【グループを極める（その１）】

今回の弱点

前回の投稿（**テーブルとビューを極める（その２）**）で、**「ビューの更新等は、元の実表の更新等ができることが条件のため、ビューが「GROUP BY句」とその仲間たちである「HAVING句」、「DISTINCT句」、計算例及び集約関数等を使用している場合は、ビューはグループ化されたビューになってしまうため、もはや実表の行を特定することはできなくなり、実表の更新等はできなくなります。」**と記載しましたが、GROUP BY等の理解が曖昧のため、関連する事項を含めて、まとめていきます。

GROUP BY句を使用したSELECT文の基本（その１）

下表（受注明細）に対して、次ページのSELECT文を実行した場合について、説明します。

表:受注明細

受注番号	商品コード	数量
001	200	3
001	300	2
001	400	6
007	200	4
007	100	2
007	300	8
007	500	10
011	400	12
011	300	5
012	100	7
012	400	9
012	500	10

SELECT 受注番号　FROM 受注明細　GROUP BY 受注番号　HAVING　COUNT（＊）＞＝ 3
③　①　②

SELECT文における①の説明

FROM句で指定した左の下表（受注明細）において、GROUP BY句で指定した列（受注番号）をキーとして、キーの値が同じ行が１つのグループになる。つまり、以下に示すとおり、００１、００７、０１１及び０１２の４つのグループができる。

SELECT文における②の説明

HAVING句で指定した条件は、各グループに対する条件であり、今回は「COUNT(＊) >= 3」なので、行数が３行以上あるグループが条件を満たすことになる。つまり、以下に示すとおり、００１、００７及び０１２の３つのグループが該当する。

SELECT文における③の説明

上記①と②の条件を満たす行をSELECT句で指定した列（受注番号）で抽出したものが右の下表（実行結果その１）であり、列（受注番号）の重複行はまとまり、3つにグループ化されたことがわかる。逆に、そうならないと、グループ化した意味がないため、当たり前の結果である。

表:受注明細

受注番号	商品コード	数量
001	200	3
001	300	2
001	400	6
007	200	4
007	100	2
007	300	8
007	500	10
011	400	12
011	300	5
012	100	7
012	400	9
012	500	10

００１グループ：行数３

００７グループ：行数４

０１１グループ：行数２

０１２グループ：行数３

表:実行結果その１

受注番号
001
007
012

BROUP BY句を使用したSELECT文の基本（その２）

前ページのSELECT文に対して、以下のとおり、SELECT句に列（**SUM（数量）AS 数量合計**）を追加して、実行した場合について、説明します。前ページの説明のとおり、FROM句以下で対象グループは、左の下表（実行途中その２）のとおり、００１、００７及び０１２の３つのグループに絞られます。そして、SELECT句で指定した集約関数「**SUM（数量）AS 数量合計**」は、絞られた３つのグループに対して、グループ毎の数量を合計して、右の下表（実行結果その２）のとおり、新たに「合計数量」という名前で表示します。列（受注番号）と列（数量合計）は1対1なので、前ページ同様、3行にまとまります。これについても、そもそも、グループ毎の数量の合計を得るためにグループ化しているため、当たり前の結果です。

```
SELECT 受注番号、SUM（数量）AS 数量合計　FROM 受注明細　GROUP BY 受注番号
    HAVING　COUNT（＊）＞＝ 3
```

表:実行途中その２

受注番号	商品コード	数量
001	200	3
001	300	2
001	400	6
007	200	4
007	100	2
007	300	8
007	500	10
012	100	7
012	400	9
012	500	10

００１グループ
→　数量合計１１

００７グループ
→　数量合計２４

０１２グループ
→　数量合計２６

表:実行結果その２

受注番号	数量合計
001	11
007	24
012	26

グループ化において書き忘れていたこと

ここまでで書き残していた事項は以下のとおりです。

①	GROUP BY句によってグループされた後は、GROUP BY句の後に続くHAVING句やORDER句の条件は、各グループに対するものになる。例外として、GROUP BY句がない場合のHAVING句は全体を１つのグループとして扱う。
②	GROUP BY句の役割は、グループ分けはもちろんのこと、集約関数を用いて、各グループの集計を行うことである。ただし、表の行数を確認する場合の「SELECT COUNT（＊）FROM 表」等のようにGROUP BY句を使用しない場合も当然ある。
③	SELECT句に指定できる列は、GROUP BY句で指定した列の他に集約関数及び定数である。つまり、GROUP BY句で指定していない列をSELECT句で指定できない。そもそも、GROUP BY句で指定していない列は、グループ化が不要だからこそ、指定していないはずなので、通常は、SELECT句にも指定する必要がないと考える。ここで、集約関数で使用している列はGROUP BY句で指定する必要はないが、集約関数以外の関数における引数が列の場合には、指定する必要がある。

GROUP BY句とWHERE句が混在する場合

```
SELECT 商品ランク, AVG（売上合計金額）…❹
    FROM 商品 LEFT OUTER JOIN 商品別売上実績 ON 商品.商品コード = 商品別売上実績.商品コード …❶
        WHERE 商品ランク = 'A' OR 商品ランク = 'B' …❷
            GROUP BY 商品ランク …❸
```

順番 ↓

まず、❶の直積及び左外部結合の結果に対して、❷により商品ランクA又はBの行だけが抽出される。この時点で、他の商品ランクは今後考える必要はない。そして、❸と❹により、商品ランクごとの売上合計金額の平均を表示することを決めている。

以上のことから、商品ランクA及びB毎の売上合計金額の平均が表示される。

更にHAVING句が加わった場合

上記のSELECT文に対して、❺を加え、❷のWHERE句を変更したものが、以下のSELECT文である。まず、❶と❷より、商品ランクに関係なく、売上合計金額が2000以上である行だけが抽出される。この結果、商品ランクAが3行、Bが2行、Cが1行抽出されている場合、❺により、商品ランクAとBの売上合計金額の平均が表示される。

```
SELECT 商品ランク, AVG（売上合計金額）…❹
    FROM 商品 LEFT OUTER JOIN 商品別売上実績 ON 商品.商品コード = 商品別売上実績.商品コード …❶
        WHERE 売上合計金額 >= 2000 …❷
            GROUP BY 商品ランク …❸
                HAVING COUNT(＊) >= 2 …❺
```

順番 ↓