

【书评:Oracle 查询优化改写】第五至十三章

1.1 BLOG 文档结构图

▲ 【书评:Oracle 查询优化改写】第五至十三章
▲ 1.1 前言部分
1.1.1 导读
1.1.2 实验环境介绍
1.1.3 相关参考文章链接
1.1.4 本文简介
▲ 1.2 第五章部分内容 字符串的处理
1.2.1 遍历字符串
1.2.2 计算字符在字符串中出现的次数
1.2.3 根据表中的行创建一个分隔列表
1.2.4 分解IP 地址
▲ 1.3 第六章部分内容 数字处理
1.3.1 生成累积和
1.3.2 求总和的百分比 RATIO_TO_REPORT
1.4 第七、八、九、十章 日期处理、分析函数
▲ 1.5 第 11 章 行列互转
1.5.1 pivot 函数
1.5.2 unpivot 函数
▲ 1.6 第 12、13 章
1.6.1 用 sql 输出九九乘法表
1.7 总结
1.8 about me

1.2 前言部分

1.2.1 导读

各位技术爱好者，看完本文后，你可以掌握如下的技能，也可以学到一些其它你所不知道的知识，~O(∩_∩)O~：

① 字符串的处理

② 常用分析函数

③ 用 sql 输出九九乘法表

本文如有错误或不完善的地方请大家多多指正，ITPUB 留言或 QQ 皆可，您的批评指正是我写作的最大动力。

1.2.2 实验环境介绍

oracle 11g

1.2.3 相关参考文章链接

前 4 章的链接参考相关连接：

【书评:Oracle 查询优化改写】第一章 <http://blog.itpub.net/26736162/viewspace-1652985/>

【书评:Oracle 查询优化改写】第二章 <http://blog.itpub.net/26736162/viewspace-1654252/>

【书评:Oracle 查询优化改写】第三章 <http://blog.itpub.net/26736162/viewspace-1660422/>

【书评:Oracle 查询优化改写】第四章 <http://blog.itpub.net/26736162/viewspace-1661906/>

行列互转内容链接：<http://blog.itpub.net/26736162/viewspace-1272538/>

1.2.4 本文简介

大家奇怪了，怎么不一章一章的写了，直接跳跃了，小麦苗告诉大家，因为第 5 到 13 章的内容大多数是开发的内容，和 SQL 调优相差太远，这里列出这几章的目录，虽说是开发 sql 的内容，但是很多实例还是比较实用的，比如对 translate 函数的应用。

第 5 章 处理字符串，包含 translate 函数和个别 oracle 的分析函数。

第 6 章 处理数字，介绍了分析函数

第 7、8 章讲了 DATE 类型的常见用法。

第 9 章仍然介绍分析函数

第 10 章的重点是结果集的分页

第 11 章讲述了行列转换函数， 用 UNPIVOT 对 UNION ALL 做一定的优化，还有 ROLLUP 及 CUBE 可以让你少写一些 UNION ALL 语句。

第 12 章讲解树形查询

第 13 章选取了部分网友的需求案例，希望读者能通过这些案例的启发找到实现自己需求的思路。

第 5 章 使用字符串

- 5.1 遍历字符串
- 5.2 字符串文字中包含引号
- 5.3 计算字符在字符串中出现的次数
- 5.4 从字符串中删除不需要的字符
- 5.5 将字符和数字数据分离
- 5.6 查询只包含字母或数字型的数据
- 5.7 提取姓名的大写首字母缩写
- 5.8 按字符串中的数值排序
- 5.9 根据表中的行创建一个分隔列表
- 5.10 提取第 n 个分隔的子串
- 5.11 分解 IP 地址
- 5.12 将分隔数据转换为多值 IN 列表
- 5.13 按字母顺序排列字符串
- 5.14 判别可作为数值的字符串

第 6 章 使用数字

- 6.1 常用聚集函数
- 6.2 生成累计和
- 6.3 计算累计差
- 6.4 更改累计和的值
- 6.5 返回各部门工资排名前三位的员工
- 6.6 计算出现次数最多的值
- 6.7 返回最值所在行数据
- 6.8 first_value
- 6.9 求总和的百分比

第 7 章 日期运算

- 7.1 加减日、月、年
- 7.2 加减时、分、秒
- 7.3 日期间隔之时、分、秒
- 7.4 日期间隔之日、月、年
- 7.5 确定两个日期之间的工作天数
- 7.6 计算一年中周内各日期的次数
- 7.7 确定当前记录和下一条记录之间相差的天数

第 8 章 日期操作

- 8.1 SYSDATE 能得到的信息
- 8.2 INTERVAL
- 8.3 EXTRACT
- 8.4 确定一年是否为闰年

- 8.5 周的计算
- 8.6 确定一年内属于周内某一天的所有日期
- 8.7 确定某月内第一个和最后一个“周内某天”的日期
- 8.8 创建本月日历
- 8.9 全年日历
- 8.10 确定指定年份季度的开始日期和结束日期
- 8.11 补充范围内丢失的值
- 8.12 按照给定的时间单位进行查找
- 8.13 使用日期的特殊部分比较记录
- 8.14 识别重叠的日期范围
- 8.15 按指定间隔汇总数据

第 9 章 范围处理

- 9.1 定位连续值的范围
- 9.2 查找同一组或分区中行之间的差
- 9.3 定位连续值范围的开始点和结束点
- 9.4 合并时间段

第 10 章 高级查找

- 10.1 给结果集分页
- 10.2 重新生成房间号
- 10.3 跳过表中 n 行
- 10.4 排列组合去重
- 10.5 找到包含最大值和最小值的记录

第 11 章 报表和数据仓库运算

- 11.1 行转列
- 11.2 列转行
- 11.3 将结果集反向转置为一列
- 11.4 抑制结果集中的重复值
- 11.5 利用“行转列”进行计算
- 11.6 给数据分组
- 11.7 对数据分组
- 11.8 计算简单的小计
- 11.9 判别非小计的行
- 11.10 计算所有表达式组合的小计
- 11.11 人员在工作间的分布
- 11.12 创建稀疏矩阵
- 11.13 对不同组/分区同时实现聚集
- 11.14 对移动范围的值进行聚集
- 11.15 常用分析函数开窗讲解
- 11.16 listagg 与小九九

第 12 章 分层查询

- 12.1 简单的树形查询
- 12.2 根节点、分支节点、叶子节点
- 12.3 sys_connect_by_path
- 12.4 树形查询中的排序
- 12.5 树形查询中的 WHERE
- 12.6 查询树形的一个分支
- 12.7 剪去一个分支
- 12.8 字段内 list 值去重

第 13 章 应用案例实现

- 13.1 从固定位置提取字符串的元素
- 13.2 搜索字母数字混合的字符串
- 13.3 把结果分级并转为列
- 13.4 构建基础数据的重要性
- 13.5 根据传入条件返回不同列中的数据
- 13.6 拆分字符串进行连接
- 13.7 整理垃圾数据
- 13.8 用“行转列”来得到隐含信息
- 13.9 用隐藏数据进行行转列
- 13.10 用正则表达式提取 clob 里的文本格式记录集

下边我针对不同的章节，选取感兴趣的部分内容分享给大家：

1.3 第五章部分内容 字符串的处理

1.3.1 遍历字符串

```
create or replace view v as
select '天天向上' as 汉字 , 'TTXS' as 首拼  from dual;
```

```
select v.汉字 , v.首拼 , level from v connect by level<=length(v.汉字);
```

汉字	首拼	LEVEL
天天向上	TTXS	1
天天向上	TTXS	2
天天向上	TTXS	3
天天向上	TTXS	4

```
SELECT v.汉字,
       v.首拼,
       LEVEL,
       substr(v.汉字, LEVEL, 1) AS 汉字拆分,
       substr(v.首拼, LEVEL, 1) AS 首拼拆分,
       'substr("'" || v.汉字 || "', " || LEVEL || ',1)' AS fun
FROM   v
CONNECT BY LEVEL <= length(v.汉字);
```

汉字	首拼	LEVEL	汉字拆分	首拼拆分	FUN
天天向上	TTXS	1	天	T	substr('天天向上',1,1) ...
天天向上	TTXS	2	天	T	substr('天天向上',2,1) ...
天天向上	TTXS	3	向	X	substr('天天向上',3,1) ...
天天向上	TTXS	4	上	S	substr('天天向上',4,1) ...

1.3.2 计算字符在字符串中出现的次数

```
create or replace view v as
select 'CLARK,KING,MILLER' as str from dual;
```

---可以有多种办法：

```
select REGEXP_COUNT(str,',')+1 as cnt from v;
select length(REGEXP_replace(str,'[^,]'))+1 as cnt from v;
```

```
select length(translate(str,','||str,','))+1 as cnt from v;
```

```
create or replace view v as
select 'CLARK$#KING$#MILLER' as str from dual;
```

```
select REGEXP_COUNT(str,'$#')+1 as cnt from v;
select length(translate(str,'$#'||str,'$#'))/length('$#')+1 as cnt from v;
```

另外也可以自己编写函数，：

```
FUNCTION fun_getSpecharcounts_lhr(p_string IN VARCHAR2) RETURN NUMBER AS
    v_count    NUMBER := 0;
    v_position NUMBER := 0; --特殊字符的位置
    v_Spechar   VARCHAR2(5) := '/';
BEGIN
    LOOP
        --找到特殊字符的位置
        SELECT instr(p_string, v_Spechar, v_position+1)
        INTO   v_position
        FROM   dual;

        IF v_position = 0 OR (v_position >= length(p_string)) THEN
            EXIT;
        END IF;

        v_count := v_count + 1;
    END LOOP;

    RETURN v_count;
END fun_getSpecharcounts_lhr;
```

1.3.3 根据表中的行创建一个分隔列表

```
SELECT a.deptno,
       SUM(a.sal) AS total_sal,
       listagg(a.ename, ',') within GROUP(ORDER BY ename) AS total_ename,
       wmsys.wm_concat(a.ename) ,
       to_char(wmsys.wm_concat(a.ename))
FROM   scott.emp a
```

DEPTNO	TOTAL_SAL	TOTAL_ENAME	WMSYS.WM_CONCAT(A.ENAME)	TO_CHAR(WMSYS.WM_CONCAT(A.ENAM
10	38750	CLARK,KING,MILLER	<CLOB>	CLARK,MILLER,KING
20	110875	ADAMS,FORD,JONES,SCOTT,SMITH	<CLOB>	SMITH,FORD,ADAMS,SCOTT,JONES
30	189400	ALLEN,BLAKE,JAMES,MARTIN,TURNER,WARD	<CLOB>	ALLEN,JAMES,TURNER,BLAKE,MARTIN,WARD
		test	<CLOB>	test

1. 3. 4 分解 IP 地址

```
SELECT regexp_substr(v.ip, '[^.]+' , 1, 1) a,
       regexp_substr(v.ip, '[^.]+' , 1, 2) b,
       regexp_substr(v.ip, '[^.]+' , 1, 3) c,
       regexp_substr(v.ip, '[^.]+' , 1, 4) d
FROM   (SELECT '192.168.59.130' ip FROM DUAL) v;
```

A	B	C	D
192	168	59	130

1. 4 第六章部分内容 数字处理

1. 4. 1 生成累积和

```
SELECT manager_id,
       last_name,
       salary,
       SUM(salary) OVER(PARTITION BY manager_id ORDER BY salary,employee_id ) l_csum,
       SUM(salary) OVER(PARTITION BY manager_id ORDER BY salary,employee_id RANGE UNBOUNDED PRECEDING) l_csum,
       SUM(salary) OVER(PARTITION BY manager_id ORDER BY salary,employee_id RANGE between UNBOUNDED PRECEDING and current row ) l_csum,
       SUM(salary) OVER(PARTITION BY manager_id ORDER BY salary,employee_id rows between UNBOUNDED PRECEDING and current row ) l_csum ,
       (SELECT listagg(b.salary, '+') within GROUP(ORDER BY salary,employee_id)
        FROM   hr.employees b
        WHERE  b.manager_id IN (101, 103, 108)
        AND    b.manager_id = t.manager_id
        AND    b.salary <=  t.salary )
FROM   hr.employees t
WHERE  manager_id IN (101, 103, 108);
```


MANAGER_ID	LAST_NAME	SALARY	L_CSUM	(SELECTLISTAGG(B.SALARY,'')WI
101	Whalen	4400.00	4400	4400
101	Mavris	6500.00	10900	4400+6500
101	Baer	10000.00	20900	4400+6500+10000
101	Greenberg	12008.00	32908	4400+6500+10000+12008+12008
101	Higgins	12008.00	44916	4400+6500+10000+12008+12008
103	Lorentz	4200.00	4200	4200
103	Austin	4800.00	9000	4200+4800+4800
103	Pataballa	4800.00	13800	4200+4800+4800
103	Ernst	6000.00	19800	4200+4800+4800+6000
108	Popp	6900.00	6900	6900
108	Sciarra	7700.00	14600	6900+7700
108	Urman	7800.00	22400	6900+7700+7800
108	Chen	8200.00	30600	6900+7700+7800+8200
108	Faviet	9000.00	39600	6900+7700+7800+8200+9000

1.4.2 求总和的百分比 RATIO_TO_REPORT

```
create table T_salary(F_depart varchar2(20),F_EMP varchar2(20),F_salary integer );
truncate table t_salary;
--
插入测试数据
insert into t_salary(f_depart, f_emp, f_salary)
select '信息管理部','张三',10000 from dual union all
select '信息管理部','李四',2000 from dual union all
select '人力资源部','王五',3000 from dual union all
select '人力资源部','赵六',10000 from dual;
commit;
```

```
select * from t_salary;
```

F_DEPART	F_EMP	F_SALARY
信息管理部	张三	10000
信息管理部	李四	2000
人力资源部	王五	3000
人力资源部	赵六	10000

--查询每个员工占所在部门的工资比例

```
SELECT f_depart,
       f_emp,
       f_salary,
       SUM(f_salary) over(PARTITION BY f_depart) sum_salary,
       ratio_to_report(f_salary) over(PARTITION BY f_depart) ratio_salary
FROM   t_salary;
```

F_DEPART	F_EMP	F_SALARY	SUM_SALARY	RATIO_SALARY
人力资源部	王五	3000	13000	0.230769230769231
人力资源部	赵六	10000	13000	0.769230769230769
信息管理部	张三	10000	12000	0.833333333333333
信息管理部	李四	2000	12000	0.166666666666667

--递归查询员工占所在部门的百分比，以及部门所占公司的工资比例。

```
SELECT f_depart,
       f_emp,
       f_salary,
```



```
g1,
SUM(f_salary) over(PARTITION BY decode(g1, 0, f_depart, NULL), g1) sum_salary,
ratio_to_report(f_salary) over(PARTITION BY decode(g1, 0, f_depart, NULL), g1) r_salary
FROM (SELECT f_depart,
f_emp,
SUM(f_salary) f_salary,
GROUPING(f_depart) + GROUPING(F_emp) g1
FROM t_salary
GROUP BY ROLLUP(f_depart, f_emp)) t;
```

F_DEPART	F_EMP	F_SALARY	G1	SUM_SALARY	R_SALARY
人力资源部	王五	3000	0	13000	0.230769230769231
人力资源部	赵六	10000	0	13000	0.769230769230769
信息管理部	李四	2000	0	12000	0.166666666666667
信息管理部	张三	10000	0	12000	0.833333333333333
信息管理部		12000	1	25000	0.48
人力资源部		13000	1	25000	0.52
		25000	2	25000	1

由于分析函数可以使用普通函数的结果作为expr参数，所以上面的代码又可以整合为下述方式。

```
SELECT f_depart,
f_emp,
SUM(f_salary) f_salary,
SUM(SUM(f_salary)) over(PARTITION BY decode(GROUPING(f_depart) + GROUPING(F_emp), 0, f_depart, NULL), GROUPING(f_depart) +
GROUPING(F_emp)) sum_salary,
ratio_to_report(SUM(f_salary)) over(PARTITION BY decode(GROUPING(f_depart) + GROUPING(F_emp), 0, f_depart, NULL),
GROUPING(f_depart) + GROUPING(F_emp)) r_salary,
GROUPING(f_depart) + GROUPING(F_emp) g1
FROM t_salary
GROUP BY ROLLUP(f_depart, f_emp);
```

F_DEPART	F_EMP	F_SALARY	SUM_SALARY	R_SALARY	G1
人力资源部	王五	3000	13000	0.230769230769231	0
人力资源部	赵六	10000	13000	0.769230769230769	0
信息管理部	李四	2000	12000	0.166666666666667	0
信息管理部	张三	10000	12000	0.833333333333333	0
信息管理部		12000	25000	0.48	1
人力资源部		13000	25000	0.52	1
		25000	25000	1	2

1.5 第七、八、九、十章 日期处理、分析函数

这个第 7、8 章就是所有的日期函数的处理，基础内容，没啥分享的，第 9 和 10 章是继续分析函数。

1.6 第 11 章 行列互转

关于该章我之前分析过我整理的内容，参考：<http://blog.itpub.net/26736162/viewspace-1272538/>

这里我再增加一些内容，就是关于 pivot 和 unpivot 分析函数的应用。

1.6.1 pivot 函数

```
SELECT *
FROM (SELECT e.job,
            e.sal,
            e.deptno
      FROM scott.emp e)
pivot(SUM(sal) AS s
FOR deptno IN(10 AS d10, 20, 30 AS d30))
ORDER BY 1;
```

	JOB	D10_S	20_S	D30_S
1	ANALYST		3000	
2	CLERK	1300	800	950
3	MANAGER	2450	2975	2850
4	PRESIDENT	5000		
5	SALESMAN			5600

---增加一个

```
SELECT *
FROM (SELECT e.job,
            e.sal,
            e.deptno ,
            comm
      FROM scott.emp e)
pivot(SUM(sal) AS s,sum(comm) as c
FOR deptno IN(10 AS d10, 20 as d20, 30 AS d30))
ORDER BY 1;
```

	JOB	D10_S	D10_C	D20_S	D20_C	D30_S	D30_C
1	ANALYST			3000			
2	CLERK	1300		800		950	
3	MANAGER	2450		2975		2850	
4	PRESIDENT	5000					
5	SALESMAN					5600	2200

1. 6. 2 unpivot 函数

```
drop table test purge;
create table test as
SELECT *
FROM   (SELECT e.deptno,
               e.sal
        FROM   scott.emp e)
pivot(COUNT(*) AS cnt, SUM(sal) AS s
FOR    deptno IN(10 AS d10, 20 AS d20, 30 AS d30))
ORDER BY 1;
```

```
SELECT * FROM test();
```

```
SELECT * FROM test unpivot(人次 FOR deptno IN(d10_cnt, d20_cnt, d30_cnt));
```

	D10_S	D20_S	D30_S	DEPTNO	人次
1	8750	6775	9400	D10_CNT	3
2	8750	6775	9400	D20_CNT	3
3	8750	6775	9400	D30_CNT	6

```
SELECT deptno AS 部门编码,
       人次,
       工资
FROM   test a
unpivot include nulls (人次 FOR deptno IN(d10_cnt as 10, d20_cnt as 20, d30_cnt as 30))
unpivot include nulls (工资 FOR deptno2 IN(d10_s as 10, d20_s as 20, d30_s as 30))
where  deptno= deptno2
;
```

	部门编码	人次	工资
► 1	10	3	8750
2	20	3	6775
3	30	6	9400

1. 7 第 12、13 章

1. 7. 1 用 sql 输出九九乘法表

这个只能给个链接了，因为这个帖子上的太经典了：<http://www.itpub.net/thread-762215-1-1.html>

这里截取几个例子：

```
select r1 || ' * ' || 1 || ' = ' || r1 * 1 A,
decode(r2, '', '', r2 || ' * ' || 2 || ' = ' || r2 * 2) b,
decode(r3, '', '', r3 || ' * ' || 3 || ' = ' || r3 * 3) C,
decode(r4, '', '', r4 || ' * ' || 4 || ' = ' || r4 * 4) D,
decode(r5, '', '', r5 || ' * ' || 5 || ' = ' || r5 * 5) E,
decode(r6, '', '', r6 || ' * ' || 6 || ' = ' || r6 * 6) F,
decode(r7, '', '', r7 || ' * ' || 7 || ' = ' || r7 * 7) G,
decode(r8, '', '', r8 || ' * ' || 8 || ' = ' || r8 * 8) H,
decode(r9, '', '', r9 || ' * ' || 9 || ' = ' || r9 * 9) I
from (select level r1,
lag(level+1, 1) over(order by level) r2,
lag(level+2, 2) over(order by level) r3,
lag(level+3, 3) over(order by level) r4,
lag(level+4, 4) over(order by level) r5,
lag(level+5, 5) over(order by level) r6,
lag(level+6, 6) over(order by level) r7,
lag(level+7, 7) over(order by level) r8,
lag(level+8, 8) over(order by level) r9
from dual
connect by level < 10);

SELECT rn,
       ltrim(MAX(sys_connect_by_path(product, ' ')), ' ') product
FROM   (SELECT rn,
               product,
               MIN(product) over(PARTITION BY rn) product_min,
               (row_number() over(ORDER BY rn, product)) +
               (dense_rank() over(ORDER BY rn)) numId
        FROM   (SELECT b.rn,
                       a.rn || ' * ' || b.rn || ' = ' || a.rn * b.rn product
        FROM     (SELECT rownum rn FROM all_objects WHERE rownum <= 9) a,
                  (SELECT rownum rn FROM all_objects WHERE rownum <= 9) b
        WHERE    a.rn <= b.rn
        ORDER   BY b.rn,
                  product))
START  WITH product = product_min
CONNECT BY numId - 1 = PRIOR numId
GROUP  BY rn
ORDER  BY rn;

select replace(reverse(sys_connect_by_path(reverse(rownum || ' * ' || lv || ' = ' || rpad(rownum * lv, 2)), '/' )), '/' )
       from (select level lv from dual connect by level < 10)
where lv = 1
connect by lv + 1 = prior lv;

select ltrim(sys_connect_by_path
       (rownum - rn1+1 || ' * ' || rownum || ' = ' || rpad(rownum * (rownum - rn1+1), 2) , ' '))
       from
       (select rownum rn1 from dual connect by rownum <=9)
where rn1 = 1
       connect by rn1+1 = prior rn1;

with t as (select level as n from dual connect by level <=9)
```

```
select max(substr(sys_connect_by_path(b.n || '*' || a.n || '=' || a.n * b.n, ','),3)) as val
  from t a, t b
 where a.n >= b.n
    start with b.n=1
 connect by a.n=prior a.n and b.n=prior b.n+1
 group by a.n
        order by val
;
```

	VAL
1	1*1=1
2	1*2=2, 2*2=4
3	1*3=3, 2*3=6, 3*3=9
4	1*4=4, 2*4=8, 3*4=12, 4*4=16
5	1*5=5, 2*5=10, 3*5=15, 4*5=20, 5*5=25
6	1*6=6, 2*6=12, 3*6=18, 4*6=24, 5*6=30, 6*6=36
7	1*7=7, 2*7=14, 3*7=21, 4*7=28, 5*7=35, 6*7=42, 7*7=49
8	1*8=8, 2*8=16, 3*8=24, 4*8=32, 5*8=40, 6*8=48, 7*8=56, 8*8=64
9	1*9=9, 2*9=18, 3*9=27, 4*9=36, 5*9=45, 6*9=54, 7*9=63, 8*9=72, 9*9=81

1.8 总结

到此 SQL 查询优化改写第 5-13 章基本 over，重点是对分析函数的领悟和掌握，希望对做 SQL 优化的童鞋有所帮助。

1.9 about me

本文作者：小麦苗，只专注于数据库的技术，更注重技术的运用

ITPUB BLOG：<http://blog.itpub.net/26736162>

本文地址：<http://blog.itpub.net/26736162/viewspace-1665934/>

本文pdf版：<http://yunpan.cn/QCwUAI9bn7g7w> 提取码：af2d

QQ：642808185 若加 QQ 请注明你所正在读的文章标题

创作时间地点：2015-05-21 09:00~ 2015-05-21 18:00 于外汇交易中心

<版权所有，文章允许转载，但须以链接方式注明源地址，否则追究法律责任!>

.....

