第 1 章　oracle 行列互换总结

1.1　概述

最近论坛很多人提的问题都与行列转换有关系，所以我对行列转换的相关知识做了一个总结，希望对大家有所帮助，同时有何错疏，恳请大家指出，我也是在写作过程中学习，算是一起和大家学习吧！

行列转换包括以下六种情况：

1)　列转行

2)　行转列

3)　多列转换成字符串

4)　多行转换成字符串

5)　字符串转换成多列

6)　字符串转换成多行

下面分别进行举例介绍。

首先声明一点，有些例子需要如下 10g 及以后才有的知识：

A.　掌握 model 子句

B.　正则表达式

C.　加强的层次查询

讨论的适用范围只包括 8i,9i,10g 及以后版本。

1.2　列转行

首先需要明白什么是列转行，简单的说就是将原表中的列名作为转换后的表的内容，这就是列转行。

```
CREATE TABLE t_col_row(
ID INT,
```

```
c1 VARCHAR2(10),
c2 VARCHAR2(10),
c3 VARCHAR2(10));

INSERT INTO t_col_row VALUES (1, 'v11', 'v21', 'v31');
INSERT INTO t_col_row VALUES (2, 'v12', 'v22', NULL);
INSERT INTO t_col_row VALUES (3, 'v13', NULL, 'v33');
INSERT INTO t_col_row VALUES (4, NULL, 'v24', 'v34');
INSERT INTO t_col_row VALUES (5, 'v15', NULL, NULL);
INSERT INTO t_col_row VALUES (6, NULL, NULL, 'v35');
INSERT INTO t_col_row VALUES (7, NULL, NULL, NULL);
COMMIT;

SELECT * FROM t_col_row;
```

ID	C1	C2	C3
1	v11	v21	v31
2	v12	v22	
3	v13		v33
4		v24	v34
5	v15		
6			v35
7			

1.2.1 UNION ALL ---主要方法

适用范围：8i,9i,10g 及以后版本

1.2.1.1 例一

```
SELECT id, 'c1' cn, c1 cv
  FROM t_col_row
UNION ALL
SELECT id, 'c2' cn, c2 cv
```

```
  FROM t_col_row
UNION ALL
SELECT id, 'c3' cn, c3 cv
  FROM t_col_row;
```

ID	CN	CV
1	c1	v11
2	c1	v12
3	c1	v13
4	c1	
5	c1	v15
6	c1	
7	c1	
1	c2	v21
2	c2	v22
3	c2	
4	c2	v24
5	c2	
6	c2	
7	c2	
1	c3	v31
2	c3	
3	c3	v33
4	c3	v34
5	c3	
6	c3	v35
7	c3	

若空行不需要转换，只需加一个 where 条件，

WHERE COLUMN IS NOT NULL 即可。

如：

```
SELECT id, 'c1' cn, c1 cv
  FROM t_col_row
  where c1   is not null
UNION ALL
SELECT id, 'c2' cn, c2 cv
  FROM t_col_row
  where c2 is not null
UNION ALL
SELECT id, 'c3' cn, c3 cv
  FROM t_col_row
```

```
  where c3 is not null;
```

ID	CN	CV
1	c1	v11
2	c1	v12
3	c1	v13
5	c1	v15
1	c2	v21
2	c2	v22
4	c2	v24
1	c3	v31
3	c3	v33
4	c3	v34
6	c3	v35

1.2.1.2 例二

```
create table TEST_LHR
(
  NAME     VARCHAR2(255),
  JANUARY  NUMBER(18),
  FEBRUARY NUMBER(18),
  MARCH    NUMBER(18),
  APRIL    NUMBER(18),
  MAY      NUMBER(18)
)
insert into TEST_LHR (NAME, JANUARY, FEBRUARY, MARCH, APRIL, MAY)
values ('长寿', 58, 12, 26, 18, 269);
insert into TEST_LHR (NAME, JANUARY, FEBRUARY, MARCH, APRIL, MAY)
values ('璧山', 33, 18, 17, 16, 206);
insert into TEST_LHR (NAME, JANUARY, FEBRUARY, MARCH, APRIL, MAY)
values ('杨家坪', 72, 73, 79, 386, 327);
insert into TEST_LHR (NAME, JANUARY, FEBRUARY, MARCH, APRIL, MAY)
values ('巫溪', 34, 9, 7, 21, 33);
insert into TEST_LHR (NAME, JANUARY, FEBRUARY, MARCH, APRIL, MAY)
values ('丰都', 62, 46, 39, 36, 91);
```

```
insert into TEST_LHR (NAME, JANUARY, FEBRUARY, MARCH, APRIL, MAY)
values ('武隆', 136, 86, 44, 52, 142);
commit;
```

NAME	JANUARY	FEBRUARY	MARCH	APRIL	MAY
长寿	58	12	26	18	269
璧山	33	18	17	16	206
杨家坪	72	73	79	386	327
巫溪	34	9	7	21	33
丰都	62	46	39	36	91
武隆	136	86	44	52	142

```
SELECT *
FROM   (SELECT t.name,
               'january' MONTH,
               t.january v_num
        FROM   TEST_LHR t
        UNION ALL
        SELECT t.name,
               'february' MONTH,
               t.february v_num
        FROM   TEST_LHR t
        UNION ALL
        SELECT t.name,
               'march' MONTH,
               t.march v_num
        FROM   TEST_LHR t
        UNION ALL
        SELECT t.name,
               'april' MONTH,
               t.april v_num
        FROM   TEST_LHR t
        UNION ALL
        SELECT t.name,
               'may' MONTH,
               t.may v_num
```

```
       FROM   TEST_LHR t)
ORDER  BY NAME;
```

NAME	MONTH	V_NUM
长寿	may	269
长寿	april	18
长寿	february	12
长寿	march	26
长寿	january	58
丰都	may	91
丰都	march	39
丰都	april	36
丰都	february	46
丰都	january	62
巫溪	february	9
巫溪	may	33
巫溪	january	34
巫溪	april	21
巫溪	march	7
武隆	april	52
武隆	january	136
武隆	march	44
武隆	may	142
武隆	february	86
杨家坪	april	386
杨家坪	february	73
杨家坪	january	72
杨家坪	march	79
杨家坪	may	327
璧山	may	206
璧山	january	33
璧山	february	18
璧山	march	17
璧山	april	16

1.2.2 insert all into ... select

首先创建需要的表，test_lhr1，

```
SQL> desc test_lhr1
Name  Type         Nullable Default Comments
----- ------------ -------- ------- --------
NAME  VARCHAR2(255) Y
MONTH VARCHAR2(8)   Y
V_NUM NUMBER(18)    Y
```

然后执行下边的 sql 语句：

```
insert all
into test_lhr1(NAME,month,v_num) values(name, 'may', may)
```

```
into test_lhr1(NAME,month,v_num) values(name, 'april', april)
into test_lhr1(NAME,month,v_num) values(name, 'february', february)
into test_lhr1(NAME,month,v_num) values(name, 'march', march)
into test_lhr1(NAME,month,v_num) values(name, 'january', january)
select t.name,t.january,t.february,t.march,t.april,t.may from test_lhr t;
commit;
```

别忘记 commit 操作，然后再查询 test_lhr1，发现表中的数据就是列转成行了。

```
select * from test_lhr1;
```

NAME	MONTH	V_NUM
长寿	april	18
长寿	january	58
长寿	february	12
长寿	may	269
长寿	march	26
丰都	april	36
丰都	march	39
丰都	february	46
丰都	january	62
丰都	may	91
巫溪	march	7
巫溪	february	9
巫溪	may	33
巫溪	january	34
巫溪	april	21
武隆	january	136
武隆	february	86
武隆	april	52
武隆	may	142
武隆	march	44
杨家坪	january	72
杨家坪	may	327
杨家坪	april	386
杨家坪	february	73
杨家坪	march	79
璧山	april	16
璧山	may	206
璧山	march	17
璧山	february	18
璧山	january	33

1.2.3 MODEL

适用范围：10g 及以后

```
SELECT id,
       cn,
```

```
       cv
FROM   t_col_row
MODEL RETURN
   UPDATED ROWS PARTITION BY(ID)
   DIMENSION BY(0 AS n)
    MEASURES('xx' AS cn, 'yyy' AS cv, c1, c2, c3)
    RULES UPSERT ALL(cn[1] = 'c1', cn[2] = 'c2', cn[3] = 'c3', cv[1] = c1[0],
cv[2] = c2[0], cv[3] = c3[0])
ORDER  BY ID,
         cn;
```

ID	CN	CV
1	c1	v11
1	c2	v21
1	c3	v31
2	c1	v12
2	c2	v22
2	c3	
3	c1	v13
3	c2	
3	c3	v33
4	c1	
4	c2	v24
4	c3	v34
5	c1	v15
5	c2	
5	c3	
6	c1	
6	c2	
6	c3	v35
7	c1	
7	c2	
7	c3	

1.2.4 COLLECTION

适用范围：8i,9i,10g 及以后版本

要创建一个对象和一个集合：

```
CREATE TYPE cv_pair AS OBJECT(cn VARCHAR2(10),cv VARCHAR2(10));

CREATE TYPE cv_varr AS VARRAY(8) OF cv_pair;
```

```
SELECT id,
       t.cn AS cn,
       t.cv AS cv
FROM   t_col_row,
       TABLE(cv_varr(cv_pair('c1', t_col_row.c1),
                     cv_pair('c2', t_col_row.c2),
                     cv_pair('c3', t_col_row.c3))) t
ORDER  BY 1,
          2;
```

ID	CN	CV
1	c1	v11
1	c2	v21
1	c3	v31
2	c1	v12
2	c2	v22
2	c3	
3	c1	v13
3	c2	
3	c3	v33
4	c1	
4	c2	v24
4	c3	v34
5	c1	v15
5	c2	
5	c3	
6	c1	
6	c2	
6	c3	v35
7	c1	
7	c2	
7	c3	

1.3 行转列

行转列就是将行数据内容作为列名。

```
CREATE TABLE t_row_col AS
SELECT id, 'c1' cn, c1 cv
  FROM t_col_row
UNION ALL
```

```
SELECT id, 'c2' cn, c2 cv
  FROM t_col_row
UNION ALL
SELECT id, 'c3' cn, c3 cv FROM t_col_row;

SELECT * FROM t_row_col ORDER BY 1,2;
```

ID	CN	CV
1	c1	v11
1	c2	v21
1	c3	v31
2	c1	v12
2	c2	v22
2	c3	
3	c1	v13
3	c2	
3	c3	v33
4	c1	
4	c2	v24
4	c3	v34
5	c1	v15
5	c2	
5	c3	
6	c1	
6	c2	
6	c3	v35
7	c1	
7	c2	
7	c3	

1.3.1 AGGREGATE FUNCTION(max+decode) ---主要方法

用聚集函数来实现，适用范围：8i,9i,10g 及以后版本

1.3.1.1 例一

```
SELECT id,
       MAX(decode(cn, 'c1', cv, NULL)) AS c1,
       MAX(decode(cn, 'c2', cv, NULL)) AS c2,
       MAX(decode(cn, 'c3', cv, NULL)) AS c3
FROM   t_row_col
GROUP  BY id
ORDER  BY 1;
```

ID	C1	C2	C3
1	v11	v21	v31
2	v12	v22	
3	v13		v33
4		v24	v34
5	v15		
6			v35
7			

☞ **注意：**

1. MAX 聚集函数也可以用 sum、min、avg 等其他聚集函数替代。
2. 被指定的**转置列**只能有一列，但**固定的列**可以有多列，如果转置列有多列可以有 2 种办法解决，① 采用 1.3.2 创建临时表的方式

   ② 可以先转为单列，单列采用字符串的格式，然后将单列转换为多列

请看下面的例子：

```
SELECT mgr,
       deptno,
       empno,
       ename
FROM   emp
ORDER  BY 1,
          2;
```

MGR	DEPTNO	EMPNO	ENAME
7566	20	7788	SCOTT
7566	20	7902	FORD
7698	30	7844	TURNER
7698	30	7654	MARTIN
7698	30	7521	WARD
7698	30	7499	ALLEN
7698	30	7900	JAMES
7782	10	7934	MILLER
7788	20	7876	ADAMS
7839	10	7782	CLARK
7839	20	7566	JONES
7839	30	7698	BLAKE
7902	20	7369	SMITH
	10	7839	KING

```
SELECT mgr,
```

```
      deptno,
      MAX(decode(empno, '7788', ename, NULL)) "7788",
      MAX(decode(empno, '7902', ename, NULL)) "7902",
      MAX(decode(empno, '7844', ename, NULL)) "7844",
      MAX(decode(empno, '7521', ename, NULL)) "7521",
      MAX(decode(empno, '7900', ename, NULL)) "7900",
      MAX(decode(empno, '7499', ename, NULL)) "7499",
      MAX(decode(empno, '7654', ename, NULL)) "7654"
  FROM emp
  WHERE mgr IN (7566, 7698)
     AND deptno IN (20, 30)
  GROUP BY mgr, deptno
  ORDER BY 1, 2;
```

MGR	DEPTNO	7788	7902	7844	7521	7900	7499	7654
7566	20	SCOTT	FORD					
7698	30			TURNER	WARD	JAMES	ALLEN	MARTIN

这里转置列为 empno，固定列为 mgr，deptno。

1、固定列数的行列转换

如

student subject grade

---

student1 语文 80

student1 数学 70

student1 英语 60

student2 语文 90

student2 数学 80

student2 英语 100

……

转换为

语文 数学 英语

student1 80 70 60

student2 90 80 100

……

语句如下：

```
select student,sum(decode(subject,'语文', grade,null)) "语文",
sum(decode(subject,'数学', grade,null)) "数学",
sum(decode(subject,'英语', grade,null)) "英语"
from table
group by student
```

2、不定列行列转换

如

```
c1 c2
--------------
1 我
1 是
1 谁
2 知
2 道
3 不
……
```

转换为

```
1 我是谁
2 知道
3 不
```

这一类型的转换必须借助于 PL/SQL 来完成，这里给一个例子

```
CREATE OR REPLACE FUNCTION get_c2(tmp_c1 NUMBER)
RETURN VARCHAR2
IS
Col_c2 VARCHAR2(4000);
BEGIN
FOR cur IN (SELECT c2 FROM t WHERE c1=tmp_c1) LOOP
```

```
Col_c2 := Col_c2||cur.c2;
END LOOP;
Col_c2 := rtrim(Col_c2,1);
RETURN Col_c2;
END;
/
```

SQL> select distinct c1 ,get_c2(c1) cc2 from table;即可

还有一种行转列的方式，就是相同组中的行值变为单个列值，但转置的行值不变为列名：

```
ID      CN_1    CV_1    CN_2     CV_2    CN_3   CV_3
1   c1   v11   c2   v21   c3   v31
2   c1   v12   c2   v22   c3
3   c1   v13   c2     c3   v33
4   c1     c2   v24   c3   v34
5   c1   v15   c2     c3
6   c1     c2     c3   v35
7   c1     c2     c3
```

这种情况可以用分析函数实现：

```
SELECT id,
       MAX(decode(rn, 1, cn, NULL)) cn_1,
       MAX(decode(rn, 1, cv, NULL)) cv_1,
       MAX(decode(rn, 2, cn, NULL)) cn_2,
       MAX(decode(rn, 2, cv, NULL)) cv_2,
       MAX(decode(rn, 3, cn, NULL)) cn_3,
       MAX(decode(rn, 3, cv, NULL)) cv_3
  FROM (SELECT id,
               cn,
               cv,
```

```
        row_number() over(PARTITION BY id ORDER BY cn, cv) rn
    FROM t_row_col)
GROUP BY ID;
```

1.3.1.2 例二

```
SELECT t.name,
       MAX(decode(t.month, 'may', t.v_num)) AS may,
       MAX(decode(t.month, 'april', t.v_num)) AS april,
       MAX(decode(t.month, 'february', t.v_num)) AS february,
       MAX(decode(t.month, 'march', t.v_num)) AS march,
       MAX(decode(t.month, 'january', t.v_num)) AS january
FROM   test_lhr1 t
GROUP  BY t.name;
```

NAME	MAY	APRIL	FEBRUARY	MARCH	JANUARY
长寿	269	18	12	26	58
丰都	91	36	46	39	62
巫溪	33	21	9	7	34
武隆	142	52	86	44	136
杨家坪	327	386	73	79	72
璧山	206	16	18	17	33

1.3.1.3 延伸

如果要实现对各个不同的区间进行统计，则：

```
SELECT *
FROM   test_lhr1 t
ORDER  BY t.name,
          t.month;
```

NAME	MONTH	V_NUM
长寿	april	18
长寿	february	12
长寿	january	58
长寿	march	26
长寿	may	269
丰都	april	36
丰都	february	46
丰都	january	62
丰都	march	39
丰都	may	91
巫溪	april	21
巫溪	february	9
巫溪	january	34
巫溪	march	7
巫溪	may	33
武隆	april	52
武隆	february	86
武隆	january	136
武隆	march	44
武隆	may	142
杨家坪	april	386
杨家坪	february	73
杨家坪	january	72
杨家坪	march	79
杨家坪	may	327
璧山	april	16
璧山	february	18
璧山	january	33
璧山	march	17
璧山	may	206

```
SELECT t.name,
       CASE
           WHEN t.v_num < 100 THEN
            '0-100'
           WHEN t.v_num >= 100 AND t.v_num < 200 THEN
            '100-200'
           WHEN t.v_num >= 200 AND t.v_num < 300 THEN
            '200-300'
           WHEN t.v_num >= 300 AND t.v_num < 400 THEN
            '300-400'
       END AS grade,
       COUNT(t.v_num) count_num
FROM   test_lhr1 t
GROUP  BY t.name,
          CASE
```

```
            WHEN t.v_num < 100 THEN
             '0-100'
            WHEN t.v_num >= 100 AND t.v_num < 200 THEN
             '100-200'
            WHEN t.v_num >= 200 AND t.v_num < 300 THEN
             '200-300'
            WHEN t.v_num >= 300 AND t.v_num < 400 THEN
             '300-400'
        END;
```

NAME	GRADE	COUNT_NUM
璧山	0-100	4
长寿	200-300	1
武隆	0-100	3
长寿	0-100	4
杨家坪	0-100	3
巫溪	0-100	5
武隆	100-200	2
丰都	0-100	5
璧山	200-300	1
杨家坪	300-400	2

```
SELECT t2.grade,
      MAX(decode(t2.name, '璧山', t2.count_num)) 璧山,
      MAX(decode(t2.name, '长寿', t2.count_num)) 长寿,
      MAX(decode(t2.name, '武隆', t2.count_num)) 武隆,
      MAX(decode(t2.name, '丰都', t2.count_num)) 丰都,
      MAX(decode(t2.name, '杨家坪', t2.count_num)) 杨家坪

FROM   (SELECT t.name,
            CASE
                WHEN t.v_num < 100 THEN
                 '0-100'
                WHEN t.v_num >= 100 AND t.v_num < 200 THEN
                 '100-200'
                WHEN t.v_num >= 200 AND t.v_num < 300 THEN
                 '200-300'
```

```
                WHEN t.v_num >= 300 AND t.v_num < 400 THEN
                 '300-400'
              END AS grade,
              COUNT(t.v_num) count_num
       FROM   test_lhr1 t
       GROUP  BY t.name,
                 CASE
                   WHEN t.v_num < 100 THEN
                    '0-100'
                   WHEN t.v_num >= 100 AND t.v_num < 200 THEN
                    '100-200'
                   WHEN t.v_num >= 200 AND t.v_num < 300 THEN
                    '200-300'
                   WHEN t.v_num >= 300 AND t.v_num < 400 THEN
                    '300-400'
                 END) t2
GROUP  BY t2.grade;
```

GRADE	璧山	长寿	武隆	丰都	杨家坪
200-300	1	1			
300-400					2
0-100	4	4	3	5	3
100-200			2		

1.3.2 创建临时表

基本的行转列可以通过创建临时表来实现，首先将基础表根据条件创建几个临时表，然后将这些临时表关联起来就可以了。

```
SELECT t1.id,
       t1.cv C1,
       t2.cv C2,
       t3.cv C3
FROM   (SELECT * FROM t_row_col t WHERE t.cn = 'c1') t1,
```

```
        (SELECT * FROM t_row_col t WHERE t.cn = 'c2') t2,
        (SELECT * FROM t_row_col t WHERE t.cn = 'c3') t3
WHERE  t1.id = t2.id
AND    t2.id = t3.id;
```

ID	C1	C2	C3
1	v11	v21	v31
2	v12	v22	
3	v13		v33
4		v24	v34
5	v15		
6			v35
7			

1.3.3 PL/SQL --存过

适用范围：8i,9i,10g 及以后版本

这种对于行值不固定的情况可以使用。

下面是我写的一个包，包中

p_rows_column_real 用于前述的第一种不限定列的转换；

p_rows_column 用于前述的第二种不限定列的转换。

```
CREATE OR REPLACE PACKAGE pkg_dynamic_rows_column AS
    TYPE refc IS REF CURSOR;

  PROCEDURE p_print_sql(p_txt VARCHAR2);

  FUNCTION f_split_str(p_str VARCHAR2, p_division VARCHAR2, p_seq INT)
    RETURN VARCHAR2;

  PROCEDURE p_rows_column(p_table      IN VARCHAR2,
                          p_keep_cols  IN VARCHAR2,
                          p_pivot_cols IN VARCHAR2,
```

```
                                p_where          IN VARCHAR2 DEFAULT NULL,
                                p_refc           IN OUT refc);

  PROCEDURE p_rows_column_real(p_table         IN VARCHAR2,
                               p_keep_cols IN VARCHAR2,
                               p_pivot_col IN VARCHAR2,
                               p_pivot_val IN VARCHAR2,
                               p_where         IN VARCHAR2 DEFAULT NULL,
                               p_refc          IN OUT refc);
END;
/
CREATE OR REPLACE PACKAGE BODY pkg_dynamic_rows_column AS

    PROCEDURE p_print_sql(p_txt VARCHAR2) IS
    v_len INT;
  BEGIN
    v_len := length(p_txt);
        FOR i IN 1 .. v_len / 250 + 1 LOOP
            dbms_output.put_line(substrb(p_txt, (i - 1) * 250 + 1, 250));
    END LOOP;
  END;

  FUNCTION f_split_str(p_str VARCHAR2, p_division VARCHAR2, p_seq INT)
    RETURN VARCHAR2 IS
    v_first INT;
    v_last    INT;
  BEGIN
        IF p_seq < 1 THEN
      RETURN NULL;
    END IF;
        IF p_seq = 1 THEN
```

```
                IF instr(p_str, p_division, 1, p_seq) = 0 THEN
            RETURN p_str;
          ELSE
            RETURN substr(p_str, 1, instr(p_str, p_division, 1) - 1);
          END IF;
        ELSE
                v_first := instr(p_str, p_division, 1, p_seq - 1);
          v_last    := instr(p_str, p_division, 1, p_seq);
          IF (v_last = 0) THEN
            IF (v_first > 0) THEN
              RETURN substr(p_str, v_first + 1);
            ELSE
              RETURN NULL;
            END IF;
          ELSE
                    RETURN substr(p_str, v_first + 1, v_last - v_first - 1);
          END IF;
        END IF;
      END f_split_str;

      PROCEDURE p_rows_column(p_table          IN VARCHAR2,
                              p_keep_cols    IN VARCHAR2,
                              p_pivot_cols IN VARCHAR2,
                              p_where          IN VARCHAR2 DEFAULT NULL,
                              p_refc           IN OUT refc) IS
        v_sql VARCHAR2(4000);
        TYPE    v_keep_ind_by    IS    TABLE OF VARCHAR2(4000) INDEX BY
    BINARY_INTEGER;
        v_keep v_keep_ind_by;

        TYPE v_pivot_ind_by IS TABLE OF VARCHAR2(4000) INDEX BY
```

```
BINARY_INTEGER;
        v_pivot v_pivot_ind_by;


        v_keep_cnt      INT;
        v_pivot_cnt    INT;
        v_max_cols      INT;
        v_partition    VARCHAR2(4000);
        v_partition1 VARCHAR2(4000);
        v_partition2 VARCHAR2(4000);
    BEGIN
        v_keep_cnt    := length(p_keep_cols) - length(REPLACE(p_keep_cols, ','))
+ 1;
        v_pivot_cnt := length(p_pivot_cols) -
                               length(REPLACE(p_pivot_cols, ',')) + 1;
             FOR i IN 1 .. v_keep_cnt LOOP
           v_keep(i) := f_split_str(p_keep_cols, ',', i);
        END LOOP;
             FOR j IN 1 .. v_pivot_cnt LOOP
           v_pivot(j) := f_split_str(p_pivot_cols, ',', j);
        END LOOP;
             v_sql := 'select max(count(*)) from ' || p_table || ' group by ';
             FOR i IN 1 .. v_keep.LAST LOOP
                  v_sql := v_sql || v_keep(i) || ',';
        END LOOP;
        v_sql := rtrim(v_sql, ',');
        EXECUTE IMMEDIATE v_sql
           INTO v_max_cols;
        v_partition := 'select ';
             FOR x IN 1 .. v_keep.COUNT LOOP
                  v_partition1 := v_partition1 || v_keep(x) || ',';
        END LOOP;
```

```
        FOR y IN 1 .. v_pivot.COUNT LOOP
            v_partition2 := v_partition2 || v_pivot(y) || ',';
    END LOOP;
    v_partition1 := rtrim(v_partition1, ',');
    v_partition2 := rtrim(v_partition2, ',');
        v_partition       := v_partition || v_partition1 || ',' || v_partition2 ||
                          ', row_number() over (partition by ' || v_partition1 ||
                                                  ' order by ' || v_partition2 || ') rn from ' || p_table;
    v_partition    := rtrim(v_partition, ',');
    v_sql              := 'select ';
        FOR i IN 1 .. v_keep.COUNT LOOP
            v_sql := v_sql || v_keep(i) || ',';
    END LOOP;
        FOR i IN 1 .. v_max_cols LOOP
            FOR j IN 1 .. v_pivot.COUNT LOOP
                v_sql := v_sql || ' max(decode(rn,' || i || ',' || v_pivot(j) ||
                 ',null))' || v_pivot(j) || '_' || i || ',';
       END LOOP;
    END LOOP;
        IF p_where IS NOT NULL THEN
            v_sql := rtrim(v_sql, ',') || ' from (' || v_partition || ' ' ||
                p_where || ') group by ';
    ELSE
            v_sql := rtrim(v_sql, ',') || ' from (' || v_partition ||
                ') group by ';
    END IF;
        FOR i IN 1 .. v_keep.COUNT LOOP
            v_sql := v_sql || v_keep(i) || ',';
    END LOOP;
    v_sql := rtrim(v_sql, ',');
    p_print_sql(v_sql);
```

```
    OPEN p_refc FOR v_sql;
  EXCEPTION
    WHEN OTHERS THEN
      OPEN p_refc FOR
        SELECT 'x' FROM dual WHERE 0 = 1;
  END;


  PROCEDURE p_rows_column_real(p_table         IN VARCHAR2,
                               p_keep_cols IN VARCHAR2,
                               p_pivot_col IN VARCHAR2,
                               p_pivot_val IN VARCHAR2,
                               p_where         IN VARCHAR2 DEFAULT NULL,
                               p_refc          IN OUT refc) IS
    v_sql VARCHAR2(4000);
    TYPE   v_keep_ind_by   IS   TABLE OF VARCHAR2(4000) INDEX BY
BINARY_INTEGER;
    v_keep v_keep_ind_by;
    TYPE v_pivot_ind_by IS TABLE OF VARCHAR2(4000) INDEX BY
BINARY_INTEGER;
    v_pivot       v_pivot_ind_by;
    v_keep_cnt INT;
    v_group_by VARCHAR2(2000);
  BEGIN
    v_keep_cnt := length(p_keep_cols) - length(REPLACE(p_keep_cols, ',')) +
1;
        FOR i IN 1 .. v_keep_cnt LOOP
      v_keep(i) := f_split_str(p_keep_cols, ',', i);
    END LOOP;
        v_sql := 'select ' || 'cast(' || p_pivot_col ||
                             ' as varchar2(200)) as ' || p_pivot_col || ' from ' || p_table ||
                ' group by ' || p_pivot_col;
```

```
    EXECUTE IMMEDIATE v_sql BULK COLLECT
      INTO v_pivot;
        FOR i IN 1 .. v_keep.COUNT LOOP
            v_group_by := v_group_by || v_keep(i) || ',';
    END LOOP;
    v_group_by := rtrim(v_group_by, ',');
    v_sql          := 'select ' || v_group_by || ',';

        FOR x IN 1 .. v_pivot.COUNT LOOP
            v_sql := v_sql || ' max(decode(' || p_pivot_col || ',' || chr(39) ||
              v_pivot(x) || chr(39) || ',' || p_pivot_val ||
              ',null)) as "' || v_pivot(x) || '",';
    END LOOP;
    v_sql := rtrim(v_sql, ',');
        IF p_where IS NOT NULL THEN
            v_sql := v_sql || ' from ' || p_table || p_where || ' group by ' ||
              v_group_by;
    ELSE
            v_sql := v_sql || ' from ' || p_table || ' group by ' || v_group_by;
    END IF;
    p_print_sql(v_sql);
    OPEN p_refc FOR v_sql;
  EXCEPTION
    WHEN OTHERS THEN
      OPEN p_refc FOR
        SELECT 'x' FROM dual WHERE 0 = 1;
  END;

END;
/
```

1.4 多列转换成字符串

```
CREATE TABLE t_col_str AS
SELECT * FROM t_col_row;
```

这个比较简单，用|| 或 concat 函数可以实现：

SELECT concat('a','b') FROM dual;

1.4.1 || OR CONCAT

适用范围：8i,9i,10g 及以后版本

```
SELECT * FROM t_col_str;
```

```
SELECT ID,
       c1 || ',' || c2 || ',' || c3 AS c123
FROM   t_col_str;
```

1.5 多行转换成字符串

```
CREATE TABLE t_row_str(
ID INT,
col VARCHAR2(10)
);

INSERT INTO t_row_str VALUES(1,'a');
INSERT INTO t_row_str VALUES(1,'b');
INSERT INTO t_row_str VALUES(1,'c');
INSERT INTO t_row_str VALUES(2,'a');
INSERT INTO t_row_str VALUES(2,'d');
INSERT INTO t_row_str VALUES(2,'e');
```

```
INSERT INTO t_row_str VALUES(3,'c');

COMMIT;

SELECT * FROM t_row_str;
```

ID	COL
1	a
1	b
1	c
2	a
2	d
2	e
3	c

1.5.1 MAX + DECODE

适用范围：8i,9i,10g 及以后版本

```
SELECT id,
       MAX(decode(rn, 1, col, NULL)) ||
       MAX(decode(rn, 2, ',' || col, NULL)) ||
       MAX(decode(rn, 3, ',' || col, NULL)) str
FROM   (SELECT id,
               col,
               row_number() over(PARTITION BY id ORDER BY col) AS rn
        FROM   t_row_str) t
GROUP  BY id
ORDER  BY 1;
```

ID	STR
1	a,b,c
2	a,d,e
3	c

1.5.2 ROW_NUMBER + LEAD

适用范围：8i,9i,10g 及以后版本

```
SELECT id,
       str
FROM   (SELECT id,
               row_number() over(PARTITION BY id ORDER BY col) AS rn,
               col || lead(',' || col, 1) over(PARTITION BY id ORDER BY col)
|| lead(',' || col, 2) over(PARTITION BY id ORDER BY col) || lead(',' || col,
3) over(PARTITION BY id ORDER BY col) AS str
        FROM   t_row_str)
WHERE  rn = 1
ORDER  BY 1;
```

ID	STR
1	a,b,c
2	a,d,e
3	c

1.5.3 MODEL

适用范围：10g 及以后版本

```
SELECT id, substr(str, 2) str FROM t_row_str
MODEL
RETURN UPDATED ROWS
PARTITION BY(ID)
DIMENSION BY(row_number()  over(PARTITION BY ID ORDER BY col) AS rn)
MEASURES (CAST(col AS VARCHAR2(20)) AS str)
RULES UPSERT
ITERATE(3) UNTIL( presentv(str[iteration_number+2],1,0)=0)
        (str[0] = str[0] || ',' || str[iteration_number+1])
```

```
ORDER BY 1;
```

ID	STR
1	a, b, c …
2	a, d, e …
3	c …

1.5.4 SYS_CONNECT_BY_PATH --主要方法

适用范围：8i,9i,10g 及以后版本

```
SELECT t.id id,
       MAX(substr(sys_connect_by_path(t.col, ','), 2)) str
FROM   (SELECT id,
               col,
               row_number() over(PARTITION BY id ORDER BY col) rn
        FROM   t_row_str) t
START  WITH rn = 1
CONNECT BY rn = PRIOR rn + 1
   AND    id = PRIOR id
GROUP  BY t.id;
```

ID	STR
1	a, b, c …
2	a, d, e …
3	c …

适用范围：10g 及以后版本

```
SELECT t.id id,
       substr(sys_connect_by_path(t.col, ','), 2) str
FROM   (SELECT id,
               col,
               row_number() over(PARTITION BY id ORDER BY col) rn
        FROM   t_row_str) t
WHERE  connect_by_isleaf = 1
```

```
START  WITH rn = 1
CONNECT BY rn = PRIOR rn + 1
   AND    id = PRIOR id;
```

1.5.5 WMSYS.WM_CONCAT(col) +dbms_lob.substr(clobcloum,2000,1)

--主要方法

适用范围：10g 及以后版本

这个函数预定义按',' 分隔字符串，若要用其他符号分隔可以用，replace 将',' 替换。

或者： `to_char(wmsys.wm_concat(dic.COLUMN_NAME))`

```
SELECT id,
      REPLACE(wmsys.wm_concat(col), ',', '/') str
FROM   t_row_str
GROUP  BY id;
```

ID	STR
1	<CLOB> …
2	<CLOB> …
3	<CLOB> …

将 clob 转换为字符串类型为：

```
SELECT id,
      dbms_lob.substr(wm_concat(DISTINCT col), 2000, 1)
FROM   t_row_str t
GROUP  BY t.id;
```

ID	DBMS_LOB.SUBSTR(WM_CONCAT(DIST
1	a,b,c …
2	a,d,e …
3	c …

我们通过 10g 所提供的 WMSYS.WM_CONCAT 函数即可以完成 行转列的效果

select t.rank, WMSYS.WM_CONCAT(t.Name) TIME From t_menu_item t GROUP BY t.rank;

DEPTNO ENAME

------ ----------

10 CLARK, KING, MILLER

20 ADAMS, FORD, JONES, SCOTT, SMITH

30 ALLEN, BLAKE, JAMES, MARTIN, TURNER, WARD

例子如下：

SQL> create table idtable (id number,name varchar2(30));

Table created

SQL> insert into idtable values(10,'ab');

1 row inserted

SQL> insert into idtable values(10,'bc');

1 row inserted

SQL> insert into idtable values(10,'cd');

1 row inserted

SQL> insert into idtable values(20,'hi');

1 row inserted

SQL> insert into idtable values(20,'ij');

1 row inserted

SQL> insert into idtable values(20,'mn');

1 row inserted

SQL> select * from idtable;

ID NAME

---------- ------------------------------

10 ab

10 bc

10 cd

20 hi

20 ij

20 mn

6 rows selected

SQL> select id,wmsys.wm_concat(name) name from idtable

2 group by id;

ID NAME

---------- --

10 ab,bc,cd

20 hi,ij,mn

SQL> select id,wmsys.wm_concat(name) over (order by id) name from idtable;

ID NAME

---------- --

10 ab,bc,cd

10 ab,bc,cd

10 ab,bc,cd

20 ab,bc,cd,hi,ij,mn

20 ab,bc,cd,hi,ij,mn

20 ab,bc,cd,hi,ij,mn

6 rows selected

SQL> select id,wmsys.wm_concat(name) over (order by id,name) name from idtable;

ID NAME

---------- --

10 ab

10 ab,bc

10 ab,bc,cd

20 ab,bc,cd,hi

20 ab,bc,cd,hi,ij

20 ab,bc,cd,hi,ij,mn

6 rows selected

个人觉得这个用法比较有趣.

SQL> select id,wmsys.wm_concat(name) over (partition by id) name from idtable;

ID NAME

---------- --

10 ab,bc,cd

10 ab,bc,cd

10 ab,bc,cd

20 hi,ij,mn

20 hi,ij,mn

20 hi,ij,mn

6 rows selected

SQL> select id,wmsys.wm_concat(name) over (partition by id,name) name from idtable;

ID NAME

--- ---

10 ab

10 bc

10 cd

20 hi

20 ij

20 mn

6 rows selected

1.5.5.1 注意

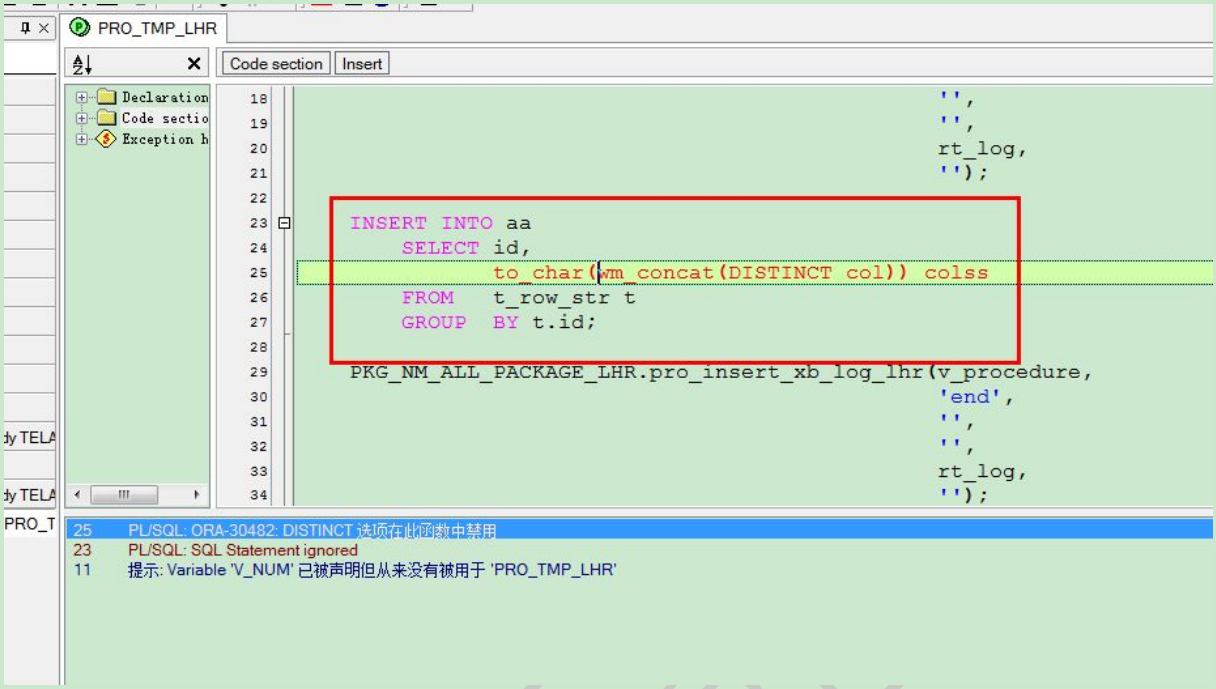
以下均不报错：

```
Create Table aa nologging As
SELECT id,
      to_char(wm_concat(DISTINCT col))  colss
FROM   t_row_str t
GROUP  BY t.id;
```

```
insert into aa
SELECT id,
      to_char(wm_concat(DISTINCT col))  colss
```

```
FROM   t_row_str t
GROUP  BY t.id;
```

但是以上语句放在存过中报错，编译不能通过：



解决办法：

1、通过动态 sql 语句执行

```
execute immediate 'INSERT INTO aa
      SELECT id,
             to_char(wm_concat(DISTINCT col)) colss
      FROM   t_row_str t
   GROUP  BY t.id';
```

2、换一种写法

```
      SELECT id,
             to_char(wm_concat(col)) colss
      FROM   (SELECT DISTINCT a.id,
                              a.col
              FROM   t_row_str a) t
   GROUP  BY t.id;
```

1.5.6 通过分析函数实现行转列

对数据库中的数据用 SQL 实现行列转换，不但需要编写复杂的程序代码，还需要编写存储过程。若引入 ORACLE 中的分析函数则会使该过程简便很多。首先找出表中所有关键字的属性个数的最大值，设为 n，其次为每个关键字新添加 n 列，并用分析函数查询关键字的属性所处列的位置，然后将每个关键字的多行属性转换成多列属性，最后把生成的多个新列拼成一个串形成一列，从而实现行列转换。

1.5.6.1 引言

分析函数的设计目的是为了解决诸如“累计计算”等问题。虽然大部分的问题都可以用 PL/SQL 解决，但是性能并不理想，首先查询本身并不容易编写，其次有些很难在 SQL 中直接做的查询但实际上是很普通的操作，比如实现数据表中行列传换。这样的问题在 SQL 中做查询就很困难。在分析函数出现以前，我们必须使用自联查询或者子查询甚至复杂的存储过程实现的语句，现在只要一条简单的 SQL 语句就可以实现了，而且在执行效率方面也有相当大的提高。本文将以一个实例来描述如何采用分析函数实现数据中的行列互换。

1.5.6.2 原理

2.1 分析函数的格式及语法

分析函数是在一个记录行分组的基础上计算它们的总值。行的分组被称窗口，并通过分析语句定义。对于每记录行，定义了一个“滑动”窗口。该窗口确定“当前行”计算的范围。窗口的大小可由各行的实际编号或由时间等逻辑间隔确定。

分析函数以如下形式开头：

Analytic-Function（<Argument>，<Argument>，...）

OVER （<Query-Partition-Clause><Order-By-Clause><Windowing-Clause>）

（1）Analytic-Function：分析函数的名称，Oracle10gR2 带的内置分析函数有多个，包括：AVG、CORR、COVAR_POP、COVAR_SAMP、COUNT、LAG、LAST、LEAD、MAX、MIN、RANK、SUM 等；对于用户自定义的分析函数，分析函数名称需要满足标识符规则。

（2）Arguments：参数，分析函数通常有 0 到 3 个参数，参数可以是任何数字类型或是可以隐式转换为数字类型的数据类型。对于用户自定义的参数，可以根据实际情况使用。

（3）OVER：是分析函数就必须使用的关键字，对于既可作为聚集函数又可作为分析函数的函数，

Oracle 无法识别，必须用 over 来标识此函数为分析函数。

（4）Query-Partition-Clause：查询分组子句，根据划分表达式设置的规则，PARTITION BY 将一个结果逻辑分成 N 个分组划分表达式。分析函数独立应用于各个分组，并在应用时重置。

（5）Order-By-Clause：（按…排序分组），是排序子句，根据一个或多个排序表达式对分组进行排序。

（6）Windowing-Clause 窗口生成语句：窗口生成语句用以定义滑动或固定数据窗口，分析函数在分组内进行分析。该语句能够对分组中任意定义的滑动或固定窗口进行计算。

2.2 实例原理介绍

本实例是将具有相同关键字的多条记录中的某一不同列合并成一列，例如在一个临时表中包含有用户的编号、电话号码、产品名称、所在营业区以及相关业务名称 5 个字段，而每个用户的业务可能有多项，这样创建数据表将会造成冗余，现在要想办法将表中编号、电话号码、产品名称、所在营业区四个字段相同的用户的相关业务属性合并成一列解决冗余问题，使用 SQL 语句会比较困难，甚至需要一定的存储过程。使用 Orcale 中的分析函数来实现这样的行列转换就比较简单方便了。

1.5.6.3 实例

1）创建临时表

```
Drop Table temp;
 Create Table temp
 (
 num varchar2(15),name varchar2(20),
 sex varchar2(2),
 classes varchar2(30),
 course_name varchar2(50)
 );
```

2）构造数据

```
insert  into   temp(num,name,sex,classes,course_name)  values ('206211','王艺','男','06-1班','保险学');

insert into temp(num,name,sex,classes,course_name) values ('206212','肖薇','女','06-2','保险学');

insert into temp(num,name,sex,classes,course_name) values ('206212','肖薇','女','06-2','财务管理');

insert into temp(num,name,sex,classes,course_name) values ('206212','肖薇','女','06-2','财务会计');

insert into temp(num,name,sex,classes,course_name) values ('206213','陈雅诗','女','06-2','电子商务');

insert into temp(num,name,sex,classes,course_name) values ('206213','陈雅诗','女','06-2','公共经济学');

insert into temp(num,name,sex,classes,course_name) values ('206213','陈雅诗','女','06-2','公司理财');

insert into temp(num,name,sex,classes,course_name) values ('206213','陈雅诗','女','06-2','管理学原理');

insert into temp(num,name,sex,classes,course_name) values ('206213','陈雅诗','女','06-2','保险学');

insert into temp(num,name,sex,classes,course_name) values ('206214','李丹阳','男','06-1','保险学');

insert into temp(num,name,sex,classes,course_name) values ('206214','李丹阳','男','06-1','财务管理');

insert into temp(num,name,sex,classes,course_name) values ('206214','李丹阳','男','06-1','财务会计');

insert into temp(num,name,sex,classes,course_name) values ('206214','李丹阳','男','06-1','电子商务');

insert into temp(num,name,sex,classes,course_name) values ('206214','李丹阳','男','06-1','公共经济学');

insert into temp(num,name,sex,classes,course_name) values ('206215','杨伊琳','女','06-3班','环境管理学');

insert into temp(num,name,sex,classes,course_name) values ('206215','杨伊琳','女','06-3班','管理学原理');

insert into temp(num,name,sex,classes,course_name) values ('206215','杨伊琳','女','06-3班','商务谈判');

insert into temp(num,name,sex,classes,course_name) values ('206216','李佳琪','男','06-2','土地估计');

Commit;
```

```
select * from temp;
```

NUM	NAME	SEX	CLASSES	COURSE_NAME
206211	王艺	男	06-1班	保险学
206212	肖薇	女	06-2	保险学
206212	肖薇	女	06-2	财务管理
206212	肖薇	女	06-2	财务会计
206213	陈雅诗	女	06-2	电子商务
206213	陈雅诗	女	06-2	公共经济学
206213	陈雅诗	女	06-2	公司理财
206213	陈雅诗	女	06-2	管理学原理
206213	陈雅诗	女	06-2	保险学
206214	李丹阳	男	06-1	保险学
206214	李丹阳	男	06-1	财务管理
206214	李丹阳	男	06-1	财务会计
206214	李丹阳	男	06-1	电子商务
206214	李丹阳	男	06-1	公共经济学
206215	杨伊琳	女	06-3班	环境管理学
206215	杨伊琳	女	06-3班	管理学原理
206215	杨伊琳	女	06-3班	商务谈判
206216	李佳琪	男	06-2	土地估计

---

3)先查一下course_name最多的组合

```
select max(count(course_name))
from   temp
group  by num,name,sex,classes;
```

MAX(COUNT(COURSE_NAME))
5

---

4) 列的位置

用分析函数中的row_number函数,在num,name,sex,classes相同的情况下course_name所处的列的位置(第几列)。

row_number函数解释：返回有序组中一行的偏移量,从而可用于按特定标准排序的行号。

```
SELECT num,
       NAME,
       sex,
       classes,
       course_name,
       row_number() over(PARTITION BY num, NAME, sex, classes ORDER BY course_name)
rn   from temp;
```

NUM	NAME	SEX	CLASSES	COURSE_NAME	RN
206211	王艺	男	06-1班	保险学	1
206212	肖薇	女	06-2	保险学	1
206212	肖薇	女	06-2	财务管理	2
206212	肖薇	女	06-2	财务会计	3
206213	陈雅诗	女	06-2	保险学	1
206213	陈雅诗	女	06-2	电子商务	2
206213	陈雅诗	女	06-2	公共经济学	3
206213	陈雅诗	女	06-2	公司理财	4
206213	陈雅诗	女	06-2	管理学原理	5
206214	李丹阳	男	06-1	保险学	1
206214	李丹阳	男	06-1	财务管理	2
206214	李丹阳	男	06-1	财务会计	3
206214	李丹阳	男	06-1	电子商务	4
206214	李丹阳	男	06-1	公共经济学	5
206215	杨伊琳	女	06-3班	管理学原理	1
206215	杨伊琳	女	06-3班	环境管理学	2
206215	杨伊琳	女	06-3班	商务谈判	3
206216	李佳琪	男	06-2	土地估计	1

---

5) 把course_name的所有的行换成列

```
SELECT num,
       NAME,
       sex,
       classes,
       MAX(decode(rn, 1, course_name, NULL)) course_name_1,
       MAX(decode(rn, 2, course_name, NULL)) course_name_2,
       MAX(decode(rn, 3, course_name, NULL)) course_name_3,
       MAX(decode(rn, 4, course_name, NULL)) course_name_4,
       MAX(decode(rn, 5, course_name, NULL)) course_name_5

FROM   (SELECT num,
               NAME,
               sex,
               classes,
               course_name,
               row_number() over(PARTITION BY num, NAME, sex, classes ORDER BY course_name)
```

```
rn
       FROM   temp)

    GROUP  BY num,
              NAME,
              sex,
           classes;
```

NUM	NAME	SEX	CLASSES	COURSE_NAME_1	COURSE_NAME_2	COURSE_NAME_3	COURSE_NAME_4	COURSE_NAME_5
206211	王艺	男	06-1班	保险学				
206212	肖薇	女	06-2	保险学	财务管理	财务会计		
206213	陈雅诗	女	06-2	保险学	电子商务	公共经济学	公司理财	管理学原理
206214	李丹阳	男	06-1	保险学	财务管理	财务会计	电子商务	公共经济学
206215	杨伊琳	女	06-3班	管理学原理	环境管理学	商务谈判		
206216	李佳琪	男	06-2	土地估计				

6）把转换后的 name 拼成一个字符串，放在一行

```
SELECT num,
       NAME,
       sex,
       classes,
       (MAX(decode(rn, 1, course_name, NULL)) ||
       MAX(decode(rn, 2, ',' || course_name, NULL)) ||
       MAX(decode(rn, 3, ',' || course_name, NULL)) ||
       MAX(decode(rn, 4, ',' || course_name, NULL)) ||
       MAX(decode(rn, 5, ',' || course_name, NULL))) NAME
FROM   (SELECT num,
               NAME,
               sex,
               classes,
               course_name,
               row_number() over(PARTITION BY num, NAME, sex, classes ORDER BY
course_name) rn
        FROM   temp)
GROUP  BY num,
```

```
        NAME,
        sex,
        classes;
```

NUM	NAME	SEX	CLASSES	NAME
206211	王艺	男	06-1班	保险学
206212	肖薇	女	06-2	保险学,财务管理,财务会计
206213	陈雅诗	女	06-2	保险学,电子商务,公共经济学,公司理财,管理学原理
206214	李丹阳	男	06-1	保险学,财务管理,财务会计,电子商务,公共经济学
206215	杨伊琳	女	06-3班	管理学原理,环境管理学,商务谈判
206216	李佳琪	男	06-2	土地估计

1.5.6.4 总结

本文中的程序能够实现以下功能：①计算具有相同关键字的最多的组合；②根据分析函数查询某一关键字所处的列的位置；③把需合并列的所有的行换成列；④把需要合并的某几列拼成一个串。

分析函数除了拥有以上所介绍的功能，还能够实现诸如求和、Top-N 查询、统计某个范围的数据行窗口、交叉表查询等功能。

1.5.7 例题

```
CREATE TABLE tab_name(ID INTEGER NOT NULL PRIMARY KEY,cName VARCHAR2(20));
CREATE TABLE tab_name2(ID INTEGER NOT NULL,pName VARCHAR2(20));

INSERT INTO tab_name(ID,cName) VALUES (1,'百度');
INSERT INTO tab_name(ID,cName) VALUES (2,'Google');
INSERT INTO tab_name(ID,cName) VALUES (3,'网易');
INSERT INTO tab_name2(ID,pName) VALUES (1,'研发部');
INSERT INTO tab_name2(ID,pName) VALUES (1,'市场部');
INSERT INTO tab_name2(ID,pName) VALUES (2,'研发部');
INSERT INTO tab_name2(ID,pName) VALUES (2,'平台架构');
INSERT INTO tab_name2(ID,pName) VALUES (3,'研发部');
COMMIT;
```

```
select * from tab_name;
```

ID	CNAME
1	百度
2	Google
3	网易

```
select * from tab_name2;
```

ID	PNAME
1	研发部
1	市场部
2	研发部
2	平台架构
3	研发部

方法一:使用wmsys.wm_concat()

```
SELECT t1.ID,
       t1.cName,
       wmsys.wm_concat(t2.pName)
FROM   tab_name  t1,
       tab_name2 t2
WHERE  t1.ID = t2.ID
GROUP  BY t1.cName,
          t1.id;
```

方法二:使用sys_connect_by_path

```
SELECT id,
       cName,
       ltrim(MAX(sys_connect_by_path(pName, ',')), ',')
FROM   (SELECT row_number() over(PARTITION BY t1.id ORDER BY cName) r,
               t1.*,
               t2.pName
        FROM   tab_name  t1,
```

```
            tab_name2 t2
      WHERE  t1.id = t2.id)
START  WITH r = 1
CONNECT BY PRIOR r = r - 1
   AND    PRIOR id = id
GROUP  BY id,
        cName
ORDER  BY id;
```

方法三:使用自定义函数

```
CREATE OR REPLACE FUNCTION coltorow(midId INT) RETURN VARCHAR2 IS
   RESULT VARCHAR2(1000);
BEGIN
   FOR cur IN (SELECT pName FROM tab_name2 t2 WHERE midId = t2.id) LOOP
      RESULT := RESULT || cur.pName || ',';
   END LOOP;
   RESULT := rtrim(RESULT, ',');
   RETURN(RESULT);
END coltorow;

SELECT t1.*,
     coltorow(t1.ID)
FROM   tab_name  t1,
     tab_name2 t2
WHERE  t1.ID = t2.ID
GROUP  BY t1.ID,
        t1.cname
ORDER  BY t1.ID;
```

ID	CNAME	WMSYS.WM_CONCAT(T2.PNAME)
1	百度	研发部,市场部
3	网易	研发部
2	Google	研发部,平台架构

1.6 字符串转换成多列

其实际上就是一个字符串拆分的问题。

```
CREATE TABLE t_str_col AS
SELECT ID,c1||','||c2||','||c3 AS c123
FROM t_col_str;
```

```
SELECT * FROM t_str_col;
```

ID	C123
1	v11,v21,v31
2	v12,v22,
3	v13,,v33
4	,v24,v34
5	v15,,
6	,,v35
7	,,

1.6.1 SUBSTR + INSTR

1.6.1.1 将单列分为多列

适用范围：8i,9i,10g 及以后版本

```
SELECT id,
       c123,
       substr(c123, 1, instr(c123 || ',', ',', 1, 1) - 1) c1,
       substr(c123,
              instr(c123 || ',', ',', 1, 1) + 1,
              instr(c123 || ',', ',', 1, 2) - instr(c123 || ',', ',', 1, 1) - 1) c2,
       substr(c123,
              instr(c123 || ',', ',', 1, 2) + 1,
              instr(c123 || ',', ',', 1, 3) - instr(c123 || ',', ',', 1, 2) - 1) c3
```

```
FROM   t_str_col

ORDER  BY 1;
```

ID	C123	C1	C2	C3
1	v11,v21,v31	v11	v21	v31
2	v12,v22,	v12	v22	
3	v13,,v33	v13		v33
4	,v24,v34		v24	v34
5	v15,,	v15		
6	,,v35			v35
7	,,			

一、 存过

将单列分为多列见以下存过

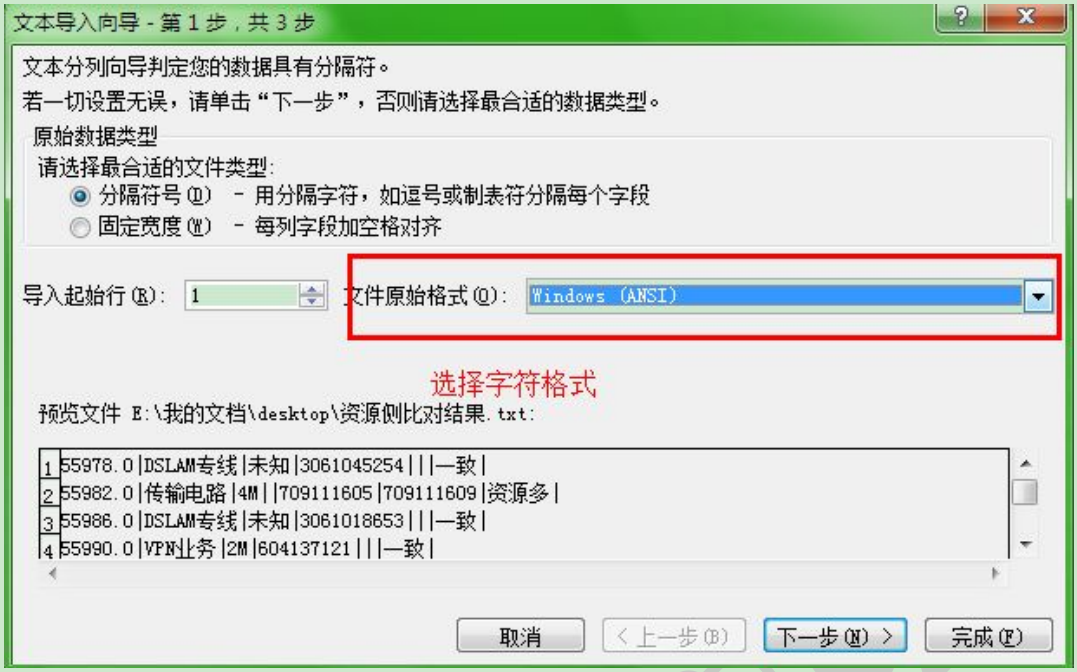
单列转换为多列的存过

二、 借助 excel 办公软件

首先将需要转换的字符串列导出到一个 txt 文件或其它文件中，然后新建一个 excel 文件，打开这个 excel 文件后，在数据选项卡里选择导入数据，见截图：

金山 WPS

微软办公软件



选择字符格式

选择分隔符号

最后确定就 OK 了，最后把 excel 里转换后的数据通过复制粘贴的方式导入到数据库里就可以了。

三、 采用 sqlldr

采用 sqlldr 的方式导入到数据库中。

1.6.1.2 将多列分为多列

```
CREATE TABLE t_str_col2 AS
SELECT ID,c1||','||c2||','||c3 AS c123,c3||','||c2 as c32
FROM t_col_str;
```

```
select * from t_str_col2;
```

ID	C123	C32
1	v11,v21,v31	v31,v21
2	v12,v22,	,v22
3	v13,,v33	v33,
4	,v24,v34	v34,v24
5	v15,,	,
6	,,v35	v35,
7	,,	,

```
SELECT id,
       c123,
       substr(c123, 1, instr(c123 || ',', ',', 1, 1) - 1) c1,
       substr(c123,
              instr(c123 || ',', ',', 1, 1) + 1,
              instr(c123 || ',', ',', 1, 2) - instr(c123 || ',', ',', 1, 1) -
1) c2,
       substr(c123,
              instr(c123 || ',', ',', 1, 2) + 1,
              instr(c123 || ',', ',', 1, 3) - instr(c123 || ',', ',', 1, 2) -
1) c3,
       c32,
       substr(c32, 1, instr(c32, ',') - 1) cc3,
       substr(c32, instr(c32, ',') + 1) cc2
FROM   t_str_col2
ORDER  BY 1;
```

ID	C123	C1	C2	C3	C32	CC3	CC2
1	v11,v21,v31	v11	v21	v31	v31,v21	v31	v21
2	v12,v22,	v12	v22		,v22		v22
3	v13,,v33	v13		v33	v33,	v33	
4	,v24,v34		v24	v34	v34,v24	v34	v24
5	v15,,	v15			,		
6	,,v35			v35	v35,	v35	
7	,,				,		

1.6.2 REGEXP_SUBSTR

适用范围：10g 及以后版本

```
SELECT id,
       c123,
       rtrim(regexp_substr(c123 || ',', '.*?' || ',', 1, 1), ',') AS c1,
       rtrim(regexp_substr(c123 || ',', '.*?' || ',', 1, 2), ',') AS c2,
       rtrim(regexp_substr(c123 || ',', '.*?' || ',', 1, 3), ',') AS c3
FROM   t_str_col
ORDER  BY 1;
```

1.7 字符串转换成多行

```
CREATE TABLE t_str_row AS
SELECT id,
     MAX(decode(rn, 1, col, NULL)) ||
           MAX(decode(rn, 2, ',' || col, NULL)) ||
           MAX(decode(rn, 3, ',' || col, NULL)) str
  FROM (SELECT id,
               col,
               row_number() over(PARTITION BY id ORDER BY col) AS rn
          FROM t_row_str) t
  GROUP BY id
  ORDER BY 1;
```

```
SELECT * FROM t_str_row;
```

ID	STR
1	a, b, c
2	a, d, e
3	c

1. 7. 1 UNION ALL

适用范围：8i,9i,10g 及以后版本

```
SELECT id,
       1 AS p,
       substr(str, 1, instr(str || ',', ',', 1, 1) - 1) AS cv
FROM   t_str_row
UNION ALL
SELECT id,
       2 AS p,
       substr(str,
              instr(str || ',', ',', 1, 1) + 1,
              instr(str || ',', ',', 1, 2) - instr(str || ',', ',', 1, 1) - 1) AS
cv
FROM   t_str_row
UNION ALL
SELECT id,
       3 AS p,
       substr(str,
              instr(str || ',', ',', 1, 1) + 1,
              instr(str || ',', ',', 1, 2) - instr(str || ',', ',', 1, 1) - 1) AS
cv
FROM   t_str_row
ORDER  BY 1,
          2;
```

适用范围：10g 及以后版本

```
SELECT id,
       1 AS p,
       rtrim(regexp_substr(str || ',', '.*?' || ',', 1, 1), ',') AS cv
FROM   t_str_row
UNION ALL
SELECT id,
       2 AS p,
       rtrim(regexp_substr(str || ',', '.*?' || ',', 1, 2), ',') AS cv
FROM   t_str_row
UNION ALL
SELECT id,
       3 AS p,
       rtrim(regexp_substr(str || ',', '.*?' || ',', 1, 3), ',') AS cv
FROM   t_str_row
ORDER  BY 1,
          2;
```

1.7.2 VARRAY

适用范围：8i,9i,10g 及以后版本

要创建一个可变数组：

```
CREATE OR REPLACE TYPE ins_seq_type IS VARRAY(8) OF NUMBER;

SELECT * FROM TABLE(ins_seq_type(1, 2, 3, 4, 5));

SELECT t.id,
       c.column_value AS p,
       substr(t.ca,
              instr(t.ca, ',', 1, c.column_value) + 1,
```

```
          instr(t.ca, ',', 1, c.column_value + 1) -
          (instr(t.ca,  ',', 1, c.column_value) + 1)) AS cv
 FROM (SELECT id,
           ',' || str || ',' AS ca,
           length(str || ',') - nvl(length(REPLACE(str, ',')), 0) AS cnt
       FROM t_str_row) t
INNER JOIN TABLE(ins_seq_type(1, 2, 3)) c ON c.column_value <=
                                             t.cnt
 ORDER BY 1, 2;
```

1.7.3 SEQUENCE SERIES

这类方法主要是要产生一个连续的整数列，产生连续整数列的方法有很多，主要有：

CONNECT BY,ROWNUM+all_objects,CUBE 等。

适用范围：8i,9i,10g 及以后版本

```
SELECT t.id,
      c.lv AS p,
      substr(t.ca,
           instr(t.ca, ',', 1, c.lv) + 1,
           instr(t.ca, ',', 1, c.lv + 1) -
           (instr(t.ca, ',', 1, c.lv) + 1)) AS cv
FROM   (SELECT id,
           ',' || str || ',' AS ca,
           length(str || ',') - nvl(length(REPLACE(str, ',')), 0) AS cnt
       FROM   t_str_row) t,
      (SELECT LEVEL lv FROM dual CONNECT BY LEVEL <= 5) c
WHERE  c.lv <= t.cnt
ORDER  BY 1,
        2;


SELECT t.id,
```

```
       c.rn AS p,
       substr(t.ca,
              instr(t.ca, ',', 1, c.rn) + 1,
              instr(t.ca, ',', 1, c.rn + 1) -
              (instr(t.ca, ',', 1, c.rn) + 1)) AS cv
FROM   (SELECT id,
               ',' || str || ',' AS ca,
               length(str || ',') - nvl(length(REPLACE(str, ',')), 0) AS cnt
        FROM   t_str_row) t,
       (SELECT rownum rn FROM all_objects WHERE rownum <= 5) c
WHERE  c.rn <= t.cnt
ORDER  BY 1,
          2;


SELECT t.id,
       c.cb AS p,
       substr(t.ca,
              instr(t.ca, ',', 1, c.cb) + 1,
              instr(t.ca, ',', 1, c.cb + 1) -
              (instr(t.ca, ',', 1, c.cb) + 1)) AS cv
FROM   (SELECT id,
               ',' || str || ',' AS ca,
               length(str || ',') - nvl(length(REPLACE(str, ',')), 0) AS cnt
        FROM   t_str_row) t,
       (SELECT rownum cb FROM (SELECT 1 FROM dual GROUP BY CUBE(1, 2))) c
WHERE  c.cb <= t.cnt
ORDER  BY 1,
          2;
```

适用范围：10g 及以后版本

```
SELECT t.id,
```

```
      c.lv AS p,
      rtrim(regexp_substr(t.str || ',', '.*?' || ',', 1, c.lv), ',') AS cv
FROM   (SELECT id,
               str,
               length(regexp_replace(str || ',', '[^' || ',' || ']', NULL)) AS cnt
        FROM   t_str_row) t
INNER  JOIN (SELECT LEVEL lv FROM dual CONNECT BY LEVEL <= 5) c
ON     c.lv <= t.cnt
ORDER  BY 1,
          2;
```

```
drop table   t_test
create table t_test (id number, names varchar2(200));

insert into t_test values (1,'a1,a2,a3,a4');
insert into t_test values (2,'b1,b2,b3');
insert into t_test values (3,'c1,c2,c3,c4,c5');
commit;
select * from t_test;
```

ID	NAMES
1	a1, a2, a3, a4
2	b1, b2, b3
3	c1, c2, c3, c4, c5

常规做法：

```
SELECT id,
       REGEXP_SUBSTR(names, '[^,]+', 1, l) AS NAME
FROM   t_test,
       (SELECT LEVEL l FROM DUAL CONNECT BY LEVEL <= 100)
WHERE  l <= LENGTH(names) - LENGTH(REPLACE(names, ',')) + 1
ORDER  BY 1,
          2;
```

```
    SELECT DISTINCT REGEXP_SUBSTR(PARAM_VALUES, '[^,]+', 1, LEVEL) AS SOC_NAME
  FROM CM9_BATCH_CONTROL
WHERE PARAM_NAME = 'OFFER'
   AND JOB_NAME = 'xxxxxxx'
   AND JOB_REC = 'ENDDAY'
CONNECT BY REGEXP_SUBSTR((SELECT PARAM_VALUES
                           FROM CM9_BATCH_CONTROL
                          WHERE JOB_NAME = 'xxxxx'
                            AND PARAM_NAME = 'OFFER'),
                         '[^,]+',
                         1,
                         LEVEL) IS NOT NULL;
```

```
SELECT CO.SOC_CD FROM (SELECT REGEXP_SUBSTR(PARAM_VALUES,'[^,]+',1,1) AS SOC_NAME
    FROM CM9_BATCH_CONTROL
           ,(SELECT LEVEL l FROM DUAL CONNECT BY LEVEL<=100)
WHERE PARAM_NAME = 'OFFER'
      AND JOB_NAME = 'xxxx'
      AND JOB_REC = 'ENDDAY'
      AND l <=LENGTH(PARAM_VALUES) - LENGTH(REPLACE(PARAM_VALUES,','))+1
)T, CSM_OFFER CO WHERE T.SOC_NAME = CO.SOC_NAME
and T.SOC_NAME is not null
```

1.7.4 HIERARCHICAL + DBMS_RANDOM

适用范围：10g 及以后版本

```
SELECT id,
       LEVEL AS p,
       rtrim(regexp_substr(str || ',', '.*?' || ',', 1, LEVEL), ',') AS cv
FROM   t_str_row
```

```
CONNECT BY id = PRIOR id
   AND    PRIOR dbms_random.VALUE IS NOT NULL
   AND    LEVEL <=
          length(regexp_replace(str || ',', '[^' || ',' || ']', NULL))
ORDER  BY 1,
          2;
```

ID	P	CV
1	1	a
1	2	b
1	3	c
2	1	a
2	2	d
2	3	e
3	1	c

1.7.5 HIERARCHICAL + CONNECT_BY_ROOT

适用范围：10g 及以后版本

```
SELECT id,
     LEVEL AS p,
     rtrim(regexp_substr(str || ',', '.*?' || ',', 1, LEVEL), ',') AS cv
FROM   t_str_row
CONNECT BY id = connect_by_root
 id
   AND    LEVEL <=
          length(regexp_replace(str || ',', '[^' || ',' || ']', NULL))
ORDER  BY 1,
          2;
```

ID	P	CV
1	1	a
1	2	b
1	3	c
2	1	a
2	2	d
2	3	e
3	1	c

1.7.6 MODEL

适用范围：10g 及以后版本

```
SELECT id, p, cv FROM t_str_row
MODEL
 RETURN UPDATED ROWS
 PARTITION BY(ID)
  DIMENSION BY( 0 AS p)
  MEASURES( str||',' AS cv)
 RULES UPSERT
  (cv
   [ FOR p
       FROM 1 TO length(regexp_replace(cv[0],'[^'||',']',null))
       INCREMENT 1
     ] = rtrim(regexp_substr( cv[0],'.*?'||',',1,cv(p)),','))
  ORDER BY 1,2 ;
```

ID	P	CV
1	1	a …
1	2	b …
1	3	c …
2	1	a …
2	2	d …
2	3	e …
3	1	c …

第 2 章　例题

```
drop table   course;
create table course (stname varchar(10), math int, english int);
insert into course values('Jame', 65, 97);
insert into course values('Tom',88,59);
insert into course values('calvin',98,99);
select * from course;
```

STNAME	MATH	ENGLISH
Jame	65	97
Tom	88	59
calvin	98	99

```
create table pivot (id int);
insert into pivot values (1);
insert into pivot values (2);
select * from pivot;
```

ID
1
2

```
select * from course;
select * from pivot;
SELECT stname,
      CASE id
         WHEN 1 THEN
          'Math'
         WHEN 2 THEN
          'English'
         ELSE
          '0'
      END AS subject,
      CASE id WHEN 1 THEN math WHEN 2 THEN english ELSE 0end AS grade
FROM   course,
```

```
pivot;
```

STNAME	SUBJECT	GRADE
Jame	Math	65
Tom	Math	88
calvin	Math	98
Jame	English	97
Tom	English	59
calvin	English	99

第 3 章　Oracle 行转列存过

oracle 行转列的通用过程

经常遇到发帖求行列转换的代码，用 max(decode(..))回复后，十有八九会再问一句：如果列名不固定，或者列数不固定怎么办。就要用存储过程来写，这些存储过程的代码都大同小异，我就想能不能写个通用点的过程

试了一下，把结果发出来

SQL code

```
create or replace procedure proc(tabname in varchar2,
                                 col1 in varchar2,
                                 col2 in varchar2,
                                 col3 in varchar2,
                                 viewname in varchar2 default 'v_tmp')
as
sqlstr varchar2(2000):='create or replace view '||viewname||' as select '||col1||' ';
c1 sys_refcursor;
v1 varchar2(100);
begin
open c1 for 'select distinct to_char('||col2||') from '||tabname;
loop
    fetch c1 into v1;
    exit when c1%notfound;
    sqlstr:=sqlstr||'
,max(decode('||col2||','''||v1||''','||col3||'))'''||v1||'''';
end loop;
close c1;
sqlstr:=sqlstr||' from '||tabname||' group by '||col1;
```

execute immediate sqlstr;

end proc;

这里的几个参数，**tabname** 指的是需要进行行列转换的表名，**col1** 是这个表中行列转换以后要根据哪一列进行分组，那一列的列名。**col2** 传入的是要将行转成列的那一列的列名，**col3** 表示需要进行统计的数据列的列名

viewname 传入希望建立的视图的名称，可以不填，默认为 **v_tmp**

这么说很难让人明白..举个例子，引用一个帖子的数据

```
create table tab (
counter varchar(20),          -- 参加考试人数
subject varchar(20),          -- 科目
class varchar(20)             -- 班级
)
```

表数据：

counter	subject	class
36	英语	一班
44	英语	二班
44	数学	二班
33	语文	一班
39	语文	三班

转换后：

一班	二班	三班

英语 36 44 0

数学 0 44 0

语文 33 0 39

编译好过程后，执行

SQL code

```
begin
proc('tab','subject','class','counter');
end;

--结果
select * from v_tmp;
```

SUBJECT 一班 三班 二班

数学 44

英语 36 44

语文 33 39

如果对这个结果不是很满意，需要自己进行一些修改，比如空值的地方用 0 代替，或者需要用别的函数聚合而不是 max。可以将过程中的 execute immediate 那句改成

dbms_output.put_line(sqlstr);

重新编译，执行，输出代码

如果用的是 pl/sql dev 的 sql 窗口，到 output 窗口查看

SQL code

```
--看到生成的代码
create or replace view v_tmp as select subject
,max(decode(class,'一班',counter))"一班"
```

,max(decode(class,'三班',counter))"三班"

,max(decode(class,'二班',counter))"二班" from tab group by subject

再加入 **nvl()**，达到修改的目的

如果不想创建这样一个过程，则改成匿名块，需要时运行

SQL code

```
declare
tabname varchar2(20):='XXX';--'XXX'分别用相应的表名和字段名代替
col1 varchar2(10):='XXX';
col2 varchar2(10):='XXX';
col3 varchar2(10):='XXX';
viewname in varchar2(10):= 'v_tmp';
sqlstr varchar2(2000):='create or replace view '||viewname||' as select '||col1||' ';
c1 sys_refcursor;
v1 varchar2(100);
begin
open c1 for 'select distinct to_char('||col2||') from '||tabname;
loop
      fetch c1 into v1;
      exit when c1%notfound;
      sqlstr:=sqlstr||'
,max(decode('||col2||','''||v1||''','||col3||'))"'||v1||'"';
end loop;
close c1;
sqlstr:=sqlstr||' from '||tabname||' group by '||col1;
--execute immediate sqlstr;
dbms_output.put_line(sqlstr);
end;
```

---

oracle 行转列（动态行转不定列)

```
----------------------------------------------------------建表
---------------判断表是否存在
declare num number;
begin
      select count(1) into num from user_tables where table_name='TEST';
      if num>0 then
         execute immediate 'drop table TEST';
      end if;
end;
---------------建表
CREATE TABLE TEST(
      WL VARCHAR2(10),
      XYSL INTEGER,
      XYCK VARCHAR2(10),
      XCLCK VARCHAR2(10),
      XCLCKSL INTEGER,
      PC INTEGER
);
---------------第一部分测试数据
INSERT INTO TEST VALUES('A1', 2, 'C1', 'C1' ,          20,          123);
INSERT INTO TEST VALUES('A1', 2, 'C1', 'C2' ,          30,          111);
INSERT INTO TEST VALUES('A1', 2, 'C1', 'C2' ,          20,          222);
INSERT INTO TEST VALUES('A1', 2, 'C1', 'C3' ,          10,          211);
INSERT INTO TEST VALUES('A2', 3, 'C4', 'C1' ,          40,          321);
INSERT INTO TEST VALUES('A2', 3, 'C4', 'C4' ,          50,          222);
INSERT INTO TEST VALUES('A2', 3, 'C4', 'C4' ,          60,          333);
INSERT INTO TEST VALUES('A2', 3, 'C4', 'C5' ,          70,          223);
```

```
COMMIT;
--select * from test;
------------------------------------------------------------行转列的存储过程
CREATE OR REPLACE PROCEDURE P_TEST IS
  V_SQL VARCHAR2(2000);
  CURSOR CURSOR_1 IS SELECT DISTINCT T.XCLCK FROM TEST T ORDER BY XCLCK;

    BEGIN
      V_SQL := 'SELECT WL,XYSL,XYCK';

      FOR V_XCLCK IN CURSOR_1
      LOOP
        V_SQL := V_SQL || ',' || 'SUM(DECODE(XCLCK,''' || V_XCLCK.XCLCK ||
                 ''',XCLCKSL,0)) AS ' || V_XCLCK.XCLCK;
      END LOOP;

      V_SQL := V_SQL || ' FROM TEST GROUP BY WL,XYSL,XYCK ORDER BY
WL,XYSL,XYCK';
      --DBMS_OUTPUT.PUT_LINE(V_SQL);
      V_SQL := 'CREATE OR REPLACE VIEW RESULT  AS '||  V_SQL;
      --DBMS_OUTPUT.PUT_LINE(V_SQL);
      EXECUTE IMMEDIATE V_SQL;
    END;
------------------------------------------------------------结果
----------执行存储过程，生成视图
BEGIN
  P_TEST;
END;
----------结果
SELECT * FROM RESULT T;
WL                                XYSL XYCK                C1          C2
```

C3 C4 C5

--- --- --- --- --- --- ---

A1 2 C1 20 50

10 0 0

A2 3 C4 40 0

0 110 70

----------第二部分测试数据

INSERT INTO TEST VALUES('A1', 2, 'C1', 'C6' , 20, 124);

INSERT INTO TEST VALUES('A2', 2, 'C1', 'C7' , 30, 121);

INSERT INTO TEST VALUES('A3', 2, 'C1', 'C8' , 20, 322);

COMMIT;

----------报告存储过程，生成视图

BEGIN

P_TEST;

END;

----------结果

SELECT * FROM RESULT T;

WL XYSL XYCK C1 C2 C3 C4 C5 C6

C7 C8

--- --- --- --- --- --- --- --- --- ---

A1 2 C1 20 50 10 0 0 20

0 0

A2 2 C1 0 0 0 0 0 0

30 0

A2 3 C4 40 0 0 110 70 0

0 0

A3 2 C1 0 0 0 0 0 0

0 20

---------- 删除实体

DROP VIEW RESULT;

DROP PROCEDURE P_TEST;

DROP TABLE TEST;

第 4 章　存过 2

环境 oracle 10g

工作关系,常做些行转列报表,报表通常不是在大数据集合上处理.

所以写了个过程.

本过程比较适合在于需要动态输出报表的地方,例如 web 中.

不是很完美,但已经可以解决绝大部分的问题.

```
create or replace function func_RowToCol(
viewName Varchar2,
grpCols Varchar2,
colCol Varchar2,
valueCol Varchar2,
fillEmptyWithZero Number:=1,
rowOrder Varchar2:='',
colOrder Varchar2:='',
rowOrderinGrp Integer:=1,
colOrderStyle Varchar2:=' asc ',
fillValue Varchar2:=''
) return varchar2
Is
/**********************************************************************************************
**********
```

名称:func_RowToCol

参数说明:

viewName 视图名称,实际上可以是数据库的表格名称,视图名称,也可以是 SQL 语句.

grpCols 需要分组的列,以格式 col1,col2..coln 传入,其中 n 是大于 0 的整数

colCol 由行转为列的那个列

valueCol 行转列后,依然作为值填充的那个列,只能是一个列

--viewIsSql 视图是否是 sql 语句,如果是则传入 1,反之传入 2,默认是 1(是 sql)

fillEmptyWithZero 用 0 来填充空值,默认空值依然保留空值.如果是 1,则只对 valueCol 为数值类型的有效.

rowOrder 结果默认的排序语句,如果有,则使用这个，这个是对结果的行排序

colOrder 对转成的列进行排序的依据.

rowOrderinGrp 行的排序列是否在分组列（grpcols)中，0 表示不是，1 表示是，默认是在分组列中。

colOrderStyle 这个参数说明了列的排序方式

fillValue 填充值，如果非空,且 fillEmptyWithZero=1，则用.

举例:有一个表格 EmpSalary(SalMonth number,EmpName varchar2(20),salary number) 其中 salMonth,EmpName 组成唯一约束

假设有以下数据:

SALMONTH EMPNAME SALARY

---------- -------------------- ----------

200801 lzf 8000

200801 wth 8000

200801 lxl 7000

200801 fjl 8000

200801 wcl 40000

200802 lzf 9000

200802 wth 8000

....

现在需要按照这样的格式输出

salaryMonth lzf wth lxl fjl wcl

200801 8000 8000 7000 8000 40000

200801 9000 8000

那么参数应该这样传递 func_RowToCol('empsalary','','salarymonth','empname','salary',0,1);

输出:

如果成功,则返回一个基于 tempdata_manycols 的查询 sql 字符串

如果失败,则返回空值.

注意事项:

本函数是基于一个叫 tempdata_manyCols 的全局临时表处理的.

并且有以下假设:

1)固定列加动态列不超过 300 列

2)原来的一个列(只能有一个列)作为行内容填充的新形成后的表格中.

3)只能处理数字或者字符的信息,如果是字符,不能超过 2000 个.但本函数的 colCol 的值不应该超过 30 个 B.

因为太长的话,行转列就没有意义了(根本没有办法看),同时 oracle 也不支持超过 30B 的列名

4)**********不建议的事情

****

*不建议对一个巨大的记录集合进行行专列操作,否则可能效率之低下是难于想象

*盖因为行转列通常用于编写报表之用.

*也严重不建议,传入的视图是基于一个很耗时的复杂查询

*最后,如果您的 sql 大于 32K 左右,本过程无能为力!

****

5)严重警告:数据源必须有数据,其次分组列应该都有数据,

6)不接受需要把数据聚集之后再显示的数据,最好自行先聚集.

修改记录:

2009-03-11 lzf 新增

2009-03-12 lzf 完成初稿,可以在简单代码上测试成功.

2009-03-27 lzf 增加一个控制转换列输出的功能 colOrder

2009-04-01 lzf 修改，以便更完善

2009-04-07 lzf 修改，增加了列排序的方式,为了节约时间，不再调整参数顺序.

增加了一个填充值，以便按照要求来填充需要的内容.

****

****

```
*/
Pragma Autonomous_Transaction;

vResultsql varchar2(32767):='';

-----处理临时数据的变量
vDatas type_str_arrs:=type_str_arrs();

vDealRows pls_integer:=0;
----最终存放数据的地方
vColNames type_str_arr:=type_str_arr(); --列名数组
vColAmount pls_integer:=0; --列的个数
vRowNames type_str_arr:=type_str_arr(); --行的内容

vGrpColAmount pls_integer:=0; --分组字段的个数,即 grpCols 的字段个数.
--vStarColPos pls_integer:=1; --返回的起点列标号，默认是 1，但是如果行分组列不在其中，则从 2 开始
vSortAmount pls_integer:=0;

vIntoSqls Varchar2(32767):=''; --用于存储插入到 tempdata_manycols 的 into 脚本
vJoinCols Varchar2(32767):=''; --插入和分组的字段
vOrderCols Varchar2(32767):=''; --排序的字段
vOrderCols2 Varchar2(32767):='';

vRecordAmount pls_integer:=0; --原始数据记录数

--填充值
vFillValue Varchar2(1000):='';

v_sSql Varchar2(32767);
```

```
Function getGrpColAmount(pGrpCols In Varchar2,vRows In Out type_str_arr) Return Pls_Integer
Is
/*
本函数的作用:计算分组字段的个数
*/

Begin
--一个自定义的函数,用于把用特定符号隔开的字符串分解到一个字符串数组中.
pkg_bit.SpilitStr(pGrpcols,',',vRows);
Return vRows.Count;
End;

Function getJoinSql(pGrpcols In Varchar2) Return Varchar2
Is
/*
本函数的作用:返回 join 中的条件,已经假定,两个表一定是 x,y
*/
vCols type_str_arr:=type_str_arr();
vResult Varchar2(32767):='';
vColName Varchar2(30):='' ; --列名
Begin
--一个自定义的函数,用于把用特定符号隔开的字符串分解到一个字符串数组中.
pkg_bit.SpilitStr(pGrpcols,',',vCols);
For i In 1..vCols.Count Loop
vColName:=vCols(i);
If i=1 Then
vResult:='y.'||vColName||'=x.'||vColName;
Else
vResult:=vResult||' and y.'||vColName||'=x.'||vColName;
End If;
End Loop;
```

```
Return vResult;
End;


Function getIntoSql(pGrpCols In Varchar2,pColCol In Varchar2:='',pRowOrderCol In Varchar2:='') Return
Varchar2
Is
/*
本函数的作用:根据分组字段和转列来决定插入到临时表中所需要的子句 sql
pColCol 目前是没有什么意义存在的.
*/
vTempStr Varchar2(32767):=pGrpCols||','||Case When pColCol Is Not Null  Then pColCol||',' Else '' End;
vPos Pls_Integer;
vAmount Pls_Integer:=0;
vResult Varchar2(32767):='';
Begin
--行排序列放在首位.
vTempstr:=Case When pRowOrderCol Is Not Null Then pRowOrderCol||',' Else '' End ||vTempstr;

vPos:=instr(vTempStr,',');
While vpos>0 Loop
vAmount:=vAmount+1;
--组成输出字段
If vAmount=1 Then
vResult:='C'||to_char(vAmount);
Else
vResult:=vResult||',C'||to_char(vAmount);
End If;
vTempStr:=substr(vTempStr,vpos+1);
vPos:=instr(vTempStr,',');
End Loop;
```

```
Return vResult;


End;


Function getOrderSql(pGrpCols In Varchar2) Return Varchar
Is
/*
本函数的作用:组成排序字段.
*/
Begin
Return getIntoSql(pGrpCols,'');
End;




Begin
--0)获得分组字段的个数
vGrpColAmount:=getGrpColAmount(grpCols,vRowNames);

--1)获得转为列之后的列名,列个数
v_sSql:='
Select Distinct '||colCol||' from ( '||viewName||' ) order by '||colCol;
Execute Immediate v_sSql Bulk Collect Into vColNames;
vColAmount:=vColNames.Count;



--2)把数据填充到临时数组表中

If colOrder Is Not Null Then --是用这个.
vOrderCols2:=' x.'||replace(grpCols,',',',x.')||',x.'||colOrder||colOrderStyle;
Else
```

```
vOrderCols2:= ' x.'||replace(grpCols,',',',x.')||',x.'||colCol||colOrderStyle;
End If;

--计算填充内容
If fillEmptyWithZero=1 And fillValue Is Not Null Then
vFillValue:=fillValue;
Elsif fillEmptyWithZero=1 And fillValue Is Null Then
vFillValue:='0';
Elsif fillEmptyWithZero=0 Then
vFillValue:='';
End If;



Execute Immediate 'truncate table tempdata_manyCols';
vJoinCols:=getJoinSql(grpCols);
v_sSql:='
select type_str_arr('||Case When rowOrderinGrp=0 Then 'x.'||rowOrder||','
Else '' End||'x.'||replace(grpCols,',',',x.')||',x.'||colCol||','||
Case When vFillValue Is Not Null Then 'nvl('||valueCol||','"||vFillValue||"') '
Else valueCol
End ||') from
(
select a.*,b.* from
(Select Distinct '||colCol||Case When colOrder Is Null Then '' Else ','||colOrder End||' from ('||viewName||')) a,
(select distinct '||grpCols||Case When rowOrderinGrp=0 Then ','||rowOrder Else '' End||' from ('||viewName||')) b
) x
left join ('||viewName||') y on '||vJoinCols||' and y.'||colCol||'=x.'||colCol||'
order by '||vOrderCols2;
Execute Immediate v_sSql Bulk Collect Into vDatas;
Commit;
```

```
--3)通过矩阵转换,读取 vDatas 的内容,插入到 tempdata_manycols
vRecordAmount:=vDatas.Count;
--设置插入列的 SQL
If rowOrderinGrp=1 Then --如果排序行的列在分组列中，那么就仅仅使用分组列即可.
vIntoSqls:=getIntoSql(grpCols);
Else --如果行的排序列不在分组列中，那么就再多插入一个列.
vIntoSqls:=getIntoSql(grpCols,'',rowOrder);
vSortAmount:=1;
End If;

/*组合 sql
*/
vDealRows:=0;
For i In 1..vRecordAmount Loop --每一行
If Mod(i,vColAmount)=0 Then --当读取到足够一行数据的时候
vDealRows:=vDealRows+1;
If vDealRows=1 Then --为了不至于搞错顺序,第一次需要把转列内容填到
--vColNames,也就是例子中的 lzf,wth...等部分
--同时,组合成真正的 intosql
For j In 1 ..vColAmount Loop
--vsortAmount 表示的是行排序列的个数.
vColNames(j):=vDatas(j)(vGrpColAmount+vSortAmount+1);
--dbms_output.put(lpad( vColNames(j-(vDealRows-1)*vColAmount),20,' '));
vIntoSqls:=vIntoSqls||',C'||to_char(vGrpColAmount+vSortAmount+j);
End Loop;

End If;

v_sSql:='';
For x In 1..vGrpColAmount+vSortAmount Loop --设置值部分分组部分的内容和排序部分内容
v_sSql:=v_sSql||Case x When 1 Then '' Else ',' End||''''||vDatas(i)(x)||'''';
```

```
End Loop;
--读取值部分填充值的信息
--真正的值的位置=分组字段个数＋排序列个数＋2
For j In (vDealRows-1)*vColAmount+1..vDealRows*vColAmount Loop
v_sSql:=v_sSql||','||Case When vDatas(j)(vGrpColAmount+vSortAmount+ 2) Is Null Then 'null'
Else ''''||vDatas(j)(vGrpColAmount+vSortAmount+2)||''''
End;
--dbms_output.put(lpad( vDatas(j)(vGrpColAmount+2),20,' '));
End Loop;
--dbms_output.put_line('');

--组合成最后的 SQL
v_sSql:=' insert into tempdata_manycols('||vIntoSqls||') values('
||v_sSql||')';
--dbms_output.put_line(v_sSql);
Execute Immediate v_sSql;
End If;
End Loop;



Commit;
--4)形成返回的 sql
/*
关键在于知道列名:=分组名称+行转列
*/
/* If rowOrder Is Not Null And rowOrderinGrp=0 Then
vStarColPos:=2;
End If;*/
vResultsql:='select ';
--分别是排序列，和转换列
For i In 1..vGrpColAmount+vColAmount Loop
```

```
If i<=vgrpcolamount Then --前一部分的字段别名是分组字段名称,是外部传入的.
vResultsql:=vResultsql||'
'||(Case i When 1 Then '' Else ',' End)||'C'||(i+vSortAmount)||' as '||vRowNames(i);
Else --后面一部分则是由原来行转过来的列的别名.
vResultsql:=vResultsql||'
'||(Case i When 1 Then '' Else ',' End)||'C'||(i+vSortAmount)||' as '''||vColNames(i-vGrpColAmount)||'''';
End If;
End Loop;
If rowOrder Is Null Then
vOrderCols:=getordersql(grpCols);
Else
--vOrderCols:=rowOrder;
--默认只有一个的情况下.正确的情况，应该是另外处理。
vOrderCols:=' c1 ';
End If;
vResultsql:=vResultsql||'
from tempdata_manycols
order by '||vOrderCols;
return(vResultsql);
Exception
When Others Then
Rollback;
Return '';
end func_RowToCol;
```

测试如下:

```
SQL> select * from empsalary;

SALMONTH EMPNAME SALARY COUNTRY
```

---------- -------------------- ---------- --------------------

200801 lzf 8000 中国

200801 wth 8000 美国

200801 lxl 7000 日本

200801 fjl 7000 巴基斯坦

200801 wcl 40000 美国

200802 lzf 9000 中国

200802 wth 8000 美国

200802 lxl 8500 日本

8 rows selected

SQL> Select func_RowToCol(' empsalary ','salmonth,country','empname','salary') From dual;

FUNC_ROWTOCOL('EMPSALARY','SAL

----

select

C1 as salmonth

,C2 as country

,C3 as "fjl"

,C4 as "lxl"

,C5 as "lzf"

,C6 as "wcl"

,C7 as "wth"

from tempdata_manycols

order by C1,C2

SQL> select

2 cast(C1 As Varchar2(10)) as salmonth

3 ,cast(C2 As Varchar2(10)) as country

```
4 ,cast(C3 As Varchar2(10)) as "fjl"

5 ,cast(C4 As Varchar2(10)) as "lxl"

6 ,cast(C5 As Varchar2(10)) as "lzf"

7 ,cast(C6 As Varchar2(10)) as "wcl"

8 ,cast(C7 As Varchar2(10)) as "wth"

9 from tempdata_manycols

10 order by C1,C2

11 /
```

SALMONTH COUNTRY fjl lxl lzf wcl wth