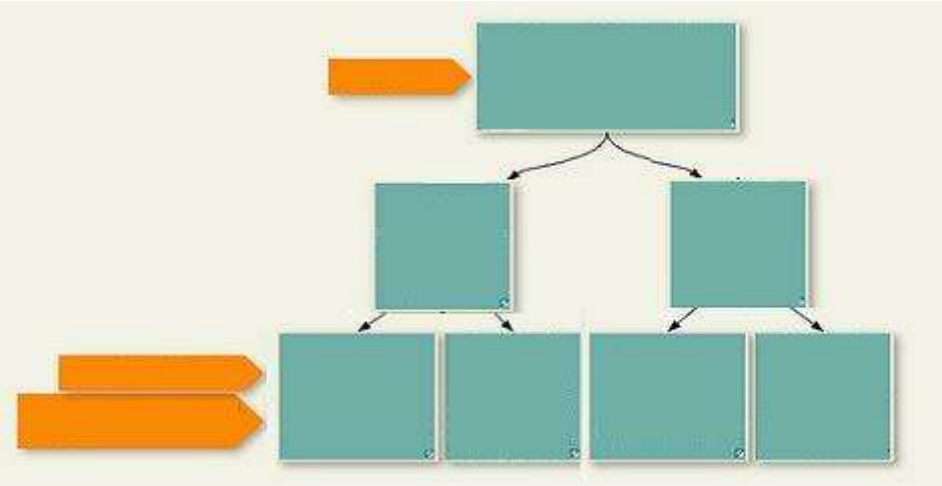
ORATABLE
ORACLE STUFF WORTH TALKING ABOUT

The Difference Between DECODE and CASE

July 19, 2010

in FAQ, Keywords, PL/SQL, SQL



DECODE and CASE statements in Oracle both provide a conditional construct, of this form:
if A = n1 then A1
else if A = n2 then A2
else X

Databases before Oracle 8.1.6 had only the DECODE function. CASE was introduced in Oracle 8.1.6 as a standard, more meaningful and more powerful function.

Everything DECODE can do, CASE can. There is a lot else CASE can do though, which DECODE cannot. We'll go through detailed examples in this article.

1. CASE can work with logical operators other than '='

DECODE performs an equality check only. CASE is capable of other logical comparisons such as < > etc. It takes some complex coding – forcing ranges of data into discrete form – to achieve the same effect with

DECODE.

An example of putting employees in grade brackets based on their salaries. This can be done elegantly with CASE.

```
SQL> select ename
  2       , case
  3           when sal < 1000
  4                then 'Grade I'
  5           when (sal >=1000 and sal < 2000)
  6                then 'Grade II'
  7           when (sal >= 2000 and sal < 3000)
  8                then 'Grade III'
  9           else 'Grade IV'
 10         end sal_grade
 11  from emp
 12  where rownum < 4;

ENAME      SAL_GRADE
---------- ---------
SMITH      Grade I
ALLEN      Grade II
WARD       Grade II
```

2. CASE can work with predicates and searchable subqueries

DECODE works with expressions that are scalar values only. CASE can work with predicates and subqueries in searchable form.

An example of categorizing employees based on reporting relationship, showing these two uses of CASE.

```
SQL> select e.ename,
  2         case
  3           -- predicate with "in"
  4           -- set the category based on ename list
  5           when e.ename in ('KING','SMITH','WARD')
  6                then 'Top Bosses'
  7           -- searchable subquery
  8           -- identify if this emp has a reportee
  9           when exists (select 1 from emp emp1
 10                        where emp1.mgr = e.empno)
 11                then 'Managers'
 12           else
 13               'General Employees'
 14         end emp_category
 15  from emp e
 16  where rownum < 5;

ENAME      EMP_CATEGORY
---------- -----------------
SMITH      Top Bosses
ALLEN      General Employees
WARD       Top Bosses
JONES      Managers
```

3. CASE can work as a PL/SQL construct

DECODE can work as a function inside SQL only. CASE can be an efficient substitute for IF-THEN-ELSE in PL/SQL.

```
SQL> declare
  2    grade char(1);
  3  begin
  4    grade := 'b';
  5    case grade
  6      when 'a' then dbms_output.put_line('excellent');
  7      when 'b' then dbms_output.put_line('very good');
  8      when 'c' then dbms_output.put_line('good');
  9      when 'd' then dbms_output.put_line('fair');
 10      when 'f' then dbms_output.put_line('poor');
 11      else dbms_output.put_line('no such grade');
 12    end case;
 13  end;
 14  /

PL/SQL procedure successfully completed.
```

CASE can even work as a parameter to a procedure call, while DECODE cannot.

```
SQL> var a varchar2(5);
SQL> exec :a := 'THREE';

PL/SQL procedure successfully completed.

SQL>
SQL> create or replace procedure proc_test (i number)
  2  as
  3  begin
  4    dbms_output.put_line('output = '||i);
  5  end;
  6  /

Procedure created.

SQL> exec proc_test(decode(:a,'THREE',3,0));
BEGIN proc_test(decode(:a,'THREE',3,0)); END;

                *
ERROR at line 1:
ORA-06550: line 1, column 17:
PLS-00204: function or pseudo-column 'DECODE' may be used inside a SQL
statement only
ORA-06550: line 1, column 7:
PL/SQL: Statement ignored


SQL> exec proc_test(case :a when 'THREE' then 3 else 0 end);
output = 3

PL/SQL procedure successfully completed.
```

4. Careful! CASE handles NULL differently

Check out the different results with DECODE vs NULL.

```
SQL> select decode(null
  2              , null, 'NULL'
  3                    , 'NOT NULL'
  4               ) null_test
  5  from dual;

NULL
```

```
----
NULL

SQL> select case null
  2         when null
  3         then 'NULL'
  4         else 'NOT NULL'
  5         end null_test
  6  from dual;

NULL_TES
--------
NOT NULL
```

The "searched CASE" works as does DECODE.

```
SQL>  select case
  2         when null is null
  3         then 'NULL'
  4         else 'NOT NULL'
  5         end null_test
  6* from dual
SQL> /

NULL_TES
--------
NULL
```

5. CASE expects datatype consistency, DECODE does not

Compare the two examples below- DECODE gives you a result, CASE gives a datatype mismatch error.

```
SQL> select decode(2,1,1,
  2                '2','2',
  3                '3') t
  4  from dual;

         T
----------
         2

SQL> select case 2 when 1 then '1'
  2              when '2' then '2'
  3              else '3'
  4         end
  5  from dual;
            when '2' then '2'
                 *
ERROR at line 2:
ORA-00932: inconsistent datatypes: expected NUMBER got CHAR
```

6. CASE is ANSI SQL-compliant

CASE complies with ANSI SQL. DECODE is proprietary to Oracle.

7. The difference in readability

In very simple situations, DECODE is shorter and easier to understand than CASE.

```
SQL> -- An example where DECODE and CASE
SQL> -- can work equally well, and
SQL> -- DECODE is cleaner

SQL> select ename
  2       , decode (deptno, 10, 'Accounting',
  3                         20, 'Research',
  4                         30, 'Sales',
  5                             'Unknown') as department
  6  from   emp
  7  where rownum < 4;

ENAME      DEPARTMENT
---------- ----------
SMITH      Research
ALLEN      Sales
WARD       Sales

SQL> select ename
  2       , case deptno
  3           when 10 then 'Accounting'
  4           when 20 then 'Research'
  5           when 30 then 'Sales'
  6           else         'Unknown'
  7           end as department
  8  from emp
  9  where rownum < 4;

ENAME      DEPARTMENT
---------- ----------
SMITH      Research
ALLEN      Sales
WARD       Sales
```

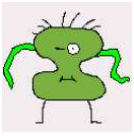
Complicated logical comparisons in DECODE, even if technically achievable, are a recipe for messy, bug-prone code. When the same can be done more cleanly with CASE, go for CASE.

[Photo by natematias]

39

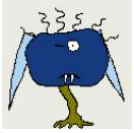
very precisely:)

{ 66 comments… read them below or add one }



mahesh kumar December 14, 2011 at 1:50 pm

veny nice explanation



subhashini March 22, 2012 at 12:46 pm

Thanks for providing the information