SQLで各グループ中の最大値を持つレコードに印を付けたい

2014年8月29日
Oracle 11g Release 11.1 で説明しています。

お客さんへの説明でいろいろやったんでついでに自分の資料用に作ってしまった。

テーブル中の各グループで最大値を持つレコードに印を付ける

各グループ中で最大値を持つレコードに印を付けたい

大きな組織では、複数の活動拠点から日毎のデータが送られてきて、それを元に経営や営業の判断に利用する、という事が普通に行われています。
例えばこんなテーブルがあるとしましょう。

```
SQL> DESC TRANTBL
 Name                                      NULL?    TYPE
 ----------------------------------------- -------- ----------------------
----
 DATE_YM                                            VARCHAR2(6)
 AREACODE                                           NUMBER(2)
 BRANCHCODE                                         NUMBER(5)
 ITEMCODE                                           VARCHAR2(4)
 VOLUME                                             NUMBER
 SALES                                              NUMBER



SQL> SELECT * FROM TRANTBL WHERE DATE_YM='201401';

DATE_YM              AREACODE BRANCHCODE ITEMCODE          VOLUME      SALES
------------------ ---------- ---------- ------------ ---------- ----------
201401                     10      10100 F100                 10        100
201401                     10      10101 F100                 20        200
201401                     10      10102 F100                 15        150
201401                     20      20201 F100                 50        500
201401                     20      20301 F100                  1         10
201401                     20      20302 F100                 30        300
201401                     30      30100 F100                 90        900
201401                     30      30200 F100                 23        230
201401                     30      30202 F100                 22        220
201401                     30      30203 F100                 19        190

10 ROWS selected.

SQL>
```

テーブルには日々レコードが追加され続け、ある程度の期間が過ぎると、該当レコードはテーブルから削除されます。

ですがこのテーブルは以降の話の為に、年月での集約集計が行われた結果が格納されている、という事にしましょう。 この例は2014年1月の情報ですね。

後輩：「先輩、上司から作業指示がありまして。
先輩：「面倒なのはいやだよ？
後輩：「面倒なんですよねー。こんな風に、エリアで売り上げ最大の拠点にマークを付けたリストが欲しいそうです。

```
TOPSALES DATE_YM              AREACODE BRANCHCODE ITEMCODE         VOLUME
SALES
-------- ------------------ ---------- ---------- ------------ ---------- --
--------
         201401                     10      10100 F100                 10
100
Y        201401                     10      10101 F100                 20
200
         201401                     10      10102 F100                 15
150
Y        201401                     20      20201 F100                 50
500
         201401                     20      20301 F100                  1
10
         201401                     20      20302 F100                 30
300
Y        201401                     30      30100 F100                 90
900
         201401                     30      30200 F100                 23
230
         201401                     30      30202 F100                 22
220
         201401                     30      30203 F100                 19
190
```

たぶん、データベースにかかわるお仕事をしていると1回はやる事になるお題です。

ある後輩氏は項目TOPSALESを含むテーブルXを作り、テーブルTRANTBLからXへ必要なレコードをINSERTして、更にテーブルXにUPDATE文でマークを付ける処理実行、という2段階の処理を行いました。
まとめてやろうとすると面倒な事も分割すると楽になるので、そういうのもありなんですが、上記はサブクエリやウインドウ関数を知っていると楽が出来ます。

サブクエリを使った例

先輩：「とりあえずサブクエリの使い方覚えてみよう。
後輩：「え！FROM句以外にも使えるんですか！？
先輩：「少ないデータ量ならこういうやり方がある。

```
SQL> SELECT
  2    CASE WHEN SALES=(SELECT MAX(SALES) FROM TRANTBL WHERE
```

```
DATE_YM=M.DATE_YM AND AREACODE=M.AREACODE) THEN 'Y' ELSE NULL END AS
"TOPSALES"
  3    ,DATE_YM
  4    ,AREACODE
  5    ,BRANCHCODE
  6    ,ITEMCODE
  7    ,VOLUME
  8    ,SALES
  9   FROM TRANTBL M
 10   WHERE M.DATE_YM='201401'
 11   ;

TOP DATE_YM              AREACODE BRANCHCODE ITEMCODE         VOLUME
SALES
--- ------------------ ---------- ---------- ------------ ---------- -------
---
    201401                     10      10100 F100                 10
100
Y   201401                     10      10101 F100                 20
200
    201401                     10      10102 F100                 15
150
Y   201401                     20      20201 F100                 50
500
    201401                     20      20301 F100                  1
10
    201401                     20      20302 F100                 30
300
Y   201401                     30      30100 F100                 90
900
    201401                     30      30200 F100                 23
230
    201401                     30      30202 F100                 22
220
    201401                     30      30203 F100                 19
190

10 ROWS selected.

SQL>
```

後輩：「おー！！こういう使い方が出来るのかー
先輩：「もう少しお勉強しようよ…

CASE構文でサブクエリを使っています。CASE構文では

- 『現在のレコードの項目DATE_YMと項目AREACODEを条件にサブクエリを実行しテーブルTRANTBLのレコード中で最大値を取得し』
- 『現在のレコードのを持つ項目SALESと最大値が同一かどうか』

を判定しています。
現在のレコードの項目SALESと最大値が等しいならそのレコードは最大値を持つレコードとなりますの

で、印である'Y'を、そうでないならNULLを、項目TOPSALESに当てはめる事になります。

構文的には1レコード毎にサブクエリが実行されます。なので項目DATE_YM='201401'□項目AREACODE=10□が3レコードありますが、同じ条件でサブクエリが3回実行されることになります。…データベースの内部処理的には、最適化処理が行われ同条件のサブクエリは1回だけの実行になります。

ウインドウ関数を使った例

先輩：「今度はウインドウ関数を使う例。どっちかというとこっちを覚えたほうがいい。
後輩：「なら最初にこっち示してくれれば良いじゃないですかーヤダー
先輩：「最初のはサブクエリの復習がてら例示しただけだもの。今度はウインドウ関数の使用例。

```
SQL> SELECT
  2    CASE WHEN SALES = MAX(SALES) OVER (PARTITION BY DATE_YM, AREACODE)
THEN 'Y' ELSE NULL END AS "TOPSALES"
  3   ,DATE_YM
  4   ,AREACODE
  5   ,BRANCHCODE
  6   ,ITEMCODE
  7   ,VOLUME
  8   ,SALES
  9  FROM TRANTBL
 10  WHERE DATE_YM='201401'
 11  ;

TOP DATE_YM              AREACODE BRANCHCODE ITEMCODE         VOLUME
SALES
--- ------------------ ---------- ---------- ------------ ---------- -------
---
    201401                    10      10100 F100                 10
100
Y   201401                    10      10101 F100                 20
200
    201401                    10      10102 F100                 15
150
Y   201401                    20      20201 F100                 50
500
    201401                    20      20301 F100                  1
10
    201401                    20      20302 F100                 30
300
Y   201401                    30      30100 F100                 90
900
    201401                    30      30200 F100                 23
230
    201401                    30      30202 F100                 22
220
    201401                    30      30203 F100                 19
190
```

```
10 ROWS selected.

SQL>
```

CASE構文は一緒なんですが、今度は直接集計関数MAXを使っています。そして、その後ろ、OVER 句以降にPARTITION BY 句があります。

これは、この集計関数MAXの集計対象レコードが、

- 『現在のSELECT句で対象となるレコードセット(この例ならテーブルTRANTBLの条件DATE_YM='201401'に当てはまるレコードセット)で』
- 『現在のレコード中の項目DATE_YMと項目AREACODEと同じ値を持っているレコード』

である事を指定します。集計関数MAXの集計範囲を指定している訳で、サブクエリで集計する範囲を指定してるのとそれほど変わらないように見えます。

でも、データベースシステム的には結構な違いがあるんですよ。

サブクエリ版とウインドウ関数版の違い

後輩：「どっちも結果が一緒なら二つ覚えなくても良いんじゃないですかね。
先輩：「正確には結果は一緒じゃないんだよね。で、データが少量ならサブクエリの方がやってる事がわかり易い。データ量が大きくなるとウインドウ関数版じゃなきゃ駄目かもしれない。
後輩：「納得いく説明が無いと許しませんよ？

では、先のサブクエリ版の実行計画を覗いてみましょう。

```
SQL> EXPLAIN PLAN FOR
  2  SELECT
  3    CASE WHEN SALES=(SELECT MAX(SALES) FROM TRANTBL WHERE
DATE_YM=M.DATE_YM AND AREACODE=M.AREACODE) THEN 'Y' ELSE NULL END AS
"TOPSALES"
  4   ,DATE_YM
  5   ,AREACODE
  6   ,BRANCHCODE
  7   ,ITEMCODE
  8   ,VOLUME
  9   ,SALES
 10  FROM TRANTBL M
 11  WHERE M.DATE_YM='201401'
 12  ;

Explained.

SQL> SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
----
Plan hash VALUE: 4245077362
```

```
-------------------------------------------------------------------------------
--
| Id  | Operation           | Name    | ROWS  | Bytes | Cost (%CPU)| TIME
|
-------------------------------------------------------------------------------
--
|   0 | SELECT STATEMENT    |         |    10 |   610 |     3   (0)| 00:00:01
|
|   1 |  SORT AGGREGATE     |         |     1 |    31 |            |
|
|*  2 |   TABLE ACCESS FULL| TRANTBL |     1 |    31 |     3   (0)| 00:00:01
|
|*  3 |  TABLE ACCESS FULL | TRANTBL |    10 |   610 |     3   (0)| 00:00:01
|
-------------------------------------------------------------------------------
--


PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
----
Predicate Information (IDENTIFIED BY operation id):
---------------------------------------------------

   2 - FILTER("DATE_YM"=:B1 AND "AREACODE"=:B2)
   3 - FILTER("M"."DATE_YM"='201401')

Note
-----
   - dynamic sampling used FOR this statement

20 ROWS selected.

SQL>
```

後輩：Id 2で読み出した結果をId 1で集計し、Id 0でId 1とId 3をマージしてるんですね。
先輩：「そういう理解でいいと思うよ。

TABLE ACCESS FULLが2回現れている事がわかります。Oracleさんがテーブル TRANTBL全体読み込みを2回繰り返しますよ、と言ってるわけです。
そして、2 - filter(..)を見ると、項目DATE_YMと項目AREACODEで集計条件を指定しているようです。

- Id 2 のテーブル読み込みは、項目DATE_YMと項目AREACODEでグループ化されたレコードに対しての集計関数MAXの値を取得するId 1の処理のため、
- Id 3 のテーブル読み込みはId 1 の結果を使った、印付与判定処理のため、

と考えられます。

次はウインドウ関数版の実行計画です。

```
SQL> EXPLAIN PLAN FOR
  2  SELECT
```

```
  3    CASE WHEN SALES = MAX(SALES) OVER (PARTITION BY DATE_YM, AREACODE)
THEN 'Y' ELSE NULL END AS "TOPSALES"
  4   ,DATE_YM
  5   ,AREACODE
  6   ,BRANCHCODE
  7   ,ITEMCODE
  8   ,VOLUME
  9   ,SALES
 10  FROM TRANTBL
 11  WHERE DATE_YM='201401'
 12  ;

Explained.

SQL> SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
----
Plan hash VALUE: 1075596177

--------------------------------------------------------------------------------
--
| Id  | Operation          | Name    | ROWS  | Bytes | Cost (%CPU)| TIME
|
--------------------------------------------------------------------------------
--
|   0 | SELECT STATEMENT   |         |    10 |   610 |     4  (25)| 00:00:01
|
|   1 |  WINDOW SORT       |         |    10 |   610 |     4  (25)| 00:00:01
|
|*  2 |   TABLE ACCESS FULL| TRANTBL |    10 |   610 |     3   (0)| 00:00:01
|
--------------------------------------------------------------------------------
--

Predicate Information (IDENTIFIED BY operation id):

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
----
---------------------------------------------------

   2 - FILTER("DATE_YM"='201401')

Note
-----
   - dynamic sampling used FOR this statement

18 ROWS selected.
```

```
SQL>
```

□TABLE ACCESS FULL□が1回しかありません。また、項目DATE_YMと項目AREACODEをキーに集計を行っている様子もありません。
これは□Id 2 のテーブル読み込みを行いながら同時に必要な集計を行っているためです。集約集計のためのキーはPARTITION BY 句で判明しているので□Id 1でキー毎の集計を取っています。 その為1回のテーブル読み込みで必要な処理が終わっています。

先輩：「1回の読み込みで必要な集計処理も同時に済ませている。なので2回読み込む必要がない。
後輩：「言われてみれば、わざわざ別けて読み直ししなくても処理できますもんね。
先輩：「それに□□SELECT * FROM TRNTBL WHERE DATE_YM='201401'□が1時間かかる場合、サブクエリ版は2時間かかる可能性があるね。実際はキャッシュ等が使われてそこまで行かないと思うけど。
後輩：「更に別の判定をサブクエリを使って追加すると、対応するTABLE ACCESS FULLが発生するから、知っていないと大変な結果を招く場合があるって事ですね。

後輩：「でもウインドウ関数版はレビュー時の説明で苦労しそうだなぁ……
先輩：「ウインドウ関数は知らない人への説明が面倒だけど、ウインドウ関数を使うとデータベースが1回で読み込みを終えてくれるようになる、と言えばきっと分かってくれると思うよ。

印を付けるレコードが一意に決まらない場合の対処例

そして、やっぱり一度はやる失敗。

後輩：「項目SALESに重複が発生してしまいました！なので複数の印がついちゃいます！
先輩：「そらそうだ、合計値が一緒になる事を想定出来ないやつがおかしい。
後輩□□orz

今回の処理対象はこれになります。

```
SQL> SELECT * FROM TRANTBL WHERE DATE_YM='201404';

DATE_YM              AREACODE BRANCHCODE ITEMCODE         VOLUME       SALES
------------------ ---------- ---------- ------------ ---------- ----------
201404                     10      10100 F170                 30         300
201404                     10      10102 F170                 44         440
201404                     10      10102 F175                 20         440
201404                     10      10104 F175                 15         330
201404                     20      20201 F170                 47         470
201404                     20      20302 F175                 15         330
201404                     20      20302 F178                120         840
201404                     30      30100 F170                 90         900
201404                     30      30203 F173                 40         800

9 ROWS selected.

SQL>
```

これにPARTITION BY句を使ったクエリを適用すると、当然ながら
AREACODE=10□BRANCHCODE=10102□の２レコードに印がついてしまいます。

```
SQL> SELECT
  2    CASE WHEN SALES = MAX(SALES) OVER (PARTITION BY DATE_YM, AREACODE)
```

```
THEN 'Y' ELSE NULL END AS "TOPSALES"
  3   ,DATE_YM
  4   ,AREACODE
  5   ,BRANCHCODE
  6   ,ITEMCODE
  7   ,VOLUME
  8   ,SALES
  9  FROM TRANTBL
 10  WHERE DATE_YM='201404'
 11  ;

TOP DATE_YM              AREACODE BRANCHCODE ITEMCODE          VOLUME
SALES
--- ------------------ ---------- ---------- ------------ ---------- -------
---
    201404                     10      10100 F170                 30
300
Y   201404                     10      10102 F170                 44
440   ←印が重複
Y   201404                     10      10102 F175                 20
440   ←印が重複
    201404                     10      10104 F175                 15
330
    201404                     20      20201 F170                 47
470
Y   201404                     20      20302 F178                120
840
    201404                     20      20302 F175                 15
330
Y   201404                     30      30100 F170                 90
900
    201404                     30      30203 F173                 40
800

9 ROWS selected.

SQL>
```

先輩：「一意に決めるコードが無いならこうなるしかないよね。
後輩：「何とかインチキできませんかね？
先輩：「水曜どうでしょう再開したんだってねぇ。…ソートを行って、常にレコードの並びを一意に出来るなら手はある。

ウインドウ関数LAGを使った例

以下OVER 句以降で指定する

- PARTITION BY 句で指定する項目AREACODE項目BRANCHCODEで指定した範囲
- ORDER BY 句で指定する項目SALES(降順)、項目AREACODE項目BRANCHCODE項目ITEMCODEでソート順を指定

のレコードセットを適用対象範囲とした、ウインドウ関数LAGで判定し印を付ける例です。

```
SQL> SELECT
  2    CASE WHEN LAG(SALES,1) OVER (PARTITION BY DATE_YM, AREACODE ORDER BY
SALES DESC, AREACODE, BRANCHCODE,ITEMCODE) IS NULL THEN 'Y' ELSE NULL END AS
"TOPSALES"
  3   ,DATE_YM
  4   ,AREACODE
  5   ,BRANCHCODE
  6   ,ITEMCODE
  7   ,VOLUME
  8   ,SALES
  9  FROM TRANTBL
 10  WHERE DATE_YM='201404'
 11  ;

TOP DATE_YM              AREACODE BRANCHCODE ITEMCODE         VOLUME
SALES
--- ------------------ ---------- ---------- ------------ ---------- -------
---
Y   201404                     10      10102 F170                 44
440□□←LAG(SALES,1)の結果はNULL
    201404                     10      10102 F175                 20
440□□←LAG(SALES,1)の結果は440
    201404                     10      10104 F175                 15
330
    201404                     10      10100 F170                 30
300
Y   201404                     20      20302 F178                120
840□□←LAG(SALES,1)の結果はNULL
    201404                     20      20201 F170                 47
470□□←LAG(SALES,1)の結果は840
    201404                     20      20302 F175                 15
330
Y   201404                     30      30100 F170                 90
900□□←LAG(SALES,1)の結果はNULL
    201404                     30      30203 F173                 40
800□□←LAG(SALES,1)の結果は900

9 ROWS selected.

SQL>
```

ウインドウ関数LAGは、現在のレコードセットの直前のレコードにあった項目の値を取得します□LAG(SALES,1)は、1レコード前の項目SALESとなります。

上記の例で、項目DATE_YM='201404'□項目AREACODE=20□の範囲に注目すると、

- レコードセット2番目のレコードでLAG(SALES,1)の結果はレコードセット1番目のレコードの項目SALESの値である840になります。
- レコードセット1番目のレコードでLAG(SALES,1)の結果は、…ひとつ前のレコードがありません

から□NULLになります。

ORDER BY句で最初に項目SALESを降順で指定したのであれば、最大値を持つレコードが先頭に集まります。 そして、最大値の同じレコードが並んでいたとしてもウインドウ関数LAGを使う事で、1番目なのか、それ以外なのかを判定できるので、先頭にあるレコードだけに印を付けてしまえばいいのです。

後輩：「並びがちょっと変わっちゃいましたね。できれば最初の並びにしたいですねー
先輩：「完全には無理だけど、こんな感じでソートしなおせばいいと思う。

```
SQL> SELECT
  2    CASE WHEN LAG(SALES,1) OVER (PARTITION BY DATE_YM, AREACODE ORDER BY
SALES DESC, AREACODE, BRANCHCODE, ITEMCODE) IS NULL THEN 'Y' ELSE NULL END
AS "TOPSALES"
  3   ,DATE_YM
  4   ,AREACODE
  5   ,BRANCHCODE
  6   ,ITEMCODE
  7   ,VOLUME
  8   ,SALES
  9  FROM TRANTBL
 10  WHERE DATE_YM='201404'
 11  ORDER BY AREACODE, BRANCHCODE□□-- ←ORDER BY 句を追加した
 12  ;

TOP DATE_YM              AREACODE BRANCHCODE ITEMCODE         VOLUME
SALES
--- ------------------ ---------- ---------- ------------ ---------- -------
---
    201404                     10      10100 F170                 30
300
Y   201404                     10      10102 F170                 44
440   ←並びが入れ替わる可能性あり↓
    201404                     10      10102 F175                 20
440   ←並びが入れ替わる可能性あり↑
    201404                     10      10104 F175                 15
330
    201404                     20      20201 F170                 47
470
Y   201404                     20      20302 F178                120
840   ←並びが入れ替わる可能性あり↓
    201404                     20      20302 F175                 15
330   ←並びが入れ替わる可能性あり↑
Y   201404                     30      30100 F170                 90
900
    201404                     30      30203 F173                 40
800

9 ROWS selected.

SQL>
```

元のクエリでもORDER BY 句がないので、この出力結果は環境依存となります。

たまたま並びがそうなった、と理解してください。“完全には無理”というのはそういうことです。

ORDER BY 句指定時の注意点

後輩：「ウインドウ関数LAGを使うときのORDER BY 句なんですが、項目SALESだけでも良いんじゃないでしょうか。
先輩：「駄目。それだと実行の度にLAG(SALES,1)がNULLになるレコードが変わってしまう。
後輩□□□□□□
先輩：「項目SALESだけの指定だと、ソートの結果が不定になってしまうから。なので□ORDER BY 句の指定でソート結果を一意にしている。

SELECT文実行時そのレコードセットの出力順序は不定です。
ORDER BY 句を指定して出力順序を決めるのですが、指定が足りなかったりすると、その足りない部分での出力順が不定となります。これは環境に依存する部分です。

例えば、以下のクエリ

```
SELECT * FROM TRANTBL
  WHERE DATE_YM='201404' AND AREACODE=10
  ORDER BY SALES DESC, AREACODE, BRANCHCODE
```

だと、次のような2通りの結果が発生する可能性があります。

```
DATE_YM              AREACODE BRANCHCODE ITEMCODE         VOLUME      SALES
------------------ ---------- ---------- ------------ ---------- ----------
201404                     10      10102 F170                 44        440
←ITEMCODEに注目
201404                     10      10102 F175                 20        440
←ITEMCODEに注目
201404                     10      10104 F175                 15        330
201404                     10      10100 F170                 30        300


DATE_YM              AREACODE BRANCHCODE ITEMCODE         VOLUME      SALES
------------------ ---------- ---------- ------------ ---------- ----------
201404                     10      10102 F175                 20        440
←ITEMCODEに注目
201404                     10      10102 F170                 44        440
←ITEMCODEに注目
201404                     10      10104 F175                 15        330
201404                     10      10100 F170                 30        300
```

OVER 句以降の記述に現れるORDER BY 句も同様で、ソート順を決める項目の指定に不足があればソート結果が異なる可能性がありますので、以下の様に印が付くレコードが一意に決まらず不安定に見える場合があります。

```
TOP DATE_YM              AREACODE BRANCHCODE ITEMCODE         VOLUME
SALES
--- ------------------ ---------- ---------- ------------ ---------- -------
---
Y   201404                   10      10102 F170                 44
```

```
440    ← このレコードに印が付く事があったり
    201404                      10      10102 F175                  20
440
    201404                      10      10104 F175                  15
330
    201404                      10      10100 F170                  30
300
Y   201404                      20      20302 F178                 120
840
    201404                      20      20201 F170                  47
470
    201404                      20      20302 F175                  15
330
Y   201404                      30      30100 F170                  90
900
    201404                      30      30203 F173                  40
800

TOP DATE_YM              AREACODE BRANCHCODE ITEMCODE          VOLUME
SALES
--- ------------------ ---------- ---------- ------------ ---------- -------
---
Y   201404                      10      10102 F175                  20
440    ← このレコードに印が付く事があったり
    201404                      10      10102 F170                  44
440
    201404                      10      10104 F175                  15
330
    201404                      10      10100 F170                  30
300
Y   201404                      20      20302 F178                 120
840
    201404                      20      20201 F170                  47
470
    201404                      20      20302 F175                  15
330
Y   201404                      30      30100 F170                  90
900
    201404                      30      30203 F173                  40
800
```

上記の不定動作を排除するため□ORDER BY 句には以下のクエリの様に、

```
SELECT * FROM TRANTBL
  WHERE DATE_YM='201404' AND AREACODE=10
  ORDER BY SALES DESC, AREACODE, BRANCHCODE, ITEMCODE
```

必要な項目を記述して明示的に結果を一意にさせます。

```
DATE_YM              AREACODE BRANCHCODE ITEMCODE          VOLUME       SALES
------------------ ---------- ---------- ------------ ---------- ----------
```

```
201404                          10       10102 F170                    44         440
←項目ITEMCODEによりこの位置で確定
201404                          10       10102 F175                    20         440
←項目ITEMCODEによりこの位置で確定
201404                          10       10104 F175                    15         330
201404                          10       10100 F170                    30         300
```

先輩：「楽そうだからターゲットを先頭のレコードとしてみただけなので、好きな手段で決めればいいと思うよ。
後輩：「考えるの面倒なので、一意に決めるキーを付ける様にお願いしてみます。
先輩：「じゃあ最初からそうして。今までの手間はなんだったのよ……

印が付くレコードだけ欲しい場合の対処例

後輩：「上司が、印が付くレコードだけ欲しいといってきたんですけど。
先輩：「まぁそうだよね。トップだけ見たいって言われるのも想像できるよね。
後輩：「WHERE句にウインドウ関数使えないみたいなんで、該当レコードだけってのがやれないんですけど…
先輩：「後輩ってさ、本当に応用力が無い子よねぇ……

以下のような結果、

```
TOP DATE_YM              AREACODE BRANCHCODE ITEMCODE          VOLUME
SALES
--- ------------------ ---------- ---------- ------------ ---------- -------
---
    201404                      10      10100 F170                 30
300
Y   201404                      10      10102 F170                 44
440
    201404                      10      10102 F175                 20
440
    201404                      10      10104 F175                 15
330
    201404                      20      20201 F170                 47
470
Y   201404                      20      20302 F178                120
840
    201404                      20      20302 F175                 15
330
Y   201404                      30      30100 F170                 90
900
    201404                      30      30203 F173                 40
800
```

を、必要なレコードだけしにして

```
DATE_YM              AREACODE BRANCHCODE ITEMCODE          VOLUME      SALES
------------------ ---------- ---------- ------------ ---------- ----------
201404                      10      10102 F170                 44        440
201404                      20      20302 F178                120        840
```

```
201404                    30      30100 F170                    90         900
```

としたい訳ですね。

難しく考えなくてもサブクエリで解決すればいいんです。既にどのレコードが対象か分かる印をつける事ができているんですから。

```
SQL>SELECT S.DATE_YM, S.AREACODE, S.BRANCHCODE, S.ITEMCODE, S.VOLUME,
S.SALES
  2 FROM (
  3 SELECT
  4     CASE WHEN LAG(SALES,1) OVER (PARTITION BY DATE_YM, AREACODE ORDER BY
SALES DESC, AREACODE, BRANCHCODE, ITEMCODE) IS NULL THEN 'Y' ELSE NULL END
AS "TOPSALES"
  5    ,DATE_YM
  6    ,AREACODE
  7    ,BRANCHCODE
  8    ,ITEMCODE
  9    ,VOLUME
 10    ,SALES
 11   FROM TRANTBL
 12   WHERE DATE_YM='201404'
 13   ORDER BY AREACODE, BRANCHCODE
 14 ) S
 15 WHERE S.TOPSALES='Y'
 16 ;

DATE_YM              AREACODE BRANCHCODE ITEMCODE         VOLUME      SALES
------------------ ---------- ---------- ---------- ---------- ----------
201404                     10      10102 F170                44        440
201404                     20      20302 F178               120        840
201404                     30      30100 F170                90        900

3 ROWS selected.

SQL>
```

先輩：「気になるならSQLの実行計画を調べてみればいいよ。
後輩：「なんかだまされた気分。サブクエリ使うなって言われたその直後に使ってるし……
先輩：「確定した情報を利用している、実に分かり易いサンプルだと思うけどなぁ。

SQL, Oracle, database, 技術資料