はじめに

■コース概要と目的

SQL での作業の幅を広げるための応用的なテクニックをご説明します。また、効率性の向上や正しい結果を得るための記述方法など、実践的な記述方法についても併せてご説明します。

※本コースは、SQL の応用的な記述テクニックとしてどのようなものがあるかを 1 日で広く浅くご理解いただくことを目的としたコースです。細かな構文やオプションの習得は目的としておりませんことをご了承ください。

■受講対象者

アプリケーション開発者。

■前提条件

弊社 Oracle 研修「SQL トレーニング」コースを受講された方、もしくは同等の知識をお持ちで、ある程度の現場経験のある方を対象としています。

■テキスト内の記述について

▼構文

[　]	省略可能
{ A | B }	A または B のどちらかを選択
n	数値の指定
_	デフォルト値

▼マーク

12.1	指定バージョンからの新機能 （左記の場合、Oracle 12cR1 からの新機能）
NOTE	知っておいたほうが良いテクニック、もしくは注意事項
	参照ページ

第１章

条件分岐

この章では、条件に応じて処理を分岐する方法について説明します。

１．CASE 式で複雑な条件分岐を実現
２．関数を使用した条件分岐
３．MERGE 文による条件に応じた DML の実行

1．CASE式で複雑な条件分岐を実現

CASE式を使用すると条件に合わせて処理を分岐できます。条件分岐処理はDECODEなどの関数でも可能ですが、関数に比べてより複雑な分岐条件を設定できるなどの特徴があります。

（1）CASE式の特徴

CASE式には次のような特徴があります。

1）コードの読みやすさ

一般的にDECODEなどの分岐関数を使用した場合、分岐の数が増えると処理内容が読みづらくなってしまいます。しかし、CASE式では分岐条件が増えても、わかりやすいコード表現を維持できます。
以下網掛けの部分は、DEPTNO列の値に応じて異なる値を戻す処理を、DECODE関数、CASE式でそれぞれ実施した例です。

■DECODE関数を使用した場合

```
SELECT ename,
       DECODE(deptno,10,'ACCOUNTING',
                      20,'RESEARCH',
                      30,'SALES',
                         'OPERATIONS')
FROM   emp;
```

■CASE式を使用した場合

```
SELECT ename,
       CASE deptno WHEN 10 THEN 'ACCOUNTING'
                   WHEN 20 THEN 'RESEARCH'
                   WHEN 30 THEN 'SALES'
                   ELSE 'OPERATIONS'
       END
FROM   emp;
```

上記例のように、CASE式はWHEN句、THEN句などのキーワードを指定するため、分岐条件やそれに対応する処理内容が明確であり、読みやすいコードになっていることがわかります。

２）複雑な分岐条件を設定できる

CASE 式では、分岐条件に演算子や副問合せを含めることができます。そのため、分岐関数に比べて複雑な分岐条件を設定できます。
以下は分岐条件に BETWEEN AND 演算子を使用し、範囲による分岐処理を行っている例です。

■より複雑な分岐条件を設定

```
SELECT ename,
       CASE WHEN sal BETWEEN    1 AND 2000 THEN '少ない'
            WHEN sal BETWEEN 2001 AND 4000 THEN '普通'
            WHEN sal BETWEEN 4001 AND 6000 THEN '多い'
            ELSE '不明'
       END
FROM emp;
```

DECODE 関数の場合、上記のような範囲検索や非等価演算子を使用するなどの複雑な分岐条件は設定できません。

３）様々な箇所で使用できる

CASE「式」であるため、他の句と組合わせ、SQL の様々な箇所で条件分岐を行うことができます。

■SUM 関数と CASE 式を組合わせ、HAVING 句に指定

```
SELECT job FROM health_chk
GROUP  BY job
HAVING COUNT(*) = SUM(CASE WHEN chk_date < '11-01-31' THEN 1
                           ELSE 0
                      END);
```

（2）CASE式の記述方法

CASE式の基本構文と記述時の注意点を解説します。

1）基本構文

CASE式の構文には、単純CASE式と検索CASE式の2つがあります。

※単純CASE式の分岐条件では等価評価しか行えませんが、検索CASE式は分岐条件で等価演算子以外の演算子も使用できます。また、単純CASE式は検索CASE式に書き換えできます。そのため、本コースでは検索CASE式を中心に解説します。

■単純CASE式

単純CASE式の分岐条件では、選択子式と条件値を等価評価します。

CASE句に選択子式として比較対象を指定し、WHEN句で等価評価する条件値をそれぞれ指定します。THEN句で各条件値に応じた処理内容を指定します。また、ELSE句でその他に対する処理内容を記述します。

```
CASE 選択子式
     WHEN 条件値 THEN 処理内容
     [ WHEN 条件値 THEN 処理内容 ]
     [ ELSE 処理内容 ]
END
```

■検索CASE式

検索CASE式では、WHEN句で分岐条件を1つ1つ指定します。この分岐条件はWHERE句の条件と同じように指定できるため、等価演算子以外の演算子も使用できます。その他の指定項目は単純CASE式と同様です。

```
CASE WHEN 条件 THEN 処理内容
     [ WHEN 条件 THEN 処理内容 ]
     [ ELSE 処理内容 ]
END
```

■単純 CASE 式と検索 CASE 式の比較

```
/* 単純 CASE 式 */
SELECT ename,
       CASE deptno WHEN 10 THEN 'ACCOUNTING'
                   WHEN 20 THEN 'RESEARCH'
                   WHEN 30 THEN 'SALES'
                   ELSE 'OPERATIONS'
       END
FROM   emp;

/* 検索 CASE 式 */
SELECT ename,
       CASE WHEN deptno = 10 THEN 'ACCOUNTING'
            WHEN deptno = 20 THEN 'RESEARCH'
            WHEN deptno = 30 THEN 'SALES'
            ELSE 'OPERATIONS'
       END
FROM   emp;
```

CASE 句の後に比較対象である DEPTNO 列を指定。
WHEN 句で DEPTNO 列が 10、 20、 30、その他における分岐処理をそれぞれ指定。

WHEN 句で分岐条件を 1 つずつ指定。

上記例のように、分岐条件で等価評価を行う場合は、単純 CASE 式、検索 CASE 式のどちらでも記述できます。

■検索 CASE 式で分岐条件に等価演算子以外の演算子を使用

```
SELECT ename,
       CASE WHEN deptno <= 10 THEN 'ACCOUNTING'
            WHEN deptno <= 20 THEN 'RESEARCH'
            WHEN deptno <= 30 THEN 'SALES'
            ELSE 'OPERATIONS'
       END
FROM   emp;
```

分岐条件が等価評価以外の場合は検索 CASE 式を使用する。

上記例のように、等価演算子以外で分岐条件を定義する場合は検索 CASE 式を使用します。

2）注意事項

・最後の END 句の書き忘れに注意

```
/* END 句を書き忘れた場合 */
SQL> SELECT CASE WHEN job = 'MANAGER'  THEN sal*1.1
  2              WHEN job = 'SALESMAN' THEN sal*1.2
  3              ELSE NULL
  4  FROM   emp;
FROM   emp
*
行 4 でエラーが発生しました。:
ORA-00905: キーワードがありません。
```

・ELSE 句の記述を推奨

ELSE 句の記述は必須ではなく、省略した場合は「ELSE NULL（NULL を戻す）」になります。
しかし、意図しない結果の表示を防止したり、コードをわかりやすくしたりするために、ELSE 句を記述しておくことをお薦めします。

■ELSE 句なし

```
SELECT ename,
       CASE WHEN job = 'MANAGER'  THEN sal*1.1
            WHEN job = 'SALESMAN' THEN sal*1.2
       END
FROM   emp;
```

JOB が MANAGER と SALESMAN に対する条件はあるが、それ以外に対する明確な指定がない。

■ELSE 句あり

```
SELECT ename,
       CASE WHEN job = 'MANAGER'  THEN sal*1.1
            WHEN job = 'SALESMAN' THEN sal*1.2
            ELSE NULL
       END
FROM   emp;
```

ELSE 句により、 JOB が MANAGER と SALESMAN 以外に対して何もしなくて良いことが明確に表現されている。

・THEN句とELSE句で戻される値の型は、全てにおいて一致している必要あり

```
SQL> SELECT ename,
  2         CASE WHEN deptno = 10
  3                   THEN 'ACCOUNTING'
  4              WHEN deptno = 20
  5                   THEN 1
  6              ELSE NULL
  7         END
  8  FROM   emp;
                   THEN 1
                        *
行5でエラーが発生しました。:
ORA-00932: データ型が一致しません: CHARが予想されましたがNUMBERです。
```

1つ目のTHEN句で文字型、2つ目のTHEN句で数字型が戻されるためエラー発生。

・WHEN句の記述順に注意

分岐条件は記述した順番に評価され、該当の条件が見つかった時点で評価は打ち切られます。そのため、上の行に記述されている条件が下の条件を含んでいる場合、意図した結果が返ってきません。このことを意識してWHEN句を記述します。

```
SQL> SELECT sal,
  2         CASE WHEN sal >= 1000 THEN '少ない'
  3              WHEN sal >= 3000 THEN '普通'
  4              ELSE '不明'
  5         END AS amount
  6  FROM   emp
  7  ORDER BY sal DESC;

       SAL AMOUNT
---------- ----------
      5000 少ない
      3000 少ない
      3000 少ない
      2975 少ない
       :    :
```

SAL列が3000以上の場合、「普通」と表示したいが、1つ目のWHEN句の条件「sal >= 1000」に合致したため、2つ目のWHEN句が評価されていない。

※上記例のように、CASE式での評価結果に対して別名を定義することをお勧めします（ENDの後に「AS 別名」と指定）。定義しないと、CASE式の指定内容がそのまま列ヘッダになってしまいます。

（3）CASE式の効果的な使用方法

CASE式を使用すると、複数のSQLを1文で実施したり、複雑な条件分岐が行えます。
本テキストでは、以下3つの使用例をご紹介しています。

1）関数では行えない複雑な条件分岐を行う

CASE式のWHEN句には、比較演算子（<、>、BETWEEN、LIKE、IN、EXISTSなど）を使用できます。また、副問合せを含めることもできます。そのため、DECODE関数などに比べて複雑な分岐条件を指定できます。

例）EMP表の部門番号30の社員の給与と、社員WARDの給与を比較し、その結果をAMOUNT列として表示する。表示内容は、WARDの給与より少なければ「少ない」、同じであれば「同じ」、多ければ「多い」とする。

■EMP表の部門番号30のデータ

```
SQL> SELECT deptno,ename,sal
  2  FROM   emp
  3  WHERE  deptno = 30;

    DEPTNO ENAME             SAL
---------- ---------- ----------
        30 ALLEN            1600
        30 WARD             1250
        30 MARTIN           1250
        30 BLAKE            2850
        30 TURNER           1500
        30 JAMES             950
```

■表示したい結果

```
 DEPTNO ENAME             SAL AMOUNT
------- ---------- ---------- ----------
     30 ALLEN            1600 多い
     30 MARTIN           1250 同じ
     30 BLAKE            2850 多い
     30 TURNER           1500 多い
     30 JAMES             950 少ない
```

WARDの給与1250と、他の社員の給与をそれぞれ比較。比較した結果を「少ない」「同じ」「多い」で表示する。

■実行するSQL

```
SELECT deptno,ename,sal,
       CASE WHEN sal < (SELECT sal FROM emp                    ----------①
                        WHERE ename = 'WARD')
                 THEN '少ない'    ----------------------------------
            WHEN sal > (SELECT sal FROM emp                    |
                        WHERE ename = 'WARD')                  |
                 THEN '多い'      ----------------------------------②
            WHEN sal = (SELECT sal FROM emp                    |
                        WHERE ename = 'WARD')                  |
                 THEN '同じ'      ----------------------------------
            ELSE 'その他'    ---------------------------------------------③
       END AS amount
FROM   emp
WHERE  deptno = 30
AND    ename != 'WARD';
```

SELECT句の選択リストの4つ目にてCASE式による表示内容の分岐を行います。

WHEN句の分岐条件で、各社員の給与とWARDの給与を比較する条件を指定します。この「WARDの給与」は1回では求められないため、副問合せを使用します（①）。また、分岐処理の結果に応じて戻す値を THEN句に指定します（②）。

また、ELSE句でその他の場合についての対応を指定します（③）。最後にEND句を記述してCASE式の指定は終了です。