERSECT，结果里就不会出现重复的行。如果想在结果里留下重复行，可以加上可选项 ALL，写作 UNION ALL。ALL 的作用和 SELECT 子句里的 DISTINCT 可选项刚好相反。

   除了运算结果以外，这两种用法还有一个不同。集合运算符为了排除掉重复行，默认地会发生排序，而加上可选项 ALL 之后，就不会再排序，所以性能会有提升。这是非常有效的用于优化查询性能的方法，所以如果不关心结果是否存在重复行，或者确定结果里不会产生重复行，加上可选项 ALL 会更好些。

2. 集合运算符有优先级

   标准 SQL 规定，INTERSECT 比 UNION 和 EXCEPT 优先级更高。因此，当同时使用 UNION 和 INTERSECT，又想让 UNION 优先执行时，必须用括号明确地指定运算顺序

3. 各个 DBMS 提供商在集合运算的实现程度上参差不齐

4. 除法运算没有标准定义

   四则运算里的和（UNION）、差（EXCEPT）、积（CROSS JOIN）都被引入了标准 SQL。但是很遗憾，商（DIVIDE BY）因为各种原因迟迟没能标准化。

1.2.3 比较表和表：检查集合相等性之基础篇

在迁移数据库的时候，或者需要比较备份数据和最新数据的时候，我们需要调查两张表是否是相等的。这里说的“相等”指的是行数和列数以及内容都相同，即“是同一个集合”的意思。

此时做法有两种。我们先看一个简单的，只用 UNION 就能实现的方法。这里先假设已经事先确认了表 `tbl_A` 和表 `tbl_B` 的行数是一样的（如果行数不一样，那就不需要比较其他的了）。

这种做法的原理是什么呢？请回忆一下导入篇：集合运算的几个注意事项里的注意事项 1。如果集合运算符里不加上可选项 ALL，那么重复行就会被排除掉。因此，如果表 `tbl_A` 和表 `tbl_B` 是相等的，排除掉重复行后，两个集合是完全重合的。

当然，我们也可以只比较表里的一部分列或者一部分行。只需要指定一下想要比较的列的名称，或者在 WHERE 子句里加入过滤条件就可以比较了。