



# **Business Intelligence for *i*Series**

## **BL04 – Business Intelligence Lab**

### **For**

## **IBM ITSO Forum, Rochester**

Document Version 1.J.3

**Jacqueline Jansen**  
Senior Consulting IT Specialist  
iSeries Solutions Center  
[jjansen@ca.ibm.com](mailto:jjansen@ca.ibm.com)

February 10, 2003



# **Business Intelligence for iSeries**

## **Business Intelligence LAB**

### **1.0 Introduction**

This exercise will take you through the steps that you might typically go through when moving data from an operational system database through a data warehouse database to an OLAP cube data mart. It will then allow you to present the information in the OLAP cube using a spreadsheet of choice (Lotus-123 or Microsoft Excel) or DB2 OLAP Server Analyzer.

You will be using a sample database installed in a library called SCSAMPLE40 on the iSeries. SCSAMPLE40 contains data from a fictitious company called The Outdoor Connection. The Outdoor Connection is a sporting goods company that sells through three channels: retail stores, speciality stores, and mail-order catalogs. Its primary product lines are skiing equipment, biking equipment, camping gear, and clothing. The Outdoor Connection is based in the United States. It also does business in Canada, Belgium, Germany, France, the United Kingdom, Japan, Australia, and New Zealand.

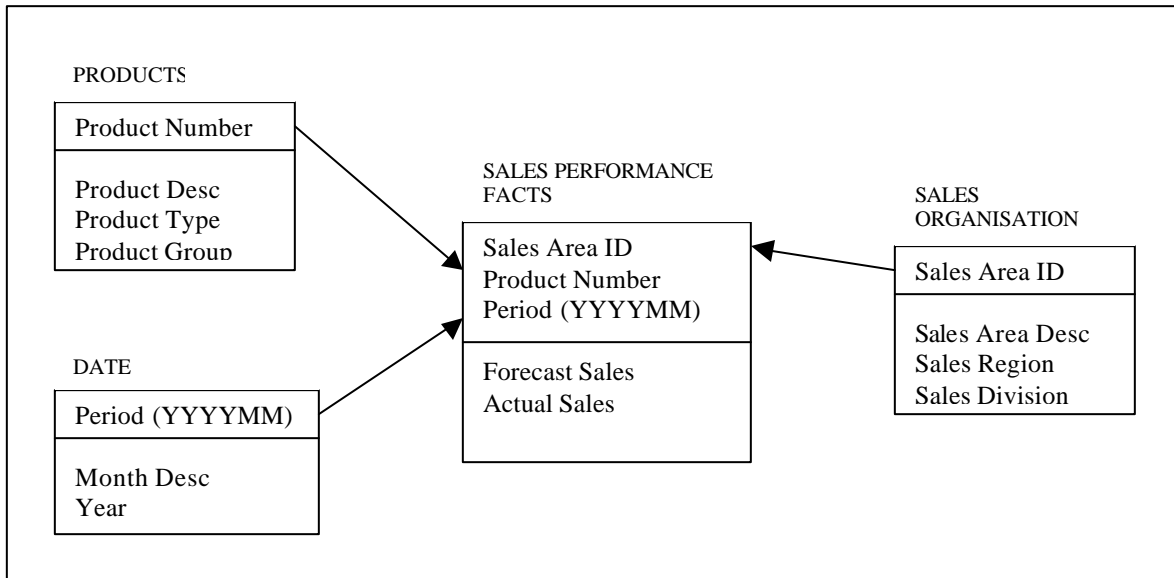
This exercise will use the Outdoor Connection database in SCSAMPLE40 and create a star-schema data model using IBM OLAP Builder. This process will make use of the data extraction, transformation and load functionality of OLAP Builder. The lab will then use IBM DB2 OLAP Server to create an OLAP cube on the iSeries and finally use both a spreadsheet application and IBM Analyzer to view the data in the cube

## 2.0 OLAP Builder

### 2.1 Database Design

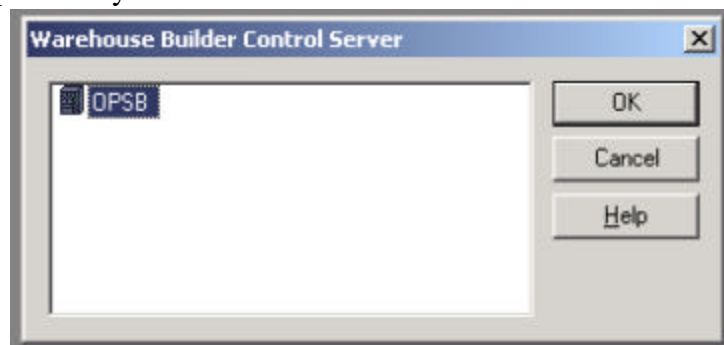
For this lab both the source data and the target data will be on the same physical system. However in a real environment it is more likely that the source data will be on one or more different physical systems.

The target star-schema data model that will be created is shown here.



### 2.2 Create Data Warehouse

Start OLAP Builder. Click on **Start** on your Windows task bar, select **Programs|IBM DB2 OLAP Server 8.1 – iSeries|Warehouse Manager and Builder|Builder Clients** and you will be presented with the following display where the Warehouse Builder Control Server will list the Showcase ODBC data sources that have been previously created.



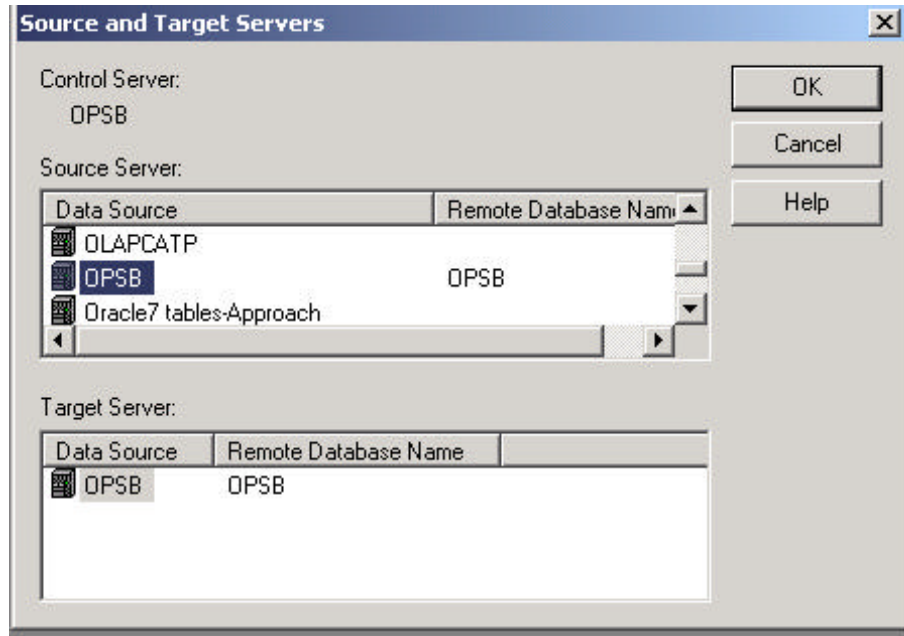
Select the Control Server iSeries (OPSB), and click on OK.

At the **Showcase ODBC Log on** window enter a valid iSeries user profile name and password and click OK.

In the **Warehouse Builder Assistant** window check that **Create a new definition** has been selected and click on OK. If the Warehouse Builder Assistant menu doesn't start up automatically, choose **Tools|Need Assistance**.

You will then be presented with the following display, and asked to select your source and target systems.

Ensure that both the source server data source and the target server data source that have been selected are the name of your iSeries ODBC data source (OPSB) and click OK.



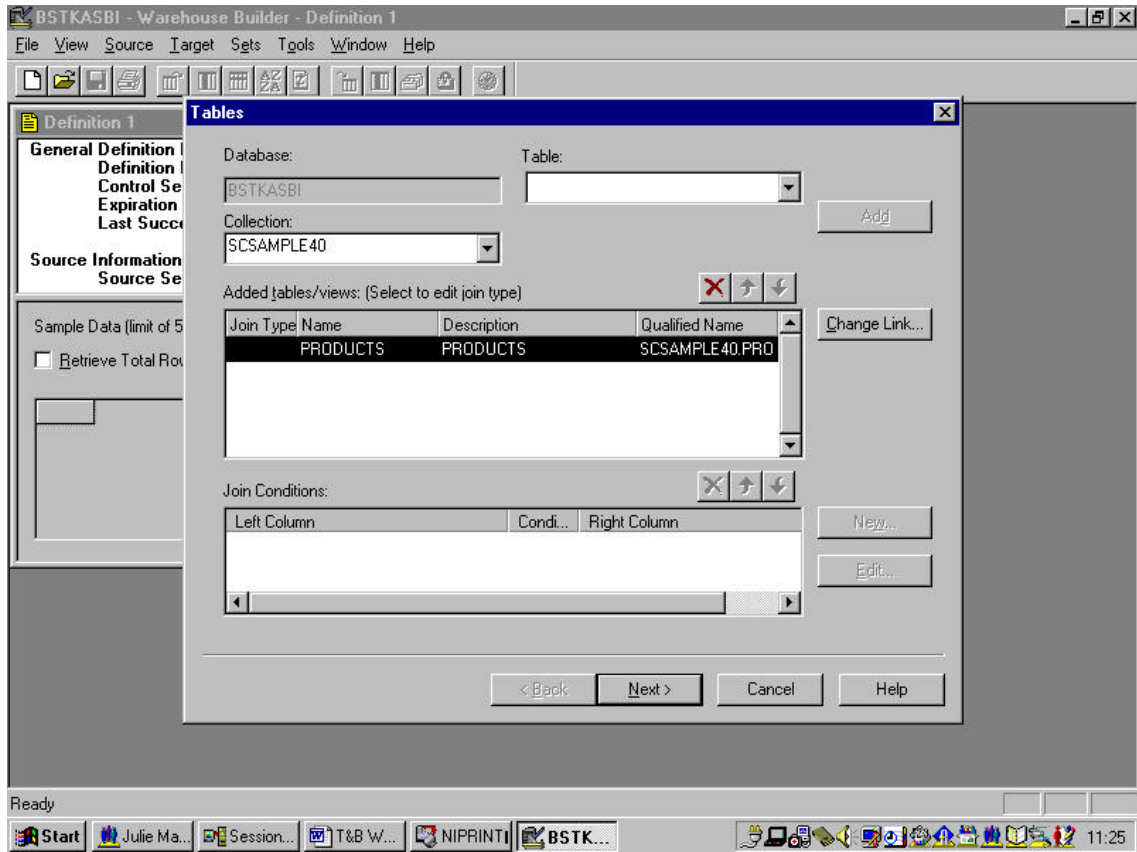
### 2.2.1 Create Definition 1

In OLAP Builder 'Definitions' are created. Definitions describe the process of loading from source to target and specify any data transformations that need to take place.

Definition 1 will be the definition required to create and load the target table TEAMxx/PRODUCTS from the source table SCSAMPLE40/PRODUCTS.

You should now be looking at a window called 'Tables'. At this stage the source database file needs to be defined to OLAP Builder.

The name of the source database will already be entered for you. Ensure that SCSAMPLE40 is in the **Collection** box (a collection is another term for library) - (use the drop-down to find SCSAMPLE40 if it is not already displayed). Again from the drop-down ensure that PRODUCTS is selected for the **Table** box. Click on Add and your screen should look as follows:

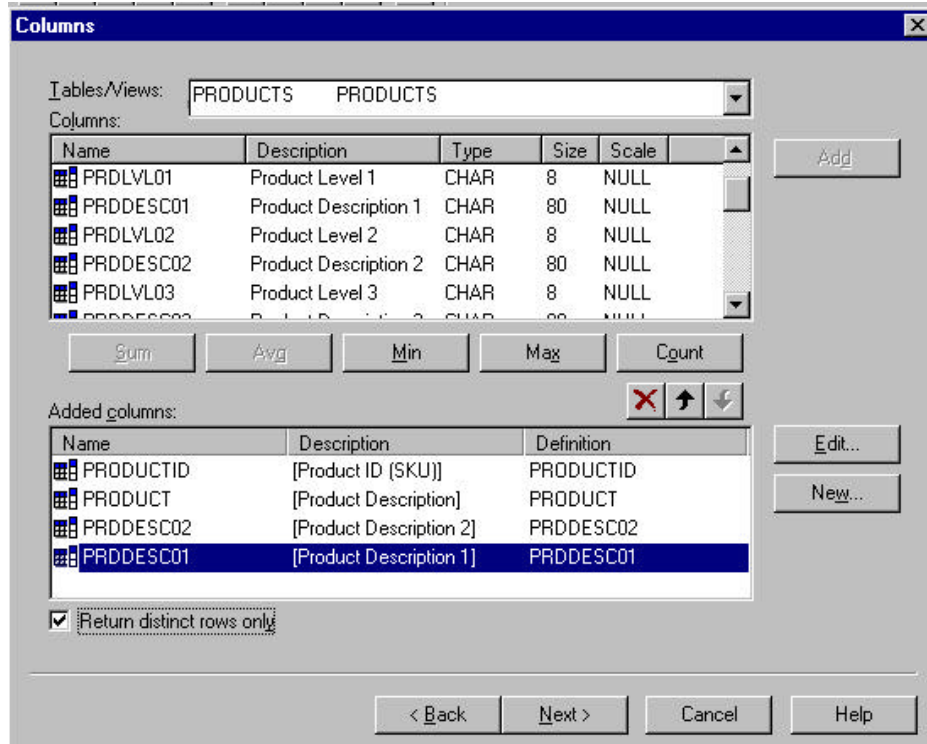


Click on **Next**.

You will then be presented with a **Columns** window where you need to select the columns that you want to extract from the SCSAMPLE40/PRODUCTS table.

Click on PRODUCTID and click on **Add**. Having done that, select the following source columns in the following order: PRODUCT, PRDDESC02, PRDDESC01. Finally check the **Return distinct rows only** checkbox.

Your screen should now look like the one below:



Click on **Next**.

The next window is a **Conditions** window where you can specify which rows you want to select from the source table. We wish to select all records from the SCSAMPLE/PRODUCTS table, so just click on **Next** to go to the next window.

The next window is a **Sort** window. No sorts are to be specified here so just click on **Finish**.

You have now completed defining the source extraction rules for definition one.

The next stage is to define the target that you want to send the data to. You should now be looking at a window called Table, and you are asked to enter the name of the target collection (library) and table. Enter **TEAMxx** for the collection name and **PRODUCTS** for the table name. Review the **Destination Options**, and note that you can either add to or replace the data that is already in the target table. In this example we are creating a new target table and will only be loading the data once, so it does not matter which option is selected. Ensure that the **Add timestamp to target table** is checked and click on **Next**.

If your collection/library doesn't exist Warehouse Builder will prompt you to create it. Select **YES**.

You will then be presented with a **Target Columns** window. In the **Target Table Columns** section you will see listed the columns that you selected from the source table. You now have the opportunity to change the naming, the data type and the size etc of each field.

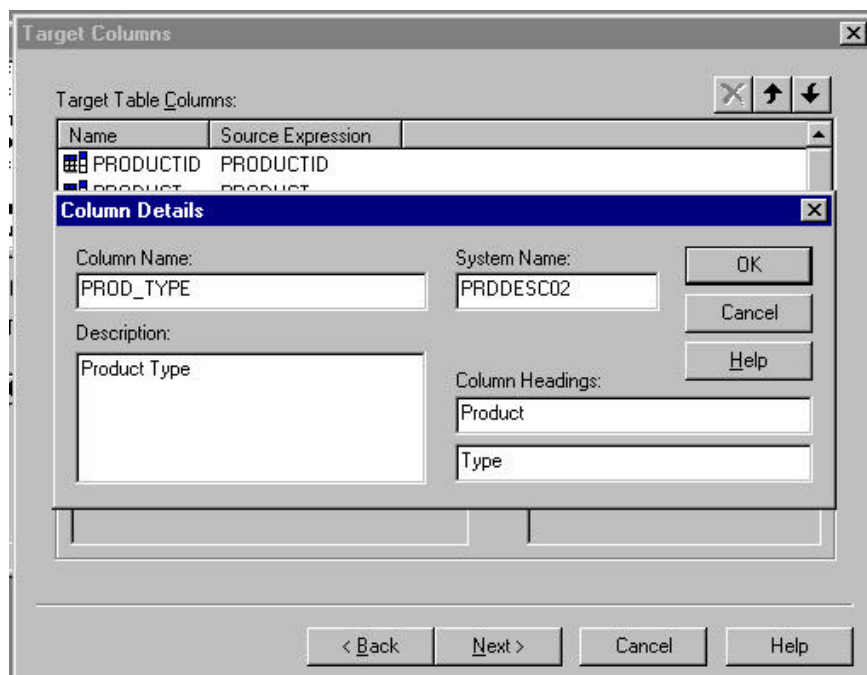
PRODUCTID is defined as numeric(8,0) which is fine for the target file.

Next click on **PRODUCT**. This column is defined as CHAR(80). Change this so that the target column is CHAR(20).

Next click on **PRDDESC02**. **PRDDESC02** is actually Product Type. Therefore we want to change the name of the field to better reflect it's meaning.

To the right of the **Column Name** box is a small box with a hand and a finger pointing in it. This is called the Target Columns Detail button. Click on this Target Columns Detail button (ensuring first that **PRDDESC02** is in the **Column Name** box).

You will be presented with a column details window. Change the column name to **PROD\_TYPE**. Change the description to Product Type and change the column headings to so that "Product" and "Type" appear on two lines. Your screen should then look like the one below.



Click on **OK** to change the column details. If you receive a warning about Dropping and Creating your new table click on **YES**.

You will then return to the **Target Columns** window. **PRDDESC02** is defined as CHAR(80). Again change this so that the target column is CHAR(20).

Next click on **PRDDESC01**. Select the **Columns Detail** button and change the column name to **PROD\_GROUP**. Change the description to Product Group and change the column headings to Product Group. Click on **OK** on the Column Details window. Change the size of the target field to CHAR(20). Click on **Next** from the Target Table Columns window to move to the **Indexes** window.

We are not going to create any indexes for this target table so just click on Next to move on to the Privileges window. We are not going to set any additional privileges, but take time to look at the options

that are available. Click on **Finish** and the task of creating Definition 1 is complete. We now need to save our definition.

From the main menu at the top of the screen select **File|Save As** and key in the definition name 'TEAMxx Product Load'. Then click Save and we now have a Definition called Product Load which we can run to populate a target table called TEAMxx/PRODUCTS from a source table called SCSAMPLE40/PRODUCTS. Ignore Warehouse Builder warnings and proceed. Select **YES**.

### 2.2.2 Create Definition 2 and 3


In the interest of time Definitions 2 and 3 have been created for you. Detail instructions to create Definition 2 and 3 are located in the appendix if you wish to create them yourself.

Definition 2 is the definition required to create and load the target table TEAMxx/SALESORG from the source table SCSAMPLE40/SALESORG.

Definition 3 is the definition required to create and load the target table TEAMxx/DATE\_TBL from the source table SCSAMPLE40/TIMEDIM.

From the main menu at the top of the screen select **File|Open**. Highlight "Master Definition 2 SalesOrg Load" and click **OPEN**. Select **Target|Table** and modify the collection name from **Team99** to your library Teamxx. Select **OK**. You will receive a question from Builder "**Replace your Target Columns Definitions with default values from your Source Columns?**" Make sure you respond NO.

From the main menu at the top of the screen select **File|Save As** and key in the definition name 'TEAMxx SalesOrg Load'. Then click **Save** and respond **Yes** to the Builder Validation Error message. You now have a Definition called SalesOrg Load which you can run to populate a target table called TEAMxx/SALESORG from a source table called SCSAMPLE40/SALESORG.

To review the target data that will be generated look at the icons on the top task bar. Click on the ninth icon . This refreshes the sample data, and having clicked on it you will see some sample data being presented to you in the Date Load window.

Follow the same steps to modify "Master Definition 3 Date Load" and create "TEAMxx Date Load".

### 2.2.4 Create Definition 4

Definition 4 will be the definition required to create and load the target table TEAMxx/SALESPERF from the source tables SCSAMPLE40/SALESPERF and SCSAMPLE/TIMEDIM.

Click on the **New** icon or select **File|New|New** to create a new definition.

Select the source and target servers as before.



In the Tables window select the tables SALESPERF and TIMEDIM from the collection SCSAMPLE40. Click on OK when you get the message telling you that a default join could not be created.

We now need to tell Builder how to join these two files together. Click on the **New** button to create to new join. In the **Add Join Tables Conditions** window select PERIOD from the SALESPERF table, and the equals sign (=) from the middle section, and finally TDATE from the TIMEDIM table and click on **OK** then **Next**.

Our source table contains the field PERIOD in date format. If you go back to the diagram of the target data model you will see that we want the field PERIOD in our target table to be in the format YYYYMM. The month and year are not split out separately in the SALESPERF source table. We will join to the TIMEDIM table so that we can pick up the TMONTH and TYEAR fields to build the new PERIOD field in our target table.

Click **Next**. You should now be in the **Columns** window.

Ensure SALESPERF is in the **Tables/Views** box. From the **Columns** table click on AREA\_ID and click on **Add**. Then click on PRODUCTID and click on **Add**. The third column that we need is our YYYYMM field. Click on **New** to add a new field.

In the New Column Assistants window select the **If-then-else Assistant** and click on **OK**.

Whilst in the **If-then-else Assistant** take some time to look at the rich set of built-in functions that are available to you to enable you to specify your data transformations. The top **Functions** box is a drop-down and you can see then different categories of functions that are available. The larger box beneath that lists all of the functions in the category selected. Select any one and press F1 to get help on what the function does and how to use it.

In our example we have a field in the source data called TYEAR which is an INTEGER(10) field and a field called TMONTH which is also an INTEGER(10) field. We need to concatenate TYEAR and TMONTH together to form our YYYYMM target field. However, we can only concatenate character fields so we need first to transform TYEAR for a CHAR(4) field and TMONTH to a CHAR(2) field. There is a further issue we need to resolve because most months are only 1 digit in length. If we have month 3 in year 2001 for example we need to ensure that in the final result we end up with is 200103 and not 20013. We therefore need to test to see whether TMONTH has one digit in it or two – and if it has only one then we need to insert a '0'.

In the **If-then-else Assistant** enter the following expression in the **If** box:

```
RIGHT( CAST( TMONTH AS CHAR( 2 ) ), 1 ) = ''
```

This means that we are converting the field TMONTH to a CHAR(2) field and are then testing the right-most character to see whether it is a blank or not. If it is a blank then we know we have a month in the range 1-9.

Enter the following in the **Then** box:

```
(( CAST ( TYEAR AS CHAR ( 4 ) ) ) CONCAT ( '0' ) CONCAT ( CAST ( TMONTH AS CHAR ( 1 ) ) ) ) )
```

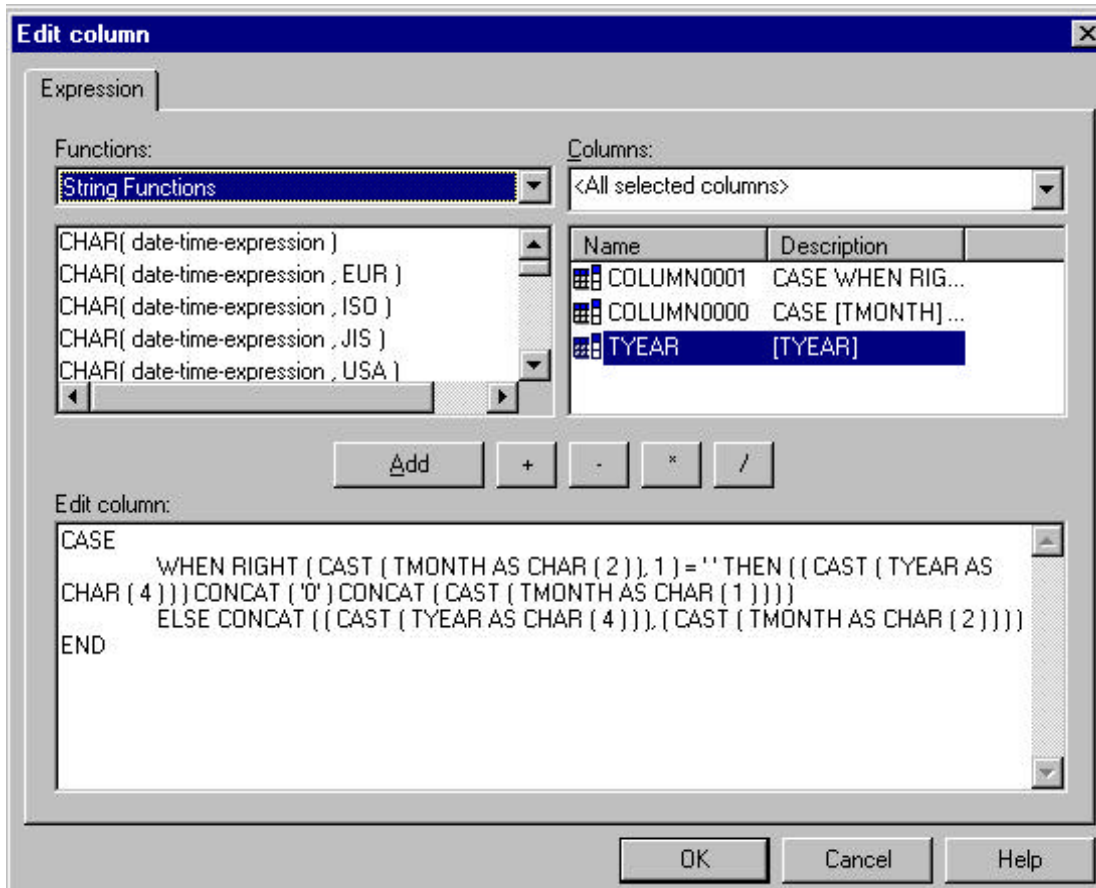
Here we are saying that now we know we have a one character month we need to concatenate TYEAR with a '0' and with the single character month.

Enter the following in the **Otherwise** box:

```
CONCAT ( ( CAST ( TYEAR AS CHAR ( 4 ) ) ) , ( CAST ( TMONTH AS CHAR ( 2 ) ) ) ) )
```

Here we are saying that if month is a two digit month then we just need to concatenate TYEAR and TMONTH together (ensuring that we convert them from numeric to character first).

Click on the **Expression** button to see the full expression. It should look something like the window below.



Return to the **Columns** window by clicking OK. We now only need to add the source data for forecast and actual sales. The source data in SALESPERF holds sales data down to the sales rep level. We will be holding the data at a higher level. We need therefore to total the sales data and GROUP BY area/product/period.

Select PRJ\_SALES and click on the **Sum** button. Finally select ACT\_SALES and click on the **Sum** button. Check the **Return distinct rows only** checkbox and click on **Next**.

The next window is the **Conditions** window. Specify here that we are only going to select those records where the date is less than the 1<sup>st</sup> of April 2001. List the columns in SALESPERF and select the column PERIOD. From the condition drop-down select the less-than sign (<). In the value section either select '4/1/2001' from the drop down or type in '2001-04-01'. Click on **Add** to add the condition and click on **Next** to move to the next window.

The next window is a **Sort** window. Click on **Finish** to complete the definition the source extraction rules for definition four.

The next stage is to define the target that you want to send the data to. In the **Table** window enter **TEAMxx** for the collection name and **SALESPERF** for the table name and click on **Next**.

You will then be presented with a **Target Columns** window.

Define the third field as CHAR(6) and call it PERIOD.

Using the **Column Details** button rename the fourth field to FCAST\_SALES. Set the system name to FC\_SALES, set the description to Forecast Sales Revenue and set the column headings to Forecast Sales.

Ensure the fifth field and last field is called ACTUAL\_SALES. Set the system name to ACT\_SALES, set the description to Actual Sales Revenue and set the column headings to Actual Sales.

Having completed all of these column detail changes, click on **Next** to move to the **Indexes** window. For this last definition we will instruct the system to create an index over our TEAMxx/SALESPERF table. Click on **Create** to create a new index.

In the **Index Definition** window leave the name to the default but specify that this is to be a unique index. From the **Available Columns** section click on AREA\_ID and click on **Add**, click on PRODUCT\_ID and click on **Add**, click on PERIOD and click on **Add**. The index definition is now complete to click on **OK**.

Click on **Next** to move to the **Privileges** window and click on **Finish**. The task of creating Definition 4 is complete. We now need to save our definition.

From the main menu at the top of the screen select **File!Save As** and key in the definition name 'TEAMxx Sales Perf Facts Load'. Then click Save and we now have a Definition which we can run to populate a target table called TEAMxx/SALESPERF from the source tables SCSAMPLE40/SALESPERF and SCSAMPLE/TIMEDIM.

### **2.2.5 Create a Warehouse Builder Set**

In order to actually run the definitions that we have defined we need to group our definitions into one or more sets. From a practical standpoint we might choose to group our dimension loading definitions into one set and have our fact table data loads in a different set. We can then run the two sets separately and

can schedule one set to run more frequently than another. For the purposes of this exercise however we will group each of our four definitions into one set.

On the main Warehouse Builder toolbar select **Sets|Work with Warehouse Builder Sets...**

In the Work with **Warehouse Builder Sets** window type 'TEAMxx Build Target' in the **New Warehouse Builder Set** box and click on **Create** to create a new set.

In the **Edit Warehouse Builder Set** window click on **Add** to add definitions to the set. From the list of definitions presented click on 'TEAMxx Product Load' and click on **Use**, next click on 'TEAMxx SalesOrg Load' and click on **Use**, next click on 'TEAMxx Date Load' and click on **Use**, and finally click on 'TEAMxx Sales Perf Facts Load' and click on **Use**. Click on **OK**.

Having returned to the **Edit Warehouse Builder Set** window select the Job Options Tab and change the job name to "TEAMxxBldr". Review the options available on the **Schedule** tab. Then click on **OK** and you will be returned to the **Work with Warehouse Builder Sets** window.

### 2.2.6 Run a Warehouse Builder Set

From the list of Warehouse Builder Sets, click on your Builder Set (called 'TEAMxx Build Target') and click on **Run**.

The definitions contained in that Warehouse Builder Set are now sent to the iSeries for processing. When finished you should be able to see your four tables created and populated in the library Target.

Click on **Statistics** to review the process of your job. Depending on how long it took you to click on Statistics, you will probably see that the status of your Warehouse Builder Set is Pending. Click on **Refresh** until you see the status change to Success. You will see details of how long the job took and how many rows were inserted into the target tables.

Click on **Definitions** and you will see statistics for each individual definition in the set. Use the drop-down definitions box to change the definition and review the statistics for each one.

To review the contents of your newly created tables you should use iSeries Navigator.

Open the iSeries Navigator icon found under IBM iSeries Access for Windows. Select + (expand) on **OPSB**, highlight **Database**, from the bottom panel select "**Select Libraries to Display**" and add Teamxx. On the right panel double click on **TEAMxx** and then double click on any tables/files that you wish to view. (Note: A faster view involves right clicking on the table name and selecting "Quick View".)

Congratulations – you have now completed the IBM DB2 OLAP Builder section of this exercise.

### 3.0 OLAP Server

This section will take you through:

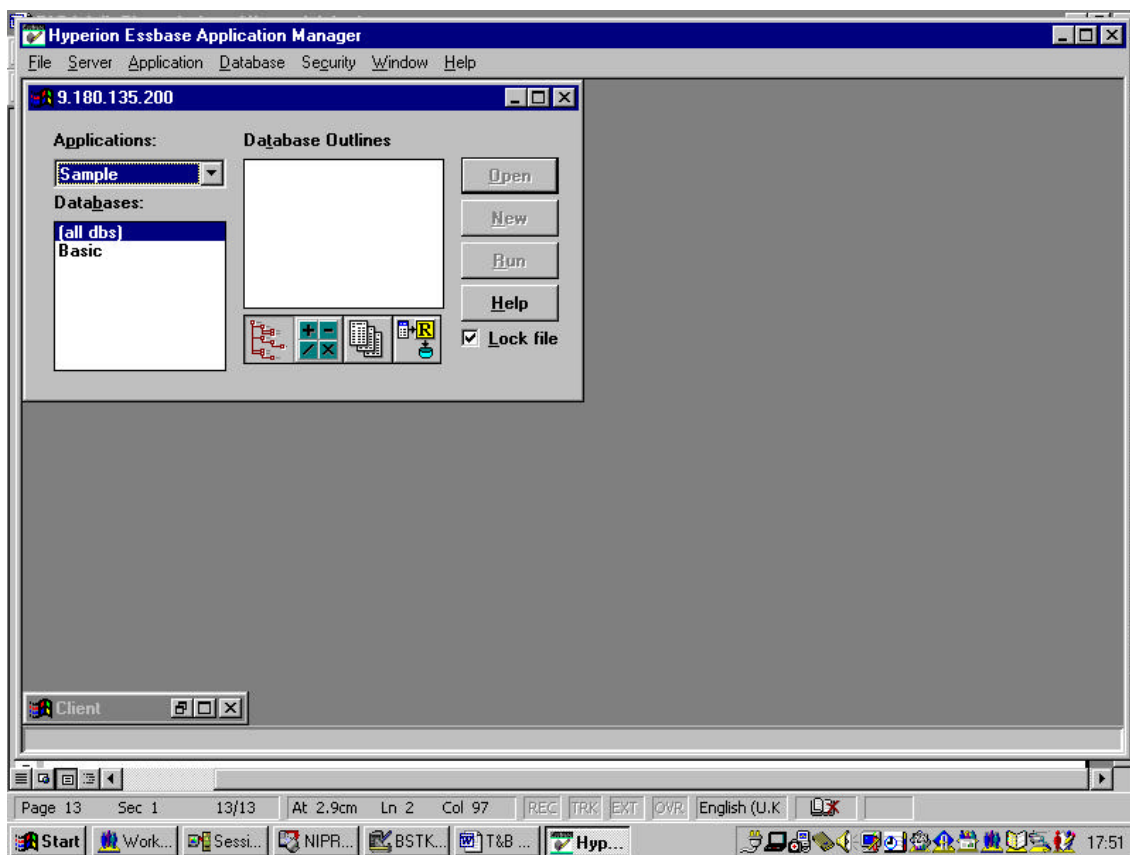
- building an OLAP cube on the iSeries
- loading the cube with the data that you created in your Target library in the previous section
- aggregating the data by running a calculation routine

Firstly we need to go into IBM OLAP Server on the client to design our OLAP cube. Specifically, we go into a part of the product called Application Manager.

Click on **Start** on your Windows task bar, select **Programs\DB2 OLAP Server 8.1 - iSeries\Application Manager**.

On the **Hyperion Essbase Application Manager** window click on **Server** from the main menu and then click on **Connect**. In the **Essbase System Login** window enter either the IP address or the name of your iSeries. Use your assigned userid and password.

You should be presented with a screen that looks similar to this:



Select **File>New!Application** from the main menu.

In the **Create New Application** window enter an application name of 'SALESxx'. Ensure that the **Server** radio button is selected. Click on **OK**.

Select **File!New!Database** from the main menu.

In the **Create New Database** window enter a database name of 'SALESDB'. Click on **OK**.

We have now given a name to the OLAP application, and have named the OLAP database that we are going to design and load data into. We must now provide OLAP Server with the details of what the database will look like. We do this by creating a something called an Outline.

On the screen you should be able to spot four Object type buttons in a row:



The first one is an Outline button, the second a Calc Script button, the third a Report Script button and the fourth a Data Load Rules button.

When the first Outline button is selected you will see the words **Database Outlines** appear above the central white box. When the fourth button is selected you will see the words **Data Load Rules** appear above the central white box.

Click on the Outline button. Ensure that your application (SALESxx) is in the Applications box, ensure that your database (SALESDB) is highlighted in the Databases box, ensure that the outline of the same name (SALESDB) is highlighted in the central white box and click on **Open**.

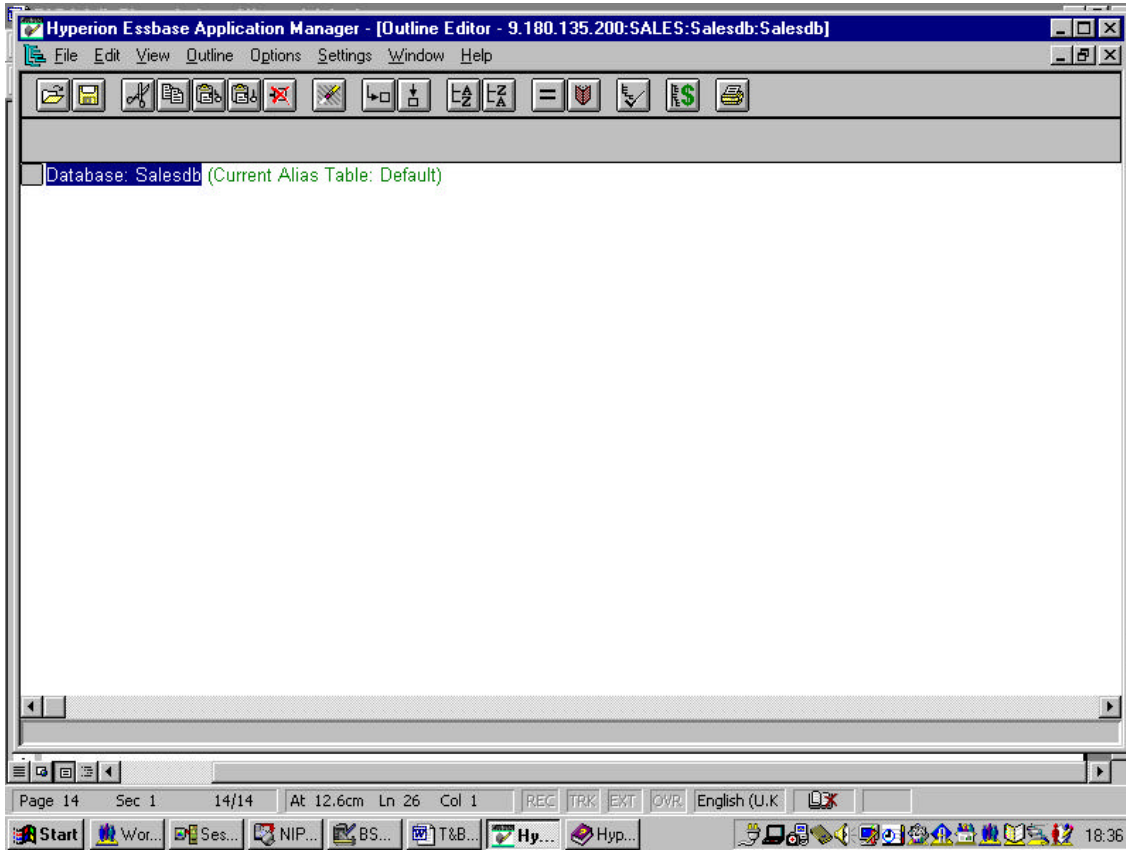
You are then taken into the Outline Editor. This is where you define your OLAP database. Within the Outline Editor you need to define your dimensions and hierarchies and measures.

We are going to have five dimensions in the SALESDB outline.

1. We will only load year 2000 data into the OLAP cube from the TEAMxx/SALESPERF table. The first dimension will therefore be the hierarchy of months into quarters into the year 2000.
2. The second dimension will be Sales Areas and will basically contain the sales organisation structure held in the TEAMxx/SALESORG table.
3. The third dimension will be Products and will contain the product hierarchy from the TEAMxx/PRODUCTS table.
4. The fourth dimension will be called Scenario and will be used to differentiate between Actual and Forecast data.
5. The fifth dimension will be Measures. In this example you will load sales revenue into measures and also create some calculated members.

We will define the year 2000 dimension, the Scenario dimension and the Measure dimension manually. We will use the SALESORG and PRODUCTS tables themselves to define the Sales Areas and Products dimensions.

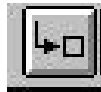
At this stage your Outline will be empty and will look like this:



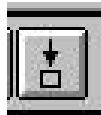
The first task is to manually enter the Year 2000 dimension.

You will need to use two buttons on the main task bar.

This is the Add Child button.



This is the Add Sibling button.



### 3.1 Create the Year 2000 Dimension

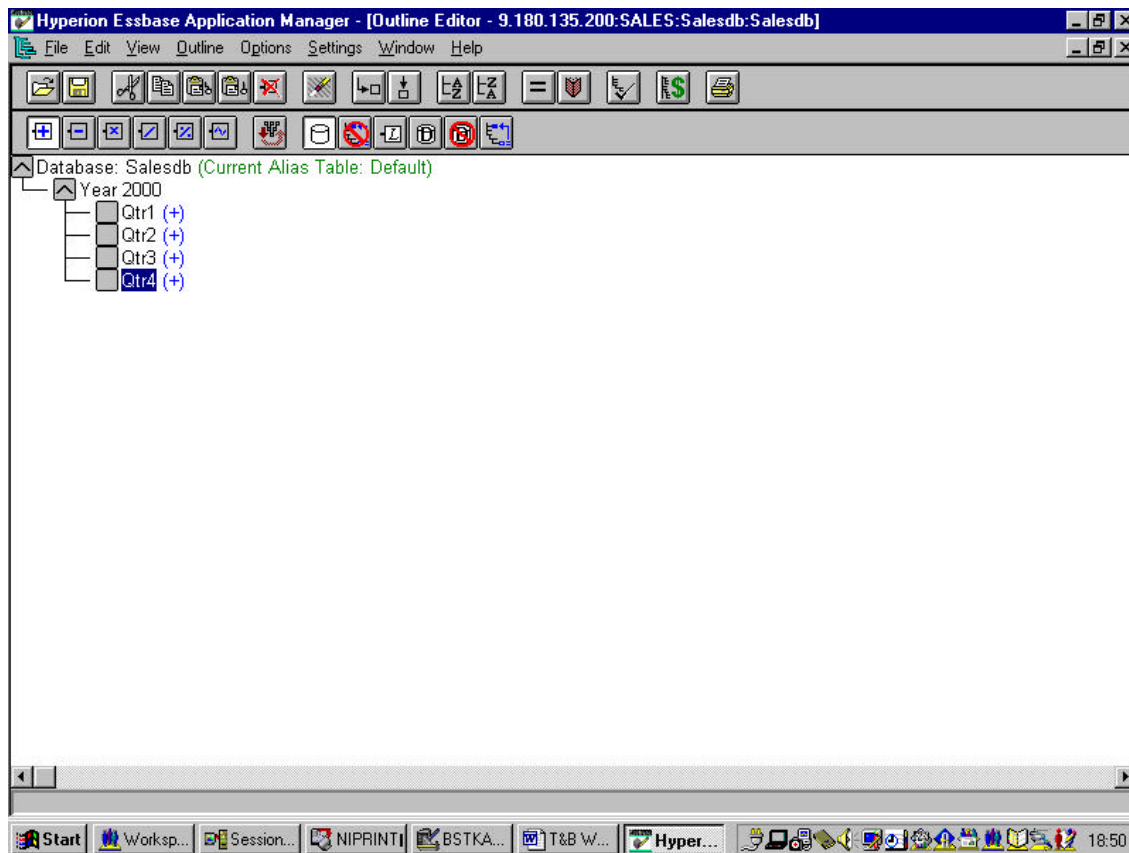
The top level of our structure (called Database: Salesdb) will be highlighted. Click on the **Add Child** button and type in **Year 2000** and press the enter key. You will be asked to confirm that you do wish to add this new member. Click on **Yes** to confirm that you do and then press the enter key.

You will not want to confirm every time you add a new member so we will now switch this off. From the main menu select **Options!Confirmation**, uncheck the **Add Member** box and click on **OK**.

With the member Year 200 highlighted, click on the **Add Child** button and type in **Qtr1**. Press the enter key.

Application Manager will assume you might want to add in another sibling at this point, so type in **Qtr2** and press enter. Repeat for Qtr3 and Qtr4. After Qtr4 you will need to press enter twice.

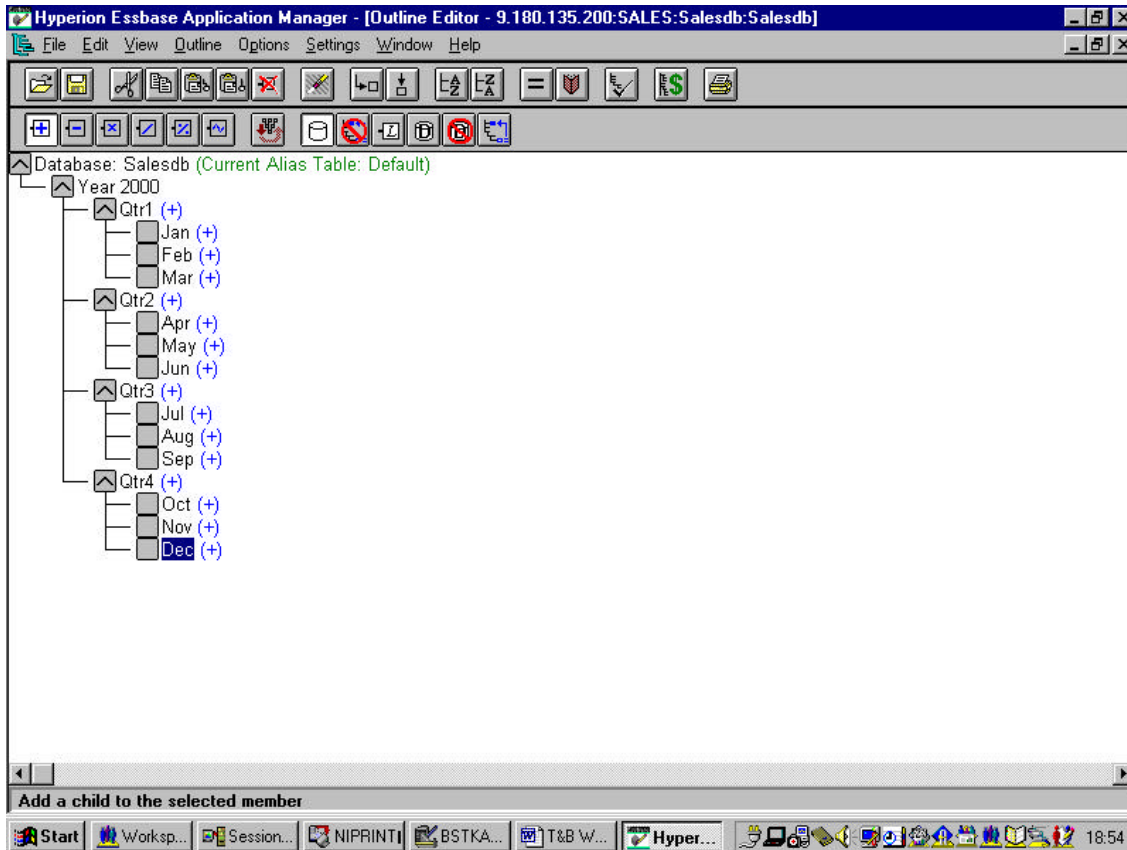
At this stage your Outline will look like this:



Highlight Qtr1 and click on the **Add Child** button. Type in the letters **Jan** and press enter. Application Manager will assume you want to add another sibling so type in **Feb** and press enter. Repeat for **Mar**. Having entered Mar press the enter key twice.

Now complete entering the other nine months for the other three quarters. In each case shorten the month name to three letters. When you have finished your Outline will look like this:





This completes the design of the first dimension.

Select **File!Save** from the main menu. Close the **Outline Editor** window once the database has been restructured.

### 3.2 Create the Products Dimension

The Products dimension is going to be defined using the structure that we already have in our table TEAMxx/PRODUCTS. We will create a **Rules File**, in order to create the dimension.

Looking at the row of four buttons on the main Application Manager window, click on the **Data Load Rules** button. You will see the text at the top of the central white box change to 'Data Load Rules'. Click on **New** to create a new rules file.

You will be presented with a **Data Prep Editor** window. At the top of this window you will see 18 buttons. The 13<sup>th</sup> and 14<sup>th</sup> buttons look like this:



These two buttons determine whether you are creating a rules file for building a dimension, or a rules file for loading data. Click on the second of these two buttons to specify that we are creating a rules file

for building a dimension. (When you position your mouse arrow over this second button you will see the words ‘View Dimension Building Fields’ appear at the bottom left of your screen).

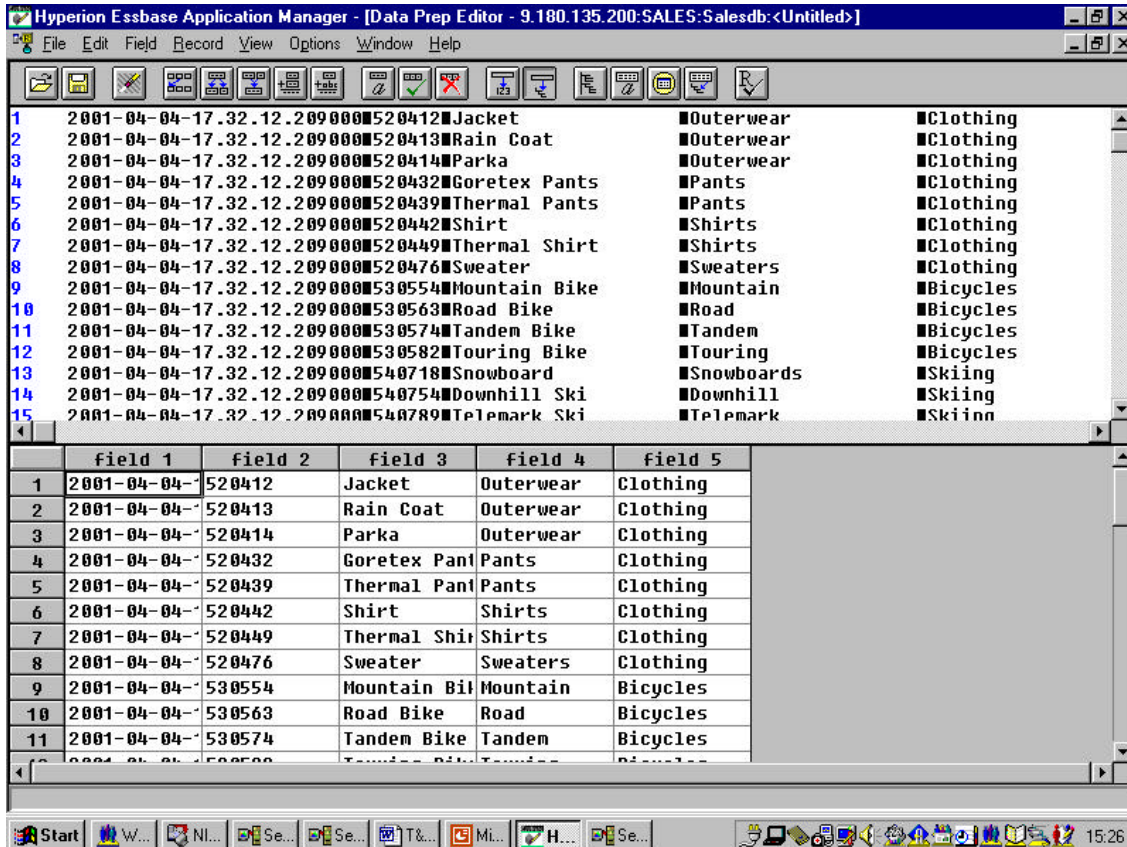
Take your mouse arrow to the top left of your screen and click on **File!Open SQL** and you will be presented with the **Select the Server, Application and Database** window. The name or IP address of your iSeries server should appear in the Essbase Server box. The Essbase Application box should display your application name of SALESxx and the database SALESDB should appear in the Essbase Database box.

Click on **OK** and you will be presented with the Define SQL Window. This is used to tell the system where to find the data that we are going to use as the basis for defining the Products dimension.

Ensure that your iSeries system (name or IP address) is highlighted in the **SQL Data Sources** box.

At the bottom half of this window you will see where you can enter an SQL SELECT statement. There will already be a “\*” in the first box. Take your cursor to the next box to enter the FROM part of the SELECT statement. Type in TEAMxx.PRODUCTS (you need to use the SQL naming convention rather than the iSeries system naming convention). Click on **OK/Retrieve**. When you get the SQL Connect window enter your iSeries userid and password (this userid must have authority to your tables in your TEAMxx library). Click on OK and the data will be brought into your Application Manager **Data Prep Editor** window.

Your screen (with a maximised Data Prep Editor window) should now look like this:



You can see that 'field 1' now contains the OLAP Builder generated timestamp, 'field2' contains the Product ID, 'field 3' contains the Product Description, 'field 4' contains the Product Type and 'field 5' contains the Product Group.

We must now link this data with the Outline that we started creating. Click on **Options\Associate Outline**, check the details and click on **OK**.

We must now provide the rules on how to use this data to build this dimension

Click on **Options\Dimension Build Settings**. You will be presented with the **Dimension Build Settings** window. Click on the **Dimension Definition** tab. Click on the **Rules File** radio button and in the **Name** box type in the name of the dimension which is **Products** and click on **Add**.

Click on **Properties** to define the properties for this dimension.

Click on the **Dimension Properties** tab header so that the Dimension Properties tab is on view. In the configuration section click on the **Sparse** radio button. Click on **OK** to go back to the **Dimension Build Settings** window. Stay in this window and click on the **Dimension Build Settings** tab. In the **Dimension** box click on **Products** if it is not already highlighted and in the Build Method section click on the **Use level References** radio button. Click on **OK**.

Having defined the general characteristics, we need now to go and tell the system what to do with each field. We have specified that we are going to use Level References for this dimension, so we now need to define the hierarchy in terms of levels. i.e. Product is level 0, Product Type is level 1 and Product Group is level 2.

You should now be back at the **Data Prep Editor** window.

Click on the column heading 'field 1' to select that entire column and click on **Field\Properties** on the menu at the top of the screen. This is just the timestamp field and we don't want to load this into the OLAP cube. Check that you are in the **Global Properties** tab, and check the checkbox that says **Ignore field during dimension build**. Click on **OK**.

You should now be back at the **Data Prep Editor** window.

Click on the column heading 'field 2' to select that entire column and click on **Field\Properties** on the menu at the top of the screen. Click on the **Dimension Building Properties** tab. We need to tell the system that the data in this field (i.e. Product Id) represents level 0 of our hierarchy. In the **Field Type** section click on **Level**. In the **Number** box type in the number zero. In the **Dimension** box click on **Products**. Click on **Next**.

You can now enter the details for field 3 which is the product description. Product description is still level 0 in the hierarchy, but we need to define it as an alias for the Product ID. Click on **Alias** in the **Field Type** section. Check that the number zero is in the **Number** box and check that **Products** is in the **Dimension** box. Click on **Next**.

You can now enter the details for field 4 which is the Product Type. Click on **Level** in the **Field Type** section. Check that the number one is in the **Number** box and check that **Products** is in the **Dimension** box. Click on **Next**.

You can now enter the details for field 5 which is the Product Group. Click on **Level** in the **Field Type** section. Check that the number two is in the **Number** box and check that **Products** is in the **Dimension** box. Click on **OK**.

You have now completed the definition of the Products dimension. Before saving this rules file we can check that everything has been defined correctly by clicking on the rules verification button. This is the last button in the row of 18 buttons. It has a tick and the letter 'R' on it. Click on this button and you should get a message telling you that the rules file is correct for dimension building. Click **OK** in this message window.

Click on **File!Save As** to display the **Save Server Object** window. In the location box check that the **Server** radio button is checked. In the Object Name box enter the name **PRODLOAD**. Click on **OK**.

Close the **Data Prep Editor** window.

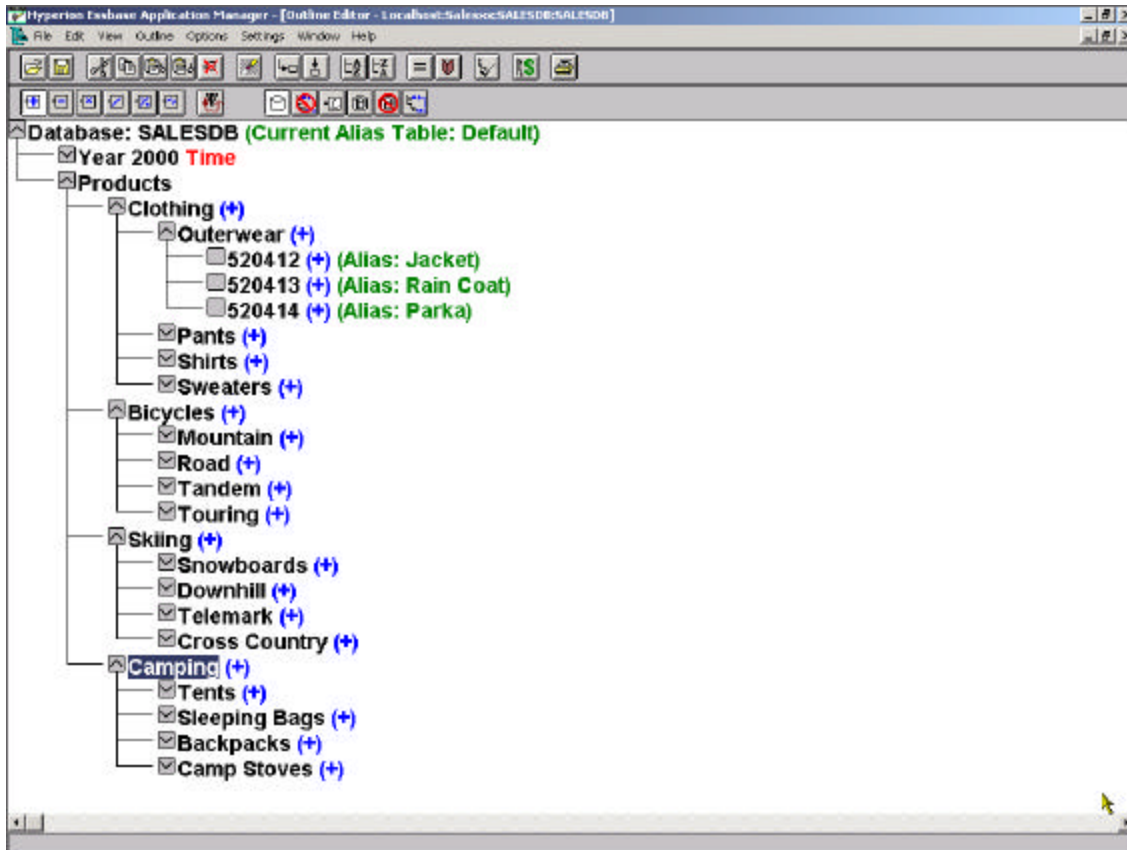
We can now update the Outline with the Products dimension.

On the **Hyperion Essbase Application Manager** main window click on the Outline button (the first one of the four buttons on the screen). You will see the words at the top of the white box change to Database Outlines. Click on **Open**.

Maximise the **Outline Editor** window. Click on **File!Update Outline**. You will see the **Outline Update** window. Click on **SQL** in the Data section and click on the **Find** button in the Rules section to find the rules file that we are going to use to update the outline. Ensure that the **Server** radio button is selected. Click on **PRODLOAD** in the list of objects so that **PRODLOAD** is in the **Object Name** box and click on **OK**, and click on **OK** again in the Update Outline window. Enter your iSeries userid and password in the **SQL Connect** window and click **OK**. The dimension Products should now appear on your outline.

Double click on the box to the left of Products to view the Product Groups. Double click on the box to the left of one of the Product Groups to view the Product Types. Double click on a box to the left of one of the Product Types to view the Products. At the lowest level (product) you will see the Alias names that we defined in the rules file displayed. The use of double click will also take you back up the hierarchy.

Your Outline will now look something like this (again depending on which levels of the hierarchy you have opened):



Select **File|Save** from the main menu. Do not close the Outline window.

### 3.3 Create the Sales Area Dimension

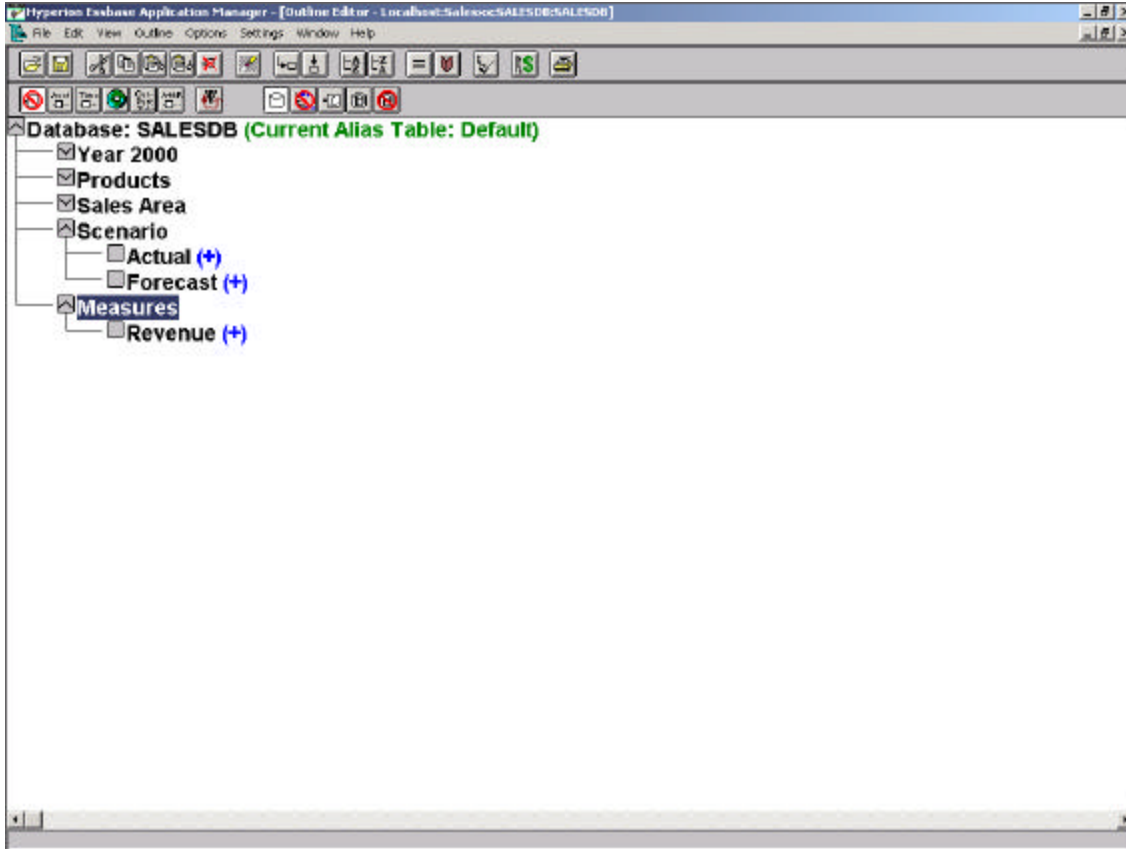
In the interest of time the sales area dimension has been created for you. Detail instructions can be found in the appendix at A.3. If you have the time you should use the instructions in the appendix rather than the instructions below.

Minimize your outline editor. Select **File|Open**. On your server open the “**SlsMstr**” application using the **SALESDB** database. The object type should read Outline. Click OK. Highlight **Sales Area** and select **Edit|Copy**. Maximize your original Salesxx|SALESDB outline, highlight **Products** and select **Edit|Paste Sibling**. Select **File|Save**. Your outline should now contain the Year 2000 dimensions, Products and Sales Area.

### 3.4 Create the Scenario and Measures Dimensions

These two dimensions are very small and can be created manually. Use the same method that was used to create the Year 2000 dimension to create Scenario and Measures. You will use the Add Child and Add Sibling buttons as before.

Create the Scenario and Measures dimensions as follows:



Scenario

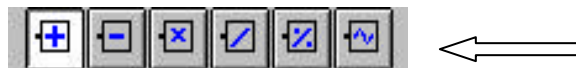
- Actual (+)
- Forecast (+)

Measures

- Revenue (+)

Note the addition signs (+) following the children of Scenario and Measures. By default OLAP Server will assume that the values for the children can be added together to form a total for the parent. This is clearly not always the case, and certainly there is no meaning in adding together the values for Actual Sales plus Forecast Sales.

We need therefore to remove the plus sign and replace with a tilde sign (~) to tell OLAP Server not to perform the addition. Click on the member **Actual**. On the bottom row of buttons at the top of the screen you will see a group of six buttons, the last of which is the tilde button.



Click on this tilde button and see the sign next to the Actual member change from + to ~.

Repeat for each of the members Forecast and Revenue.

In the same way we don't want to store any data values for the parent members Scenario and Measures. On the same row of buttons that we used before you will see that the 10<sup>th</sup> button along has the letter L on it. It looks like this:



Click on Scenario and then click on this Label Only button. You will see the words (Label Only) appear on the Outline. Repeat for Measures.

You now need to tell OLAP Server if any of your dimensions are 'special' dimensions. The Year2000 dimension is a Time dimension and the Measures dimension is an Accounts dimension. Click on the Year 2000 dimension. The group of six buttons at the start of the bottom row of buttons now looks like this:



The second one of these buttons is used to denote a dimension as being an Accounts dimension and the third one of these is used to denote a dimension as being a Time dimension.

Ensuring that the **Year2000** member is still highlighted, click on the **Time** button. Click on the **Measures** member and click on the **Accounts** button.

Much of the power of IBM OLAP Server comes from the sophisticated analytical functions that can be easily built into the OLAP database. We are going to add two more measures that will be dynamically calculated at run time and not input from a database. They will be Prior Month Revenue and Monthly Variance % which calculates the variance % difference between current month and previous month revenues.

Add a new sibling beneath the member Revenue and call it Prior Month Revenue. Click on the tilde button on the tool bar to change the plus sign that appears to the right of the newly added member from a plus to a tilde. This measure will not be stored and calculated in the OLAP Database, but will be dynamically calculated at retrieval time. To specify this we need to click on the Dynamic Calc Member button which is approximately in the middle of the bottom row of the toolbar.



Now we need to specify the formula for this function. On the top row toolbar you will see a button with an equals sign on it (fifth along from the right).



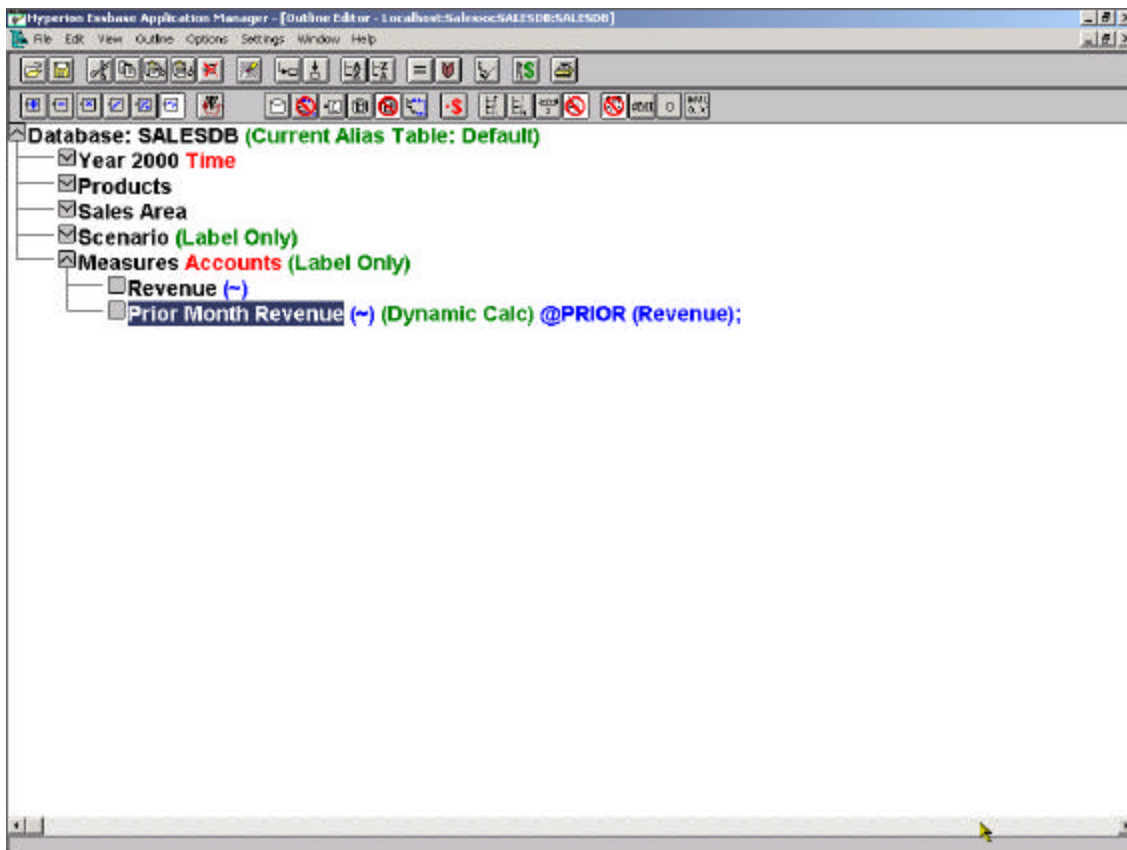
Click on this formula button and you will be presented with the **Formula Editor** window. You can either type in the formula directly or you can select a function from the list of functions. Click on the Select a function button (nine buttons along from the left)



And you will be presented with a **Function Templates** window. From the list of categories on the left click on **Range (Financial)**, and from the list of templates on the right click on **@PRIOR** and click on **OK**. Position your cursor between the open and close brackets and in the box of dimensions at the bottom of the screen click on **Measures**. You will see **Measures** appear in the **Members** box. Double-Click on the downward pointing arrow to the left of Measures (in the members box) and then click on **Revenue**. The member name of Revenue will now appear in the main formula editor box between the opening and closed parentheses. Enter a semi-colon at the end of the line to end the statement and then click the formula syntax checker which is the last button on the toolbar with a tick on it.



You should get a message at the bottom of the window that says: **No errors**. Click on **File!Close** and click on **Yes** to save the formula. Your outline should now look something like this:



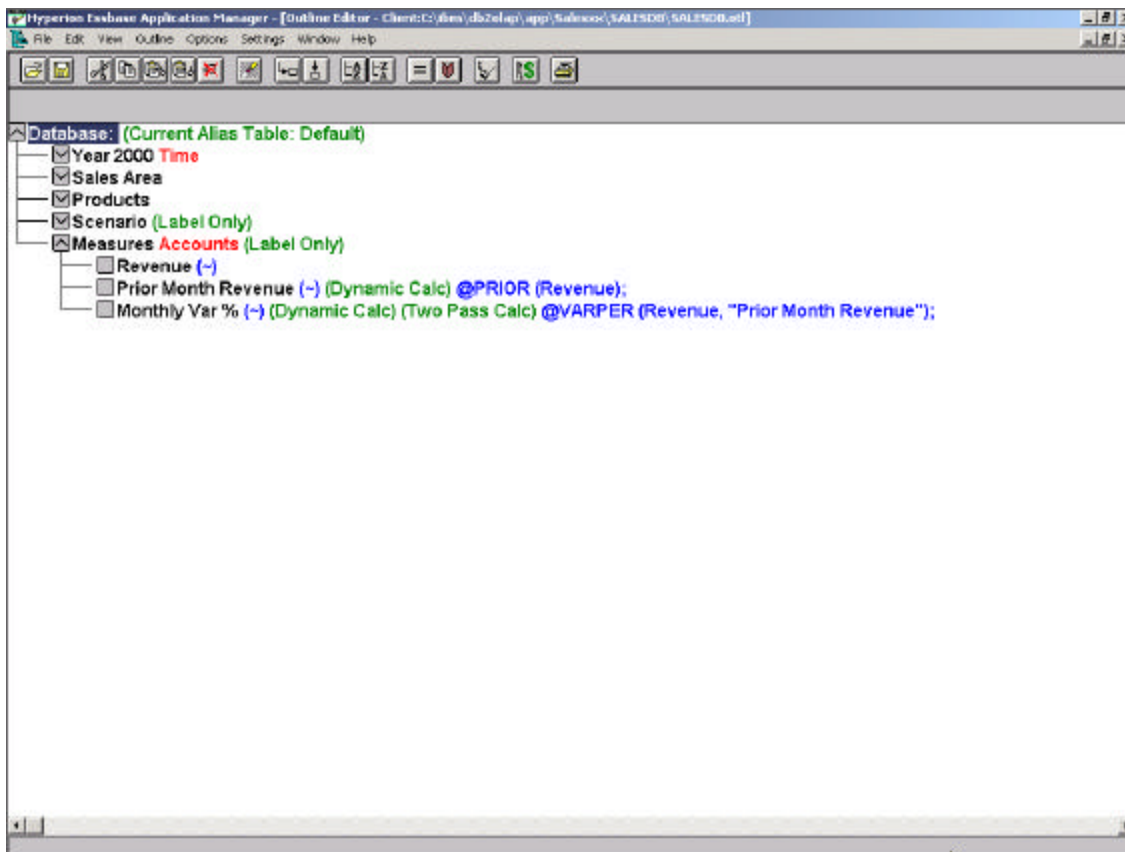


In your outline add **Monthly Var %** to your Measures. Change the plus sign to a tilde and specify dynamic calculation. The Month Var % member should be specified as being a two-pass calculation. The two-pass calculation button is the seventh one long on the bottom toolbar. Click on this button.



Having done this, click on the **Monthly Var %** member and click on the Formula button. In the **Formula Editor** window click on the Function button. From the list of categories on the left, click on **Math**, and from the list of templates on the right, click on **@VARPER**. Check the **Insert Arguments** checkbox and click on **OK**. Now we need to specify the two members that we want to calculate the variance between. Highlight **mbrName1** within the brackets, and in the box of dimensions at the bottom of the screen click on **Measures**. You will see **Measures** appear in the **Members** box. Double-Click on the downward pointing arrow to the left of Measures (in the members box) and then double-click on **Revenue**. The member name of Revenue will now be inserted into the formula. Highlight **mbrName2** within the brackets, and in the members box at the bottom of the window double-click on **Prior Month Revenue**. The member name of Prior Month Revenue will now be inserted into the formula. Insert a semi-colon to enter the formula. Verify the syntax of the formula and close and save the formula.

Your outline should now look like this:



Select **File!Save** from the main menu. Close the **Outline Editor** window after the database has been restructured.

### 3.5 Define the Sales Data Load

The final part of the definition stage is to define how the data is going to be loaded.

The data is to be loaded from the TEAMxx/SALESPERF table that we created using OLAP Builder. You will also use TEAMxx/DATE\_TBL to pick up the month name. The definition for the data load will be held in a **Rules File**.

In Application Manager looking at the row of four buttons on the main window, click on the **Data Load Rules** button. You will see the text at the top of the central white box change to 'Data Load Rules'. Click on **New** to create a new rules file.

You will be presented with a **Data Prep Editor** window. At the top of this window you will see 18 buttons. The 12<sup>th</sup> and 13<sup>th</sup> buttons look like this:



This time ensure that the first of these two buttons is selected to specify that we are creating a rules file for performing a data load. (When you position your mouse arrow over the first button you will see the words 'View Data Load Fields' appear at the bottom left of your screen).

Take your mouse arrow to the top left of your screen and click on **File!Open SQL** and you will be presented with the **Select the Server, Application and Database** window. The name or IP address of your iSeries server should appear in the Essbase Server box. The Essbase Application box should display your application name of SALESxx and the database SALESDB should appear in the Essbase Database box.

Click on **OK** and you will be presented with the Define SQL Window. This is used to tell the system where to find the data that we are going to use as the basis for defining the Products dimension.

Ensure that your iSeries system (name or IP address) is highlighted in the **SQL Data Sources** box. Your system name will be different from the one below.

At the bottom half of this window you will see that you can enter an SQL SELECT statement. Instead of retrieving all the fields from the two files, this time you will specify the 5 fields that you need to use for the data load. In the **Select** box enter (without the quotes) “area\_id, productid, tmonth, fcast\_sales, act\_sales”.

You will be accessing DATE\_TBL to pick up the 3 character month name (Jan, Feb...). You could have avoided the join and told the rules file to replace all fields with 01 in PERIOD with JAN, 02 with FEB etc. Since this data already exists in the DATE\_TBL you will save yourself some keying by using a table join instead. In the **From** box you should enter “teamxx.salesperf salesperf inner join teamxx.date\_tbl date\_tbl on salesperf.period=date\_tbl.period”.

In the **Where** box specify “tyear=2000”.

Click on **OK/Retrieve**.

When you get the SQL Connect window enter your iSeries userid and password. Click on OK and the data will be brought into your Application Manager **Data Prep Editor** window.

You must now link this data with the Outline. Click on **Options!Associate Outline**, check the details and click on **OK**.

You must now provide the rules on how to use this data to perform the data load.

Click on **Options!Data Load Settings**. You will be presented with the **Data Load Settings** window. Click on the **Header Definition** tab. You are going to load the actual and forecast numbers in the same

pass. You need to tell Application Manager that the actual and forecast numbers refer to the Revenue member in the Measures dimension.

On the **Header Definition** tab, click on **Measures** in the **Dimension** box. In the **Member** box expand Measures. Click on **Revenue** and you will see Revenue appear in the **Header Name** box. Click on **OK**.

Click on the first column heading, **AREA\_ID**, to select that entire column and click on **Field Properties** on the menu at the top of the screen. You must now associate this field with a member in our outline. Click on the **Data Load Properties** tab. In the **Dimension** box click on Sales Areas. In the **Member** box click on Sales Areas. You will see the text in the **Field Name** box change to "Sales Areas"AREA\_ID. Delete the text AREA\_ID so that the only text remaining on the **Field Name** box is the dimension name "Sales Areas". Click on **Next**.

In the **Dimension** box click on Products. In the **Member** box click on Products. You will see the text in the **Field Name** box change to ProductsPRODUCTID. Delete the text PRODUCTID so that the only text remaining on the **Field Name** box is the dimension name Products. Click on **Next**.

In the **Dimension** box click on Year 2000. In the **Member** box click on Year 2000. You will see the text in the **Field Name** box change to "Year 2000"TMONTH. Delete the text TMONTH so that only the text "Year 2000" remains in the **Field Name** box. Click on **Next**.

In the **Dimension** box click on Scenario. In the **Member** box click on Forecast. You will see the text in the **Field Name** box change to ForecastFCAST\_SALES. Delete the text FCAST\_SALES so that only the text Forecast remains in the **Field Name** box. Click on **Next**.

In the **Dimension** box click on Scenario. In the **Member** box click on Actual. You will see the text in the **Field Name** box change to ActualACT\_SALES. Delete the text ACT\_SALES so that only the text Actual remains in the **Field Name** box. Click on **OK**.

You have now completed the definition of the Sales dataload. Before saving this rules file we can check that everything has been defined correctly by clicking on the rules verification button. This is the last button in the row of 18 buttons. It has a tick and the letter 'R' on it. Click on this button and you should get a message telling you that the rules file is correct for data loading.

Click on **File Save As** to display the **Save Server Object** window. In the location box check that the **Server** radio button is checked. In the Object Name box type in the name **DATALOAD**. Click on **OK**. Close the **Data Prep Editor** window.

### 3.6 Load the Sales data

We are now going to load the data into the OLAP cube (which resides on the iSeries).

On the **Hyperion Essbase Application Manager** window click on **Database Load Data**. You will be presented with a **Data Load** window. In the **Type** section select the **SQL** radio button. Enter your iSeries userid and password in the SQL User and Password boxes. Check the Use Rules checkbox, and click on the **Find** button. You will be presented with the **Open Server Rules Object** window. Select

DATALOAD from the list of objects so that DATALOAD is displayed in the **Object Name** box. Click on **OK** and click **OK** again on the **Data Load** window.

You should get a message telling you that the SQL Dataload was successful. Click on **OK**.

The OLAP cube is now fully populated with the low level data. We will need to run a calculation routine to populate the higher level data

### **3.7 Calculate**

On the **Hyperion Essbase Application Manager** window click on **Database\Calculate**. You will be presented with a **Calculate Database** window. Select the default calculation and click on **OK**.

The OLAP cube is now fully populated and all combinations have been calculated.

Close the **Hyperion Essbase Application Manager** window.

### **3.8 Optional**

The loading and calculating of the OLAP cube can be automated and scheduled via IBM DB2 OALP Builder. You just manually loaded and calculated your SALESDB cube. An alternative would have been to use the procedure below. If you want to try this, it will still work as the first step is to clear all existing data in your cube and then start loading the data from the beginning.

## **OLAP Builder**

OLAP Builder can also be used to schedule and run the building of dimensions using rules files, the loading of data using rules files and the calculating of the data in the cubes. Start OLAP Builder and create a Warehouse Builder Set called "TEAMxx SalesDB Load and Calc". Select Essbase. Select your application and database. Add the following Essbase functions:

Clear Data:

SQL Load: DATALOAD

Calc Script:

Select OK.

Notice that you could also have built the dimensions this way. **RUN** the Builder Set you just created. To see your job messages select **STATISTICS**. Refresh until status shows "Successful with Info". Select **MESSAGES**. Check the box next to Operations (all 3 should be checked now) and select **REFRESH**. You should now be able to see your data load and calculation completion messages.

Congratulations, you have now completed the OLAP Server section of this lab. Clearly, only the basics of OLAP Server have been covered here – but hopefully you now have an appreciated of what is involved in building a simple OLAP cube.

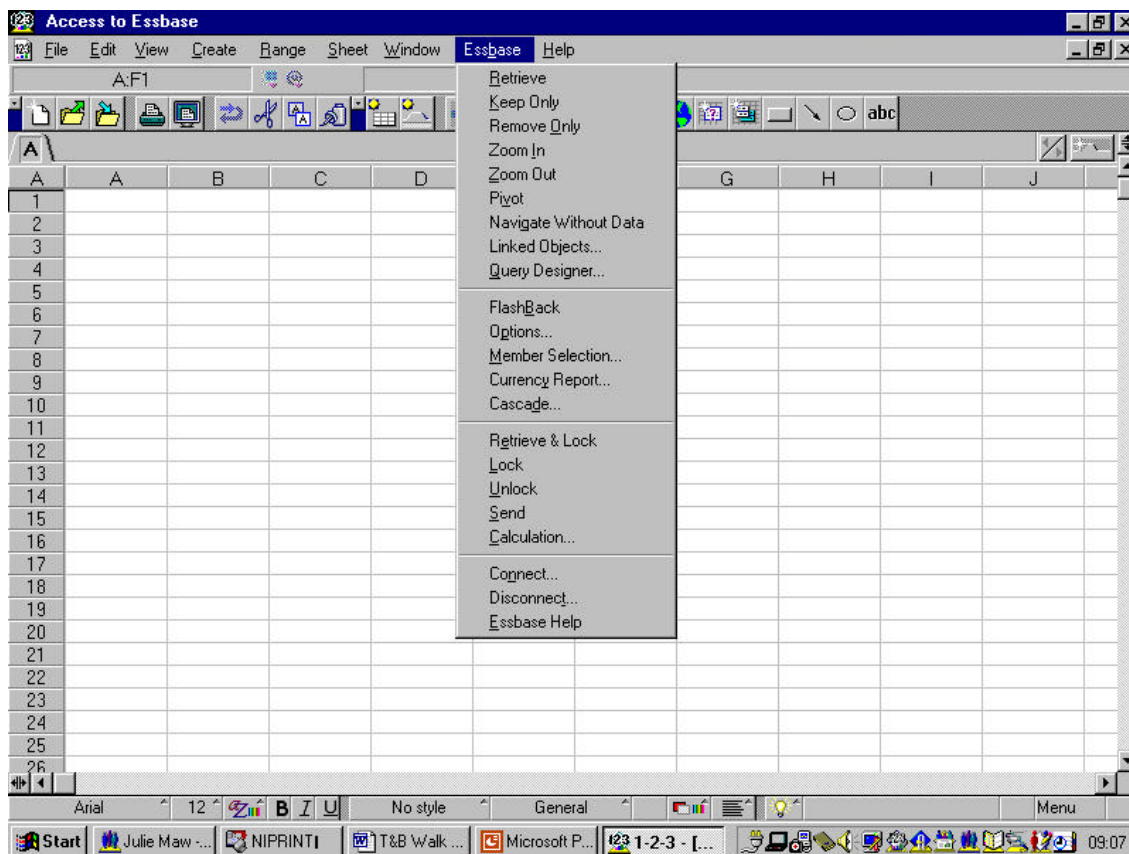
Your datamart is now available to be queried by many applications. The two that you will use in the following sections are first a spreadsheet application (Lotus 123 or MS Excel) and secondly IBM Analyzer. IBM Analyzer tool is fully integrated with OLAP Server and provides a full-function graphical end-user query and reporting tool. Other tools that could have been used to query this data include ShowCase Query and Report Writer, BrioQuery, Business Objects, Cognos PowerPlay, Alphablox and many more.

## 4.0 Spreadsheet Add-In Client

The Spreadsheet Add-In feature is part of OLAP Server. This section of the exercise will assume that you already have skills in either Lotus 123 or Microsoft Excel, so once you have been able to successfully load the data from OLAP Server into your spreadsheet, then you can utilise your own skills in order to manipulate the data.

First of all start up either Lotus 123 or Microsoft Excel and take the option to create a new spreadsheet.

Looking at the list of menu options at the top of the screen you will see that there is now an Essbase menu option. When you click on **Essbase** your spreadsheet will look like this (the example below is from Lotus 123):



The first step is to connect to your OLAP application. From the list that you see above click on **Connect**. You will be presented with an **Essbase System Login** window.

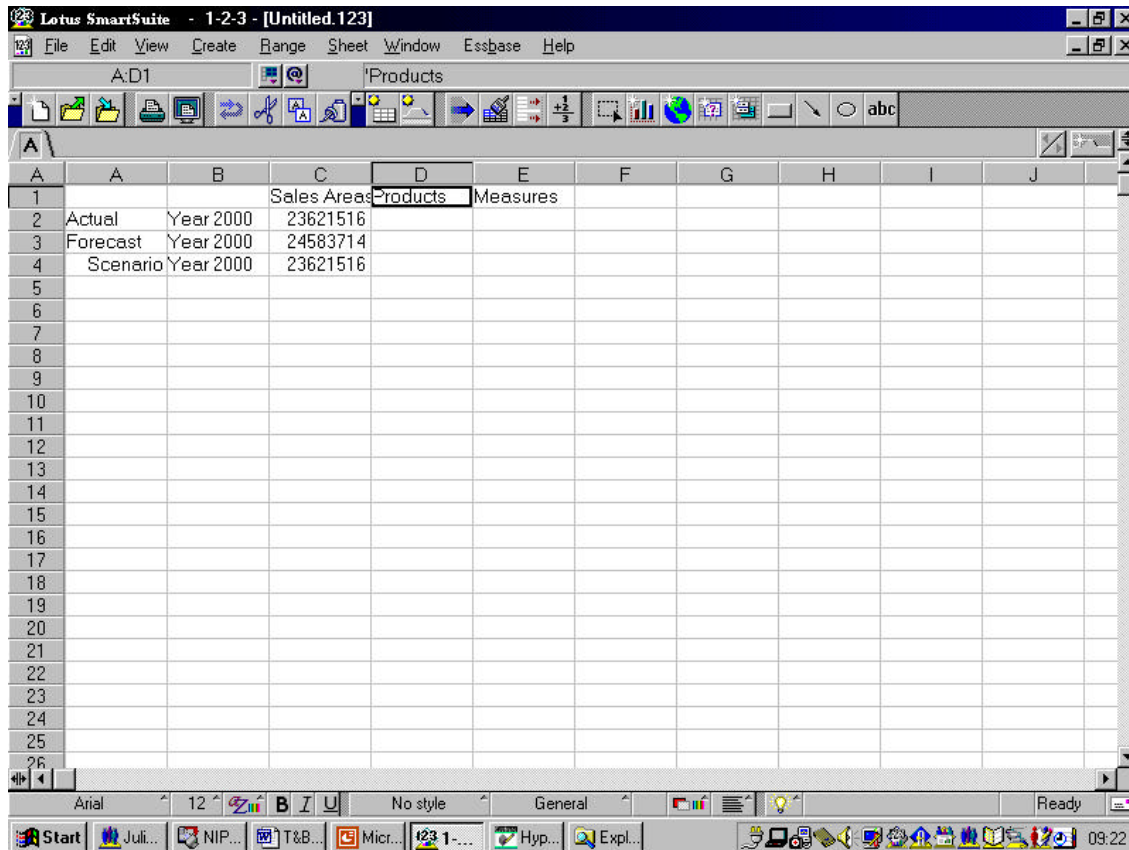
Note: If you are ever missing an option from a menu drop down list, look for the double down arrow at the bottom of the list. This will expand the list to include all the options available, not just the most recently used options.

Check that the IP address or the name of your iSeries server is in the **Server** box. In the **Username** box enter your OLAP Server userid and in the **Password** box enter your OLAP Server password. Click on

**OK.** In the **Application/Database** box select the name of your application and database, which should be SALESxx and SALESDB. Click on **OK**.

From the main menu click on **Essbase!Retrieve**. You will see your dimensions Sales Areas, Products, Scenario and Measures along the top and the dimension Year 2000 down the side.

Double click on the cell that contains the dimension name Scenario (probably cell D1). Your screen should look like the one below.



We have successfully drilled-down one-level into the Scenario dimension to show the actual and forecast sales figures for all sales areas and all products. Click on the cell that contains the word Scenario (cell A4 in the example above), click on **Essbase!Remove Only** to remove the total and just leave the two actual and forecast figures.

Double click on the cell that contains the words Sales Areas to drill down to the Sales Divisions. Then click in the cell that contains the words North America (probably cell A2) and use your right mouse button to drag the cell to just below the cell that contains the words Products. Very quickly you can pivot dimensions on the cross-tab.

Click in the cell that contains the word Europe and click on **Essbase!Keep Only**. Then drill down again to show the Sales Regions within the Europe Division. Your spreadsheet should now look like this:



The screenshot shows a Lotus SmartSuite spreadsheet window titled '1-2-3 - [Untitled.123]'. The spreadsheet has columns labeled A through J and rows 1 through 26. The data is organized as follows:

	A	B	C	D	E	F	G	H	I	J
1			Products	Measures						
2			Belgium	Germany	France	United Kingdom	Europe			
3	Actual	Year 2000	1823881	320221	361875	304644	2810621			
4	Forecast	Year 2000	1676352	391348	467826	296913	2832439			
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										
15										
16										
17										
18										
19										
20										
21										
22										
23										
24										
25										
26										

If you wish to undo anything that you have done click on **Essbase\Flashback**.

Click on **Essbase\Options** and review the controls in this window. On the display tab you can choose to display alias names. So in our example we can display the Area Descriptions and the Product Descriptions instead of the Area Codes and Product Codes. In the zoom tab you can control what happens when you zoom in or out.

Try out the various options and use your spreadsheet skills to present the data in a chart/graph.

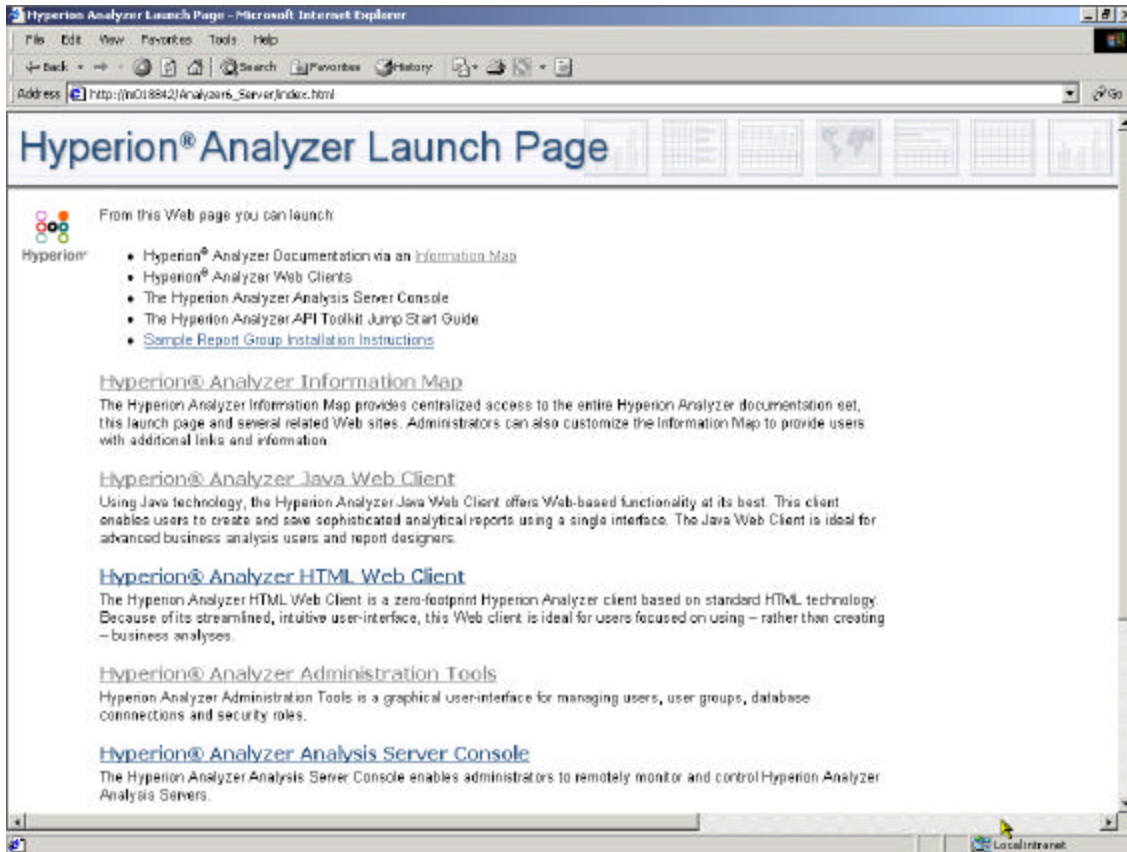
Ad hoc reporting also works. Insert a new spreadsheet and enter “Belgium”, “Germany”, “France” in a column starting in cell A3 through A5. Enter “Actual”, and “Forecast” going across starting in cell B2. Select **Essbase\Retrieve**. This should also populate the spreadsheet. There is also a query wizard under the **Essbase** toolbar to assist in data retrieval.

Notice that the cells in the spreadsheet contain actual values not formulas. This means that the users can disconnect and walk away and have a perfectly normal spreadsheet. Any calculations or graphing that they would usually use to analyze this data can be done on this spreadsheet. When you next attach up to the iSeries you can simply retrieve and update the data.

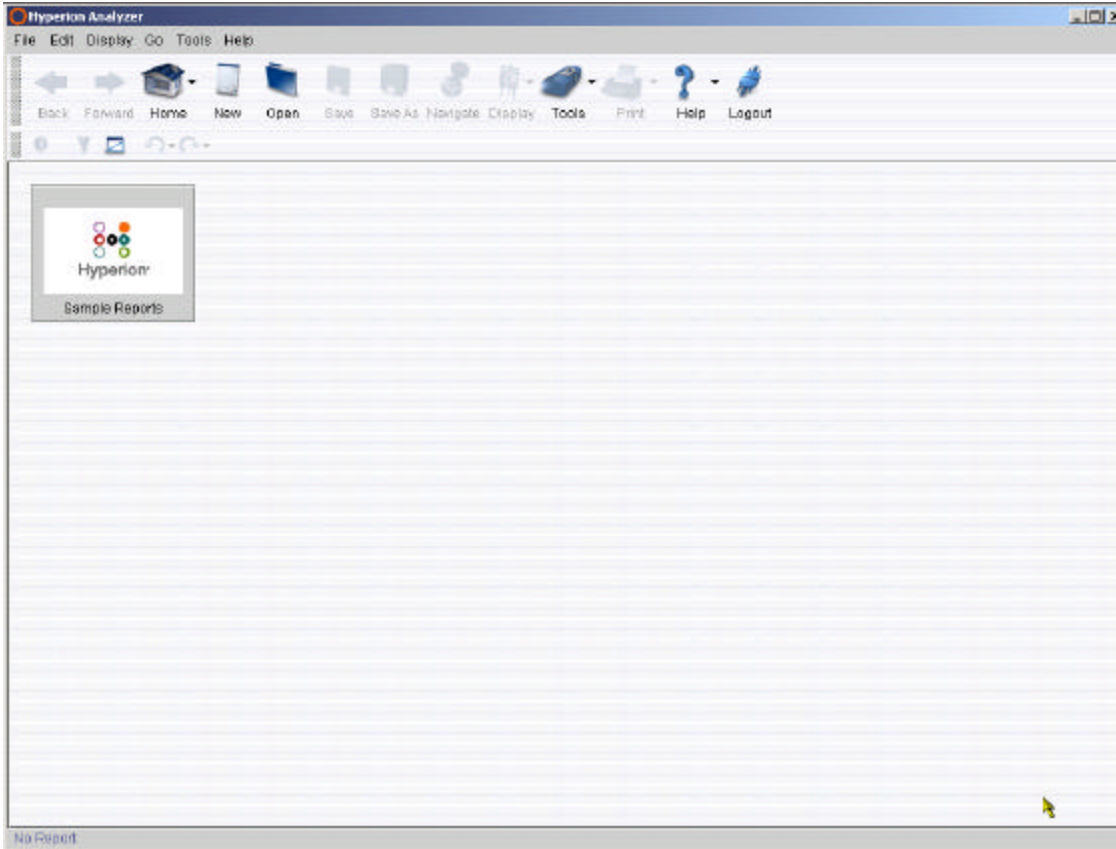
As you can see, users who are already skilled in using spreadsheets can very speedily exploit the power of the OLAP Server database.

## 5.0. IBM Analyzer

In this section you will create several Analyzer Reports to analyze the SALESDB that you just created. Open up MS Internet Explorer or Netscape. The URL for the Analyzer Server will be provided by the instructor.



Select the option to start the “Hyperion Analyzer Java Web Client”. You will then be prompted to sign on. Use the same userid and password that you used in previous exercises. Your screen should look like:

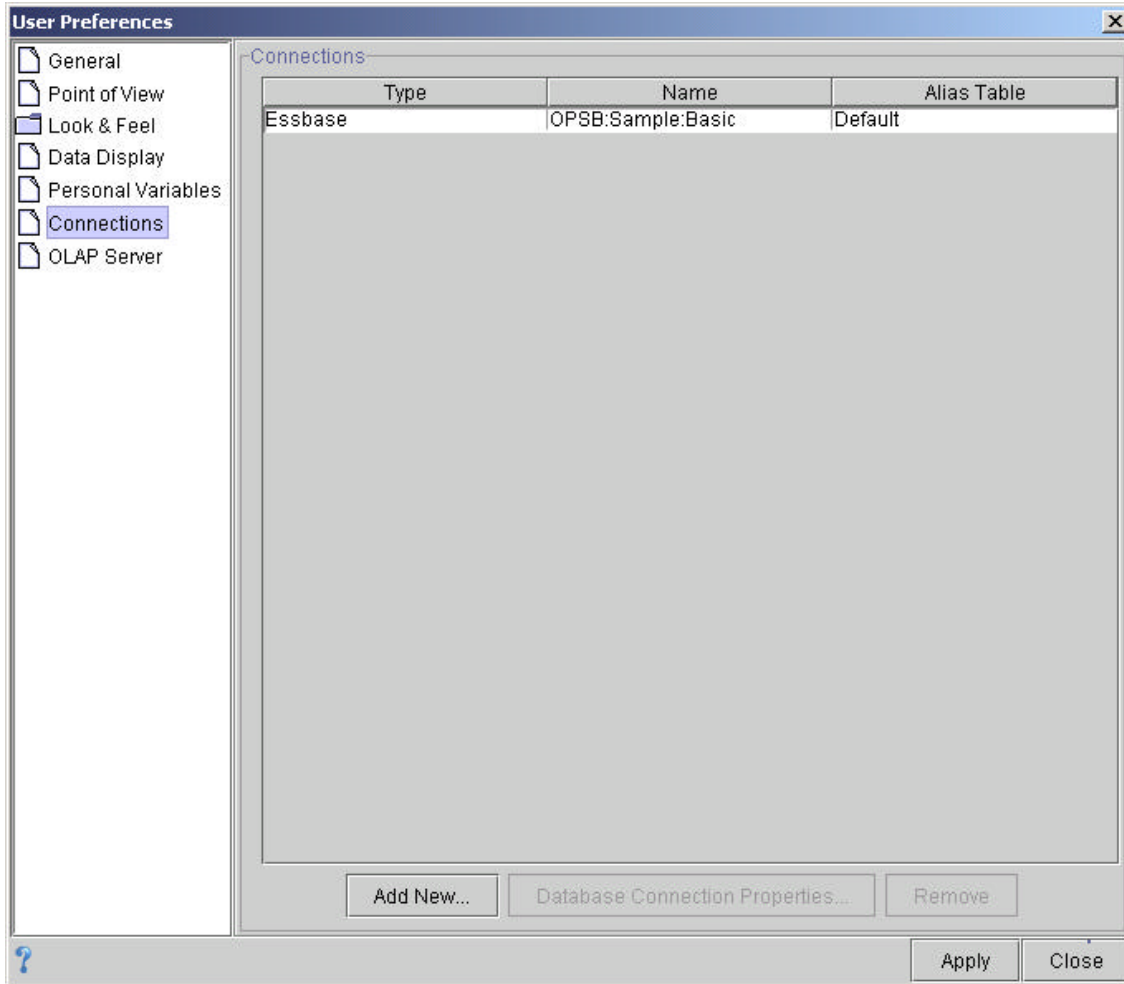


If you don't see the "**Sample Reports**" Group click on the **Home** icon. Browse the reports in the Sample Group to see the types of queries that you can easily create with IBM Analyzer. A script is provided in the "Getting Started" manual that leads you through each report and its features. After browsing through the different reports click on the **Home** icon to return to the home page.

You are going to create your own report group consisting of 3 basic reports. First you need to define the datamart or SALESDB cube that you just created to Analyzer.

### 5.1. Define SALESxx:SALESDB to Analyzer

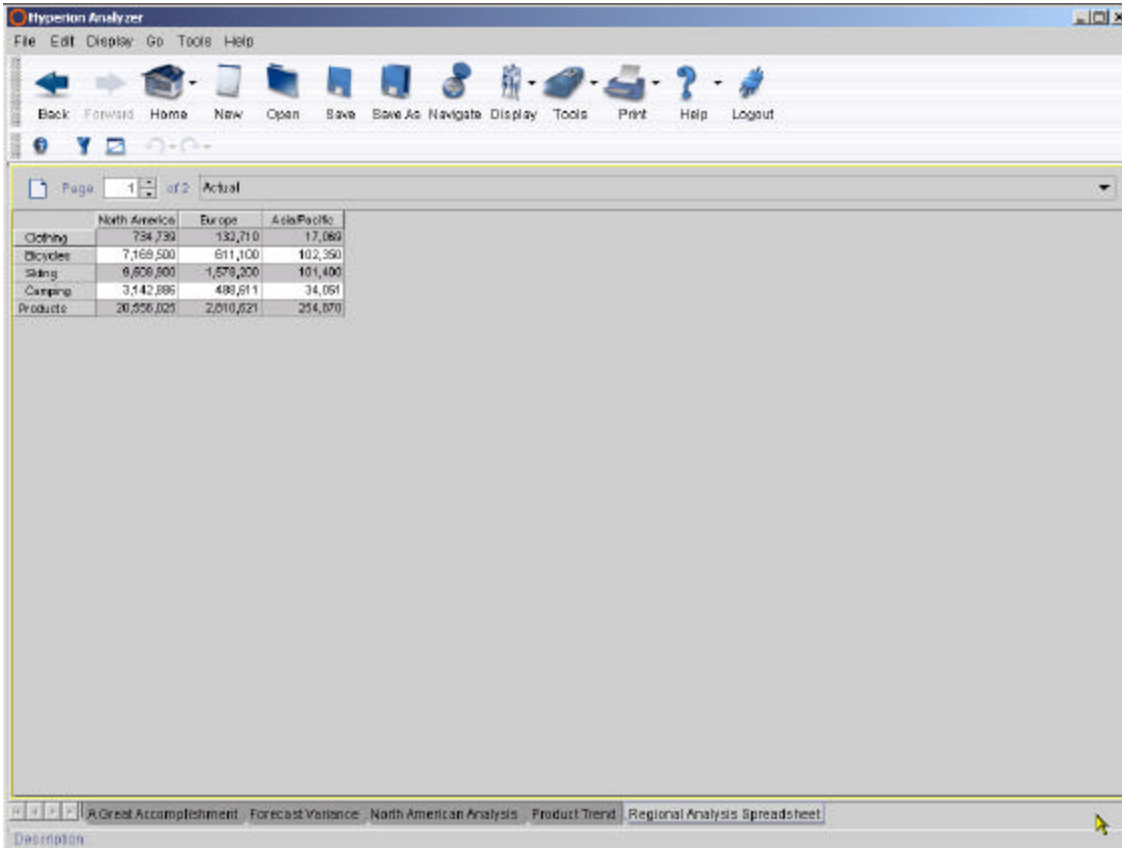
This step will often be performed centrally by the Analyzer administrator. Select **Tools|User Preferences|Connections**. You should see the following screen.



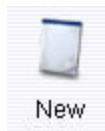
Click **“Add New...”** then select **“Essbase”** as your database type. Enter your server name (OPSB) and your userid and password. You should then see a list of datamarts or multidimensional databases available to you. Select the SALESDB that you created under Salesxx. If you didn’t complete the previous exercise choose SALESDB under the application “SlSMstr”. Press the green arrow to move the database from the **“Available Databases”** column to the **“Selected Databases”** column. Click on **OK** then **Close**.

You have now defined your DB2 OLAP cube to IBM Analyzer as one that you can create ad hoc queries against.

## 5.2. Creating Regional Analysis Reports



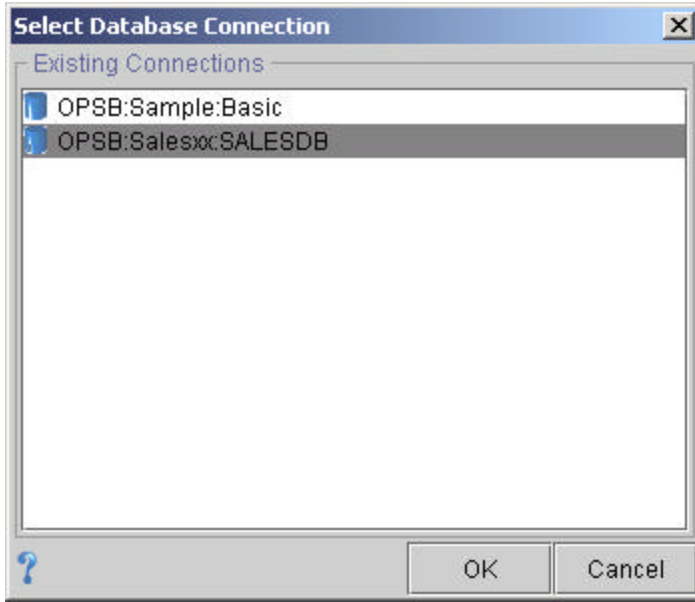
The first report that you will create should answer the questions: How much actual revenue did you sell, by Product Group, across every region and also what did you forecast to sell.



Select the icon **New**. It looks a bit like a clipboard.

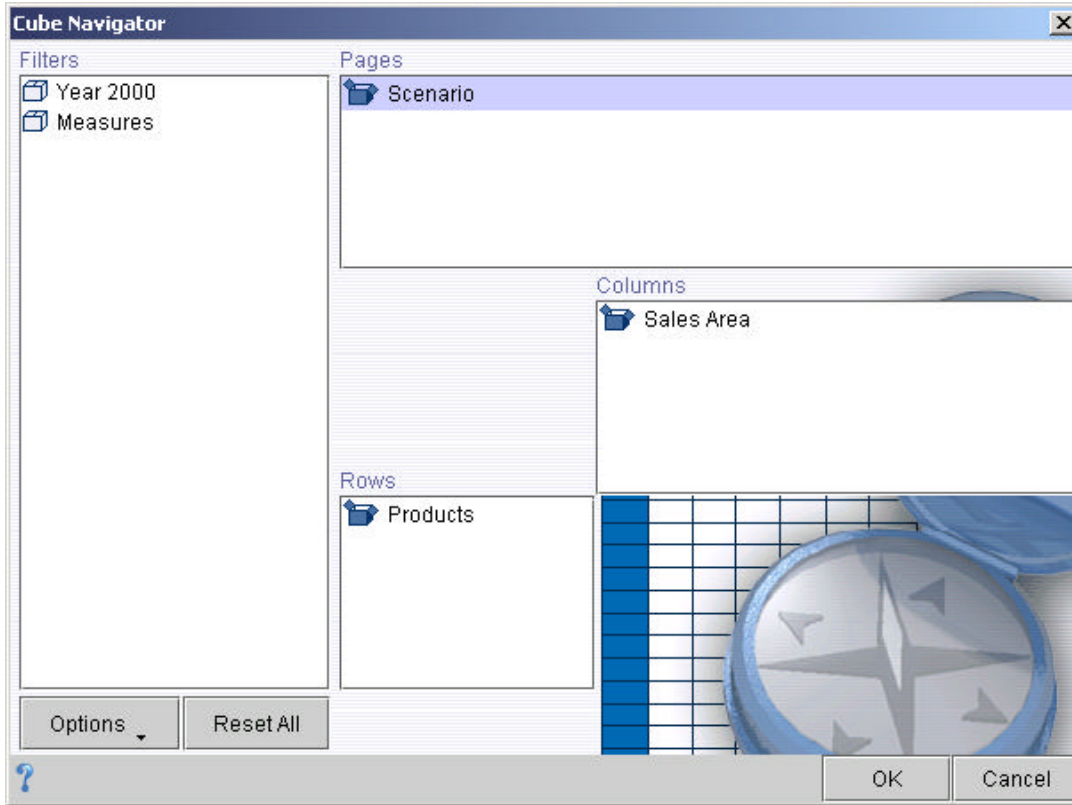
Select the default **Spreadsheet** and click on **OK**.

Select the **SALESDB** connection that you just defined and press **OK**.



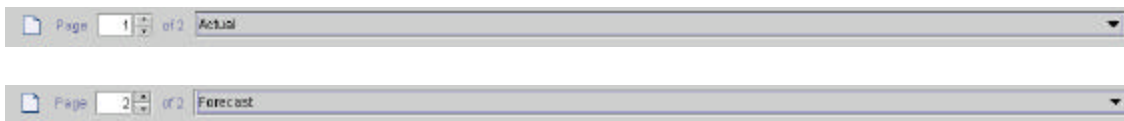
You may need to sign on to the DB2 OLAP database. You should now be in the cube navigator. Drag the **“Sales Area”** dimension to the **Columns** box. Expand (click on +) **Sales Area** one level. Click on **“North America”**, **“Europe”** and **“Asia/Pacific”**. This should put a check mark next to each member and move the name over to the **Selected** column. Ensure that **“Show Descriptions”** is selected and click **OK**. **“Show Descriptions”** should be your choice in all the reports you create today. Drag **Products** to the **Rows** box. This time, instead of individually selecting each of the children you should right click on **Products** and select the option **“Also Select Children”**. This will bring all Product Groups into the report even if new ones are added or deleted. With Sales Area you explicitly specified which Divisions to bring into the report. Using this explicit method, if you started selling in South America that data would not be included in this report. Click on **OK**.

Drag **Scenario** to the **Pages** box and expand **Scenario**. Check **Actual** and **Forecast**. Click **OK**. Your Cube Navigator window should look like this.



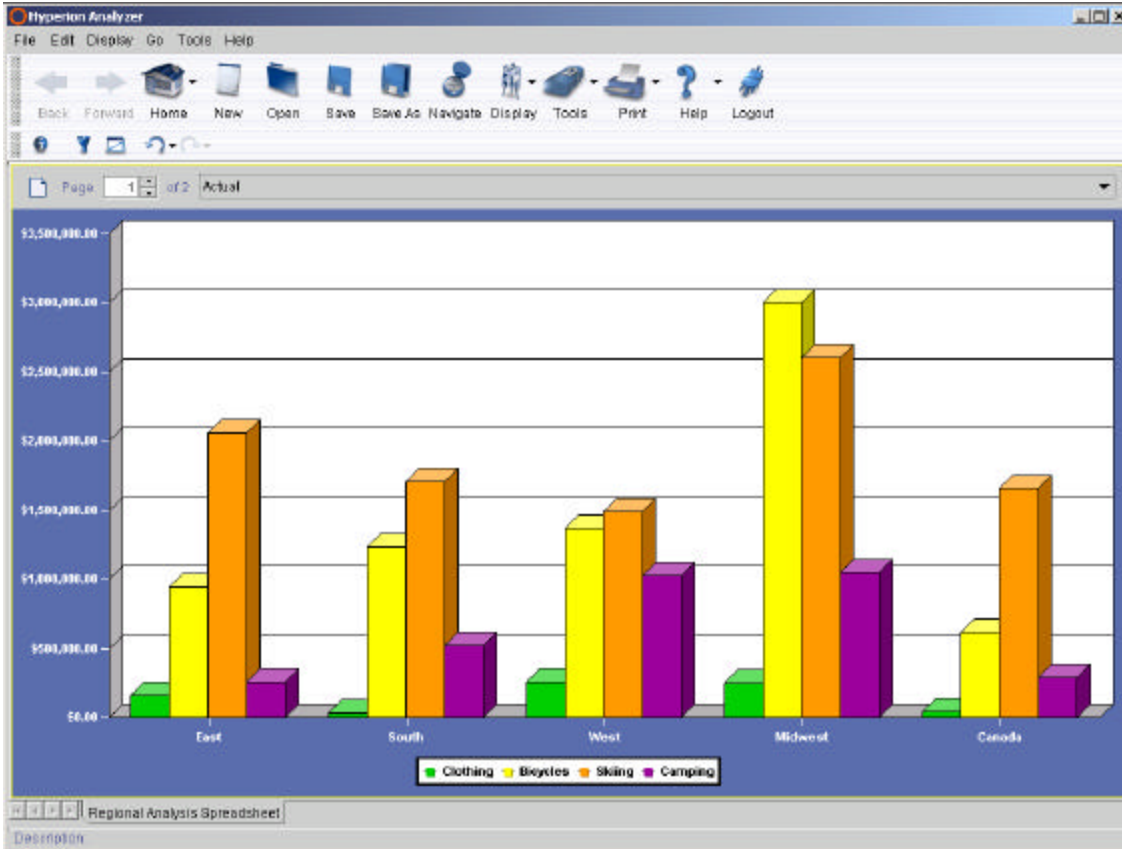
Click on **OK**.

You should now be looking at a spreadsheet result set that answers the original questions: How much actual revenue did you sell, by Product Group, across every region and also what did you forecast to sell. You can switch between **Actual** and **Forecast** revenue by using the Page Control line.



To save this report choose **File|Save As** then enter the **Report Name** “Regional Analysis Spreadsheet”. To create a new report group simply enter the **Report Group** name “Teamxx Analysis”. Click on **OK**.

While still viewing the spreadsheet select the icon **Display|Charts|Bar**. You should see the same data now represented as a bar chart. When displaying data in chart format we often want to eliminate the parent or total value as this greatly affects the scale of the chart. Right click on the chart and select **Data Display** and de-select “**Show Selected Member**”. Now double click on **North America**. Your chart should look like this.



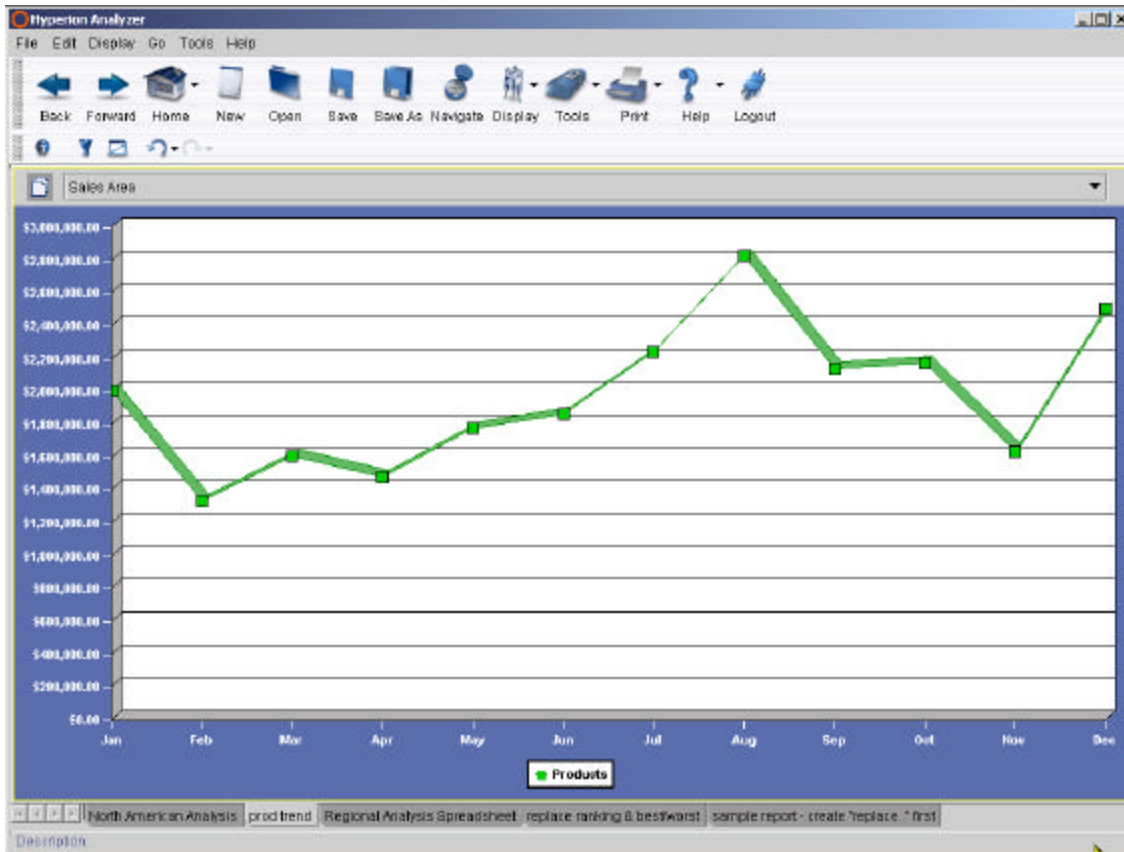
Notice that you can still page between **Actual** and **Forecast**. Save this report as “North American Analysis” in the report group “Teamxx Analysis” that you just created. You can use the File menu or right click on the tab for this report at the bottom of the screen. You can continue to drill down one more level under area. You can also double click on the individual Product Group bars to drill down to Product Type.

### 5.3. Creating a 12 month Product Trend Report

Try and create the next two reports without reading the detail instructions. Refer to the detail instructions when you have problems or encounter new functionality.

This report is a line graph that shows how your products have been doing over a 12 month period and allows you to page between various sales divisions.





Select the **New** icon to create a new view. Select **Chart** and click on **OK**. Select your **SALESDB** connection click **OK**. Drag **Year 2000** to the **Columns** box. Right click on **Year 2000** and choose **Select Dim Bottom**. This will choose the 12 months at the bottom of the Year 2000 dimension. Drag **Products** to the **Rows** box and simply check **Products** at the top level. Finally drag **Sales Area** to the **Pages** box. Expand **Sales Area**. Check **Sales Area, North America, Europe and Asia/Pacific**. Click **OK, OK**. If your default chart is not a line chart then click on the icon **Display|Charts|Line**. You should now see your revenue trend for all Products for one year. Right click on the chart and select **Data Display** and de-select “**Show Selected Member**”. Save this in your report group (Teamxx Analysis) and name it “Product Trend”. You can now drill down anywhere on the Product line to see more details about which Product Groups or Product Types might be having problems.

#### 5.4. Create the Forecast Variance Report

The Forecast Variance report shows your Actual and Forecast revenue numbers by Region with the ability to page through the different Product Groups. You need to add a calculated field to this report (Var %) and then use Traffic Lighting to show which geographies are within +/- 10% of their forecasts. Enjoy.

The screenshot shows the Hyperion Analyzer interface with a spreadsheet titled 'Products'. The spreadsheet displays sales data for various regions, comparing Actual revenue against Forecast revenue and calculating the Variance (Var %). The data is as follows:

	Actual	Forecast	Var %
Midwest	6,802,269	7,122,855	-3.10%
West	4,134,270	4,602,833	-10.02%
South	3,463,448	3,427,492	1.02%
East	3,422,684	3,622,794	-5.44%
Canada	2,602,364	2,298,201	13.21%
Belgium	1,822,881	1,076,252	5.02%
France	261,875	457,626	-42.67%
Germany	326,221	391,348	-16.17%
United Kingdom	304,844	298,313	2.16%
Australia	125,037	191,856	-34.72%
Japan	77,567	95,104	-18.45%
New Zealand	52,276	49,111	6.44%

Select the **New** icon to create a new view. Select the default **Spreadsheet** and click on **OK**. Select your **SALESDB** connection, click **OK**. Drag **Sales Area** to the **Rows** box. Expand **Sales Area**, expand **North America**, right click on **East** and choose **Also Select Level**. This will select all the other regions at the same level as East. Drag **Scenario** to the **Columns** box and expand to select **Actual** and **Forecast**. Drag **Products** to the **Pages** box and simply check **Products** at the highest level. Click **OK|OK** to redisplay your spreadsheet.

We want this report sorted by revenue, with the geographies with the highest revenue coming first. On the spreadsheet right click on **Actual|Analysis Tools|Sorting|Add**. Select **Actual, Descending** then click **OK** and **Close**. Your data should now be in sequence by highest to lowest revenue.

Next we need to calculate the variance between the Actual revenue and the Forecast revenue. On the spreadsheet right click on **Actual** and select **Analysis Tools|Calculations|Add**. Call your calculated field "Var %". From the drop down Function list choose "**Percent of Difference**". In the Select Members box select **Actual** first and **Forecast** second. Click **OK** then **Close**. On the spreadsheet right click on **Var %**, choose **Analysis Tools|Data Formatting|Add**. Select the member **Var %** and click on **Format Data**. Specify a minimum and maximum of 2 decimal positions and a % sign as a positive and negative suffix. Click **OK|OK|Close**.

To make the report more meaningful quickly we are going to use traffic lighting to spot problem areas at a glance. Right click on **Var %**, choose **Analysis Tools|Traffic Lighting|Add**. **Apply Traffic Lighting To Var %, Comparing It To Fixed Value**. If we hadn't created the variance field we could have added traffic lighting to the Actual revenue column based on the percentage difference between


Actual and Forecast. If you right click in the traffic lighting color bar (Assign Limits box) you will notice that you can add additional break points and change the colors. This allows you to use shading to indicate how good or how bad something is. Save this report in your report group and call it “Forecast Variance”. Since we used % differences instead of actual dollar value differences, when we drill down below the region to the areas (ie Canada to Alberta) you will see that the traffic lighting is still applied appropriately.


## 5.5. Creating a More Complex Report


	Actual	Forecast	Var %
Midwest	6,902,389	7,122,965	-3.10%
West	4,134,270	4,802,833	-13.32%
South	3,493,448	3,427,492	1.92%
East	3,423,684	3,822,784	-10.49%
Canada	2,602,354	2,238,201	16.21%
Belgium	1,623,881	1,676,262	-3.80%
France	261,875	487,826	-22.85%
Germany	300,221	301,348	-18.17%
United Kingd.	304,644	296,913	2.60%
Australia	125,037	191,695	-24.77%
Japan	77,557	95,104	-18.95%
New Zealand	52,275	49,111	6.44%


Note:  
United Kingdom states that their numbers were greatly affected by the coalminers strike.  
This has been resolved and business should return to normal

You are now entering the final stretch of this lab. You are going to add some controls to the Forecast Variance report that you just created. Right click on the Forecast Variance report tab at the bottom of your screen and choose Reload Report. This will restore the saved version of your report in case you have made any changes since you saved it.


Select the icon **Tools|Design Report**. You will see the top portion of your screen (menu and icons) change. Right click on the bottom half of the screen and choose **Alignment|None**. Now size and move the spreadsheet so it looks similar to the spreadsheet portion of the report above. Drag the Combo Box icon  onto your report. Ensure that **ReportDataSrc1** is selected and click **OK**. This links the combo box to the same data source that your spreadsheet used. In essence, this enables the combo box to control what you see on the spreadsheet. Select **Products|Add**. Now you choose the values you want to see in the drop down combo box. Expand **Product**, check **Product**, right click on **Clothing**, choose **Also Select Siblings**, click **OK**. Under members, check the dynamic box next to **Siblings of Clothing**.

This says, that as you develop new products they will automatically appear in the combo box. Click **OK**. Size and move the combo box if necessary. Drag the label icon  to the left of the combo box. Enter “Select Product” and click **font**. Choose bold, size 14, color dark blue. Click **OK|OK**.

Combo boxes, radio buttons and sliders allow you to choose one member to concentrate on. If you wish to select multiple members you need the Checkbox. Drag the Checkbox Subscription icon  onto your report. Ensure that **ReportDataSrc1** is selected and click OK. Select **Sales Area|Add**. Now you choose the values you want to see in the drop down combo box. Expand **Sales Area** and **North America**, check **Sales Area**, choose all divisions that are **North America**'s siblings, finally choose all regions that are at the same level as **East**. Click **OK**. Click next to **Sales Area**, under the dynamic heading but not actually in the dynamic checkbox. This should enable the additional options at the bottom of the window. Choose rename and change **Sales Area** to Worldwide Sales. Check the dynamic box for both the **North American siblings** and the **level east**. Click **OK** and again size and place the check box where you want it. Instead of creating a new label from scratch for our checkbox we will duplicate the label we create for the combo box and simply change the text. This way we keep the same font choices as before. Highlight the label **Select Product** and choose **Edit|Copy**. Click to the left of your Checkbox and choose **Paste**. Right click on the new label, choose **Properties** and enter the new label “Select one or more (*new line*) Geographies”.

Finally drag the text area icon  onto your report. Type the following note.  
Note:

United Kingdom states that their numbers were greatly affected by the coal miners strike but that is now over and they expect everything to return to normal.

Again size and move the text box as appropriate. To have your note stand out drag the panel icon  on top of your text. Select a color of white. Size the panel to completely cover the text box. Right click on the panel and select **Send to Back**. You should now see your text on a white background.

Click the Save As icon and call the new report “A Great Accomplishment”. To return to your standard view click the Normal View icon. Now try out your new report. Try using the combo box and the checkbox and see what happens.

To see all that you have accomplished click the Home icon, then select your report group TEAMxx Analysis. You should now see all the reports that you just created.

Congratulations, you have completed the IBM Business Intelligence Lab.

## Appendix:

### A.1 Create Definition 2

Definition 2 will be the definition required to create and load the target table TEAMxx/SALESORG from the source table SCSAMPLE40/SALESORG.

Click on the **New** icon or select **File!New!New** to create a new definition.

Select the source and target servers as before.

In the Tables window select the table SALESORG from the collection SCSAMPLE40.

In the **Columns** window select the following source columns in the following order:

AREA\_ID, AREA, REGION and DIVSN. Click on **Next**.

The next window is the **Conditions** window. Click on **Next** as there are no selection criteria to specify.

The next window is a **Sort** window. We want to sort the data by area , so click on AREA\_ID and click on the **Ascending** button. Finally, click on **Finish**.

You have now completed defining the source extraction rules for definition two.

The next stage is to define the target that you want to send the data to. In the **Table** window enter **TEAMxx** for the collection name and **SALESORG** for the table name and click on **Next**.

You will then be presented with a **Target Columns** window.

For AREA change the column name to AREA\_DESC, change the description to Area Description and change the column headings to Area Description. Click **OK**. Also, change the size of the target field to CHAR(30).

For REGION change the column name to REGION\_DESC, change the description to Region Description and change the column headings to Region Description. Click **OK**. Also, change the size of the target field to CHAR(20).

For DIVSN change the column name to DIV\_DESC, change the description to Division Description and change the column headings to Division Description. Click **OK**. Also, change the size of the target field to CHAR(20).

Having completed all of these column detail changes, click on **Next** to move to the **Indexes** window. Click on **Next** to move to the **Privileges** window and click on **Finish**. The task of creating Definition 2 is complete. We now need to save our definition.

From the main menu at the top of the screen select **File!Save As** and key in the definition name 'TEAMxx SalesOrg Load'. Then click Save and we now have a Definition called SalesOrg Load which we can run to populate a target table called TEAMxx/SALESORG from a source table called SCSAMPLE40/SALESORG.

## A.2 Create Definition 3

Definition 3 will be the definition required to create and load the target table TEAMxx/DATE\_TBL from the source table SCSAMPLE40/TIMEDIM.

Click on the **New** icon or select **File!New!New** to create a new definition.

Select the source and target servers as before.

In the Tables window select the table TIMEDIM from the collection SCSAMPLE40.

If you go back to review the Data Model that we are trying to create, you will notice that the key to this file is a column called Period which is in the format YYYYMM e.g. 200101 would represent period 1 in the year 2001. However, there is no such column in this format in the TIMEDIM source table. We will therefore need to create such a column from the data that is available.

In the **Columns** window click on **New** to add a new column.

In the New Column Assistants window select the **If-then-else Assistant** and click on **OK**.

Whilst in the **If-then-else Assistant** take some time to look at the rich set of built-in functions that are available to you to enable you to specify your data transformations. The top **Functions** box is a drop-down and you can see then different categories of functions that are available. The larger box beneath that lists all of the functions in the category selected. Select any one and press F1 to get help on what the function does and how to use it.

In our example we have a field in the source data called TYEAR which is an INTEGER(10) field and a field called TMONTH which is also an INTEGER(10) field. We need to concatenate TYEAR and TMONTH together to form our YYYYMM target field. However, we can only concatenate character fields so we need first to transform TYEAR for a CHAR(4) field and TMONTH to a CHAR(2) field. There is a further issue we need to resolve because most months are only 1 digit in length. If we have month 3 in year 2001 for example we need to ensure that in the final result we end up with is 200103 and not 20013. We therefore need to test to see whether TMONTH has one digit in it or two – and if it has only one then we need to insert a '0'.

In the **If-then-else Assistant** enter the following in the **If** box:

```
RIGHT( CAST( TMONTH AS CHAR( 2 ) ), 1 ) = ''
```

This means that we are converting the field TMONTH to a CHAR(2) field and are then testing the right-most character to see whether it is a blank or not. If it is a blank then we know we have a month in the range 1-9.

Enter the following in the **Then** box:

```
(( CAST ( TYEAR AS CHAR ( 4 ) ) ) CONCAT ( '0' ) CONCAT ( CAST ( TMONTH AS CHAR ( 1 ) ) ) ) )
```

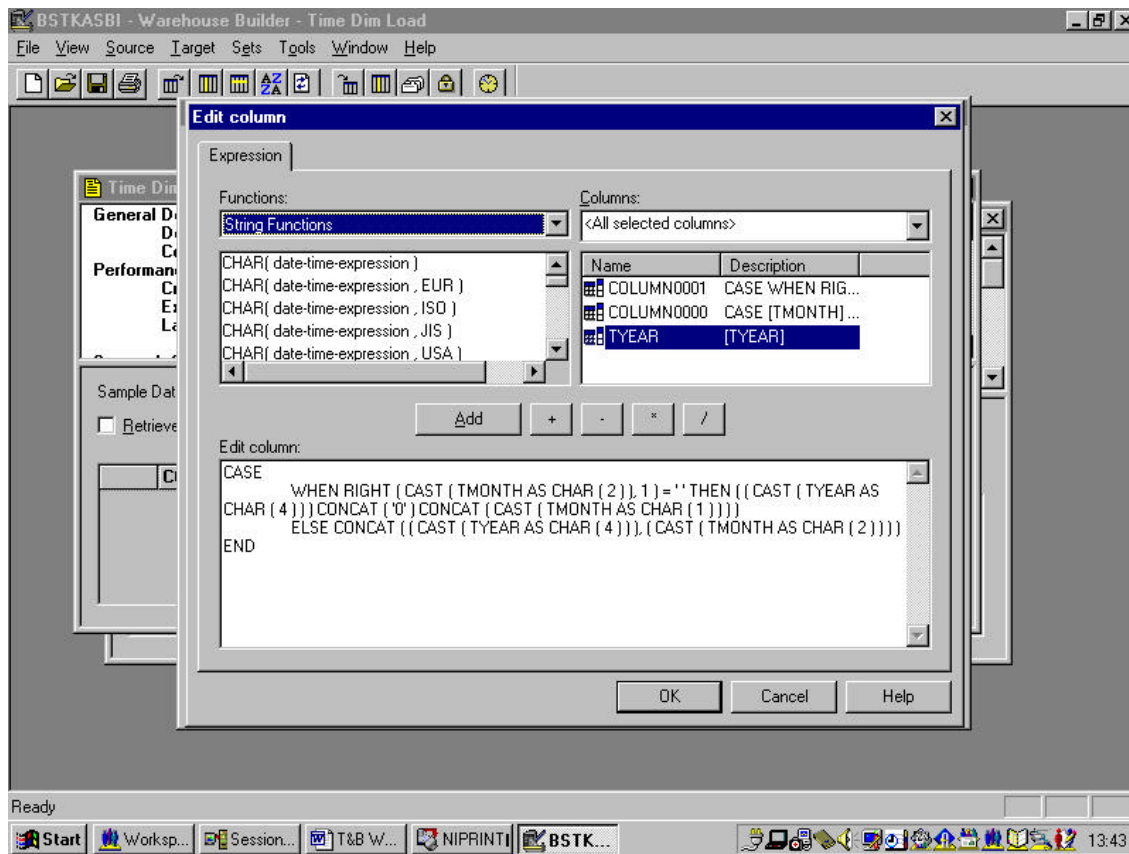
Here we are saying that now we know we have a one character month we need to concatenate TYEAR with a '0' and with the single character month.

Enter the following in the **Otherwise** box:

```
CONCAT ( ( CAST ( TYEAR AS CHAR ( 4 ) ) ) , ( CAST ( TMONTH AS CHAR ( 2 ) ) ) ) )
```

Here we are saying that if month is a two digit month then we just need to concatenate TYEAR and TMONTH together (ensuring that we convert them from numeric to character first).

Click on the **Expression** button to see the full expression. It should look something like the window below:



Click **OK** and return to the source **Columns** window.

The second column that we want in our target DATE\_TBL table is the Month description i.e. Jan, Feb, Mar etc. These text descriptions do not exist in the source data, so again we are going to have to create a new column.

Click on the **New** button, and this time select the **Value Substitution Assistant**. In the **Value Substitution Assistant** window click on TMONTH from the list of available columns and click on **Add** and then click on **Next**. On the following screen click on **Get Values** and the values 1 to 12 (i.e. the values in the field TMONTH) will be presented. For each value type in the substitution three character month name e.g. Jan for month 1 and Feb for month 2. Ensure you only type in three character month names as we are going to define our target field as CHAR(3) later on.

When you have finished click on **Finish**.

The third column we want in our DATE\_TBL table is the actual year which can be sourced directly from TYEAR. In the **Columns** window select TYEAR and click on **Add**. We now have our three columns.

Check the **Return Distinct Rows Only** checkbox and click on **Next**.

There are no conditions or sorts to specify for this source definition. Continue through to the target definition stage.

Define the target that you want to send the data to. In the **Table** window enter **TEAMxx** for the collection name and **DATE\_TBL** for the table name and click on **Next**.

You will then be presented with a **Target Columns** window.

Define the first field as CHAR(6) and then call it PERIOD.  
Define the second field as CHAR(3) and call it TMONTH.

Leave the third field as an integer field but name it TYEAR.

Having completed all of these column detail changes, click on **Next** to move to the **Indexes** window. Click on **Next** to move to the **Privileges** window and click on **Finish**. The task of creating Definition 3 is complete. We now need to save our definition.

From the main menu at the top of the screen select **File!Save As** and key in the definition name 'TEAMxx Date Load'. Then click **Save** and we now have a Definition called Date Load which we can run to populate a target table called TEAMxx/DATE from a source table called SCSAMPLE40/TIMEDIM.

Before we move on to create the last definition for our data model, lets review the target data that will be generated to ensure that our definitions are correct. Look at the icons on the top task bar. Click on the



nineth icon. This refreshes the sample data, and having clicked on it you will see some sample data being presented to you in the Date Load window.

### A.3 Create the Sales Areas Dimension

The Sales Areas dimension is going to be defined using the structure that we already have in our table TEAMxx/SALESORG. We need therefore to create a **Rules File**, in OLAP Server terms, that points the system to the TEAMxx/SALESORG file and tells the system what to do with that file in order to create the dimension.

Looking at the row of four buttons on the main Application Manager window, click on the **Data Load Rules** button. You will see the text at the top of the central white box change to 'Data Load Rules'. Click on **New** to create a new rules file.

You will be presented with a **Data Prep Editor** window. At the top of this window you will see 18 buttons. The 13<sup>th</sup> and 14<sup>th</sup> buttons look like this:



These two buttons determine whether you are creating a rules file for building a dimension, or a rules file for loading data. Click on the second of these two buttons to specify that we are creating a rules file for building a dimension. (When you position your mouse arrow over this second button you will see the words 'View Dimension Building Fields' appear at the bottom left of your screen).

Take your mouse arrow to the top left of your screen and click on **File!Open SQL** and you will be presented with the **Select the Server, Application and Database** window. The name or IP address of your iSeries server should appear in the Essbase Server box. The Essbase Application box should display your application name of SALESxx and the database SALESDB should appear in the Essbase Database box.

Click on **OK** and you will be presented with the Define SQL Window. This is used to tell the system where to find the data that we are going to use as the basis for defining our Sales Areas dimension.

Ensure that your iSeries system (name or IP address) is highlighted in the **SQL Data Sources** box.

At the bottom half of this window you will see where you can enter an SQL SELECT statement. There will already be a "\*" in the first box. Take your cursor to the next box to enter the FROM part of the SELECT statement. Type in TEAMxx.SALESORG (you need to use the SQL naming convention rather than the iSeries system naming convention). Click on **OK/Retrieve**. When you get the SQL Connect window enter your iSeries userid and password (this userid must have authority to your tables in your TEAMxx library). Click on OK and the data will be brought into your Application Manager **Data Prep Editor** window.

Your screen (with a maximised Data Prep Editor window) should now look like this:

	field 1	field 2	field 3	field 4	field 5
1	2001-04-04-17.32.12.20900000	00	District of Columbia	East	North America
2	2001-04-04-17.32.12.20900001	01	Alabama	South	North America
3	2001-04-04-17.32.12.20900002	02	Alaska	West	North America
4	2001-04-04-17.32.12.20900003	03	Arizona	West	North America
5	2001-04-04-17.32.12.20900004	04	Arkansas	South	North America
6	2001-04-04-17.32.12.20900005	05	California	West	North America
7	2001-04-04-17.32.12.20900006	06	Colorado	Midwest	North America
8	2001-04-04-17.32.12.20900007	07	Connecticut	East	North America
9	2001-04-04-17.32.12.20900008	08	Delaware	East	North America
10	2001-04-04-17.32.12.20900009	09	Florida	South	North America
11	2001-04-04-17.32.12.20900010	10	Georgia	South	North America
12	2001-04-04-17.32.12.20900011	11	Hawaii	West	North America
13	2001-04-04-17.32.12.20900012	12	Idaho	West	North America
14	2001-04-04-17.32.12.20900013	13	Illinois	Midwest	North America
15	2001-04-04-17.32.12.20900014	14	Indiana	East	North America

You can see that 'field 1' now contains the OLAP Builder generated timestamp, 'field2' contains the Sales Area ID, 'field 3' contains the Sales Area Description, 'field 4' contains the Sales Region and 'field 5' contains the Sales Division.

We must now link this data with the Outline that we started creating. Click on **Options\Associate Outline** (at the top of the screen), check the details and click on **OK**.

We must now provide the rules on how to use this data to build our dimension. For example – we need to explain how the Sales Area hierarchy works so that it is clear that a Region is made up of many areas and not the other way round.

Click on **Options\Dimension Build Settings**. You will be presented with the **Dimension Build Settings** window. Click on the **Dimension Definition** tab. Click on the **Rules File** radio button and in the **Name** box type in the name of the dimension, which is **Sales Areas**, and click on **Add** (do not click on OK).

Click on **Properties** to define the properties for this dimension.

Click on the **Dimension Properties** tab header so that the Dimension Properties tab is on view. In the configuration section click on the **Sparse** radio button. Click on **OK** to go back to the **Dimension Build Settings** window. Stay in this window and click on the **Dimension Build Settings** tab. In the **Dimension** box click on **Sales Areas** and in the Build Method section click on the **Use level References** radio button. Click on **OK**.

Having defined the general characteristics, we need now to go and tell the system what to do with each field. We have specified that we are going to use Level References for this dimension, so we now need to define the hierarchy in terms of levels. i.e. starting from the bottom, Area is level 0, Region is level 1 and Division is level 2.

You should now be back at the **Data Prep Editor** window.

Click on the column heading 'field 1' to select that entire column and click on **Field!Properties** on the menu at the top of the screen. This is just the timestamp field and we don't want to load this into the OLAP cube. Check that you are in the **Global Properties** tab, and check the checkbox that says **Ignore field during dimension build**. Click on **OK**.

You should now be back at the **Data Prep Editor** window.

Click on the column heading 'field 2' to select that entire column and click on **Field!Properties** on the menu at the top of the screen. Click on the **Dimension Building Properties** tab. We need to tell the system that the data in this field (i.e. Area Id) represents level 0 of our hierarchy. In the **Field Type** section click on **Level**. In the **Number** box type in the number zero. In the **Dimension** box click on **Sales Areas**. Click on **Next**.

You can now enter the details for field 3 which is the area description. Area description is still level 0 in the hierarchy, but we need to define it as an alias for the area ID. Click on **Alias** in the **Field Type** section. Check that the number zero is in the **Number** box and check that **Sales Areas** is in the **Dimension** box. Click on **Next**.

You can now enter the details for field 4 which is the Region description. Click on **Level** in the **Field Type** section. Check that the number one is in the **Number** box and check that **Sales Areas** is in the **Dimension** box. Click on **Next**.

You can now enter the details for field 5 which is the Division description. Click on **Level** in the **Field Type** section. Check that the number two is in the **Number** box and check that **Sales Areas** is in the **Dimension** box. Click on **OK**.

You have now completed the definition of the Sales Areas dimension. Before saving this rules file you can check that everything has been defined correctly by clicking on the rules verification button. This is the last button in the row of 18 buttons. It has a tick and the letter 'R' on it. Click on this button and you should get a message telling you that the rules file is correct for dimension building. Click **OK** in this message window.

Click on **File!Save As** to display the **Save Server Object** window. In the location box check that the **Server** radio button is checked. In the Object Name box enter the name **AREALOAD**. Click on **OK**.

Close the **Data Prep Editor** window.

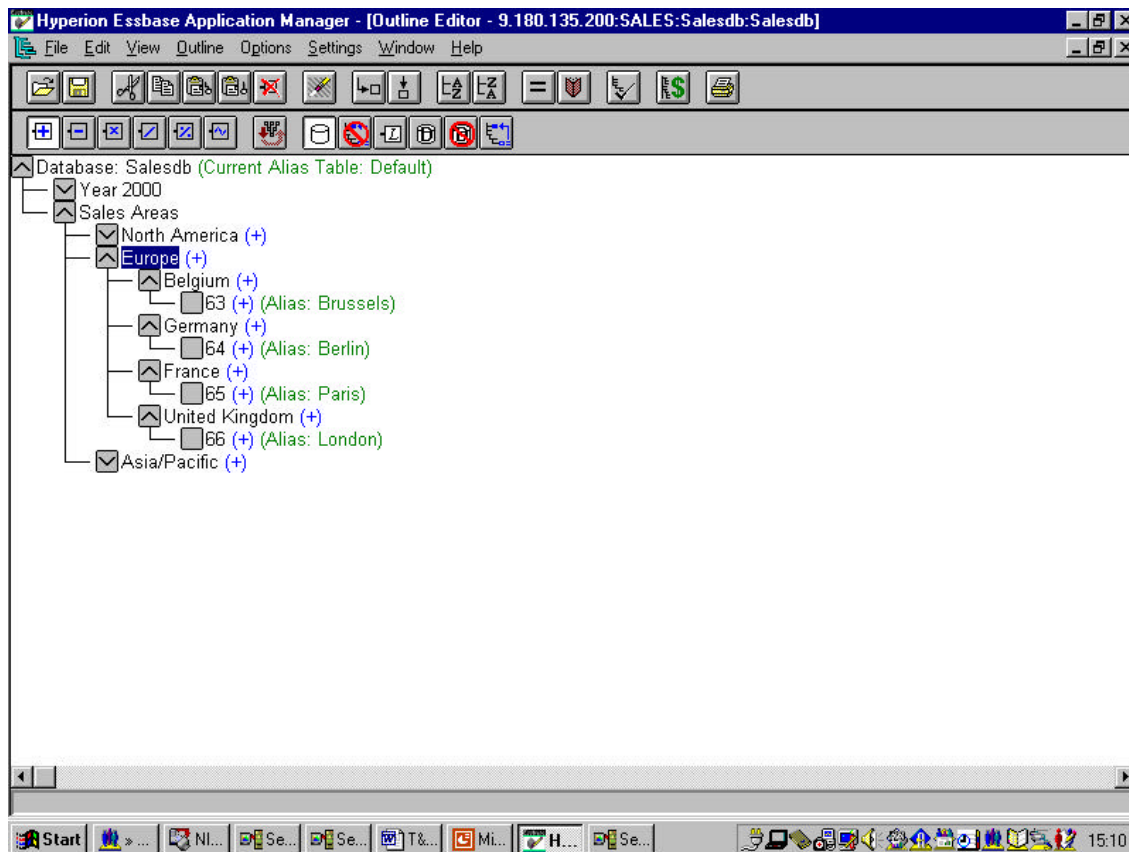
You have now successfully created a rules file for the creation of a dimension build. We can now update the Outline with the Sales Area dimension.

On the **Hyperion Essbase Application Manager** main window click on the Outline button (the first one of the four buttons on the screen). You will see the words at the top of the white box change to Database Outlines. Click on **Open**.

Maximise the **Outline Editor** window. Click on **File!Update Outline**. You will see the **Outline Update** window. Click on **SQL** in the Data section and click on the **Find** button in the Rules section to find the rules file that we are going to use to update the outline. Ensure that the **Server** radio button is selected. Ensure that the name of the rules file (AREALOAD) is in the **Object Name** box and click on **OK**, and click on **OK** again in the Update Outline window. Enter your iSeries userid and password in the **SQL Connect** window and click **OK**. The dimension Sales Areas should now appear on your outline.

Double click on the little box (with a downward pointing arrow) to the left of Sales Areas to view the divisions. Double click on the box to the left of one of the divisions to view the regions. Double click on a box to the left of one of the regions to view the areas. At the lowest level (area) you will see the Alias names that we defined in the rules file displayed. The use of double click will also take you back up the hierarchy.

Your Outline will now look something like this (depending on which levels of the hierarchy you have opened up):



Select **File!Save** from the main menu. Close the **Outline Editor** window after the database has been restructured.

## Appendix B:

### B.1 The CUSTOMERS Table

The CUSTOMERS table contains information about The Outdoor Connection's clients. It includes name and address information and it divides the clients into various geographical categories. The Outdoor Connection consists of three divisions (North America, Europe, and Asia/Pacific); 12 regions (East, West, South, Midwest, Canada, Belgium, Germany, France, United Kingdom, Japan, Australia, and New Zealand); 70 areas (one for each state and province, including the District of Columbia, plus one for each European and Asia/Pacific country); and three distribution facilities (Seattle, Kansas City, and New York). In addition, the Customers table includes various other information, including a currency code for each client, a status code that tracks whether a client is active or inactive, and a sales representative code that identifies each client's sales representative.

The CUSTOMERS table contains sufficient data to analyze customers according to two separate hierarchical structures:

#### **Sales organization structure**

- Division
- Region
- Area

#### **Geographic location**

- Country
- State/Province
- City
- Postal code

This table contains 157 rows of data.

Column	Description	Heading	Data Type	Contents
CUST_ID	Customer ID	Customer ID	Numeric(8)	Account ID of client
CUSTNAME	Customer Name	Customer Name	Char(100)	Name of client
LNAME	Surname	Last Name	Char(50)	Surname of client contact
FNNAME	Given Name	First Name	Char(50)	Given name of client contact
ADDRESS1	Address 1	Address 1	Char(100)	Address line 1
ADDRESS2	Address 2	Address 2	Char(100)	Address line 2
CITY	City	City	Char(50)	City name
STATE	State/Province Abbreviation	State/Province Abbr	Char(2)	State/province postal abbreviation
STATEDESC	State/Province Description	State/Province	Varchar(80)	State/province name
POSTALCODE	Postal Code	Postal Code	Char(16)	Postal code
COUNTY	County	County	Varchar(80)	County name
COUNTRY	Country Code	Country Code	Varchar(12)	Country code
COUNT RYDESC	Country Description	Country	Varchar(80)	Country name
PHONE	Phone Number	Phone Number	Char(24)	Phone number
E_ADDRESS	E-mail Address	E-mail Address	Char(100)	E-mail address of client contact
LOB	LOB	LOB Code	Char(8)	Line of business code
LOBDESC	Line of Business Description	Line of Business	Char(80)	Line of business description
REPCODE	Sales Rep Code	Rep Code	Char(4)	Account representative's code
ENTRY_DATE	Entry Date	Entry Date	Date	Date added
ACTIVE	Active Code	Active	Char(1)	Active status (Y or N)
DIVSN_ID	Division ID	Division ID	Char(8)	Division code
DIVSN	Division	Division	Char(80)	Division name
REGION_ID	Region ID	Region ID	Char(8)	Region code
REGION	Region	Region	Char(80)	Region name
AREA_ID	Area ID	Area ID	Char(8)	Area code
AREA	Area	Area	Char(80)	Area name
DIST_ID	Distribution Center ID	Distribution Cntr ID	Char(8)	Distribution center code
DIST_LOC	Distribution Center	Distribution Center	Char(80)	Distribution center name
CURR_CODE	Currency Code	Currency Code	Char(3)	Currency code

## B.2 The DATETYPES Table

The DATETYPES table is used to demonstrate how OLAP Builder handles date values stored as a data type other than DATE. Specifically, the DATEINV column has been replicated into several formats. You can use this table independently or join it to the ORDERS table on ORDNUM. It contains 14,379 rows of data.

Column	Description	Heading	Data Type
ORDNUM	Order Number	Order Number	Decimal(8)
DATEINV	Date Invoiced	Date Invoiced	Date
DATEINV_YY	Two Digit Year Value	YY	Decimal(2)
DATEINV_MM	Two Digit Month Value	MM	Decimal(2)
DATEINV_DD	Two Digit Day Value	DD	Decimal(2)
DATEINV_EXCELSRL	Five Digit Excel Serial Date	EXCELSRL	Decimal(5)
DATEINV_YYDDD	Five Digit YYDDD Date	YYDDD	Decimal(5)
DATEINV_YYMMDD	Six Digit YYMMDD Date	YYMMDD	Decimal(6)
DATEINV_YYDDMM	Six Digit YYDDMM Date	YYDDMM	Decimal(6)
DATEINV_DDMMYY	Six Digit DDMMYY Date	DDMMYY	Decimal(6)
DATEINV_MMDDYY	Six Digit MMDDYY Date	MMDDYY	Decimal(6)
DATEINV_CYYDDD	Six Digit CYYDDD Date	CYYDDD	Decimal(6)
DATEINV_CYYMMDD	Seven Digit CYYMMDD Date	CYYMMDD	Decimal(7)
DATEINV_CYYDDMM	Seven Digit CYYDDMM Date	CYYDDMM	Decimal(7)
DATEINV_YYYYDDD	Seven Digit YYYYDDD Date	YYYYDDD	Decimal(7)
DATEINV_YYYYMMDD	Eight Digit YYYYMMDD Date	YYYYMMDD	Decimal(8)
DATEINV_YYYYDDMM	Eight Digit YYYYDDMM Date	YYYYDDMM	Decimal(8)
DATEINV_MMDDYYYY	Eight Digit MMDDYYYY Date	MMDDYYYY	Decimal(8)
DATEINV_DDMMYYYY	Eight Digit DDMMYYYY Date	DDMMYYYY	Decimal(8)
DATEINV_CHAR	Eight Character YYYYMMDD Date	YYYYMMDD	Char(8)
DATEINV_HYF	Five Digit Infimum Date (EXCELSRL – 1)	HYF	Decimal(5)

## B.3 The EMPLOYEES Table

The EMPLOYEES table contains human resources information about The Outdoor Connection's employees. Specifically, it lists employees' home address and phone number, their e-mail address, their gender, their date of birth, and, where applicable, their social security number. It also contains employees' compensation type and rate, their hire date, and, where applicable, a termination date and reason.

This table contains 39 rows of data.

Column	Description	Heading	Data Type	Contents
EMP_ID	Employee ID	Employee ID	Numeric(8)	Employee number
LASTNAME	Surname	Last Name	Varchar(100)	Surname of employee
FIRSTNAME	Given Name	First Name	Varchar(100)	Given name of employee
ADDRESS1	Address 1	Address 1	Char(100)	Address line 1
ADDRESS2	Address 2	Address 2	Char(100)	Address line 2
CITY	City	City	Char(50)	City name
STATE	State/Province Abbreviation	State/Province Abbr	Char(2)	State/province postal abbreviation
POSTALCODE	Postal Code	Postal Code	Char(16)	Postal code
COUNTRY	Country	Country	Char(50)	Country name
PHONE	Phone Number	Phone Number	Char(24)	Phone number
E_ADDRESS	E-mail Address	E-mail Address	Char(100)	E-mail address of client contact
SEX	Gender	Sex	Char(1)	Gender
DOB	Date of Birth	Date of Birth	Date	Date of birth
SSN	Social Security Number	Social Security No.	Char(11)	Social security number
COMPTYPE	Compensation Type	Comp Type	Char(1)	Compensation type
SALARY	Base Monthly Salary	Base Salary	Numeric(10,2)	Base monthly salary
HOURLY	Hourly Rate	Hourly Rate	Numeric(4,2)	Hourly wage
COMMISSION	Commission Rate	Commission	Numeric(4,4)	Commission percentage
CURR_CODE	Currency Code	Currency Code	Char(3)	Currency code
HIREDATE	Hire Date	Hire Date	Date	Hire date
TERMDATE	Termination Date	Termination Date	Date	Termination date
TERMTYPE	Termination Reason	Termination Reason	Char(20)	Reason for termination

## B.4 The ORDERS Table

The ORDERS table is the basis of all transaction information. It contains the order number, the location placing the order, the date of the order, the promised delivery date, the date shipped, etc. The table also includes a column that records the location to which each order is shipped. This accommodates the situations in which the order is shipped to a location other than that where the order originates.

The ORDERS table also contains the product-related specifics of each order. Such information includes the quantity ordered, quantity shipped, quantity backlogged, and quantity canceled for each product. In addition, the table is used to track the unit price, unit cost, extended price, and extended cost for each product ordered. Finally, for international sites, the table includes exchange rates. The rates are used to calculate the foreign extended price and foreign extended cost for each product based on the exchange rate at the time of the order.

The exchange rate is an unsigned percentage and can be mapped to a currency code for each country in which The Outdoor Connection conducts business. It is assumed that all monetary amounts entered into the database are entered in U.S. dollars. Therefore, by definition, the exchange rate for U.S. dollars is 1.0. The exchange rate for other currencies is based on the amount that one U.S. dollar will buy. For example, \$1 US recently bought \$1.4575 CA. Therefore, the exchange rate for Canadian dollars is 1.4575. To convert an amount in the database into local values, multiply the amount by the exchange rate: (\$10 US X 1.4575 = \$14.58 CA or \$10 US X 0.6105 = £6.11).

This table contains 14,379 rows of data.

Column	Description	Heading	Data Type	Contents
ORDNUM	Order Number	Order Number	Decimal(8)	Order number
CUST_ID	Customer ID	Customer ID	Numeric(8)	Account ID of client
SHIP_TO	Shipping Destination	Ship To	Numeric(8)	Account ID of shipping destination
PRODUCTID	Product ID (SKU)	Product ID	Numeric(8)	Product ID
PRDLVL03	Product Level 3	Product Level 3	Char(8)	Third-tier product category
QTYORD	Quantity Ordered	Quantity Ordered	Decimal(15)	Quantity ordered
QTYSHIP	Quantity Shipped	Quantity Shipped	Decimal(15)	Quantity shipped
QTYBACK	Quantity Backordered	Quantity Backordered	Decimal(15)	Quantity back-ordered
QTYCAN	Quantity Cancelled	Quantity Cancelled	Decimal(15)	Quantity cancelled
DATEREQ	Date Requested	Date Requested	Date	Date requested for delivery
DATEORD	Date Ordered	Date Ordered	Date	Date ordered
DATEPROM	Date Promised	Date Promised	Date	Date promised for delivery
DATESHIP	Date Shipped	Date Shipped	Date	Date shipped
DATEINV	Date Invoiced	Date Invoiced	Date	Date invoiced
UNIT_PRICE	Unit Price	Unit Price	Decimal(15,4)	Unit retail price
UNIT_COST	Unit Cost	Unit Cost	Decimal(15,4)	Unit wholesale cost (cost of goods sold)
EXTPRICE	Extended Price	Extended Price	Decimal(15,4)	Extended retail total (QTYORD * UNIT_PRICE)
EXTCOST	Extended Cost	Extended Cost	Decimal(15,4)	Extended wholesale total (QTYORD * UNIT_COST)
MEASURE	Unit of Measure	Unit of Measure	Char(4)	Unit of measure code
CURR_CODE	Currency Code	Currency Code	Char(3)	Currency code
EXCHG_RATE	Exchange Rate	Exchange Rate	Decimal(15,7)	Exchange rate
FRGN_XPRIC	Foreign Extended Price	Foreign Ext Price	Decimal(15,4)	Foreign extended retail total
FRGN_XCOST	Foreign Extended Cost	Foreign Ext Cost	Decimal(15,4)	Foreign extended wholesale total
DIVSN_ID	Division ID	Division ID	Char(8)	Division code
REGION_ID	Region ID	Region ID	Char(8)	Region code
AREA_ID	Area ID	Area ID	Char(8)	Area code
DIST_ID	Distribution Center ID	Distribution Cntr ID	Char(8)	Distribution center code
REPCODE	Sales Rep Code	Rep Code	Char(4)	Account representative's code



## B.5 The PRODUCTS Table

The PRODUCTS table contains all of The Outdoor Connection's product information and groups products hierarchically. Each product has three levels associated with it. The first identifies a product group (e.g.; clothing, biking, skiing, etc.), the second a type (e.g.; outerwear, shirts, pants, etc.), the third a style (e.g.; black, red, yellow, etc.). In addition, the PRODUCTS table includes a unit price and a unit cost for each product. It also contains a currency code so that the unit price and cost can be converted for various countries. Finally, it includes a measures column that a unit type (e.g.; one bicycle, a pair of skis, etc.) and a date that records when information was last changed.

This table contains 120 rows of data.

Column	Description	Heading	Data Type	Contents
PRODUCTID	Product ID (SKU)	Product ID	Numeric(8)	Product ID
PRODUCT	Product Description	Product Description	Char(80)	Product description
PRDLVL01	Product Level 1	Product Level 1	Char(8)	First tier product category
PRDDESC01	Product Description 1	Description Lvl 1	Char(80)	First tier product description
PRDLVL02	Product Level 2	Product Level 2	Char(8)	Second tier product category
PRDDESC02	Product Description 2	Description Lvl 2	Char(80)	Second tier product description
PRDLVL03	Product Level 3	Product Level 3	Char(8)	Third tier product category
PRDDESC03	Product Description 3	Description Lvl 3	Char(80)	Third tier product description
PRDLVL04	Product Level 4	Product Level 4	Varchar(8)	Fourth tier product category
PRDDESC04	Product Description 4	Description Lvl 4	Varchar(27)	Fourth tier product description
PRDLVL05	Product Level 5	Product Level 5	Varchar(8)	Fifth tier product category
PRDDESC05	Product Description 5	Description Lvl 5	Varchar(27)	Fifth tier product description
UNITPRICE	Unit Price	Unit Price	Decimal(15,4)	Unit retail price
UNITCOST	Unit Cost	Unit Cost	Decimal(15,4)	Unit wholesale cost (cost of goods sold)
CURR_CODE	Currency Code	Currency Code	Char(3)	Currency code
MEASURE_UOM	Code	UOM Code	Char(4)	Unit of measure code
MEASURE_DESC	Unit of Measure	Unit of Measure	Char(80)	Unit of measure description
ENTRY_DATE	Entry Date	Entry Date	Date	Entry/update date

## B.6 The SALESORG Table

The SALESORG table provides a key to the geographic structure used in the database. Essentially, it maps the various divisions, regions, and areas to each other. It also indicates which distribution facility serves each area or foreign region. As explained previously, the database contains three divisions, 12 regions, 70 areas, and three distribution facilities.

This table contains 70 rows of data.

Column	Description	Heading	Data Type	Contents
DIVSN_ID	Division ID	Division ID	Char(8)	Division code
DIVSN	Division	Division	Char(80)	Division name
REGION_ID	Region ID	Region ID	Char(8)	Region code
REGION	Region	Region	Char(80)	Region name
AREA_ID	Area ID	Area ID	Char(8)	Area code
AREA	Area	Area	Char(80)	Area name
DIST_ID	Distribution Center ID	Distribution Cntr ID	Char(8)	Distribution center code
DIST_LOC	Distribution Center	Distribution Center	Char(80)	Distribution center name

## B.7 The SALESPERF Table

The SALESPERF table is used to track a number of performance measurements. It includes sales projections as well as actual sales information. It also includes sales quota information for each sales representative. The table also allows you to determine actual and projected sales, as well as quotas, on a geographical basis.

This table contains 12,008 rows of data.

Column	Description	Heading	Data Type	Contents
PERIOD	Period	Period	Date	Time period
REPCODE	Sales Rep Code	Rep Code	Char(4)	Account representative's code
AREA_ID	Area ID	Area ID	Char(8)	Area code
PRODUCTID	Product ID (SKU)	Product ID	Numeric(8)	Product ID
PRJ_UNITS	Projected Unit Sales	Projected Units	Numeric(15,0)	Projected unit sales
ACT_UNITS	Actual Unit Sales	Actual Units	Numeric(15,0)	Actual units sold
PRJ_SALES	Projected Sales	Projected Sales	Numeric(15,4)	Projected sales amount
ACT_SALES	Actual Sales	Actual Sales	Numeric(15,4)	Actual sales amount
QUOTA	Quota	Quota	Numeric(15,4)	Sales quota

## B.8 The SALESREPS Table

The SALESREPS table maps a salesperson's code to a name, employee ID, and manager code. It also maps the salespeople to regions so that salespeople can be tracked and evaluated by geography. Each domestic region contains three sales people, Canada and each European region have two, and each Asia/Pacific region has one.

This table contains 25 rows of data.

Column	Description	Heading	Data Type	Contents
EMP_ID	Employee ID	Employee ID	Numeric(8)	Employee number
REPCODE	Sales Rep Code	Rep Code	Char(4)	Account representative's code
MGRCODE	Manager Code	Manager Code	Char(4)	Supervisor's code
LASTNAME	Surname	Last Name	Varchar(100)	Surname of employee
FIRSTNAME	Given Name	First Name	Varchar(100)	Given name of employee
REGION_ID	Region ID	Region ID	Char(8)	Region code
REGION	Region	Region	Varchar(80)	Region name
ENTRY_DATE	Entry Date	Entry Date	Date	Entry date (date of hire)
ACTIVE	Active	Active	Char(1)	Active status (Y or N)

## B.9 The TIMEDIM Table

The TIMEDIM table contains time-related information for specific dates. This table enables you to perform a variety of time-related analyses.

This table contains 2,192 rows of data.

Column	Description	Heading	Data Type	Contents
TDATE	TDATE	TDATE	Date	Date
TYEAR	TYEAR	TYEAR	Integer	Year
TMONTH	TMONTH	TMONTH	Integer	Month number
TDAY	TDAY	TDAY	Integer	Day of month
DAYOFWEEK	DAYOFWEEK	DAYOFWEEK	Integer	Day of week number
DAYNAME	DAYNAME	DAYNAME	Varchar(9)	Day
DAYOFYEAR	DAYOFYEAR	DAYOFYEAR	Integer	Day of year number
TWEEK	TWEEK	TWEEK	Integer	Week number
HOLCODE	HOLCODE	HOLCODE	Integer	Holiday code
HOLIDAY	HOLIDAY	HOLIDAY	Varchar(30)	Holiday name