收获，不止 SQL 优化

第 十五 章

动手，分析函数让 SQL 飞

E-Mail:45240040@qq.com

目录

1.说说分析函数的 rows 和 range 的差异，请举例说明 2
1.1 rows 和 range 的差异 2
1.2 示例说明 2
1.2.1 环境准备 2
1.2.2 rows 3
1.2.3 range 5
2.说说分析函数的 order by 和普通的 order by 的差异 8
2.1 两者之间的差异 8
2.2 演示两者的差异 8
3.说说分析函数 lag 的应用，有空研究一下最后一个案例 9
3.1 分析函数 lag 的应用 9

1.说说分析函数的 rows 和 range 的差异，请举例说明

1.1 rows 和 range 的差异

rows 和 range 是分析函数中常用到的用来界定计算范围的两个属性,其中 rows 表示物理的窗口范围,按照行的位置计算窗口范围,可以通过 UNBOUNDED PRECEDING 、 UNBOUNDED FOLLOWING 、 CURRENT ROWPRECEDING 、 FOLLOWING 属性来指定,在 order by 子句中默认有一个基于 rows 窗口的开窗子句。Rang 表示逻辑的窗口范围,它是按照单元格的位置及偏移量来计算窗口范围 , 可以通过 UNBOUNDED PRECEDING 、 UNBOUNDED FOLLOWING 、 CURRENT ROW、PRECEDING、FOLLOWING 来指定或计算。

1.2 示例说明

1.2.1 环境准备

```
SQL> CREATE TABLE emp
  2  (emp_id    NUMBER(6),
  3  ename  VARCHAR2(45),
  4  dept_id   NUMBER(4),
  5  hire_date DATE,
  6  sal    NUMBER(8,2) );

Table created.

SQL> INSERT INTO emp (emp_id, ename, dept_id, hire_date, sal)
  2  VALUES (101,  'Tom',   20,  TO_DATE('21-09-1989', 'DD-MM-YYYY'), 2000);

1 row created.

SQL> INSERT INTO emp (emp_id, ename, dept_id, hire_date, sal)
  2  VALUES (102,  'Mike',  20,  TO_DATE('13-01-1993', 'DD-MM-YYYY'), 8000);

1 row created.

SQL> INSERT INTO emp (emp_id, ename, dept_id, hire_date, sal)
  2  VALUES (120,  'John',  50,  TO_DATE('18-07-1996', 'DD-MM-YYYY'), 1000);

1 row created.
```

```
SQL> INSERT INTO emp (emp_id, ename, dept_id, hire_date, sal)
  2  VALUES (121,  'Joy',   50,  TO_DATE('10-04-1997', 'DD-MM-YYYY'), 4000);

1 row created.

SQL> INSERT INTO emp (emp_id, ename, dept_id, hire_date, sal)
  2  VALUES (122,  'RICH',  50,  TO_DATE('01-05-1995', 'DD-MM-YYYY'), 3000);

1 row created.

SQL> INSERT INTO emp (emp_id, ename, dept_id, hire_date, sal)
  2  VALUES (123,  'Kate',  50,  TO_DATE('10-10-1997', 'DD-MM-YYYY'), 5000);

1 row created.

SQL> INSERT INTO emp (emp_id, ename, dept_id, hire_date, sal)
  2  VALUES (124,  'Jess',  50,  TO_DATE('16-11-1999', 'DD-MM-YYYY'), 6000);

1 row created.

SQL> INSERT INTO emp (emp_id, ename, dept_id, hire_date, sal)
  2  VALUES (100,  'Stev',  10,  TO_DATE('01-01-1990', 'DD-MM-YYYY'), 7000);

1 row created.

SQL> commit;

Commit complete.
```

1.2.2 rows

```
SQL> SELECT emp_id,ename,dept_id,hire_date,sal,
  2  SUM(sal) OVER (PARTITION BY dept_id ORDER BY hire_date
  3     ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) sum_1_to_last,1
  4  SUM(sal) OVER (PARTITION BY dept_id ORDER BY hire_date
  5     ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) sum_1_to_cur,2
  6  SUM(sal) OVER (PARTITION BY dept_id ORDER BY hire_date
  7     ROWS BETWEEN UNBOUNDED PRECEDING AND 1/*value_expr*/ PRECEDING) sum_1_to_curbef1,3
  8  SUM(sal) OVER (PARTITION BY dept_id ORDER BY hire_date
  9     ROWS BETWEEN UNBOUNDED PRECEDING AND 1 FOLLOWING) sum_1_to_curaft1 4
 10  FROM emp order by dept_id,hire_date;

    EMP_ID ENAME      DEPT_ID HIRE_DATE       SAL SUM_1_TO_LAST SUM_1_TO_CUR SUM_1_TO_CURBEF1 SUM_1_TO_CURAFT1
---------- ---------- ---------- --------- ---------- ------------- ------------ ---------------- ----------------
       100 Stev               10 01-JAN-90       7000          7000         7000                             7000
       101 Tom                20 21-SEP-89       2000         10000         2000                            10000
       102 Mike               20 13-JAN-93       8000         10000        10000             2000           10000
       122 RICH               50 01-MAY-95       3000         19000         3000                             4000
       120 John               50 18-JUL-96       1000         19000         4000             3000            8000
       121 Joy                50 10-APR-97       4000         19000         8000             4000           13000
       123 Kate               50 10-OCT-97       5000         19000        13000             8000           19000
       124 Jess               50 16-NOV-99       6000         19000        19000            13000           19000
8 rows selected.                                                  1            2                3               4
```

分析上图结果集：

✧ 标号为 1 的分析函数及结果集：

```
SUM(sal) OVER (PARTITION BY dept_id ORDER BY
                                                hire_date
ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
sum_1_to_last,
```

这个分析函数表示了以 dept 为分区条件，以 hire_date 排序，求出从窗口第一行到窗口最后一行的结果和；从结果集中我们可以看到（以部门号为 50 的部门为例），开窗函数没一行的值都是从分区中第一行到最后一行的和，所以结果集都是 3000+1000+4000+5000+6000=19000.

✧ 标号为 2 的分析函数及结果集：

```
SUM(sal) OVER (PARTITION BY dept_id ORDER BY hire_date ROWS
BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) sum_1_to_cur,
```

这个分析函数计算的结果集是当前分区的第一行到当前行的结果集，所以结果集中为：

EMP_ID=122 值为 3000（因为它是当前分区的第一行，所的值就是 3000+0）

EMP_ID=120 值为 4000（3000+1000）

EMP_ID=121 的值为 8000（3000+1000+4000）

EMP_ID=123 的值为 13000 (3000+1000+4000+5000)

EMP_ID=124 的值为 19000 (3000+1000+4000+5000+6000)

✧ 标号为 3 的分析函数及结果集：

```
SUM(sal) OVER (PARTITION BY dept_id ORDER BY hire_date
ROWS BETWEEN UNBOUNDED PRECEDING AND 1/*value_expr*/
PRECEDING) sum_1_to_curbef1,
```

这个分析函数的计算范围是从窗口的第一个位置到本行前一行位置，所以它的结果集为：

EMP_ID=122 值为空（因为它是当前分区的第一行，所以前一行没有值）

EMP_ID=120 值为 3000（前一行为 EMP_ID=122 的值）

EMP_ID=121 的值为 4000（1000+3000）

EMP_ID=123 的值为 8000 (1000+3000+4000)

EMP_ID=124 的值为 13000 (1000+3000+4000+5000)

✧ 标号为 4 的分析函数及结果集：

```
SUM(sal) OVER (PARTITION BY dept_id ORDER BY hire_date ROWS
BETWEEN UNBOUNDED PRECEDING AND 1 FOLLOWING) sum_1_to_curaft1
```

这个分析函数的计算范围是从窗口的第一个位置到本行后一行位置，所以它的结果集为：

EMP_ID=122 值为 4000（3000+1000）

EMP_ID=120 值为 8000（3000+1000+4000）

EMP_ID=121 的值为 13000（3000+1000+4000+5000）

EMP_ID=123 的值为 19000 (3000+1000+4000+5000+6000)

EMP_ID=124 的值为 19000 (因为为最后一行，1000+3000+4000+5000)

1.2.3 range

```
SQL> SELECT emp_id,ename,dept_id,hire_date,sal,
  2  SUM(sal) OVER (PARTITION BY dept_id ORDER BY sal
  3       RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) sum3,
  4  SUM(sal) OVER (PARTITION BY dept_id ORDER BY sal
  5       RANGE BETWEEN CURRENT ROW AND CURRENT ROW) sum4,
  6  SUM(sal) OVER (PARTITION BY dept_id ORDER BY sal
  7       RANGE BETWEEN CURRENT ROW AND 2500/*value_expr*/ FOLLOWING) sum5,
  8  SUM(sal) OVER (PARTITION BY dept_id ORDER BY sal
  9  RANGE BETWEEN 2500/*value_expr*/ PRECEDING AND UNBOUNDED FOLLOWING) sum6
 10  FROM emp;

    EMP_ID ENAME      DEPT_ID HIRE_DATE        SAL       SUM3       SUM4       SUM5       SUM6
---------- ---------- ---------- --------- ---------- ---------- ---------- ---------- ----------
       100 Stev               10 01-JAN-90       7000       7000       7000       7000       7000
       101 Tom                20 21-SEP-89       2000      10000       2000       2000      10000
       102 Mike               20 13-JAN-93       8000       8000       8000       8000       8000
       120 John               50 18-JUL-96       1000      19000       1000       4000      19000
       122 RICH               50 01-MAY-95       3000      18000       3000      12000      19000
       121 Joy                50 10-APR-97       4000      15000       4000      15000      18000
       123 Kate               50 10-OCT-97       5000      11000       5000      11000      18000
       124 Jess               50 16-NOV-99       6000       6000       6000       6000      15000

8 rows selected.
```

分析上图结果集：

✧ Sum3 分析函数及结果集：

```
SUM(sal) OVER (PARTITION BY dept_id ORDER BY sal RANGE
BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) sum3,
```

Sum3 表示的范围是从当前行到最后一行的累计值，所以：

EMP_ID=120 值为 19000（1000+3000+4000+5000+6000）

EMP_ID=122 值为 18000（3000+4000+5000+6000）

EMP_ID=121 的值为 15000（4000+5000+6000）

EMP_ID=123 的值为 11000 (5000+6000)

EMP_ID=124 的值为 6000 (因为为最后一行，值为它本身)

✧ Sum4 分析函数及结果集：

```
SUM(sal) OVER (PARTITION BY dept_id ORDER BY sal RANGE
BETWEEN CURRENT ROW AND CURRENT ROW) sum4,
```

Sum4 表示的范围是从当前行到当前行，其实就是本行的值，与 sal 值一样。

✧ Sum5 分析函数及结果集：

```
SUM(sal) OVER (PARTITION BY dept_id ORDER BY sal RANGE
BETWEEN CURRENT ROW AND 2500/*value_expr*/ FOLLOWING) sum5,
```

Sum5 表示的范围是从当前行到小于本记录 sal 多 2500 的所有的薪资累计，也就是说求出落在当前记录~当前记录+2500 之间的记录之和。

EMP_ID=120 值为 4000

1000+2500=3500 所以在从当前记录开始 1000 到 3500 之间的记录值之和为 1000+3000=4000

EMP_ID=122 值为 12000

3000+2500=5500 所以在从当前记录开始 3000 到 5500 之间的记录值之和为:3000+4000+5000=12000

EMP_ID=121 的值为 15000

4000+2500=6500 所以在从当前记录开始 4000 到 6500 之间的记录值之和为:4000+5000+6000=15000

EMP_ID=123 的值为 11000

5000+2500=7500 所以在当前记录开始 5000 到 7500 之间的记录值之和为：

5000+6000=11000

EMP_ID=124 的值为 6000

6000+2500=8500 因为当前记录为最后一条，所以值为就为其本身。

Sum6 分析函数及结果集：

```
SUM(sal) OVER (PARTITION BY dept_id ORDER BY sal RANGE
BETWEEN 2500/*value_expr*/ PRECEDING AND UNBOUNDED
FOLLOWING) sum6
```

Sum6 表示的范围是当前值减去 2500 到当前区间的最后一行的值得范围之和：

EMP_ID=120 值为 19000

1000-2500=-1500 所以在-1500 到 6000 范围内记录值之和为：

1000+3000+4000+5000+6000=19000

EMP_ID=122 值为 19000

3000-2500=500 所以在 500 到 6000 范围内记录值之和为：

1000+3000+4000+5000+6000=19000

EMP_ID=121 的值为 18000

4000-2500=1500 所以在 1500 到 6000 范围内记录值之和为：

3000+4000+5000+6000=18000

EMP_ID=123 的值为 18000

5000-2500=2500 所以在 2500 到 6000 范围内记录值之和为：

3000+4000+5000+6000=18000

EMP_ID=124 的值为 15000

6000-2500=3500 所以在 3500 到 6000 范围内记录值之和为：

4000+5000+6000=15000

2.说说分析函数的 order by 和普通的 order by 的差异

2.1 两者之间的差异

order by 属性主要用来对指定的列进行排序，在普通的 order by 中排序的对象是整个查询结果集，而在分析函数中的 order by 不仅能够对指定的列进行排序，而且其排序的范围仅是指当前分区内，在使用分析函数中的 order by 时，如果不使用 between … and 指定一个范围时，order by 会默认添加一个开窗子句，其开窗的范围是分区的第一行到分区的最后一行。

2.2 演示两者的差异

接第一题的环境：

```
SELECT
  emp_id,ename,dept_id,hire_date,sal,                    分析函数的order by
  SUM(sal) OVER (PARTITION BY dept_id ORDER BY hire_date) sum_sal1,
  SUM(sal) OVER (PARTITION BY dept_id ORDER BY hire_date DESC) sum_sal2,
  SUM(sal) OVER (PARTITION BY dept_id ORDER BY hire_date DESC nulls LAST) sum_sal3,
  SUM(sal) OVER (PARTITION BY dept_id ) sum_sal4,
  SUM(sal) OVER ( ) sum_sal5
  8    FROM emp;

    EMP_ID ENAME         DEPT_ID HIRE_DATE       SAL   SUM_SAL1   SUM_SAL2   SUM_SAL3   SUM_SAL4   SUM_SAL5
---------- ---------- ---------- --------- ---------- ---------- ---------- ---------- ---------- ----------
       100 Stev               10 01-JAN-90       7000       7000       7000       7000       7000      36000
       101 Tom                20 21-SEP-89       2000       2000      10000      10000      10000      36000
       102 Mike               20 13-JAN-93       8000      10000       8000       8000      10000      36000
       122 RICH               50 01-MAY-95       3000       3000      19000      19000      19000      36000
       120 John               50 18-JUL-96       1000       4000      16000      16000      19000      36000
       121 Joy                50 10-APR-97       4000       8000      15000      15000      19000      36000
       123 Kate               50 10-OCT-97       5000      13000      11000      11000      19000      36000
       124 Jess               50 16-NOV-99       6000      19000       6000       6000      19000      36000

8 rows selected.

SQL> select emp_id,ename,dept_id,hire_date,sal
  2  from emp order by dept_id;                         普通查询的order by

    EMP_ID ENAME         DEPT_ID HIRE_DATE       SAL
---------- ---------- ---------- --------- ----------
       100 Stev               10 01-JAN-90       7000
       101 Tom                20 21-SEP-89       2000
       102 Mike               20 13-JAN-93       8000
       124 Jess               50 16-NOV-99       6000
       123 Kate               50 10-OCT-97       5000
       122 RICH               50 01-MAY-95       3000
       120 John               50 18-JUL-96       1000
       121 Joy                50 10-APR-97       4000

8 rows selected.
```

从上面的图片中可以看出分析函数的 order by 中排序是针对 PARTITION 指

定的分区进行排序，排序被分在了 3 段中分别进行，每一段中是有序的，但整体是没有序的，而在普通查询中 order by 中，可以看到整个一个查询集是一个有序的整体，所以他是对整个结果集进行排序的。

3.说说分析函数 lag 的应用，有空研究一下最后一个案例

3.1 分析函数 lag 的应用

Lag 函数是获取结果集中按照一定排序所排列的当前行正向（向上）偏移量的值，它有三个参数其中第一个参数为列名，第二参数为偏移量，第三个参数是超出记录窗口的默认值。其实 Lag 分析函数在统计报表中应该会经常用到，通过 lag 函数可以在分析报表中展现不同分组情况下与上一个值对比情况，如按照月统计收入，可以在一个报表中展现本部门当前月的收入情况及上一个月收入情况，进而可以得出环比增长或减少率。诸如类似上面的情况求同期与上期及某个时期情况及比率，在一张报表中展示，我觉得都可以用到 lag 函数。

下面我们使用 EMP 表演示 lag 函数的用法：

```
SQL> select empno,deptno,sal,lag(sal,1,sal)
  2        over(partition by deptno order by empno) lag
  3  from emp;

     EMPNO     DEPTNO        SAL        LAG
---------- ---------- ---------- ----------
      7782         10       2450       2450
      7839         10       5000       2450
      7369         20        800        800
      7566         20       2975        800
      7788         20       3000       2975
      7876         20       1100       3000
      7902         20       3000       1100
      7499         30       1600       1600
      7521         30       1250       1600
      7654         30       1250       1250
      7698         30       2850       1250
      7844         30       1500       2850
      7900         30        950       1500

13 rows selected.
```

同部门内员工薪水比较