SQL基础

Basic SQL Tutorial

Stephen CUI[1]

November 4, 2022

[1]cuixuanStephen@gmail.com

目录

Chapter 1

集合运算

1.1 表的加减法

1.2 联结（以列为单位对表进行联结）

学习的联结（JOIN）运算，简单来说，就是将其他表中的列添加过来，进行“添加列”的运算。该操作通常用于无法从一张表中获取期望数据（列）的情况。

Chapter 2

SQL高级处理

2.1 窗口函数

2.1.1 窗口函数的语法

```
<function_name> over ([partition by columns_A] order by <columns_B>)
```

其中重要的关键字是partition by和order by。

能够作为窗口函数使用的函数

窗口函数大体可以分为以下两种：

1. 能够作为窗口函数的聚合函数（sum、avg、count、max、min）
2. rank、dense_rank、row_number等专用窗口函数

其中第二种是标准SQL定义的OLAP专用函数

2.1.2 语法的基本使用方法——使用RANK函数

正如其名称所示，RANK是用来计算记录排序的函数。

```
select product_id, product_type, sale_price,
        rank() over (partition by product_type order by sale_price) as ranks
    from product;
```

partition by能够设定排序的对象范围。

order by能够指定按照哪一列、何种顺序进行排序，可以通过关键字ASC/DESC来指定升序或降序，默认升序可省略。

PARTITION BY 在横向上对表进行分组，而 ORDER BY 决定了纵向排序的规则。

窗口函数兼具之前我们学过的GROUP BY子句的分组功能以及 ORDER BY子句的排序功能。但是，PARTITION BY子句并不具备GROUP BY子句的汇总功能。因此，使用 RANK 函数并不会减少原表中记录的行数。

法则

窗口函数兼具分组和排序两种功能。

通过 PARTITION BY 分组后的记录集合称为窗口。此处的窗口并非“窗户”的意思，而是代表范围。

法则

通过 PARTITION BY 分组后的记录集合称为窗口。

此外，各个窗口在定义上绝对不会包含共通的部分。就像刀切蛋糕一样，干净利落。这与通过 GROUP BY 子句分割后的集合具有相同的特征。

2.1.3 无需指定PARTITION BY

使用窗口函数时起到关键作用的是 PARTITION BY 和 GROUP BY。其中，PARTITION BY 并不是必需的，即使不指定也可以正常使用窗口函数。

```
1 select product_id, product_type, sale_price,
2        rank() over (order by sale_price) as ranks
3   from product;
```

2.1.4 专用窗口函数的种类

- RANK()函数

  计算排序时，如果存在相同位次的记录，则会跳过之后的位次。

  e.g. 有 3 条记录排在第 1 位时：1 位、1 位、1 位、4 位···

- DENSE_RANK()函数

  同样是计算排序，即使存在相同位次的记录，也不会跳过之后的位次。

  e.g. 有 3 条记录排在第 1 位时：1 位、1 位、1 位、2 位···

- ROW_NUMBER()函数

  赋予唯一的连续位次。

  e.g. 有 3 条记录排在第 1 位时：1 位、2 位、3 位、4 位···

```
select product_id, product_type, sale_price,
        rank() over (order by sale_price) as ranking,
    dense_rank() over (order by sale_price) as dense_ranking,
    row_number() over (order by sale_price) as row_num
    from product;
```

使用 RANK 或 ROW_NUMBER 时无需任何参数，只需要像 RANK ()或者 ROW_NUMBER() 这样保持括号中为空就可以了。这也是专用窗口函数通常的使用方式，请大家牢记。这一点与作为窗口函数使用的聚合函数有很大的不同

法则

由于专用窗口函数无需参数，因此通常括号中都是空的。

2.1.5 窗口函数的适用范围

目前为止我们学过的函数大部分都没有使用位置的限制，最多也就是在 WHERE 子句中使用聚合函数时会有些注意事项。但是，使用窗口函数的位置却有非常大的限制。更确切地说，窗口函数只能书写在一个特定的位置。

这个位置就是 SELECT 子句之中。反过来说，就是这类函数不能在 WHERE 子句或者 GROUP BY 子句中使用[1]。

法则

原则上窗口函数只能在SELECT子句中使用

下面我们就来简单说明一下其中的理由。

其理由就是，在 DBMS 内部，窗口函数是对 WHERE 子句或者 GROUP BY 子句处理后的“结果”进行的操作。大家仔细想一想就会明白，在得到用户想要的结果之前，即使进行了排序处理，结果也是错误的。在得到排序结果之后，如果通过 WHERE 子句中的条件除去了某些记录，或者使用 GROUP BY 子句进行了汇总处理，那好不容易得到的排序结果也无法使用了[2]。

2.1.6 作为窗口函数使用聚合函数

所有的聚合函数都能用作窗口函数，其语法和专用窗口函数完全相同。

```
select product_id, product_name, sale_price,
        sum(sale_price) over (order by product_id) as current_sum
        from product;
```

[1]语法上，除了SELECT子句，ORDER BY子句或者UPDATE语句的SET子句中也可以使用。但因为几乎没有实际的业务示例，所以开始的时候大家只要记得“只能在SELECT子句中使用”就可以了。

[2]之所以在ORDER BY子句中能够使用窗口函数，是因为 ORDER BY子句会在SELECT 子句之后执行，并且记录保证不会减少。

使用 SUM 函数时，并不像 RANK 或者 ROW_NUMBER 那样括号中的内容为空，而是和之前我们学过的一样，需要在括号内指定作为汇总对象的列。但是窗口函数中的聚合函数默认只统计之前行到当前行的累计聚合值。

```
select product_id, product_name, sale_price,
       avg(sale_price) over (order by product_id) as current_avg,
       sum(sale_price) over (order by product_id) as current_sum
       from product;
```

从结果中我们可以看到，current_avg的计算方法确实是计算平均值的方法，但作为统计对象的却只是“排在自己之上”的记录。像这样以“自身记录（当前记录）”作为基准进行统计，就是将聚合函数当作窗口函数使用时的最大特征。

2.1.7 计算移动平均

窗口函数就是将表以窗口为单位进行分割，并在其中进行排序的函数。其实其中还包含在窗口中指定更加详细的汇总范围的备选功能，该备选功能中的汇总范围称为框架（frame）。

指定框架（汇总范围）

```
select product_id, product_name, sale_price,
       avg(sale_price) over (order by product_id rows 2 preceding) as moving_avg
       from product;

-- Below codes are equivalent to the above shorthand,
-- but it is better to explicitly indicate the scope of the frame,
-- especially when following is used
select product_id, product_name, sale_price,
       avg(sale_price) over (order by product_id
              rows between 2 preceding and current row) as moving_avg
       from product;
```

这里我们使用了 ROWS（“行”）和 PRECEDING（“之前”）两个关键字，将框架指定为“截止到之前n行”，因此ROWS 2 PRECEDING就是将框架指定为“截止到之前2行”，也就是将作为汇总对象的记录限定为如下的“最靠近的3行”。

也就是说，由于框架是根据当前记录来确定的，因此和固定的窗口不同，其范围会随着当前记录的变化而变化。

这样的统计方法称为移动平均（moving average）。由于这种方法在希望实时把握“最近状态”时非常方便，因此常常会应用在对股市趋势的实时跟踪当中。

使用关键字 FOLLOWING（“之后”）也有PRECEDING类似的思想，就可以指定“截止到之后n行”作为框架了。

```
select product_id, product_name, sale_price,
        avg(sale_price) over (order by product_id
                rows between current row and 2 following) as moving_avg
        from product;
```

将当前记录的前后行作为汇总对象

如果希望将当前记录的前后行作为汇总对象时，同时使用 PRECEDING（“之前”）和 FOLLOWING（“之后”）关键字来实现。

```
select product_id, product_name, product_type,
        avg(sale_price) over (order by product_id
                rows between 1 preceding and 1 following) as moving_avg
        from product;
```

2.1.8 两个ORDER BY

最后我们来介绍一下使用窗口函数时与结果形式相关的注意事项，那就是记录的排列顺序。因为使用窗口函数时必须要在 OVER 子句中使用ORDER BY，所以可能有读者乍一看会觉得结果中的记录不会按照该ORDER BY 指定的顺序进行排序。但其实这只是一种错觉。**OVER 子句中的 ORDER BY 只是用来决定窗口函数按照什么样的顺序进行计算的，对结果的排列顺序并没有影响。**

```
-- arbitrary order
select product_id, product_type, sale_price,
        rank() over (order by sale_price) as ranking
        from product;

select product_id, product_type, sale_price,
        rank() over (order by sale_price) as ranking
        from product
        order by ranking;
```

也许大家会觉得在一条 SELECT 语句中使用两次 ORDER BY 会有点别扭，但是尽管这两个 ORDER BY 看上去是相同的，但其实它们的功能却完全不同。

法则

将聚合函数作为窗口函数使用时，会以当前记录为基准来决定汇总对象的记录

2.2 GROUPING运算符

2.2.1 同时得到合计行