Table of Contents

The Case expressions are used to perform selection logic. The case expression is part of standard SQL and corresponds closely to selection logic found in most programming languages. The case expression is not a function but it is a bit more complex that the simpler expressions we used in earlier units.

1. Searched Case Expression

The searched Case expression requires a logical expression to be evaluated at each WHEN clause. It is faster than evaluating the DECODE function and is closer to the logic found in many programming languages. (Decode is discussed in another document in this unit.)

The data type of the return expressions must all be of the same type family or all numeric; that type becomes the return type of the case expression; if return expressions are all numeric then the return type of the case expression is the highest precedence of the various return expression numeric types.

You can use a variety of tests- In lists, Between, wildcard tests and you can mix the tests in a single case expression. You can nest case expressions.

Demo 01: We want to give customers a 5% savings for each pet supply item, 5% for each sporting goods item and 10% for each appliance. As a first step we will determine the saving percent.

```
select  catg_id, prod_id, prod_list_price
, CASE
     WHEN catg_id ='PET'   THEN 0.95
     WHEN catg_id ='SPG'   THEN 0.95
     WHEN catg_id ='APL'   THEN 0.90
  ELSE  1
  END  "Savings %"
from prd_products
order by catg_id;
```

--- selected rows shown

```
CATG_I    PROD_ID PROD_LIST_PRICE  Savings %
------ ---------- --------------- ----------
APL          1126             850         .9
APL          1125             500         .9
GFD          5001               5          1
GFD          5000            12.5          1
HD           5002              23          1
HD           5005              45          1
HW           4575           49.95          1
HW           1090          149.99          1
HW           1000             125          1
HW           1070            25.5          1
PET          1140           14.99        .95
PET          4577           29.95        .95
PET          4576           29.95        .95
PET          4568          549.99        .95
SPG          1060          255.95        .95
SPG          1050          269.95        .95
```

Demo 02: We can use the calculated percent to determine the sales price

```
select  catg_id, prod_id, prod_list_price
, CASE
     WHEN catg_id ='PET'   THEN 0.95
     WHEN catg_id ='SPG'   THEN 0.95
     WHEN catg_id ='APL'   THEN 0.90
  ELSE  1
  END  * prod_list_price AS "Today's Price"
from prd_products
order by catg_id;
--- selected rows shown
```

CATG_I	PROD_ID	PROD_LIST_PRICE	Today's Price
APL	4569	349.95	314.955
APL	1120	549.99	494.991
APL	1130	149.99	134.991
GFD	5001	5	5
GFD	5000	12.5	12.5
HD	5002	23	23
HD	5005	45	45
HW	1070	25.5	25.5
PET	1140	14.99	14.2405
PET	4576	29.95	28.4525
PET	1141	99.99	94.9905
PET	1142	2.5	2.375
SPG	1060	255.95	243.1525

Demo 03: You should include an Else clause unless you are certain that all possible values are handled. Here I have removed the else clause and products which do not fall into one of the three categories tested, get a value of null from the case expression and therefore have a null value for the last column. This does not follow the business rule of demo 01.

```
select  catg_id, prod_id, prod_list_price
, CASE
     WHEN catg_id ='PET'   THEN 0.95
     WHEN catg_id ='SPG'   THEN 0.95
     WHEN catg_id ='APL'   THEN 0.90
  END  * prod_list_price AS "Today's Price"
from prd_products
order by catg_id;
--- selected rows shown
```

CATG_I	PROD_ID	PROD_LIST_PRICE	Today's Price
APL	4569	349.95	314.955
APL	1120	549.99	494.991
APL	1130	149.99	134.991
GFD	5001	5	
GFD	5000	12.5	
HD	5002	23	
HW	1000	125	
HW	1070	25.5	
PET	1140	14.99	14.2405
PET	4576	29.95	28.4525
PET	1142	2.5	2.375
SPG	1060	255.95	243.1525

For more maintainable code you might wish to have a table of categories and current discounts and do a join. That way if the discount rates changed or if new categories were added, you would not have to rewrite the query.

1.1. Return type consistency

We cannot write a case expression such as the following queries.

Demo 04: This fails since the return type was expected to be a number and one of the return expression is a character 'no discount' as the return value.

```
select  catg_id, prod_id, prod_list_price
, CASE
     WHEN catg_id ='PET'   THEN 0.95
     WHEN catg_id ='SPG'   THEN 0.95
     WHEN catg_id ='APL'   THEN 0.90
  ELSE  'no discount'
  END  "Savings %"
from prd_products
order by catg_id
;
```

```
SQL Error: ORA-00932: inconsistent datatypes: expected NUMBER got CHAR
```

Demo 05: This fails since the return type was expected to be a char and other the return expressions are numbers

```
select  catg_id, prod_id, prod_list_price
, CASE
     WHEN catg_id ='PET'   THEN 'no discount'
     WHEN catg_id ='SPG'   THEN 0.95
     WHEN catg_id ='APL'   THEN 0.90
  ELSE  1
  END  "Savings %"
from prd_products
order by catg_id
;
```

```
SQL Error: ORA-00932: inconsistent datatypes: expected CHAR got NUMBER
```

Demo 06: This fails since we have mixed return expression types even though we do not have any catg_id value of 'XXX' and that expression would never be evaluated.

```
select  catg_id, prod_id, prod_list_price
, CASE
     WHEN catg_id ='PET'   THEN 0.95
     WHEN catg_id ='SPG'   THEN 0.95
     WHEN catg_id ='APL'   THEN 0.90
     WHEN catg_id ='XXX'   THEN 'Invalid'
  ELSE  1
  END  "Savings %"
from prd_products
order by catg_id
;
```

```
SQL Error: ORA-00932: inconsistent datatypes: expected NUMBER got CHAR
```

1.2. Including other functions

Demo 07: We can then include the round function to improve the results.

```
select  catg_id, prod_id, prod_list_price
, Round(
     CASE
```

```
         WHEN catg_id ='PET'   THEN 0.95
         WHEN catg_id ='SPG'   THEN 0.95
         WHEN catg_id ='APL'   THEN 0.90
      ELSE  1
      END  * prod_list_price, 2 ) AS "Today's Price"
from prd_products
order by catg_id;
--- selected rows shown
```

CATG_I	PROD_ID	PROD_LIST_PRICE	Today's Price
APL	4569	349.95	314.96
APL	1120	549.99	494.99
APL	1130	149.99	134.99
APL	1126	850	765

In the next example we want the discount to apply only to products with a list price of $50 or higher. The first When clause with a true value determines the result

Demo 08: –The first When clause with a true value determines the result. Items with prices under $50 are not considered for a discount. This one uses the To_Char function to format the column.

```
select  catg_id, prod_id, prod_list_price
,  To_Char(
      CASE
         WHEN prod_list_price < 50 THEN 1
         WHEN catg_id ='PET'   THEN 0.95
         WHEN catg_id ='SPG'   THEN 0.95
         WHEN catg_id ='APL'   THEN 0.90
      ELSE  1
      END  * prod_list_price, '9999.00') AS "Today's Price"
from prd_products
order by catg_id
;
--- selected rows
```

CATG_I	PROD_ID	PROD_LIST_PRICE	Today's Price
APL	4569	349.95	314.96
APL	1120	549.99	494.99
APL	1130	149.99	134.99
APL	1126	850	765.00
APL	1125	500	450.00
GFD	5001	5	5.00
GFD	5000	12.5	12.50
HD	5002	23	23.00
HD	5005	45	45.00
HD	5004	15	15.00
HD	5008	12.5	12.50

The next case structure looks daunting in code but look at the output first. With appliances we merely report back that this is an appliance item. With pet supplies and sporting good we break these down into cost categories (high, low, medium). The break points for sporting goods and pet supplies are different. For all other categories we do not report anything.

The outer case structure is based on the category id- there is a block for PET, another block for SPG, a third block for APL and there is no Else block. Items which do not fit in one of these categories do not get a block and the case returns a null. When you develop this code, you should write and test the outer case structure first.

The inner case structure for PET and the inner case structure for SPG are based on the prod_list_price

Demo 09: A nested Case structure using prd_products

```
select  catg_id, prod_id, prod_list_price
, CASE
     WHEN catg_id ='PET'   THEN
       CASE
          WHEN prod_list_price < 10 THEN 'LowCost pet item'
       ELSE 'HighCost pet item'
       END
     WHEN catg_id ='SPG'   THEN
       CASE
          WHEN prod_list_price < 25               THEN 'LowCost sports item'
          WHEN prod_list_price between 25 and 150 THEN 'MidCost sports item'
       ELSE 'HighCost sports item'
       END
     WHEN catg_id ='APL'   THEN 'appliance item'
  END  AS "Result"
from prd_products
order by prod_id;
--- selected rows shown
```

```
CATG_I    PROD_ID PROD_LIST_PRICE Result
------ ---------- --------------- -------------------
HW           1000             125
SPG          1010             150 MidCost sports item
SPG          1020           12.95 LowCost sports item
SPG          1030           29.95 MidCost sports item
SPG          1040          349.95 HighCost sports ite
HW           1090          149.99
HW           1100           49.99
APL          1120          549.99 appliance item
APL          1130          149.99 appliance item
PET          1140           14.99 HighCost pet item
PET          1141           99.99 HighCost pet item
PET          1142             2.5 LowCost pet item
PET          1152              55 HighCost pet item
HW           1160          149.99
PET          4567          549.99 HighCost pet item
PET          4568          549.99 HighCost pet item
APL          4569          349.95 appliance item
HW           4575           49.95
PET          4577           29.95 HighCost pet item
```

If we want to display a message instead of the missing value, we can wrap a coalesce function around the entire case expression.: `Coalesce(CASE  . . . END, 'No information available') as "Result"`

Demo 10: We have a look up table for the credit ratings. This is another approach. If the credit levels for the rating terms were to change frequently, the lookup table would be a better approach.

```
select  customer_id, customer_credit_limit
,  CASE
      WHEN customer_credit_limit >= 10001  THEN 'Superior'
      WHEN customer_credit_limit >=  5001  THEN 'Excellent'
      WHEN customer_credit_limit >=  2001  THEN 'High'
      WHEN customer_credit_limit >=  1001  THEN 'Good'
   ELSE 'Standard'
   END  AS  Rating
from cust_customers;
--- selected rows shown
```

```
CUSTOMER_ID CUSTOMER_CREDIT_LIMIT RATING
----------- --------------------- ---------
```

```
     001250                  750 Standard
     001890                 1750 Good
     002100                  750 Standard
     002110                  750 Standard
     002120                  750 Standard
     002500                      Standard
     003000                 6000 Excellent
     003500                 6000 Excellent
     003750                 6000 Excellent
     003760                 6000 Excellent
     004000                 3500 High
     004100                 3500 High
```

2. Simple Case Expression

Oracle has another version of the Case expression called a Simple Case expression.

Demo 11: Simple case; only one attribute is being compared; the comparisons are all equality tests.

```
select  catg_id, prod_id, prod_list_price
, CASE catg_id
     WHEN 'PET'   THEN 0.95
     WHEN 'SPG'   THEN 0.95
     WHEN 'APL'   THEN 0.90
  ELSE  1
  END  * prod_list_price AS "Today's Price"
from prd_products;

--- selected rows shown
```

```
CATG_I    PROD_ID PROD_LIST_PRICE Today's Price
------ ---------- --------------- -------------
HW           1000             125           125
SPG          1010             150         142.5
SPG          1020           12.95       12.3025
HW           1090          149.99        149.99
HW           1100           49.99         49.99
PET          1140           14.99       14.2405
PET          1141           99.99       94.9905
PET          1142             2.5         2.375
SPG          1040          349.95      332.4525
SPG          1050          269.95      256.4525
SPG          1060          255.95      243.1525
HW           1160          149.99        149.99
PET          4567          549.99      522.4905
APL          4569          349.95       314.955
HW           4575           49.95         49.95
APL          1125             500           450
APL          1126             850           765
APL          1130          149.99       134.991
GFD          5000            12.5          12.5
GFD          5001               5             5
HD           5002              23            23
HD           5004              15            15
```

Demo 12: Organizing sales by season

```
select  order_id, To_char(order_date, 'yyyy-mm-dd') AS OrderDate
, CASE TO_CHAR(order_date, 'q')
     WHEN '1'   THEN 'winter'
```

```
      WHEN '2'   THEN 'spring'
      WHEN '3'   THEN 'summer'
      WHEN '4'   THEN 'fall'
  END   AS "Season"
from oe_orderHeaders ;

--- selected rows shown
```

ORDER_ID	ORDERDATE	Season
540	2015-06-02	spring
390	2015-06-04	spring
395	2015-06-04	spring
609	2015-09-27	summer
610	2015-09-27	summer
611	2015-09-30	summer
105	2015-10-01	fall
106	2015-10-01	fall
402	2015-10-18	fall
405	2015-11-19	fall

Demo 13: Using a case to do a special sort. We want to sort the products by the categories but not alphabetically. The order we want to use is PET, SPG, APL, HW.

```
select  catg_id, prod_id, prod_list_price
from prd_products
order by CASE catg_id
            WHEN 'PET'   THEN 1
            WHEN 'SPG'   THEN 2
            WHEN 'APL'   THEN 3
            WHEN 'HW'    THEN 4
         ELSE  9999
         END,
         catg_id, prod_id;

--- selected rows shown
```

CATG_I	PROD_ID	PROD_LIST_PRICE
PET	1140	14.99
PET	1141	99.99
PET	1142	2.5
PET	1150	4.99
PET	1151	14.99
PET	1152	55
SPG	1030	29.95
SPG	1060	255.95
APL	1120	549.99
APL	1125	500
APL	1126	850
HW	1090	149.99
HW	1100	49.99
HW	1110	49.99
HW	1160	149.99
HW	4575	49.95
GFD	5000	12.5
GFD	5001	5
HD	5005	45
HD	5008	12.5