Unit 4.2

DATAWAREHOUSING

UNIT 4

CHAPTER 2

1. Designing and building an ETL mapping

- Designing our staging area, Designing the staging area contents, Building the staging area table with the Data Object Editor, Designing our mapping, Review of the Mapping Editor, Creating a mapping.

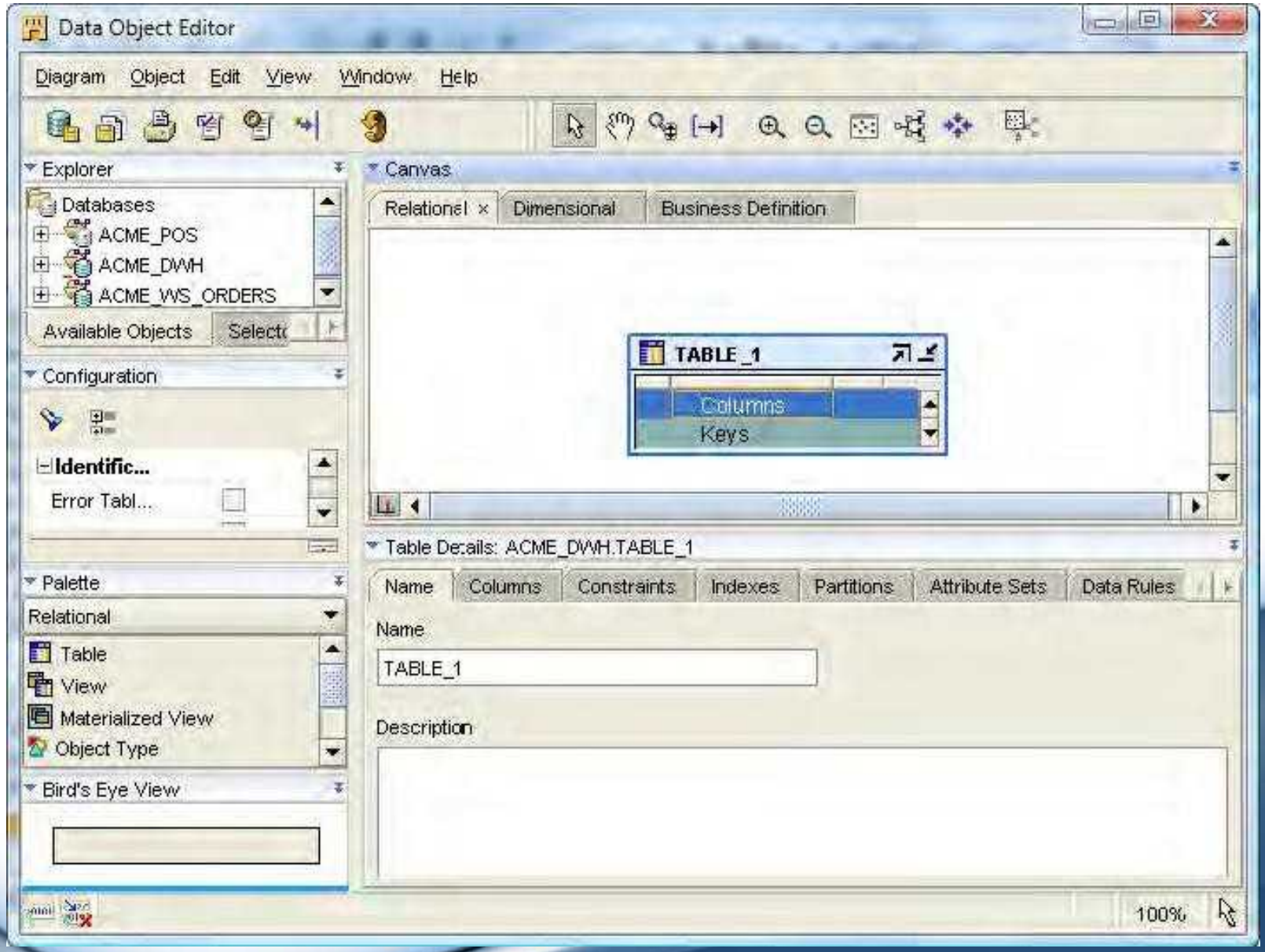
Designing the staging area contents

- The dimensional objects in our target are
  - Sales
    - Quantity
    - Sales amount
  - Date
    - Date of sale
  - Product
    - SKU
    - Name
    - List price
    - Department
    - Category
    - Brand
  - Store
    - Name
    - Number
    - Address1
    - Address2
    - City
    - State
    - Zip postal code
    - Country
    - Region

- We know the data elements we're going to need. Now let's put together a structure in our database that we'll use to stage the data prior to actually loading it into the target.
- Staging areas can be in the same database as the target, or in a different database, depending on various factors such as size and space issues, and availability of databases.
- For our purpose, we'll create a staging area as a single table in our target database schema for simplicity and will use the Warehouse Builder's **Data Object Editor to manually create the table.**

Building the staging area table with the Data Object Editor

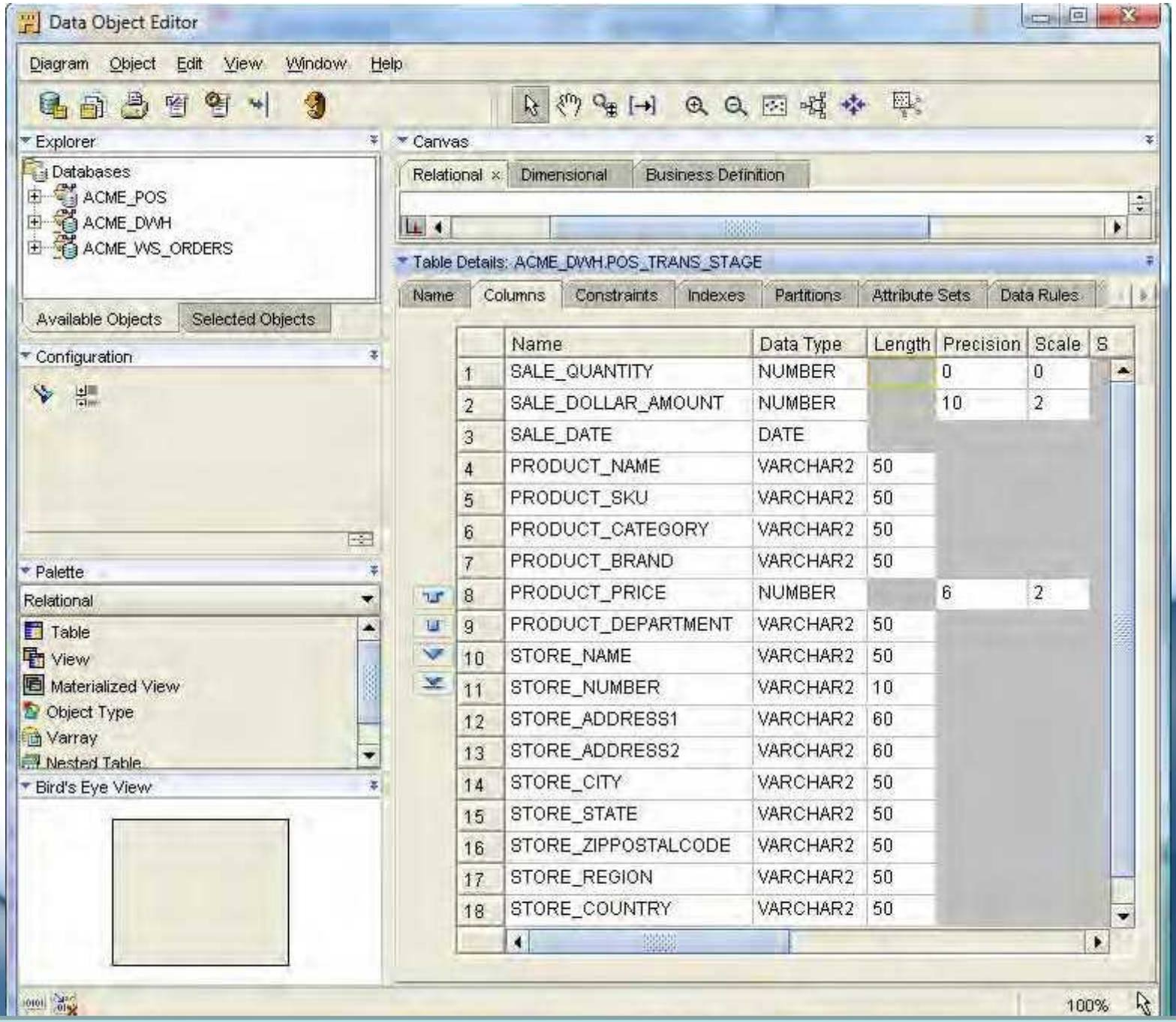
1. Navigate to the **Databases | Oracle | ACME_DWH module.** We will create our staging table under the Tables node, so let's right-click on that node and **select New**… from the pop-up menu.
2. Upon selecting **New...,** we are presented with the Data Object Editor screen
3. The first tab is the **Name tab** where we'll give our new table a name. Let's call it POS_TRANS_STAGE for Point-of-Sale transaction staging table. We'll just enter the name into the Name field, replacing the default TABLE_1


Data Object Editor
Diagram
Object
Edit
View
Window
Help
Explorer
Databases
ACME_POS
ACME_DWH
ACME_WS_ORDERS
Available Objects
Select
Configuration
Identific...
Error Tabl...
Palette
Relational
Table
View
Materialized View
Object Type
Bird's Eye View
Canvas
Relational
Dimensional
Business Definition
TABLE_1
Columns
Keys
Table Details: ACME_DWH.TABLE_1
Name
Columns
Constraints
Indexes
Partitions
Attribute Sets
Data Rules
Name
TABLE_1
Description
100%

4. click on the **Columns tab** next and enter the information that describes the columns of our new table.

- One key point to keep in mind here is that we want to make sure the sizes and types of the fields will match the fields we want to pull the data from.
- The following will then be the column names, types, and sizes we'll use for our staging table

1. SALE_QUANTITY NUMBER(0,0)
2. SALE_DOLLAR_AMOUNT NUMBER(10,2)
3. SALE_DATE DATE
4. PRODUCT_NAMEVARCHAR2(50)
5. PRODUCT_SKUVARCHAR2(50)
6. PRODUCT_CATEGORYVARCHAR2(50)
7. PRODUCT_BRANDVARCHAR2(50)
8. PRODUCT_PRICE NUMBER(6,2)
9. PRODUCT_DEPARTMENTVARCHAR2(50)
10. STORE_NAMEVARCHAR2(50)
11. STORE_NUMBERVARCHAR2(10)
12. STORE_ADDRESS1VARCHAR2(60)
13. STORE_ADDRESS2VARCHAR2(60)
14. STORE_CITYVARCHAR2(50)
15. STORE_STATEVARCHAR2(50)
16. STORE_ZIPPOSTALCODEVARCHAR2(50)
17. STORE_REGIONVARCHAR2(50)
18. STORE_COUNTRYVARCHAR2(50)


Data Object Editor
Diagram Object Edit View Window Help
Explorer
Databases
ACME_POS
ACME_DWH
ACME_WS_ORDERS
Available Objects
Selected Objects
Configuration
Palette
Relational
Table
View
Materialized View
Object Type
Varray
Nested Table
Bird's Eye View
Canvas
Relational
Dimensional
Business Definition
Table Details: ACME_DWH.POS_TRANS_STAGE
Name
Columns
Constraints
Indexes
Partitions
Attribute Sets
Data Rules
Name
Data Type
Length
Precision
Scale
1 SALE_QUANTITY NUMBER 0 0
2 SALE_DOLLAR_AMOUNT NUMBER 10 2
3 SALE_DATE DATE
4 PRODUCT_NAME VARCHAR2 50
5 PRODUCT_SKU VARCHAR2 50
6 PRODUCT_CATEGORY VARCHAR2 50
7 PRODUCT_BRAND VARCHAR2 50
8 PRODUCT_PRICE NUMBER 6 2
9 PRODUCT_DEPARTMENT VARCHAR2 50
10 STORE_NAME VARCHAR2 50
11 STORE_NUMBER VARCHAR2 10
12 STORE_ADDRESS1 VARCHAR2 60
13 STORE_ADDRESS2 VARCHAR2 60
14 STORE_CITY VARCHAR2 50
15 STORE_STATE VARCHAR2 50
16 STORE_ZIPPOSTALCODE VARCHAR2 50
17 STORE_REGION VARCHAR2 50
18 STORE_COUNTRY VARCHAR2 50
100%

- There are three groupings of data elements, which correspond to the three-dimensional object we created—our Sales cube and two dimensions, Product and Store.
  - The second thing to note is that these data elements aren't all going to come from a single table in the source.
  - For instance, the Store dimension has a STORE_REGION and STORE_COUNTRY column, but this information is found in the REGIONS table in the ACME_POS source database.
  - This means we are going to have to join this table with the STORES table if we want to be able to extract these two columns.

5. save our work using the *Ctrl+S keys*

other tabs in Data Object Editor for a table

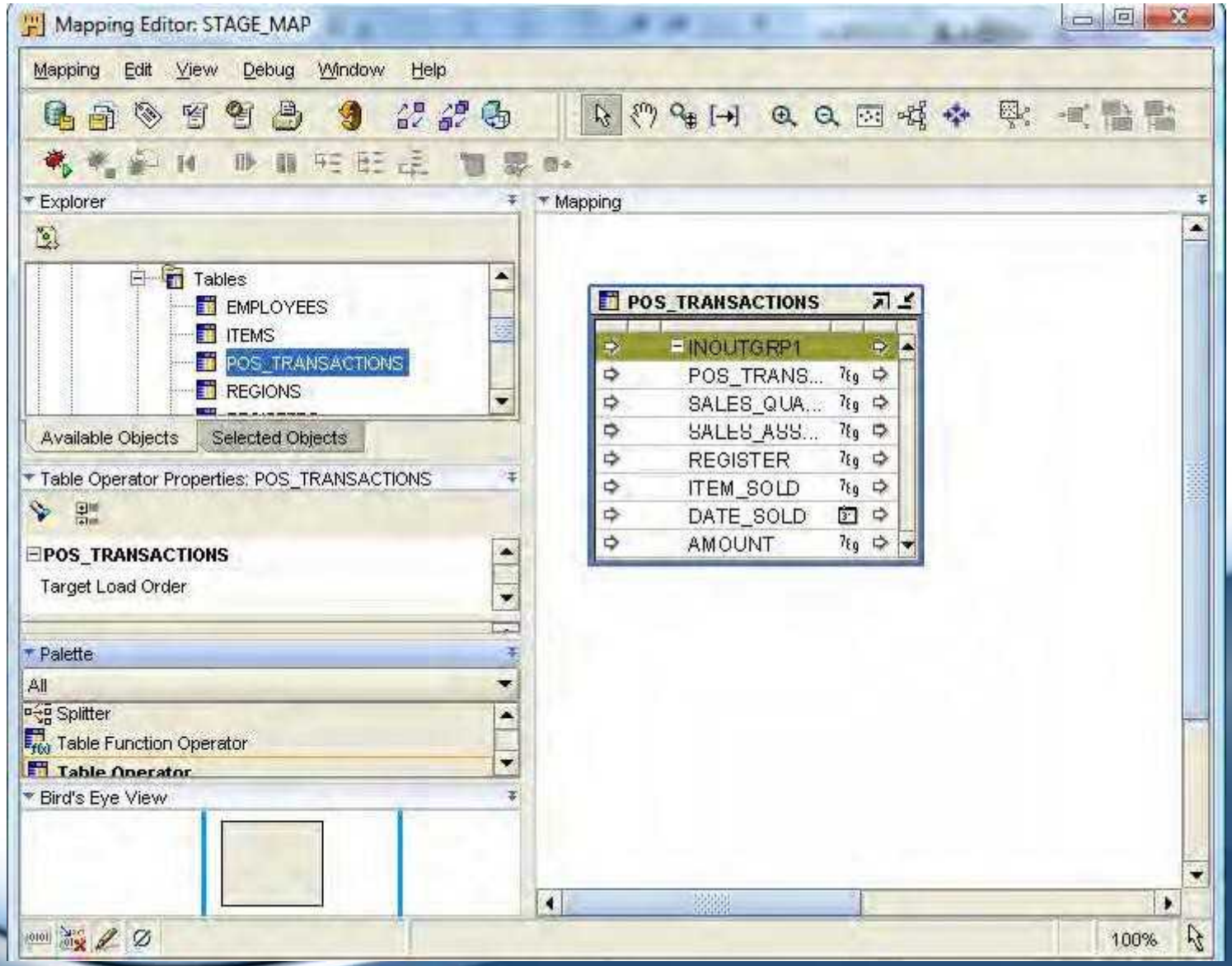
- **Constraints**
  - **Check constraint**—a constraint on a particular column that indicates the acceptable values that can be stored in the column.
  - **Foreign key**—a constraint on a column that indicates a record must exist in the referenced table for the value stored in this column.
  - **Primary key**—a constraint that indicates the column(s) that make up the unique information that identifies one and only one record in the table.
  - **Unique key**—a constraint that specifies the column(s) value combination(s) cannot be duplicated by any other row in the table.

- **Indexes**
  - An index can greatly facilitate rapid access to a particular record. It is generally useful for permanent tables that will be repeatedly accessed in a random manner by certain known columns of data in the table.
- **Partitions**
  - A partition is a way of breaking down the data stored in a table into subsets that are stored separately.This can greatly speed up data access for retrieving random records .
- **Attribute Sets**
  - An **Attribute Set** is a way to group attributes of an object in an order that we can specify when we create an attribute set.
- **Data Rules**
  - A **data rule** can be specified in the Warehouse Builder to enforce rules for data values or relationships between tables. It is used for ensuring that only high-quality data is loaded into the warehouse.
- **DataViewer**
  - for viewing cubes and dimensions
  - useful when we've actually loaded data to this staging table to view it and verify that we got the results we expected.

Designing our mapping

- **Review of the Mapping Editor**
  - In **Design Center, navigate to the ACME_DW_PROJECT | Databases | Oracle | ACME_DWH | Mappings node** if it is not already showing**. Right-click on it and select New..**
  - We'll name this mapping **STAGE_MAP** to reflect what it is being used for, and click on the OK button.

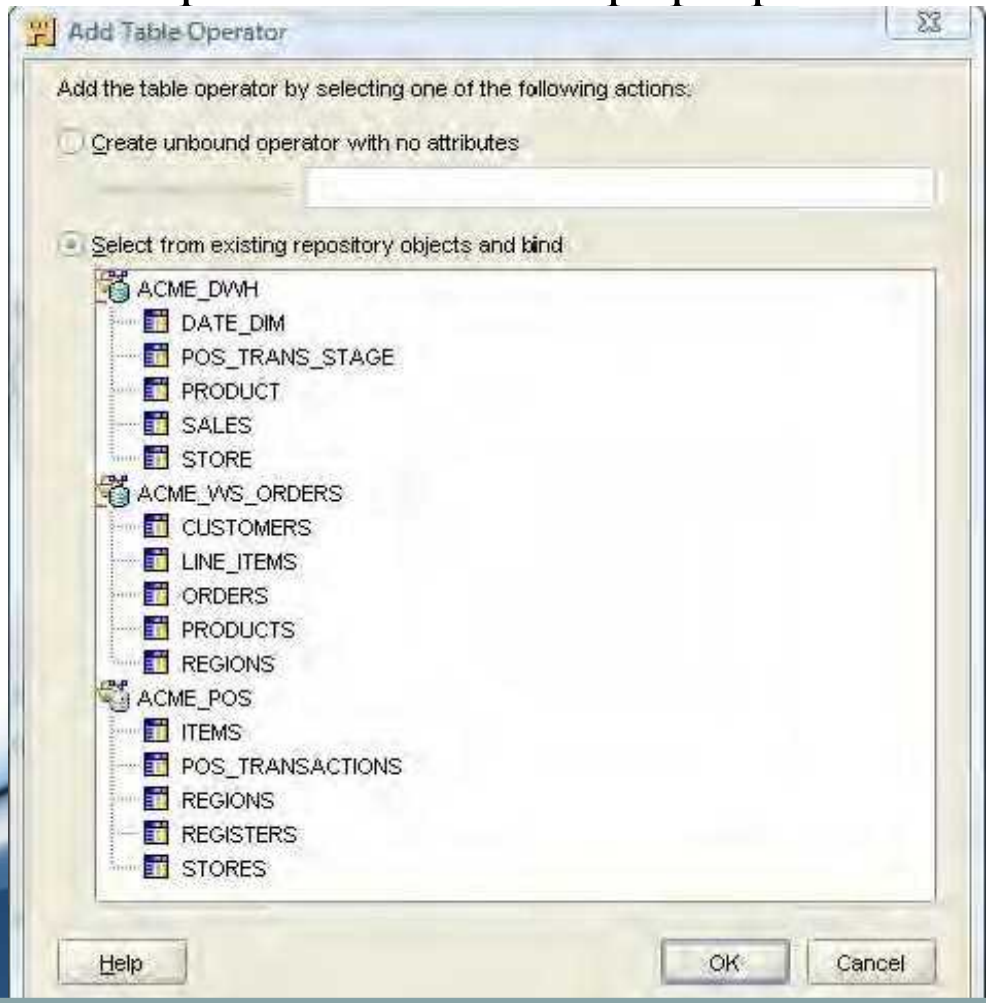
- **Creating a mapping**
- In designing any mapping in OWB, there will be a source(s) that we pull from, a target(s) that we will load data into, and several operators in between depending on how much manipulation of data we need to do between source and target. The layout will begin with sources on the left and proceed to the final targets
- **Adding source tables**
  - our ACME_POS source database, we know that data is stored in the POS_Transactions table. Therefore, we'll start our mapping by including this table.
  - In the **Explorer window,** we will use the **Available Objects** tab to find the table that we want to include in our mapping.
  - **POS_TRANSACTIONS** table is defined under the ACME_POS module**. (**So let's navigate to the **Databases | Non-Oracle | ODBC | ACME_POS node in the Available Objects tab to find the POS_TRANSACTIONS table entry.)**
  - Click and hold the left mouse button on **POS_TRANSACTIONS,** drag it over to the Mapping window


Mapping Editor: STAGE_MAP
Mapping Edit View Debug Window Help
Explorer
Tables
EMPLOYEES
ITEMS
POS_TRANSACTIONS
REGIONS
Available Objects
Selected Objects
Table Operator Properties: POS_TRANSACTIONS
POS_TRANSACTIONS
Target Load Order
Palette
All
Splitter
Table Function Operator
Bird's Eye View
Mapping
POS_TRANSACTIONS
INOUTGRP1
POS_TRANS...
SALES_QUA...
SALES_ASS...
REGISTER
ITEM_SOLD
DATE_SOLD
AMOUNT
100%

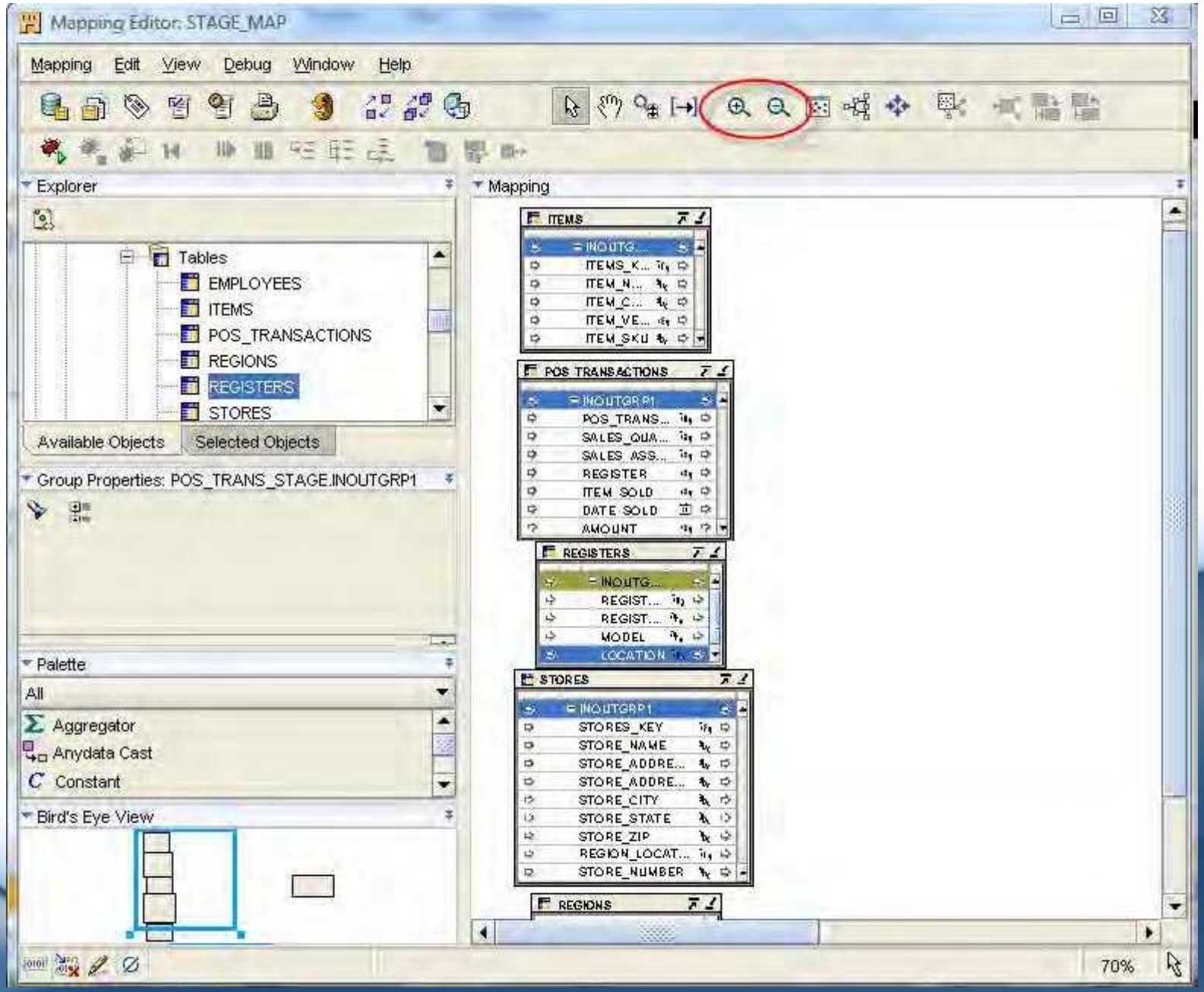
- The objects that make up a mapping are called **operators .**
- Click and drag the **Table Operator** from the **Palette** window onto the **Mapping** window. As soon as we drag it onto the Mapping window, we are presented with a pop up

With unbound operators, you can actually use the **Mapping Editor to create a data object.**

we're going to select the ITEMS table under the ACME_POS entry in the list of table names that the pop-up window presents to us. We will click on the OK button to include the ITEMS table operator in our mapping.



- source tables that we're going to need into our mapping—the REGISTERS, STORES, and REGIONS tables.
- why do we need the REGISTERS table?
  - POS_TRANSACTIONS table only had a foreign key column for the register—not the store or region. This information was kept in separate tables with a foreign key to the store, which is stored in the REGISTERS table. So, if we hope to be able to retrieve the store and region information out of the source database, we're going to need the REGISTERS table to get there.
  - See next screen
    - the ITEMS operator on top, then POS_TRANSACTIONS, then the REGISTERS operator next, then the STORES operator, and then the REGIONS operator at the bottom.


Mapping Editor: STAGE_MAP
Mapping Edit View Debug Window Help
Explorer
Tables
EMPLOYEES
ITEMS
POS_TRANSACTIONS
REGIONS
REGISTERS
STORES
Available Objects
Selected Objects
Group Properties: POS_TRANS_STAGE.INOUTGRP1
Palette
All
Aggregator
Anydata Cast
Constant
Bird's Eye View
Mapping
ITEMS
INOUTG...
ITEMS_K...
ITEM_N...
ITEM_C...
ITEM_VE...
ITEM_SKU
POS_TRANSACTIONS
INOUTGRP1
POS_TRANS...
SALES_QUA...
SALES_ASS...
REGISTER
ITEM_SOLD
DATE_SOLD
AMOUNT
REGISTERS
INOUTG...
REGIST...
REGIST...
MODEL
LOCATION
STORES
INOUTGRP1
STORES_KEY
STORE_NAME
STORE_ADDRE...
STORE_ADDRE...
STORE_CITY
STORE_STATE
STORE_ZIP
REGION_LOCAT...
STORE_NUMBER
REGIONS
70%

Adding a target table

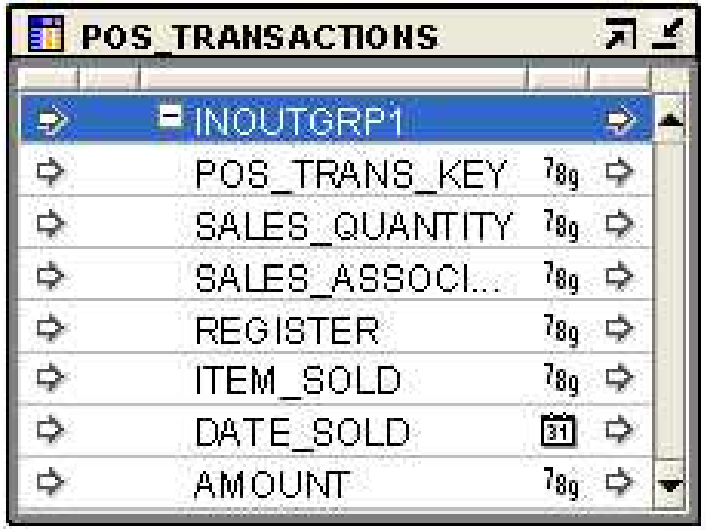
- As this is a staging-related mapping, we're going to be loading our staging table and so that will become our target.
- Let's find the **POS_TRANS_STAGE** table in the Explorer window.
- We'll navigate to **Databases | Oracle | ACME_DWH | Tables | POS_TRANS_STAGE** in the **Explorer,** and click and drag the **POS_TRANS_STAGE** table to the righthand side of our source tables in the **Mapping** window.

Connecting source to target

- operators that can be used for data flows, and one of them was a **Joiner operator.**
- joiner is exactly what we need because we have to take multiple source tables and combine (or join) them into one record in the target.
- We also discussed an **Aggregator** operator that can be used to aggregate data.
- drag this operator into the **Mapping** window, and drop it between the sources and target.

Joiner operator attribute groups

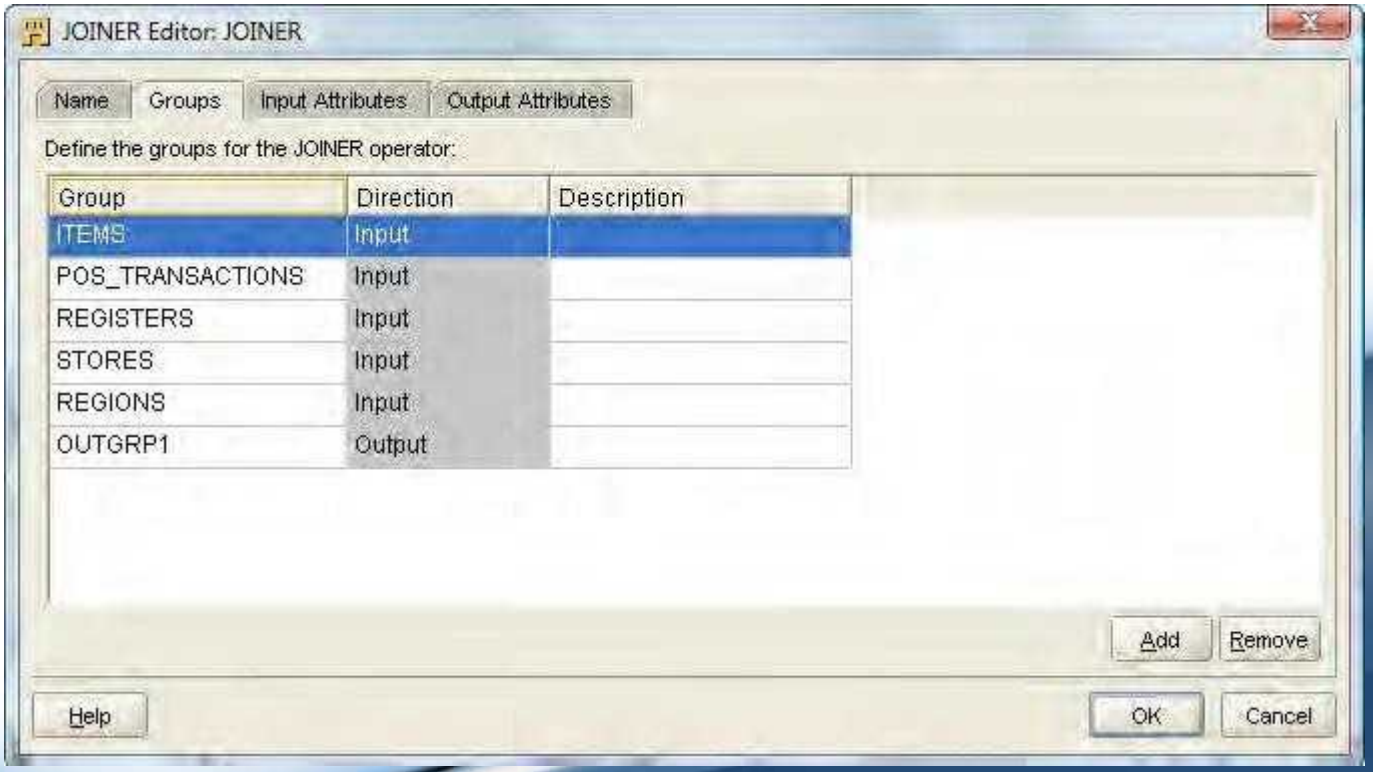
- In our table operators we can see that there is only one group as we mentioned, called **INOUTGRP1**.The following screenshot is an example of using the **POS_TRANSACTIONS** table operator**:**



If we look at the **Joiner operator** we just dropped into our mapping, we can see that there are no attributes defined in any of the three groups.

- We have five source tables to join together, but this Joiner operator has only two input groups.
- Have no fear; a Joiner can have more than two input groups. We have to edit this Joiner to add three more input groups.
- To edit it, right-click on the header of the box and select **Open Details... to open the Joiner Editor.**
- This dialog box will allow us to edit the number of groups as well as change the group names if we want something different from **INGRP1 and INGRP2.**
- With the **Joiner Editor open**, let's click on the Groups tab. We'll click three times on the Add button in the lower-right corner to add three more groups.

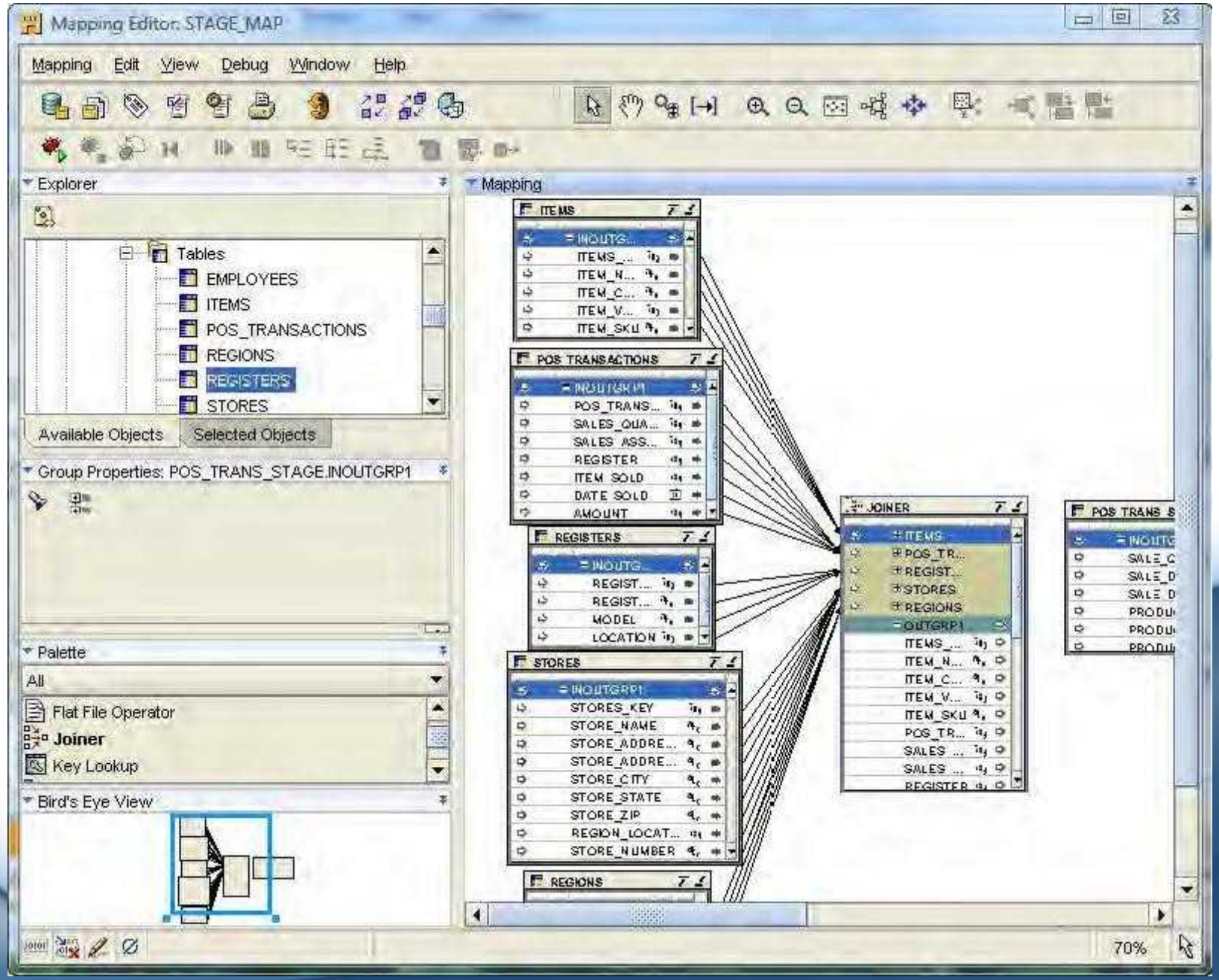
- INGRP1 to ITEMS
- INGRP2 to POS_TRANSACTIONS
- INGRP3 to REGISTERS
- INGRP4 to STORES
- INGRP5 to REGIONS



Connecting operators to the Joiner

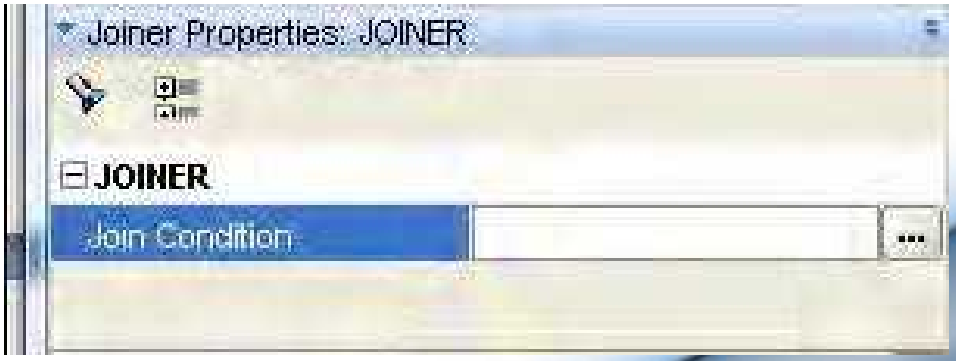
- click and drag **INOUTGRP1 of the ITEMS table operator onto the ITEMS group of the JOINER.**
- **Do the same for others tables to joiner**

- if we now scroll down our **JOINER** operator to where the **OUTGRP1** is visible, we can see that it automatically added attributes to the output group corresponding to each of the input group attributes.

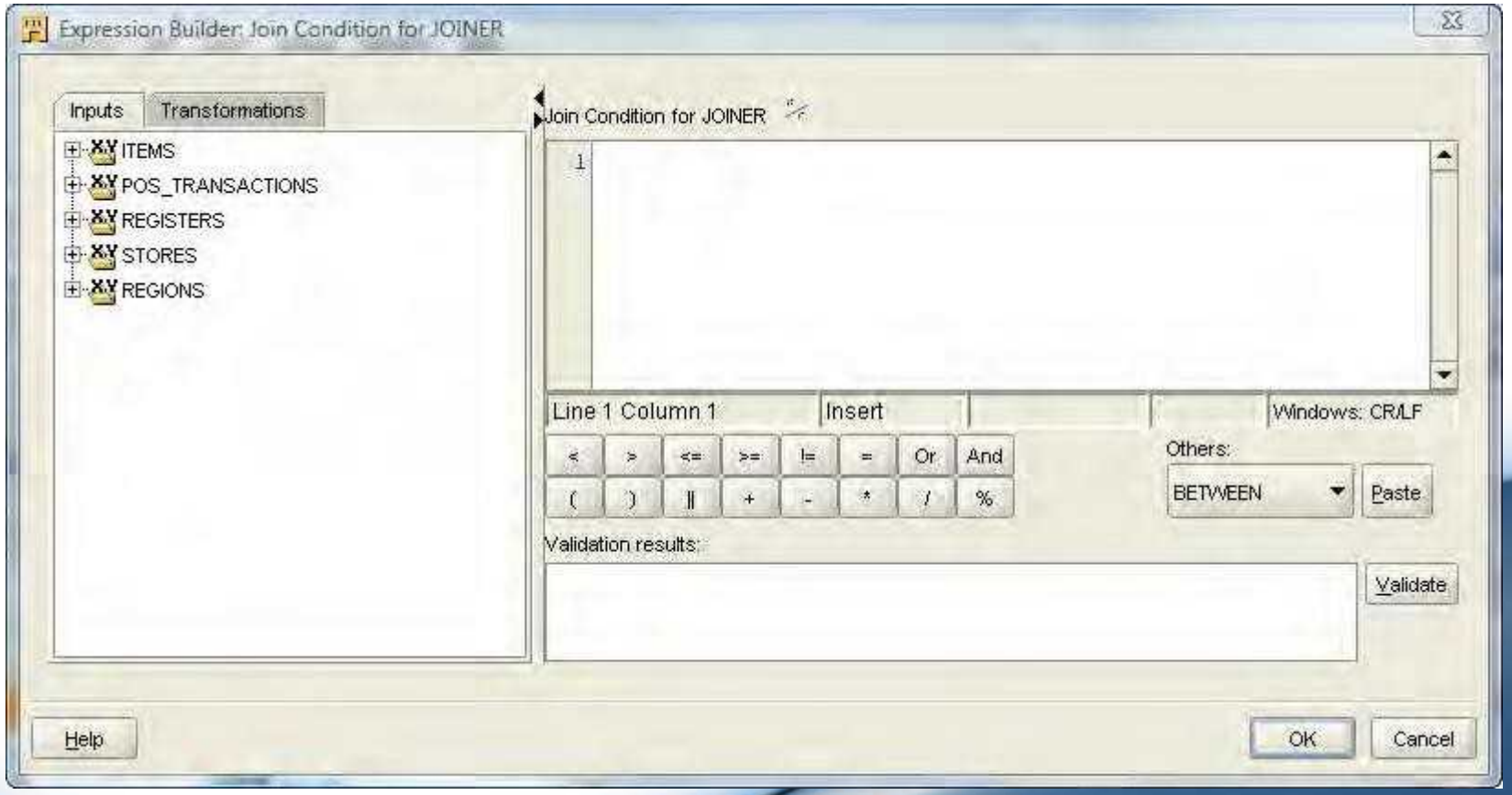

Mapping Editor: STAGE_MAP
Mapping Edit View Debug Window Help
Explorer
Tables
EMPLOYEES
ITEMS
POS_TRANSACTIONS
REGIONS
REGISTERS
STORES
Available Objects
Selected Objects
Group Properties: POS_TRANS_STAGE.INOUTGRP1
Palette
All
Flat File Operator
Joiner
Key Lookup
Bird's Eye View
Mapping
ITEMS
POS_TRANSACTIONS
REGISTERS
STORES
REGIONS
JOINER
70%

Defining operator properties for the Joiner

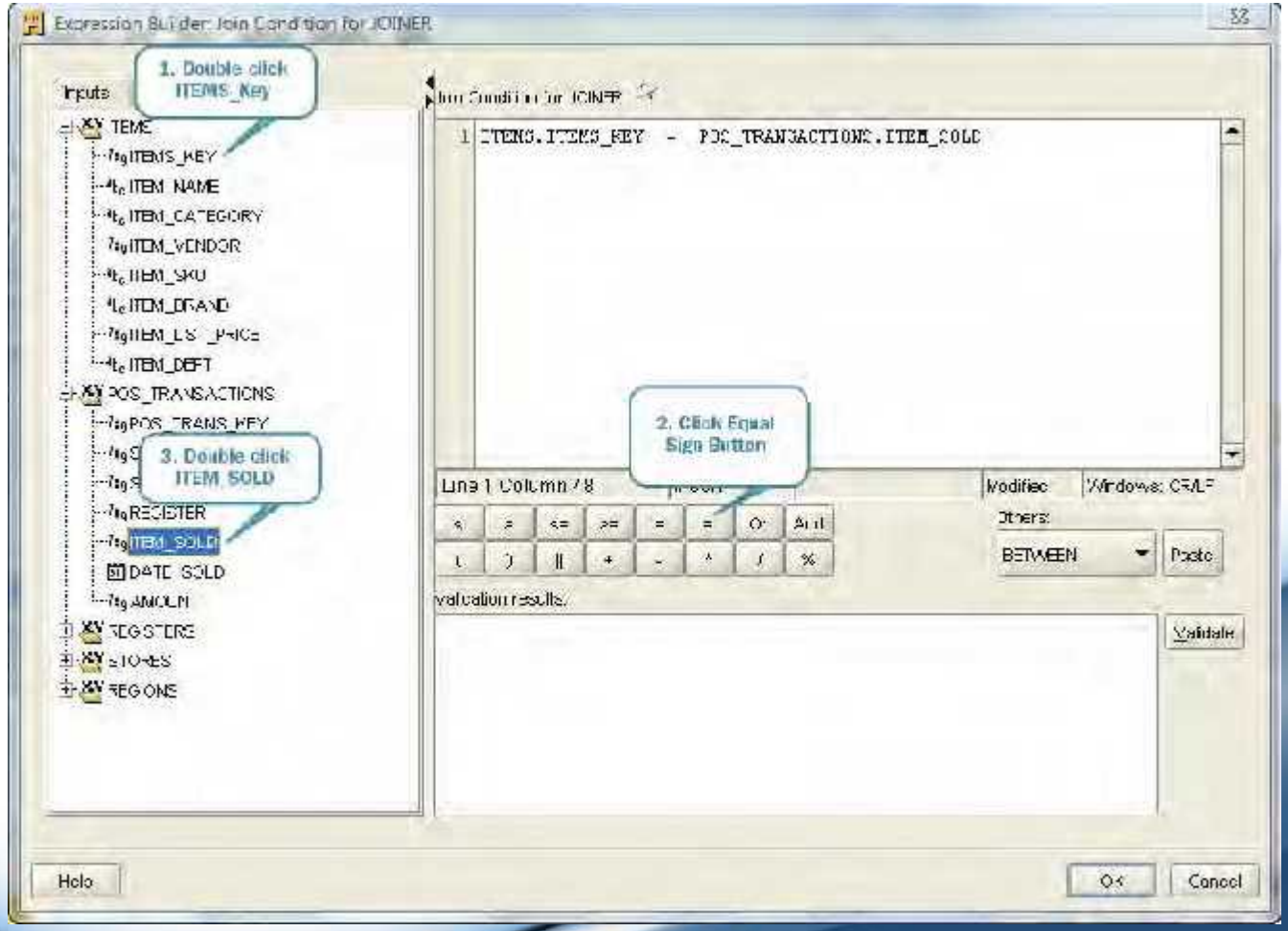
- If the **JOINER operator** is not already selected, click once on the header of the box to select it and the Properties window will immediately change to display **the properties** of the selected object.
- We can see one property mentioned there, **Join Condition**
- Click on the blank box to the right of the **Join Condition label.**



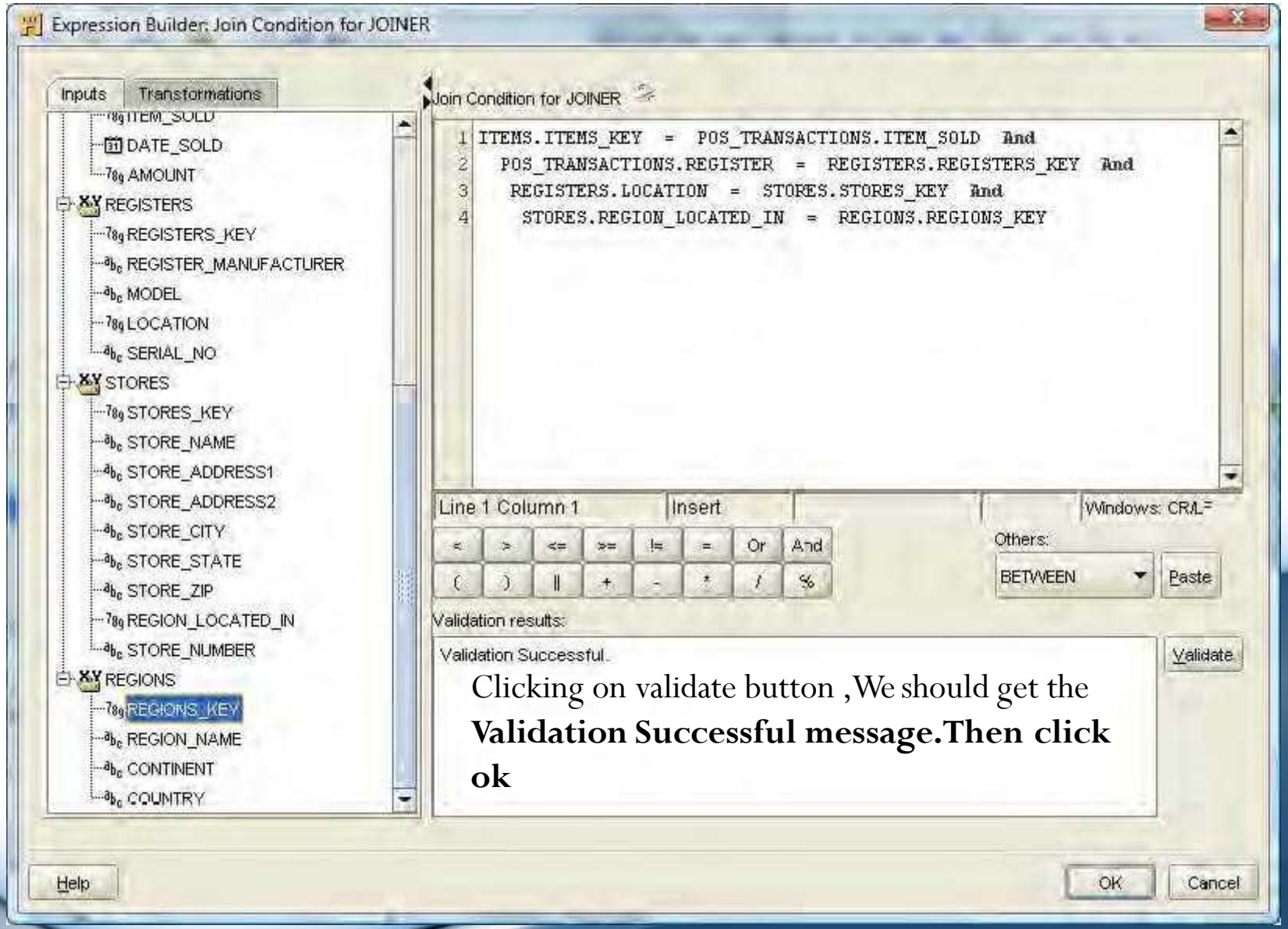
- OWB provides the **Expression Builder.** This is a dialog box we can invoke to interactively build our Joiner condition.


Expression Builder: Join Condition for JOINER
Inputs
Transformations
ITEMS
POS_TRANSACTIONS
REGISTERS
STORES
REGIONS
Join Condition for JOINER
1
Line 1 Column 1
Insert
Windows: CR/LF
< > <= >= != = Or And
( ) || + - * / %
Others:
BETWEEN
Paste
Validation results:
Validate
Help
OK
Cancel

How to add an join equation



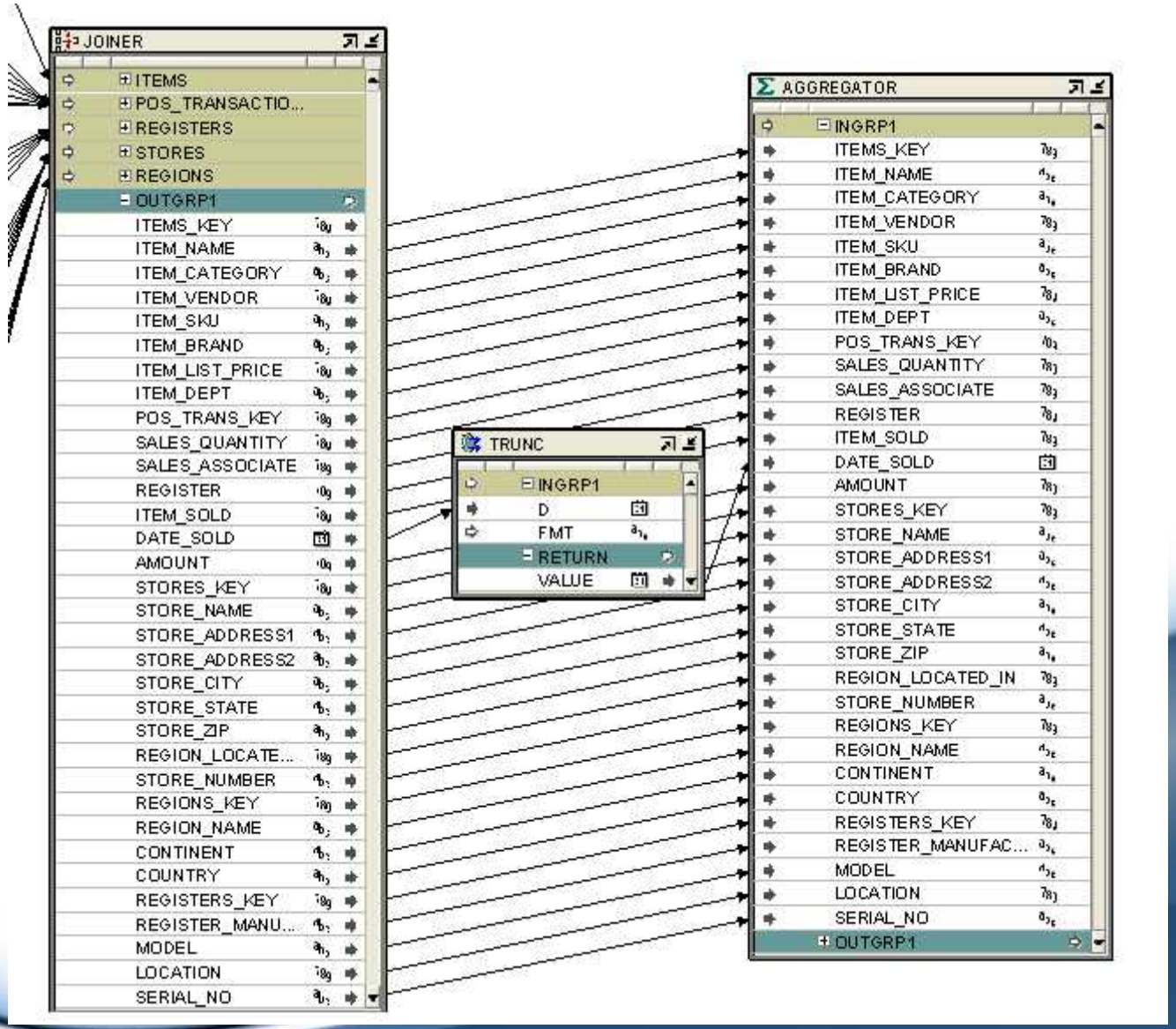
Continue till we satisfy our join eqations


Expression Builder: Join Condition for JOINER
Inputs
Transformations
DATE_SOLD
AMOUNT
REGISTERS
REGISTERS_KEY
REGISTER_MANUFACTURER
MODEL
LOCATION
SERIAL_NO
STORES
STORES_KEY
STORE_NAME
STORE_ADDRESS1
STORE_ADDRESS2
STORE_CITY
STORE_STATE
STORE_ZIP
REGION_LOCATED_IN
STORE_NUMBER
REGIONS
REGIONS_KEY
REGION_NAME
CONTINENT
COUNTRY
Join Condition for JOINER
1 ITEMS.ITEMS_KEY = POS_TRANSACTIONS.ITEM_SOLD And
2 POS_TRANSACTIONS.REGISTER = REGISTERS.REGISTERS_KEY And
3 REGISTERS.LOCATION = STORES.STORES_KEY And
4 STORES.REGION_LOCATED_IN = REGIONS.REGIONS_KEY
Line 1 Column 1
Insert
Windows: CR/L
< > <= >= != = Or And
( ) || + - * / %
Others:
BETWEEN
Paste
Validation results:
Validation Successful.
Validate
Clicking on validate button ,We should get the Validation Successful message.Then click ok
Help
OK
Cancel

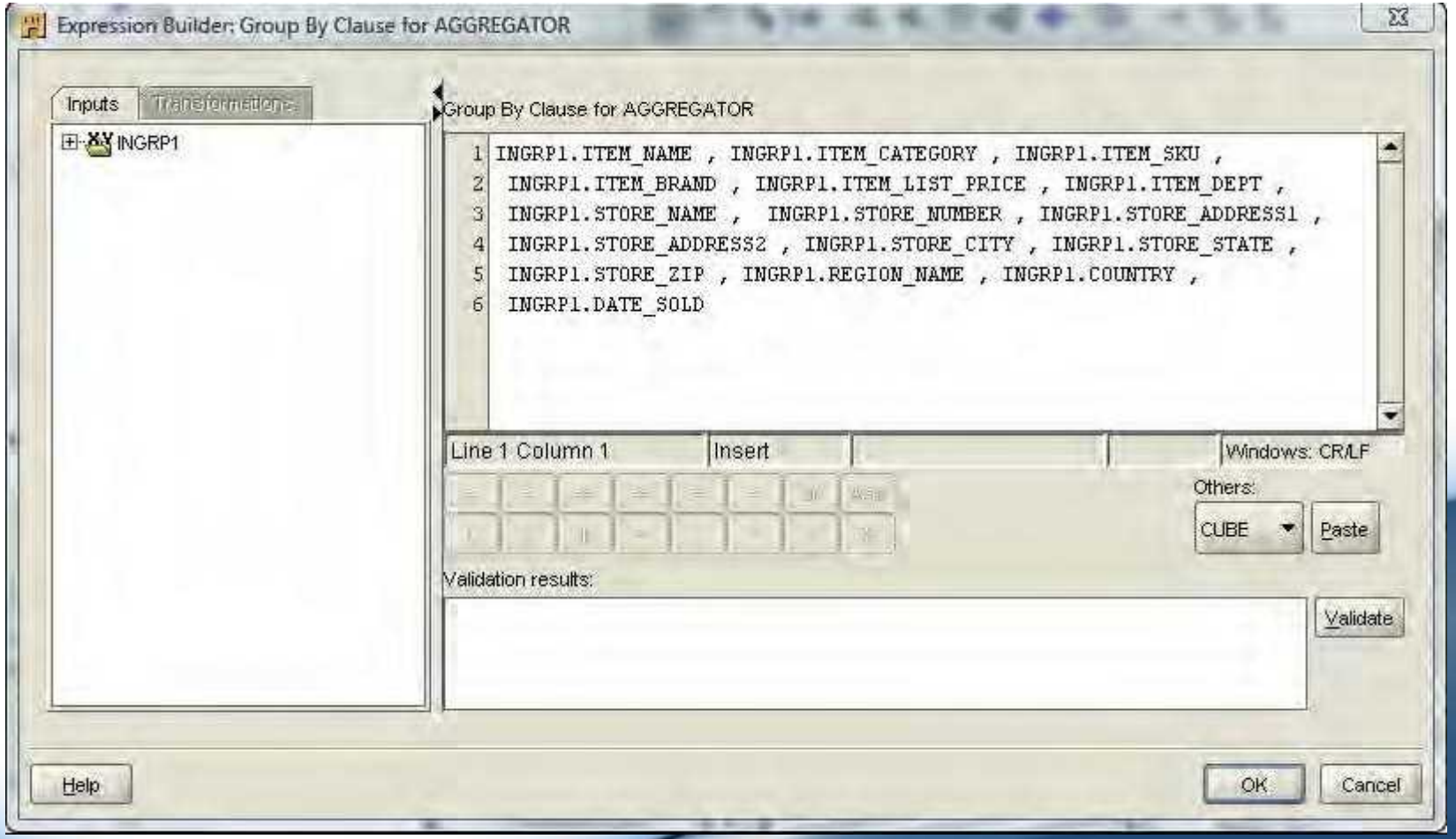
Adding an Aggregator operator

1. Drag an **AGGREGATOR** operator from the Palette window to the canvas and drop it to the right of the **JOINER** operator between that operator and the **POS_TRANS_STAGE** target operator.
2. Connect the output attributes from the JOINER operator as input to the **AGGREGATOR** operator by dragging the **OUTGRP1** output group and dropping it on the **INGRP1** input group of the **AGGREGATOR** operator.
3. We need to remove the line that got dragged to the input of the **Aggregator** operator for the **DATE_SOLD** attribute by clicking on the line and **pressing the *Delete key, or right-clicking and selecting Delete from the pop-up menu.***
4. Drag a **Transformation Operator** from the Palette window and drop it on the canvas between the **Joiner operator and the Aggregator** operator near the **DATE_SOLD attribute.**
   - In the resulting pop up that appears, we'll scroll down the window until the Date() functions appear and then select the TRUNC() function. It will look like the following:
   - TRUNC(IN DATE, IN VARCHAR2) return DATE

5. Connect the **DATE_SOLD** attribute in the **OUTGRP1 group** of the Joiner to the **D attribute of the INGRP1** of the **TRUNC transformation operator.** Then connect the **VALUE** attribute of the **RETURN output group** of the **TRUNC operator** to the **DATE_SOLD attribute** of the **INGRP1 group** of the Aggregator operator**.**


JOINER
ITEMS
POS_TRANSACTIO...
REGISTERS
STORES
REGIONS
OUTGRP1
ITEMS_KEY
ITEM_NAME
ITEM_CATEGORY
ITEM_VENDOR
ITEM_SKU
ITEM_BRAND
ITEM_LIST_PRICE
ITEM_DEPT
POS_TRANS_KEY
SALES_QUANTITY
SALES_ASSOCIATE
REGISTER
ITEM_SOLD
DATE_SOLD
AMOUNT
STORES_KEY
STORE_NAME
STORE_ADDRESS1
STORE_ADDRESS2
STORE_CITY
STORE_STATE
STORE_ZIP
REGION_LOCATE...
STORE_NUMBER
REGIONS_KEY
REGION_NAME
CONTINENT
COUNTRY
REGISTERS_KEY
REGISTER_MANU...
MODEL
LOCATION
SERIAL_NO
TRUNC
INGRP1
D
FMT
RETURN
VALUE
AGGREGATOR
INGRP1
ITEMS_KEY
ITEM_NAME
ITEM_CATEGORY
ITEM_VENDOR
ITEM_SKU
ITEM_BRAND
ITEM_LIST_PRICE
ITEM_DEPT
POS_TRANS_KEY
SALES_QUANTITY
SALES_ASSOCIATE
REGISTER
ITEM_SOLD
DATE_SOLD
AMOUNT
STORES_KEY
STORE_NAME
STORE_ADDRESS1
STORE_ADDRESS2
STORE_CITY
STORE_STATE
STORE_ZIP
REGION_LOCATED_IN
STORE_NUMBER
REGIONS_KEY
REGION_NAME
CONTINENT
COUNTRY
REGISTERS_KEY
REGISTER_MANUFAC...
MODEL
LOCATION
SERIAL_NO
OUTGRP1

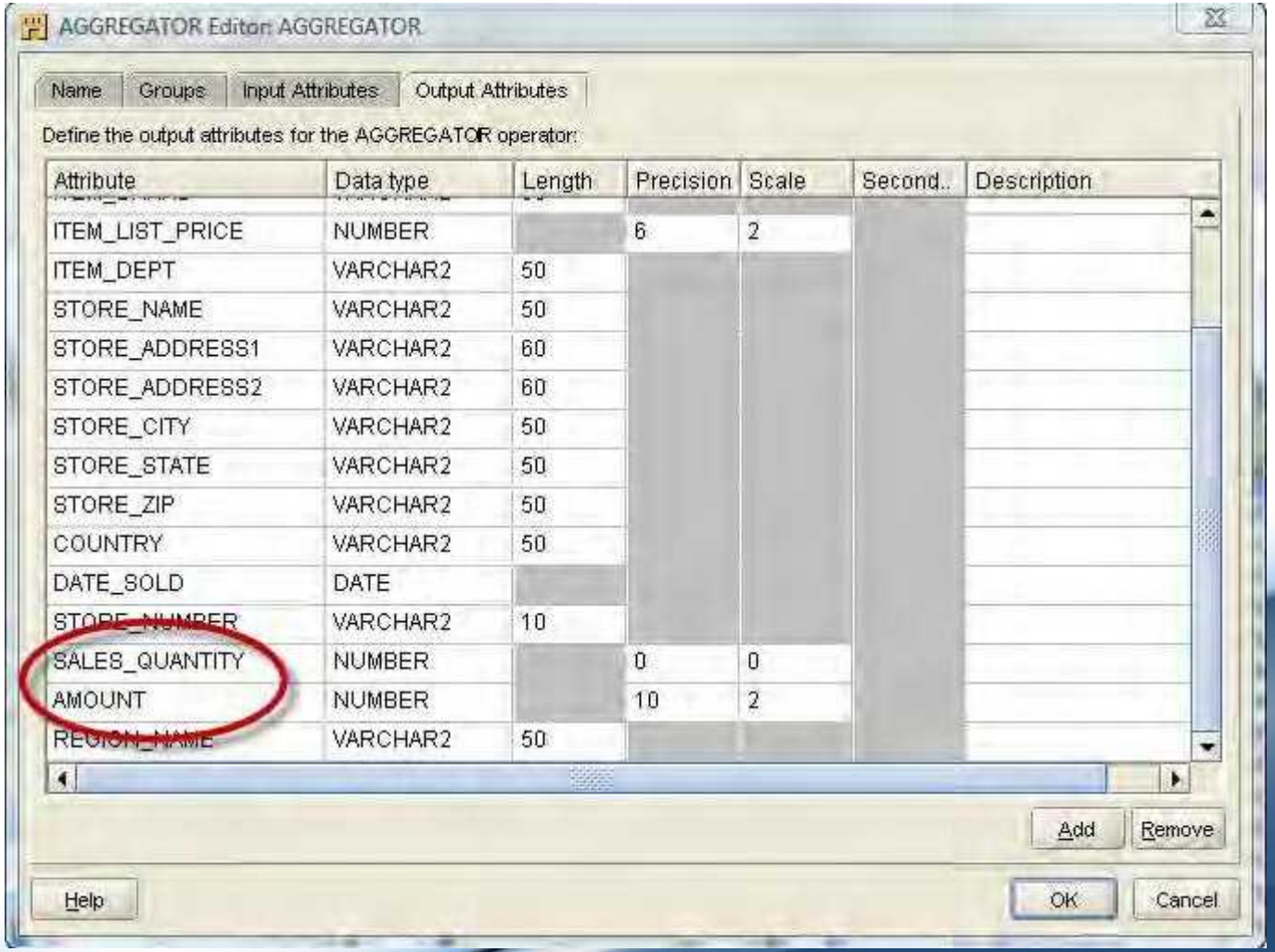
6. We have our input set for the Aggregator operator and now we need to address the output. Let's select the Aggregator operator by clicking on the title bar of the window where it says **AGGREGATOR.** The Properties window of the Mapping Editor will display the properties for the aggregator.
7. The very first attribute listed is **Group By Clause.** We'll click on the ellipsis (…) on its right to open the Expression Builder for the Group By Clause.
8. Enter the following attributes separated by commas by double-clicking each in the **INGRP1 entry in the left window:**
INGRP1.ITEM_NAME , INGRP1.ITEM_CATEGORY ,
INGRP1.ITEM_SKU , INGRP1.ITEM_BRAND ,
INGRP1.ITEM_LIST_PRICE , INGRP1.ITEM_DEPT ,
INGRP1.STORE_NAME , INGRP1.STORE_NUMBER ,
INGRP1.STORE_ADDRESS1 , INGRP1.STORE_ADDRESS2 ,
INGRP1.STORE_CITY , INGRP1.STORE_STATE ,
INGRP1.STORE_ZIP, INGRP1.REGION_NAME ,
INGRP1.COUNTRY , INGRP1.DATE_SOLD


Expression Builder: Group By Clause for AGGREGATOR
Inputs
Transformations
INGRP1
Group By Clause for AGGREGATOR
1 INGRP1.ITEM_NAME , INGRP1.ITEM_CATEGORY , INGRP1.ITEM_SKU ,
2 INGRP1.ITEM_BRAND , INGRP1.ITEM_LIST_PRICE , INGRP1.ITEM_DEPT ,
3 INGRP1.STORE_NAME , INGRP1.STORE_NUMBER , INGRP1.STORE_ADDRESS1 ,
4 INGRP1.STORE_ADDRESS2 , INGRP1.STORE_CITY , INGRP1.STORE_STATE ,
5 INGRP1.STORE_ZIP , INGRP1.REGION_NAME , INGRP1.COUNTRY ,
6 INGRP1.DATE_SOLD
Line 1 Column 1
Insert
Windows: CR/LF
Others:
CUBE
Paste
Validation results:
Validate
Help
OK
Cancel

9. We'll click on the OK button to close the Expression Builder dialog box and looking at the **AGGREGATOR** now, we can see that it **added an output attribute** for each of these attributes in our group by clause. **This list of attributes has every attribute needed for the POS_TRANS_STAGE** operator **except for the two number measures, SALE_QUANTITY and SALE_DOLLAR_AMOUNT. So let's add them manually.**

10. We'll right-click on the **OUTGRP1** attribute group of the **AGGREGATOR** operator and select **Open Details...** from the pop up.

- We'll click on the Output Attributes tab, and then on the **Add button twice to add two new output attributes.** Let's click on the first one it added (OUTPUT1) and change its name to SALES_QUANTITY and leave the type NUMBER with 0 for precision and scale. We'll click on the second one (OUTPUT2) and change the name to AMOUNT and make it type NUMBER with precision 10 and scale 2.


AGGREGATOR Editor: AGGREGATOR
Name
Groups
Input Attributes
Output Attributes
Define the output attributes for the AGGREGATOR operator:
Attribute
Data type
Length
Precision
Scale
Second..
Description
ITEM_LIST_PRICE
NUMBER
6
2
ITEM_DEPT
VARCHAR2
50
STORE_NAME
VARCHAR2
50
STORE_ADDRESS1
VARCHAR2
60
STORE_ADDRESS2
VARCHAR2
60
STORE_CITY
VARCHAR2
50
STORE_STATE
VARCHAR2
50
STORE_ZIP
VARCHAR2
50
COUNTRY
VARCHAR2
50
DATE_SOLD
DATE
STORE_NUMBER
VARCHAR2
10
SALES_QUANTITY
NUMBER
0
0
AMOUNT
NUMBER
10
2
REGION_NAME
VARCHAR2
50
Add
Remove
Help
OK
Cancel

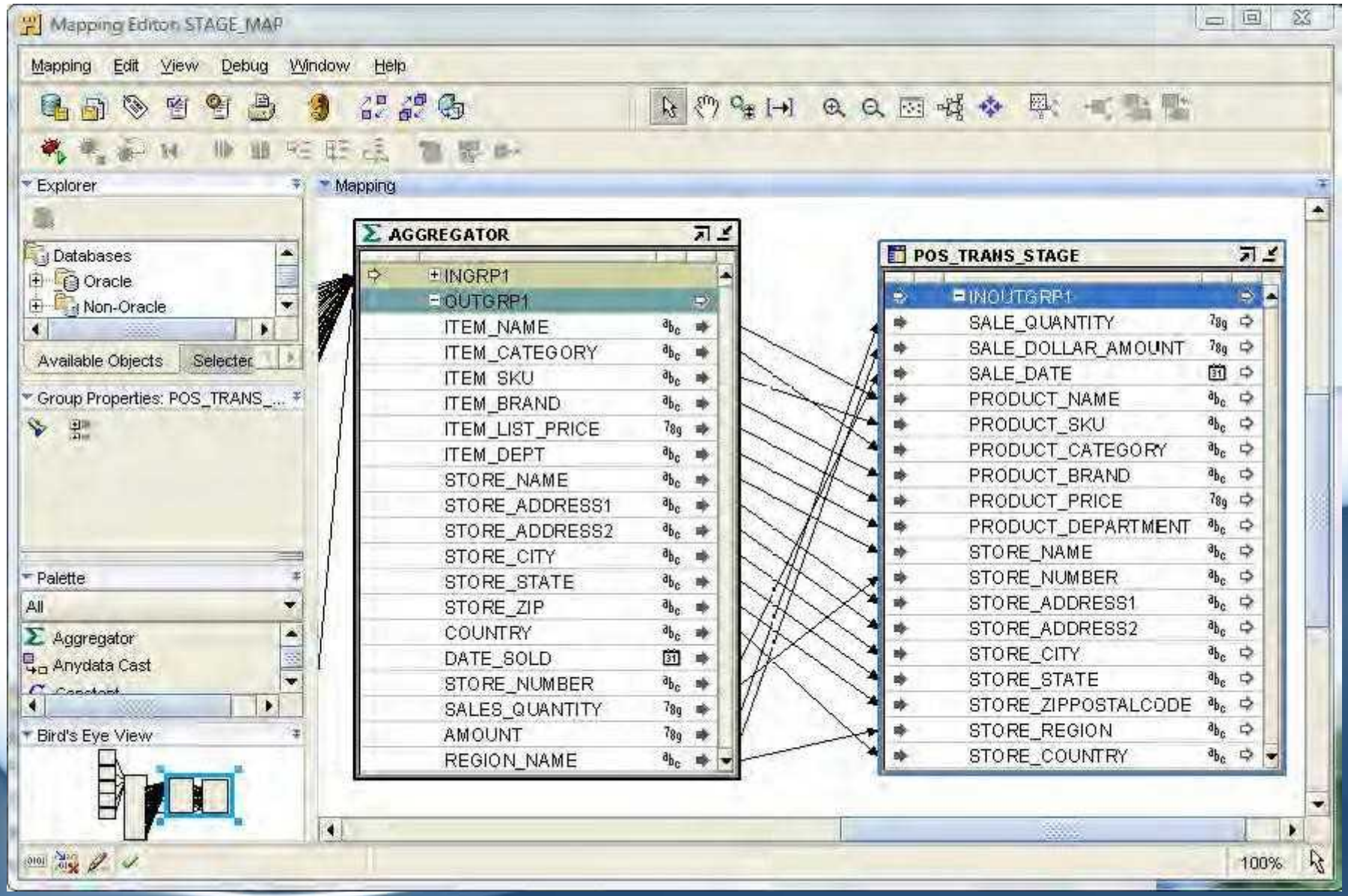
11. Now we need to apply the SUM() function to these two new attributes, so we'll click on **SALES_QUANTITY in the OUTGRP1 group** of the Aggregator. In the Properties window of the Mapping Editor on the left, click on the ellipsis beside the Expression property to launch the Expression editor for this attribute.

12. The **Expression** editor for output attributes of an Aggregator is custom built to apply aggregation functions. We'll select **SUM** from the **Function** drop-down menu**, ALL** from the **ALL/DISTINCT** drop-down menu, and **SALES_QUANTITY** from the **Attribute** drop-down menu. We'll then click on the Use Above Values button and the expression will fill in showing the SUM function applied to the SALES_QUANTITY attribute.

We'll click on the OK button to save the expression and close the dialog box.Then we'll do the same thing for the AMOUNT output attribute of the Aggregator, but will select AMOUNT for the Attribute drop-down menu.

- Make the following attribute connections between the Aggregator and the POS_TRANS_STAGE table by clicking and dragging a line between attributes.

- This completes the staging mapping.


Mapping Editor STAGE_MAP
Mapping Edit View Debug Window Help
Explorer
Databases
Oracle
Non-Oracle
Available Objects
Selected
Group Properties: POS_TRANS_...
Palette
All
Aggregator
Anydata Cast
Bird's Eye View
Mapping
AGGREGATOR
INGRP1
OUTGRP1
ITEM_NAME
ITEM_CATEGORY
ITEM_SKU
ITEM_BRAND
ITEM_LIST_PRICE
ITEM_DEPT
STORE_NAME
STORE_ADDRESS1
STORE_ADDRESS2
STORE_CITY
STORE_STATE
STORE_ZIP
COUNTRY
DATE_SOLD
STORE_NUMBER
SALES_QUANTITY
AMOUNT
REGION_NAME
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
100%

DATAWAREHOUSING

END OF UNIT 4

END OF CHAPTER 2