# Using the CASE Statement as a Transformation

## The Scenario

In many cases an OLTP source system delivers codes or numbered values to represent certain states of the source data. In order to transform this data into more legible values, a transformation is needed.

Two cases are most frequently used in business intelligence environments:

- Lookup to a list of values.
- A CASE statement to translate the values.

The lookup is typically used in cases where many values need to be translated or where the list is dynamic in itself.

The CASE statement is used in scenarios where the list of values is relatively short or static, which allows for incorporation of the values within the statement itself.

The lookup would be implemented with the key lookup operator. This example does not cover this solution.

This example will cover three ways of applying a CASE statement within Warehouse Builder.

## Case study environment

The environment used in this example consists of the following components:

- Oracle 9.0.1 database
- Warehouse Builder 9.0.33.3.0 or 9.0.3.35.0

The schema and data used in this example are coming from the OE schema in the Oracle 9.0.1 database, more specifically the tables:

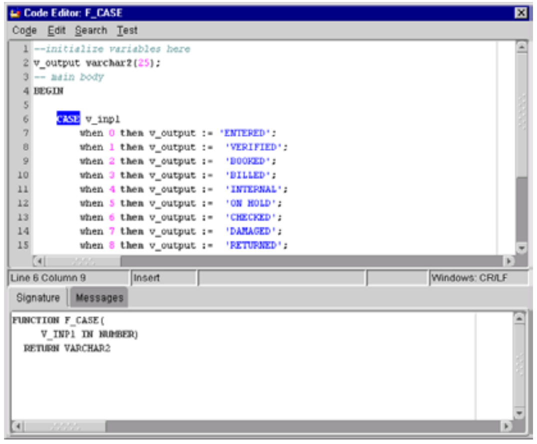
- Orders

The Orders table contains an order_status column that holds numerical representations for order status. When analyzing orders it is clearer to your users if you translate the numerical values in meaningful string values. A status of 0 does not mean a lot to the average user, but a status of 'Entered' does.

## Method 1 Using a Transformation

In order to use a function in Warehouse Builder as a transformation, the code first needs to be written and registered. In this case you create a function that returns one row for each lookup.

In the example file we have defined a function F_CASE with the following code:

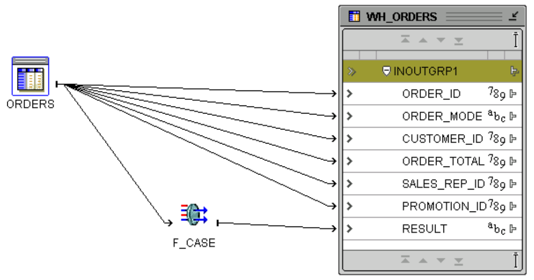


```
--initialize variables here
v_output varchar2(25);
-- main body
BEGIN

    CASE v_inp1
        when 0 then v_output := 'ENTERED';
        when 1 then v_output :=  'VERIFIED';
        when 2 then v_output :=  'BOOKED';
        when 3 then v_output :=  'BILLED';
        when 4 then v_output :=  'INTERNAL';
        when 5 then v_output :=  'ON HOLD';
        when 6 then v_output :=  'CHECKED';
        when 7 then v_output :=  'DAMAGED';
        when 8 then v_output :=  'RETURNED';
```

```
FUNCTION F_CASE(
     V_INP1 IN NUMBER)
  RETURN VARCHAR2
```

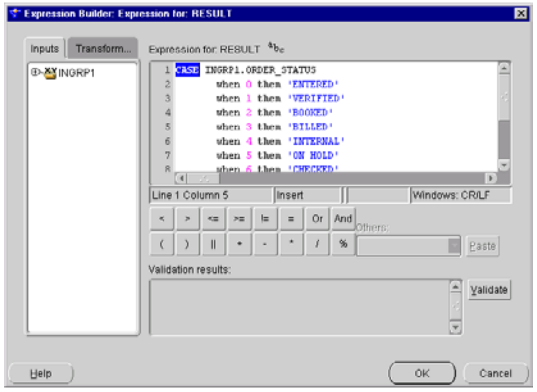
This function will transform the numerical values in the ORDER_STATUS column of OE.ORDERS into a more expressive string.



The transformation is used in the mapping CASE_MAP_TRFM to transform and transport data into the target table WH_ORDERS.

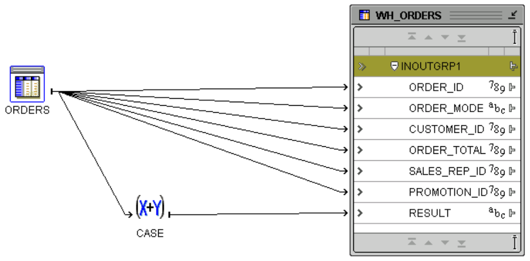
## Method 2 Using an Expression

When using an expression in Warehouse Builder you are essentially replicating the approach chosen in the first method, with the difference that you will add the entire case statement explicitly into the SQL code.



```
CASE INGRP1.ORDER_STATUS
        when 0 then 'ENTERED'
        when 1 then 'VERIFIED'
        when 2 then 'BOOKED'
        when 3 then 'BILLED'
        when 4 then 'INTERNAL'
        when 5 then 'ON HOLD'
        when 6 then 'CHECKED'
```

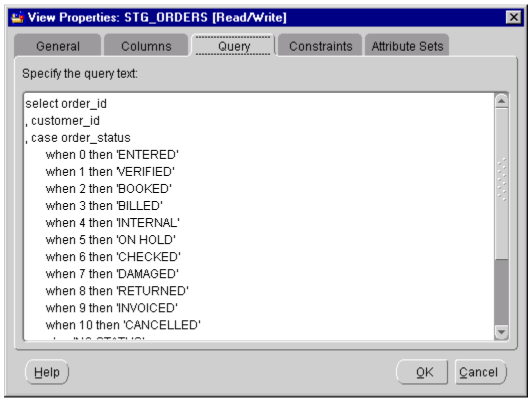
The mapping CASE_MAP_EXPR shows this example.



The draw back of this method is that the expression cannot be reused since it is not stored as a separate object in the repository.

## Method 3 Using a View

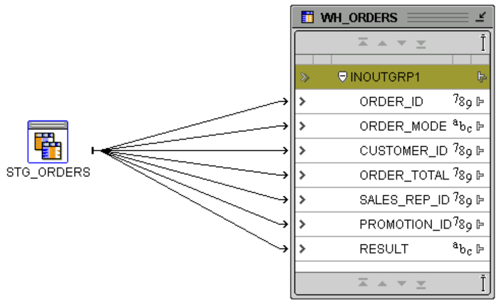
The third method that can be applied is the usage of a view. Using a view hides the transformation logic from the user because it is not visible in the mapping.



```
select order_id
, customer_id
, case order_status
    when 0 then 'ENTERED'
    when 1 then 'VERIFIED'
    when 2 then 'BOOKED'
    when 3 then 'BILLED'
    when 4 then 'INTERNAL'
    when 5 then 'ON HOLD'
    when 6 then 'CHECKED'
    when 7 then 'DAMAGED'
    when 8 then 'RETURNED'
    when 9 then 'INVOICED'
    when 10 then 'CANCELLED'
```

The approach used in a view is exactly the same as the approach used in method 2. In other words the expression is handled in SQL and added to the selection.

As an example take a look at the view STG_ORDERS.



Using the view in a mapping will then eliminate the usage of transformations in the mapping and the load process would map one-to-one to the target.

## CASE vs. DECODE

Warehouse Builder generates various types of code, some in row-based mode, some in set-based mode. To guarantee that the code works in all environments it is recommended to use CASE instead-of DECODE. The CASE statement does not work in row-based mode generation in an Oracle8i database.

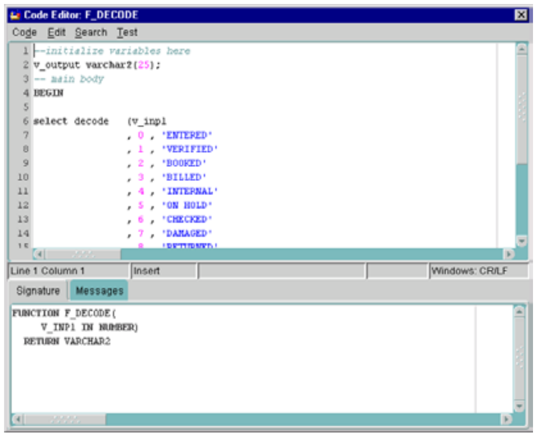
Another reason to use the CASE statement is that it is faster than the DECODE statement, which can be an important characteristic in an ETL environment.

## Oracle 8*i* vs. Oracle 9*i*

CASE is available in Oracle 8i however only in the SQL context. This means that the CASE statement poses the same problem in the Oracle 8i environment as the DECODE does.

Providing a PL/SQL wrapper (using a select from dual to gather the data) around the DECODE or the CASE statement solves this issue.

The function F_DECODE is created in Warehouse Builder to achieve decode on the Oracle 8.1.7 instance.

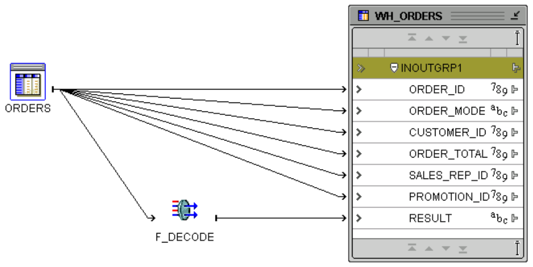


```
--initialize variables here
v_output varchar2(25);
-- main body
BEGIN

select decode    (v_inp1
                 , 0 , 'ENTERED'
                 , 1 , 'VERIFIED'
                 , 2 , 'BOOKED'
                 , 3 , 'BILLED'
                 , 4 , 'INTERNAL'
                 , 5 , 'ON HOLD'
                 , 6 , 'CHECKED'
                 , 7 , 'DAMAGED'
```

```
FUNCTION F_DECODE(
    V_INP1 IN NUMBER)
 RETURN VARCHAR2
```

Then you apply the transformation in mapping DECODE_MAP_TRFM. The construct is valid in all mapping code implementations and on both the 8*i* and 9*i* environment.



## Conclusion

With its versatile user interface Warehouse Builder lets you decide the implementation most suitable for your environment.

If reuse is important creating a function is the optimal solution, however if this is of no concern than both other methods can be used. Using a view makes the implementation less transparent, especially when considering lineage and impact analysis.

## More Information

For more information please visit the following sites.

General information about Oracle:

www.oracle.com

For more detailed information about Warehouse Builder:

http://otn.oracle.com/products/warehouse/content.html

Or join the Warehouse Builder forum on OTN:

http://www.oracle.com/forums/forum.jsp?id=1164496