



ESSBASE[®] INTEGRATION SERVICES

RELEASE 7.1.2

DATA PREPARATION GUIDE



Copyright 1998–2004 Hyperion Solutions Corporation. All rights reserved.

May be protected by Hyperion Patents, including U.S. 5,359,724 and U.S. 6,317,750

“Hyperion,” the Hyperion “H” logo and Hyperion’s product names are trademarks of Hyperion. References to other companies and their products use trademarks owned by the respective companies and are for reference purpose only.

No portion of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser’s personal use, without the express written permission of Hyperion.

The information contained in this manual is subject to change without notice. Hyperion shall not be liable for errors contained herein or consequential damages in connection with the furnishing, performance, or use of this material.

This software described in this manual is licensed exclusively subject to the conditions set forth in the Hyperion license agreement. Please read and agree to all terms before using this software.

GOVERNMENT RIGHTS LEGEND: Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the applicable Hyperion license agreement and as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (Oct 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14, as applicable.

Hyperion Solutions Corporation
1344 Crossman Avenue
Sunnyvale, California 94089

Printed in the U.S.A.

Contents

Preface	v
Purpose	v
Audience	v
Document Structure	vi
Sample Applications	vi
Where to Find Hyperion Documentation	vi
Conventions	viii
Additional Support	ix
Education Services	ix
Consulting Services	ix
Technical Support	ix
Documentation Feedback	ix
CHAPTER 1 About OLAP and Essbase Integration Services	11
About OLAP	12
About Multidimensional Databases	12
About SAP BW Data Warehouses	13
User Interactions with Data	14
Sources of Data	15
About Essbase Integration Services	16
Components of Essbase Integration Services	17
Essbase Integration Services Console	17
Essbase Integration Server	17
Workflow for Using Essbase Integration Services	18
About OLAP Models	19
About Using OLAP Models	19
About the Fact Table	19
About Dimension Tables	20
About Metaoutlines	20
Components of Metaoutlines	21
Using Hierarchies	22
Types of Hierarchies	23
Deploying Hierarchies	24

CHAPTER 2 Preparing Data Sources	27
Defining User Needs	28
Defining Data Sources	29
Consolidating Data into a Single Data Source	29
Deciding Whether to Create a Staging Area	30
Cleansing Data	31
Creating Views, Tables, and User-Defined Tables	31
Deciding Whether to Create a View or a New Table	32
Deciding Whether to Create a User-Defined Table	32
Fact Tables in SAP BW	32
Building Analytic Services Hierarchies from Recursive Tables	33
Removing Unions	35
Transposing Columns and Rows	35
Denormalizing Source Data Tables	36
Adding Columns to Tables	37
Setting Up Columns to Support Member and Measure Properties	38
Member and Measure Properties	38
Joining Tables	38
Optimizing Tables for Use with SAP BW	38
Indexes for Small Tables	38
Push Down Operations	39
Creating Indexes	39
Transforming Data	40
Deciding Which Tables Are Available to OLAP Model Creators	41
Selecting Tables for the Fact Table	42
Selecting Tables for Dimensions	43
Creating Aliases for Dimension and Member Names	44
Creating Time and Accounts Dimensions	45
Preparing Data for Time Analysis	45
Associating Time Data with Measure Data	45
Working with Summary Data	45
Accounts Dimensions and SAP BW	48
Time Dimensions and SAP BW	48
Slowly Changing Dimensions in SAP BW	48
Handling Attribute-Enabled Columns in SAP BW	49
Accessing Tables in the OLAP Metadata Catalog	52
Mapping to DB2 Cube Views	52
Using Text Files as Data Sources	53
Index	55

Preface

Purpose

This guide outlines steps for preparing data from a data source for a smooth transition to Essbase® Analytic Services databases. In addition, this guide provides an introduction to OLAP concepts and familiarizes you with Essbase Integration Services components. Finally, it presents an overview of the information that you need for creating and working with OLAP models and metaoutlines.

Audience

This guide is for database administrators who are responsible for installing, implementing, and deploying Integration Services.

To use the information in this book, you need the following skills:

- Knowledge of where the data for your business resides; for example, in a relational database or a data warehouse.
- Knowledge of how to create and maintain Open Database Connectivity (ODBC) data sources
- Knowledge of the data requirements for your business, so that you can apply the Integration Services product family to your specific application
- A fundamental understanding of Microsoft Windows and basic Microsoft Windows terminology, such as dialog box, list box, and button
- Experience with the setup and operation of Essbase Analytic Server
- A basic understanding of multidimensional concepts

Document Structure

This document contains the following information:

- [Chapter 1, “About OLAP and Essbase Integration Services,”](#) introduces basic OLAP and multidimensional concepts and describes how to use the Essbase Integration Services Console to create OLAP models and metaoutlines.
- [Chapter 2, “Preparing Data Sources,”](#) describes how to prepare relational data for transfer to an Analytic Services database.
- [Index](#), lists Integration Services terms and their page numbers. In the PDF version of this guide, select an index entry to view the relevant page.

Sample Applications

The examples used in this book are based on a derivative of the two sample databases provided with Integration Services. The databases are called TBC (external data source) and TBC_MD (OLAP Metadata Catalog). The TBC_MD OLAP Metadata Catalog database contains a sample OLAP model and a sample metaoutline. The OLAP model is called TBC Model, and the metaoutline is called TBC Metaoutline.

Note: The capitalization of column and table names in the sample relational database applications depends on the relational database management system (RDBMS) that you use.

The system administrator who installs Integration Services is responsible for making the sample databases, the OLAP model, and the metaoutline available to end users. It is the Integration Services system administrator’s responsibility to respond if any of the following problems occur when users launch Integration Services Console:

- Users cannot find the sample databases, the OLAP model, or the metaoutline.
- Users do not have adequate access to the sample databases, the OLAP model, or the metaoutline.
- Users do not see any data in the sample databases.

Where to Find Hyperion Documentation

All Integration Services documentation is accessible from the following locations:

- *Essbase Integration Services Information Map* provides access to all Integration Services documentation in PDF, Windows Help, and HTML formats. The Information Map is located at

`ISHOME/essintegration/docs/eis_info_map.htm`

- Online help is accessible from Integration Services Console. Start the product and click the Help button on dialog boxes or use the Help menu command. Online help includes procedural and conceptual information as well as dialog box help and a glossary.

- Intelligent Help is displayed in a dockable, relocatable window within Integration Services Console and provides procedural information along with links to automatic-detection options and frequently used functions.
- The Hyperion Solutions Web site is located at <http://www.hyperion.com>.
- The Hyperion Download Center can be accessed from <http://hyperion.subscribenet.com> or from <http://www.hyperion.com>.

➤ To access documentation through the Hyperion Solutions Web site:

- 1 Log on to <http://www.hyperion.com>.
- 2 Select the **Support** link and type your username and password to log on.

Note: New users must register to receive a username and password.

3 Click the **Hyperion Download Center** link and follow the on-screen instructions.

➤ To access documentation from the Hyperion Download Center:

- 1 Log on to <http://hyperion.subscribenet.com>.
- 2 In the **Login ID** and **Password** text boxes, enter your assigned login ID name and password.
- 3 In the **Language** list box, select the appropriate language and click **Login**.
- 4 If you are a member on multiple Hyperion Download Center accounts, select the account that you want to use for the current session.
- 5 Perform one of the following actions:
 - To access documentation online, from the **Product List**, select the appropriate product and follow the on-screen instructions.
 - To order printed documentation, from the **Information** section in the left frame, select **Order Printed Documentation**, then follow the on-screen instructions.


➤ To order printed documentation if you do not have access to the Hyperion Download Center:

- In the United States, call Hyperion Solutions Customer Support at 877-901-4975.
- From outside the United States, including Canada, call Hyperion Solutions Customer Support at 203-703-3600. Clients who are not serviced by support from North America should call their local support centers.

Conventions

The following table shows the conventions used in this document:

Table i Conventions Used in This Document

Item	Meaning
	Arrows indicate the beginning of a procedure consisting of sequential steps or one-step procedures.
Brackets []	In examples, brackets indicate that the enclosed elements are optional.
Bold	Bold in procedural steps highlights user interface elements on which the user must perform actions.
CAPITAL LETTERS	Capital letters denote commands and various IDs. (Example: CLEARBOCK command)
Ctrl + 0	Keystroke combinations shown with the plus SIGN (+) indicate that you should press the first key and hold it while you press the next key. Do not type the plus sign. For consecutive keystroke combinations, a comma indicates that you press the combinations consecutively.
<i>Example text</i>	Courier font indicates that the example text is code or syntax.
<i>Courier italics</i>	Courier italic text indicates a variable field in command syntax. Substitute a value in place of the variable shown in Courier italics.
<i>ARBORPATH</i>	When you see the environment variable <i>ARBORPATH</i> in italics, substitute the value of <i>ARBORPATH</i> from your site.
<i>n, x</i>	Italic <i>n</i> stands for a variable number; italic <i>x</i> can stand for a variable number or a letter. These variables are sometimes found in formulas.
Ellipses (...)	Ellipsis points indicate that text has been omitted from an example.
Mouse orientation	This document provides examples and procedures using a right-handed mouse. If you use a left-handed mouse, adjust the procedures accordingly.
Menu options	Options in menus are shown in the following format. Substitute the appropriate option names in the placeholders, as indicated. Menu name > Menu command > Extended menu command For example: 1. Select File > Desktop > Accounts .

Additional Support

In addition to providing documentation and online help, Hyperion offers the following product information and support. For details on education, consulting, or support options, click the Services link on the Hyperion Web site at <http://www.hyperion.com>.

Education Services

Hyperion offers instructor-led training, custom training, and eTraining covering all Hyperion applications and technologies. Training is geared to administrators, end users, and information systems (IS) professionals.

Consulting Services

Experienced Hyperion consultants and partners implement software solutions tailored to clients' particular reporting, analysis, modeling, and planning requirements. Hyperion also offers specialized consulting packages, technical assessments, and integration solutions.

Technical Support

Hyperion provides enhanced electronic-based and telephone support to clients to resolve product issues quickly and accurately. This support is available for all Hyperion products at no additional cost to clients with current maintenance agreements.

Documentation Feedback

Hyperion strives to provide complete and accurate documentation. We value your opinions on this documentation and want to hear from you. Send us your comments by clicking the link for the Documentation Survey, which is located on the Information Map for your product.

1

About OLAP and Essbase Integration Services

Essbase Integration Services provides a suite of graphical tools to create OLAP models and metaoutlines and populate Essbase Analytic Services databases. You use your data source to define a logical model that represents the data in an online analytical processing (OLAP) context. You then use the OLAP model to create a metaoutline that serves as a template for an Analytic Services database outline.

This chapter explains basic OLAP concepts, provides an overview of Integration Services, describes OLAP models and metaoutlines, and defines the workflow for creating OLAP models and metaoutlines.

In This Chapter	About OLAP	12
	About Multidimensional Databases	12
	About SAP BW Data Warehouses	13
	User Interactions with Data	14
	Sources of Data	15
	About Essbase Integration Services	16
	About OLAP Models	19
	About Metaoutlines	20
	Using Hierarchies	22

About OLAP

OLAP is a decision-support computing environment for business managers who need to analyze consolidated enterprise data in real time. OLAP enables users to answer complex “what if” questions and to create sales and marketing scenarios to test budgeting, sales promotion, and sales planning strategies.

Analytic Services supports the OLAP approach, making possible a multidimensional, multiuser database that enterprise users can access using standard retrieval tools, such as spreadsheets.

Analytic Server supports multiple views of data sets for users who need to analyze relationships between data categories. For example, a decision-support user might ask the following questions:

- How did Product A sell last month? How does this figure compare to sales in the same month over the last five years? How did Product A sell by branch, by region, and by territory? How will Product A sell next month, next quarter, and next year?
- Did Product A sell better in particular regions? Do regional trends exist?
- Did customers return Product A last year? Were returns due to product defects? Did the company manufacture defective products in a specific plant?
- Did commissions and pricing affect how salespeople sold Product A? Did particular salespeople do a better job of selling the product?

You can use Integration Services to create an Analytic Services database that enables users to answer these types of questions quickly and easily. You can use Integration Services Console to create a logical data model that represents the data source database.

About Multidimensional Databases

A *multidimensional database* (MDDB) stores consolidated data at the intersections of its members and dimensions. For example, if a company sells a total of 20 units of all products in the East region in the first quarter, Analytic Services stores 20 at the intersection of Product, East, Quarter1, and Unit Sales.

In a multidimensional database, a *dimension* is a data category that represents a core component of a business plan, and it often relates to a business function. Product, Region, and Year are typical dimensions. In most databases, dimensions are static, rarely changing over the life of the application.

In a multidimensional database, a *member* is an individual component of a dimension. For example, Product A, Product B, and Product C might be members of the Product dimension. Each member has a unique name. A dimension can contain a large number of members. In some dimensions, members change frequently over the life of the application.

Simultaneously, members can be parents of some members and children of other members. The Analytic Services outline indents members below one another to indicate a consolidation relationship. For example, sales totals for the Products dimension might be totalled by product description, broken down by product code, and further broken down by product ID.

About SAP BW Data Warehouses

SAP Business Information Warehouses (BW) store data in nonrelational, process-oriented data structures.

Note: Support for SAP BW is available through Hyperion Data Integration Connector, a separately purchased and licensed option.

SAP BW data warehouses use different terminology for components that are analogous to multidimensional components. The following list denotes differences in terminology between SAP BW data warehouses and multidimensional databases:

- In SAP BW, a dimension is referred to as a *characteristic*.
- In SAP BW, the term *dimension* is used specifically to denote a group of related characteristics; for example, a time dimension could refer to the characteristics, Calendar Year and Fiscal Year.
- In SAP BW, a member is referred to as a *characteristic value*.
- In SAP BW, a measure is referred to as a *key figure*. It is a quantifiable value; for example, a key figure could be Gross Sales.

Table 2, below, outlines how SAP BW data source components map to relational and XML targets.

Table 2 SAP BW to Relational/XML Model Mapping

Source (BW/ODBO)	Target (Relational Model)	Target (XML Model)
Dimension Name		ModelDim element, name attribute
Time dimensions	Time table	ModelDim element, modelDim attribute
	Fact table	ModelDim element, modelDim attribute ModelLogicalJoin element, view1Name attribute
Level Name		ModelView element, name attribute ModelLogicalJoin element, viewiName attribute
Level Name	Dimension table	
Hierarchy Name	Parent/child table (for alternate hierarchies)	ModelHierarchy element, name attribute ModelPhysicalJoin element, table1Name attribute, table2Name attribute
Member Unique Name		ModelHierarchyMember element, viewMemberName attribute
Member Unique Name	Column in fact, dimension, parent/child tables	ModelViewMember element, name attribute ModelLogicalJoin element, memberiName attributes ModelPhysicalJoin element, column1Name attribute

Table 2 SAP BW to Relational/XML Model Mapping (Continued)

Source (BW/ODBO)	Target (Relational Model)	Target (XML Model)
Member Name	Column in dimension, parent/child tables (used to join parent/child table to dimension table)	ModelViewMember element, name attribute ModelLogicalJoin element, memberName attributes
Member Caption	Column in dimension, parent/child tables	ModelViewMember element, name attribute
Parent Unique Name	Column in parent/child tables	ModelViewMember element, name attribute ModelPhysicalJoin element, column2Name attribute
Property Name (dimension properties)	Columns in dimension, parent/child tables	ModelViewMember element, name attribute
Property Name (dimension properties)	Columns in dimension, parent/child tables	ModelViewMember element, drill-through type attribute
Measure Unique Name	Column in fact table	ModelViewMember element, name attribute
Measure Aggregator		ModelViewMember element, aggregateType attribute

User Interactions with Data

Analytic Server consolidates and calculates data to provide different views of the data. Using a multidimensional database, here are some of the tasks that users can perform:

Consolidate (aggregate or roll up) data

In a multidimensional database, the *consolidation* process computes the data relationships for all parent and child combinations within a dimension. For example, the consolidation for the Year dimension is as follows:

$$\text{Year} = \text{Quarter1} + \text{Quarter2} + \text{Quarter3} + \text{Quarter4}$$

A consolidation is typically additive, but it can be any type of calculation.

Create sophisticated “what if” scenarios

Assume that you set a sales goal of ten percent growth for the upcoming year on all product lines. With Analytic Services, you can compare sales forecasts estimated by planners with actual sales data (retrieved from the *online transaction processing* [OLTP] database) to see how close you are to achieving your sales goals. If actual sales run lower than projected sales for a given period, salespeople can access the forecast data, input new product sales scenarios (for example, What if I sell 2000 widgets to our biggest corporate customer?), update the forecast data, and provide revised figures to headquarters.

Input strategic planning assumptions

Assume that your company is planning for 50 percent growth over the next three years. You have a fairly good idea of how many new products you need, but how do you know how many new engineers, salespeople, and support personnel you can afford to add while still optimizing profits and gross margin?

With Analytic Services, you can input projected sales and expenses for each product and calculate downward to determine the cost of goods sold. Thus, you obtain a realistic picture of the bottom line. If the picture does not look practical, you can create different scenarios with different mixes of products, people, and expenses until you produce the profit picture that you require.

Conduct spreadsheet operations

To *drill down* or *drill up* on data retrieves progressively more detailed or progressively more generalized data relative to a selected dimension. Drilling down on a multidimensional database dimension provides you with greater detail on the dimension. Drilling up provides you with a higher level (more summarized) view of the dimension.

For example, you can drill down on the Year dimension to view the values for each quarter. Then you can drill down on Quarter1 to view the values for each month of Quarter1. You can view Quarter1 sales for the Chicago office and then drill up to view sales totals for the entire Central region.

Pivoting data alters the data perspective. When Analytic Server retrieves a dimension, it displays a configuration of rows and columns. A user can pivot (rearrange) the data to obtain a different viewpoint.

For more information about spreadsheet operations, such as drill down and pivot, see the Essbase Spreadsheet Add-in or Essbase Spreadsheet Services documentation.

Sources of Data

The data in a multidimensional database can originate from a variety of sources, such as OLTP databases, data warehouses, text files, and spreadsheet files. OLTP databases contain a wealth of operational data, such as the following information:

- How many units of Product A are on hand?
- What is the current customer's name, address, and billing status?
- What is an employee's salary, job title, and address?

Data in a database must be up-to-date and easy to change, and must use as little space as possible. In relational databases, the data is stored in rows and columns; in a data warehouse such as an SAP BW, data is stored in QueryCubes and InfoCubes.

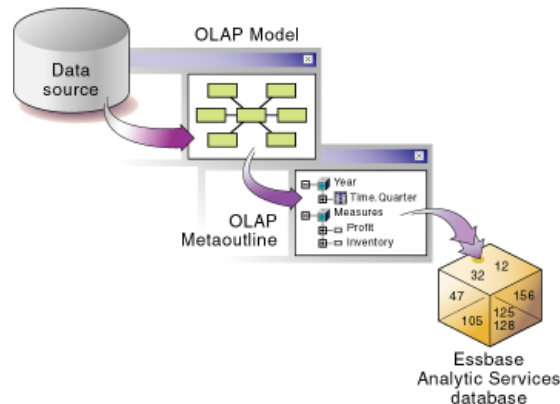
Integration Services enables you to access data for OLAP models or metaoutlines from one or more data sources that support SQL 89 or later. You can access data in the data sources using Open Database Connectivity (ODBC) such as the DataDirect Wire Protocol, or the client software provided by the RDBMS manufacturer.

About Essbase Integration Services

Essbase Integration Services quickly transfers data from data sources to an Analytic Services database. To perform the transfer, you must determine which data to transfer and consolidate the selected data into a form that is useful for decision-support users. Then you must identify the tables, rows, or columns that contain the data and determine how they map to the structure of the multidimensional database.

Figure 1 illustrates the Integration Services workflow:

Figure 1 Workflow for Creating an Analytic Services Database from a Data Source



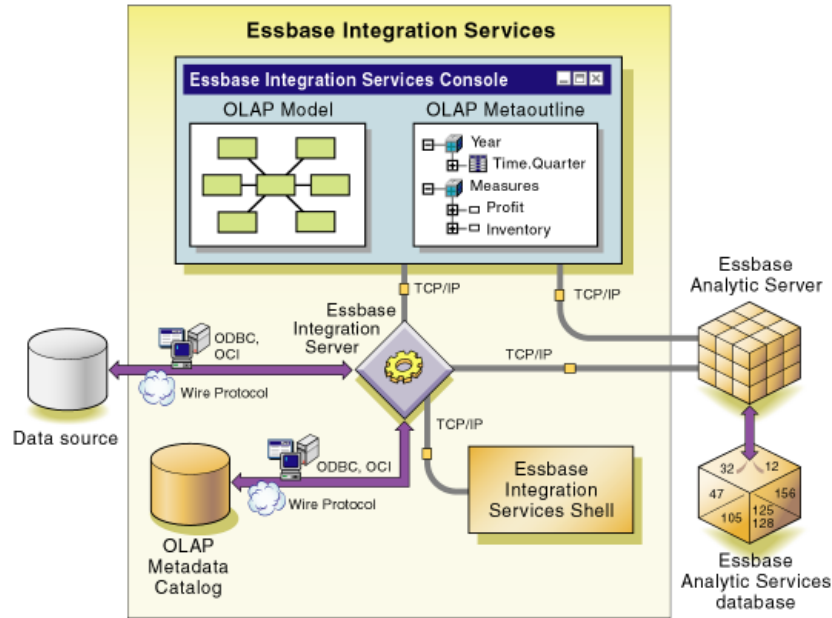
Use Essbase Integration Services Console to perform the following tasks:

- Use the tables, views, and columns in one or more data source databases to create an OLAP model. An *OLAP model* is a logical *star schema* consisting of a fact table that is surrounded by related dimension tables.
- Use the OLAP model to create a *metaoutline*, an outline template that contains the structure and the rules required to generate an Analytic Services outline.
- Use the metaoutline to create and populate an Analytic Services database.

Components of Essbase Integration Services

Essbase Integration Services, pictured in [Figure 2](#), consists of two major components: Integration Services Console and Integration Server.

Figure 2 Essbase Integration Services



Essbase Integration Services Console

You use Essbase Integration Services Console, a graphical user interface, to create OLAP models and metaoutlines and to populate Analytic Services databases.

To create an OLAP model or metaoutline, you must connect to an OLAP Metadata Catalog and the appropriate data sources. For more information on creating a model or metaoutline, refer to the Integration Services online help.

Essbase Integration Server

Essbase Integration Server is the primary component of Integration Services. Integration Server is software that uses the information stored in the OLAP Metadata Catalog to extract from data sources the dimension and member names needed to build an associated Analytic Services outline. If your data source is an SAP BW data warehouse, Integration Services extracts characteristic names and characteristic value names.

When the outline is complete, Integration Server extracts data from the data sources, performs the operations specified in the associated metaoutline, and loads the data into the Analytic Services database. For more information about Integration Server, see the *Essbase Integration Services System Administrator's Guide*.

Integration Server includes the following subcomponents, as illustrated in [Figure 2 on page 17](#):

- **OLAP Metadata Catalog:** *OLAP Metadata Catalog* is a structured query language (SQL) relational database that contains the following information:
 - Metadata describing the nature, source, location, and type of data to retrieve from the data sources
 - Metadata describing the information required to generate an Analytic Services outline
 - OLAP models and metaoutlines

You can create more than one OLAP Metadata Catalog to store OLAP models and metaoutlines. Using XML Import/Export, you can move OLAP models and metaoutlines to a different OLAP Metadata Catalog after you have created and saved them to a specific catalog.

The OLAP Metadata Catalog is a data source that is configured for Open Database Connectivity (ODBC). If you do not know how to create an ODBC data source, see the *Essbase Integration Services Installation Guide* or the ODBC user documentation.

- **Essbase Integration Services Shell:** Essbase Integration Services Shell is a command-line tool used to access Integration Server to load members and data to an Analytic Services database. For more information on using Essbase Integration Services Shell, see the *Essbase Integration Services System Administrator's Guide*.

Workflow for Using Essbase Integration Services

Figure 2 on page 17 provides an overview of the Integration Services components that you use to prepare the data for OLAP reporting through Analytic Server.

➤ To create an Analytic Services database from a data source:

1 Build an OLAP model that is based on the appropriate data sources.

Integration Server stores the OLAP model and the information necessary to retrieve the relevant tables in OLAP Metadata Catalog.

2 Create a metaoutline from the OLAP model.

Integration Server stores the metaoutline in OLAP Metadata Catalog.

3 Use the metaoutline to load members and data into the Analytic Services database.

After you have created the Analytic Services database, you can update it with new members and data as needed to satisfy your reporting requirements.

About OLAP Models

OLAP models are based on the idea that values in a data source can be categorized as either facts or dimensions of facts. Facts (or key figure data values in SAP BW) are the numeric, variable values in the database, such as sales figures and numbers of units sold. Associated with facts are related data values that provide additional information, such as store locations and product IDs of units sold. An OLAP model contains a fact table, one or more dimension tables, and one or more dimension branches. An OLAP model may also contain time and accounts dimensions.

Unlike other integration products, Integration Services creates an OLAP model that is a logical model, not a physical star schema. The OLAP model is a logical representation of the data values that you select from the appropriate data sources and that you want to report in Analytic Services.

The sample application provided with Integration Services includes a sample OLAP model named TBC Model.

About Using OLAP Models

Use an OLAP model to create one or more metaoutlines. A metaoutline contains the basic structure required to build an Analytic Services outline and to load data into Analytic Services. You can use one OLAP model as the basis for another OLAP model by saving the original OLAP model under a different name and editing it as needed to meet reporting requirements. You can create any number of OLAP models for use in building metaoutlines. Each metaoutline, however, is based on one, specific OLAP model.

OLAP models have the following features:

- They are reusable. You can use the same OLAP model as the basis for multiple metaoutlines.
- They provide a layer of abstraction that insulates the Analytic Services database outline from changes in the source database.
- They enable you to create hierarchies to structure and summarize data from a source database. You can use the hierarchies in multiple metaoutlines.

About the Fact Table

The *fact table* serves as a container for all numeric facts (for the measures data values that vary over time). In an SAP BW data warehouse, facts are key figure values. In the OLAP model included with the sample application, the fact table consists of the SALES relational table: a table that contains sales figures, cost of goods sold figures, opening and ending inventory quantities, and other data columns of variable measures.

About Dimension Tables

A *dimension table*, such as a Product dimension table, serves as a container for relational tables. Each dimension table contains data that is related to the facts in the fact table. For example, the Product dimension table in the sample OLAP model contains a relational table in which the PRODUCTID is related to the sales figures and inventory quantities in the SALES table.

When a dimension table joins to the fact table, the dimension table and all dimension tables joined to it form a dimension. A dimension in an OLAP model represents a dimension (or characteristic from a data warehouse) that you want to report in Analytic Services. For more information on Analytic Services dimensions, see [“About Multidimensional Databases” on page 12](#).

When you create a dimension in an OLAP model, the dimension becomes available for use in creating a dimension in an associated metaoutline. You can drag a predefined OLAP model dimension directly into the metaoutline to create a dimension. The newly created metaoutline dimension then becomes a dimension in the Analytic Services database that you create when you perform a member or data load.

If your data source is a relational database, when you build a metaoutline, you can create user-defined dimensions that do not exist in the associated OLAP model. Use this option when you need a dimension, such as Scenario, that does not exist in the data source. If you create a Scenario dimension, you can add both Actual and Budget members to track measures, such as actual and projected revenue and profit, that might not be stored in the data source.

About Metaoutlines

A metaoutline is a template containing the structure and rules for creating an Analytic Services outline. A metaoutline is based on the structure of an OLAP model that you specify. Metaoutlines have the following features:

- They are reusable. You can use a metaoutline as the basis for more than one Analytic Services outline.
- They are centralized. You can define a metaoutline at a central location and use it to create multiple Analytic Services outlines in multiple locations.
- They enable you to create an Analytic Services outline that is based on criteria that you specify.
- They enable you to create Analytic Services databases on demand. You can create or update an Analytic Services database whenever it is necessary, either immediately or by scheduling periodic updates.
- They enable you to view sample Analytic Services outlines before you build them.
- They automatically generate the SQL necessary to retrieve data from an external data source.
- They enable you to filter data from data sources before you build the associated Analytic Services outline.

- They enable you to transform data as you create an Analytic Services outline. You can transform member names as you create an Analytic Services outline or transform data as you load it into an Analytic Services database.

Components of Metaoutlines

A metaoutline contains the following components:

- **One or more measures.** *Measures* are data values tracked by businesses. Measures typically include items such as SALES and COGS (cost of goods sold). Every metaoutline that you use to build an Analytic Services outline must include at least one measure.

In an SAP BW data warehouse, measures are referred to as *key figures*. They are always numerical, quantifiable values.

The Analytic Services database calculates the measures for each dimension intersection of the associated metaoutline. Measures can be aggregated in a pre-defined order using an SQL expression. The SQL expression uses an SQL template and a list of specified columns, attributes, and measures.

- **Two or more dimensions.** A *dimension* is a data category, containing *members*, used to organize business data for retrieval and preservation of data values.

In an SAP BW data warehouse, dimensions are referred to as *characteristics*. These characteristics contain *characteristic values* (or members),

In an SAP BW data warehouse, the term *dimension* is used specifically to denote a group of related characteristics; for example, a time dimension could refer to the characteristics, Calendar Year and Fiscal Year.

A dimension in a metaoutline creates a dimension in the associated Analytic Services outline. For example, if you include a Product dimension in a metaoutline, the Analytic Services outline that you generate from the metaoutline provides a Product dimension. Every metaoutline that you use to create an Analytic Services outline must include at least two dimensions.

- **One or more member levels.** A *member level* is a hierarchical level of detail within a dimension. A member level in a metaoutline creates one or more members at the same level in the associated Analytic Services outline.

For example, if the Product dimension of a metaoutline contains a PRODUCT_DESC member level, the Product dimension in the Analytic Services outline contains members, such as Birch Beer and Caffeine Free Cola, that correspond to the values in the PRODUCT_DESC member level in the source database.

You can designate your own user-defined members as shared members. This option enables your user-defined members to share storage space with other members of the same name.

- **Filters.** You can define a set of *filters* to control access to data. Filters determine which members of a member level that Integration Services adds to the associated Analytic Services outline. You can also define transformation rules that determine what transformations, if any, Integration Services performs on the members of a member level as it builds the associated Analytic Services outline.

- **Optional attribute dimensions.** Attribute dimensions in the metaoutline are based on attribute-enabled columns in the OLAP model. After an attribute dimension and member are created, you can define attribute properties, such as Boolean and numeric ranges, that enable you to view business data in finer detail than would otherwise be easily available.
- **Optional Hybrid Analysis low-level members.** Hybrid Analysis integrates a source database with an Analytic Services multidimensional database so that applications and reporting tools can directly retrieve data within both databases. When Hybrid Analysis is enabled for dimension members in a metaoutline, users of spreadsheets and report writer interfaces can access data contained in the Analytic Services database and drill down to data accessed directly from the data source.
- **Optional drill-through report members.** Integration Services enables you to define drill-through reports. A drill-through report is based on an *intersection level* (member combination) that spreadsheet users can double-click to start the drill-through process. Spreadsheet users can view or customize pre-defined drill-through reports that retrieve the relevant detail columns from the data source. Because Integration Server captures the metadata necessary to create the Analytic Services outline, it returns the drill-through report in the context of the data that spreadsheet users are viewing.
 - **Optional drill-through report to a URL.** You can specify a URL as a data source. To take advantage of this option, customize the drill-through SQL template by replacing the data source with an HTML source. Integration Services passes the URL to Essbase Spreadsheet Add-in or Essbase Spreadsheet Services (or another drill-through client), which then launches the URL in a browser.
 - **Optional drill-through report to a secondary source.** You can replace the primary data source with a secondary data source.
 - **Optional drill-through report to attributes and members enabled for Hybrid Analysis.** You can select from the following options for attribute drill-through reporting:
 - You can select attribute dimensions and attribute members as OLAP intersections.
 - You can select Hybrid Analysis dimensions as OLAP intersections.
 - You can select Hybrid Analysis member columns as intersections.
 - You can select attributes associated with members that have been enabled for Hybrid Analysis.

Using Hierarchies

Dimensions are usually structured to contain a *hierarchy* of related members. For example, for relational database users, the Time dimension might include members such as Year, Quarter, and Month. This hierarchy creates an Analytic Services outline with members such as 2004, Quarter1, and January.

Hierarchies also make use of *attributes* to classify members logically within a dimension. (For example, a Product dimension can have attributes such as Size and Flavor.) Hierarchies keep track of how the attributes relate to one another and how they are structured within a dimension.

You can create hierarchies for a metaoutline as you create an OLAP model. Refer to Integration Services Console online help for information on hierarchies.

The sample application provided with Integration Services includes a sample metaoutline, TBC Metaoutline. You can see the dimensions named Year, Market, Measures, Product, Scenario, and Supplier. The Year dimension contains a hierarchy—Quarter and Month.

Types of Hierarchies

There are several types of hierarchies used in data retrieval and analysis.

Balanced Hierarchy

A balanced hierarchy has two or more branches in a hierarchical tree, and each member is consistent with other members at the same level in each branch. For example, in a relational database, a dimension might have a branch for the year 2003 and a branch for 2004. In each of these two branches, Q1 would be at the same level, as would the months Jan, Feb, and Mar. Time dimensions typically have balanced hierarchies. In a data warehouse, a characteristic would have analogous branches.

Unbalanced Hierarchy

An unbalanced hierarchy contains branches with unequal numbers of member levels although the parent-child relationships are usually consistent from branch to branch. For example, a Sales Personnel dimension might have a branch for Sales Manager East and a branch for Sales Manager West. Each of these two branches then might have States. The Sales Manager East, however, might have four states and the Sales Manager West might have two states. In this case, the parent-child relationships on both branches are the same, but the member levels are not equivalent. Human resource dimensions sometimes have unbalanced hierarchies.

Ragged Hierarchy

A ragged hierarchy occurs when a dimension has branches in which there are different numbers of levels. For example, a Sales Regions dimension might have one branch for North America and a branch for Europe. Both branches have member level attributes for Country, State, and City.

In this example, the North America branch might have United States, Massachusetts, and Boston. The Europe branch might have Greece, Athens because the country of Greece does not have individual states in the sense that the United States does. Thus the State level for Greece is empty, causing a ragged hierarchy.

Geographical dimensions and product dimensions often have ragged hierarchies. For example, branches in a Product dimension may have member level attributes for Size, Color, Weight, and Frozen. Some of these member level attributes may be empty on products for which the attributes are not applicable.

Alternate Hierarchy

An alternate hierarchy is based upon an existing “primary” hierarchy but has alternate levels in the dimension. In [Table 3](#) below, the primary levels are in the four left columns, and the alternate levels are in the two right columns.

Table 3 Alternate Hierarchy

Prod Code	Prod Alias	Gen_01	Gen_02	Alt_Gen_01	Alt_Gen_02
100-10	Cola	Cola vs. Non-cola	Colas	Regular vs. Diet	Regular
100-20	Diet Cola	Cola vs. Non-cola	Colas	Regular vs. Diet	Diet
100-30	Decaf Cola	Cola vs. Non-cola	Colas	Regular vs. Diet	Regular
200-10	Vanilla	Cola vs. Non-cola	Non-colas	Regular vs. Diet	Regular
200-20	Diet Vanilla	Cola vs. Non-cola	Non-colas	Regular vs. Diet	Diet
200-30	Cream	Cola vs. Non-cola	Non-colas	Regular vs. Diet	Regular
200-40	Diet Cream	Cola vs. Non-cola	Non-colas	Regular vs. Diet	Diet
400-10	Grape	Cola vs. Non-cola	Fruit Sodas	Regular vs. Diet	Regular
400-20	Diet Grape	Cola vs. Non-cola	Fruit Sodas	Regular vs. Diet	Diet
400-30	Orange	Cola vs. Non-cola	Fruit Sodas	Regular vs. Diet	Regular
400-40	Diet Orange	Cola vs. Non-cola	Fruit Sodas	Regular vs. Diet	Diet

Deploying Hierarchies

A hierarchy is deployed using one of two methods:

- Standard
- Recursive

Standard Hierarchies

Under standard deployment, each attribute in the hierarchy defines one level. For example, in a Time dimension, the hierarchy might be organized so that the attribute Year defines one level, Quarter defines a second level, and Month defines a third level. This is illustrated in [Table 4](#).

Note: Standard deployment may be used with all hierarchy types.

Table 4 Standard Hierarchy Deployment

Year	Quarter	Month
2003	1st	Jan
2003	1st	Feb
2003	1st	Mar
2004	1st	Jan
2004	1st	Feb
2004	1st	Mar

Recursive Hierarchies

In a recursive deployment, the parent-child relationships of attributes are used to organize the hierarchy. For example, in a Sales Personnel dimension, the hierarchy might be organized under the parent-child relationships shown in [Table 5](#).

Note: A recursive deployment may be used with unbalanced hierarchies only.

Table 5 Recursive Hierarchy Deployment

Parent Attribute	Child Attribute	Child Attribute
Sales Manger East	New York	Albany
Sales Manger East	Pennsylvania	Pittsburgh
Sales Manger East	Massachusetts	Boston
Sales Manger East	Connecticut	Hartford
Sales Manger West	California	Los Angeles
Sales Manger West	Arizona	Phoenix

Recursive Hierarchies and SAP BW

In an SAP BW data warehouse, base dimension tables contain only leaf level (level 0) members. A recursive hierarchy dimension table contains parent-child hierarchical information which is used along with the leaf level members in the base dimensions to build a recursive hierarchy.

Note: It is strongly recommend that you not edit any recursive hierarchies imported from an SAP BW data warehouse into Integration Services unless you are familiar with data warehousing concepts and data modeling.

2

Preparing Data Sources

An OLAP model is a dimensional model of relational data in the form of a star schema. OLAP models are based on the idea that values in an external data source can be categorized as either facts (or key figure data values in SAP BW) or dimensions of facts.

Before you can create an effective OLAP model, you must understand and define the business needs and data sources that are available to you. You may need to modify some data sources to make the transition from relational databases or data warehouses to multidimensional databases as easy and efficient as possible.

For information about creating an OLAP model, refer to the Essbase Integration Services Console Help.

This chapter contains the following topics to help you prepare relational data:

In This Chapter	Defining User Needs	28
	Defining Data Sources	29
	Consolidating Data into a Single Data Source	29
	Deciding Whether to Create a Staging Area	30
	Cleansing Data	31
	Creating Views, Tables, and User-Defined Tables	31
	Adding Columns to Tables	37
	Creating Indexes	39
	Transforming Data	40
	Deciding Which Tables Are Available to OLAP Model Creators	41
	Creating Time and Accounts Dimensions	45
	Accessing Tables in the OLAP Metadata Catalog	52
	Mapping to DB2 Cube Views	52
	Using Text Files as Data Sources	53

Note: As noted in the appropriate topics in this chapter, many of the above tasks cannot be performed within Integration Services if you are using data from an SAP BW data warehouse. For information on performing these tasks, see the SAP BW documentation.

Defining User Needs

Before you start to design an OLAP model, consider the questions in this topic. By carefully constructing answers to these questions, you begin to create an effective OLAP model.

Remember that when you create an OLAP model, the ultimate goal is to create a multidimensional Essbase Analytic Services database. This topic assumes that you are familiar with the design principles for a multidimensional database. For more information, see [“About OLAP” on page 12](#) and the *Essbase Analytic Services Database Administrator’s Guide*.

What information (data) do users want to see in the Analytic Services database?

Examples of types of information are orders, sales, invoices, and the general ledger. Select data by combining an understanding of the business with an understanding of what data is available. After reviewing the data that you select, you decide whether to create one or multiple Analytic Services databases.

What is the maximum level of detail of data that users want to see in the Analytic Services database?

Select the level of detail by combining an understanding of the business with an understanding of the performance requirements of the Analytic Services database. (In general, the less detail stored in the Analytic Services database, the faster the performance.) For example, you can opt to store data that is aggregated by quarter or by month. The level of detail that you select is the basic level of detail that you represent in the fact table. In an OLAP model, the fact table is where the numeric measurements of the business are represented.

You can also use drill-through reports or Hybrid Analysis to offer users at the spreadsheet level alternative views of data and direct access to the source data.

Which dimensions apply to each fact table row?

A dimension is a data category. Typical dimensions are Product, Market, and Time. Base the selection of dimensions on an understanding of how users want to view data. In an SAP BW data warehouse, multidimensional dimensions are referred to as *characteristics*.

Each dimension has a number of members (characteristic values in data warehouses). For example, the Market dimension can include members representing New York, Boston, San Francisco, and Los Angeles. Each row in the fact table represents a combination of members, one from each dimension. For example, a row in the fact table can store the sales for Product A in New York in February.

Which measures do you want to represent in the fact table?

Measures are numeric quantities that vary over time. In an SAP BW data warehouse, measures are referred to as key figures. Examples of measures are quantity sold, cost of goods sold, and profit. Not all numeric values are measures, but all measures are numeric values. For example, although product size is a numeric value, you probably prefer to represent it as a member of a product dimension rather than as a member of a measure dimension because users do not want to compare product size over time.

Defining Data Sources

Now that you have defined the data that users want to see in the Analytic Services database, you need to define the data sources. Consider the following questions:

- **Is the data clean?**
Consider the quality and integrity of the source data. For more information, see [“Cleansing Data” on page 31](#).
- **Is the data calculated by a procedure and not stored; for example, a discount figure that is calculated for a specific product at a specific time?**
You need to create such information as tables in the chosen source database. Consider creating a staging area to perform this task. See [“Deciding Whether to Create a Staging Area” on page 30](#).
- **Is the data in a single structured query language (SQL) database or in multiple SQL data sources?**
Essbase Integration Server can access multiple SQL sources; however, you can consolidate SQL tables into a single SQL data source for each chosen source database.
- **Is the data in flat files?**
Integration Server does not access flat files; you need to create flat files as tables in the chosen source databases. Consider creating a staging area to perform this task. See [“Deciding Whether to Create a Staging Area” on page 30](#). Alternatively, after you build the Analytic Services database outline, you can use Essbase Administration Services to load data. For more information, see the Administration Services online help.
- **Is the data in an SAP BW data warehouse?**
If so, you should be familiar with differences between the terminology used in relational databases and the terminology used in SAP BW data warehouses. See [“About SAP BW Data Warehouses” on page 13](#).

Consolidating Data into a Single Data Source

When you create an OLAP model, the data that you need can come from multiple SQL data sources. You can, however, consolidate your data into a single data source.

- ▶ To consolidate your data into a single data source:
 - 1 Select a target data source to connect to the server running Integration Server.**
For a list of supported relational database management systems (RDBMS), see the *Essbase Integration Services ODBC Support Matrix*.
 - 2 Create any data that does not exist as an SQL data source; for example, create relational tables in the chosen data source for any information that is currently in flat files.**
 - 3 Consolidate the data source tables into a single database.**

When you consolidate the data, you may want to create a staging area.

Deciding Whether to Create a Staging Area

A *staging area* is an RDBMS database that you create to meet the needs of a specific application. Typically, a staging area is small (a maximum of 1 to 2 GB) in comparison to a data warehouse or an online transaction processing (OLTP) application. It is a snapshot of data; that is, it is not constantly updated with new data but is refreshed periodically from source data.

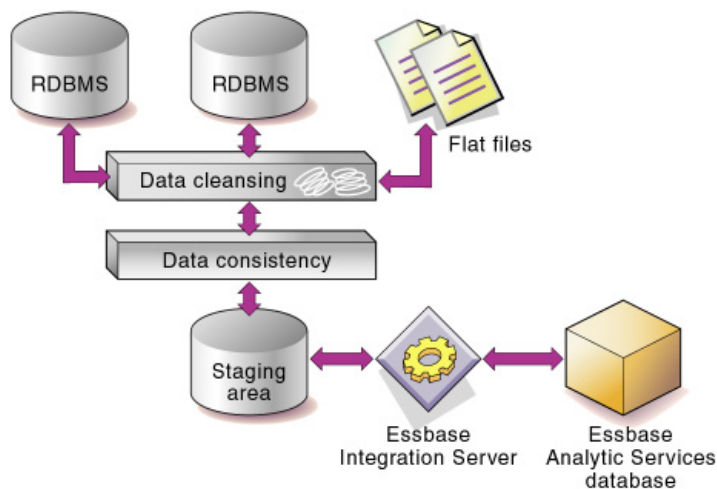
Note: For information on creating a staging area for an SAP BW data warehouse, see your SAP BW documentation.

Using a staging area provides advantages, because you can use it to perform any of the following tasks:

- Combine data from disparate or heterogeneous platforms without changing the original data in the data source
For example, you can combine data from various RDBMSs and flat files.
- Fine-tune data for a specific application
For example, you can calculate a subset of the data source data and store the calculated data in a staging area. You can then run faster queries on the precalculated data in the staging area.
- Create tables or views to denormalize the data so that it maps more easily to an OLAP model (see [“Creating Views, Tables, and User-Defined Tables”](#) on page 31)
- Transform data that is not consistently described (see [“Transforming Data”](#) on page 40)
The disadvantage of using a staging area is that the data in the staging area is a snapshot and therefore may not be truly representative data.

Figure 3 shows a staging area as part of a data warehouse. Data is copied into the staging area from the source data and converted from flat files and other data sources.

Figure 3 Data Warehouse with Staging Area



Cleansing Data

Integration Server does not cleanse invalid data for you. Data needs cleansing if it is inconsistent. Inconsistent data may include values that are incorrect, are not of the correct data type, or do not match integrity constraints (containing rows that do not have entries for the required key columns). Also, data may be inconsistent because the same value is entered in different forms. Such an inconsistency occurs, for example, when data is drawn from more than one source or when users enter data incorrectly.

If source data is inconsistent, you must fix, or cleanse, the data. Cleansing data can be a simple process, such as making suspect data into nulls, or a more complex process that requires the use of a data-cleansing tool. Similar data requires similar cleansing rules.

Creating Views, Tables, and User-Defined Tables

Integration Server does not distinguish between the tables and the views of a data source. You can use either tables or views that you create in your data source or user-defined tables that you create in the OLAP Model main window to build an OLAP model.

You may want to create views, tables, or user-defined tables specifically for use with Integration Server to provide security and convenience, and to make the transition from your data source to a multidimensional database as efficient as possible. By creating views, tables, or user-defined tables, the underlying structure in the data source remain unchanged.

Consider creating views, new tables, or user-defined tables if you want to reach either of these results:

- To build an Analytic Services hierarchy down to a specific level in a recursive table, see [“Building Analytic Services Hierarchies from Recursive Tables” on page 33](#).
- To create aliases in the Analytic Services database from data stored in multiple columns in the source data tables, see [“Creating Aliases for Dimension and Member Names” on page 44](#).

Also consider creating views, tables, or user-defined tables if the source tables meet any of the following criteria:

- The source tables contain unions. Integration Server does not generate SQL for unions. See [“Removing Unions” on page 35](#).
- The source tables have columns that you want to transpose to rows. See [“Transposing Columns and Rows” on page 35](#).
- The source tables are in a packaged application. If the source tables are in a packaged application, you may not know which tables contain the data that you need because the table names provided by the application may not be meaningful in terms of your business needs. You will probably need to ask an application specialist to create the required tables and views (with meaningful names) in a staging area in the target data source. See [“Deciding Whether to Create a Staging Area” on page 30](#).

- The source tables are highly normalized. Normalized data is appropriately grouped and does not include redundant data. Consider denormalizing data to facilitate the transition of the data to an Analytic Services database. See [“Denormalizing Source Data Tables” on page 36](#).

Deciding Whether to Create a View or a New Table

In most cases, you create views of the data source data (instead of new tables), because views ensure that data is current and efficiently stored. Also, with views, you do not need to maintain two sets of the same data.

When deciding whether to create a view or a new table, consider the following questions:

- **Does the data already exist?**
If the data does not exist, you need to create a new table. See [“Consolidating Data into a Single Data Source” on page 29](#).
- **Do you want to index columns that are not indexed in the source table?**
You need to create a new table because you cannot index columns in a view. See [“Creating Indexes” on page 39](#).
- **Do you want to index columns that contain data that you need to transform?**
Because many data sources ignore indexes on columns that have transformations, you probably need to create a new table. See [“Transforming Data” on page 40](#).
If you are using a staging area, you create tables and views in the staging area. See [“Deciding Whether to Create a Staging Area” on page 30](#).

Deciding Whether to Create a User-Defined Table

You create one or more user-defined tables for the same reasons described in [“Deciding Whether to Create a View or a New Table” on page 32](#). User-defined tables, however, are created directly in the OLAP Model main window of the Essbase Integration Services Console, not in the relational database or data warehouse.

Fact Tables in SAP BW

When creating an OLAP model from a relational data source, you have the option of selecting one or more relational source tables that you want to use as a fact table, time dimension, accounts dimension, and other relevant dimensions.

When working with an SAP BW data source, you do not, however, have the option of individually selecting views to use as components of the OLAP model.

When you import an SAP BW model, Hyperion Data Integration Connector examines the metadata of the selected InfoObject and extracts the information necessary to create the fact table. It also extracts information to create other OLAP model elements such as a time

dimension, other relevant dimensions, hierarchies, and attributes. Integration Services Console then displays the resulting OLAP model in the right frame of the OLAP Model main window.

Though it may be possible to edit the fact table, editing is neither advised nor recommended. If you have issues with the fact table derived by Hyperion Data Integration Connector, contact Hyperion Technical Support.

Building Analytic Services Hierarchies from Recursive Tables

A *recursive table* contains information in one column that is a parent or child of information in a second column. Essbase Integration Server can build Analytic Services outline hierarchies from a recursive source table.

When creating an Analytic Services hierarchy from a recursive table, follow these guidelines:

- To associate aliases or user-defined attributes (UDA) with members created from a recursive table, ensure that the column with which you will associate the alias or UDA is fully defined.
- When creating an OLAP model, join the recursive table to itself. For more information and for an example of a recursive source table, refer to the Integration Services Console online help.
- When creating a metaoutline, select the parent or child column that you want to filter on as a member level in the metaoutline. Refer to the online help for information on filters.

Building a Hierarchy Down to a Specific Level

To build the Analytic Services outline down to a specific level, you must create a view that contains data for only the levels that you want to build.

Creating Aliases or UDAs for Members in a Recursive Table

When you create an Analytic Services outline, you can associate aliases or UDAs with members in the outline.

If you are working with a recursive source table and you want to associate aliases or UDAs with members created from the recursive source table, you may need to prepare the data. The steps that you must take to associate aliases or UDAs vary, depending on whether the alias or UDA data is contained in the recursive source table or in a separate table.

If the alias or UDA data is in a separate table, you must complete specific steps when creating an OLAP model. The steps that you complete vary, depending on whether the column, with which you are associating the alias or UDA, is fully defined (see [Table 6](#) and [Table 4](#)). For more information on the specific steps, refer to the Integration Services Console online help.

If the alias or UDA data is in the recursive source table, you must complete specific steps when creating the OLAP model. The column with which you associate the alias or UDA *must* be fully defined, and all alias or UDA information *must* relate to the fully defined column.

If you want to associate an alias or UDA with the parent column of a recursive table, the parent column *must* be fully defined. A recursive table parent column is fully defined when the parent column contains every value (every member proposed for the Analytic Services hierarchy). Thus the parent column contains the lowest-level value in the hierarchy with a NULL value in the child column. In [Table 6](#), the GEO_PARENT column is fully defined because the GEO_PARENT column contains the lowest-level value, 01010, with a NULL child in the GEO_CHILD column.

Table 6 Fully Defined Parent Column

GEO_PARENT	GEO_CHILD
USA	East
East	Maine
Maine	Bangor
Bangor	01010
01010	<NULL>

If you want to associate an alias or UDA with the child column in a recursive table, the child column *must* be fully defined. A recursive table child column is fully defined when the child column contains every value (every member proposed for the Analytic Services hierarchy). Thus the child column contains the highest-level value in the hierarchy, with a NULL value in the parent column.

In [Table 7](#), the GEO_CHILD column is fully defined because the GEO_CHILD column contains the highest-level value, USA, with a NULL parent in the GEO_PARENT column.

Table 7 Fully Defined Child Column

GEO_PARENT	GEO_CHILD
<NULL>	USA
USA	East
East	Maine
Maine	Bangor
Bangor	01010

For more information on the specific steps that you must perform, refer to the Integration Services Console online help.

Removing Unions

Integration Server does not generate SQL for unions. A *union* is a type of join that combines the results of two SELECT statements. A union is often used to merge lists of values that are contained in two tables. If the source tables use unions, you must create a view (or views) of the data that does not use the unions before you can start to work with the data in Essbase Integration Services. For more information on unions, see the RDBMS documentation or the data warehouse documentation for your specific data source.

Transposing Columns and Rows

If you want to transpose columns and rows so that you can more easily transition data from various data sources to an Analytic Services database, transpose the columns before you start to work with the data in Integration Server. For example, if you want to create multiple Analytic Services members from a single relational database column, you can create a new table or view that transposes row values to column values.

Consider the following example in which you want to create multiple Analytic Services measures (Init_Sales, Subsequent_Sales, and Return_Sales) from a single database column (SALESTYPE).

The SALESACTUALS table contains the following columns:

Table 8 SALESACTUALS Table

PRODID	MKTID	TIMEID	INITSALES	SALESTYPE	PRODID
100	2	01-01-2000	100.00	Sales	100
100	2	01-02-2000	10.00	Returns	100
100	2	01-03-2000	50.00	Subsequent	100
100	2	01-04-2000	20.00	Returns	100
100	2	01-01-2000	100.00	Sales	100
:	:	:	:	:	:

You want to create the Analytic Services outline in the hierarchy illustrated in [Figure 4](#):

Figure 4 Accounts Hierarchy



You want each SALESTYPE value to form the Analytic Services member: SALES to form the Init_Sales member, RETURNS to form the Return_Sales member, and SUBSEQUENT to form the Subsequent_Sales member. To achieve these goals, create a view or a new table that transposes row values into column values. This example produces the following table:

Table 9 View of SALESACTUALS Table with Transposed Columns

PRODID	MKTID	TIMEID	INITSALES	RETURNS	SUBSEQUENT
100	2	01-01-2000	100.00	0.00	0.00
100	2	01-02-2000	0.00	10.00	0.00
100	2	01-03-2000	0.00	0.00	50.00
100	2	01-04-2000	0.00	20.00	0.00
:	:	:	:	:	:

The following example of Oracle SQL defines the view shown in [Table 9](#):

```

)
Create view SalesActuals_vw as
  (Select ProdId, MktId, TimeId,
    decode (SalesType, 'Sales', Sales, 0) "InitSales",
    decode (SalesType, 'Returns', Sales, 0) "Returns",
    decode (SalesType, 'Subsequent', Sales, 0) "Subsequent"
    from SalesActuals
  )

```

For more information on using SQL to define views, see the documentation for the relational database or data warehouse that you use.

When you have defined a new table or view, use it to create an OLAP model.

Denormalizing Source Data Tables

If data is highly normalized, it may not map clearly to an OLAP model. Normalized data is appropriately grouped and does not include redundant data. Even though you can use normalized source tables to create an OLAP model by specifying joins (refer to the Integration Services Console Help), it may be more efficient to create a new table of denormalized data in the RDBMS.

If you use the denormalized data source, Integration Server does not need to compute the joins when it builds the OLAP model.

Consider the following example. In the first three tables, data is highly normalized so that redundant data is minimized:

Table 10 Normalized Product Family Data

FAMILYID	FAMILYDESC
100	Colas
200	Root Beer

Table 11 Normalized Product Data

PRODID	FAMILYID
100-10	100
100-20	100
100-30	100

Table 12 Normalized Product Description Data

PRODID	PRODESC
100-10	Cola
100-20	Diet Cola
100-30	Caffeine Free Cola

The following table shows the denormalized data in one table:

Table 13 Denormalized Product Data

FAMILYID	FAMILYDESC	PRODID	PRODESC
100	Colas	100-10	Cola
100	Colas	100-20	Diet Cola
100	Colas	100-30	Caffeine Free Cola
100	Colas	100-10	Cola
200	Root Beer	200-10	Root Beer

Adding Columns to Tables

You can add columns to relational database tables to support member and measure properties in Integration Services. Setting up columns in the relational source database eliminates several manual tasks that are required by Integration Services and saves significant amounts of time when you later log in to Integration Services.

Setting Up Columns to Support Member and Measure Properties

Three basic tasks make up the process of adding columns to relational database tables so that you can use the member and measure properties in Integration Services:

1. Create columns in the relational database source.
2. Name the columns.
3. Define member and measure properties for the columns.

Note: For purposes of explaining the use of member and measure properties in columns, this chapter uses the column names of the Integration Services sample application.

Member and Measure Properties

Depending upon the nature and purpose of a column, a property is defined for the column. The following topics discuss the member and measure properties used by Integration Services in creating OLAP models and metaoutlines.

Joining Tables

In the data source tables, set up primary and foreign keys to join each of the following tables:

- Join tables that form the fact table. See [“Selecting Tables for the Fact Table” on page 42](#).
- Join tables within each dimension branch. See [“Selecting Tables for Dimensions” on page 43](#).
- Join dimension tables to the tables in the fact table.

When you join tables in the RDBMS, Integration Server automatically detects the join when it builds the OLAP model.

Optimizing Tables for Use with SAP BW

To take advantage of SAP BW data warehouses, Integration Services provides several features enabling you to optimize table access and thereby increase your performance.

Indexes for Small Tables

Sometimes you may have multiple joins across several tables. In this environment, system performance may be degraded because each join requires a full scan through each table for the data that you are attempting to access. Integration Services uses indexes so that a search is performed on a specific row rather than on an entire table. This is especially useful in small tables (those containing fewer than 10,000 rows).

Push Down Operations

Integration Services uses several different “push down” operations to speed access to and from tables. These operations are discussed in the following topics:

Push Down Projections

Integration Services “pushes down” to the desired column in a fact table rather than pulling in all the fields in the table.

Push Down Filters

Integration Services uses the WHERE clause rather than the SELECT clause in MDX statements to find the applicable table field.

Push Down Jobs

Integration Services takes multi-table SQL statements and places them into a single MDX statement, with each table having its own MDX statement. This results in more efficient join operations.

Push Down Group By

Integration Services provides push down aggregation in the SAP BW cube logic, so that the GROUP BY count is always equal to one.

Creating Indexes

Integration Server detects indexes (including bitmapped indexes) that you have defined on the source tables and uses them to create an Analytic Services outline and to load data. *Indexes* are pointers that are logically arranged by the values of a key. Indexes optimize access to relational data. Bitmapped indexes are specialized indexes that may improve performance during analysis of numeric data.

You can significantly improve performance by defining indexes on the appropriate data in the source tables.

Note: In an SAP BW data warehouse, aggregates can be created but not indexes that can be used by Integration Services.

Consider taking the following actions to improve performance:

- Define indexes on columns that you use to filter data in the source database or in the OLAP model. For example, if the source database contains columns for city and state and you filter on city or state (for example, `SELECT * FROM Region WHERE State = Ca%`), then index the columns on which you are filtering (in this example, the column State).

- If you are creating alias names and you filter on alias names, index the column containing alias names. For more information on alias names, see the *Essbase Analytic Services Database Administrator's Guide*.
- Define bitmapped indexes on numeric data that you use to filter the database. For example, if you filter on sales values `SELECT Product FROM ProdSales GROUPBY Product ORDERBY ProdId HAVING SUM(Sales)>15000`, then consider defining a bitmapped index on sales values. Most source databases support bitmapped indexes. For more information on bitmapped indexes, see the documentation for the RDBMS or data warehouse that you are using.

Transforming Data

You may need to transform some of the data, for example, to change date formats or to ensure that data is described consistently.

Note: For information on transforming data in an SAP BW data warehouse, see the SAP BW documentation.

Assume that you want to create an OLAP model that combines data from the sales databases and data from the financial databases. If the sales databases specify New York as `New_York` and the financial databases specify New York as `NY`, you can transform `NY` to `New_York` in the staging area (see [“Deciding Whether to Create a Staging Area” on page 30](#)) without changing the data in the original financial databases.

You can do some data transformations in OLAP models and metatoutlines. For a complete list of available transformations, refer to Integration Services Console online help. If, however, you must do significant transformations on the data, you may want to use a data transformation tool before you use the data in Integration Server.

Transformations that you must perform in the source database, and not in Integration Services, include these:

- The most frequently accessed transformations

For example, assume that you want the Analytic Services database outline to include members for Year, Qtr, and Month and that the data source table contains a column called `TRANSDATE`. `TRANSDATE` holds the transaction date for each row.

Transform the data ahead of time, creating a new source table that contains the columns `YEAR`, `QTR`, and `MONTH`. The new columns contain the data transformed from the `TRANSDATE` column. You can then index the `YEAR`, `QTR`, and `MONTH` columns to improve performance.

Note: Many data source databases ignore indexes on columns that contain transformations. You may need to transform the data and create a physical table with new columns that can be indexed.

- Transformations to take the intelligence out of key columns

Some key columns may include categories of information that you want to split out before using the data in Integration Server. For example, if the source table has a PRODID column containing product identification information, transform the data in the data source so that it maps to the data categories that you want to display in the Analytic Services outline. In [Figure 5](#), for example, the categories are CUSTID, STATE, and PRODNUM.

Figure 5 Transforming a Product Code Stored in a PRODID Column



Deciding Which Tables Are Available to OLAP Model Creators

When you connect to one or more source databases, Integration Services Console displays in the left frame all tables to which the source database user ID provides you access (SELECT permission).

For all source databases, if you have partial access to a table, Integration Server displays the whole table. (For example, if you have SELECT permission for only half of the columns in a table, you can see the whole table.) In the OLAP model, however, do not use a column to which you do not have SELECT permission. Integration Server cannot filter, load data, or load members in a column for which you do not have SELECT permission.

Note: For DB2, Integration Server displays all tables that are in the database, regardless of whether you have access to them. If, however, if you do not have access (SELECT permission) to a table, you cannot view the columns in the table or use the table to build an OLAP model.

Note: Your system performance can suffer significantly if the username employed to access the RDBMS data source has SELECT permission to several more tables than are used by Integration Services

For added security and ease of administration, you may want to define which tables the users can see by taking one or more of the following actions:

- Create specific RDBMS user IDs for users who are creating OLAP models.
- Create views of some data and provide users who are creating OLAP models with access to the views rather than access to the original tables.
- Use conditions to restrict a view so that users who are creating an OLAP model can see only a subset of the view. For example, you can restrict the view to show data only for the current year although the table contains data for three years. For more information on adding restrictive conditions, see the documentation for the RDBMS or data warehouse that you use.

Note: When you connect to the source database, Integration Server provides read-only access to it, regardless of the access provided by the source database user ID. For example, if the source database user ID has read and write access, you have only read-only access when creating OLAP models.

Note: When configuring ODBC data for Teradata, you should always use the UseXViews option.

Selecting Tables for the Fact Table

A fact table is a logical container for the relational tables that define the data values (measures) for each dimension intersection in an OLAP model. In an SAP BW data warehouse, measures are referred to as *key figures* and facts are *key figure* values; dimensions are *characteristics*. (See [“About SAP BW Data Warehouses” on page 13.](#)) For additional information on selecting tables for fact tables, refer to Integration Services Console online help.

The type of Analytic Services database that you want to create determines which source tables you must include in the fact table. See [“Defining User Needs” on page 28.](#)

When you select source tables to include in the fact table, consider both how the data is distributed among the tables and how the data relates to the dimensions in the Analytic Services database.

Select a table or tables that contain all the measure data that Analytic Services end users want to see. For example, if the source database contains the following ORDERS and ORDERDETAILS tables, you must include both tables if you want the fact table to include data from both tables.

Table 14 ORDERS and ORDERDETAILS Tables for the Fact Table

Orders	ORDERID	TRANSDATE	SHIPID	EMPTYID
Order Details	ORDERID	PRODID	NUMUNITS	UTPRICE

Select a table or tables that represent all the dimensions (data categories) on which the Analytic Services end users want to analyze data. For example, if end users need to analyze product sales by time, by geography, and by sales channel, select a table or tables that contain key columns for each of these dimensions.

Table 15 Sales Table for the Fact Table

Sales	PRODID	SALES	DATESOLD	CITY	CHANNEL	UNTSSOLD
								:

Note: If your data source is a relational database, you can improve performance by setting up primary and foreign keys, and joining the tables that form the fact table. See [“Adding Columns to Tables” on page 37](#).

Selecting Tables for Dimensions

A dimension is a data category. In an SAP BW data warehouse, the components analogous to multidimensional dimensions are referred to as *characteristics*, and the term *dimension* denotes a group of related characteristics. See [“About SAP BW Data Warehouses” on page 13](#). Typical dimensions are Product, Geography, and Time.

A dimension table is a logical container within an OLAP model. A dimension table includes one or more relational tables that define a potential Analytic Services dimension. A dimension table can join to other dimension tables, forming a *dimension branch*.

The data that you want to make available for analysis in the Analytic Services application that you create determines which source tables you use to create dimensions. See [“Defining User Needs” on page 28](#). When you select source tables for each dimension, select a source table or tables that include the maximum amount of information about the data category (dimension). For example, if you have the following PRODUCTS and ORDERS source tables, select the PRODUCTS table to form the Products dimension.

Table 16 ORDERS and PRODUCTS Tables for the Dimension Table

Products	PRODID	PRODNAME	SIZE	COLOR
Orders	PRODID	PRODNAM	ORDERID	DATE

The ORDERS table contains product information for only products that have been sold. The PRODUCTS table contains data for all products. Thus, Analytic Services end users can use the PRODUCTS table information to analyze products that are and are not selling.

Creating Aliases for Dimension and Member Names

An *alias* is an alternative name for a member or dimension in an Analytic Services outline. An alias often provides an easily identifiable label, such as a product name or product description, for a column in a relational data source. For example, in the Analytic Services Sample Basic database, the 200-20 member in the Product dimension has the alias Diet Root Beer.

Note: In an SAP BW data warehouse, an alias is the text for a characteristic.

You can create up to nine alias tables (including the Default alias table) for a given metaoutline. You can use the alias tables that you create to store up to nine aliases for a given dimension or member (including attribute dimensions, attribute members, and user-defined members).

If the source data contains information that you want to use to create one or more aliases in an Analytic Services outline, include the relevant data as one or more columns in the associated OLAP model. Then, when you create a metaoutline, you can retrieve alias information from one or more columns of the OLAP model.

One column of the source data can be used to create multiple aliases. Conversely, multiple columns of the source data can be used to create a single alias.

If the alias information from the source data for a single alias is in more than one column, you must complete one of the following data preparation procedures:

- In the source database, create a view or a new table to concatenate (and, if necessary, transform) the alias information into a single column. Creating a view or new table may be the easiest method, especially if the alias information is stored in three or more columns.
- In the OLAP model, concatenate (and, if necessary, transform) the columns that contain the alias information. For more information on concatenating and transforming columns, refer to Integration Services Console online help.
- In the OLAP model, use the pass-through feature to run a procedure that uses relational logic to retrieve alias information from the column or columns that you specify. For example, you can use the pass-through feature to run a procedure in DB2. The DB2 procedure uses CASE logic to retrieve alias information from two different columns; if a field in one column is empty, the procedure tells the source database to retrieve alias information for that field from a different column.

For more information on pass-through transformations, refer to Integration Services Console online help. For more information on procedures and CASE logic, see the RDBMS documentation or the data warehouse documentation.

Note: Although Analytic Services databases allow ten alias tables for each outline, you can only associate up to nine alias tables (including the Default alias table) for each Analytic Services outline that you create using Integration Services.

See Integration Services online help for information on creating alias tables and assigning multiple aliases in the metaoutline.

Creating Time and Accounts Dimensions

In an OLAP model, you can create time and accounts dimensions that carry over directly to the Analytic Services outline as the dimensions tagged as time and accounts. The *time dimension* table contains date-related columns from the relational data source. The *accounts dimension* table contains measures columns that duplicate the columns contained in the fact table.

Note: In an SAP BW data warehouse, multidimensional dimensions are referred to as *characteristics*, and the term *dimension* denotes a group of related characteristics. See [“About SAP BW Data Warehouses” on page 13](#).

Preparing Data for Time Analysis

If the business application requires time-related analysis, such as examining trends over time or making seasonal sales comparisons, include a time dimension in the Analytic Services outline. A time dimension includes members for the time periods that you report on; for example, monthly, quarterly, and yearly sales.

To support a time dimension, the data source should include the following elements:

- The data source should include one or more time-related columns associated with measure data. For example, if you want to report on sales per month and per quarter, one of the tables that you include in the fact table needs a date column to reflect sales by month and quarter.
- If needed, the data source should include one or more tables containing user-defined calendars. A user-defined calendar maps the time-related column in the fact table to a specific business calendar.

Associating Time Data with Measure Data

To enable Integration Server to map the data that you are measuring (such as sales or costs) to specific time periods (such as month, quarter, or year), the source data for the fact table must contain one or more time-related columns. For example, from a transaction date column, you can determine the day, week, and month in which the associated transaction occurred. Or, the transaction data can include separate columns for each time period. These time periods are potential members of the dimension tagged as time.

Working with Summary Data

If the measures columns in the source data contain summary data, the time-related column must indicate the time period for which the data is summarized. For example, assume that the SALESINVACT table contains measures columns SALES and COGS; a date-related column, TRANSDATE; and STATE and PRODCODE columns that categorize by sales date, state, and product code.

Table 17 TBC_MD Data in the SALESINVACT Table

STATE	PRODCODE	TRANSDATE	SALES	COGS	...
IL	100-10-C001-S0002-P001	Jan 4 2000 12:00AM	672.00	217.00	
IL	100-10-C001-S0002-P001	Feb 4 2000 12:00AM	241.00	70.00	
IL	100-10-C001-S0002-P001	Mar 3 2000 12:00AM	287.00	84.00	
IL	100-10-C001-S0002-P001	Apr 7 2000 12:00AM	295.00	85.00	
IL	100-10-C001-S0002-P001	May 3 2000 12:00AM	702.00	207.00	

Each row in the SALESINVACT table summarizes product sales and costs for a particular date for customers from a particular state. Instead of summarizing data by day, a row can summarize data by other periods; for example, by week or by month.

Formatting the Date

Integration Server supports three *data types*: string, numeric, and datetime. The basic Open Database Connectivity (ODBC) data types map to four data types within Integration Services: text, numeric, Boolean, and datetime. The datetime data type is a timestamp that includes the year, month, day, and time of day.

The data type of the source data column that contains time-related data does not matter. For example, the date can be a timestamp in a column with a datetime data type, or it can be keyed data in a numeric column (for example, 09232000 or 20000923). A string column can contain dates with or without separating characters such as slashes or periods (for example, 09/23/2000 or 23.09.2000 or 09232000). Portions of the date (such as the day, week, month, or quarter) can be included in the source data as separate string or numeric columns.

It is important that the values in the time-related columns can be mapped to the time periods that you want in the time dimension. If dates are stored as datetime data types and you want quarterly totals, Integration Server can determine the quarter. If dates are stored in numeric or string columns and you want quarterly totals, you must have an explicit column containing the associated quarter for the data that you want to consolidate.

Working with Data in Similar Time Periods

In certain applications, such as general ledger and legacy systems, tables may be organized so that columns identify similar time periods. For example, a single row may include all months of a year. In this case, each subsequent row includes a measure category as a data value; for example, a specific account number.

Table 18 Example of a Table Where Columns Define the Time Periods

PRODCODE	STATE	ACCOUNT	JAN	FEB	MAR	...
100-10-C001-S002-P001	AZ	Sales	672.00	241.00	287.00	
100-10-C001-S002-P001	AZ	COGS	403.00	132.00	177.00	
100-10-C001-S002-P001	CA	Sales	401.00	143.00	378.00	
100-10-C001-S002-P001	CA	COGS	260.00	101.00	226.00	

To create a dimension of the type time, Integration Server requires that all time identifiers be in one column and that accounts identifiers be in separate columns, as shown in [Table 19](#). If, for example, data is stored with each month as an individual column, as shown in [Table 18](#), you should restructure the data. Individual months should be data values in a MONTH column, and the measures categories, such as SALES and COGS, should be separate columns.

Table 19 Example of a Table Where Time Periods Are Stored as Data Values

PRODCODE	STATE	MONTH	SALES	COGS
100-10-C001-S002-P001	AZ	1	672.00	403.00
100-10-C001-S002-P001	AZ	2	241.00	132.00
100-10-C001-S002-P001	AZ	3	287.00	177.00
100-10-C001-S002-P001	CA	1	401.00	260.00
100-10-C001-S002-P001	CA	2	143.00	101.00
100-10-C001-S002-P001	CA	3	378.00	226.00
100-10-C001-S002-P001	AZ	1	672.00	403.00

You can write a program to convert the data to a table that uses the appropriate format or use the source database to transpose the columns (see [“Transposing Columns and Rows” on page 35](#)).

Accounts Dimensions and SAP BW

When you import an SAP BW model to create an OLAP model, Hyperion Data Integration Connector examines the InfoObject that you selected for measures, or accounts, information. The accounts information is collected and presented in the OLAP model as one coherent accounts dimension that contains the appropriate entities given the structure of the SAP BW database.

There are situations in which the SAP BW data source does not contain accounts elements. In such a case, Hyperion Data Integration Connector cannot create an accounts dimension.

Though it may be possible to create or edit an accounts dimension, it is neither advised nor recommended. If you have issues with the accounts dimension derived by Hyperion Data Integration Connector, contact Hyperion Technical Support.

Note: You cannot enable the accounts dimension or its members for Hybrid Analysis.

Time Dimensions and SAP BW

In an SAP BW data source, time entities such as days, months, and years are generally stored as separate dimensions. When you import an SAP BW model to create an OLAP model, Hyperion Data Integration Connector examines the InfoObject that you selected for time information. The various SAP BW time dimensions are collected and presented in the OLAP model as one coherent time dimension that contains the appropriate entities given the structure of the SAP BW database.

For example, suppose an SAP BW data source contains separate dimensions for the time elements Day, Month, Quarter, and Year. When you create an OLAP model using Hyperion Data Integration Connector, the OLAP model will contain a dimension Time with the members Year, Quarter, Month, and Day arranged in a hierarchy.

There are situations in which the SAP BW data source does not contain time elements. In such a case, Hyperion Data Integration Connector cannot create an accounts dimension.

Though it may be possible to create or edit a time dimension, is not advised or recommended. If you have issues with the time dimension derived by Hyperion Data Integration Connector, contact Hyperion Technical Support.

Slowly Changing Dimensions in SAP BW

In business environments, some metadata and data can be very stable over long periods of time then experience a change in structure; for example, Seattle could be in the Northwest Region for several years before being moved to the Pacific Coast Region. In SAP BW data warehouses, this concept is known as *slowly changing dimensions*.

When you access a slowly changing dimension, you see the data from a given point in time. SAP BW contains a *key date*, often referred to as the *active date range*, however this is not supported in Integration Services Release 7.1.2. Therefore, when you access data from a slowly changing dimension in SAP BW, you always see the latest value and only the latest value.

Handling Attribute-Enabled Columns in SAP BW

When you import an SAP BW model to create an OLAP model, Hyperion Data Integration Connector examines the InfoObject that you selected for attribute information and marks the appropriate columns as attributes.

To view the columns that have been marked as attribute-enabled in the resulting OLAP model, right-click a dimension table name in the OLAP model and select Properties and then select the Columns tab. Columns marked as attribute-enabled contain a “Y” in the Attribute column.

You can edit the attribute status of the columns in a dimension table by deselecting any of the items that are already selected, or selecting items that are not already selected.

Consolidation Properties

The following table defines the values for consolidation properties that Integration Services recognizes when loading these properties from a database column.

Table 20 Consolidation Properties Settings

Code	Description
+	Addition—Adds the member to the result of previous calculations performed on other members. This operator is the default operator.
-	Subtraction—Multiplies the member by -1 and then adds the result to the sum of previous calculations performed on other members.
*	Multiplication—Multiplies the member by the result of previous calculations performed on other members.
/	Division—Divides the member by the result of previous calculations performed on other members.
%	Percent—Divides the member by the sum of previous calculations performed on other members and multiplies the result by 100 to yield a percentage value.
~	Ignore—Does not use the member in the consolidation to its parent.

Two Pass Calculation Properties

The following table defines the values for two pass calculation properties that Integration Services recognizes when loading these properties from a database column.

Table 21 Two Pass Calculation Properties Settings

Code	Description
(Blank)	A two-pass calculation will not be used on the member.
T	Two Pass Calculation—A two-pass calculation will be used on the member.

Data Storage Properties

The following table defines the values for data storage properties that Integration Services recognizes when loading these properties from a database column.

Table 22 Data Storage Properties Settings

Code	Description
S	Store Data—The member stores data. This is the default storage property.
V	Dynamic Calc and Store—The data associated with the member is not calculated until a calculation is requested by a user. The calculated data is then stored.
X	Dynamic Calc—The data associated with the member is not calculated until it is requested by a user. The calculated data is not stored; it is discarded after the request is completed.
N	Never Share—The data associated with the member is duplicated with its parent or child only if an implied shared relationship exists.
O	Label Only—Although the member has no data associated with it, it can still display a value.

Time Balance Properties

The following table defines the values for time balance properties that Integration Services recognizes when loading these properties from a database column.

Table 23 Time Balance Properties Settings

Code	Description
(Blank)	None—This property is the default value. When you set the time balance property as none, Analytic Services rolls up parents in the time dimension in the usual way—the value of a parent is based on the formulas and consolidation properties of the children of the parent.
F	First—Set the time balance as first when you want the parent value to represent the value of the first member in the branch (often at the beginning of a time period).
L	Last—Set the time balance as last when you want the parent value to represent the value of the last member in the branch (often at the end of a time period).
A	Average—Set the time balance as average when you want the parent value to represent the average of the children values.

Skip Properties

The following table defines the values for skip properties that Integration Services recognizes when loading these properties from a database column.

Table 24 Skip Properties Settings

Code	Description
(Blank)	None Does not skip data when calculating the parent value. This property is the default value. If, however, Essbase encounters #MISSING data when calculating an average, it does not divide by the total number of members. It divides by the number of members with actual values. Therefore, setting the skip property to none or #MISSING does not affect average (but does affect first and last).
M	Missing Skips #MISSING data when calculating the parent value.
Z	Zeros Skips data that equals zero when calculating the parent value.
B	Both Missing and Zeros Skips both #MISSING data and data that equals zero when calculating the parent value.

Variance Reporting Properties

The following table defines the values for variance reporting properties that Integration Services recognizes when loading these properties from a database column.

Table 25 Variance Reporting Properties Settings

Code	Description
	Non Expense For non-expense items, such as sales, actual expenses should be higher than budgeted expenses. When actual expenses are less than budget, variance is negative. The @VAR function calculates ACTUAL - BUDGET. For example, if budgeted sales are \$100 and actual sales are \$110, the variance is 10. By default, members are non-expense.
E	Expense For expense items, actual expenses should be lower than budgeted expenses. When actual expenses are greater than budgeted expenses, variance is negative. The @VAR function calculates BUDGET - ACTUAL. For example, if budgeted expenses are \$100 and actual expenses are \$110, the variance is -10.

Currency Conversion Properties

The following table defines the values for currency conversion properties that Integration Services recognizes when loading these properties from a database column.

Table 26 Currency Conversion Properties Settings

Code	Description
(Blank)	None This dimension has no relationship to currency conversion. This is the default setting.
	No Conversion This dimension is not converted because it is not a currency value. It can be a value such as a quantity or percentage.

Accessing Tables in the OLAP Metadata Catalog

If you log on with a particular user ID when you create OLAP Metadata Catalog and then log on with a different user ID, you cannot access the tables in the OLAP Metadata Catalog unless you create an alias for the user ID (Sybase, SQL Server) or synonyms for the tables (DB2, Oracle). For more information, see the *Essbase Integration Services Installation Guide*.

You must perform a different procedure when time measures are stored as columns that include accounts information, as described in the Integration Services Console Help.

Mapping to DB2 Cube Views

Integration Services supports a utility provided by IBM DB2 Cube Views that enables you to import OLAP model and metaoutline metadata objects from DB2 Cube Views into Integration Services.

To use the utility (called *the bridge*), import your models or metaoutlines into an XML file. The XML file maps Integration Services metadata objects with DB2 Cube Views metadata objects. The XML file is then used to import the metadata objects from the DB2 catalog into the Integration Services metadata catalog.

[Table 27](#), below, lists how Integration Services metadata objects map to DB2 Cube Views metadata objects.

Table 27 Mapping Integration Services Objects to DB2 Cube Views Objects

DB2 Cube Views metadata object	Integration Services metadata object
Cube model	Model
Facts	Fact
Measure	Numeric member of fact
Dimension	Dimension

Table 27 Mapping Integration Services Objects to DB2 Cube Views Objects (Continued)

DB2 Cube Views metadata object	Integration Services metadata object
Attribute	Dimension view member
Hierarchy	Hierarchy
Join	Join (logical and physical)
Cube	Metaoutline
Cube dimension	Non-accounts dimension
Cube facts	Accounts dimension
Cube hierarchy	Metaoutline dimension hierarchy

Using Text Files as Data Sources

In Integration Services, you can use a text file (often referred to as a *flat file*) as a data source; however, the following conditions may affect your use of the product:

- Online database connectivity (ODBC) driver cannot extract month names from date fields in flat files. For example, the driver cannot extract “January” from the date field “01/30/04.” The ODBC driver does, however, support the numerical representation. During a member load, you can apply a transformation so that the numerical value is converted into the name of the month.
- If the data source is a flat file, the default Date Hierarchies provided by Integration Services will not function with month names. If you wish to use the Date Hierarchy functionality, define your month as a numerical value.
- If the data source consists of several flat files that resemble or consist of tables, the importing of data will be much like that of importing relational data. If the data source is one large, contiguous flat file with several different columns, it is likely that you will need to “cleanse” your data beforehand (see [“Cleansing Data” on page 31](#)). You may also need to define your joins explicitly for optimal performance (see [“Joining Tables” on page 38](#) and refer to the Integration Services Console Help).

Index

A

- accessing tables in OLAP Metadata Catalog, [52](#)
- accounts dimension
 - sample hierarchy, [35](#)
 - use in Analytic Services outline, [45](#)
- adding columns to tables, [37](#)
- aggregation
 - choosing level of, [28](#)
 - defined, [14](#)
- aliases
 - concatenating, [44](#)
 - creating for dimension or member names, [44](#)
 - multiple aliases, using, [44](#)
 - pass-through transformations and, [44](#)
 - recursive tables and, [33](#)
- Analytic Server
 - overview of, [14](#)
- Analytic Server, defined, [12](#)
- Analytic Services
 - accounts dimension, [45](#)
 - Analytic Server, defined, [12](#)
 - databases, workflow, [18](#)
 - designing a database for user needs, [28](#)
 - hierarchies, creating, [33](#)
 - members, compared to metaoutline member levels, [21](#)
 - users, [14](#)
- applications, sample, [vi](#)

B

- balanced hierarchies, [23](#)
- batch processing, Integration Services Shell, [18](#)
- bitmapped indexes, defined, [39](#)

C

- characteristic values, [13](#)
- characteristics, [13](#)
- cleansing data, [31](#)
- columns
 - adding to tables, [37](#)
 - availability, aliases, [44](#)
 - capitalization, [vi](#)
 - fully-defined in recursive tables, [33](#)
 - indexes, [39](#)
 - multiple sources, [29](#)
 - supporting member and measure properties, [38](#)
 - time-related, defining in source data, [45](#)
 - transposing with rows, [35](#)
- concatenation and aliases, [44](#)
- Console. *See* Integration Services Console
- consolidation
 - defined, [14](#)
 - drill down and, [15](#)
 - example, [14](#)
 - members, [12](#)
- consolidation properties, [49](#)
- consulting services, [ix](#)
- creating
 - Analytic Services databases, workflow, [18](#)
 - hierarchies, [33](#)
 - indexes, [39](#)
- currency conversion properties, [52](#)
- customer comparisons, [12](#)

D

data

- cleansing, 31
- combining multiple sources, 30
- denormalizing, 36
- drill down, 15
- ensuring consistency, 31
- missing, in sample databases, vi
- pivoting to change view, 15
- preparing for time analysis, 45
- source preparation, 27
- transforming in data source, 40
- types of, 28

data sources

- build levels, 33
- defining, 29
- OLTP databases, 15
- parent-child, 33
- sample, vi
- spreadsheet files, 15
- SQL, 15
- text files, 15
- workflow for transferring, 18

data storage properties, 50

data types and cleansing, 31

data values. *See* measures

data warehouse, diagram, 30

data warehouses

- source of data, 15
- terminology, 13

databases. *See* data sources

dates, formatting in source data, 46

DB2, tables displayed in, 41

denormalizing data, 36

deploying hierarchies, 24

deploying hierarchies, 24 to 25

detail data

- choosing level of, 28
- retrieving, 15

dimension branches, defined, 43

dimension names, creating aliases for, 44

dimension tables

- defined, 20, 43
- OLAP models, 20

dimensions

- choosing, 28
- choosing tables for, 43
- compared with characteristics, 13
- creating in a metaoutline, 20
- defined, 12, 19, 21, 27, 43
- examples, 12
- frequency of change, 12
- members, 12
- SAP BW, 28
- Scenario, 20
- TBC Metaoutline and, 23
- use in consolidation, 12
- using in metaoutlines, 20

documents

- Hyperion Download Center, vii
- ordering print documents, vii

drill down, 15

drill up, 15

drill-through report members, 22

drill-through reports, 22

EEssbase Analytic Server. *See* Analytic ServerEssbase Analytic Services. *See* Analytic ServicesEssbase Integration Server. *See* Integration ServerEssbase Integration Services Shell. *See* Integration Services ShellEssbase Integration Services. *See* Integration ServicesEssbase. *See* Analytic Services**F**

fact table

- choosing tables for, 42
- compared with dimensions, 28
- defined, 19, 42
- measures and, 42
- role in OLAP model, 19
- SAP BW, 19

facts, defined, 19, 27

filters, 39

flat files

- as a data source, 29, 53
- combining with SQL data sources, 30

forecasting, [14](#)

fully defined columns, defined, [33](#)

G

general ledger data, [46](#)

H

hierarchies

balanced, [23](#)

building from recursive tables, [33](#)

consolidation and, [12](#)

creating, [33](#)

data sources for building to a specific level, [33](#)

deploying, [24 to 25](#)

ragged, [23](#)

recursive, [24 to 25](#)

standard, [24](#)

types, [23](#)

unbalanced, [23](#)

using, [22](#)

Hyperion Connect, [13](#)

Hyperion Consulting Services, [ix](#)

Hyperion Download Center, accessing, [vii](#)

Hyperion Essbase. *See* Analytic Services

Hyperion Integration Server Desktop. *See* Integration Services Console

Hyperion Integration Server. *See* Integration Services

Hyperion product information, [ix](#)

Hyperion support, [ix](#)

I

indenting, [12](#)

indexes

SAP BW, [38](#)

working with, [39 to 40](#)

indexes, working with, [39 to 40](#)

Integration Server, defined, [17](#)

Integration Services

about, [16](#)

data warehouse, [30](#)

defined, [11](#)

Integration Services Console, defined, [17](#)

Integration Services Shell, defined, [18](#)

OLAP Metadata Catalog, defined, [18](#)

product family, [17 to 18](#)

workflow, [16, 18](#)

Integration Services Console, defined, [17](#)

Integration Services Shell

batch processing, [18](#)

defined, [18](#)

integrity constraints, cleansing data, [31](#)

intersections, defined, [12](#)

J

joins

joining tables in your RDBMS, [38](#)

reducing through denormalization, [36](#)

unions, removing, [35](#)

K

key figures, [13](#)

keys, columns, transformations and, [41](#)

L

legacy data, preparing, [46](#)

levels, data sources for building, [33](#)

logical model, description, [19](#)

M

measure filters. *See* filters

measures

bitmapped indexes and, [40](#)

compared with key figures, [13](#)

data, associating with time periods, [45](#)

defined, [21, 28](#)

fact table and, [42](#)

properties, [38](#)

SAP BW, [21, 28](#)

using with columns, [38](#)

member filters. *See* filters

member levels, defined, [21](#)

member names, creating aliases, [44](#)

members

compared with characteristic values, [13](#)

consolidation, [12](#)

defined, [12](#)

dimensions, [12](#)

- examples, [12](#)
- OLAP models, [12](#)
- properties, [38](#)
- using with columns, [38](#)
- metadata
 - accessing OLAP Metadata Catalog, [52](#)
 - defined, [18](#)
- metadata repository. *See* OLAP Metadata Catalog
- metadata, OLAP Metadata Catalog storage, [18](#)
- metaoutlines
 - advantages of, [20](#)
 - components, [21](#), [23](#)
 - creating dimensions, [20](#)
 - Integration Services sample, [vi](#)
 - sample, [vi](#)
- models. *See* OLAP models
- multidimensional databases
 - defined, [12](#)
 - source data, [15](#)
 - uses, [14](#)
 - workflow, [16](#), [18](#)
- multiple alias tables, using, [44](#)

N

- normalized data, denormalizing, [36](#)
- numeric data, indexes and, [40](#)

O

- ODBC, SQL versions supported, [15](#)
- OLAP Builder. *See* metaoutlines
- OLAP Catalog. *See* OLAP Metadata Catalog
- OLAP Command Interface. *See* Integration Services Shell
- OLAP Integration Server. *See* Integration Server
- OLAP Metadata Catalog
 - accessing tables, [52](#)
 - configuring for ODBC, [18](#)
 - defined, [18](#)
- OLAP metaoutline. *See* metaoutlines
- OLAP models
 - choosing which tables are displayed, [41](#)
 - components, [19](#)
 - creating, [17](#)
 - defined, [19](#)
 - dimension tables, [20](#)

- dimensions, [12](#)
- fact table, [19](#)
- graphic illustration, [19](#)
- Integration Services sample, [vi](#)
- members, [12](#)
- sample, [vi](#)
- uses, [11](#)

OLAP Server. *See* Analytic Server

Online Analytical Processing (OLAP), defined, [12](#)

Online Transaction Processing (OLTP) data, examples of contents, [15](#)

Open Database Connectivity. *See* ODBC

operational data, [15](#)

P

- parent-child data source, [33](#)
- pass-through transformations, aliases and, [44](#)
- performance
 - bitmapped indexes and, [39](#)
 - increasing through denormalization, [36](#)
- permissions, required in the SQL data source, [41](#)
- perspective, changing data on a spreadsheet, [15](#)
- pivoting, using to change data view, [15](#)
- privileges. *See* permissions
- product code, example, [41](#)
- product comparisons, [12](#)
- properties
 - consolidation, [49](#)
 - currency conversion, [52](#)
 - data storage, [50](#)
 - members, [38](#)
 - skip, [51](#)
 - time balance, [50](#)
 - two pass calculation, [49](#)
 - variance reporting, [51](#)
- push down operations with SAP BW, [39](#)

R

- ragged hierarchies, [23](#)
- read-only access, SQL data sources and, [42](#)
- recursive hierarchies, [24](#) to [25](#)
- recursive tables
 - aliases and, [33](#)
 - creating Analytic Services hierarchies from recursive tables, [33](#)

- defined, 33
- defining for alias or UDAs, 33
- fully-defined columns for, 33
- rules for, 33

redundant data, denormalizing, 36

relational databases. *See* data sources

repository. *See* OLAP Metadata Catalog

roll-ups, defined, 14

rows, transposing with columns, 35

S

sample

- applications, vi
- hierarchy, accounts dimension, 35
- metaoutline, description, vi
- OLAP model, description, vi

SAP BW

- aliases, 44
- data source, 29
- dimensions, 28
- fact table, 19, 42
- indexes, 38 to 39
- Integration Server, 17
- measures, 21, 28
- performing tasks, restrictions, 27
- push down operations, 39
- staging area, 30
- star schema, 27
- support of Integration Services, 13
- tables, 38, 43
- terminology, 13
- transforming data, 40

scenarios, 15

security

- SQL data sources, 41
- views and, 31

SELECT

- permission, 41
- statements, combining, 35

skip properties, 51

snapshot, staging area, 30

source data

- denormalizing, 36
- multidimensional databases, 15
- transforming, 40

sources. *See* data sources

spreadsheet files, data sources, 15

SQL

- data sources, 15
- SQL 89, 15
- versions supported, 15

SQL data sources

- combining, 30
- combining flat files, 30
- consolidating multiple, 29
- read-only access to, 42
- required permission for, 41
- security, 41
- staging area, 30

SQL Drill-Through. *See* drill-through reports

staging area

- advantages of, 30
- deciding whether to create, 30
- defined, 30
- diagram, 30

standard hierarchies, 24

star schemas. *See* OLAP models

strategic planning, 15

summary source data, associating with time periods, 45

T

tables

- accessing tables in OLAP Metadata Catalog, 52
- choosing which tables are displayed in OLAP models, 41
- creating to denormalize data, 36
- deciding whether to create new tables, 31
- denormalizing, 36
- joining, 38
- reasons to create, 32
- recursive
 - aliases and, 33
 - defined, 33
 - fully-defined columns for, 33
 - rules for, 33
 - UDAs or aliases, 33
- SAP BW optimizing, 38

tables, capitalization, vi

TBC Metaoutline

parts of, [23](#)

sample, [vi](#)

TBC Model, [vi](#)

TBC_MD, [vi](#)

templates. *See* metaoutlines

text file as a data source, [15](#), [53](#)

time

analysis, preparing data for, [45](#)

associating with measures data, [45](#)

time-related columns, defining in source data, [45](#)

time balance properties, [50](#)

time dimension

defined, [45](#)

use in Analytic Services outline, [45](#)

transformations

defining in SQL data sources, [40](#)

example in data source, [41](#)

indexes and, [40](#)

key columns and, [41](#)

transposing columns and rows, [35](#)

trends, [12](#)

two pass calculation properties, [49](#)

U

UDAs, creating in recursive tables, [33](#)

unbalanced hierarchies, [23](#)

unions, removing, [35](#)

user needs, defining, [28](#)

user privileges. *See* permissions

user-defined dimensions, defined, [20](#)

user-defined tables, [31](#)

users, Analytic Services, [14](#)

V

values. *See* measures

variables, defined, [28](#)

variance reporting properties, [51](#)

views

deciding whether to create new views, [31](#)

reasons to create, [32](#)

W

what if scenarios, example, [14](#)

WHERE clauses. *See* filters

workflow

Integration Services, [16](#)

Integration Services product family, [18](#)