Education

Infor Education:
Ready, Set, Succeed.

Infor BI: Configuring Databases with ImportMaster Training Workbook

Infor BI
Version 10.5
February 1, 2016
Course Code: 01_0061050_IEN0094_BSA

Legal notice

Important Notices

The material contained in this publication (including any supplementary information) constitutes and contains confidential and proprietary information of Infor.

By gaining access to the attached, you acknowledge and agree that the material (including any modification, translation or adaptation of the material) and all copyright, trade secrets and all other right, title and interest therein, are the sole property of Infor and that you shall not gain right, title or interest in the material (including any modification, translation or adaptation of the material) by virtue of your review thereof other than the non-exclusive right to use the material solely in connection with and the furtherance of your license and use of software made available to your company from Infor pursuant to a separate agreement, the terms of which separate agreement shall govern your use of this material and all supplemental related materials ("Purpose").

In addition, by accessing the enclosed material, you acknowledge and agree that you are required to maintain such material in strict confidence and that your use of such material is limited to the Purpose described above. Although Infor has taken due care to ensure that the material included in this publication is accurate and complete, Infor cannot warrant that the information contained in this publication is complete, does not contain typographical or other errors, or will meet your specific requirements. As such, Infor does not assume and hereby disclaims all liability, consequential or otherwise, for any loss or damage to any person or entity which is caused by or relates to errors or omissions in this publication (including any supplementary information), whether such errors or omissions result from negligence, accident or any other cause.

Without limitation, U.S. export control laws and other applicable export and import laws govern your use of this material and you will neither export or re-export, directly or indirectly, this material nor any related materials or supplemental information in violation of such laws, or use such materials for any purpose prohibited by such laws.

Trademark Acknowledgements

Table of contents

About This Workbook

Welcome to this Infor Education course! We hope you will find this learning experience enjoyable and instructive. This Training Workbook is designed to support the following forms of learning:

- Classroom instructor-led training
- Virtual instructor-led training

This Training Workbook is not intended for self-study or as a product user guide.

Activity Data

You will be asked to complete some practice exercises during this course. Step-by-step instructions are provided in this guide to assist you with completing the exercises. Where necessary, data columns are included for your reference.

Your instructor will provide more information on systems used in class, including server addresses, login IDs, and passwords.

Reference Materials

Infor BI ImportMaster reference materials are available from the following locations:

- Infor BI ImportMaster Online Help
- Infor Xtreme

Symbols used in this workbook


Hands-on exercise ("Exercise")


For your reference


Instructor demonstration ("Demo")


Your notes


Scenario


Question


Note


Answer

Course Overview

This course introduces the basic features of Infor BI ImportMaster and provides learners with the foundational knowledge needed to import data from a variety of operational systems to a multidimensional OLAP database.

Course Length

2 days

Course Goal

This course provides an introduction to the basic ImportMaster processes required to import data from a relational database to a multidimensional database, including defining the source and destination databases and defining the structure of the destination database by creating dimensions, elements, and cubes manually, via mapping, and through the use of scripts.

Learning Objectives

Upon completion of this course, you will be able to:

- Describe the purpose of using ImportMaster.
- Create and save an import definition.
- Identify the main components of the ImportMaster user interface.
- Protect an import definition.
- Create a relational database connection and use the SQL Query Builder to check the database.
- Identify the global folders that apply to all multidimensional databases.
- Create a multidimensional database connection.
- Create dimensions using the New Dimension wizard.
- Create and execute jobs.
- Manually create static elements in a new dimension.
- Explain the process and characteristics of mapping.
- Identify the various source and destination objects used in mapping.
- Use the Debug feature in a mapping.
- Identify the main programming features of CWScript.
- Create a dimension using CWScript and debug a script.
- Create a cube.
- Import data from a relational table into an OLAP database.

Audience

- Customer User
- Pre-Sales Consultant
- Business Consultant
- Technical Consultant
- Support
- System Administrator

System Requirements

- Infor BI Training Environment

Prerequisite Knowledge

- Advanced knowledge of the Infor BI OLAP server
- Programming skills and knowledge of relational databases, such as SQL and MS Access, is preferred.

Course Agenda

The purpose of this course is to introduce the basic features of Infor BI ImportMaster and provide learners with the foundational knowledge needed to import data from a variety of operational systems to a multidimensional OLAP database. This training is applicable for the following Infor BI versions: 10.4 and all later versions. This course is a two-day instructor-led course. (Course code: 01_0061040_IEN0003).

Lesson	Lesson title	Learning objectives	Estimated time
Course Overview		• Review course expectations.	30 minutes
Lesson 1	ImportMaster Overview	• List the main components of Infor BI. • Describe the purpose of using ImportMaster.	30 minutes
Lesson 2	Getting Started with Infor BI ImportMaster	• Create and save an import definition. • Identify the main components of the ImportMaster user interface. • Protect an import definition.	1 hour
Lesson 3	Defining a Relational Database	• Create a relational database connection. • Use the SQL Query Builder to check a relational database.	30 minutes
Lesson 4	Defining a Multidimensional Database	• Identify the global folders that apply to all multidimensional databases. • Create a multidimensional database connection.	30 minutes
Lesson 5	Creating Static Dimensions and Elements	• Describe static dimensions. • Create dimensions using the New Dimension wizard. • Create and execute jobs. • Manually create static elements in a new dimension.	1 hour
Lesson 6	Using Mapping to Create Dimensions	• Explain the process and characteristics of mapping. • Identify the source and destination objects used in mapping. • Use the Debug feature in a mapping.	2 hours
Lesson 7	Using Scripts to Create Dimensions	• Identify the main programming features of CWScript • Create a dimension using CWScript. • Debug a script.	2 hours
Lesson 8	Creating Other Dimensions	• Create a Currency dimension. • Create a Measure dimension.	30 minutes
Lesson 9	Creating a Cube and Importing Data	• Create a cube. • Import data from a relational table into an	1 hour

Lesson	Lesson title	Learning objectives	Estimated time
		OLAP database.	
Course Summary		• Debrief course.	30 minutes
TOTAL ESTIMATED TIME			10 hours

Lesson 1: ImportMaster Overview

Estimated Time

30 minutes

Learning Objectives

After completing this lesson, you will be able to:

- List the main components of Infor BI.
- Describe the purpose of using ImportMaster.

Topics

- Introduction to Infor BI
- Introduction to ImportMaster

Introduction to Infor BI

Infor BI is a fully integrated solution suite that supports various types of financial, operational, and sales business intelligence requirements. It can be used for standard reporting, flexible ad-hoc reporting and analysis, dashboard creation, business planning, budgeting, forecasting, and financial consolidation.

The following table identifies the components of the Infor BI suite and provides a brief description of each:

Component	Description
Application Studio	This component is a web-based front-end used for visualizing data through reports, analysis, dashboards, and data entry.
Office Plus	This component is a fully integrated Microsoft® Excel® add-in mainly used for ad-hoc analysis and reporting.
OLAP	This component is A real-time, in-memory, online, analytical processing (OLAP) database server for multidimensional analysis, planning, and modeling.
Designer	This component is a tool for designing and creating Infor BI OLAP Server databases.
Planning	This component is an application for financial, operational, and overall business budgeting and forecasting.
Consolidation	This component is A statutory and management consolidation application that leverages the Infor BI OLAP Server. **Note:** In Infor BI 10.4.1, Planning and Consolidation have been combined.
ImportMaster	This component is an extract, transform, and load (ETL) layer that facilitates integration with both Infor and non-Infor source systems.
DeltaMiner	This component is a tool providing statistical analysis, data mining, and other advanced analytic capabilities.
Business Analytics	This component is an application that includes predefined, role-based content for sales, finance, production, and other functional areas.

Introduction to ImportMaster

In most systems, data is managed in a relational database. Relational databases have proven to have weaknesses with analysis processes, such as complex queries via SQL statements and hard-coded analyses. This is one of the reasons OLAP technology was developed.

Infor BI ImportMaster acts as the interface between existing relational database systems and individual OLAP databases. It is an automated solution for importing data from a variety of Infor and non-Infor source systems to multidimensional OLAP databases. Infor BI ImportMaster can read in the structure of an existing database to use as a basis for populating an OLAP database. It can also be used as a tool to further define the structure of the OLAP database.

Infor BI ImportMaster imports data cleanly, which is an essential prerequisite to having a successful ad-hoc analytical environment. Data can be integrated from any source system that can be accessed via ODBC or that can export CSV or flat files. This data can be imported into the system on an ad-hoc or scheduled basis.

ImportMaster establishes a connection between one or more data sources and one or more destination databases. Installing Infor BI ImportMaster on the same system as the destination database may improve performance. When the data sources are available on the same system, Infor BI ImportMaster can bypass the network and access the hard disk directly. However, since most data sources are centrally located, this is not always possible, and therefore, you should ensure that the Infor BI ImportMaster system has access to a high-speed network connection.

This training uses a specific business scenario to help you understand which factors affect the creation process and the structure of an OLAP database and to learn how to differentiate between the various types of data in a source system.

Scenario

CW Cars is an internationally active company in the car retail trade. It has European locations in Germany, Switzerland, and Poland. CW Cars in Germany has a central IT system which processes all of its business transactions. Dealerships are connected to headquarters and one another via a global network. You have been asked to develop a controlling system based on an existing Microsoft® SQL database.

In order to complete this task, you must create a suitable database structure based on functional requirements, including determining the dimensions, elements, cubes, attributes, and subsets that are required, and then populate your database structure by importing the relevant data from the CWCars Microsoft SQL database. The imported data can then be used to produce reports from the OLAP database.

The new controlling system will have to meet the following functional requirements:

- Analysis of the business transactions, subdivided by:
  - Month/Quarter/Year from 2010 through 2013
  - Customers
  - Vehicle type
  - Manufacturer/Model
  - Sales structure

Optional Demo: Create a new database for ImportMaster training

Your instructor will demonstrate how to create a new database for the training. **Note**: This Demo/Exercise is only needed if the trainer is conducting ImportMaster training before any other Infor BI training. ImportMaster is typically completed last in the series of Infor BI training classes.

_____

_____

_____

_____

Optional Exercise 1.1: Create a new database for the ImportMaster training

In this exercise, you will create a new database for the training. **Note**: This Demo/Exercise is only needed if the trainer is conducting ImportMaster training before any other Infor BI training. ImportMaster is typically completed last in the series of Infor BI training classes.

Optional Exercise Steps

Note: This Demo/Exercise is only needed if the trainer is conducting ImportMaster training before any other Infor BI training. ImportMaster is typically completed last in the series of Infor BI training classes.

Part 1: Create a new database to be used for the ImportMaster training

1. Double-click the **OLAP Administration** icon on the desktop. The **Infor BI OLAP Administration** application opens.
2. Double-click the **Computer Configuration** folder.
3. Right-click the **Local** computer. A menu list displays.
4. Click **New Database…**from the menu. A **New Database Wizard** displays.
5. Click **Next >**.
6. Type *ACADEMY_IM* in the **Database** name field.
7. Click **Next >**.
8. Select **Tutor Repository** from the **Repository registration** drop-down list.
9. Type *Admin* in the **User name** field.
10. Leave the **Password** field blank.
11. Click **Next >**.
12. Select **Academy** from the **Project name** drop-down list.
13. Select **Academy** from the **OLAP Permission Management** drop-down list.

14. Click **Next >** to review the summary of settings.
15. Click **Finish**. The **Infor BI OLAP Administration** application displays.

Part 2: Confirm new database is working for the ImportMaster training

1. Double-click the **Favorite Databases** folder.
2. Click the **Register Database** icon.
3. Select **LOCAL/ACADEMY_IM** from the **Database** drop-down list.
4. Leave the **Alias** field blank.
5. Type *Admin* in the **User name** field.
6. Leave the **Password** field blank.
7. Click **OK**. **Note**: If the database is set up correctly and working then it will appear in the list with a green check mark.
8. Click the **x** to close the application.
9. Your database is now ready for training.

Lesson 2: Getting Started with Infor BI ImportMaster

Estimated Time

1 hour

Learning Objectives

After completing this lesson, you will be able to:

- Create and save an import definition.
- Identify the main components of the ImportMaster user interface.
- Protect an import definition

Protect an import definition.Topics

- The Import Definition
- The User Interface
- Protecting an Import Definition

Product Activation

Before starting to work on ImportMaster, a technical developer license must be added to the solution. The license is issued and sent via email. When installing, the company name must match the name of the license issued. Afterwards a process that involves an email registration needs to take place.

Infor BI ImportMaster Help

What is the product activation process?

Demo: Activate ImportMaster

Your instructor will demonstrate how to activate Infor BI ImportMaster and create and save an import definition.

_____

_____

_____

_____

Exercise 2.1: Activate ImportMaster

In this exercise, you will start Infor BI ImportMaster and create and save an import definition.

Exercise Steps

1. Double-click the **Product Activation** icon on the desktop. The **Product Activation** application opens.
2. Click **Product**, if the product Infor BI Import Master is shown as not activated. Otherwise this exercise is not needed.
3. Click the **Activate Product…** button.
4. Follow the onscreen prompts and select **I already have activation code**.
5. Type the *<4-digit code provided by the instructor>* in the **Demo: Activate ImportMaster**.
6. Click **OK**.
7. Click **Close**.

The Import Definition

All work in Infor BI ImportMaster is based on an import definition. All definitions for source and destination databases, and imports, such as information about the transferring and processing of data, are created and managed in the import definition.

Infor BI ImportMaster Help

What is an import definition?

Demo: Create and Save an Import Definition

Your instructor will demonstrate how to start Infor BI ImportMaster, and create and save an import definition.

_____

_____

_____

_____

Exercise 2.2: Create and Save an Import Definition

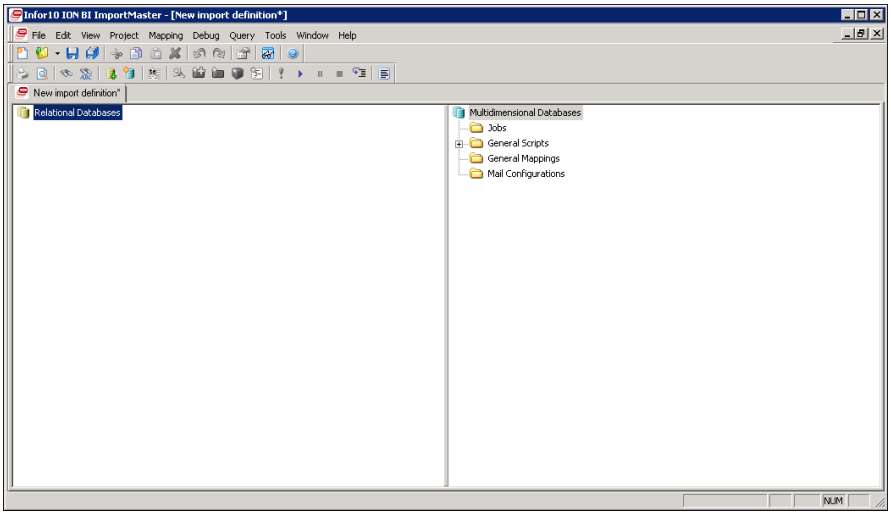
In this exercise, you will start Infor BI ImportMaster, and create and save an import definition.

Exercise Steps

1. Double-click the **ImportMaster** icon on the desktop. The **ImportMaster** application opens.
2. Select **File > New**. A **New import definition** tab displays.
3. Right-click the **New import definition** tab and select **Save** from the drop-down menu. The **Save As** dialog box opens.
4. Click **Desktop** in the left pane.
5. Type *ImportMasterTraining<YourInitials>* in the **File Name** field.
6. Click **Save**. The file is saved on the desktop with an .imd file extension.

The User Interface

Infor BI ImportMaster interfaces with surrounding IT systems using a standard interface with an intuitive environment that utilizes drag-and-drop functionality to define dimension structures and data-upload logic.



ImportMaster User Interface

The ImportMaster user interface consists of the following main components:

- Menu bar
- Toolbars
- Tab bar
- Relational Databases pane
- Multidimensional Databases pane

Infor BI ImportMaster also includes a certified interface to extract data from SAP NetWeaver® Business Warehouse and SAP® ERP applications.

Menu Bar

The Infor BI ImportMaster menu bar includes the following menus:

- File
- Edit
- View
- Project
- Mapping
- Debug
- Query
- Tools
- Window

Toolbars

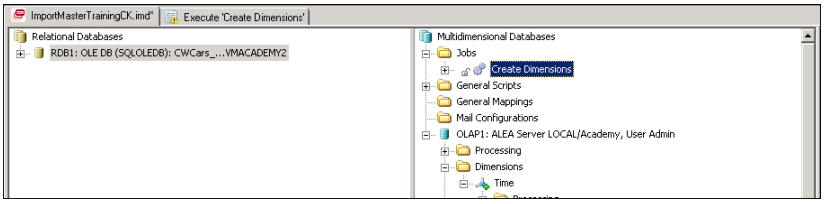
The toolbar buttons in ImportMaster provide quick access to the most important features and functionality. The following toolbars are available depending on the window that is active:

- General
- Project
- Mapping
- Script
- SQL
- BAPI
- Dimension
- Rule
- Documentation

Note: You can activate or deactivate the current view's toolbars by selecting View > Toolbars from the menu bar.

Tab Bar

The tab bar displays each open window as a tab identified with its icon and name. The tab bar is always visible unless all the work windows have been closed.

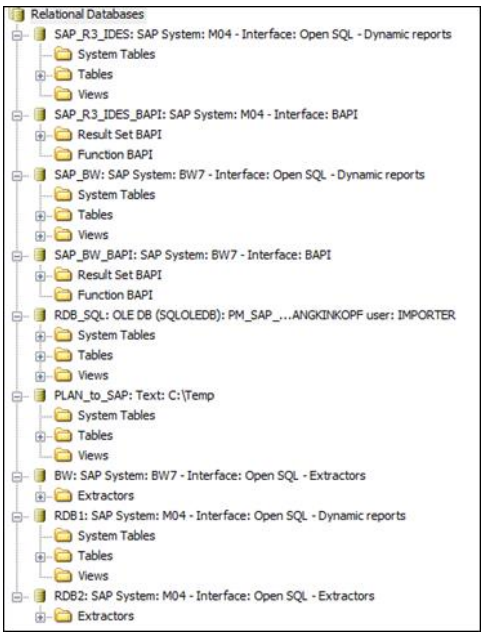


Tab Bar

- Clicking a tab activates and displays the associated window.
- Right-clicking a tab allows you to close work windows and use the Save and Save Import commands.
- Double-clicking a tab minimizes and maximizes the associated window.

Relational Databases Pane

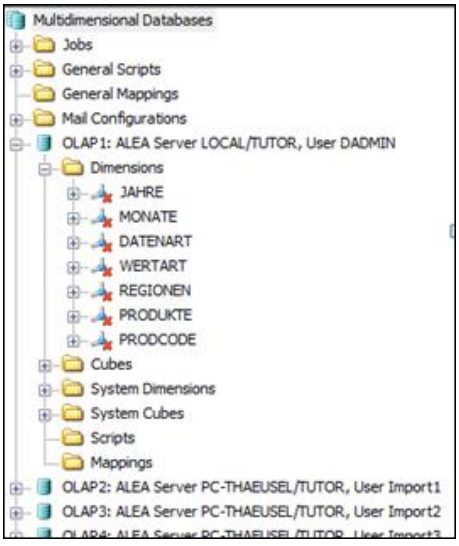
An open import definition appears in a window that is divided into two panes. All connections to relational databases (source databases) that have been defined for the import definition are displayed in the left pane. For example, OLE DB-capable databases, ODBC-capable databases, SAP R/3 (including BAPIs), and text files will display in this pane.



Relational Databases Pane

Multidimensional Databases Pane

The right pane displays a list of the multidimensional databases (destination databases) included in the import definition. For example, MIS Alea/Infor BI OLAP Server, Microsoft Analysis Services, Cognos TM/1, Jedox Palo, PowerOLAP, and star schema text files display here.



Multidimensional Databases Pane

Debug Pane

ImportMaster allows you to verify that scripts will execute and behave correctly. The Debug pane allows you to monitor the contents of individual variables at run time. This pane can be activated and deactivated for each window by selecting View > Debug Watch from the menu bar. When displayed, the pane appears in the lower portion of the window.

The following table displays the tabs in the Debug pane and provides a brief description of each:

Tab	Description
Watch	This tab allows you check the contents of variables and messages in debug mode. You can enter the variable names in the Name column and the value of the variable will appear in the Value column. The values are displayed in different colors. • When a new break point is reached or stepped through and the value of a monitored variable changes, the value is displayed in red. • If the value does not change, it is displayed in black. • If you enter a variable that does not exist, the text "Cannot display value" will appear in blue.
Auto-Watch	This tab lists all of the variables used in the current script and all global variables.
Log-View	This tab provides information on the progress of the import.

Protecting an Import Definition

The Content Protection Wizard allows you to protect an import definition and any selected objects, such as mappings, scripts, rule sets, and jobs, against unauthorized viewing or modification.

There are two options for protecting an import definition:

- Activate object protection and define a protection password: This option allows you to activate protection for the individual objects after you have specified a password.
- Define password to protect this import definition: This option allows you to protect the import definition with a password. The password is queried every time the import definition is opened, preventing unauthorized access. You can use the Content Protection Wizard to change this password at any time.

If you forget your password or if you no longer have access to the password, you can no longer access the import definition or the individual protected objects.

Demo: Protect an Import Definition

Your instructor will demonstrate how to password protect an import definition.

_____

_____

_____

_____

Exercise 2.3: Protect an Import Definition

In this exercise, you will password protect an import definition.

Exercise Steps

1. Select **Project > Content Protection Wizard**. The **Content Protection Wizard** opens.
2. Verify the **Activate object protection and define a protection password** option is selected.
3. Click **Next**. The **Create or change the password** screen of the **Content Protection Wizard** displays.
4. Type *training* in the **New Password** field.
5. Retype *training* in the **Repeat Password** field.
6. Click **Finish**. The **Content Protection Wizard** closes.

Check your understanding

What are the main components of the ImportMaster user interface?

_____

_____

_____

_____

_____

_____

_____

_____

The Define password to protect this import definition option allows you to activate protection for the individual objects after you have specified a password.

a) True
b) False

Lesson 3: Defining a Relational Database

Estimated Time

30 minutes

Learning Objectives

After completing this lesson, you will be able to:

- Create a relational database connection.
- Use the SQL Query Builder to check a relational database.

Topics

- Creating a Relational Database Connection
- Use the SQL Query Builder to Check the Relational Database

Creating a Relational Database Connection

After creating a new import definition, you use the New Relational Database Connection wizard to define the source database, indicating where the data will be imported from. Infor BI ImportMaster currently supports all ODBC and OLE DB-capable databases and the integration of flat files. The structure of a relational database is read in once and will be stored internally within ImportMaster.

Demo: Create a Relational Database Connection

Your instructor will demonstrate how to create a relational database connection.

_____

_____

_____

_____

Exercise 3.1: Create a Relational Database Connection

In this exercise, you will create a relational database connection.

Exercise Steps

1. Right-click **Relational Databases** in the left pane and select **New relational Database**. The **Database Driver** screen of the **New Relational Database Connection wizard** displays. **Note:** You can also select **Project > Relational database > New relational Database** from the menu bar.
2. Select the **OLE DB** database driver. **Note:** OLE DB stands for Object Linking and Embedding Database, which is an interface that allows connections to be established to databases from OLE DB providers.
3. Click **Next**. The **Database Connect** screen of the **New Relational Database Connection wizard** displays.
4. Click the **Connect…** button. The **Provider** tab of the **Data Link Properties** dialog box displays.
5. Select **Microsoft OLE DB Provider for SQL Server**.
6. Click **Next**. The **Connection** tab displays.
7. Type *or localhost* in the **Select or enter a server name** field.
8. Select the **Use a specific user name and password** option.
9. Type *InforBI* in the **Username** field.
10. Type <*the password provided by the instructor*> in the **Password** field.
11. Select **Allow Saving Password** from the checkbox.
12. Select **CWCars_2013_EN** from the **Select the database on the server** drop-down menu.
13. Click **OK**. The **Connect** dialog box displays.

Note: The Connect dialog box allows you to enter and store custom authentication information in the Login fields, if desired.

14. Click **OK** to return to the **New Relational Database Connection wizard** without entering authentication information.
15. Click **Next**. The **SQL Syntax** screen of the **New Relational Database Connection wizard** displays.
16. Select **Microsoft SQL Server** (Transact SQL).

The SQL syntax will be used by the SQL Query Builder to generate queries; therefore, you should always select a SQL syntax that matches the syntax of the data source. If the appropriate syntax for your relational database is not listed, select the one that is most similar to it. After the SQL Query Builder has generated the SQL statement, you can edit the statement manually.

17. Click **Next**. The **Table Structure and Table Filter** screen of the **New Relational Database Connection wizard** displays allowing you to specify whether or not the table structure of the data source should be read in when the wizard is finished.
18. Verify the **Do not read in the table structure** option is selected. **Note:** Selecting this option saves time by allowing you to display the columns of individual tables only when you need to.

Databases can contain a large number of tables. However, for import purposes, only certain tables are usually required. You can select the “Define a table filter to preselect the tables you need” check box to apply a filter that allows you to select only certain tables. Selecting this check box displays an additional screen in the wizard allowing you to define the table selection. A table filter can only be created if you are using the OLE DB or ODBC driver.

Do not select the Read in the table structure completely (columns and meta data) after finishing the wizard option, as this operation can take a very long time, especially with source systems that have a large number of tables.

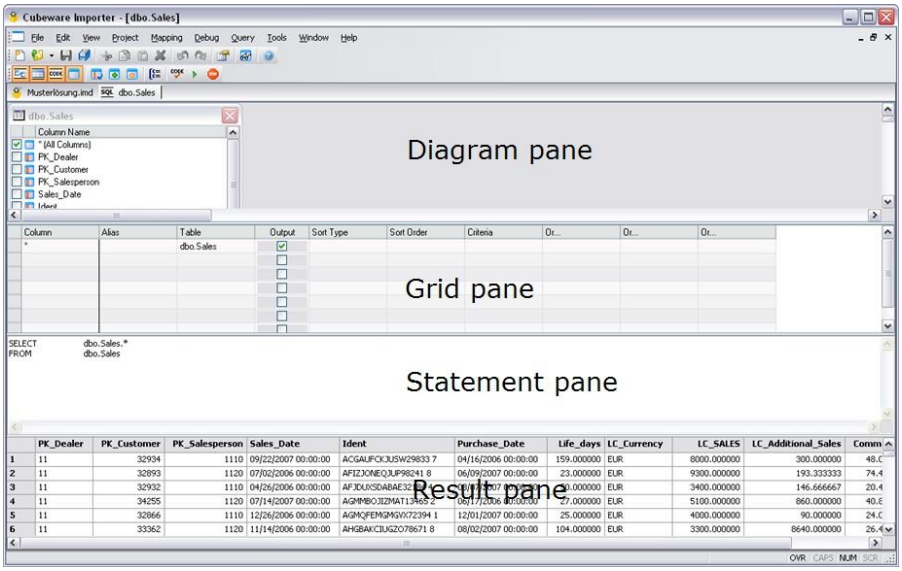
19. Click **Finish** to add the new relational database to the import definition. The new connection appears as **RDB1** below the **Relational Databases** folder.
20. Right-click **RDB1** and select **Properties**. A **Properties of Database ‘RDB1’** dialog box displays the connection’s database driver, alias, and connection information on the **Database Connection** tab.
21. Click **OK** to close the **Properties of Database ‘RDB1’** dialog box.

Using the SQL Query Builder to Check the Relational Database

After you have created a new relational database connection and integrated the SQL server data, you can verify the data that will be used by ImportMaster. You can do this by executing a query through the SQL Query Builder. The SQL Query Builder creates complex SQL queries to a relational database at different points of an import definition.

The SQL Query Builder can be used in the following three instances:

- When checking a database or testing queries to the database.
- In mapping – when defining the query of a result set object.
- In a script – when defining the query using the Result Set Script Wizard.



SQL Query Builder

The table below identifies the panes in the SQL Query Builder and displays a brief description of each:

Pane	Description
Diagram	This pane includes a graphical display of the tables used in the query. You can select columns and join tables in this pane.
Grid	This pane allows you to edit the selected columns. For example, you can assign an alias or define formulas or conditions.
Statement	This pane displays the SQL statement generated by the SQL Query Builder based on the query defined in the Diagram and Grid panes. It uses the SQL syntax selected for the data source for this purpose. **Note:** You can manually create or edit a SQL statement at any time.
Result	This pane displays the result of the SQL statement when issued. The result is for display purposes only and cannot be edited.

The Query menu allows you to manually display and hide the SQL Query Builder panes.

Demo: Use the SQL Query Builder to Create a Query

Your instructor will demonstrate how to use the SQL Query Builder to create a query to verify the relational database connection can be accessed.

_____

_____

_____

_____

Exercise 3.2: Use the SQL Query Builder to Create a Query

In this exercise, you will use the SQL Query Builder to perform a data check by creating a query that joins the dbo.Sales table to the dbo.Vehicles table in order to list the sales revenue from vehicle sales.

Exercise Steps

1. Click the **plus sign (+)** to the left of the **RDB1** folder to expand the folder.
2. Click the **plus sign (+)** to the left of the **Tables** folder to display the database tables.
3. Right-click the **dbo.Sales** table and select **Browse....**. The **Statement** pane displays a query to the table:

```
SELECT      dbo.Sales.*
FROM        dbo.Sales
```

4. Right-click the **SELECT dbo.Sales.* FROM dbo.Sales** query and select **Views > All Panes** to display all SQL Query Builder panes.
5. Right-click in the **Diagram** pane and select **Add Table** to build a new query with another table. The **Add Tables** dialog box displays.
6. Select the **dbo.Vehicles** table.
7. Click **Apply**.
8. Click **Close** to close the **Add Tables** dialog box. The **dbo.Vehicles** table displays to the right of the **dbo.Sales** table.
9. Drag the **VehicleId** column of the **dbo.Sales** table to the **Id** column of the **dbo.Vehicles** table to join the two tables. The join is indicated by a connecting line.
10. Right-click the connecting line and select **Properties**. The **Join Properties** dialog box displays allowing you to specify settings for the connection.
11. Click **Close** to accept the default settings.
12. Clear the ***(All Columns)** check box in the **dbo.Sales** table.
13. Select the **DealerId**, **DateOfSales**, and **SalesPrice** columns in the **dbo.Sales** table.

14. Select all of the column check boxes in the **dbo.Vehicles** table except for the *** (All Columns** and **Id** check boxes. The **Statement** pane is updated with the names of the selected columns.
15. Right-click anywhere in the **dbo.sales** table and select **Execute** to display the result for the current SQL statement.

	DealerId	DateOfSales	SalesPrice	Model	Manufacturer	Colour	Type	Condition	ConstructionMonthYear	
1	WIER	04/25/2012 00:00:00	48674.160000	SEAT IBIZA 1.4 HATTRICK	Seat	red	Sedan	Used Vehicle	8/2000	1
2	WIER	08/08/2012 00:00:00	320.520000	MAZDA 323 5 T 1.4	Mazda	silver	Sedan	Used Vehicle	7/1988	19
3	GEMA	04/13/2012 00:00:00	11718.500000	TOYOTA YARIS 1.0 LINEA TERRA	Toyota	blue	3-door	One Year Old	1/2001	
4	AUHI	09/27/2011 00:00:00	376.050000	TRABANT TRABANT DELUXE	Trabant	blue	2-door	Used Vehicle	5/1990	4
5	AUHI	11/15/2011 00:00:00	11634.840000	LANCIA KAPPA 2.4 LS SW	Lancia	blue	Station wagon	Used Vehicle	7/1997	4
6	AUFI	01/20/2011 00:00:00	13175.820000	VOLKSWAGEN GOLF BON JOVI	Volkswagen	blue	Sedan	Used Vehicle	2/1997	7
7	IMEX	08/24/2011 00:00:00	68227.920000	PORSCHE 993 RS CLUBSPORT	Porsche	red	Coupe	Used Vehicle	9/1997	2

16. Repeat this process to output only the data of **Porsches**. You will need to enter the Porsche criteria in the **Ref** column in the **Grid** pane.
17. Add the **Dealers** table in the **Diagram** pane and join the Nr columns of the **Dealers** table with the **Dealer** column of the **Sales** table if you want to obtain more detailed information about the dealer.
18. Execute the statement again and check the result.

You can also use the SQL Query Builder to create nested queries (in which a query uses the result of a query contained within it). There is no limit to the nesting depth. To add the result of a query to a query rather than a new table, select the Derived Table command rather than the Add Table command when right-clicking in the Diagram pane.

19. Right-click the **dbo.Sales** tab and select **Close**.

Check Your Understanding

In what three instances can you use the SQL Query Builder?

1. _____

_____

_____

2. _____

_____

_____

3. _____

_____

_____

What are the four panes of the SQL Query Builder?

1. _____
2. _____
3. _____
4. _____

Lesson 4: Defining a Multidimensional Database

Estimated Time

30 minutes

Learning Objectives

After completing this lesson, you will be able to:

- Identify the global folders that apply to all multidimensional databases.
- Create a multidimensional database connection.

Topics

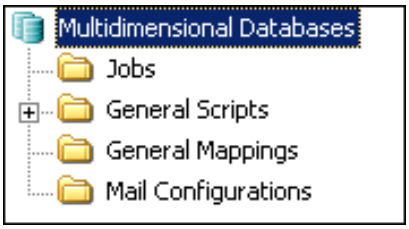
- Creating a Multidimensional Database Connection

Creating a Multidimensional Database Connection

After you have defined a source database and indicated where data will be imported from, you can define a destination database to indicate where the imported data will be written. You define a destination database using the New Multidimensional Database Connection wizard. Infor BI ImportMaster currently interfaces with MIS ALEA and the Infor BI OLAP Server and to Analysis Services for Microsoft® SQL Server®.

Multidimensional Database Folders

An import definition can integrate multiple multidimensional databases; therefore, the folders below the Multidimensional Databases icon apply to the entire import definition, regardless of all OLAP databases defined.



Multidimensional Database Folders

The following table identifies the global multidimensional database folders and provides a brief description of each:

Folder	Description
Jobs	A job consists of different processes and operations, such as the creation of dimensions or execution of mappings. For example, in a job, dimensions can be created and combined to form a cube, which is then filled with data. The different components of a job are successive processes that occur in a fixed order. The various steps involved may consist of different scripts or mappings in which programming controls the various actions individually.
General Scripts	This folder initially contains only the global script, which is where the global settings for the entire import definition can be made. For example, you can set a default value for a variable. The global script is automatically executed first in every job without having to be listed in the job.
General Mappings	This folder initially is empty. It is used to store mappings that don't have a direct relation to a special cube or a special dimension. For example, you could store mappings that deal with user management or other aspects that affecting more than one cube or dimension.
Mail Configurations	This folder stores mail profiles that can be used to send emails with Infor BI ImportMaster, such as messages about data errors.

Demo: Create a Multidimensional Database Connection

Your instructor will demonstrate how to create a multidimensional database connection.

_____

_____

_____

_____

Exercise 4.1: Create a Multidimensional Database Connection

In this exercise, you will create a multidimensional database connection.

Exercise Steps

1. Right-click **Multidimensional Databases** in the left pane and select **New Multidimensional Database Connection**. The **Database Driver** screen of the **New Multidimensional Database Connection wizard** displays.
2. Select the **Infor PM OLAP 10.1 onwards** database driver.
3. Click **Next**. The **Database Connect** screen of the **New Multidimensional Database Connection wizard** displays.
4. Click **Connect…**. The **Connection to MIS Alea** dialog box displays.
5. Verify that **Admin** appears in the **Username** field.
6. Leave the **Password** field blank.
7. Click **Next**. The **Server** dialog box displays.
8. Select the **LOCAL/Academy** OLAP database from the **Server** drop-down menu.
9. Click **Next** to display the **Extended Properties** dialog box.
10. Click **Finish** to return to the New Multidimensional Database Connection wizard.
11. Click **Finish**. The new database displays as **OLAP1** below Multidimensional Databases.
12. Expand the **OLAP1** database icon. Folders for **Processing**, **Dimensions**, **Cubes**, **Scripts**, and **Mappings** appear below the database icon. These folders apply to the OLAP1 multidimensional database only.

Check Your Understanding

What folder below Multidimensional Databases consists of different processes and operations, such as the creation of dimensions or execution of mappings?

a) Jobs
b) General Scripts
c) General Mappings
d) Mail Configurations

What folders are created when you connect to a multidimensional database?

1. _____
2. _____
3. _____
4. _____
5. _____

Lesson 5: Creating Static Dimensions and Elements

Estimated Time

1 hour

Learning Objectives

After completing this lesson, you will be able to:

- Describe static dimensions.
- Create dimensions using the New Dimension wizard.
- Create and execute jobs.
- Manually create static elements in a new dimension.

Topics

- Creating Static Dimensions
- Creating and Executing a Job
- Creating Static Elements

Creating Static Dimensions

After relational (source) and multidimensional (destination) databases have been defined indicating where data is coming from and where it will be written, you need to define the structure of the multidimensional database by creating dimensions, elements, and cubes.

Static dimensions are dimensions that are not created from a source. Typical static dimensions are time dimensions or value-type dimensions. The New Dimension wizard allows you to quickly and easily create a new dimension in a multidimensional database. The wizard takes you through a series of screens allowing you to specify the properties of the dimension and then creates the dimension and adds it to the database definition.

Infor BI ImportMaster does not require a connection to a database be established when defining the database structure for importing data. It is only necessary to establish a connection when defining a relational database or a multidimensional database, and when an import is actually executed. Upon completion of an import, all connections are immediately terminated.

Demo: Create a Time Dimension Using the New Dimension Wizard

Your instructor will demonstrate how to create a Time dimension using the New Dimension wizard.

_____

_____

_____

_____

Exercise 5.1: Create a Time Dimension Using the New Dimension Wizard

In this exercise, you will create a Time dimension using the New Dimension wizard.

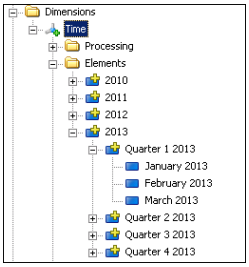
Exercise Steps

1. Right-click the **Dimensions** folder below the **OLAP1** multidimensional database and select **New Dimension**. The **New Dimension** wizard displays.
2. Type *Time* in the **Name** field.
3. Select the **Time dimension** check box. This allows you to save data with a time factor.
4. Click **Next**.
5. Type *<Alea:ODBOType Code="1"/>* in the **Extended Property** box.
6. Click **Next**. The **Time Period** for the **Dimension** screen of the **New Dimension** wizard displays. **Note:** This screen of the wizard only displays when creating a Time dimension.
7. Type *01/01/2010* in the **Begin** field.
8. Type *12/31/2013* in the **End** field. This creates a time dimension period from the first day of 2010 to the last day of 2013.

9. Click **Next**. The **Time Hierarchy** screen of the **New Dimension** wizard displays allowing you to select the type of the elements to be created. **Note:** This screen of the wizard only displays when creating a Time dimension.
10. Select the time hierarchy elements and their display formats using the following data:

Field	Selection	Data
Years	Selected	<yyyy>
Quarters	Selected	Quarter <q> <yyyy>
Months	Selected	<mt> <yyyy>
Weeks	Not selected	
Days	Not selected	

11. Click **Next**. The **Import Settings** screen of the **New Dimension** wizard displays.
12. Verify the **Delete all dimension elements before each import** check box is cleared. **Note:** When this check box is selected, the elements of the dimension will be deleted before the import. If elements already exist in this dimension that are not filled during import, but are needed for the calculation, you should not delete them, since this would prevent access to them.
13. Click **Next**. The next **Import Settings** screen of the **New Dimension** wizard displays.
14. Verify the **Create C elements automatically when required** check box is cleared.
15. Click **Next**. The last **Import Settings** screen of the **New Dimension** wizard displays.
16. Verify the **Use Unknown Element** check box is cleared. **Note:** It is assumed that all dimensions will be assigned the necessary elements and that the data includes the correct element assignments; therefore, for the purpose of this exercise, you do not need to use an unknown element when creating this dimension.
17. Click **Next**. The **final screen** of the **New Dimension** wizard displays.
18. Review the settings you specified.
19. Click **Finish**. The new **Time** dimension displays below the **Dimensions** folder.
20. Expand the **Time** dimension. The **Processing**, **Elements**, **Attributes**, **Subsets**, **Scripts**, and **Mappings** folders associated with the dimension display.
21. Expand the **Elements** folder to display the element years **2010 through 2013**. **Note:** You can only view elements in the Elements subfolder of a dimension when the elements have been manually created or created using the Time dimension wizard.
22. Expand the **2013** and **Quarter 1 2013** elements. The Time dimension element structure should look as follows:



23. Collapse the **Time** dimension.

You can change the properties of a dimension by right-clicking the dimension and selecting Properties.

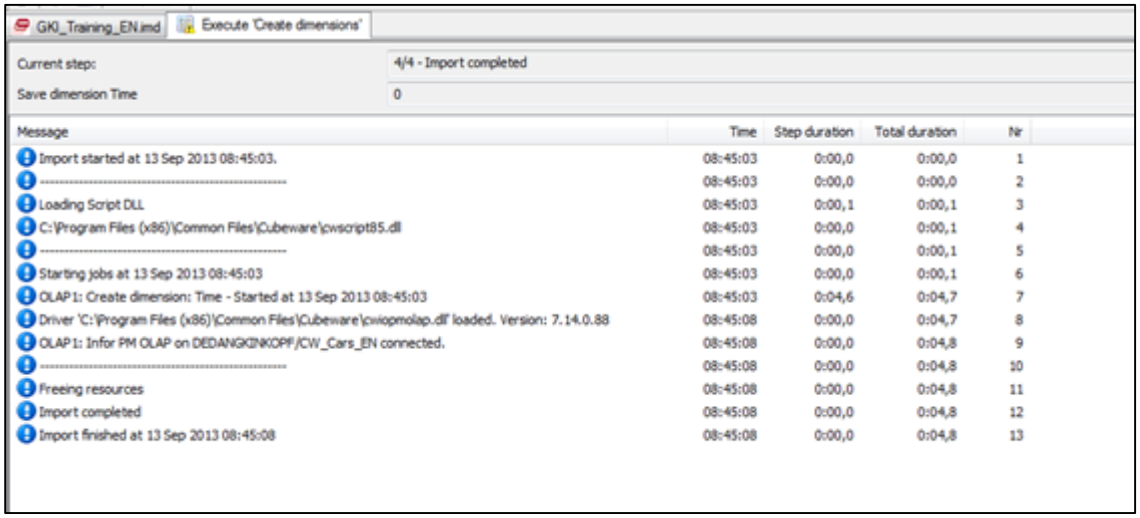
Creating and Executing a Job

All jobs are displayed in a Jobs folder below Multidimensional Databases. A job is used to create and populate cubes and dimensions. Jobs can be programmed to run at periodic intervals or can be started manually.

When you create a Time dimension using the New Dimension wizard, elements that you specify are automatically created within the dimension. In this first step, the dimension structure is created within ImportMaster, but it is not yet actually created in the OLAP database. In order to create the dimension in the OLAP database, the dimension must be dragged into a job. When the job is executed, the dimension will be created in the OLAP database.

Static elements are added to an existing hierarchy; existing hierarchies are not overwritten.

When manually executing a job, all of the job execution messages will display during the execution in the Execute <Job Name> window.



Execute <Job Name> Window

Messages are automatically numbered and time stamped and categorized as follows:

- Notes - identified with a blue icon
- Warnings - identified with a yellow icon
- Errors - identified with a red icon

You can sort the messages by any column, such as Message or Time, by clicking the column heading. When sorting by Message, errors are sorted first, followed by warnings, and then notes.

Demo: Create and Execute a Job

Your instructor will demonstrate how to create and manually execute a job.

_____

_____

_____

_____

Exercise 5.2: Create Execute a Job

In this exercise, you will create and manually execute a job.

Exercise Steps

1. Right-click the **Jobs** folder below **Multidimensional Databases** and select **New Folder**.
2. Type *Automatic Jobs* as the new folder name.
3. Right-click the **Automatic Jobs** folder below **Multidimensional Databases** and select **New Job**.
4. Type *Create Dimensions* as the name of the job.
5. Drag the **Time** dimension and drop it on the **Create Dimensions** job.
6. Right-click the **Create Dimensions** job and select **Execute Job**. The **Execute Create Dimensions** window displays job execution messages.
7. Right-click the **Execute 'Create Dimensions'** tab and select **Close**.

Creating Static Elements

When you create a Time dimension using the New Dimension wizard, elements are automatically created. You can also manually create elements for a dimension by right-clicking the Elements folder below the dimension and selecting either the New N Element, New C Element, New S Element, or New R Element option.

Demo: Create Static Elements

Your instructor will demonstrate how to create static elements within a dimension.

_____

_____

_____

_____

Exercise 5.3: Create Static Elements

In this exercise, you will create a Category dimension, and then manually create the New Car, Used Car, Company Car, and Veteran elements and group them under the All Categories element. This will allow you to analyze cube data by vehicle category.

Exercise Steps

1. Right-click the **Dimensions** folder and select **New Dimension**. The **New Dimension** wizard displays.
2. Type *Condition* in the **Name** field.
3. Verify the **Time dimension** check box is cleared.
4. Click **Next** two times. The **Import Settings** screen of the **New Dimension** wizard displays
5. Verify the **Delete all dimension elements before each import** check box is cleared.
6. Click **Next**. The next **Import Settings** screen of the **New Dimension** wizard displays.
7. Verify the **Create C elements automatically when required** check box is cleared.
8. Click **Next**. The last **Import Settings** screen of the **New Dimension** wizard displays.
9. Verify the **Use Unknown Element** check box is cleared.
10. Click **Next**. The **final screen** of the **New Dimension** wizard displays.
11. Review the settings.
12. Click **Finish**. The new **Category** dimension displays below the **Dimensions** folder.
13. Expand the **Category** dimension. The **Processing**, **Elements**, **Attributes**, **Subsets**, **Scripts** and **Mappings** folders associated with the dimension display.
14. Right-click the **Elements** folder below the **Category** dimension and select **New N Element**. The **Create N Element** dialog box displays.

15. Type *New Car* in the **Element name** field. **Note:** You should always be sure to spell all names correctly when manually creating static elements in order to avoid errors when importing data.
16. Click **OK** to create the New Vehicle static element.
17. Create the following three additional N elements by repeating steps 14-16:
    - **Used Vehicle**
    - **One Year Old**
    - **Veteran**
18. Right-click **Elements** below the **Category** dimension and select **New C Element**. The **Create C Element** dialog box displays.
19. Type *All Conditions* in the **Element name** field.
20. Click **OK**.
21. Drag the **New Vehicle** element and drop it on the **All Conditions** element. A **Copy or Move** dialog box displays.
22. Click the **Move** button.
23. Drag and drop the **Used Vehicle**, **One Year Old**, and **Veteran** elements below the **All Conditions** element.
24. Drag the **Condition** dimension and drop it on the **Create Dimensions** job.
25. Expand the **Create Dimensions** job.
26. Right-click **the OLAP 1: Create dimension: Category** job and select **Execute immediately**. The **Execute: 'OLAP 1': Create Dimension: 'Category'** tab displays job execution messages.
27. Right-click the **Execute: 'OLAP 1': Create Dimension: 'Category'** tab and select **Close**.

Lesson 6: Using Mapping to Create Dimensions

Estimated Time

1 hour

Learning Objectives

After completing this lesson, you will be able to:

- Explain the process and characteristics of mapping.
- Identify the source and destination objects used in mapping.
- Use the Debug feature in a mapping.

Topics

- Creating Dimensions by Mapping
- Debugging a Mapping

Creating Dimensions by Mapping

Mapping is the process by which individual data objects are created and a link between the objects is defined. With mapping, you tell each bit of data in a source object exactly where to go in a destination object. The most important thing to think about prior to importing data is field mapping, which is the process by which data fields in the source object are associated, or mapped, to fields in the destination object.

ImportMaster uses a simple interface—the mapping editor—which allows you to map objects in order to import data from a source object to a destination object without using scripts. Dimensions and cubes can be created and filled with data from a source with almost no programming.

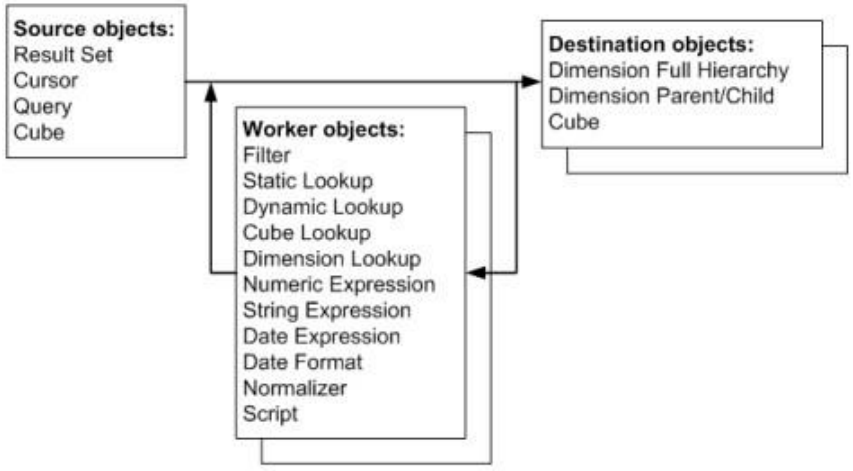
Mapping Characteristics

The following are characteristics of mapping:

- Mapping represents a data flow.
- The data is not manipulated in the mapping data flow; rather, it is forwarded.
- The sequence in which the individual objects are processed depends on the dependencies between the objects.
- If a field in an object contains the value "*__BLOCK__*", the next object will not be processed, even if other input fields in the next object contain valid values.
- A destination object can be followed by another destination object. For example, a cube destination object can be followed by a logging destination object. The logging object contains all of the fields from the cube object and is thus able to log every write operation on the cube.
- A relational table can also be defined as a destination, in which case a result set is used as the destination object.

The Mapping Process

Mapping always involves a single source object and one or more destination objects. There may be any number of worker objects between the source and destination objects.



Mapping Process

The mapping process consists of the following:

- The source object obtains a record. Each column of the source is assigned a value.

- These values are passed along the arrows, and the next object in each case carries out the subsequent processing.
- When all the arrows have been passed through, the destination object can proceed with processing.
- The process then starts again from the beginning.

When creating a mapping in ImportMaster, you must define the source and destination objects.

The following table identifies the source objects available in ImportMaster and provides a brief description of each:

Source Object	Description
Result Set	The result set is a source object based on a SQL statement. A result set is an economical way (utilizing low resources) of obtaining a result list by querying relational databases and then making this data list available for further processing. The data to be transferred is limited to the information defined in the SQL statement.
Dimension Parent/Child	The parent/child dimension source object reads data from an existing dimension and makes it available in a parent/child structure.
Dimension Full Hierarchy	The full hierarchy dimension source object reads data from an existing dimension and makes it available in a full hierarchy structure.
Cube	The cube as a source object is based on an iterator over a cube. This object is comparable to the iterator in the script. Iterators are used to process the data of a cube. An iterator can be limited to particular areas of a cube. For example, you can use an iterator to set all the ACTUAL figures in a cube to 0 or to replace PLAN figures with ACTUAL figures from the previous year. An iterator is always bound to a cube.

The following table identifies the destination objects available in ImportMaster and provides a brief description of each:

Destination Object	Description
Dimension Parent/Child	The parent/child dimension object corresponds to the dimension of an OLAP database. This object is used whenever only the parent element of an element is known in a data record.
Dimension Full Hierarchy	The full hierarchy dimension object corresponds to the dimension of an OLAP database. This object is used whenever the complete hierarchy of an element is known in a data record.
Subset	The subset object allows you to fill a subset in your database.
Cube	The cube destination object corresponds to the cube of an OLAP database.
Result Set	The result set is a destination object based on an SQL statement. A result set makes it possible to write data to relational database tables.
Logging	Logging writes the content of fields into a log file or text file with a logging object.

In this lesson, a specific business scenario will be used to discuss the process of mapping.

Scenario

CW Cars sells a variety of new and used cars. Therefore, there are a considerable number of car models for each manufacturer that need to be accounted for in your database. You will use mapping to populate a new Cars dimension with the manufacturer, model, and type of vehicle information.

Demo: Use Mapping to Create a Dimension

Your instructor will demonstrate how to create a dimension and then create an attribute and a new mapping for the dimension.

_____

_____

_____

_____

Exercise 6.1: Use Mapping to Create a Dimension

In this exercise, you will create a Cars dimension and add a Cartype attribute to account for the type of vehicle. You will then create a new mapping for the dimension.

Exercise Steps

Part 1: Create a dimension and attribute table

1. Following the process outlined in Exercise 5.1 to create a new dimension called **Vehicle** using the **New Dimension** wizard. **Note:** Select the **Delete all dimension elements before each import** check box so that the dimension is newly created each time a job is executed and any changes in the source system will be immediately available in the OLAP database.
2. Right-click the **Attributes** subfolder below the **Cars** dimension and select **New Attribute**. The **Edit Attribute** dialog box displays.
3. Enter the following criteria:

Field	Data
Table	1
Fieldname	Vehicletype
Type	String
Length	50

4. Click **OK**.

Part 2: Create a new mapping

1. Right-click the **Mappings** folder within the **Cars** dimension and select **New Mapping**.
2. Type *Create Vehicle Dimension* as the mapping name.
3. Double-click **Create Vehicle Dimension**. An empty window opens.

Part 3: Define the source object

1. Right-click in the empty window and select **Source Objects**. The **Source Objects** menu displays. To create the Cars dimension, you will access the CWCars relational database using a result set. All vehicle models present in the vehicle master data will be created as elements of the dimension and sorted by manufacturer.
2. Select **Result Set**. The **Result Set** wizard displays.

In virtually all the objects of a mapping, all input fields can also be used as output fields. The fields of a destination object cube can, for instance, be used to pass data to a logging object.

3. Type *Vehicles* in the **Name** field.
4. Type *Vehicles Table of CWCars* in the **Description of the object** field.
5. Click **Next**. The **Relational Database Connection** screen of the **Result Set** wizard displays with **CWCars** already selected as **RDB1**.
6. Click **Next**. The **Source** screen of the **Result Set** wizard displays. To group the result list together, we are going to add a Group By clause for the selected columns.
7. Click **SQL Query Builder…**. The **Add Tables** dialog box displays.
8. Select **dbo.Vehicles** and click **Apply** to add the dbo.Vehicles table to the SQL Query Builder. The **dbo.Vehicles** window opens.
9. Click **Close** to close the Add Tables dialog box.
10. Select the **Model**, **Manufacturer**, and **Type** columns in the **dbo.Vehicles** window.
11. Right-click in the **dbo.Vehicles** window and select **Functions**. A **Group By** column is added in the Grid pane of the SQL Query Builder, in which different functions can be selected.
12. Leave the functions for the rows set to **Group By**.
13. Review the statement in the **Statement** pane. It should display as follows:

```
SELECT   dbo.Vehicles.Manufacturer, dbo.Vehicles.Model, dbo.Vehicles.Type
FROM     dbo.Vehicles
GROUP BY dbo.Vehicles.Manufacturer, dbo.Vehicles.Model, dbo.Vehicles.Type
```

14. Right-click anywhere in the **Diagram** pane of the **SQL Query Builder** and select **Execute Query** to display the result for the current SQL statement.
15. Review the results of the query execution and accept the query by clicking the **green check mark**. The **Source** screen of the Result Set dialog box displays with the SQL Query statement.
16. Click **Next**. The **Creation of Field List** screen of the **Result Set** wizard displays
17. Verify **Automatically (recommended)** is selected. This option specifies how the field list is to be created. An attempt will made to get the field list from the database. If there is no connection to the database, the field list is created without accessing the database.

The field list initially displays the columns of the SQL query. Previously used aliases of the SQL statement are used here. In the mapping, the different fields are the placeholders for the corresponding values of the result list, which are processed record by record. The names of the fields can also be edited manually.

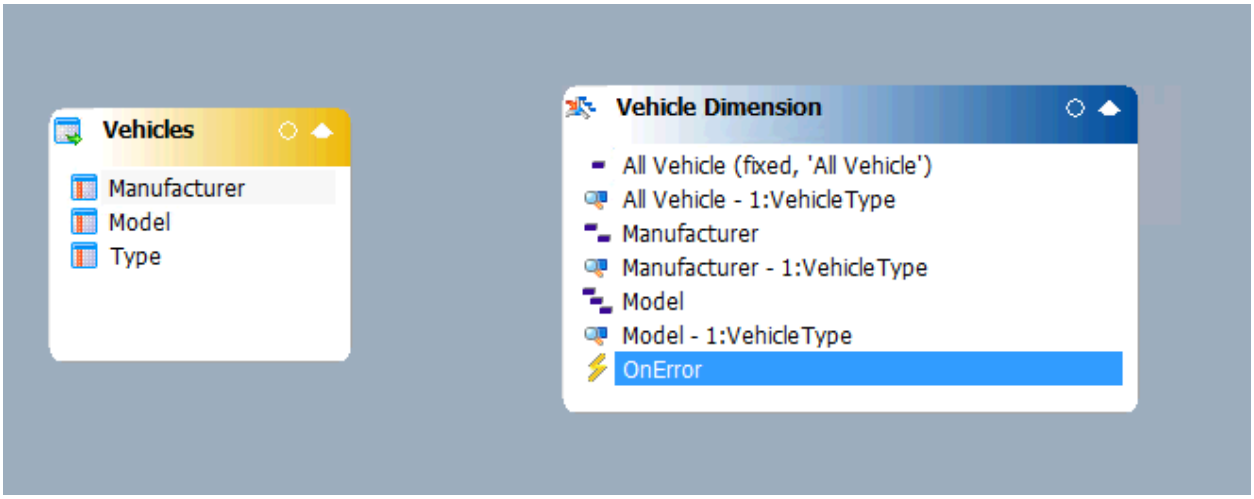
18. Click **Next**. The **Output Fields** screen of the **Result Set** wizard displays. From here, you can edit the various field names or change the order in which they occur. You will not modify this screen in this exercise.
19. Click **Finish** to complete defining the result set. The object appears in the mapping editor.

You can use the context menu to change the Properties settings at any time.

Part 4: Define the destination object

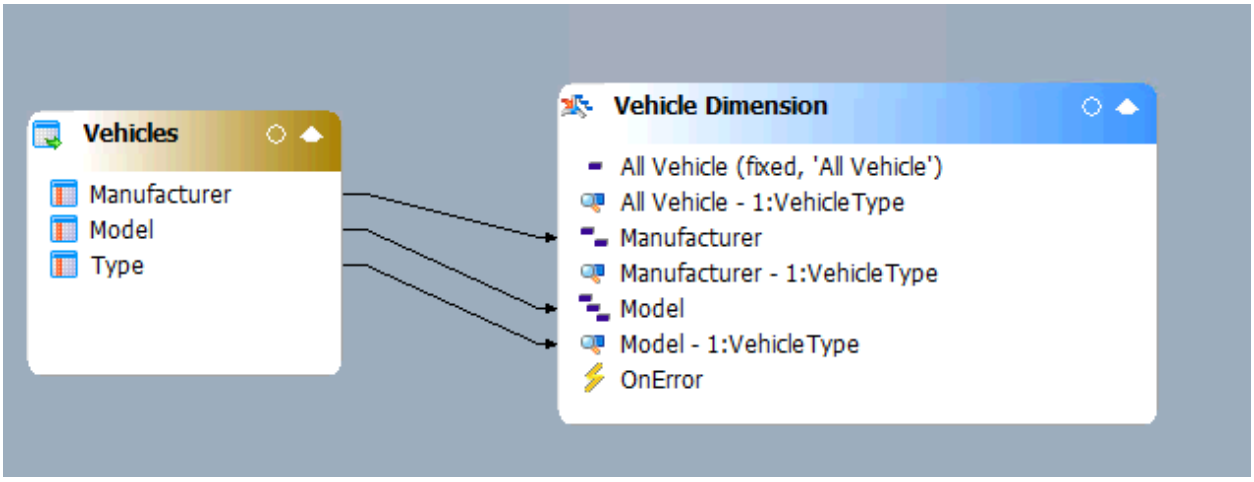
1. Select **Mapping > Destination Objects > Dimension Full Hierarchy** from the menu bar. The **Name** dialog box opens.
2. Type *Vehicle* in the **Name** field.
3. Type *Vehicle Dimension* in the **Description of the object** field.
4. Click **Next**. The **Dimension** screen displays.
5. Click the **Browse (…)** button next to the **Dimension** field. The **Select Dimension** dialog box opens.
6. Select **Vehicle** and click **OK**. You return to the **Dimension** dialog box with **OLAP1** displayed in the **Alias** field and **Vehicle** displayed in the **Dimension** field.
7. Select the **Hide weighting fields** check box, so that the **Weight** input field is not shown in the destination module.
8. Click **Next**. The **Levels** screen displays allowing you to define the individual levels. Currently the Cars dimension consists of only one level. You will add two levels so that the Cars dimension consists of three levels: All Vehicles, Manufacturer, and Model.
9. Click the **Add** button. **Level 2** appears below the **Top Level**.
10. Click the **Add** button. **Level 3** appears below **Level 2**.
11. Select **Top Level** in the list and click **Edit**. The **Edit Level** dialog box opens allowing you to change the level name.
12. Type *Total* to replace the **Top Level** text in the in the **Level Name** field.
13. Select the **Fixed value for element** check box to store the element as a fixed element. The field to the right of the check box becomes activated.
14. Type *All Vehicles* in the field.
15. Click **OK** to return to the **Levels** dialog box.
16. Type *Manufacturer* to rename **Level 2**.
17. Type *Model* to rename **Level 3**.
18. Click **Next**. The **Attributes** screen displays. From here, you can create attributes to be filled in the dimension object. In this exercise, the required Cartype attribute has already been defined for the Cars dimension.

19. Click **Finish**. The destination object displays in the **Mapping Editor**.



Part 5: Define the data flow by creating joins

1. Drag **Manufacturer** in the **Vehicles** source object and drop it on **Manufacturer** in the **Vehicle** destination object. A dialog box displays asking if you want to automatically join fields with the same name between objects.
2. Select the **Join selected mapping fields** option to join the fields with mapping.
3. Click **OK**. A connector line appears between the two objects indicating they are joined.
4. Join **Model** in the **Vehicles** source object with **Model** in the **Vehicle** destination object and join **Type** in the source object to **Model – 1:Vehicletype** in the destination object, using the same method.



For each record, the content read from the Manufacturer column is created as an element of the Cars dimension, the model information is created as a child N element, and for this N element, the Cartype attribute is assigned the value of the Type field.

5. Select **File > Save** to save the mapping.
6. Right-click the **Create Vehicle Dimension** tab and select **Close**. The window closes.

7. Select **File > Save** to save the .imd file.

Part 6: Execute a mapping

Note: Before a mapping can be executed, it has to be added to a job and the job has to be started.

1. Drag the **Cars** dimension and drop it on the **Create Dimensions** job.
2. Drag the **Create Cars Dimension** mapping and drop it on the **Create Dimensions** job.
3. Right-click the **OLAP1: Create dimension: Cars** job and select **Execute immediately**.
4. Check the output in the **Execute 'OLAP1: Create dimension: Cars'** window and then close the window.
5. Right-click the **OLAP1: Create dimension: Cars** job and then click the **Browse (...)** button to review the result.

Debugging a Mapping

ImportMaster allows you to debug mappings. Debugging performs tests to determine if a mapping is executable and behaves correctly.

Note: You cannot modify a mapping while it is in debug mode.

There are two ways to start the debugging process:

- Execute an entire job in debug mode. You can also check several mappings or scripts one after the other. This method should be used if the mapping is to be tested as part of a complex execution sequence.
- Debug an open mapping. Only the mapping that is currently open is processed.

Debug Menu

When a mapping is open, the Debug menu allows you to access all the debugging commands.

The following table identifies the commands on Debug menu provides a brief description of each:

Command	Description
Start Debug (F5)	This command starts debugging and runs until the first breakpoint. During debugging: Runs until the next breakpoint. When you use this command, ensure that you have defined a breakpoint at which execution of the mapping will be halted.
Break (Ctrl+F9)	Halt at the current processing position in the mapping.
Step Into (F11)	The selected object is executed the step-by-step. Changes to the output fields can be monitored in the Watch window. If the object is a script object, the debugger goes to the first line of the script and then stops again.
Step Over (F10)	The selected object is executed and the debugger stops again when the next object is reached. The next object is identified by the numbering.
Run to current position (Shift+F11)	Debugging is executed until the next market object is reached.
Stop Debug (Shift+F5)	This command Stops the debugger. It is only available when the debugger is running.
Toggle Breakpoint (F9)	This command sets or removes a breakpoint on the selected object.
Conditional Breakpoint...	The conditional breakpoint can be inserted for the selected object. The condition can be set either on the basis of the number of cycles or by comparing a value with the contents of a variable. The same conditions can be used as for a filter. The type of breakpoint and the condition are specified in the Breakpoint dialog box.
Add to Watch (F8)	A marked field within the mapping object can be added to the Watch function. This command is only available while the debugger is running.

To start the debugging of the mapping, you can use the Start Debug, Step Into, Step Over, or Run to current position command, depending on when you want the debugger to stop next. At every point at

which the execution of the mapping is halted, you can review the values of variables or use the menu commands to execute mapping step-by-step or up to the next breakpoint.

Once execution of the mapping has been halted, you can set new breakpoints and remove existing ones. The variables are displayed in the lower part of the work window, in the Debug Watch and Debug Auto-Watch areas where they can be monitored.

Demo: Use the Debugger

Your instructor will demonstrate how to use the debugger on a mapping.

_____

_____

_____

_____

Exercise 6.2: Use the Debugger

In this exercise, you will use the debugger on the Create Cars Dimension mapping.

Exercise Steps

1. Open the **Create Vehicle Dimension** mapping.
2. Right-click the **Vehicles** object and select **Toggle Breakpoint**. A red dot, indicating the breakpoint, appears to the right of the mapping object.
3. Select **Debug > Start Debug** from the menu bar. The debugging process starts and the Debug Watch pane opens allowing you to monitor a selection of fields you define. To do this, the fields can be entered manually in the Name column or by choosing Add to Watch from the context menu of the object field allowing you to inspect the contents of the fields. As soon as the first breakpoint is reached, the object is highlighted in yellow with a number in the mapping to indicate that the current status. The mapping objects have been numbered according to the sequence in which they are processed.
4. Click the **Auto-Watch** tab in order to display all field contents grouped by object.
5. Click the plus sign **(+)** next to **<a field>** to expand the information.
6. Select **Debug > Toggle Breakpoint** to remove the breakpoint.
7. Select **Debug > Step Into** to run through the remaining records without the breakpoint.

If the "NOT SET" value is output for a field, no content has yet been assigned to the field. If you enter a variable that does not exist, "Cannot display field" is shown in blue.

8. Close the **Create Vehicle Dimension** mapping. **Note:** If the debug process is not complete, a dialog box will appear asking if you want to stop the debug process. Click **OK**.

Check Your Understanding

List the source and destination objects available in ImportMaster and provide a brief description of each.

_____

_____

_____

_____

_____

_____

_____

_____

You must ensure that you have defined a breakpoint at which the execution of a mapping will be halted when you use which of the following commands? Select all that apply.

a) Start Debug
b) Break
c) Step Into
d) Step Over
e) Toggle Breakpoint
f) Stop Debug

Lesson 7: Using Scripts to Create Dimensions

Estimated Time

1 hour

Learning Objectives

After completing this lesson, you will be able to:

- Identify the main programming features of CWScript.
- Create a dimension using CWScript.
- Debug a script.

Topics

- Using CWScript
- Debugging Scripts

Using CWScript

ImportMaster uses the CWScript programming language, by means of which even complex data imports can be defined and executed efficiently. Just as with other programming languages, CWScript has its own grammar and syntax.

By means of an integrated script language the structures of the dimensions can be adapted to suit the requirements of any company. Calculations and plausibility checks of the source data can be carried out in the scripts that are executed when dimensions are created or data is imported. In addition, conditions for data selection can be queried.

You can use CWScript to:

- Check the extraction of the data from the source system.
- Specify what is to be done with the extracted data, whether records are to be transformed or ignored.
- Create the structures (dimensions) of the OLAP database.
- Adapt existing OLAP structures on the basis of new information from the source system.
- Control the data import to the OLAP database.

CWScript is case-sensitive and uses syntax highlighting. Commands, strings, values and comments appear in different colors.

Infor BI ImportMaster Help

Common Components > TCL - script language

The following table identifies some of the main features of CWScript and provides a brief description of each:

Feature	Description
Command Call	As in many other programming languages, CWScript only has functions. A function always delivers a return value. If the return value of the function is not assigned to a variable, it is automatically rejected. The interpreter is keyword-driven and the first word in each line is a command. The basic syntax looks like this: command [arg1 arg2 arg3 ...] “Command” represents the name of a built-in command or of a function you have written. Command names and their arguments [arg1 arg2 arg3 ...] are separated by either spaces or tab characters. A command always ends with a line feed or a semicolon.
Command Reference	Groups of commands that belong together appear in the same section. The following rules apply to all the syntax descriptions: • A normal word describes a command name, a condition, or a statement. • A word in pointed brackets (<>) specifies a mandatory parameter. • A word in square brackets ([ ]) specifies an optional parameter. • An ellipsis signifies a continuation of the entry in the same style. Therefore, it does not matter whether there are one or more or no repetitions. • All other characters, such as curly brackets, must be taken exactly as they are written.

Feature	Description
Return values	Commands that return a value must be placed in square brackets ([ ]) to ensure that substitution works correctly and the return value can be obtained.
Comments	Comments always begin with the number sign (#). A comment is treated as a separate command, but one that has no effect. The command character must appear at the beginning of a line, so that the characters that come after it are not processed as code. Example: # A simple comment set *a* 10; # Comment at the end of a line
Variables	Variables do not have to be declared before they can be used; they are created automatically the first time they are used. Variables do not have a type, which means that the programmer must ensure in the case of numerical commands, for example, that the variables really do contain numerical values. If the validity range of a variable is violated, the variable is automatically destroyed. Variable names can be any length and can contain any character. They are case sensitive, which means that a distinction is drawn between uppercase and lowercase. The set command is used to assign a value to a variable. This command requires two arguments: the name of the variable to which a value is to be assigned and the value itself.
Command Substitution	Command substitution is a variant of general substitution. In contrast to other programming languages, the interpreter does not recognize function calls itself.

Demo: Use a Script to Create a Dimension

Your instructor will demonstrate how to use a script to create a dimension.

_____

_____

_____

_____

Exercise 7.1: Use a Script to Create a Dimension

In this exercise, you will create a Cars dimension using a script in order to make a direct comparison between using mapping to create a dimension and using a script to create a dimension.

Exercise Steps

1. Right-click the **Scripts** subfolder below the **Vehicle** dimension and select **New Script**.
2. Type *Vehicle Script* as the name.
3. Open the **Vehicle Script**.

   Three components are required in the script: the connection to the relational database, the connection to the dimension and the processing of the data (i.e. the creation of the elements in the dimension), connection to the source database, and definition of the query. The Result Set Script Wizard is available in the script editor for defining the connection to the relational database. The wizard helps you to define the query and inserts the corresponding lines of code in the script. The steps involved in the Result Set Script Wizard correspond to the definition of a result set object in a mapping.
4. Right-click in the **Vehicle Scripts** window and select **Insert Result Set**. The **Result Set Script** Wizard opens.
5. Type *ResultsetVehicle* in the **Name** field. This name will be used in the code of the result set as a variable for the result set object.
6. Type *Create Dimension Vehicle, dbo.Vehicles table of CWCars* in the **Description of the object** field. The description will be added to the script as a comment.
7. Click **Next**. The **Relational Database Connection** screen of the wizard displays with the **CWCars** relational database selected.
8. Click **Next**. The **Cursor Type** screen of the wizard displays. You can use the cursor type to determine the navigation method of the cursor and specify whether the result set is to be opened for writing.
9. Click **Next** to accept the default Forward only selection. The **Source** screen of the wizard displays. From here, you must define the query using the SQL Query Builder just as you defined the result set object in the mapping.
10. Start the **SQL Query Builder**, add the **dbo.Vehicles** table, select the **Model, Manufacturer**, and **Type** columns, and activate the **Group By** function. Type the following statement:

```
SELECT   dbo.Vehicles.Model, dbo.Vehicles.Manufacturer, dbo.Vehicles.Type
FROM     dbo.Vehicles
GROUP BY dbo.Vehicles.Model, dbo.Vehicles.Manufacturer, dbo.Vehicles.Type
```

11. Click the **green check mark** to accept the query. The **Source** screen of the **Result Set Script** Wizard displays with the SQL Query statement.
12. Click **Next**. The **Creation of Field List** screen of the **Result Set Script** Wizard displays
13. Verify **Automatically (recommended)** is selected.
14. Click **Next**. The **Output Fields** screen of the **Result Set Script Wizard** displays.
15. Select the **Manufacturer** field and click **Edit**. The **Edit Mapping** field dialog box opens.
16. Type *s_Manufacturer* in the **Field Name** field. The field name is used in the script as a variable. The preceding "s" stands for "string" and indicates to the user the type of the variable's contents.
17. Click **OK** to return to the Output Fields screen.
18. Rename the **Model** and **Type** fields to **s_Model** and **s_Type**.

These prefixes are not mandatory; however, it is recommended that you use meaningful and unique object and variable names and assign prefixes to indicate the type of the object or variable. These conventions make it easier to read and edit scripts.

19. Click **Finish** to generate the code.

```
# Generated code from script wizard by Infor BI ImportMaster 10.5.0


RDB1 NewResultSet ResultsetVehicle

ResultsetVehicle Open {SELECT    dbo.Vehicles.Model, dbo.Vehicles.Manufacturer, dbo.Vehicles.Type\
FROM  dbo.Vehicles\
GROUP BY dbo.Vehicles.Model, dbo.Vehicles.Manufacturer, dbo.Vehicles.Type}  forwardonly

ResultsetVehicle BindCol {Model} s_Model
ResultsetVehicle BindCol {Manufacturer} s_Manufacturer
ResultsetVehicle BindCol {Type} s_Type

while {[ResultsetVehicle MoveNext] == 1} {

# TODO: Insert your program code here to read out the Result Set data.

}

DeleteResultSet ResultsetVehicle
```

The following table displays the different command lines and their meanings.

Command Line	Description
RDB1 NewResultSet ResultSetVehicle	NewResultSet creates a new result set for the data source RDB1 and gives it the object name ResultSetVehicle.
ResultsetVehicle Open {SELECT dbo.Vehicles.Manufacturer, dbo.Vehicles.Model,dbo.Vehicles.Type\ FROM dbo.Vehicles \ GROUP BY dbo.Vehicles_Manufacturer, dbo.Vehicles.Model,dbo.Vehicles.Type}	The result set ResultSetVehicle is opened by issuing the "SELECT …." query.
ResultSetVehicle BindCol {Manufacturer} sFK_Manufacturer ResultsetVehicle BindCol {Model} sModel ResultsetVehicle BindCol {Type} sType .	The "BindCol" command assigns the required columns of the result list of the result set (in this case the Type column, for instance) to a variable (in this case sType, for instance).
while {[ResultSetVehicle MoveNext] == 1} {…}	A while loop is required to process the result list of the result set record by record. This is repeated while the condition that there is a further record available ({[ResultSetType MoveNext] == 1) is met.
DeleteResultSet ResultSetVehicle	Closes the query and deletes the result set

Connection to the Destination Database

The connection to the relational database was created in its entirety by the Result Set Script Wizard. However, additional commands must be added so that the script will also process the information of the result list and create the elements in the dimension. The required commands are described individually below and then added to the script in the correct order.

In order to create the Vehicle dimension using a script, the dimension must be addressed in the code. Two new script lines are required for this:

Command Line	Description
`OLAP1 SelectDimension "Vehicle" dimVehicle`	The Cars dimension is selected with "SelectDimension" and assigned the object name "dimVehicle". Add at the beginning of the script
`CloseDimension dimVehicle`	This closes and saves the dimension. Add at the end of the script

Data Processing

The individual C and N elements are created using the information from the data source:

```
set sTotal "All Vehicles"

dimVehicle NewCElement $sTotal

dimVehicle NewCElement $s_Manufacturer

dimVehicle MemberOf $sTotal

dimVehicle NewNElement $s_Model

dimVehicle PutAttrib "1:VehicleType" $s_Type

dimVehicle MemberOf $s_Manufacturer
```

The line 'set sTotal "All Vehicles"' assigns the contents of "All Vehicles" to the variable sTotal. Then the "All Vehicles" element is created as a C element with the command NewCElement.

The NewCElement command creates a C element for the dimension dimVehicle with the contents of the variable sFK_Manufacturer as its name.

The MemberOf command inserts the new C element as a member of the "All Vehicles" element.

The NewNElement command creates an N element for the dimension dimVehicle with the contents of the variable sModel as its name.

The PutAttrib command assigns the contents of the variable sType to the "VehicleType" attribute of the N element you have just created.

The MemberOf command inserts the N element below the previously created C element in the hierarchy.

To complete the script, add the various commands to the script. Be sure to add them in the correct order.

The whole script should look like this

```
# Generated code from script wizard by Infor BI ImportMaster 10.5.0

OLAP1 SelectDimension "Vehicle" dimVehicle
RDB1 NewResultSet ResultsetVehicle


ResultsetVehicle Open {SELECT    dbo.Vehicles.Model, dbo.Vehicles.Manufacturer, dbo.Vehicles.Type\
FROM  dbo.Vehicles\
GROUP BY dbo.Vehicles.Model, dbo.Vehicles.Manufacturer, dbo.Vehicles.Type}  forwardonly

ResultsetVehicle BindCol {Model} s_Model
ResultsetVehicle BindCol {Manufacturer} s_Manufacturer
ResultsetVehicle BindCol {Type} s_Type


while {[ResultsetVehicle MoveNext] == 1} {


set sTotal "All Vehicles"
dimVehicle NewCElement $sTotal
dimVehicle NewCElement $s_Manufacturer
dimVehicle MemberOf $sTotal
dimVehicle NewNElement $s_Model
dimVehicle PutAttrib "1:VehicleType" $s_Type
dimVehicle MemberOf $s_Manufacturer
}

DeleteResultSet ResultsetVehicle
CloseDimension dimVehicle
```

Creating the Dimension

20. Replace the mapping for creating the **Vehicle** dimension with the script you have just created in the **Create Dimensions** job.
21. Start the job.
22. Check the result.

Debugging Scripts

Just as with mapping, ImportMaster allows you to debug scripts. Debugging performs tests to determine if a script will execute and behave correctly.

Note: You cannot modify a script while it is in debug mode.

There are two methods to start script debugging:

- By debugging an open script.
- By starting an entire job in debug mode.

To give you a better idea of the execution of the script and let you see what happens in the various different steps, you can also execute the script line by line and inspect it. As is the case with mapping, the script can be tested as part of the job or directly under the script editor in debug mode.

The commands available for debugging a script are the same as those available for debugging a mapping.

The script is executed in the background during debugging. Scripts may have to be executed in their entirety in debug mode to prevent an incomplete status or even data loss from occurring. To test database-independent script commands (e.g. formatting or string handling), these parts of the script can be copied to a new script and checked there using the debugger.

Demo: Debug a Script

Your instructor will demonstrate how to debug a script.

_____

_____

_____

_____

Exercise 7.2: Debug a Script

In this exercise, you will debug the Create Cars Dimension script.

Exercise Steps

1. Open the **Create Cars Dimension** script.
2. Select **Debug >Toggle Breakpoint** to insert a breakpoint in the **RDB1 NewResultSet...** line. A red dot, indicating the breakpoint, appears to the left of the line in the script.
3. Select **Debug > Start Debug** from the menu bar. When the initial breakpoint is reached, the line is highlighted in yellow in the script to indicate the current status. As soon as a variable is defined in the line of the script currently being processed during debugging, the variable and its content are listed in the Auto-Watch tab of the Debug-Watch pane. In addition, the variables can be entered manually in the Name column of the Watch tab or can be added by

choosing the **Add to Watch** command from the context menu for the variables in the script editor.

4. Select **Debug > Step Into** to run through the script line by line and check the contents of the variables sTotal, sManufacturer, sModel and sType in the Debug-Watch pane.

Demo: Script-Controlled Creation of the Customer Dimension

Your instructor will demonstrate how to create a Customer dimension and then create a new script for the Customer dimension.

_____

_____

_____

_____

Exercise 7.3: Script-Controlled Creation of the Customer Dimension

In this exercise, you will create a Customer dimension and then create a new script for the Customer dimension. The option of analyzing the sales per customer should provide optimum analysis of the data. To allow this, a Customer dimension is required. The customers should be grouped by country and the top level element should be "All Customer".

Exercise Steps

1. Create a **Dealer** dimension following the steps in Exercise 5.1.
2. Enable the **Delete all dimension elements before each import** option.
3. Create the following attributes:

Field Name	Type	Length
Address	String	250
Name	String	50

4. Analyze the relational database and determine what data should be included.
5. Create a new script for the **Dealer** dimension and name it *Dealer Script* following the steps in Exercise 7.1.
6. Open the script and start the **Result Set Script Wizard** to define the source data to be included.
7. Define a query that takes the columns **Id**, **Name**, **Street**, **City ZIPCode** and **CountryName** from the **dbo.Dealers** table following the steps in Exercise 7.1. The code generated by the Result Set Script Wizard should look like this:

```
# Generated code from script wizard by Infor BI ImportMaster 10.5.0


RDB1 NewResultSet ResultsetDealer

ResultsetDealer Open {SELECT  dbo.Dealers.Id, dbo.Dealers.Name, dbo.Dealers.Street, dbo.Dealers.City, dbo.Dealers.ZIPCode,
   dbo.Dealers.CountryName\
FROM  dbo.Dealers}  forwardonly

ResultsetDealer BindCol {Id} s_Id
ResultsetDealer BindCol {Name} s_Name
ResultsetDealer BindCol {Street} s_Street
ResultsetDealer BindCol {City} s_City
ResultsetDealer BindCol {ZIPCode} s_ZIPCode
ResultsetDealer BindCol {CountryName} s_CountryName

while {[ResultsetDealer MoveNext] == 1} {

# TODO: Insert your program code here to read out the Result Set data.

}

DeleteResultSet ResultsetDealer
```

8. Add the following code to the script.

At the beginning: *OLAP1 SelectDimension "Dealer" dimDealer*

At the end: *CloseDimension dimDealer*

Bellow the **#TODO**, the following lines:

set sTotal "All Dealer"

dimDealer NewCElement $sTotal

dimDealer NewCElement $s_CountryName

dimDealer MemberOf $sTotal

dimDealer NewNElement $s_Id

dimDealer PutAttrib "1:Address" "$s_Street, $s_City, $s_CountryName"

dimDealer PutAttrib "1:Name" $s_Name

dimDealer MemberOf $s_CountryName

The script should look like this:

```
# Generated code from script wizard by Infor BI ImportMaster 10.5.0

OLAP1 SelectDimension "Dealer" dimDealer
RDB1 NewResultSet ResultsetDealer

ResultsetDealer Open {SELECT  dbo.Dealers.Id, dbo.Dealers.Name, dbo.Dealers.Street, dbo.Dealers.City, dbo.Dealers.ZIPCode, \
   dbo.Dealers.CountryName\
FROM  dbo.Dealers}  forwardonly

ResultsetDealer BindCol {Id} s_Id
ResultsetDealer BindCol {Name} s_Name
ResultsetDealer BindCol {Street} s_Street
ResultsetDealer BindCol {City} s_City
ResultsetDealer BindCol {ZIPCode} s_ZIPCode
ResultsetDealer BindCol {CountryName} s_CountryName

while {[ResultsetDealer MoveNext] == 1} {


set sTotal "All Dealer"
dimDealer NewCElement $sTotal
dimDealer NewCElement $s_CountryName
dimDealer MemberOf $sTotal
dimDealer NewNElement $s_Id
dimDealer PutAttrib "1:Address" "$s_Street, $s_City, $s_CountryName"
dimDealer PutAttrib "1:Name" $s_Name
dimDealer MemberOf $s_CountryName


}

DeleteResultSet ResultsetDealer
CloseDimension dimDealer
```


To display help for a command (blue font), position the cursor on the command and press **F1**.

9. Right click on the **script** name to add the dimension and script to the **Create dimensions** job and start the job.
10. Check the completed dimension and the contents of the attribute tables.

Lesson 8: Creating Other Dimensions

Estimated Time

1 hour

Learning Objectives

After completing this lesson, you will be able to:

- Create a Currency dimension.
- Create a Measure dimension.

Topics

- Creating a Currency Dimension
- Creating a Measure Dimension

Creating a Currency Dimension

CWCars operates in several countries and therefore, different currencies will need to be taken into account when creating a controlling system. This will enable comparisons to be made between areas using different currencies.

Demo: Create a Currency Dimension

Your instructor will demonstrate how to create a Currency dimension.

_____

_____

_____

_____

Exercise 8.1: Create a Currency Dimension

In this exercise, you will create a Currency dimension.

Exercise Steps

1. Right-click the **Dimensions** folder and select **New Dimension**. The **New Dimension** wizard displays.
2. Type *Currency* in the **Name** field.
3. Verify the **Time dimension** check box is cleared.
4. Click **Next** two times. The **Import Settings** screen of the **New Dimen**sion wizard displays
5. Verify the **Delete all dimension elements before each import** check box is cleared.
6. Click **Next**. The next **Import Settings** screen of the **New Dimension** wizard displays.
7. Verify the **Create C elements automatically when required** check box is cleared.
8. Click **Next**. The last **Import Settings** screen of the **New Dimension** wizard displays.
9. Verify the **Use Unknown Element** check box is cleared.
10. Click **Next**. The **final screen** of the **New Dimension** wizard displays.
11. Review the settings.
12. Click **Finish**. The new **Currency** dimension displays below the **Dimensions** folder.
13. Expand the **Currency** dimension. The **Processing**, **Elements**, **Attributes**, **Subsets**, **Scripts** and **Mappings** folders associated with the dimension display.
14. Right-click the **Elements** folder below the **Currency** dimension and select **New N Element**. The **Create N Element** dialog box displays.
15. Type *EUR* in the **Element name** field.

16. Click **OK**.
17. Right-click **Elements** below the **Currency** dimension and select **New C Element**. The **Create C Element** dialog box displays.
18. Type *LOCAL* in the **Element name** field.
19. Click **OK**.
20. Create the following N elements below the **LOCAL** element:
    - **USD**
    - **CHF**
    - **PLN**
21. Right-click the **USD** element and select **Properties**. The **Properties** dialog box opens.
22. Click the **Weighting** tab.
23. Type *0* in the **Weighting** field.
24. Click **Apply**.
25. Click **OK**.
26. Change the weighting to **0** for the **CHF** and **PLN** elements.
27. Drag the **Currency** dimension and drop it on the **Create Dimensions** job.
28. Right-click the **OLAP 1: Create dimension: Currency** job and select **Execute immediately**. The **Execute: 'OLAP 1': Create Dimension: 'Category'** tab displays job execution messages.
29. Right-click the **Execute: 'OLAP 1': Create Dimension: 'Currency** tab and select **Close**.

Creating a Measure Dimension

So far in this course, you have created a Time dimension to identify time periods for your controlling system, a Category dimension to categorize the category of cars (used, new, and so on), a Cars dimension to provide information about the types cars (make, model, and so on), and a Currency dimension to account for the different currencies in the system.

You now need to create a Measure dimension, which can help to answer questions, such as:

- Is the contribution margin for a particular make of car positive or negative?
- How much has the company spent on sales commission?

The Measure dimension makes it possible to assign different information to the other dimensions, such as the sales commission when a particular vehicle is sold, the overall contribution margin for a vehicle type, and the revenue earned by each branch from additional packages.

Demo: Create a Measure Dimension

Your instructor will demonstrate how to create a Measure dimension.

_____

_____

_____

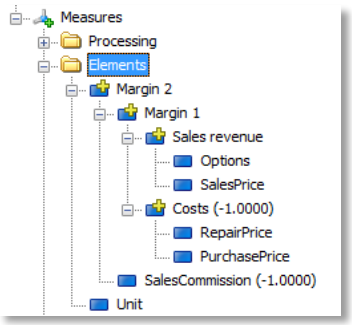
_____

Exercise 8.2: Create a Measure Dimension

In this exercise, you will create a Measure dimension.

Exercise Steps

1. Create a **Measure** dimension following the steps in Exercise 8.1.
2. Create elements below the **Measure** dimension using the following structure:



3. Drag the **Measure** dimension, drop it on the **Create Dimensions** job, and execute the job.

Lesson 9: Creating a Cube and Importing Data

Estimated Time

45 minutes

Learning Objectives

After completing this lesson, you will be able to:

- Create a cube.
- Import data from a relational table into an OLAP database.

Topics

- Creating a Cube
- Importing Data

Creating a Cube

Once you have created all of the required dimensions, you can create a cube. You will not be able to import data until this cube has been created with the appropriate dimensions. The values will be provided in the form of a contribution margin calculation and because CW Cars is an international company, the source data will be in different currencies.

Demo: Create a Sales Cube

Your instructor will demonstrate how to create a Sales cube.

_____

_____

_____

_____

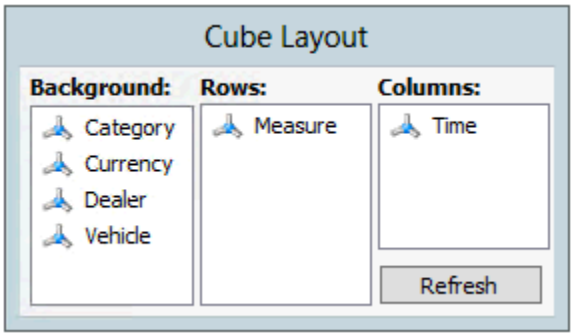
Exercise 9.1: Create a Sales Cube

In this exercise, you will create a Sales cube.

Exercise Steps

1. Right-click the **Cubes** folder below the **OLAP1** multidimensional database and select **New Cube**. The **New Cube** wizard displays.
2. Type *Sales* in the **Name** field.
3. Select the **Build new cube on every new import** check box.
4. Click **Next**.
5. Add each dimension to the cube by selecting the dimension in the **All dimensions** list and clicking the right arrow button to move the dimension to the **Used in new cube** list. The dimensions should be added in the following order:
   - **Category**
   - **Currency**
   - **Dealer**
   - **Vehicle**
   - **Time**
   - **Measure**
6. Click **Next**.
7. Type in the **Extended Properties** box: *<Alea:MeasureDimension Name="Measure" ExplicitlyDefined="true"/>.*
8. Click **Next**.

9. Verify the **Use Unknown Element** check box is cleared. If errors exist in the source data, this setting can be adjusted at a later time.
10. Click **Next**.
11. Click **Finish**.
12. Create a new job in the **Automatic Jobs** folder entitled *Create Cube*.
13. Drag the **Sales** cube, drop it on to the **Create Cube** job, and execute the job.
14. Close the **'OLAP1: Create cube: Sales'** window.
15. Right-click the **OLAP1: Create cube: Sales'** Sales cube and select **Browse…** to check the cube. The browser tool allows you to check the cube definition, the data, and the structures. **Note:** If a dialog box opens informing you that ImportMaster is unable to browse a cube with an empty dimension, click **OK**. The **Cube Layout** should look like this:



Importing Data

Importing data can be affected by the technical peculiarities of OLAP technology. A value cannot be imported into the multidimensional database unless the correct coordinates are specified. For each dimension, you must specify an element to which the value is assigned in this dimension.

An import is only possible to a combination of N elements: If a value is assigned to a C element in a dimension, this leads to an error when the value is written. This is because there is no information indicating how the value is distributed to the child elements.

Prior to importing data, you should check the data source and identify all data that you would like to import. You should also determine the source and destination objects that will be required to complete the data import.

The following are additional specifications for importing data into the Sales cube. Data will be imported into the Sales cube in two steps:

- First, the data available in the EUR currency will be imported. No currency conversion is required for the data at this point.
- The data in other currencies will then be added to the cube in the second step.

Demo: Import Data

Your instructor will demonstrate how to import data into the Sales cube.

_____

_____

_____

_____

Exercise 9.2: Import Data

In this exercise, you will import data into the Sales cube.

Exercise Steps

1. Create a new mapping in the **Mappings** folder of the **Sales** cube entitled *Sales cube – import data* following the steps in Part 2 of Exercise 6.1.
2. Open the **Sales cube – import data** mapping.
3. Create a result set object that contains the following query, reading all the relevant data from the dbo.Sales, dbo.Purchase and dbo.Dealers tables following the steps in Part 3 of Exercise 6.1:

```
SELECT      dbo.Dealers.Currency, dbo.Purchase.DealerId, dbo.Sales.DateOfSales,
dbo.Purchase.PurchasePrice,

        dbo.Purchase.RepairPrice, dbo.Sales.Options, dbo.Sales.SalesCommission,
dbo.Sales.SalesPrice,

        1 AS Unit
```

FROM dbo.Sales INNER JOIN

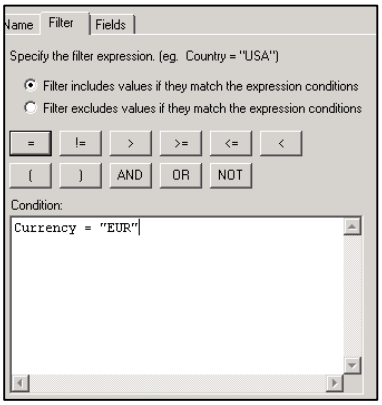
dbo.Purchase ON dbo.Sales.VehicleId = dbo.Purchase.VehicleId INNER JOIN

dbo.Dealers ON dbo.Purchase.DealerId = dbo.Dealers.Id

4. Select **Mappings > Destination Objects > Cube** from the menu bar. The **Name** screen of the **Cube** wizard opens.
5. Click **Next** to accept the default name of Cube. The **Cube** screen displays.
6. Type *OLAP1* in the **Alias** field.
7. Click the **Browse (…)** button next to the **Cube** field. The **Select Cube** dialog box opens.
8. Select the **Sales** cube.
9. Click **OK** to return to the Cube screen.
10. Select the **Add values to existing values** option.
11. Click **Next**. The **Dimensions** screen of the **Cube** wizard displays. From here, you can specify the fixed elements. For this data import, all dimension elements will be determined by the results in the result set.
12. Click **Finish** to complete the definition of the cube destination object.

If a value is assigned to a dimension element that does not exist in the dimension while data is being imported, the value can be trapped using the unknown element. When this occurs, the OnUnknownElement field will contain a message which can be further processed in a logging object.

13. Select **Mapping > Worker Objects > Filter** from the menu bar. A **Filter** object is created. This filter will be used to check the contents of the Currency field.
14. Drag the **Currency** field and drop it on the **Filter** object. A dialog box opens, asking how to insert the output field.
15. Verify the **Insert selected output** field is selected.
16. Click **OK**.
17. Right-click the **Filter** field and select **Properties**. The **Properties: Filter** dialog box opens.
18. Verify the **Filter includes values if they match the expression conditions** option is selected.
19. Click the **Insert Field** button to insert the Currency field in the Condition area.
20. Click the **=** button.
21. Type *"EUR"* after the = symbol. The filter expression should display as follows:



You can click the **Name** tab and specify a name for the Filter object to help differentiate it from other objects. This name will also be used in the Auto-Watch pane during debugging.

22. Click **OK** to close the Properties: Filter dialog box.
23. Connect the **Currency** field of the **Filter** object to the **Currency** field of the **Cube** object.
24. Select **Mapping > Worker Objects > Date Format** from the menu bar to create an object to format the date. A **Name** dialog box opens.
25. Click **Next**.
26. Select **MM/DD/YYYY** from the **Input format** drop-down menu.
27. Select **<mt>/<DD>/<YYYY>** from the **Output** format drop-down menu.
28. Verify the **Invalid input causes runtime error** option is selected.
29. Click **Next**. The **Months** dialog box displays allowing you to edit the names of the months.
30. Click **Finish**.
31. Drag the **DateOfSales** field from the result set object to the **Input** field of the **Date** object and then from the **Result** field of the **Date** object to the **Time** field of the **Cube** object.
32. Drag the **Type** field from the result set to the **Type** field of the **Cube** object. The element information for the **Type** dimension is already available in the correct form in the **Type** field of the result set.
33. Drag the **DealerId** field from the result set object to the **Dealer** field of the **Cube** object.

    A further mapping object is required to assign the Cars to Category and Model. The model and the category can be read from the dbo.Vehicles vehicle master data table by means of a dynamic lookup in the basis of the vehicle identification number (Ident) without the need to expand the source object result set.

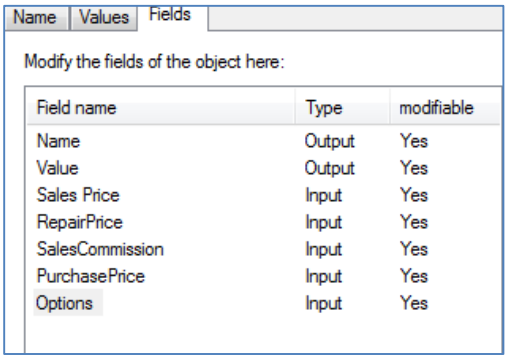
    In this scenario, it is theoretically possible for the table to the added to the query of the result set object without difficulty. It would even be possible to ensure that all the transaction data is read even if information is missing from the Vehicle master data. However, in practice, it is not always possible to link the various relational data together. The dynamic lookup object allows you to read additional relational data depending on the information read from the result set object.

34. Right click on any part of the screen and select the **Dynamic Object** option to create a dynamic lookup object.
35. Drag the **"Id"** field of the result set object onto the new object.
36. Using the context menu of the object, add two output fields: Cars and Category.
37. Right click on **Dynamic Lookup** and select **Properties**. Open the Properties of the dynamic lookup object to define the relational data to the object.
38. Select the **Source** tab.
39. Select **Lookup is filled by Result Set**.
40. Click **Define**.
41. Click **Define** to identify a result set that queries the columns Id, Model and Condition from the dbo.Vehicles table.

```
SELECT          dbo.Vehicles.Id, dbo.Vehicles.Model, dbo.Vehicles.Condition
```

```
FROM  dbo.Vehicles
```

42. Work through the dialog boxes of the result set and then return to the Properties of the dynamic lookup by clicking Finish.
43. Select the option "**If input is invalid, pass on these values**."
44. Type "*No Vehicle Data*" in the **Cars** and **Category** columns.
45. Link the **"Id"** field to the source columns in the Column Tab.
46. Select the **"Id"** column in the **Select Column** dialog box, and confirm your choice by clicking **OK**.
47. Repeat steps 42 and 43 for the **Vehicles** and **Category** fields.
48. Click **OK** to close the Properties of the dynamic lookup.
49. Drag and drop the **Vehicle** and **Category** fields of the dynamic lookup object to the same fields in the **Cube** object.
50. Right click to create a normalizer object. This worker object is required for the fields of the values to be imported for SalesPrice (selling price) and Options (additional packages). All of these fields contain value types or measures that must be addressed individually in the Measure dimension. A field in the destination object can only have a single join. The object required here is known as a normalizer. The normalizer object carries out iterations across a number of input fields and outputs the results one after another in two output fields. The object is used whenever parallel data has to be converted into sequential data.
51. Drag the following fields from the **result set** object to the **normalizer** object:
    - **SalesPrice**
    - **Options**
    - **PurchasePrice**
    - **RepairPrice**
    - **SalesCommission**
52. Right-click the **Normalizer** object and select **Properties**. The **Properties** dialog box opens.
    Note: The field names on the Fields tab have to be changed. The normalize passes the defined field name to the "Name" field and the corresponding value of the field to the "Value" column. The field names must correspond to the destination elements of the "Measure" dimension.
53. Edit the field names as follows:



54. Click **OK** to apply the changes.
55. Add the other joins by dragging and dropping the elements: The "Name" field of the normalizer must be associated with the "Measure" field of the cube object, and the "Value" field of the normalizer with the "Value" field of the cube object.

To improve the clarity of the joins, you can drag the fields within an object or in the Fields tab in the object Properties to change their sequence.

56. Save the mapping and add it to the "Create cube" job.
57. Right click on the **job name** to start the job
58. Select **Execute Immediately** to check the messages.

Completing the Currency Conversion

So far, only the data in the EUR currency has been imported. In order to complete the data in the cube, the data available in other currencies must now be imported.

This data should be imported to the appropriate "local currency" element in the original currency and to the "EUR" element after conversion to euros.

The CW Cars database contains a view named "ExchRate" that provides an exchange rate for each currency for each month. Assess what processing steps are required in the "Sales cube – import data" mapping in order to implement currency conversion and what new objects are required.

Demo: Complete the Currency Conversion

Your instructor will demonstrate how to complete the currency conversion.

_____

_____

_____

_____

Exercise 9.3: Complete the Currency Conversion

In this exercise, you will complete the currency conversion.

Exercise Steps

1. Open the **Sales cube – import data** mapping.
2. Copy the **Filter** object, and drag and drop the existing result set object to the new filter.
3. Select the **Filter excludes values if they match the expression conditions** option in the **Properties: Filter** dialog box.
4. Convert the date of the relational records to the format used in the exchange rate table.
5. Connect the **"DateOfSales"** field of the result set object to the **"Input"** field of the **Date Format** object **"Extract year"** that you have just created.

6. Change the name of the **"Result**" field to **"Year"** by selecting the **Properties** of the date format object and changing the name in the **Fields** tab.
7. Copy the data format object and modify it so that only the month is extracted from the date. Connect the "DateOfSales" field to the input field of the new data format object "Extract month". Here also, change the name of the "Result" field, in this case to "Month".
8. Now create a new dynamic lookup object with the name "Exchange rate" to read the exchange rates depending on the date and the currency.
9. Drag the "Year" and "Month" fields of the two new date format objects and the "Currency" field of the new filter into the dynamic lookup object, which is still empty.
10. Create a new output field "ExchRate".
11. Now, in the Source tab of the Properties of the dynamic lookup, create a result set which queries the "Year", "Month", "ISO" and "ExchRate" columns of the dbo.View_ExchRate view.
12. In the Columns tab, map the fields of the object to the columns of the result set.
13. Before objects are created that convert the values of the individual measures on the basis of the exchange rate that has been read, copy the cube object and give the new cube object the name "LC in EUR SalesCube".
14. Connect the dimension information of the new "data path" to the new cube object.
15. Because we are converting the values to euros, they should always be imported to the "EUR" element of the "Currency" dimension. Open the Properties of the new cube object. In the Dimensions tab, you can specify the "EUR" element as the "fixed" element for the "Currency" dimension.

Objects that have been fully defined and no longer need to be edited can be minimized by clicking the arrow in the top-right corner.

16. Create a numeric expression object for converting the monetary values.
17. Drag the "ExchRate" field of the dynamic lookup to the empty object.
18. Add a new input field with the name "Value".
19. In the Expression tab of the Properties of the numeric expression object, enter the formula and confirm your settings by clicking OK.
20. Connect the "Value" field of the normalizer object to the "Value" field of the numeric expression object. Connect the "ExchRate" field of the dynamic lookup object to the corresponding field in the numeric expression object.
21. Connect the "Name" field of the normalizer object to the "Measure" field of the new cube object.
22. Save the mapping, and start the "Create cube" job.
23. Now check the data in the cube.
24. Now also add a third cube object to the mapping: The values for the local currencies should also be imported to the appropriate local currency element (USD, AUD, CAD etc.). Do this by creating an additional cube object with the name "LC CubeSales".
25. Connect the dimension fields of the "LC CubeSales" object to the dimension fields of the second cube object and the fields for the monetary values of the result set to the measures of the "LC CubeSales" object.
26. Start the **Create cube** job again and check the cube.

Course Summary

Estimated Time

30 minutes

Course Objectives

Now that you have completed this course, you should be able to:

- Describe the purpose of using ImportMaster.
- Create and save an import definition.
- Identify the main components of the ImportMaster user interface.
- Protect an import definition.
- Create a relational database connection and use the SQL Query Builder to check the database.
- Identify the global folders that apply to all multidimensional databases.
- Create a multidimensional database connection.
- Create dimensions using the New Dimension wizard.
- Create and execute jobs.
- Manually create static elements in a new dimension.
- Explain the process and characteristics of mapping.
- Identify the various source and destination objects used in mapping.
- Use the Debug feature in a mapping.
- Identify the main programming features of CWScript.
- Create a dimension using CWScript and debug a script.
- Create a cube.
- Import data from a relational table into an OLAP database.

Topics

- Course Review

Course Review

Which of the following is an extract, transform, and load (ETL) layer that facilitates integration with both Infor and non-Infor source systems?

a) Application Studio
b) ImportMaster
c) DeltaMiner
d) Consolidation

Infor BI ImportMaster acts as the interface between existing relational database systems and individual OLAP databases. It is an automated solution for importing data from a variety of Infor and non-Infor source systems to multidimensional OLAP databases.

a) True
b) False

Import definition files in Infor BI ImportMaster are saved with what extension?

a) .imd
b) .imp.
c) .def
d) .idf

In Infor BI ImportMaster, an open import definition appears in a window that is divided into panes. Connections to source databases that have been defined for the import definition are displayed in what pane?

a) The Relational Database pane
b) The Multidimensional Database pane
c) The OLAP Database pane
d) The ODBC Database pane

Which of the following statements about relational databases in Infor BI ImportMaster are true? Select all that apply.

a) The New Relational Database Connection wizard allows you to define the source database.
b) The relational database indicates where data will be imported from.
c) Infor BI ImportMaster supports all ODBC and OLE DB-capable databases and the integration of flat files.
d) The structure of a relational database is read in once and will be stored internally within Infor BI ImportMaster.

Which SQL Query Builder pane includes a graphical display of the tables used in a query and allows you to select columns and join tables?

a) Diagram
b) Grid
c) Statement
d) Result

Which of the following folders display below the Multidimensional Databases icon in ImportMaster and apply to an entire import definition? Select all that apply.

a) Jobs
b) General Scripts
c) General Mappings
d) Mail configurations

When you create a new multidimensional database in ImportMaster, folders for which of the following appear below the database icon? Select all that apply.

a) Processing
b) Dimensions
c) Cubes
d) Scripts
e) Mappings

_____ dimensions are dimensions that are not created from a source. These dimensions are typically time dimensions or value-type dimensions.

a) Dynamic
b) Static
c) Relational
d) Multidimensional

In Infor BI ImportMaster, which of the following screens only display in the New Dimension wizard when creating a Time Dimension? Select all that apply.

a) Time Period for Dimension
b) Use unknown element
c) Time Hierarchy
d) Delete all dimension elements before import

Which of the following statements about relational databases in Infor BI ImportMaster are true? Select all that apply.

a) All jobs are displayed in a Jobs folder below Multidimensional Databases.
b) A job is used to create and populate cubes and dimensions.
c) Jobs can be programmed to run at periodic intervals.
d) Jobs can be started manually.

Which of the following elements can be manually created for a dimension by right-clicking the Elements folder below a dimension? Select all that apply.

a) N Element
b) C Element
c) S Element
d) R Element

Mapping makes it possible to carry out an import without using scripts.

a) True
b) False

Which of the following are source objects used when mapping in Infor BI ImportMaster? Select all that apply.

a) Dimensions
b) Cubes
c) Logging
d) Result Sets

Mapping always involves one or more source objects and a single destination object.

a) True
b) False

When using CWScript with ImportMaster, comments always begin with _____.

a) Square brackets([ ])
b) A semicolon
c) The number sign (#)
d) The dollar sign ($)

When using CWScript with ImportMaster, commands that return a value must be placed in square brackets ([ ]) to ensure that substitution works correctly and the return value can be obtained. If the return value for an individual command is not clearly described, as a general rule, every command that is executed correctly returns what value?

a) 0
b) 1
c) 2
d) 10

A missing reference is a typical error that occurs when importing data. Which of the following provides an automated method of error handling when there are missing references?

a) Creating a script
b) Resetting the data area in a cube
c) Using an unknown element
d) Protecting an import definition

Jobs in ImportMaster should be scheduled and run automatically. Import definitions can be scheduled to start automatically by which of the following? Select all that apply.

a) The ImportMaster Service Manager
b) The ImportMaster Remote Console
c) The IMRUN runtime module
d) The SQL Query Builder