はじめに

■コース概要

条件分岐の方法や複雑な集計の手法など、SQL のコーディングの幅を広げるためのテクニックについて説明します。また、パフォーマンスを考慮した記述方法や正しい結果を取得するための記述方法などについても、あわせて説明します。

※本コースでは、実践的な SQL の記述手法を広く浅く紹介することを目的としているため、細かな構文やオプションの習得を目的とはしていないことを、あらかじめご了承ください。

■コースのゴール

・CASE 式や関数、MERGE 文を使用して条件分岐ができる。
・GROUP BY 句と CASE 式を組み合わせて、複雑な集計処理ができる。
・分析関数を使用して、グループを構成する 1 つ 1 つの要素に対して個別に集計結果を表示できる。
・相関副問い合わせを使用して、主問い合わせと連動した複雑な比較を行える。
・EXISTS 条件を使用して該当データの存在確認を行える。
・NULL の特性を考慮した SQL を記述できる。

■受講対象者

SQL を使用してアプリケーション開発をされる方。

■前提条件

「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の方。

■テキスト内の記述について

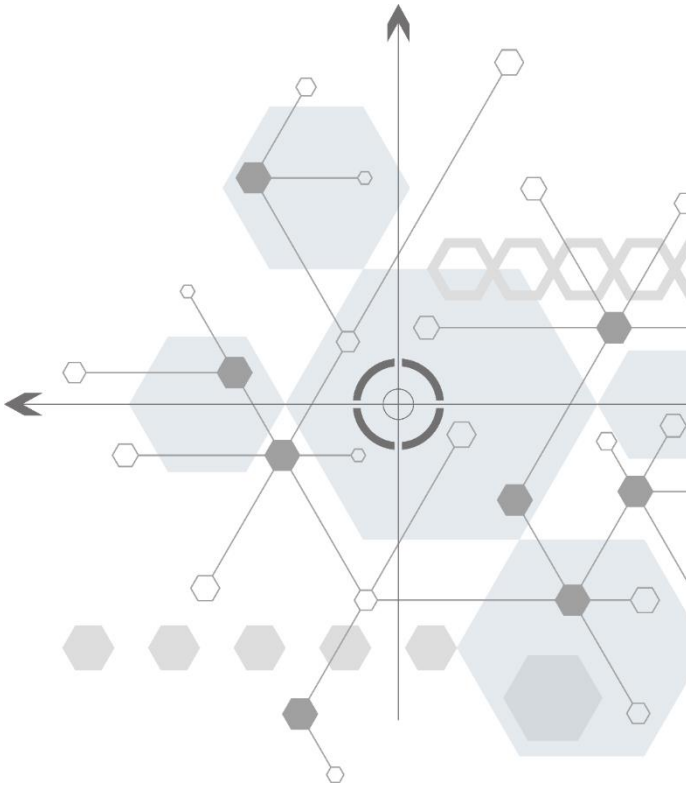
▼構文

[ ]	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

	指定バージョンからの新機能 （左記の場合、Oracle Database 23ai からの新機能）
	注意事項
	参考情報

本テキストは、Oracle Database 19c～23ai に対応しています。Oracle Database 23ai は、2025 年 10 月より、Oracle AI Database 26ai という名称に置き換わりました。



第 1 章
条件分岐

SQL における条件分岐の方法について説明します。

01　CASE 式
02　関数を使用した条件分岐
03　MERGE 文

本章のゴール

・CASE 式および関数を使用した条件分岐が行えるようになる。
・MERGE 文を使用して、2 つの表の比較結果に応じた UPDATE、INSERT 処理を行えるようになる。

01 CASE 式

CASE 式を使用すると、条件に応じて処理を分岐できます。

（1）特徴

CASE 式には次のような特徴があります。
・コードの読みやすさ
・複雑な分岐条件を設定可能
・さまざまな箇所で使用可能

1）コードの読みやすさ

DECODE 関数などの分岐関数を使用しても条件分岐は可能ですが、分岐関数を使用した場合は引数の意味を理解していないと、コードが読みづらくなります。CASE 式の場合は、分岐条件と処理内容をわかりやすく表現できます。

例）部門番号（DEPTNO 列）の値に応じて表示内容を変更する（DECODE 関数を使用して条件分岐）。

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

引数の意味を理解していないと分岐条件と処理内容の関係を把握できない。

例）部門番号（DEPTNO 列）の値に応じて表示内容を変更する（CASE 式を使用して条件分岐）。

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

WHEN 句や THEN 句などのキーワードによって、分岐条件と処理内容の関係が把握しやすい。

2）複雑な分岐条件を設定可能

CASE 式では分岐条件に演算子や副問い合わ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。

例）給与（SAL 列）の値に応じて表示内容を変更する。

```
SELECT ename,sal,
       CASE WHEN sal BETWEEN    1 AND 2000 THEN 'LOW'
            WHEN sal BETWEEN 2001 AND 4000 THEN 'MIDDLE'
            WHEN sal BETWEEN 4001 AND 6000 THEN 'HIGH'
            ELSE '不明'
       END
FROM   emp;
```

3）さまざまな箇所で使用可能

CASE「式」であるため、他の句と組み合わせて SQL のさまざまな箇所で条件分岐を行えます。

例）2000 以下の給与（SAL 列）を 2000 に切り上げて平均給与を算出する（AVG 関数と CASE 式の組み合わせ）。

```
SELECT AVG(CASE WHEN sal > 2000 THEN sal
                ELSE 2000
           END)
FROM   emp;
```

関数の引数以外にも、CHECK 制約の条件に CASE 式を含めることで、より複雑な制約を設定したり、UPDATE 文で CASE 式を使用することで、1 つの UPDATE 文で条件に応じた更新処理を行ったりすることができます。

参考：「CASE 式の使用方法」（A-1）

(2) 構文

CASE 式の構文には、単純 CASE 式と検索 CASE 式があります。

1) 単純 CASE 式

選択子と条件値を等価評価して、処理を分岐できます。

構文

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

選択子	比較対象となる値を指定します。
条件値	選択子と等価評価する値を指定します。
処理内容	選択子との等価評価が成立した際に行う処理内容を指定します。 ※ELSE 句には、すべての条件を満たさなかった場合に行う処理内容を指定します。

例）DEPTNO 列の値に応じて表示内容を変更する。

```
SQL> SELECT ename,deptno,
  2         CASE deptno WHEN 10 THEN 'ACCOUNTING'
  3                     WHEN 20 THEN 'RESEARCH'
  4                     WHEN 30 THEN 'SALES'
  5                     ELSE 'OPERATIONS'
  6         END
  7  FROM   emp
  8  WHERE  job = 'CLERK'
  9  ORDER BY deptno;

ENAME          DEPTNO CASEDEPTNO
---------- ---------- ----------
MILLER             10 ACCOUNTING
SMITH              20 RESEARCH
ADAMS              20 RESEARCH
JAMES              30 SALES
```

2）検索 CASE 式

分岐条件を 1 つずつ指定して、処理を分岐できます。分岐条件には等価評価以外の条件も指定できます。

構文

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

分岐条件	分岐条件を指定します。
処理内容	分岐条件を満たした場合に行う処理内容を指定します。 ※ELSE 句には、すべての条件を満たさなかった場合に行う処理内容を指定します。

例）DEPTNO 列の値に応じて表示内容を変更する。

```
SQL> SELECT ename,deptno,
  2         CASE WHEN deptno = 10 THEN 'ACCOUNTING'
  3              WHEN deptno = 20 THEN 'RESEARCH'
  4              WHEN deptno = 30 THEN 'SALES'
  5              ELSE 'OPERATIONS'
  6         END
  7  FROM   emp
  8  WHERE  job = 'CLERK'
  9  ORDER BY deptno;

ENAME          DEPTNO CASEWHENDE
---------- ---------- ----------
MILLER             10 ACCOUNTING
SMITH              20 RESEARCH
ADAMS              20 RESEARCH
JAMES              30 SALES
```

例）部門番号 30 の社員と WARD の給与を比較した結果に応じて「Few」「Same」「Many」「Others」と表示する。

```
SQL> SELECT deptno,ename,sal,
  2         CASE WHEN sal < (SELECT sal FROM emp
  3                          WHERE  ename = 'WARD') THEN 'Few'
  4              WHEN sal = (SELECT sal FROM emp
  5                          WHERE  ename = 'WARD') THEN 'Same'
  6              WHEN sal > (SELECT sal FROM emp
  7                          WHERE  ename = 'WARD') THEN 'Many'
  8              ELSE 'Others'
  9         END AS amount
 10  FROM  emp
 11  WHERE deptno = 30
 12  AND   ename != 'WARD';

    DEPTNO ENAME             SAL AMOUNT
---------- ---------- ---------- ------
        30 ALLEN            1600 Many
        30 MARTIN           1250 Same
        30 BLAKE            2850 Many
        30 TURNER           1500 Many
        30 JAMES             950 Few
```

WARD の給与よりも少ない場合の動作を指定。

WARD の給与と同じ場合の動作を指定。

WARD の給与よりも多い場合の動作を指定。

すべての条件を満たさない場合の動作を指定。

WITH 句を使用すると、分岐条件に指定する副問い合わせの記述をシンプルにすることができます。

参考：「WITH 句」(A-5)

⚠注意事項

単純 CASE 式や検索 CASE 式を記述する際には、次の点に注意が必要です。

・ELSE 句の記述

ELSE 句の記述を省略した場合は「ELSE NULL（NULL を戻す）」となります。

分岐条件をすべて満たさない場合に NULL を戻す場合であっても意図が伝わるように、ELSE 句は省略しないことをおすすめします。

例）職種（JOB 列）に応じて新給与（NEW_SAL）を計算する（ELSE 句を省略）。

```
SQL> SELECT ename,job,
  2         CASE WHEN job = 'MANAGER'  THEN sal*2
  3              WHEN job = 'SALESMAN' THEN sal*3
  4         END AS new_sal
  5  FROM   emp;

ENAME      JOB           NEW_SAL
---------- --------- ----------
SMITH      CLERK
ALLEN      SALESMAN         4800
…省略…
```

ELSE 句の記述が省略されているため、分岐条件をすべて満たさない場合に NULL を戻すことが意図した動作なのかがわからない。

例）職種（JOB 列）に応じて新給与（NEW_SAL）を計算する（ELSE 句を省略しない）。

```
SQL> SELECT ename,job,
  2         CASE WHEN job = 'MANAGER'  THEN sal*2
  3              WHEN job = 'SALESMAN' THEN sal*3
  4              ELSE NULL
  5         END AS new_sal
  6  FROM   emp;

ENAME      JOB           NEW_SAL
---------- --------- ----------
SMITH      CLERK
ALLEN      SALESMAN         4800
…省略…
```

ELSE 句が記述されているため、分岐条件をすべて満たさない場合に NULL を戻すことが意図した動作であることがわかる。

・戻される値のデータ型

THEN 句と ELSE 句で戻される値のデータ型は、すべて一致している必要があります。

例）戻される値のデータ型を変えて CASE 式を実行する。

```
SQL> SELECT ename,deptno,
  2         CASE WHEN deptno = 10 THEN 'ACCOUNTING'
  3              WHEN deptno = 20 THEN 1
  4              ELSE NULL
  5         END
  6  FROM   emp;
           WHEN deptno = 20 THEN 1
                                 *
行 3 でエラーが発生しました。:
ORA-00932: 式
(1)のデータ型は NUMBER ですが、これは予期されたデータ型 CHAR と互換性がありません ヘルプ:
https://docs.oracle.com/error-help/db/ora-00932/
```

1 つ目の分岐条件で文字型、
2 つ目の分岐条件で数字型が
戻されるため、エラーが発生。

・分岐条件の記述順

分岐条件は記述順に評価され、条件が成立した時点で評価が打ち切られるため、先に記述した条件が後に記述した条件を含む場合は意図しない結果となります。そのため、先に記述した条件が後に記述した条件を含まないようにする、または複数の条件が重複しないように記述するなどの注意が必要です。

例）給与（SAL 列）の値に応じて表示内容を変更する。

```
SQL> SELECT sal,
  2         CASE WHEN sal >= 2500 THEN 'Few'
  3              WHEN sal >= 5000 THEN 'Many'
  4              ELSE 'Unknown'
  5         END AS amount
  6  FROM   emp
  7  ORDER BY sal DESC;

       SAL AMOUNT
---------- ------
      5000 Few
      3000 Few
…省略…
```

分岐条件は記述順に評価される。

SAL 列が 5000 の場合、「Many」と表示したいが、
1 つ目の分岐条件（SAL 列が 2500 以上）に合致するため、
2 つ目の分岐条件（SAL 列が 5000 以上）が評価されていない。

CASE 式の評価結果に対しては、列の別名を定義することをおすすめします。列の別名を定義していない場合は、CASE 式の記述がそのまま列ヘッダーに表示されます。