SQLでデータベーステーブルの縦・横変換

2006年07月10日

クエリを思いつかなかった子のために。これはあくまでヒントだぞ。 以降の例では Oracle9iリリース２で説明しています。

テーブルの横→縦変換

データベースのテーブルには、何故か無駄に横長のものがあったりします。 ホスト時代のデータセットイメージをそのままテーブルフォーマットにしちゃったようなやつですね。項目名に「FILLER1□とか名前がついている（ ▽ ；

こんなテーブルがあるとします。

```
SQL> DESC 売上ランキング横;
名前            NULL? 型
 ------------------------------ -------- ----------------------------
売上年                                    VARCHAR2(4)
１位製品                                  VARCHAR2(10)
１位売上              NUMBER(5)
２位製品                                  VARCHAR2(10)
２位売上              NUMBER(5)
３位製品                                  VARCHAR2(10)
３位売上              NUMBER(5)
４位製品                                  VARCHAR2(10)
４位売上              NUMBER(5)
５位製品                                  VARCHAR2(10)
５位売上              NUMBER(5)

SQL>  SELECT * FROM 売上ランキング横;

売上 １位製品   １位売上 ２位製品   ２位売上 ３位製品   ３位売上 ４位製品   ４位売上 ５位製品   ５
位売上
 ---- ---------- ---------- ---------- ---------- ---------- ---------- ----
------ ---------- ---------- ----------
 2004 みかん        40 りんご        37 ばなな        24 すもも        15 ぶどう         5
 2005 みかん        60 ばなな        30 りんご        29 すもも        10 ぶどう         3
 2006 みかん        90 まんごー       88 さくらんぼ      70 ぶどう         10 めろん         6

経過: 00:00:00.00
SQL>
```

みかん強いですねー。売上げの単位は何円なんだろう。さて。

先輩：「**あのさ、１レコードに**製品一つだけのテーブルを作ってよ。**プリンタ、**桁が少ないんで横長で出すの無理なんだわ」
後輩：「**どんなおもちゃ使ってるんです**」
先輩：「**うるさい早くやれ**」

```
SQL> DESC 売上ランキング縦;
名前          NULL? 型
 ----------------------------- -------- ----------------------------
売上年                                    VARCHAR2(4)
順位              NUMBER(1)
製品                                      VARCHAR2(10)
売上              NUMBER(5)

SQL>
```

テーブルはこんなものでよいでしょう。
さて、売上ランキング横 → 売上ランキング縦 へどうやって変換しますか？
Excelに落としてカット＆ペーストで変形する？ご冗談を(^^
まぁレコード数少ないんでそれでもいいですけど、もし「 売上年」が「 売上年月日 」だったらどうします？１０００レコード超えますぞ。

□□□

実際にはこんな□□□で変換できます。

```
SQL> INSERT INTO 売上ランキング縦
  2    SELECT *
  3      FROM (
  4                   SELECT 売上年, 1, "1位製品", "1位売上" FROM 売上ランキング横
  5             UNION SELECT 売上年, 2, "2位製品", "2位売上" FROM 売上ランキング横
  6             UNION SELECT 売上年, 3, "3位製品", "3位売上" FROM 売上ランキング横
  7             UNION SELECT 売上年, 4, "4位製品", "4位売上" FROM 売上ランキング横
  8             UNION SELECT 売上年, 5, "5位製品", "5位売上" FROM 売上ランキング横
  9           )
 10  ;

15行が作成されました。

経過: 00:00:00.00
SQL> SELECT * FROM 売上ランキング縦 ORDER BY 売上年, 順位;

売上    順位 製品       売上
---- ---------- ---------- ----------
2004          1 みかん       40
2004          2 りんご       37
2004          3 ばなな       24
2004          4 すもも       15
2004          5 ぶどう        5
2005          1 みかん       60
2005          2 ばなな       30
2005          3 りんご       29
2005          4 すもも       10
2005          5 ぶどう        3
2006          1 みかん       90
2006          2 まんごー      88
```

```
2006          3 さくらんぼ     70
2006          4 ぶどう        10
2006          5 めろん         6

15行が選択されました。

経過: 00:00:00.00
SQL>
```

単に縦長にして結果を見たいだけなら□□□最初の“ INSERT INTO 売上ランキング縦 ”はいらないです□ FROM句のサブクエリとして SELECT 文を横項目の数だけ書き連ねます。 一見面倒に見えますが□Excel の切り張りをすることを考えたら全然楽です。

ポイントはUNION 演算子で各SELECT文の結果を統合しているところです。
UNION演算子は、各SELECT文の結果を縦方向に繋ぎ合わせるような働きをします。
この時、重複するようなレコードが存在したら１レコードに纏められます。
重複を許したい時は“ UNION ALL”演算子を使ってください。

以下の例でその違いを見てもらえればいいかな。

簡単なUNION□UNION ALLの例

```
SQL> CREATE TABLE X( ITEM CHAR(2), VAL NUMBER(1,0) );

表が作成されました。

経過: 00:00:00.00
SQL> INSERT INTO X VALUES('A',1);
   2 INSERT INTO X VALUES('A',2);
   3 INSERT INTO X VALUES('A',1);
   4 INSERT INTO X VALUES('B',2);

4行が作成されました。

経過: 00:00:00.00
SQL> SELECT * FROM X;

IT         VAL
-- ----------
A           1
A           2
A           1
B           2

経過: 00:00:00.00
SQL> SELECT * FROM X WHERE ITEM='A'
  2  UNION
  3  SELECT * FROM X WHERE ITEM='B'
  4  ;
```

```
IT         VAL
-- ----------
A           1
A           2
B           2

経過: 00:00:00.00
SQL> SELECT * FROM X WHERE ITEM='A'
  2  UNION ALL
  3  SELECT * FROM X WHERE ITEM='B'
  4  ;

IT         VAL
-- ----------
A           1
A           2
A           1
B           2

経過: 00:00:00.00
SQL>
```

UNION ALL では全レコードが出力されていることがわかります。
UNION では ('A' ,1)のレコードが統合され１レコードのみになっています。

次の例への下準備

先輩：「さっきの縦テーブルにだいずとここあ追加しといてくれない？」
後輩：「わかりました」

```
 SQL> INSERT INTO 売上ランキング縦 VALUES('2007', 3, 'だいず', 100);
    2 INSERT INTO 売上ランキング縦 VALUES('2007', 2, 'ここあ', 290);
2行が作成されました。

経過: 00:00:00.00
SQL> SELECT * FROM 売上ランキング縦 ORDER BY 売上年, 順位;

売上   順位 製品       売上
---- ---------- ---------- ----------
2004          1 みかん         40
2004          2 りんご         37
2004          3 ばなな         24
2004          4 すもも         15
2004          5 ぶどう          5
2005          1 みかん         60
2005          2 ばなな         30
2005          3 りんご         29
2005          4 すもも         10
2005          5 ぶどう          3
2006          1 みかん         90
```

```
2006           2 まんごー       88
2006           3 さくらんぼ      70
2006           4 ぶどう        10
2006           5 めろん         6
2007           2 ここあ        290
2007           3 だいず        100

17行が選択されました。

経過: 00:00:00.00
SQL>
```

後輩：「テレビすげーなぁあ…」

テレビの影響ってすごいね。「だいず」と「ここあ」って例の番組で出てたやつじゃないか？(^^

テーブルの縦→横変換

先輩：「ごめん、さっき横長テーブル消しちゃったんだわ。縦長テーブルからデータ作り直してくれる？」
後輩：「てめーぶん殴る」
先輩：「３倍返ししてもいいなら殴って良いぞ」

どうやって縦方向のレコードを横に並べるか。横に繋ぐ□□□構文を探しても多分見つかりません。

「じゃあ、キーでOUTER JOINをやるか…」

それでもいいんですけどね。ちょっとした工夫で OUTER JOIN はいらなくなります。それに、状況によってはOUTER JOINで結合できない事もあるのです□OUTER JOIN抜きでやってみましょうか。
たとえば、次のような□□□を実行してみます。

基本的考え方

```
SQL>  SELECT 売上年, CASE 順位 WHEN 1 THEN 製品 ELSE '' END AS "1位製品", CASE 順
位 WHEN 1 THEN 売上 ELSE NULL END AS "1位売上"
  2                 , CASE 順位 WHEN 2 THEN 製品 ELSE '' END AS "2位製品", CASE
順位 WHEN 2 THEN 売上 ELSE NULL END AS "2位売上"
  3                 , CASE 順位 WHEN 3 THEN 製品 ELSE '' END AS "3位製品", CASE
順位 WHEN 3 THEN 売上 ELSE NULL END AS "3位売上"
  4                 , CASE 順位 WHEN 4 THEN 製品 ELSE '' END AS "4位製品", CASE
順位 WHEN 4 THEN 売上 ELSE NULL END AS "4位売上"
  5                 , CASE 順位 WHEN 5 THEN 製品 ELSE '' END AS "5位製品", CASE
順位 WHEN 5 THEN 売上 ELSE NULL END AS "5位売上"
  6       FROM 売上ランキング縦
  7  ;

売上 1位製品   1位売上 2位製品   2位売上 3位製品   3位売上 4位製品   4位売上 5位製品   5
位売上
---- ---------- ---------- ---------- ---------- ---------- ---------- ----
------ ---------- ---------- ----------
```

```
 2004 みかん       40
 2004             りんご       37
 2004                       ばなな       24
 2004                                 すもも       15
 2004                                           ぶどう       5
 2005 みかん       60
 2005             ばなな       30
 2005                       りんご       29
 2005                                 すもも       10
 2005                                           ぶどう       3
 2006 みかん       90
 2006             まんごー      88
 2006                       さくらんぼ     70
 2006                                 ぶどう       10
 2006                                           めろん       6
 2007                       だいず      100
 2007             ここあ       290

 17行が選択されました。

経過: 00:00:00.00
 SQL>
```

確かに横に並ぶけどなー、なんか歯抜けだらけだし違うじゃないの？とおっしゃるあなた。
ここであなたの想像力・思考力が問われます。 この□□□の実行結果は、横に並べたと言うよりは、縦長テーブルの項目数を増やした状態と思ってください。

SELECT文中のCASE構文で、値を表示する項目の表示とダミー表示□□□□□□の判定を行っています。
項目「売上年」の隣に来る項目は「順位が1位の製品」です。その判定を項目「順位」で行っています。
その隣は「順位が2位の製品」です。その判定をやはり項目「順位」で行っています。
これを5位まで繰り返せば、上記のような結果となります。

この□□□にちょっとした追記を行うだけでお待ちかねのモノが出来上がるのです。

□□□

```
 SQL>   SELECT 売上年, MAX(CASE 順位 WHEN 1 THEN 製品 ELSE '' END) AS "1位製品",
SUM(CASE 順位 WHEN 1 THEN 売上 ELSE NULL END) AS "1位売上"
   2                  , MAX(CASE 順位 WHEN 2 THEN 製品 ELSE '' END) AS "2位製品",
SUM(CASE 順位 WHEN 2 THEN 売上 ELSE NULL END) AS "2位売上"
   3                  , MAX(CASE 順位 WHEN 3 THEN 製品 ELSE '' END) AS "3位製品",
SUM(CASE 順位 WHEN 3 THEN 売上 ELSE NULL END) AS "3位売上"
   4                  , MAX(CASE 順位 WHEN 4 THEN 製品 ELSE '' END) AS "4位製品",
SUM(CASE 順位 WHEN 4 THEN 売上 ELSE NULL END) AS "4位売上"
   5                  , MAX(CASE 順位 WHEN 5 THEN 製品 ELSE '' END) AS "5位製品",
SUM(CASE 順位 WHEN 5 THEN 売上 ELSE NULL END) AS "5位売上"
   6       FROM 売上ランキング縦
   7     GROUP BY 売上年
   8     ORDER BY 売上年
   8  ;
```

```
売上 1位製品    1位売上 2位製品    2位売上 3位製品    3位売上 4位製品    4位売上 5位製品    5
位売上
 ---- ---------- ---------- ---------- ---------- ---------- ---------- ----
------ ---------- ---------- ----------
 2004 みかん        40 りんご        37 ばなな        24 すもも        15 ぶどう         5
 2005 みかん        60 ばなな        30 りんご        29 すもも        10 ぶどう         3
 2006 みかん        90 まんごー       88 さくらんぼ      70 ぶどう        10 めろん         6
 2007              ここあ       290 だいず        100

経過: 00:00:00.00
 SQL>
```

今度の□□□の結果は、見事に横並びとなっています。
□□□的には項目「製品」をMAX関数に与え、項目「売上」をSUM関数に与えています。
そして□GROUP BY 句で「売上年」が同じレコードをグルーピングしています。

もう一度、最初の□□□の結果の売上年 “ 2004 ” の部分を見てみます。

```
売上 1位製品    1位売上 2位製品    2位売上 3位製品    3位売上 4位製品    4位売上 5位製品    5
位売上
 ---- ---------- ---------- ---------- ---------- ---------- ---------- ----
------ ---------- ---------- ----------
 2004 みかん        40
 2004              りんご        37
 2004                          ばなな        24
 2004                                      すもも        15
 2004                                                  ぶどう         5
                    ├┬─┤
       ├┬─┤     □└SUMの結果は40
      □└MAXでみかんを選択
```

MAX関数は、グルーピングされた項目内で、一番大きな値を持つ項目を返します。
これは数値のほか、文字列でも可能で、文字列の場合は「文字コード＆文字列長さ」により大小関係が計算されます。
SUM関数は、グルーピングされた項目の集計を行った結果を返します。

１位製品の部分を見ると、「みかん」の他は全て長さゼロの文字列(NULL)です□MAX関数は みかん と 長さゼロの文字列 の中から最大値である みかん を選び出します。
１位売上の部分を見ると 40 の値がひとつだけで後は全てNULLです□SUM関数は 40 と NULL を集計します□NULLは計算の対象とされないので、40 が結果となります。

同じように２位～５位もMAX関数およびSUM関数で計算され、結果、売上年単位で集約された１レコードに変換されます。

この結果を横長テーブルへ入れればよいのです。

```
 SQL> INSERT INTO 売上ランキング横
   2     SELECT 売上年 , MAX(CASE 順位 WHEN 1 THEN 製品 ELSE '' END) AS "1位製品",
SUM(CASE 順位 WHEN 1 THEN 売上 ELSE NULL END) AS "1位売上"
   3                   , MAX(CASE 順位 WHEN 2 THEN 製品 ELSE '' END) AS "2位製品",
SUM(CASE 順位 WHEN 2 THEN 売上 ELSE NULL END) AS "2位売上"
   4                   , MAX(CASE 順位 WHEN 3 THEN 製品 ELSE '' END) AS "3位製品",
SUM(CASE 順位 WHEN 3 THEN 売上 ELSE NULL END) AS "3位売上"
```

```
   5                   , MAX(CASE 順位 WHEN 4 THEN 製品 ELSE '' END) AS "4位製品",
SUM(CASE 順位 WHEN 4 THEN 売上 ELSE NULL END) AS "4位売上"
   6                   , MAX(CASE 順位 WHEN 5 THEN 製品 ELSE '' END) AS "5位製品",
SUM(CASE 順位 WHEN 5 THEN 売上 ELSE NULL END) AS "5位売上"
   7       FROM 売上ランキング縦
   8      GROUP BY 売上年
   9   ;

 4行が作成されました。

経過: 00:00:00.00
 SQL> SELECT * FROM 売上ランキング横 ORDER BY 売上年;

売上 1位製品   1位売上 2位製品   2位売上 3位製品   3位売上 4位製品   4位売上 5位製品   5
位売上
 ---- ---------- ---------- ---------- ---------- ---------- ---------- ----
------ ---------- ---------- ----------
 2004 みかん        40 りんご        37 ばなな        24 すもも        15 ぶどう         5
 2005 みかん        60 ばなな        30 りんご        29 すもも        10 ぶどう         3
 2006 みかん        90 まんごー       88 さくらんぼ      70 ぶどう         10 めろん        6
 2007             ここあ       290 だいず       100

経過: 00:00:00.00
 SQL>
```

追加した 2007 年のレコードもきちんと処理されました。

注意点

横に並べる為には、横に並べる順番を示す項目が元のテーブルに存在しなければなりません。
本例では、項目「順位」を使いました。

追記:OUTER JOINで書く場合

OUTER JOIN で書くとどうなるかという話。
１位～５位の各レコードを別けて個別に仮テーブルとみなしそれらをJOINする、という流れで書いたのが以下。

```
 SQL> SELECT A.売上年
   2         ,A.製品 AS "1位製品"
   3         ,A.売上 AS "1位売上"
   4         ,B.製品 AS "2位製品"
   5         ,B.売上 AS "2位売上"
   6         ,C.製品 AS "3位製品"
   7         ,C.売上 AS "3位売上"
   8         ,D.製品 AS "4位製品"
   9         ,D.売上 AS "4位売上"
  10         ,E.製品 AS "5位製品"
  11         ,E.売上 AS "5位売上"
```

```
  12    FROM
  13     ( SELECT 売上年, 製品, 売上 FROM 売上ランキング縦 WHERE 順位=1 ) A
  14        LEFT OUTER JOIN ( SELECT 売上年, 製品, 売上 FROM 売上ランキング縦 WHERE 順
位=2 ) B ON (A.売上年=B.売上年)
  15         LEFT OUTER JOIN ( SELECT 売上年, 製品, 売上 FROM 売上ランキング縦 WHERE
順位=3 ) C ON (B.売上年=C.売上年)
  16          LEFT OUTER JOIN ( SELECT 売上年, 製品, 売上 FROM 売上ランキング縦 WHERE
順位=4 ) D ON (C.売上年=D.売上年)
  17           LEFT OUTER JOIN ( SELECT 売上年, 製品, 売上 FROM 売上ランキング縦
WHERE 順位=5 ) E ON (D.売上年=E.売上年)
  18   ;

 売上 1 位製品   1 位売上 2 位製品   2 位売上 3 位製品   3 位売上 4 位製品   4 位売上 5 位製品   5
位売上
 ---- ---------- ---------- ---------- ---------- ---------- ---------- ----
------ ---------- ---------- ----------
 2004 みかん       40 りんご       37 ばなな       24 すもも       15 ぶどう        5
 2005 みかん       60 ばなな       30 りんご       29 すもも       10 ぶどう        3
 2006 みかん       90 まんごー      88 さくらんぼ     70 ぶどう       10 めろん        6

 経過: 00:00:00.00
 SQL>
```

Oracleっぽくない、って言う事であるなら以下でもいい。

```
 SQL> SELECT A.売上年
   2         ,A.製品 AS "1 位製品"
   3         ,A.売上 AS "1 位売上"
   4         ,B.製品 AS "2 位製品"
   5         ,B.売上 AS "2 位売上"
   6         ,C.製品 AS "3 位製品"
   7         ,C.売上 AS "3 位売上"
   8         ,D.製品 AS "4 位製品"
   9         ,D.売上 AS "4 位売上"
  10         ,E.製品 AS "5 位製品"
  11         ,E.売上 AS "5 位売上"
  12    FROM ( SELECT 売上年, 製品, 売上 FROM 売上ランキング縦 WHERE 順位=1 ) A
  13         ,( SELECT 売上年, 製品, 売上 FROM 売上ランキング縦 WHERE 順位=2 ) B
  14         ,( SELECT 売上年, 製品, 売上 FROM 売上ランキング縦 WHERE 順位=3 ) C
  15         ,( SELECT 売上年, 製品, 売上 FROM 売上ランキング縦 WHERE 順位=4 ) D
  16         ,( SELECT 売上年, 製品, 売上 FROM 売上ランキング縦 WHERE 順位=5 ) E
  17   WHERE A.売上年=B.売上年(+)
  18     AND B.売上年=C.売上年(+)
  19     AND C.売上年=D.売上年(+)
  20     AND D.売上年=E.売上年(+)
  21   ;

 売上 1 位製品   1 位売上 2 位製品   2 位売上 3 位製品   3 位売上 4 位製品   4 位売上 5 位製品   5
位売上
 ---- ---------- ---------- ---------- ---------- ---------- ---------- ----
------ ---------- ---------- ----------
```

```
 2004 みかん       40 りんご       37 ばなな       24 すもも       15 ぶどう        5
 2005 みかん       60 ばなな       30 りんご       29 すもも       10 ぶどう        3
 2006 みかん       90 まんごー      88 さくらんぼ     70 ぶどう       10 めろん        6

経過: 00:00:00.00
 SQL>
```

問題がお判りであろうか。…そう、2007年のレコードが無い。
2007年のデータは、2位と3位のレコードしかない。よって、1位の仮テーブルを作ると2007年のレコードは存在しないので、後続の仮テーブルにいくら2007年のレコードが存在しても結合が出来ない事になります。

つまり、欠損レコードが存在する場合にはそれを補う様なレコードを補填しない限り意図通り結合は出来ないのです。

欠損を補う例としては以下の様な方法があります。

```
 SQL> DESC 売上年;
 名前                                      NULL?    型
  ----------------------------------------- -------- ----------------------------
-----
 売上年                                                VARCHAR2(4)

経過: 00:00:00.00
 SQL> SELECT * FROM 売上年;

売上
 ----
 2004
 2005
 2006
 2007

経過: 00:00:00.00
 SQL> SELECT T.売上年
   2         ,A.製品 AS "1位製品"
   3         ,A.売上 AS "1位売上"
   4         ,B.製品 AS "2位製品"
   5         ,B.売上 AS "2位売上"
   6         ,C.製品 AS "3位製品"
   7         ,C.売上 AS "3位売上"
   8         ,D.製品 AS "4位製品"
   9         ,D.売上 AS "4位売上"
  10         ,E.製品 AS "5位製品"
  11         ,E.売上 AS "5位売上"
  12     FROM
  13       ( SELECT 売上年 FROM 売上年 ) T
  14         LEFT OUTER JOIN( SELECT 売上年, 製品, 売上 FROM 売上ランキング縦 WHERE 順
位=1 ) A ON (T.売上年=A.売上年)
  15          LEFT OUTER JOIN ( SELECT 売上年, 製品, 売上 FROM 売上ランキング縦 WHERE 順
位=2 ) B ON (T.売上年=B.売上年)
  16           LEFT OUTER JOIN ( SELECT 売上年, 製品, 売上 FROM 売上ランキング縦 WHERE
```

```
順位=3 ) C ON (T.売上年=C.売上年)
  17           LEFT OUTER JOIN ( SELECT 売上年, 製品, 売上 FROM 売上ランキング縦 WHERE
順位=4 ) D ON (T.売上年=D.売上年)
  18            LEFT OUTER JOIN ( SELECT 売上年, 製品, 売上 FROM 売上ランキング縦
WHERE 順位=5 ) E ON (T.売上年=E.売上年)
  19    ;

 売上１位製品   １位売上２位製品   ２位売上３位製品   ３位売上４位製品   ４位売上５位製品   ５
位売上
 ---- ---------- ---------- ---------- ---------- ---------- ---------- ----
------ ---------- ---------- ----------
 2004 みかん       40 りんご       37 ばなな       24 すもも       15 ぶどう        5
 2005 みかん       60 ばなな       30 りんご       29 すもも       10 ぶどう        3
 2006 みかん       90 まんごー      88 さくらんぼ     70 ぶどう       10 めろん        6
 2007            ここあ       290 だいず       100

 経過: 00:00:00.00
 SQL>
```

結合のキーは 売上年 なので、全売上年を保持しているテーブル「 売上年」を作りました。 このテーブルに１位～５位の仮テーブルを結合させています。

処理相手となるデータの性質をきちんと踏まえたうえで、どの様なクエリを出すべきか考えなくてはいけません。 機械的に「縦横変換はこのクエリ！」という風にはいかないので、注意が要ります。

旧ページコメント

- これ、プリントアウトしていいですか？っていうかもうしちゃいましたけど。。。 - むらかみ (2007年09月10日 23時35分01秒)
- いいけど実名は出しちゃ駄目だろ(^^; - 510 (2007年09月11日 11時54分38秒)
- 得ろ意 - 高村 (2010年10月25日 11時02分14秒)
- ちょうど探してたんだ。使わなかったけど。灯台下暗しだな□ - aka (2011年03月25日 14時16分56秒)
- 意外な人がコメント残していくな□□□ - 510 (2011年03月26日 15時11分29秒)

database, SQL, Oracle, 技術資料