行列转换总结

Caizhuoyi

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。

2. 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;
```

2.1 UNION ALL

适用范围：8i,9i,10g 及以后版本

```
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;
```

若空行不需要转换，只需加一个 where 条件，
WHERE COLUMN IS NOT NULL 即可。

2.2 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;
```

2.3 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;
```

3. 行转列

```
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;
```

3.1 AGGREGATE FUNCTION

适用范围：8i,9i,10g 及以后版本

```
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;
```

MAX 聚集函数也可以用 sum、min、avg 等其他聚集函数替代。

被指定的转置列只能有一列，但固定的列可以有多列，请看下面的例子：

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

这里转置列为 empno，固定列为 mgr，deptno。

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;
```

3.2 PL/SQL

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;
/
```

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;
```

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;
```

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;
```

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;
```

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;
```

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;
```

5.4 SYS_CONNECT_BY_PATH

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;
```

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;
```

5.5 WMSYS.WM_CONCAT

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。

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

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;
```

6.1 SUBSTR + INSTR

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;
```

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;
```

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;
```

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;
```

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;
```

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;
```

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;
```

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;
```

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
```