東京ITスクール
TOKYO IT SCHOOL

サブクエリ

目次

1. はじめに

サブクエリは、ビューを基本とした技術です。サブクエリの特徴を一言で表わすと、「使い捨てのビュー」です。

ビューとは、データそのものを保存するのではなく、データを取り出す SELECT 文だけを保存するという方法で、ユーザーの利便性を高める機能でした。それに対してサブクエリは、ビュー定義の SELECT 文をそのまま FROM 句に持ち込んだものです。

2. サブクエリ

1　サブクエリ

サブクエリの構文を以下に示します。

(構文) サブクエリ

```
SELECT <列名>, ……
   FROM (SELECT 文)
[WHERE JOIN 等];
```

それでは、ビュー作成時の例で使用した SQL 文を基にサブクエリを作成してみましょう。

(サンプルコード) サブクエリ

```
SELECT shop_id, shop_name, item_id, item_name, sel_price, quantity, stock
FROM (SELECT SI.shop_id, SI.shop_name, SI.item_id, I.item_name,
I.sel_price, SI.quantity, StI.stock
FROM ShopItem SI INNER JOIN Item I ON SI.item_id = I.item_id
INNER JOIN StockItem StI ON SI.item_id = StI.item_id
WHERE StI.house_id = 'S001');
```

(実行結果)

SHOP_ID	SHOP_NAME	ITEM_ID	ITEM_NAME	SEL_PRICE	QUANTITY	STOCK
000D	福岡	0001	シャツ	1000	100	0
000A	東京	0001	シャツ	1000	30	0
000B	仙台	0002	ホッチキス	500	30	120
000A	東京	0002	ホッチキス	500	50	120
000C	大阪	0003	セーター	4000	20	200
000B	仙台	0003	セーター	4000	120	200
000A	東京	0003	セーター	4000	15	200
000C	大阪	0004	包丁	3000	50	3
000B	仙台	0004	包丁	3000	20	3
000C	大阪	0006	フォーク		90	99
000B	仙台	0006	フォーク		10	99
000C	大阪	0007	スプーン	790	70	999
000B	仙台	0007	スプーン	790	40	999

13 行選択されました

得られる結果はビューと同様です。このように、ビュー定義の SELECT 文を、そのまま FROM 句の中に入れてしまったのがサブクエリです。サブクエリ（subquery）とは「下位の（sub）」の「問い合わせ（query）」という意味です。

実際、この SELECT 文は、入れ子構造になっていて、まず FROM 句の中の SELECT 文が実行され、その後に外側の SELECT 文が実行される、という順番になります。

また、サブクエリの階層数には原則的に制限はないので、サブクエリの中の FROM 句に更にサブクエリを使って、その中の FROM 句に更にサブクエリを……というように、いくらでも入れ子を深くすることが可能です。

2　スカラ・サブクエリ

続いて、サブクエリの一種である「スカラ・サブクエリ（scalar subquery）」という概念を学びましょう。「スカラ」とは、「単一の」という意味の言葉で、データベース以外の分野でも使用されます。先ほど学んだサブクエリは、基本的に複数行を結果として返します。構造的にはテーブルと同じですので当然ではあります。

これに対してスカラ・サブクエリは、「必ず 1 行 1 列だけの戻り値を返す」という制限をつけたサブクエリのことです。スカラ・サブクエリの戻り値が単一の値ということは、この戻り値を、=、<>など、スカラ値を入力する比較演算子の入力として利用することが出来るます。

また、スカラ・サブクエリを書ける場所は、WHERE 句などでの比較演算子の入力部分だけに限りません。基本的に、スカラ値が書けるところにはどこにでも書けます。ということは、定数や列名を書くことの出来る場所全てとなり、SELECT 句でも GROUP BY 句でも HAVING 句でも ORDER BY 句でも、ほとんどあらゆる場所に書くことが可能です。

前項で学んだサブクエリは実際にはあまり利用価値はありませんが、スカラ・サブクエリは非常に利用価値があります。

例えば、これまでに例として使用してきた商品（Item）テーブルに対して、「販売単価が、全体の平均の販売単価より高い商品だけを検索する」場合、じつは一筋縄ではいきません。AVG関数を使って次のようなSQLを書いたとしても、エラーになってしまうのです。

(サンプルコード) WHERE句にAVG関数

```
--WHERE 句に集約関数は使えない
SELECT item_id, item_name, pur_price
FROM Item
WHERE sel_price > AVG(sel_price);
```

SELECT文の意味としてはこれであっていそうに見えますが、集約関数をWHERE句に書くことは出来ないという制限のため、このSELECT文は誤りとなります。

このような場面でスカラ・サブクエリが活きてきます。まず、Itemテーブルに含まれている商品（Item）の平均の販売単価（sel_price）を求めるには、次のSELECT文で可能です。

(サンプルコード) 平均の販売単価

```
SELECT AVG(sel_price) FROM Item;
```

(実行結果)

```
AVG(SEL_PRICE)
----------------------
2381.66667
```

このSQL文をスカラ・サブクエリとして、先程のエラーとなったSQL文のWHERE句にしてみましょう。

(サンプルコード) WHERE句にスカラ・サブクエリ

```
--WHERE 句に集約関数は使えない
SELECT item_id, item_name, pur_price
FROM Item
WHERE sel_price > (SELECT AVG(sel_price) FROM Item);
```

(実行結果)

```
ITEM_ID   ITEM_NAME            PUR_PRICE
----------- ---------------------- -----------------
0003      セーター               2800
0004      包丁                   2800
0005      フライパン             2800
```

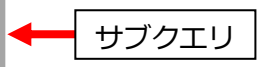
3　スカラ・サブクエリを使うときの注意点

スカラ・サブクエリを使うときに最も注意しなければならないことは、「絶対にサブクエリが複数行を返さないようにする」ことです。というのも、サブクエリが複数行を返す時点で既にそれはスカラ・サブクエリではなく、ただのサブクエリになってしまいます。すると、=、<>といったスカラ値を入力する演算子も利用出来ませんし、SELECT 句などに書くことも出来なくなります。

次の SELECT 文はエラーになります。

(サンプルコード)通常のサブクエリを SELECT 句にて使用

```
--スカラ・サブクエリではないので SELECT 句に書けない
SELECT item_id,
       item_name,
       sel_price,
       (SELECT AVG(sel_price)
       FROM Item
       GROUP BY category_id) avg_price
FROM Item;
```



エラーの理由は簡単で、このサブクエリは、複数行を返すからです。そのため、上記の SELECT 文は、「サブクエリが複数行を返すため実行出来ない」という理由のエラーが返されることになります。

3. 相関サブクエリ

1　相関サブクエリの利用

前節で学習したように、「販売単価（sel_price）が、全体の平均の販売単価より高い商品」を選び出すには、サブクエリを使えば実現出来ます。今度は少しこの条件を変えて「商品分類（category_id）ごとに平均販売単価より高い商品」を、商品分類のグループから選び出すことを考えてみましょう。

グループ分けの際は、GROUP BY 句を使用します。しかし、単純に以下のように GROUP BY 句を使用するとエラーとなってしまいます。

(サンプルコード) サブクエリにて GROUP BY 句を使用

```
--エラーになるサブクエリ
SELECT item_id, item_name, sel_price
FROM Item
WHERE sel_price > (SELECT AVG(sel_price) FROM Item GROUP BY category_id);
```

エラーになる理由は、前節でも述べたとおり、このサブクエリが複数行を返してしまい、スカラ・サブクエリにならないからです。WHERE 句でサブクエリを使用する場合は、必ず結果は 1 行である必要があります。

このような場合に使用するのが相関サブクエリです。相関サブクエリは、テーブル全体ではなく、テーブルの一部のレコード集合に限定した比較をしたい場合に使います。

まずは、以下の SELECT 文を見てください。

(サンプルコード) 相関サブクエリ

```
SELECT item_id, item_name, sel_price
FROM Item I1
WHERE sel_price > (SELECT AVG(sel_price)
                   FROM Item I2
                   WHERE I1.category_id = I2.category_id
                   GROUP BY category_id);
```

(実行結果)

```
ITEM_ID  ITEM_NAME            SEL_PRICE
-------- -------------------- ----------
0003     セーター             4000
0004     包丁                 3000
0005     フライパン           5000
```

WHERE I1.category_id = I2.category_id によって、商品分類を 1 種類に絞って（つまりその商品自身の商品分類）平均販売単価を計算しているので、サブクエリはスカラ・サブクエリとなります。これを「縛る」と呼んだりもします。

2　相関サブクエリの注意点

ここで、SQL の初心者が相関サブクエリを使うときに、よくやってしまう間違いを 1 つ紹介しておきましょう。それは、「縛る」ための結合条件をサブクエリの内側ではなく、外側に書いてしまうというものです。具体的には、次の SELECT 文を見てください。

(サンプルコード) 結合条件を外側に記述した相関サブクエリ

```
SELECT item_id, item_name, sel_price
FROM Item I1
WHERE I1.category_id = I2.category_id
AND sel_price > (SELECT AVG(sel_price)
                 FROM Item I2
                 GROUP BY category_id);
```

これは、サブクエリの中にあった条件を、外側に移動させただけで、その他は何の変更も加えていません。ところが、この SELECT 文はエラーになって正しく実行出来ないのです。

エラーの原因は相関名のスコープです。相関名というのは、I1 や I2 など、テーブルの別名としてつけた名前です。サブクエリ内部でつけられた相関名は、そのサブクエリ内でしか使用出来ません。I1 は外側でつけられた名前なので SELECT 文全体で使用出来ますが、I2 はサブクエリ内でつけられた名前なので、サブクエリの括弧内でしか使用は出来ません。