# Unit 5.1

# DATAWAREHOUSING

UNIT 5

CHAPTER 1

# 1.ETL: Transformations and Other Operators:

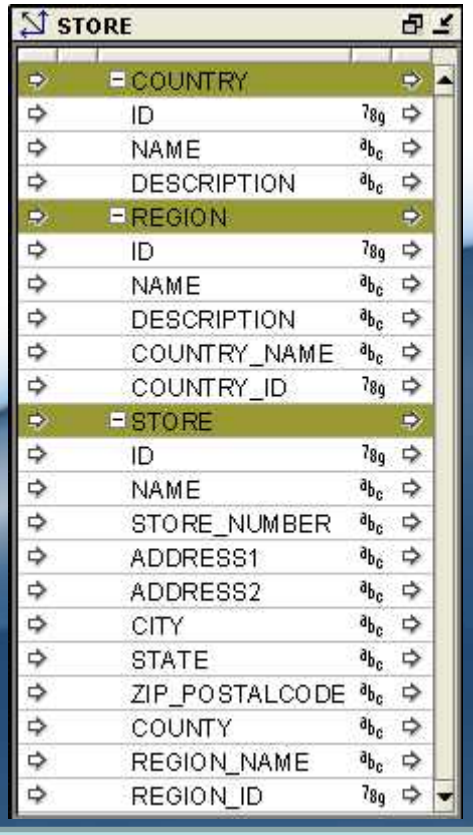
- STORE mapping, Adding source and target operators, Adding Transformation Operators, Using a Key Lookup operator, Creating an external table, Creating and loading a lookup table, Retrieving the key to use for a Lookup Operator, Adding a Key Lookup operator, PRODUCT mapping, SALES cube mapping, Dimension attributes in the cube, Measures and other attributes in the cube, Mapping values to cube attributes, Mapping measures' values to a cube, Mapping PRODUCT and STORE dimension values to the cube, Mapping DATE_DIM values to the cube, Features and benefits of OWB.

# STORE mapping

- In the **Design Center,** we will right-click on the Mappings node of the **ACME_DW_PROJECT | Databases | Oracle | ACME_DWH database** and select New…. Enter **STORE_MAP**

# Adding source and target operators

- Use POS_TRANS_STAGE table as our **source table**.
- **The target** for this mapping is going to be the STORE dimension

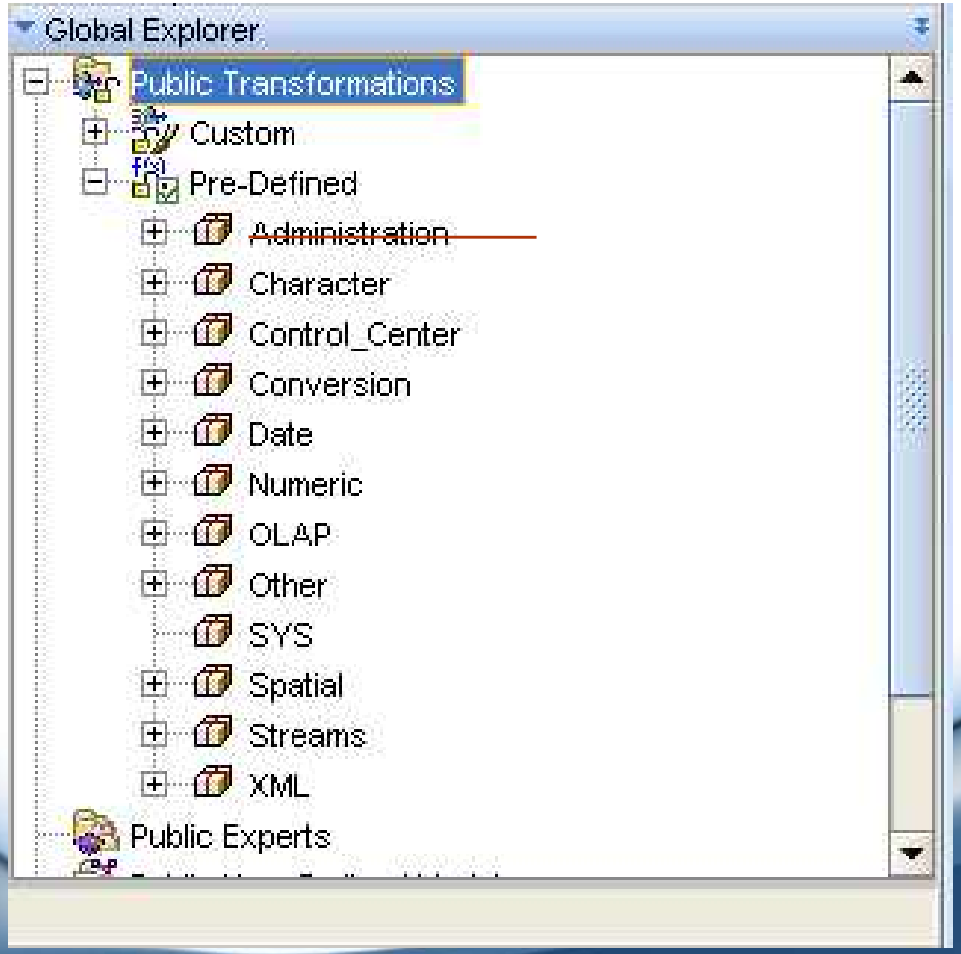


# connect the following attributes together:

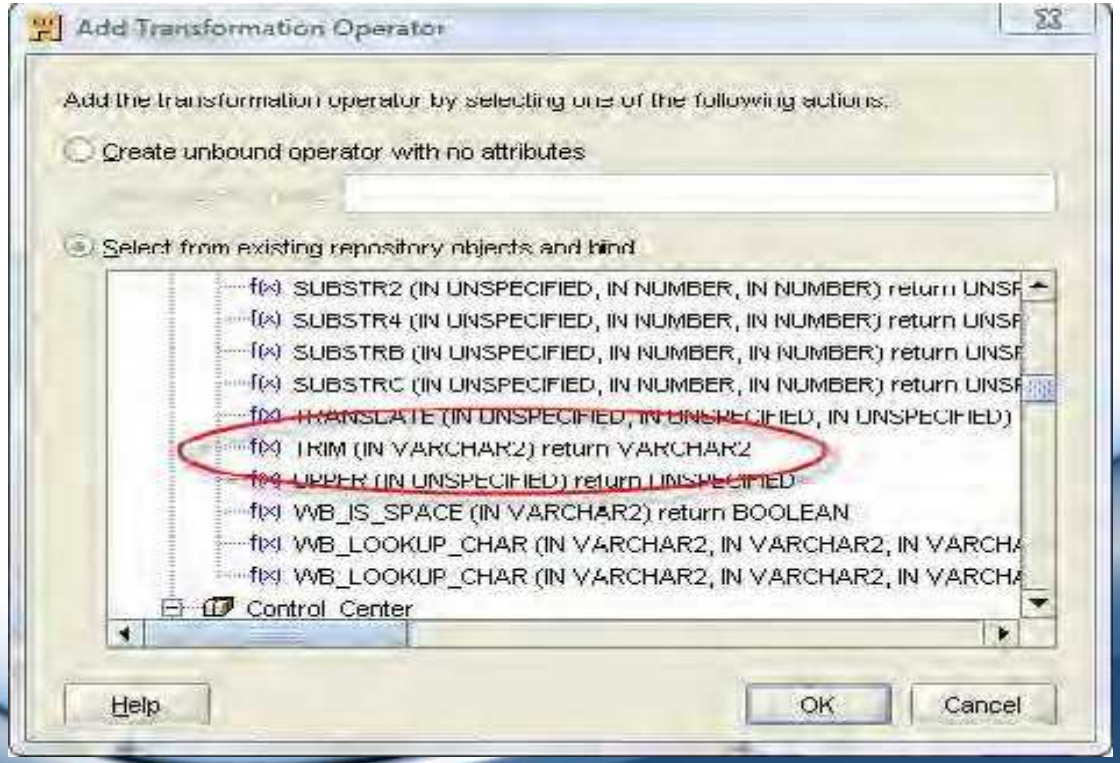
- STORE_NAME to NAME
- STORE_NUMBER to STORE_NUMBER
- STORE_ADDRESS1 to ADDRESS1
- STORE_ADDRESS2 to ADDRESS2
- STORE_CITY to CITY
- STORE_STATE to STATE
- STORE_ZIPPOSTALCODE to ZIP_POSTALCODE
- STORE_REGION to REGION_NAME

- STORE_NAME, STORE_NUMBER: We need to trim spaces and change these attributes to uppercase to facilitate queries as they are part of the business identifier
- STORE_ADDRESS1, ADDRESS2, CITY, STATE, and ZIP_POSTALCODE: We need to trim spaces and change the STATE attribute to uppercase
- STORE_REGION: We need to trim spaces and change this attribute to uppercase

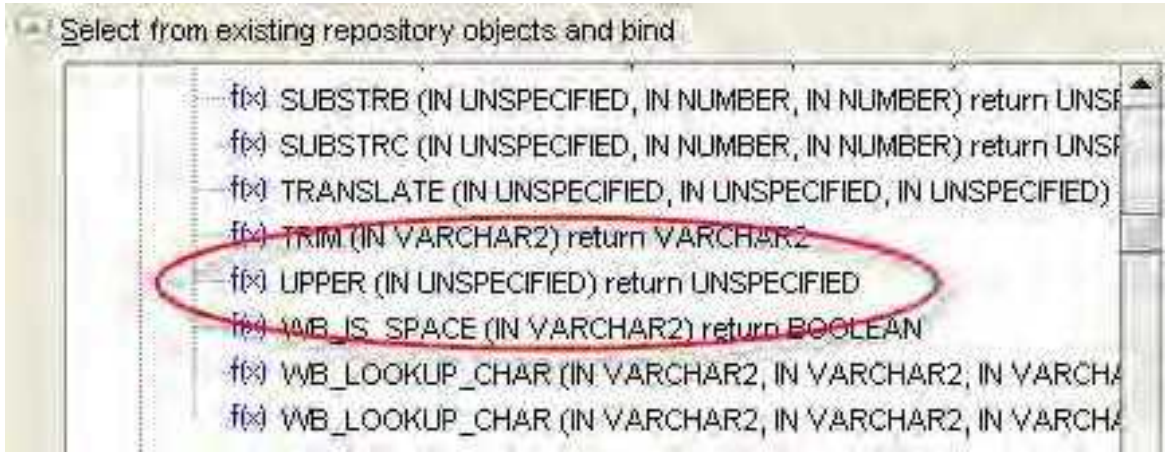
# Adding Transformation Operators



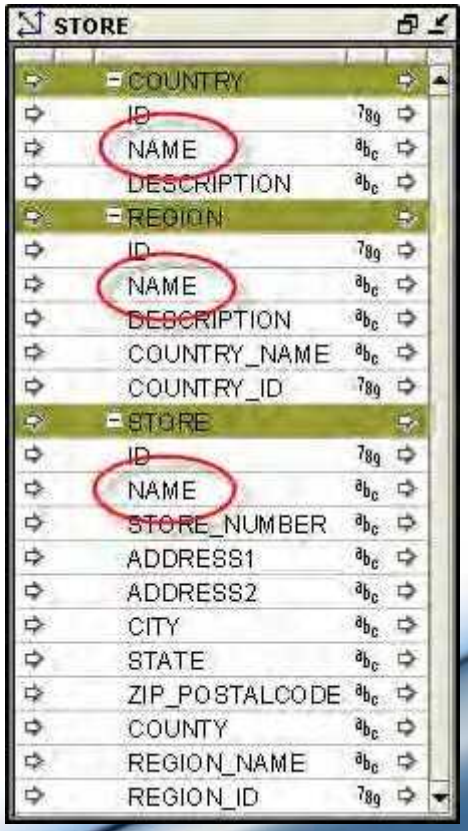

The particular transformation names we need under the character heading are the upper() function to convert to uppercase and the trim() function to remove whitespace.

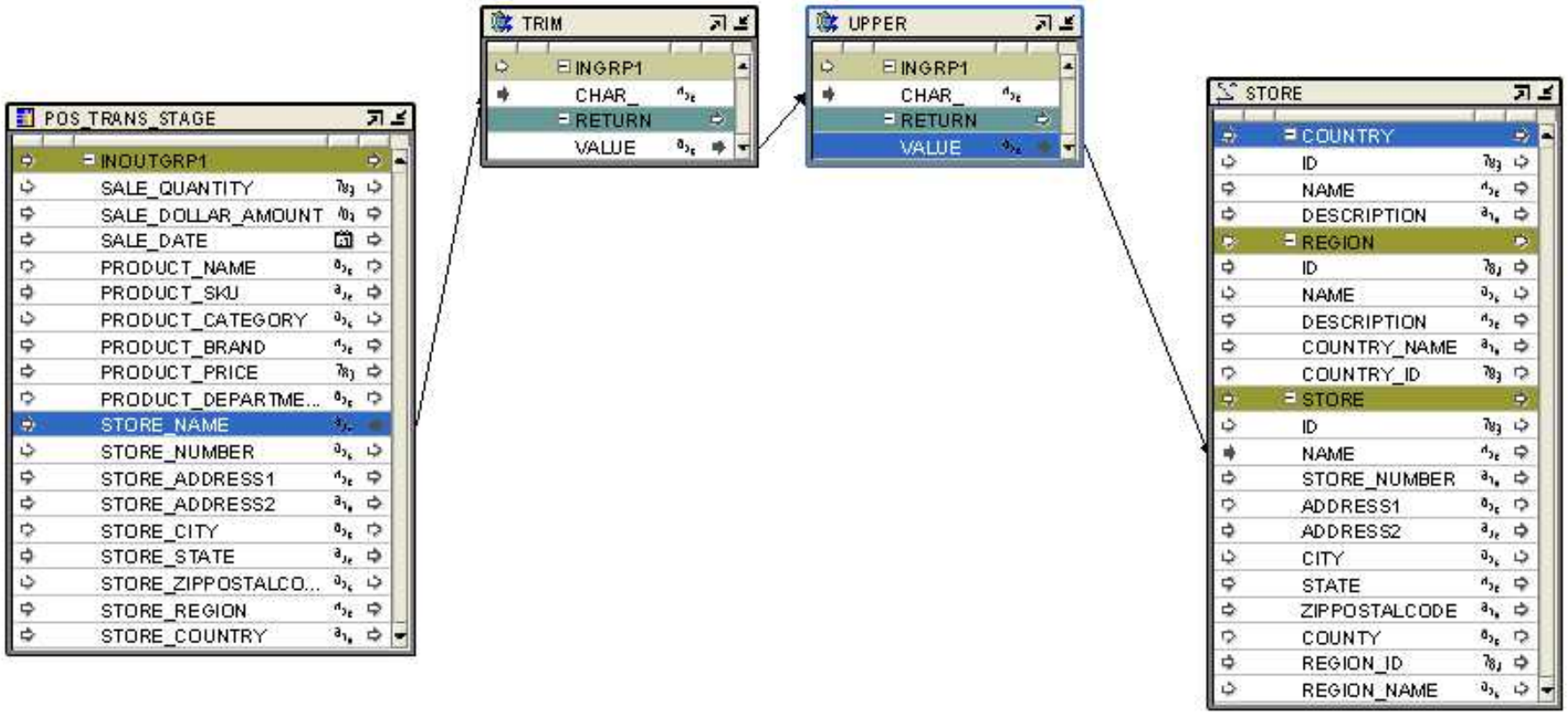




After dropping the Transformation Operator on the mapping, it will pop up a dialog box where we select the transformation we want to use.

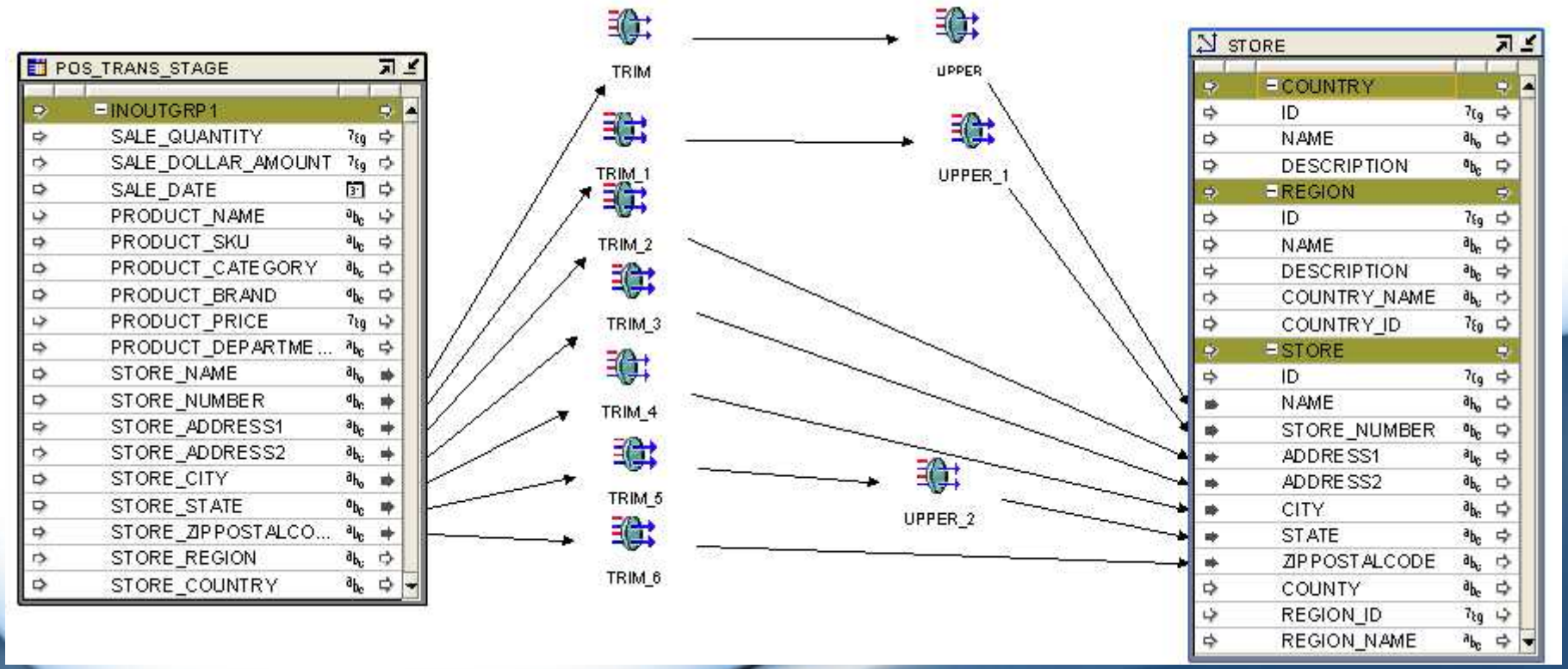

Select from existing repository objects and bind
f(x) SUBSTRB (IN UNSPECIFIED, IN NUMBER, IN NUMBER) return UNSP
f(x) SUBSTRC (IN UNSPECIFIED, IN NUMBER, IN NUMBER) return UNSP
f(x) TRANSLATE (IN UNSPECIFIED, IN UNSPECIFIED, IN UNSPECIFIED)
f(x) TRIM (IN VARCHAR2) return VARCHAR2
f(x) UPPER (IN UNSPECIFIED) return UNSPECIFIED
f(x) WB_IS_SPACE (IN VARCHAR2) return BOOLEAN
f(x) WB_LOOKUP_CHAR (IN VARCHAR2, IN VARCHAR2, IN VARCHA
f(x) WB_LOOKUP_CHAR (IN VARCHAR2, IN VARCHAR2, IN VARCHA


UPPER
INGRP1
CHAR_
RETURN
VALUE

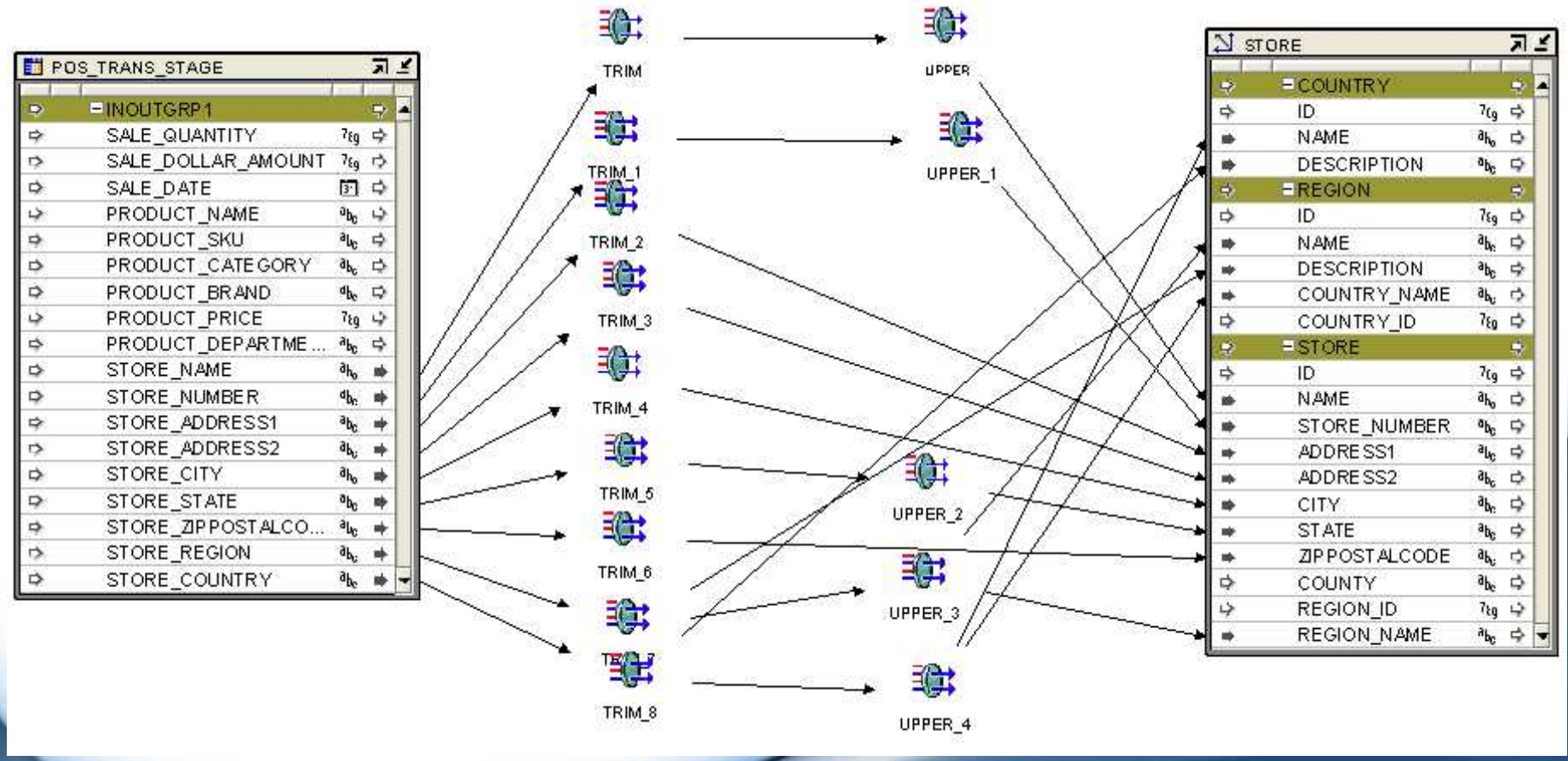

STORE
COUNTRY
ID
NAME
DESCRIPTION
REGION
ID
NAME
DESCRIPTION
COUNTRY_NAME
COUNTRY_ID
STORE
ID
NAME
STORE_NUMBER
ADDRESS1
ADDRESS2
CITY
STATE
ZIP_POSTALCODE
COUNTY
REGION_NAME
REGION_ID


POS_TRANS_STAGE
INOUTGRP1
SALE_QUANTITY
SALE_DOLLAR_AMOUNT
SALE_DATE
PRODUCT_NAME
PRODUCT_SKU
PRODUCT_CATEGORY
PRODUCT_BRAND
PRODUCT_PRICE
PRODUCT_DEPARTME...
STORE_NAME
STORE_NUMBER
STORE_ADDRESS1
STORE_ADDRESS2
STORE_CITY
STORE_STATE
STORE_ZIPPOSTALCO...
STORE_REGION
STORE_COUNTRY
TRIM
INGRP1
CHAR_
RETURN
VALUE
UPPER
INGRP1
CHAR_
RETURN
VALUE
STORE
COUNTRY
ID
NAME
DESCRIPTION
REGION
ID
NAME
DESCRIPTION
COUNTRY_NAME
COUNTRY_ID
STORE
ID
NAME
STORE_NUMBER
ADDRESS1
ADDRESS2
CITY
STATE
ZIPPOSTALCODE
COUNTY
REGION_ID
REGION_NAME

- After making all these connections, our mapping should now look similar to the following with all the Transformation Operators collapsed into their icon views:



# step2


POS_TRANS_STAGE
INOUTGRP1
SALE_QUANTITY
SALE_DOLLAR_AMOUNT
SALE_DATE
PRODUCT_NAME
PRODUCT_SKU
PRODUCT_CATEGORY
PRODUCT_BRAND
PRODUCT_PRICE
PRODUCT_DEPARTME...
STORE_NAME
STORE_NUMBER
STORE_ADDRESS1
STORE_ADDRESS2
STORE_CITY
STORE_STATE
STORE_ZIPPOSTALCO...
STORE_REGION
STORE_COUNTRY
TRIM
TRIM_1
TRIM_2
TRIM_3
TRIM_4
TRIM_5
TRIM_6
TRIM_8
UPPER
UPPER_1
UPPER_2
UPPER_3
UPPER_4
STORE
COUNTRY
ID
NAME
DESCRIPTION
REGION
ID
NAME
DESCRIPTION
COUNTRY_NAME
COUNTRY_ID
STORE
ID
NAME
STORE_NUMBER
ADDRESS1
ADDRESS2
CITY
STATE
ZIPPOSTALCODE
COUNTY
REGION_ID
REGION_NAME

- Key Lookup operators, as the name implies, are used for looking up information from other sources based on some key attribute(s) in a mapping.

# Using a Key Lookup operator

# Creating an external table

- an external table's data is stored in a flat file that is external to the database.
- External tables are created under the **Oracle | ACME_DWH | External Tables** node in the Design Center, so we'll right-click on it and select **New...** from the **pop-up menu.** This will launch the **ExternalTable wizard**
  - **Step 1.We'll name this external table COUNTIES**
  - **Step 2. select the file that contains the metadata for the external table (COUNTIES_CSV )**
  - **Step 3. will select the default location to use for this table . The drop-down menu on this screen will display the file locations that have been defined in the Design Center.We will select the ACME_FILES_LOCATION entry, which is for the files that exist for this project.**


Create External Table - Summary
External Table definition is complete
When you click Finish, the wizard will create the object with the following settings:
Name
COUNTIES
Columns
COUNTY_NAME VARCHAR2
ID INTEGER
STATE VARCHAR2
Metadata Source
File COUNTIES_CSV
Default Location
ACME_FILES_LOCATION
Help
< Back
Finish
Cancel

# Creating and loading a lookup table

- we have our source table defined for our new lookup table let's create a new mapping called COUNTIES_LOOKUP_MAP using the same method we've used previously
  - 1. Right-click on the Mappings node, select New…, enter COUNTIES_LOOKUP_MAP in the name field, and click on the OK button.
  - 2. In the Mapping Editor that pops up, let's drag an ExternalTable Operator from the Palette window onto the mapping.
  - 3. On the Add External Table Operator pop-up window that appears, our COUNTIES external table is visible. We will select that and click on the OK button to continue. This will drop an External Table Operator on our mapping that is bound to our COUNTIES external table.
  - 4. We need to get that data loaded into a regular table in the database, so next we'll drag a Table Operator onto the mapping.
  - 5. In the resulting AddTable Operator pop up that appears, we specify what table we want to add.


We'll click on OK and it will drop a Table Operator onto our mapping with no attributes defined in it.

- Let's drag a line from the output group (OUTGRP1) of our COUNTIES external table over to the input/output group (INOUTGRP1) of our new COUNTIES_LOOKUP table.
- This mapping is done. However, there is one more step we need to take to actually create the lookup table definition. Remember we created our table operator as an unbound operator, which means it's not associated with any database object.
- If we look in the ACME_DWH | Tables node, there is no table named COUNTIES_LOOKUP.

1. When we right-click on the unbounded operator, the pop-up menu has a menu selection called Create and Bind…. With this option we will create a new table object in the OWB Tables node and bind this operator to it.


2. The name is the same as what we gave to the operator.
3. The Create in: text field is to specify the module in which to create the new table under our project in the Design Center.
4. When we click on the OK button on this dialog box, a table is created in the Tables node and is bound to the operator.

- When we use the option to create a table in this manner, it creates a basic, no-frills table with no constraints defined on it. To add a primary key, we'll perform the following steps:

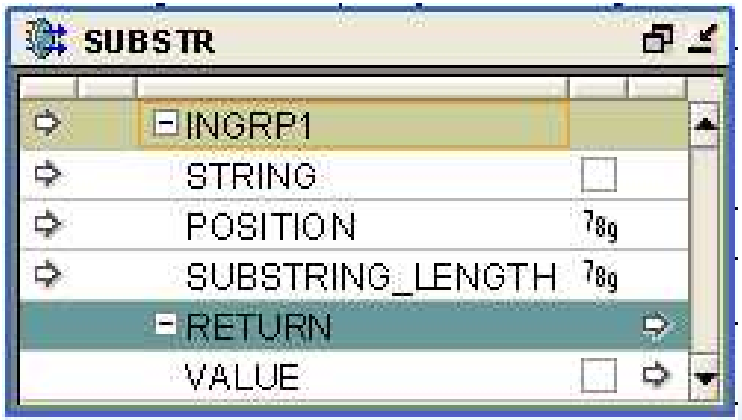
1. In the Design Center, open the COUNTIES_LOOKUP table in the Data Object Editor by double-clicking on it under the Tables node.
2. Click on the Constraints tab.
3. Click on the Add Constraint button.
4. Type PK_COUNTIES_LOOKUP (or any other naming convention we might choose) in the Name column.
5. In the Type column, click on the drop-down menu and select Primary Key.
6. Click on the Local Columns column, and then click on the Add Local Column button.
7. Click on the drop-down menu that appears and select the ID column.
8. Close the Data Object Editor.

# Retrieving the key to use for a Lookup Operator

- The county ID is only a portion of the entire STORE_NUMBER field, so we can't just use the STORE_NUMBER from input as the direct key to a Key Lookup Operator.
- actually a fixed known format, and the positions three through six of the number are actually the code for the location of the store in the county.
- The county ID is only a portion of the entire STORE_NUMBER field, so we can't just use the STORE_NUMBER from input as the direct key to a Key Lookup Operator. We will have to extract the ID number out of it and then convert it to a number before we can use it to look up the county.

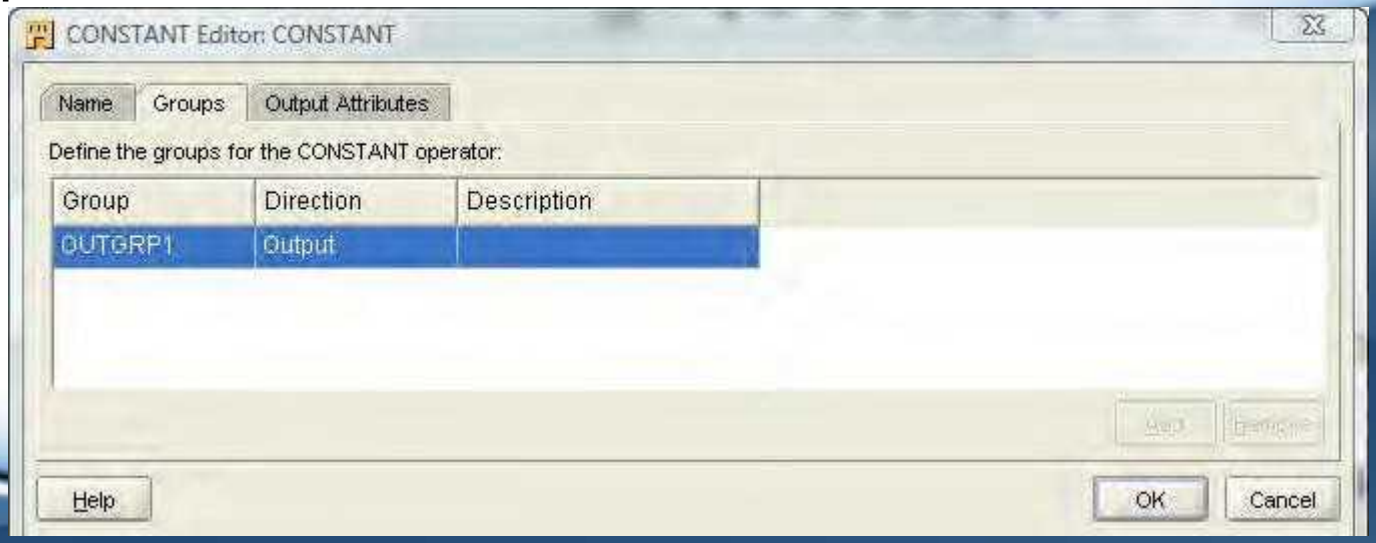
# Adding a SUBSTR Transformation Operator

- The Transformation Operators available to us in OWB include a SUBSTR (or substring) transformation that will do exactly what we need to extract the county ID value out of the STORE_NUMBER field.
- The SUBSTR transformation takes three parameters—the string we want to extract the substring from, a number indicating the start position of the substring within the string, and a number indicating the length of the substring to extract.
- drag a Transformation Operator onto the STORE_MAP mapping between the POS_TRANS_STAGE table and the STORE dimension below all the other Transformation Operators.


SUBSTR
INGRP1
STRING
POSITION
SUBSTRING_LENGTH
RETURN
VALUE

# Adding a Constant operator

- We'll click and drag a Constant operator onto the mapping to the left of the SUBSTR Transformation Operator.
- We can see that it has an output group called OUTGRP1 by default.
- We'll right-click on it and select Open Details... from the pop-up menu.



- Clicking on the **Output Attributes tab** we see that there are no attributes currently existing for this operator.
- This is where we will add our constants that we need for the **SUBSTR operator.**
- **OUTPUT1** We'll change the name to **position.**

# Adding a TO_NUMBER transformation

- The **SUBSTR** value is ready and we can use it to look up the **county ID,** but there's one more transformation we need to apply before we can use it to look up the county name.
- it needs to be converted into a number to match the data type of the ID field in the COUNTIES_LOOKUP table
- To do this, we will use the **TO_NUMBER() function.**
- let's drag a **Transformation Operator** onto our mapping to the right of the **SUBSTR operator** and select **TO_NUMBER** from the resulting pop up.

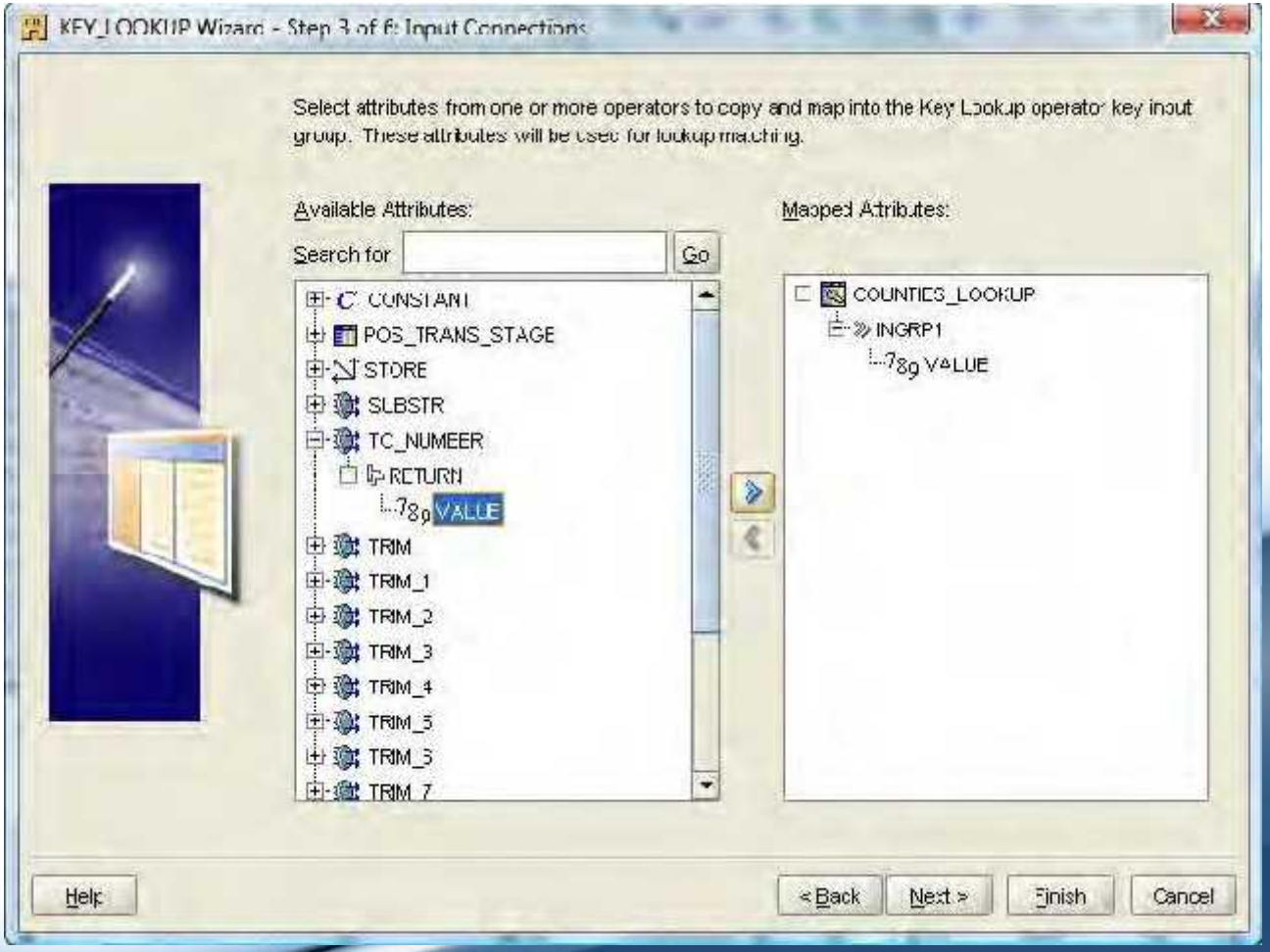
# Adding a Key Lookup operator

- After that little side trip to quickly create our lookup table and add a **SUBSTR operator** with a **TO_NUMBER transformation** to convert the result to a number.
- Let's drag a **Key Lookup** operator onto the mapping and drop it to the right of the **TO_NUMBER operator.**
- the **KEY_LOOKUPWizard is launched**

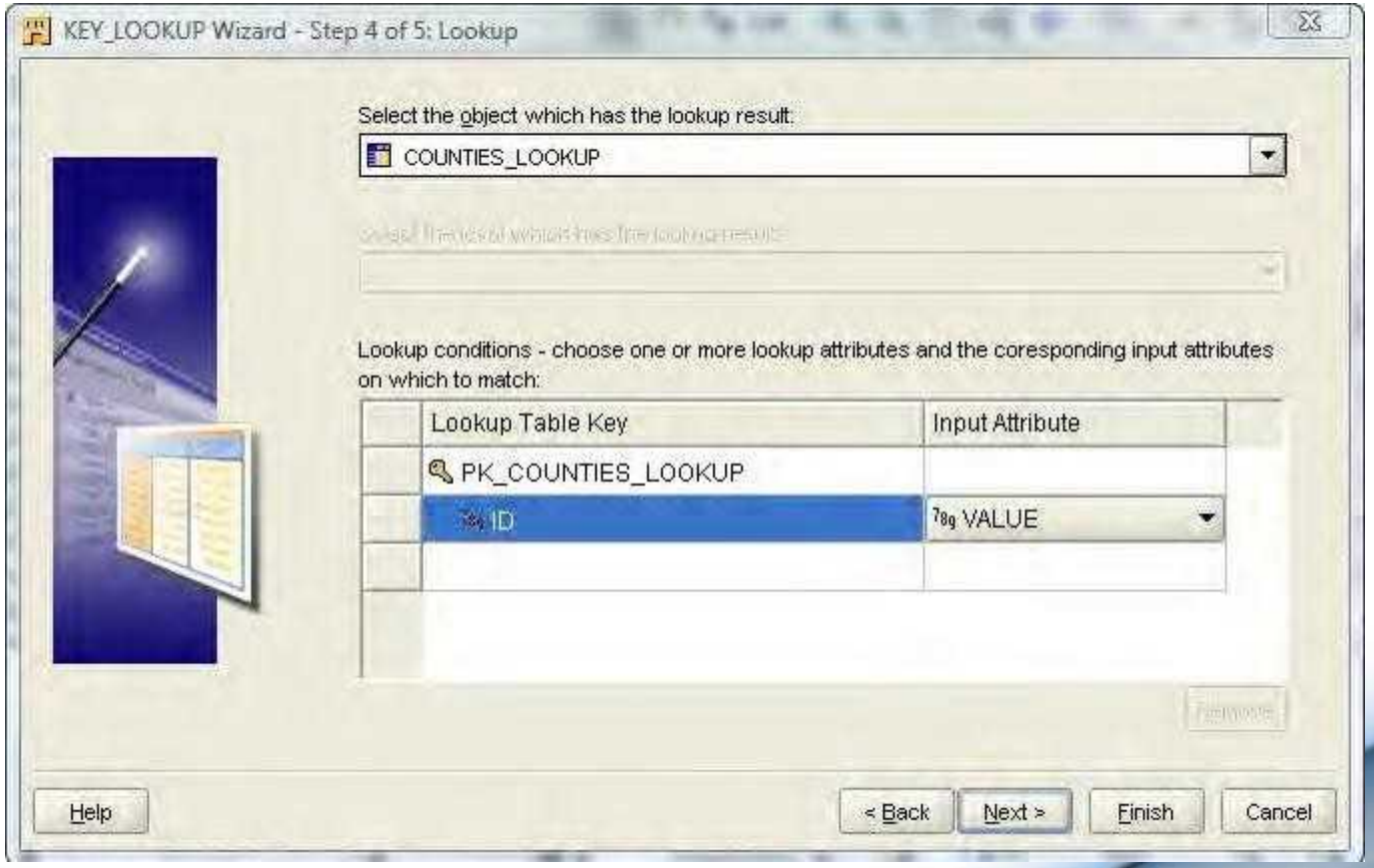

KEY_LOOKUP Wizard - Welcome
Welcome to the Key Lookup Wizard
This wizard guides you through the creation of a Key Lookup operator:
Step 1: Name
Step 2: Groups
Step 3: Input Attributes
Step 4: Lookup Condition
Step 5: Type 2 Dimension History Lookup Condition
Step 6: No-match results policy
Click Next to continue

1. **Rename it to COUNTIES_LOOKUP**
2. Key Lookup operators require one input and one output group (**INGRP1 and OUTGRP1** )
3. select one or more operators to map into the input group of the Key Lookup operator

# Step 3


KEY_LOOKUP Wizard - Step 3 of 6: Input Connections
Select attributes from one or more operators to copy and map into the Key Lookup operator key input group. These attributes will be used for lookup matching.
Available Attributes:
Search for
Go
CONSTANT
POS_TRANS_STAGE
STORE
SUBSTR
TC_NUMBER
RETURN
VALUE
TRIM
TRIM_1
TRIM_2
TRIM_3
TRIM_4
TRIM_5
TRIM_6
TRIM_7
Mapped Attributes:
COUNTIES_LOOKUP
INGRP1
VALUE
Help
< Back
Next >
Finish
Cancel

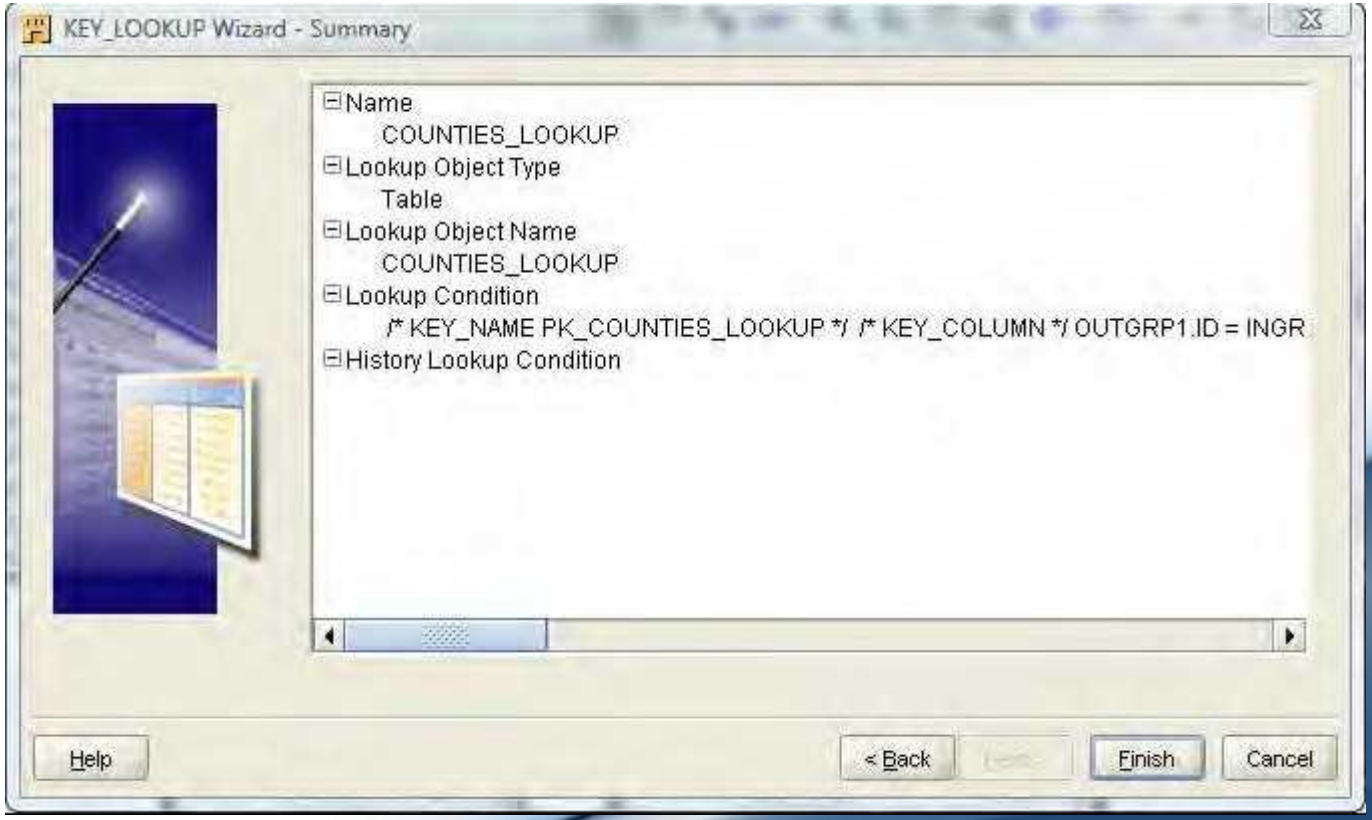
4. will specify in which object to look up the value.



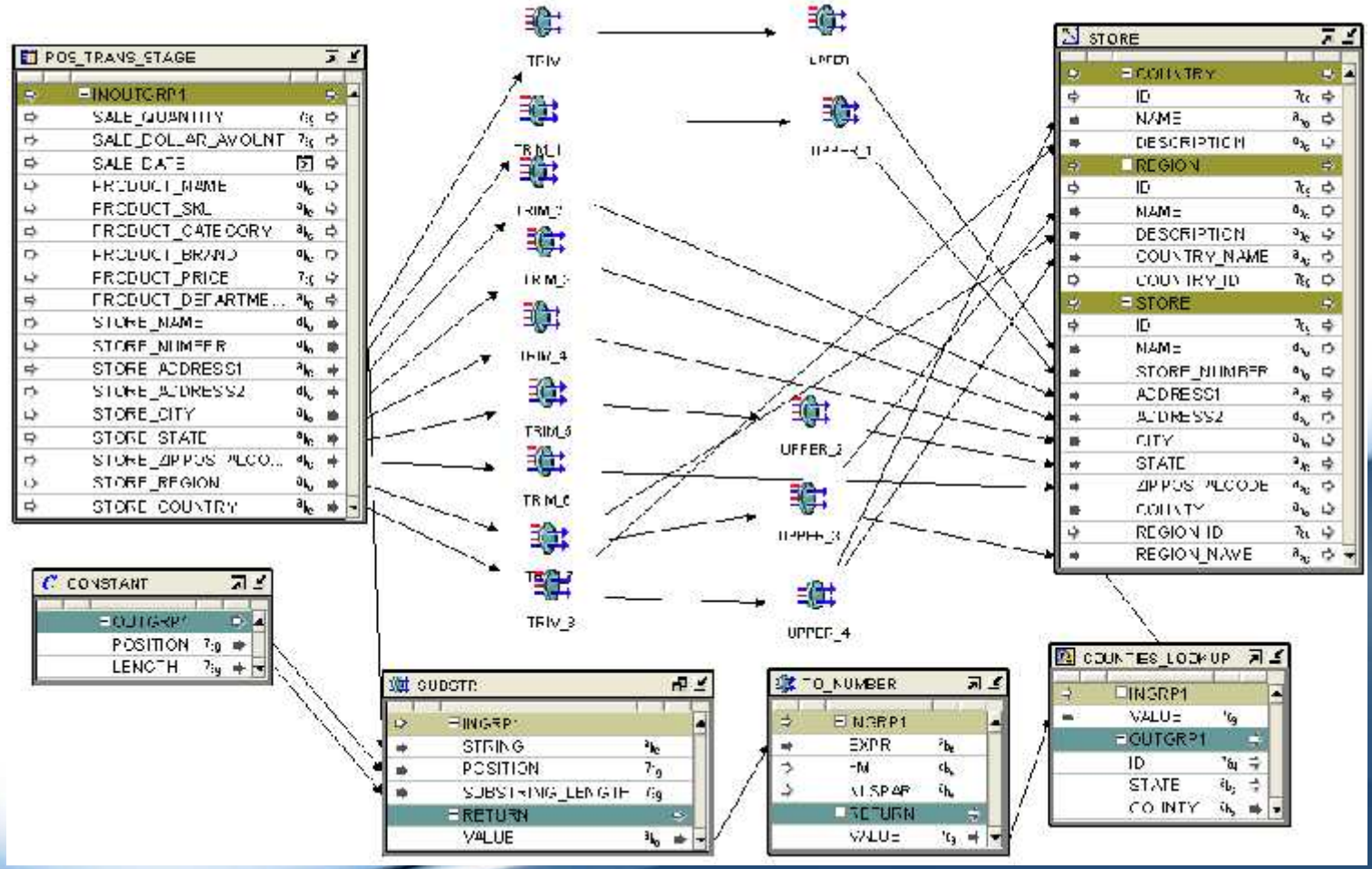

We will click on the Next button to proceed to the final step where we will specify what to return if no record is found in the lookup table.

So we'll click on NULL that currently appears as the default for the COUNTY_NAME value and type in 'UNKNOWN' in the box.

We will click on the Next button to proceed to the Summary screen

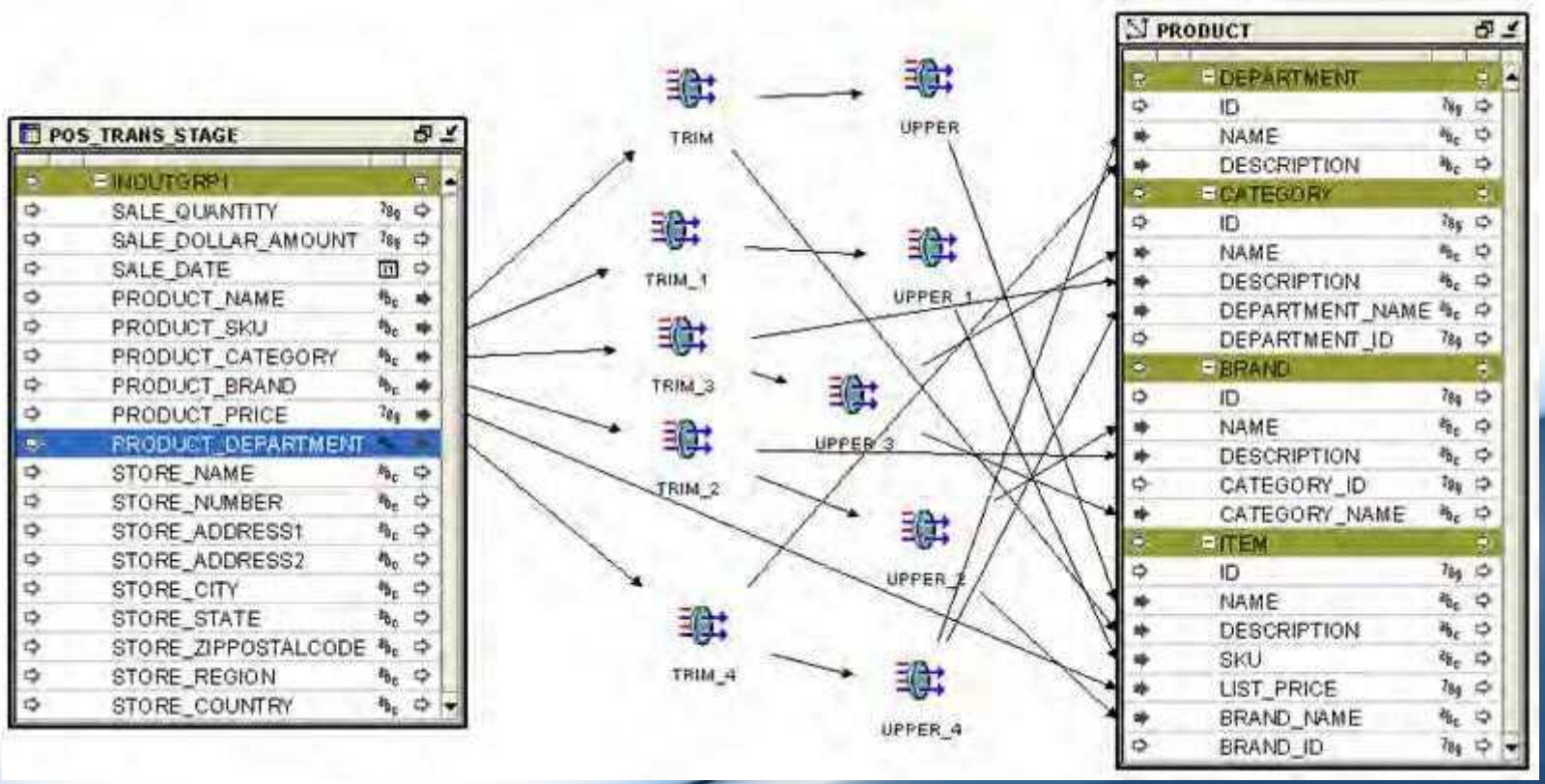

KEY_LOOKUP Wizard - Summary
Name
COUNTIES_LOOKUP
Lookup Object Type
Table
Lookup Object Name
COUNTIES_LOOKUP
Lookup Condition
/* KEY_NAME PK_COUNTIES_LOOKUP */ /* KEY_COLUMN */ OUTGRP1.ID = INGR
History Lookup Condition
Help
< Back
Finish
Cancel

# Now we have a completed mapping that will populate our STORE dimension



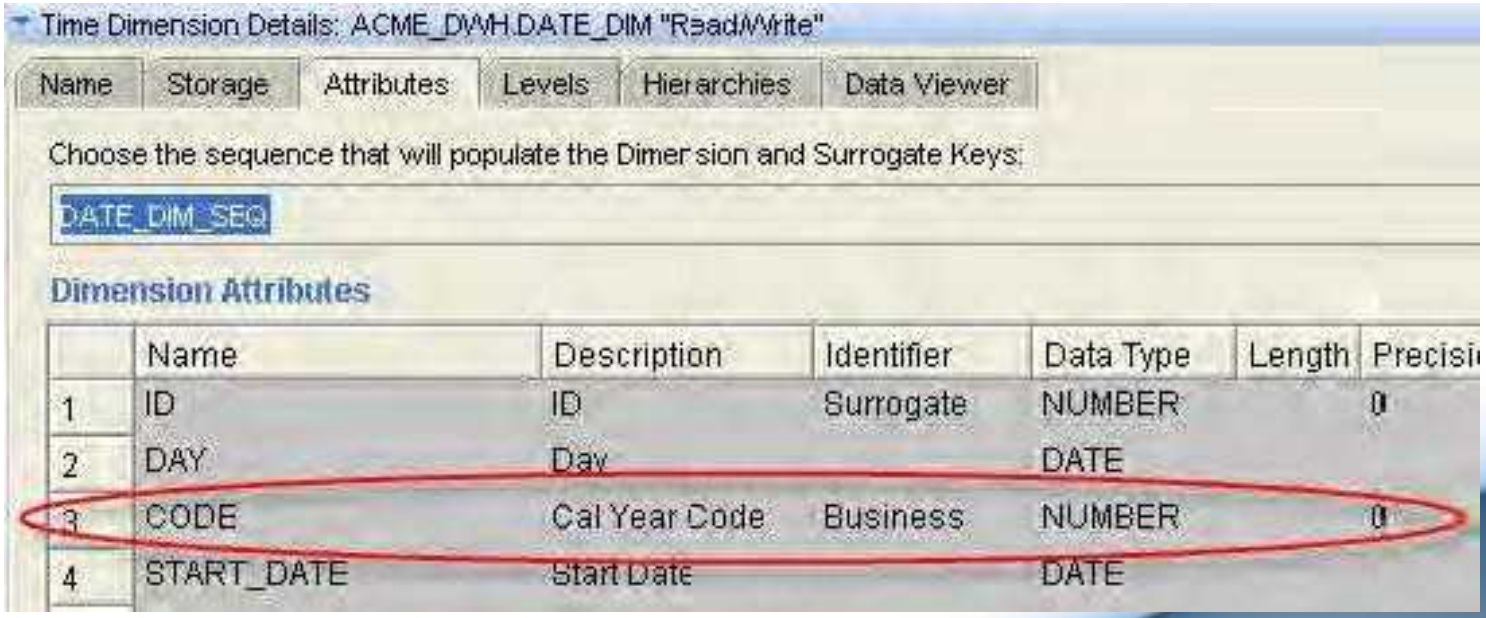
# PRODUCT mapping

- The mapping for the PRODUCT dimension will be similar to the STORE mapping

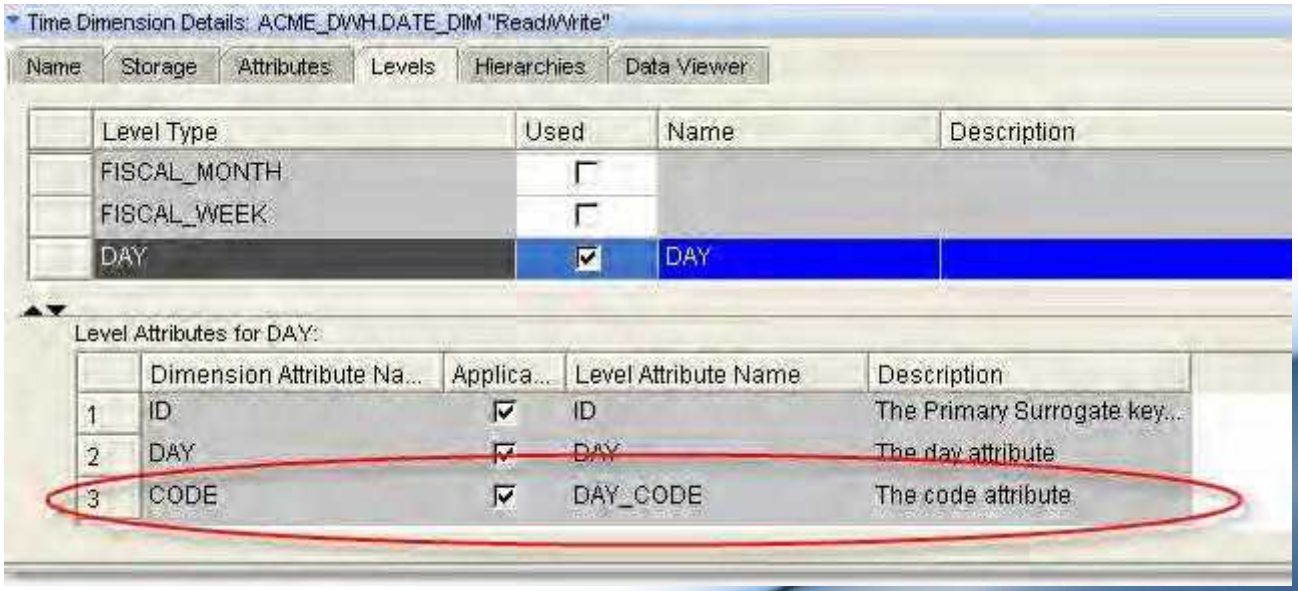


# SALES cube mapping

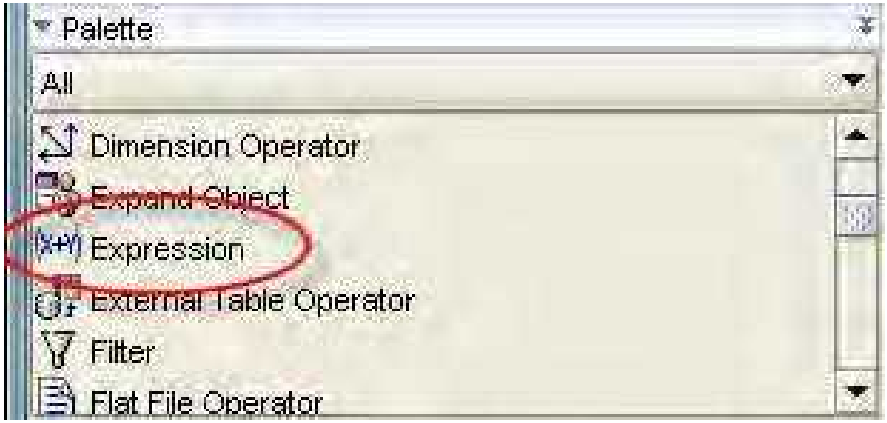
- drag each of these onto our mapping using Table Operator for the POS_TRANS_STAGE table and Cube Operator for the SALES cube.
- The POS_TRANS_STAGE table is very familiar to us as we have used it for the two dimensions, but the SALES cube is new. It looks slightly different than the dimensions we worked with earlier .
- For the STORE dimension, we identified NAME and STORE_NUMBER as the business identifiers.
- The third dimension we have defined for our cube is the DATE_DIM dimension.
- To open it in the Data Object Editor, navigate in the Design Center to **ACME_DWH | Dimensions |DATE_DIM and double-click to open it.**



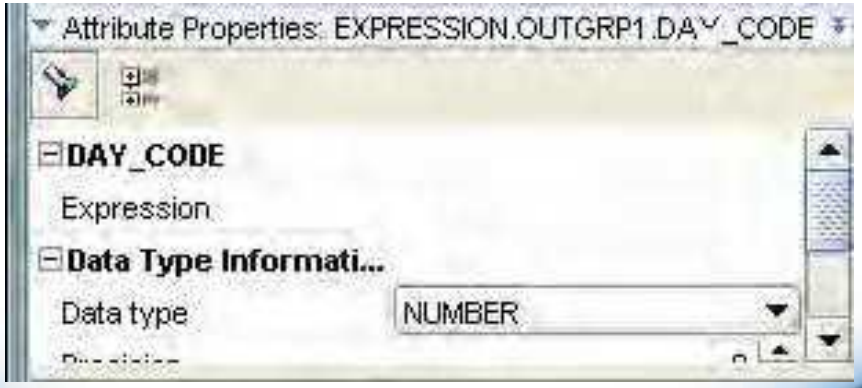

look at the Levels tab where we can inspect each level of the dimension and see what attributes were identified for each level. We can scroll down and click on the DAY level


Time Dimension Details: ACME_DWH.DATE_DIM "Read/Write"
Name
Storage
Attributes
Levels
Hierarchies
Data Viewer
Level Type
Used
Name
Description
FISCAL_MONTH
FISCAL_WEEK
DAY
DAY
Level Attributes for DAY:
Dimension Attribute Na...
Applica...
Level Attribute Name
Description
1
ID
ID
The Primary Surrogate key...
2
DAY
DAY
The day attribute
3
CODE
DAY_CODE
The code attribute

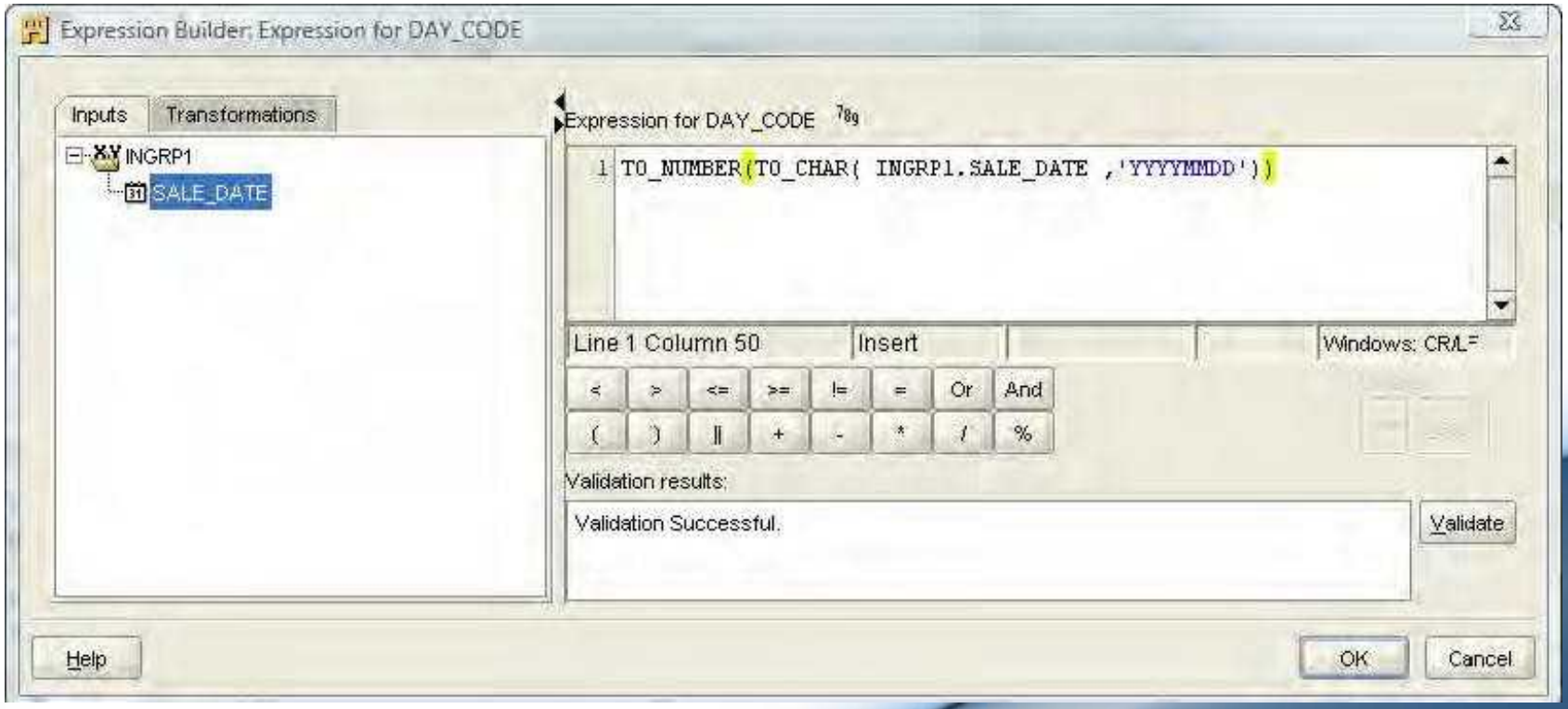
# Mapping an Expression operator(for date)

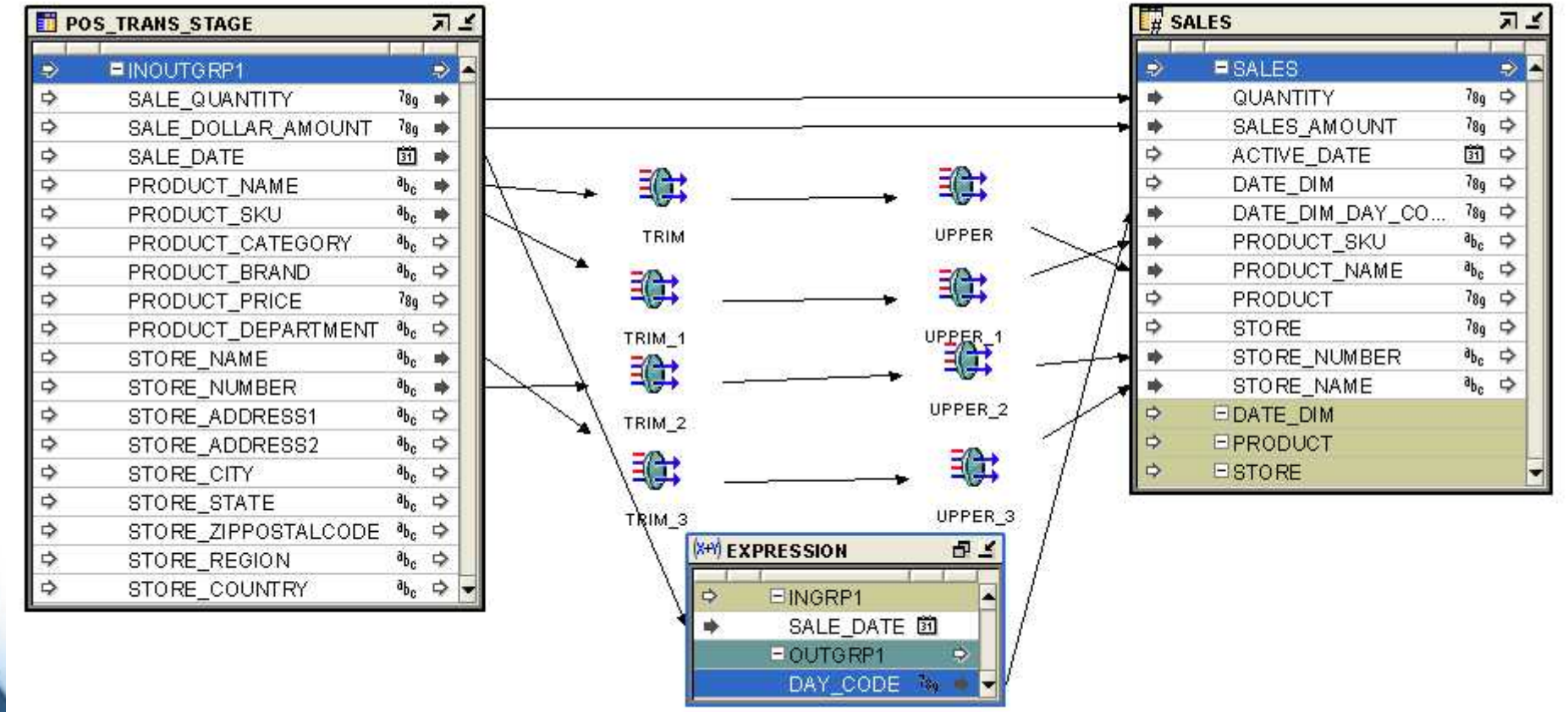







We can launch the Expression Builder by clicking on the button with three dots to the right of the Expression input field that appeared when we clicked on it. The Expression Builder dialog box appears and we can create our expression.


Expression Builder: Expression for DAY_CODE
Inputs
Transformations
INGRP1
SALE_DATE
Expression for DAY_CODE
1 TO_NUMBER(TO_CHAR( INGRP1.SALE_DATE ,'YYYYMMDD'))
Line 1 Column 50
Insert
Windows: CR/LF
< > <= >= != = Or And
( ) || + - * / %
Validation results:
Validation Successful.
Validate
Help
OK
Cancel


POS_TRANS_STAGE
INOUTGRP1
SALE_QUANTITY
SALE_DOLLAR_AMOUNT
SALE_DATE
PRODUCT_NAME
PRODUCT_SKU
PRODUCT_CATEGORY
PRODUCT_BRAND
PRODUCT_PRICE
PRODUCT_DEPARTMENT
STORE_NAME
STORE_NUMBER
STORE_ADDRESS1
STORE_ADDRESS2
STORE_CITY
STORE_STATE
STORE_ZIPPOSTALCODE
STORE_REGION
STORE_COUNTRY
TRIM
TRIM_1
TRIM_2
TRIM_3
UPPER
UPPER_1
UPPER_2
UPPER_3
EXPRESSION
INGRP1
SALE_DATE
OUTGRP1
DAY_CODE
SALES
SALES
QUANTITY
SALES_AMOUNT
ACTIVE_DATE
DATE_DIM
DATE_DIM_DAY_CO...
PRODUCT_SKU
PRODUCT_NAME
PRODUCT
STORE
STORE_NUMBER
STORE_NAME
DATE_DIM
PRODUCT
STORE

# Features and benefits of OWB

- By providing us the ROLAP option, the OracleWarehouse Builder opens to us the design features of cubes and dimensions even though we'll be implementing them relationally with tables in our database.
- The **Expression Builder provides us a powerful tool to use for interactively building expressions.**
- Support for **Slowly Changing Dimensions is built in with the Enterprise ETL option.**

# DATAWAREHOUSING

UNIT 5

END OF CHAPTER 1