# Hyperion® Integration Server Desktop®

Release 2.0

## *OLAP Model User's Guide*




Hyperion Solutions Corporation

P/N: H70D09-2000000

U.S. Patent Number: 5,359,724



Hyperion Solutions Corporation
1344 Crossman Avenue
Sunnyvale, CA 94089

Printed in the U.S.A.

# Contents

## Chapter 3: Creating and Working with OLAP Models

## Chapter 4: Creating the Fact Table, Accounts Dimension, and Time Dimension

## Chapter 5: Creating and Working with Dimensions



## Chapter 6: Editing Columns in Dimension Tables and the Fact Table

## Chapter 7: Transforming Columns



## Chapter 8: Joining Dimension Tables



## Chapter 9: Creating and Working with Hierarchies

# Preface

## Purpose

This guide provides the information that you need to build OLAP models. It explains all Hyperion® Integration Server™ OLAP model features and options, and contains the concepts, processes, procedures, formats, tasks, and examples that you need to use the Hyperion Integration Server Desktop software.

## Audience

This guide is for Hyperion Integration Server database administrators who are responsible for installing, implementing, and deploying the system.

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.
- Knowledge of how to create and maintain Open Database Connectivity (ODBC) data sources.
- Knowledge of the data requirements for your business, so you can apply the Hyperion Integration Server 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. See Microsoft Windows documentation for more information on these terms.
- Experience with the setup and operation of Hyperion® Essbase® OLAP Server.
- A basic understanding of multidimensional concepts.

# Document Structure

This document contains the following information:

Chapter 1, "About Hyperion Integration Server," introduces basic OLAP and multidimensional concepts and describes how to use the desktop to create OLAP models.

Chapter 2, "Preparing Relational Data Sources," describes how to prepare relational data for transfer to a Hyperion Essbase database.

Chapter 3, "Creating and Working with OLAP Models," describes how to open an OLAP model, connect to OLAP Metadata Catalog and to a relational data source, and create, save, delete, and print an OLAP model.

Chapter 4, "Creating the Fact Table, Accounts Dimension, and Time Dimension," describes how to create a fact table, an accounts dimension, and a time dimension from one or more tables in the relational data source.

Chapter 5, "Creating and Working with Dimensions," describes how to create and edit dimensions.

Chapter 6, "Editing Columns in Dimension Tables and the Fact Table," describes how to view and change column properties.

Chapter 7, "Transforming Columns," describes how to create columns, change column properties, and transform columns.

Chapter 8, "Joining Dimension Tables," describes how to create, view, modify, and delete joins to dimension tables.

Chapter 9, "Creating and Working with Hierarchies," describes how to create and edit hierarchies, from defining hierarchy filters to sorting and transforming data in a hierarchy.

Appendix A, "Creating a Sample OLAP Model with Attribute Dimensions," provides procedures for using the sample TBC application to create an OLAP model that contains attribute dimensions.

Glossary, lists and defines key Hyperion Integration Server terms.

Index, lists and defines Hyperion Integration Server 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 two sample databases provided with Hyperion Integration Server. The databases are called TBC (external data source) and TBC_MD (OLAP Metadata Catalog). The TBC_MD OLAP Metadata Catalog 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 applications depends on the relational database management system (RDBMS) that you use.

The person who installs Hyperion Integration Server is responsible for making the sample databases, the OLAP model, and the metaoutline available to end users. If any of the following problems occur when you launch Hyperion Integration Server Desktop, contact the Hyperion Integration Server administrator:

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

# Related Documentation

Hyperion provides the following documentation for this product:

Hyperion Integration Server documents:

- *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide,* for information about basic multidimensional concepts and how to design, create, and maintain one or more metaoutlines based on an OLAP model.
- *Hyperion Integration Server System Administrator's Guide*, for information about how to use OLAP Integration Server, OLAP Metadata Catalog, and OLAP Command Interface, and about how to create and manage ODBC connections.
- *Hyperion Integration Server Installation Guide,* for information about installing the software, about Hyperion Integration Server directories and files, and about connecting relational data sources to an OLAP Metadata Catalog.

Hyperion Essbase OLAP Server documents:

- *Hyperion Essbase Database OLAP Server Administrator's Guide,* for information about basic multidimensional concepts and descriptions of how to design, create, and maintain a Hyperion Essbase database.
- *Hyperion Essbase SQL Interface Guide,* for information about the data sources supported through Hyperion Essbase SQL Interface.
- *Hyperion Essbase Spreadsheet Add-in User's Guide for Excel,* for information about how to use Hyperion Essbase Spreadsheet Add-in with Microsoft Excel.
- *Hyperion Essbase Spreadsheet Add-in User's Guide for Lotus 1-2-3,* for information about how to use Hyperion Essbase Spreadsheet Add-in with Lotus 1-2-3 for Windows.
- The online *Quick Technical Reference* in the `DOCS` directory for information about Hyperion Essbase calculation and report commands and functions.

# Online Help

➤ To access online help:

1. In a dialog box, click Help.
2. In the help dialog box, to display information about an item, click the item.

➤ To print an online help topic, display the topic and select File > Print.

# Online Guides

The online guides are electronic versions of the printed documentation.

➤ To display an online guide, select Help > Help Topics.

# 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. |
| Brackets [] | In examples, brackets indicate that the enclosed elements are optional. |
| **Bold** | Bold text indicates words or characters that you type exactly as they appear on the page. Bold in procedural steps highlights major interface elements. |
| CAPITAL LETTERS | Capital letters denote commands and various IDs. (Example: LOADALL command) |
| `Example text` | Courier font indicates that the material shown is a code or syntax example. |
| Ctrl + 0 | Keystroke combinations shown with the plus symbol (+) indicate that you should press the first key and hold it while you press the next key. Do not type the + symbol. |
| *`Courier italics`* | Courier italic text indicates a variable field in command syntax. Substitute a value in place of the variable shown in Courier italics. |
| *Italics* | Italics in a product-related term indicates that the term is also included in the glossary of the book. |
| 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. |

*Table i: Conventions Used in This Document (Continued)*

| Item | Meaning |
|---|---|
| Menu options | Options in menus are shown in the following format:<br>*Menu name* > *Menu command* > *Extended menu command*<br>For example:<br>Connections > OLAP Metadata Catalog > Connect |
| *n*, *x* | The variable *n* indicates that you must supply a generic number; the variable *x* indicates that you must supply a generic letter. |

**Note:** The term right-click, used throughout this guide, means to click the secondary mouse button to open a pop-up menu.

Chapter 1

# About Hyperion Integration Server

Hyperion Integration Server provides a suite of graphical tools to create OLAP models and metaoutlines. You use tables in a relational 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 a Hyperion Essbase database outline.

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

This chapter contains the following topics:

# 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. Hyperion Essbase OLAP Server supports a multidimensional, multi-user database that enterprise users can access using standard retrieval tools, such as spreadsheets.

Hyperion Essbase OLAP 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 Hyperion Integration Server to create a Hyperion Essbase database that enables users to answer these types of questions quickly and easily.

You can use Hyperion Integration Server Desktop to create a logical data model that represents the *relational database* that contains the data that you want to analyze. Thus, you take the first step in advancing from a relational database to a Hyperion Essbase 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, Hyperion Essbase stores `20` at the intersection of Product, East, Quarter1, and Unit Sales.

*Dimensions* represent the core components of a business plan and often relate to business functions. Product, Region, and Year are typical dimensions. In most databases, dimensions are static, rarely changing over the life of the application.

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 an unlimited 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 Hyperion Essbase 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.

## User Interactions with Data

Hyperion Essbase OLAP Server consolidates and calculates data to provide different views of the data. With a multidimensional database, users can do all of the following:

### Consolidate (aggregate or roll up) data

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:

```
Year = Quarter1 + Quarter2 + Quarter3 + 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 Hyperion Essbase, 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 Hyperion Essbase, 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.

### Spreadsheet operations

Drilling down or drilling up on data retrieves progressively more detailed or progressively more generalized data relative to a selected dimension. Drilling down on a 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 East region.

Pivoting data alters the data perspective. When Hyperion Essbase 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 *Hyperion Essbase Spreadsheet Add-in User's Guide for Excel* or the *Hyperion Essbase Spreadsheet Add-in User's Guide for Lotus 1-2-3*.

## Sources of Data

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

- 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 an OLTP database must be up to date and easy to change and must use as little space as possible. The data is stored in rows and columns in relational database tables.

Hyperion Integration Server enables you to access data for an OLAP model from a data source that is configured for Open Database Connectivity (ODBC) and that supports SQL 89 or later.

# About Hyperion Integration Server

Hyperion Integration Server transfers data from relational tables to a Hyperion Essbase database quickly. 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.

As illustrated in Figure 1-1, Hyperion Integration Server provides graphical tools to help you to do the following tasks:

- Use the tables, views, and columns in a relational database 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 a Hyperion Essbase outline.
- Use the metaoutline to create and populate a Hyperion Essbase database.

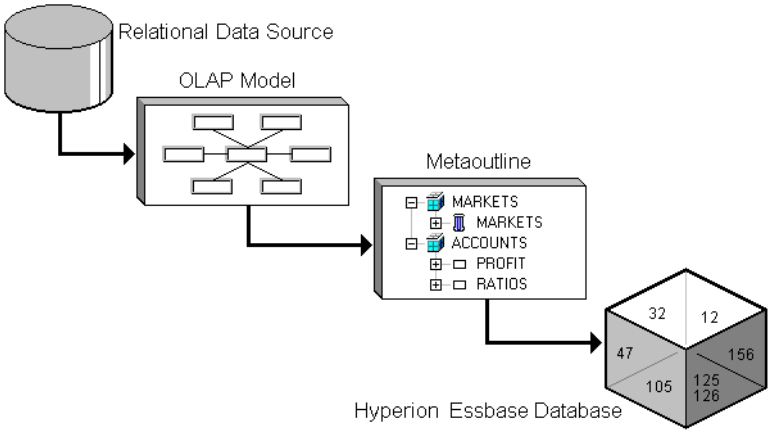



*Figure 1-1: Workflow for Creating a Hyperion Essbase Database from a Relational Data Source*

## Components of Hyperion Integration Server

Hyperion Integration Server, pictured in Figure 1-2, consists of two major components: Hyperion Integration Server Desktop and OLAP Integration Server.

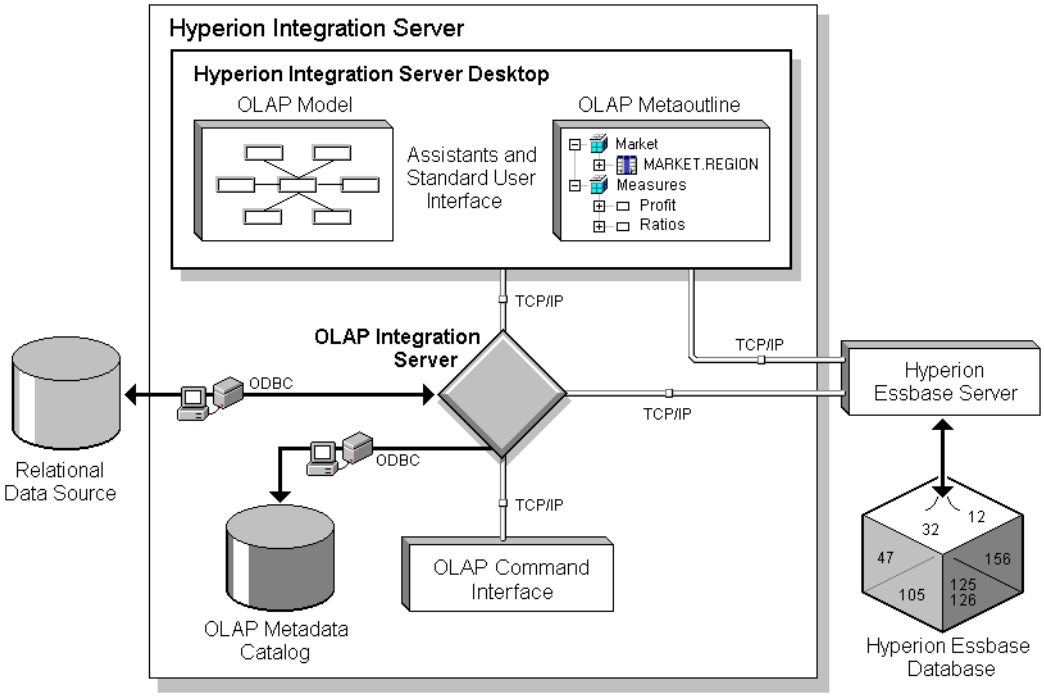



*Figure 1-2: Hyperion Integration Server*

## Hyperion Integration Server Desktop

*Hyperion Integration Server Desktop*, a graphical user interface, is used for creating OLAP models and metatoutlines. Hyperion Integration Server Desktop includes the following subcomponents:

- **OLAP Model Assistant**: OLAP Model Assistant is a wizard for building centralized and reusable OLAP models that are based on a relational data source. The relational data source consists of the tables and views in a single relational database and contains the data that you want to report in Hyperion Essbase.

  Use OLAP Model Assistant to create a basic OLAP model. For more complex model-building functions, such as recursive self-joins, use the standard Hyperion Integration Server Desktop user interface. Use the OLAP models that you create from information in the relational data source to build one or more metaoutlines.

- **OLAP Metaoutline Assistant**: OLAP Metaoutline Assistant is a wizard for creating one or more metaoutlines from an OLAP model. A metaoutline determines the structure and content of the final Hyperion Essbase database outline.

  Use OLAP Metaoutline Assistant to create a basic metaoutline. For more complex metaoutline functions, such as scheduling member loads, use the standard Hyperion Integration Server Desktop user interface. After you create a metaoutline, you can build a Hyperion Essbase database and load data from the relational data source directly into the Hyperion Essbase database.

  **Note:** This user's guide provides detailed procedures for creating and editing an OLAP model using the standard user interface. To create an OLAP model using OLAP Model Assistant, refer to "Working in OLAP Model Assistant" on page 3-7 (or click Overview in the Welcome dialog box), the assistant online help, and the inline help that appears within each assistant tab.

- **Hyperion Integration Server Desktop standard user interface**: The Hyperion Integration Server Desktop standard user interface is a graphical user interface for creating OLAP models and metaoutlines. Use the standard user interface to perform advanced functions, such as performing recursive self-joins in an OLAP model or scheduling member loads for a metaoutline.

To create an OLAP model using the standard user interface, you must connect to OLAP Metadata Catalog and the relational data source. For more information, see Chapter 3, "Creating and Working with OLAP Models."

## OLAP Integration Server

*OLAP Integration Server* is the primary component of Hyperion Integration Server. OLAP Integration Server is software that uses the information stored in OLAP Metadata Catalog to extract from the relational data source the dimension and member names needed to build an associated Hyperion Essbase outline. When the Hyperion Essbase outline is complete, OLAP Integration Server extracts data from the relational data source, performs the operations specified in the associated metaoutline, and loads the data into the Hyperion Essbase database. For more information about OLAP Integration Server, see the *Hyperion Integration Server System Administrator's Guide*.

OLAP Integration Server includes the following subcomponents, as illustrated in Figure 1-2:

- **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 relational data source
  - Metadata describing the information required to generate a Hyperion Essbase outline
  - OLAP models and metaoutlines

    You can create more than one OLAP Metadata Catalog to store OLAP models and metaoutlines. However, you cannot 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 *Hyperion Integration Server Installation Guide* or the ODBC user documentation.
- OLAP Command Interface: *OLAP Command Interface* is a command-line tool used to access OLAP Integration Server to perform operations on the Hyperion Essbase outline and the data in the Hyperion Essbase database. For more information, see the *Hyperion Integration Server System Administrator's Guide*.

## Workflow for Using Hyperion Integration Server

Figure 1-2 provides an overview of the Hyperion Integration Server components that you use to prepare the relational data for OLAP reporting through Hyperion Essbase OLAP Server.

To create a Hyperion Essbase database from a relational data source:

1. Build an OLAP model that is based on the tables in the relational data source. OLAP 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. OLAP Integration Server stores the metaoutline in OLAP Metadata Catalog.
3. Load members and data into the Hyperion Essbase database.
4. Update the Hyperion Essbase database with new members and data.

# About OLAP Models

An OLAP model contains a star schema. OLAP models are based on the idea that values in a relational database can be categorized as either facts or dimensions of facts. Facts 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 contain time and accounts dimensions.

## About Hyperion Integration Server OLAP Models

Unlike other integration products, Hyperion Integration Server 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 tables in the relational database and that you want to report in Hyperion Essbase.

The sample OLAP model included with Hyperion Integration Server, TBC Model, contains the components shown in Figure 1-3.

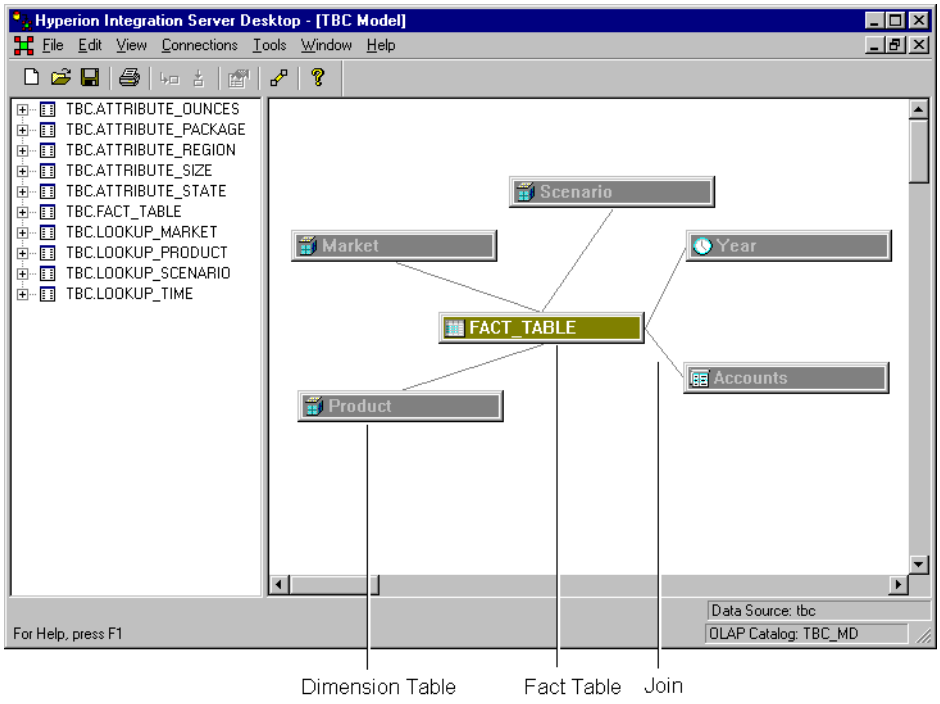



*Figure 1-3: Sample OLAP Model*

## About OLAP Models and Metaoutlines

Use an OLAP model to create one or more metaoutlines. A metaoutline contains the basic structure required to build a Hyperion Essbase outline and to load data into Hyperion Essbase. 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. However, each metaoutline 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 Hyperion Essbase database outline from changes in the relational database.
- They enable you to create hierarchies to structure and summarize data from a relational 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 the OLAP model shown in Figure 1-3, the fact table consists of the FACT_TABLE relational table, a table that contains sales figures, cost of goods sold figures, opening and ending inventory quantities, and other variable measures data columns.

## About Dimension Tables

*Dimension tables,* such as the Market and Product dimension tables shown in Figure 1-3, serve as containers 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, PRODUCT_GROUP_ID, PRODUCT_DESC (description), and PRODUCT_GROUP_DESC (product group description) are related to the sales figures and inventory quantities in the fact 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 that you want to report in Hyperion Essbase. For more information on Hyperion Essbase dimensions, see "About Multidimensional Databases" on page 1-3.

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 Hyperion Essbase outline that you create when you perform a member or data load.

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 relational 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 may not be stored in the relational data source. For more information on user-defined dimensions, see the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*.

## Workflow for Creating an OLAP Model

Before you can create an OLAP model, you must have the following in place:

- A running and properly prepared external data source
- The OLAP Metadata Catalog where you want to store the model that you create
- A server that is running OLAP Integration Server

  See the *Hyperion Integration Server System Administrator's Guide*.

For more information on making the appropriate connections, see "Working with OLAP Models" on page 3-3.

After you make the necessary connections, use the following workflow as illustrated in Figure 1-4 to create an OLAP model:

1. Start Hyperion Integration Server Desktop and use OLAP Model Assistant or the standard user interface to create an OLAP model.

   See "Starting Hyperion Integration Server Desktop" on page 3-3 and "Creating or Editing OLAP Models" on page 3-4.

2. View a list of source tables in the relational database and select the ones that you want to use in the OLAP model.

   See "Working in the OLAP Model Standard User Interface" on page 3-11.

3. To create an OLAP model that contains a fact table; dimension tables; dimension branches; and optional time, accounts, and standard dimensions, drag source tables from the left frame to the right frame.

   See Chapter 4, "Creating the Fact Table, Accounts Dimension, and Time Dimension" and Chapter 5, "Creating and Working with Dimensions."

4. If required, edit existing dimensions and dimension tables by moving, renaming, or deleting them.

   See Chapter 5, “Creating and Working with Dimensions.”

5. If required, edit existing columns by creating, transforming, or changing properties, or by deleting the columns.

   See Chapter 6, “Editing Columns in Dimension Tables and the Fact Table” and Chapter 7, “Transforming Columns.”

6. As necessary, perform joins, including joining each dimension table to the fact table and joining each additional dimension table to a primary dimension table, and, for recursive dimensions, joining a dimension table to itself. If tables are properly joined in the data source, you may not need to specify the joins again.

   See Chapter 8, “Joining Dimension Tables.”

7. As necessary, organize dimension table columns into hierarchies to define how Hyperion Essbase consolidates dimensions. When you create a hierarchy, you can filter, sort, and transform the data in the hierarchy.

   See Chapter 9, “Creating and Working with Hierarchies.”

8. Validate and save the OLAP model.

   See “Validating OLAP Models Manually” on page 3-18.

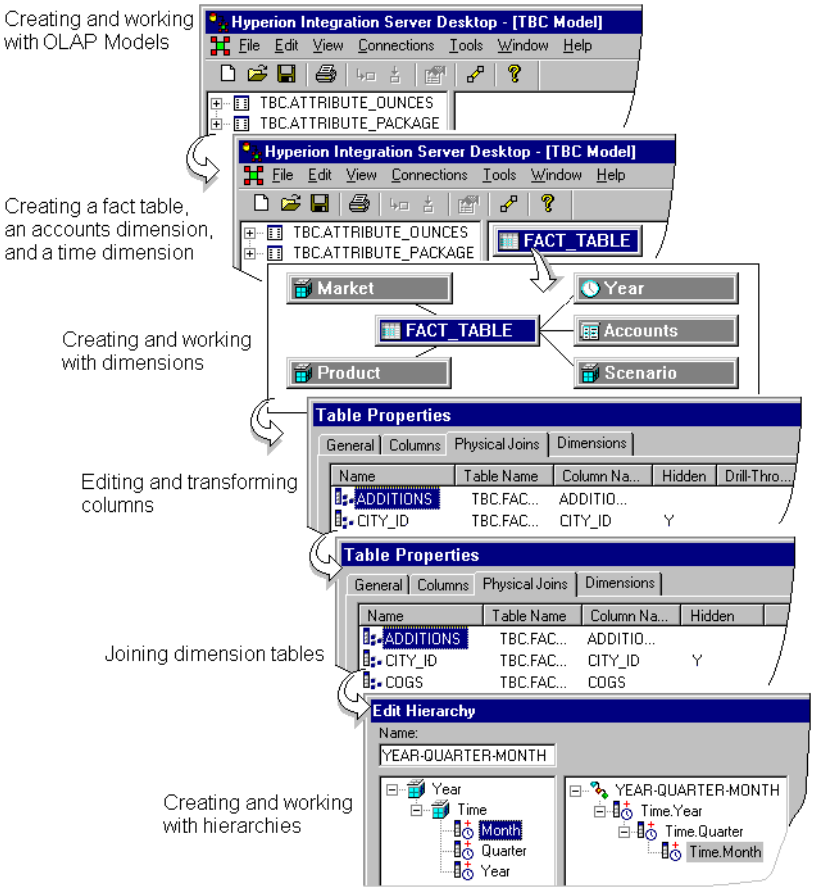



*Figure 1-4: OLAP Model Workflow*

When you are finished, you have a completed OLAP model stored in an OLAP Metadata Catalog. You use the OLAP model to create a metaoutline, as described in the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*.

Chapter 2

# Preparing Relational Data Sources

An OLAP model is a dimensional model of relational data in the form of a star schema. Before you can create an effective OLAP model, you need to understand and define the business needs and data sources that you have available. You may need to modify some data sources to make the transition from relational to multidimensional as easy and efficient as possible.

For information about creating an OLAP model, see Chapter 3, "Creating and Working with OLAP Models."

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

# Defining User Needs

Before you start to design an OLAP model, consider the questions in this topic. By carefully defining 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 Hyperion Essbase database. This topic assumes that you are familiar with the design principles for a multidimensional database. For more information, see "About OLAP" on page 1-2 and the *Hyperion Essbase Database OLAP Server Administrator's Guide*.

**What information (data) do users want to see in the Hyperion Essbase 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 Hyperion Essbase databases.

**What is the maximum level of detail of data that users want to see in the Hyperion Essbase database?**

Select the level of detail by combining an understanding of the business with an understanding of the performance requirements of the Hyperion Essbase database. (In general, the less detail stored in the Hyperion Essbase 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.

**What dimensions apply to each fact table row?**

A dimension is a data category. Typical dimensions are Product, Geography, and Time. Base the selection of dimensions on an understanding of how users want to view data. Each dimension has a number of members. For example, the Geography 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.

**What measures do you want to represent in the fact table?**

*Measures* are numeric quantities that vary over time. Examples of measures are quantity sold, cost of goods sold, and profit. Not all numeric values are measures. 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 measures dimension because users do not want to compare product size over time.

# Defining Data Sources

Now that you have defined what data users want to see in the Hyperion Essbase 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 2-6.
- 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 relational database management system (RDBMS). Consider creating a staging area to perform this task. See "Deciding Whether to Create a Staging Area" on page 2-4.
- Is the data in a single structured query language (SQL) database?
- Is the data in multiple SQL data sources? OLAP Integration Server does not access multiple SQL sources; you must consolidate SQL tables into a single SQL data source in the chosen RDBMS.
- Is the data in flat files? OLAP Integration Server does not access flat files; you need to create flat files as tables in the chosen RDBMS. Consider creating a staging area to perform this task. See "Deciding Whether to Create a Staging Area" on page 2-4. Alternatively, after you build the Hyperion Essbase database, you can use Hyperion Essbase Application Manager to load data. For more information, see the *Hyperion Essbase Database OLAP Server Administrator's Guide*.

# Consolidating Data into a Single Data Source

When you create an OLAP model, all the data that you need must be in a single SQL data source.

If all data is not in a single data source, you need to take the following steps:

1. Select a target RDBMS to connect to the server running OLAP Integration Server. For a list of supported RDBMSs, see the *Hyperion Integration Server Installation Guide*.
2. Create any data that does not exist as an SQL data source; for example, create relational tables in the chosen RDBMS for any information that is currently in flat files.
3. Consolidate the data source tables into a single database in the chosen RDBMS.

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 1–2 GB) in comparison to a data warehouse or online transaction processing (OLTP) application. It is a snapshot of data. In other words, it is not constantly updated with new data but is refreshed periodically from source data.

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

- Combine data from heterogeneous platforms without changing the original data in the RDBMS. 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 RDBMS data and store the calculated data in a staging area. You can then run faster queries on the pre-calculated 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 and Tables" on page 2-6).

- Transform data that is not consistently described (see "Transforming Data" on page 2-14).
- Prepare data in the staging area for issues related to the year 2000. Year-2000 issues may occur if the RDBMS stores the year as two digits (for example, 99) and not four digits (for example, 1999).

The disadvantage of using a staging area is that the data in the staging area is a snapshot and may not represent the current data.

Figure 2-1 shows a staging area as part of a data warehouse. Data is copied into the staging area from the RDBMS source data and converted from flat files and other RDBMS sources.

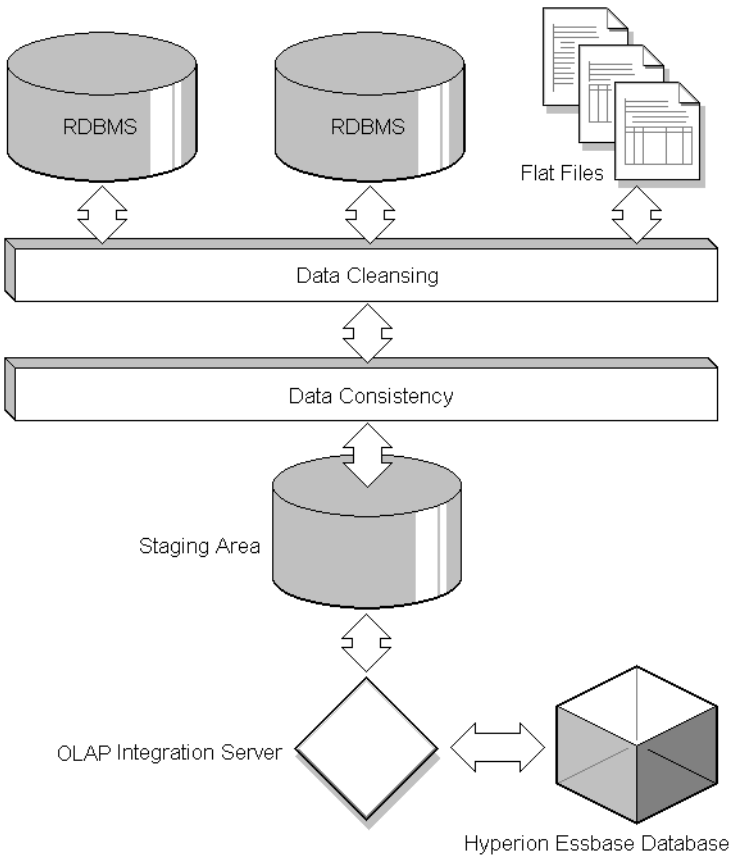



*Figure 2-1: Data Warehouse with Staging Area*

# Cleansing Data

OLAP 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 need to fix, or cleanse, the data. Cleansing data may 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 and Tables

OLAP Integration Server does not distinguish between the tables and the views of an RDBMS; you can use either tables or views to create an OLAP model. You may want to create views or tables specifically for use with OLAP Integration Server to provide security and convenience and to make the transition from relational to multidimensional as efficient as possible.

Consider creating views or new tables if you want do either of the following steps:

- If you want to build a Hyperion Essbase hierarchy down to a specific level in a recursive table, see "Building Hyperion Essbase Hierarchies from Recursive Tables" on page 2-8.
- If you want to create aliases in the Hyperion Essbase database from data stored in multiple columns in the source data tables, see "Creating Aliases for Dimension and Member Names" on page 2-19.

Also consider creating views or tables if the source tables meet any of the following criteria:

- The source tables contain unions. OLAP Integration Server does not generate SQL for unions. See "Removing Unions" on page 2-10.
- The source tables have columns that you want to transpose to rows. See "Transposing Columns and Rows" on page 2-10.

- 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 table names may not be meaningful in business terms. You probably need to ask an application specialist to create the required tables and views (with meaningful names) in a staging area in the target RDBMS. See "Deciding Whether to Create a Staging Area" on page 2-4.
- 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 a Hyperion Essbase database. See "Denormalizing Source Data Tables" on page 2-12.

## Deciding Whether to Create a View or a New Table

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

When deciding whether to create a view and or whether to create 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 2-4.
- Do you want to index columns that are not indexed in the source RDBMS table? You need to create a new table because you cannot index columns in a view. See "Creating Indexes" on page 2-14.
- Do you want to index columns that contain data that you need to transform? Because many RDBMSs ignore indexes on columns that have transformations, you probably need to create a new table. See "Transforming Data" on page 2-14.

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 2-4.

## Building Hyperion Essbase Hierarchies from Recursive Tables

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

When creating a Hyperion Essbase hierarchy from a recursive table, use the following guidelines:

- To associate aliases or user-defined attributes (UDAs) with members created from a recursive table, ensure that the column with which you will associate the alias or UDA, is fully defined. See "Creating Aliases or UDAs for Members in a Recursive Table" on page 2-8.
- When creating an OLAP model, join the recursive table to itself. For more information and for an example of a recursive source table, see "About Recursive Tables" on page 9-4 and "Defining a Recursive Table" on page 8-15.
- When creating a metaoutline, select the parent or child column that you want to filter on as a member level in the metaoutline. See the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*.

## Building a Hierarchy Down to a Specific Level

To build the Hyperion Essbase outline down to a specific level, you need to 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 a Hyperion Essbase 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 need to 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 need to 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 2-1 and Table 2-2). For more information on the specific steps, see "Preparing Recursive Tables to Be Used with Aliases and User-Defined Attributes" on page 8-16.

**If the alias or UDA data is in the recursive source table,** you need to 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 Hyperion Essbase hierarchy). Thus the parent column contains the lowest-level value in the hierarchy with a NULL value in the child column. In Table 2-1, 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 2-1: Fully Defined Parent Column*

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

**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 Hyperion Essbase hierarchy). Thus the child column contains the highest-level value in the hierarchy, with a NULL value in the parent column.

In Table 2-2, 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 2-2: Fully Defined Child Column*

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

For more information on the specific steps that you need to perform, see "Preparing Recursive Tables to Be Used with Aliases and User-Defined Attributes" on page 8-16.

## Removing Unions

OLAP 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 need to create a view (or views) of the data that does not use the unions before you can start to work with the data in OLAP Integration Server. For more information on unions, see the RDBMS documentation.

## Transposing Columns and Rows

If you need to transpose columns and rows so that you can more easily transition data from a data source to a Hyperion Essbase database, transpose the columns before you start to work with the data in OLAP Integration Server. For example, if you want to create multiple Hyperion Essbase 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 Hyperion Essbase measures (Init_Sales, Subsequent_Sales, and Return_Sales) from a single database column (SALESTYPE).

The SALESACTUALS table contains the following columns:

*Table 2-3: SALESACTUALS Table*

| PRODID | MKTID | TIMEID | MONTH | 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 Hyperion Essbase outline in the hierarchy illustrated in Figure 2-2:

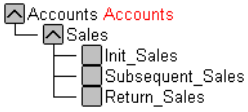



*Figure 2-2: Accounts Hierarchy*

You want each SALESTYPE value to form the Hyperion Essbase 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 2-4: 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 2-4:

```
)
   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 RDBMS 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 (see Chapter 8, "Joining Dimension Tables"), it may be more efficient to create a new table of denormalized data in the RDBMS.

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

*Table 2-5: Normalized Product Family Data*

| FAMILYID | FAMILYDESC |
|---|---|
| 100 | Colas |
| 200 | Root Beer |

*Table 2-6: Normalized Product Data*

| PRODID | FAMILYID |
|---|---|
| 100-10 | 100 |
| 100-20 | 100 |

*Table 2-7: Normalized Product Description Data*

| PRODID | PRODDESC |
|---|---|
| 100-10 | Cola |
| 100-20 | Diet Cola |

The following table shows the denormalized data in one table:

*Table 2-8: Denormalized Product Data*

| FAMILYID | FAMILYDESC | PRODID | PRODDESC |
|---|---|---|---|
| 100 | Colas | 100-10 | Cola |
| 100 | Colas | 100-20 | Diet Cola |
| 100 | Colas | 100-30 | Caffeine Free Cola |
| 100 | Colas | 100-10 | Cola |

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

# 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 2-17.
- Join tables within each dimension branch. See "Selecting Tables for Dimensions" on page 2-18.
- Join dimension tables to the tables in the fact table.

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

# Creating Indexes

OLAP Integration Server detects indexes (including bitmapped indexes) that you have defined on the source tables and uses them to create a Hyperion Essbase 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.

Consider taking the following actions to improve performance:

- Define indexes on columns that you use to filter data in the RDBMS source 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 columns Region and 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 *Hyperion Essbase Database OLAP Server Administrator's Guide*.
- Define bitmapped indexes on numeric data that you use to filter the database. For example, if you filter on sales values (for example, `SELECT Product FROM ProdSales GROUPBY Product ORDERBY ProdId HAVING SUM(Sales)>15000`), then consider defining a bitmapped index on sales values. Most RDBMSs support bitmapped indexes. For more information on bitmapped indexes, see the documentation for the RDBMS 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. 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 2-4) 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, see Chapter 7, "Transforming Columns," and the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*. However, if you need to do significant transformations on the data, you may want to use a data transformation tool before you use the data in OLAP Integration Server.

Transformations that you need to perform in the RDBMS, and not in Hyperion Integration Server, include the following items:

- The most frequently accessed transformations

  For example, assume that you want the Hyperion Essbase database outline to have 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 RDBMSs ignore indexes on columns that have transformations. You 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 OLAP Integration Server. For example, if the source table has a PRODID column containing product identification information, transform the data in the RDBMS so that it maps to the data categories that you want to display in the Hyperion Essbase outline. In Figure 2-3, for example, the categories are CUSTID, STATE and PRODNUM.

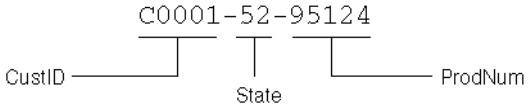



*Figure 2-3: Transforming a Product Code Stored in a PRODID Column*

# Deciding Which Tables Are Available to OLAP Model Creators

When you connect to the source database, OLAP Integration Server displays in the left frame all tables to which the RDBMS user ID provides you access (SELECT permission).

For all RDBMSs, if you have partial access to a table, OLAP 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.) However, in the OLAP model, do not use a column to which you do not have SELECT permission. OLAP Integration Server cannot filter, data load, or member load a column for which you do not have SELECT permission.

**Note:** For DB2, OLAP Integration Server displays all tables that are in the database, irrespective of whether you have access to them. 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.

For added security and ease of administration, you may want to define which tables the users can see by doing 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 providing 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 that you use.

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

## 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. For more information, see "About the Fact Table" on page 1-12. The type of Hyperion Essbase database that you want to create determines which source tables you need to include in the fact table. See "Defining User Needs" on page 2-2.

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 Hyperion Essbase database.

Select a table or tables that contain all of the measures data that Hyperion Essbase end users want to see. For example, if the relational database contains the following ORDERS and ORDERDETAILS tables, you need to include both tables if you want the fact table to include data from both tables.

*Table 2-9: 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 of the dimensions (data categories) on which the Hyperion Essbase 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 2-10: Sales Table for the Fact Table*

| | **PRODID** | **SALES** | **DATESOLD** | **CITY** | **CHANNEL** | **UNTSSOLD** | **...** | **...** |
|---|---|---|---|---|---|---|---|---|
| **Sales** | | | | | | | | : |

**Note:** You can improve performance by setting up primary and foreign keys and joining the tables that form the fact table. See "Joining Tables" on page 2-13.

## Selecting Tables for Dimensions

A dimension is a data category. 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 Hyperion Essbase dimension. A dimension table can join to other dimension tables, forming a *dimension branch*.

The data that you want to be available for analysis in the Hyperion Essbase application that you create determines which source tables you use to create dimensions. See "Defining User Needs" on page 2-2. 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 2-11: 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 those products that have been sold. The PRODUCTS table contains data for all products. Thus, Hyperion Essbase end users can use the PRODUCTS table information to analyze products that are or are not selling.

## Creating Aliases for Dimension and Member Names

*Aliases* are alternative names for the members and dimensions of a Hyperion Essbase outline. An alias is often a more easily identifiable product name or product description for a column in a relational data source. For example, in the Hyperion Essbase Sample Basic database, the 200-20 member in the Product dimension has the alias Diet Root Beer.

If the source data contains information that you want to use to create aliases in a Hyperion Essbase outline, include the relevant data as a column in the associated OLAP model.

When you create a metaoutline, you can retrieve alias information from only one column in the OLAP model. If the alias information in the source data is in more than one column, complete one of the following data preparation procedures:

- In the RDBMS, 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, see Chapter 7, "Transforming Columns."
- In the OLAP model, use the pass-through feature to run a procedure that uses RDBMS 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 RDBMS to retrieve the alias information for that field from a different column.

  For more information, see "Defining Pass-Through Transformations" on page 7-14. For more information on procedures and CASE logic, see the RDBMS documentation.

**Note:** You can associate one alias table for each Hyperion Essbase database that you create using Hyperion Integration Server.

# Creating Time and Accounts Dimensions

In an OLAP model, you can create time and accounts dimensions that carry over directly to the Hyperion Essbase 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.

## 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 Hyperion Essbase 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 measures 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 Measures Data

To enable OLAP 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 2-12: 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

OLAP Integration Server works with three *data types*: string, numeric, and datetime. "About Column Data Types" on page 6-3 shows how the basic Open Database Connectivity (ODBC) data types map to string, numeric, 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, OLAP 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 2-13: 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, OLAP Integration Server requires that all time identifiers be in one column and that accounts identifiers be in separate columns, as shown in Table 2-14. If, for example, data is stored with each month as an individual column, as shown in Table 2-13, you should restructure the data. Individual months should be data values in a MONTH column, and the measure categories, such as SALES and COGS, should be separate columns.

*Table 2-14: 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 an RDBMS to transpose the columns (see "Transposing Columns and Rows" on page 2-10).

# Accessing Tables in OLAP Metadata Catalog

If you log on with a particular user ID when you create an 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 *Hyperion Integration Server Installation Guide*.

You must perform a different procedure when time measures are stored as columns that include accounts information, as described in "Switching the Time and Accounts Dimensions" on page 4-14.

Chapter 3

# Creating and Working with OLAP Models

This chapter describes how to create and work with OLAP models—from opening or creating an OLAP model through validating and saving an OLAP model. Before you read this chapter, read Chapter 1, "About Hyperion Integration Server" to familiarize yourself with the definition and components of an OLAP model.

This chapter contains the following topics:

# Understanding the OLAP Model Workflow

Figure 3-1 illustrates the workflow for creating an OLAP model. This workflow includes creating and working with an OLAP model; creating a fact table, an accounts dimension, and a time dimension; creating and working with dimensions; editing and transforming the columns of dimension tables; joining dimension tables; and creating and working with hierarchies. This chapter focuses on how to create an OLAP model and manage existing OLAP models.

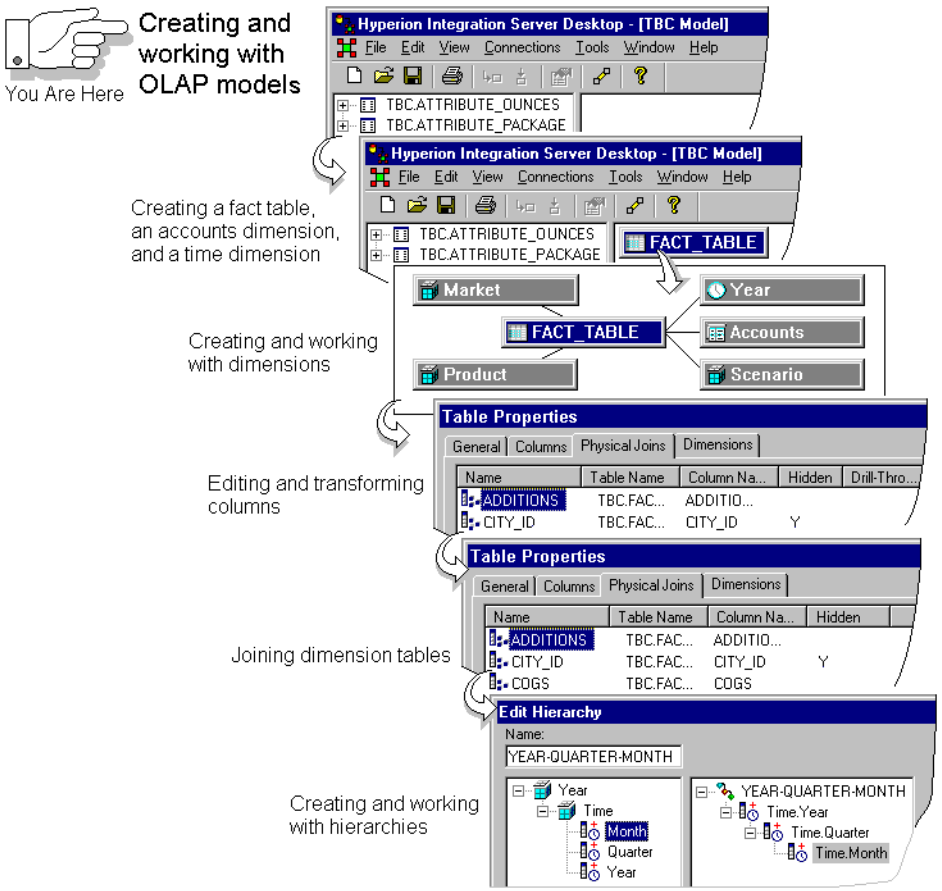



*Figure 3-1: OLAP Model Workflow*

# Working with OLAP Models

Before you start working with an OLAP model, make sure that the OLAP Metadata Catalog where your OLAP model will be stored, the server on which OLAP Integration Server is running, and the data source are accessible. These items can be on servers on a network; they do not need to be on your local computer to be available to Hyperion Integration Server Desktop. For more information, see the *Hyperion Integration Server System Administrator's Guide*.

To work with an OLAP model, start Hyperion Integration Server Desktop and choose whether to use OLAP Model Assistant (to be guided through creating or editing an OLAP model), or to use the OLAP Model standard user interface (to create or edit an OLAP model without guidance). For more information, see "Starting Hyperion Integration Server Desktop" on page 3-3 and "Creating or Editing OLAP Models" on page 3-4.

## Starting Hyperion Integration Server Desktop

To start Hyperion Integration Server Desktop, use any of the following methods:

- From the Windows Start menu, select Hyperion Integration Server Desktop.
- Double-click the Hyperion Integration Server Desktop icon,  .
- On the command line, type the executable name (`olapbldr`).

  For example, if Hyperion Integration Server Desktop is installed in the `\hyperion\is\bin` directory, type the following path:

  **`\hyperion\is\bin\olapbldr`**

**Note:** The default client directory, which is specified during installation is `\hyperion\is`. Your application may have a different installation directory. Consult the person who installed Hyperion Integration Server at your site if you are not sure of the directory specification.

Now you can start working on an OLAP model. See "Creating or Editing OLAP Models" on page 3-4.

## Creating or Editing OLAP Models

To create or edit OLAP models, start Hyperion Integration Server Desktop and connect to OLAP Integration Server, an OLAP Metadata Catalog, and a data source.

For more information about connecting to OLAP Integration Server, OLAP Metadata Catalog or a data source, see the *Hyperion Integration Server Installation Guide*. If you have problems connecting, see "Correcting Connection Problems" on page 3-6.

If you just started Hyperion Integration Server Desktop, the Welcome dialog box that you use to choose whether to create or edit an OLAP model opens automatically.

If you are already running Hyperion Integration Server Desktop and are connected to OLAP Metadata Catalog, select File > New or File > Open to open the Welcome dialog box.

Using OLAP Model Assistant is the quickest, easiest way to create basic OLAP models or to edit OLAP models. If, however, you want to perform more complex tasks, you must use the OLAP Model standard user interface. For more information, see "Tasks That Cannot Be Performed Using OLAP Model Assistant" on page 3-10.

➤ To use OLAP Model Assistant to create a new OLAP model:

1. Select 
.
2. If you want to place write-lock privileges on the OLAP model, select **Exclusive Access**.

   **Exclusive Access** enables you to modify and save the OLAP model. If **Exclusive Access** is not selectable, another user has write-lock privileges and you have read-only or no privileges.
3. Click Open.

➤ To use OLAP Model standard user interface to create a new OLAP model:

1. Select 

.
2. If you want to place write-lock privileges on the OLAP model, select **Exclusive Access**.

   **Exclusive Access** enables you to modify and save the OLAP model. If **Exclusive Access** is not selectable, another user has write-lock privileges and you have read-only or no privileges.
3. Click Open.

➤ To use OLAP Model Assistant to edit an OLAP model:

1. In the **Welcome** dialog box, select the **Existing** or **Recent** tab.
2. Select the OLAP model name. For more information about the OLAP model, look in the **Details** group box.
3. Make sure that **Open with Assistant** is selected.
4. If you want to place write-lock privileges on the OLAP model, select **Exclusive Access**.

   **Exclusive Access** enables you to modify and save the OLAP model. If **Exclusive Access** is not selectable, another user has write-lock privileges and you have read-only or no privileges.
5. Click Open.
6. If necessary, log on to the appropriate data source.

➤ To use OLAP Model standard user interface to edit an OLAP model:

1. In the **Welcome** dialog box, select the **Existing** or **Recent** tab.
2. Select the OLAP model name. For more information about the OLAP model, look in the **Details** group box.
3. Make sure that **Open with Assistant** is not selected.
4. If you want to place write-lock privileges on the OLAP model, select **Exclusive Access**.

   **Exclusive Access** enables you to modify and save the OLAP model. If **Exclusive Access** is not selectable, another user has write-lock privileges and you have read-only or no privileges.

5. Click Open.
6. If necessary, log on to the appropriate data source.

## Correcting Connection Problems

If you have problems connecting from Hyperion Integration Server Desktop to OLAP Integration Server, OLAP Metadata Catalog, Hyperion Essbase server, or the data source, use the following checklist to help diagnose the problem:

- Was the server temporarily down or not available? Try again to connect.
- Did you enter the correct server name? Your network configuration may require a fully-qualified internet host name; for example, `myserver.mycompany.com`.
- Did you enter the correct user name and password? Passwords are usually case-sensitive. Make sure that the Caps Lock key is not activated.
- Do you have a network connection to the server? Check your network connection by using the ping command. See Microsoft Windows online help for more information about using ping.

If the above steps do not resolve the problem, contact your database administrator or system administrator. For more information about troubleshooting server connections, see the *Hyperion Integration Server System Administrator's Guide*.

# Working in OLAP Model Assistant

OLAP Model Assistant is a wizard that steps you through creating or editing an OLAP model quickly and easily. OLAP Model Assistant contains multiple tabs. If you are creating an OLAP model, complete the tabs and their tasks in order.

After you complete the required tasks in all tabs, you have a valid OLAP model that you can use to create one or more metaoutlines. If you are editing an existing OLAP model, work through the tabs in any order to revise the model as needed.

When you open OLAP Model Assistant, it displays the first tab, **Select Fact Table**, as shown in Figure 3-2.

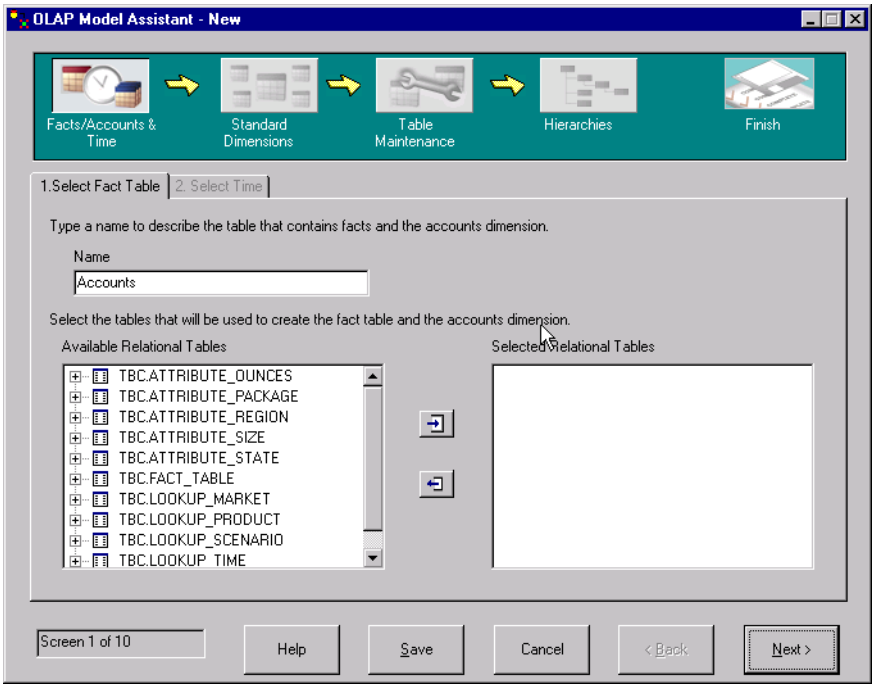



*Figure 3-2: Select Fact Table Tab in OLAP Model Assistant*

## Process Graphics

The process graphics along the top of OLAP Model Assistant represent related tasks. When you create an OLAP model, the tasks must be completed in order from left to right. After you complete all the tabs that a process graphic represents, the next process graphic becomes accessible and you can begin work on the task within that task group. When you edit an existing OLAP model, all process graphics are clearly displayed and all tabs are accessible.

**Note:** For best visibility of the OLAP Model Assistant process buttons, a color palette of 65536 colors is recommended. To set this palette, select Start > Settings > Control Panel. Double-click the Display icon, select 65536 from the Color Palette drop-down list, click Apply, and then click OK. (Your PC may not support a palette of 65536 colors.)

## Inline Help Text

Each tab contains brief descriptions of the tasks that you need to perform. If you need additional information on a task, click the Help button.

## Tasks to Complete

Each tab contains tasks that you must complete to define a particular part of an OLAP model. In the first tab, for example, you must select the table to use as the fact table. For help on using any tab, click the Help button.

OLAP Model Assistant contains the following tabs:

*Table 3-1: OLAP Model Assistant Tabs*

| Tab Name | More Information |
|---|---|
| **Select Fact Table** | "About the Fact Table" on page 4-3 |
| **Select Time** | "About the Time Dimension" on page 4-8 |
| **Name Dimensions** | "Creating Dimensions" on page 5-3 |
| **Select Relational Table** | "Adding Source Tables to Create Dimension Branches" on page 5-6 |
| **Fact Table Joins** | "About Joins" on page 8-3 |
| **Dimension Table Joins** | "About Joins" on page 8-3 |

*Table 3-1: OLAP Model Assistant Tabs (Continued)*

| Tab Name | More Information |
|---|---|
| **Edit Tables** | Chapter 6, "Editing Columns in Dimension Tables and the Fact Table" |
| **Define Hierarchies** | "About Hierarchies" on page 9-3 |
| **Preview Hierarchies** | "Previewing the Hyperion Essbase Outline" on page 9-28 |
| **Finish** | "Saving OLAP Models" on page 3-20 |



## Buttons

The following buttons are always displayed along the bottom of OLAP Model Assistant:

- Help button. Click to view detailed help for the selected tab. The help describes each field in the OLAP Model Assistant tabs and explains what you must do to complete the tasks in the selected tab before moving to the next tab.
- Save button. Click to save the OLAP model. You can save an OLAP model in any tab and then continue to move through the other tabs. Changes that you made to all previous tabs are saved.

  When you save an OLAP model, it is validated automatically. You can save an OLAP model with validation problems, but you cannot use the OLAP model to create metaoutlines.
- Cancel button. Click to close the OLAP model without saving. Changes that you saved in previous tabs are not discarded. Changes in previous tabs that have not been saved are discarded, as are all changes in the current tab.
- Back button. Click to move to the previous tab. You can move back through previous tabs and make changes at any time.
- Next button. Click to move to the next tab. If you are creating an OLAP model and have not completed all mandatory tasks in a tab (such as selecting a fact table), you cannot move to the next tab.

## Tasks That Cannot Be Performed Using OLAP Model Assistant

You cannot use OLAP Model Assistant to perform the following tasks. If you need to perform these tasks in an OLAP model, use the OLAP Model standard user interface.

- Create an OLAP model that contains attribute dimensions.
- Create an OLAP model that contains dimension branches. If you want to add multiple dimension tables to a dimension, you must combine the tables.
- Create a fact table or an accounts dimension from multiple source tables.
- Join a dimension table to itself; that is, perform a recursive self-join on a table.
- Manually validate the OLAP model at any time during the creation or editing process.
- Set up to switch the time and accounts dimensions.
- Rename a dimension table; that is, give a dimension table a name other than the source table name.
- Rename an existing dimension.
- Delete columns or rename columns.
- Create drill-through or user-defined columns.
- Define retrieval optimization preferences for Oracle databases.
- Filter, port, or transform columns in hierarchies.
- View relational source table data.

**Note:** This user's guide provides detailed procedures for creating and editing an OLAP model using the standard user interface. To create an OLAP model using OLAP Model Assistant, refer to the assistant overview information (or click overview in the Welcome dialog box), to the online help, and to the inline help that appears within each assistant tab.

# Working in the OLAP Model Standard User Interface

To create an OLAP model using the standard user interface, you must connect to OLAP Metadata Catalog and the to relational data source. For more information, see "Working with OLAP Models" on page 3-3. After you click the OLAP Model icon, Hyperion Integration Server Desktop displays the OLAP Model main window, as shown in Figure 3-3. The window displays source tables in the left frame and a blank modeling area in the right frame.



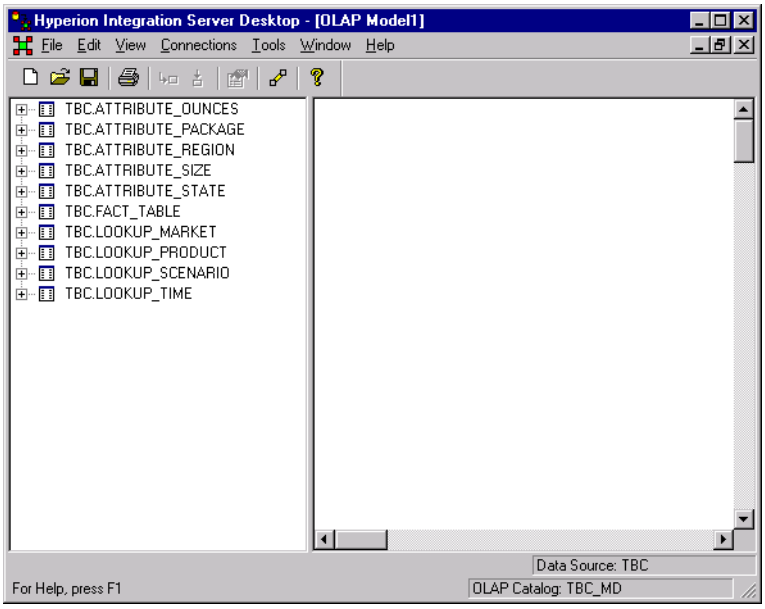



*Figure 3-3: OLAP Model Main Window*

## Left Frame

The *left frame* displays a list of source tables. You can use the listed source tables to create an OLAP model. You can view tables, views, and synonyms in the left frame. Although you can view tables, views, synonyms, or any combination of tables, views, and synonyms in the left frame, this guide refers only to tables. Whenever tables in the left frame are discussed, synonyms and views are implied. To view synonyms, select View > Show Synonyms.

The name of each table in the relational data source is displayed after a plus symbol, ⊞, or minus symbol, ⊟.

- The plus symbol, ⊞, indicates that you can expand the table to see the columns it contains. For example, if MARKET is not expanded, clicking MARKET expands the table to show CITY, CITYID, and so on.
- The minus symbol, ⊟, indicates that you can collapse the table. For example, if MARKET is expanded, clicking MARKET collapses the table to show the table name but not the table columns.

You can view the contents of a table in the left frame in the following ways:

- Select the table and select View > View Table Data.
- Select the table, right-click, and select View Table Data.

For more information, see "Viewing Table Data" on page 3-17.

## Right Frame

The *right frame* is where you build an OLAP model. You drag source tables from the left frame to the right frame to create a fact table, accounts dimension, optional time dimension, standard (default) dimensions, and dimension branches.

When you create an OLAP model, the relational tables that you drag to the right frame become the fact table and the dimension tables. You can add, hide, delete, transform, and perform other operations on columns in these dimension tables to refine the data that you want reported in Hyperion Essbase. After you complete and save an OLAP model, open it to create a metaoutline. At that time, the dimension tables can become dimensions.



## Tools and Toolbars

Use the dual-function Join tool,  , on the toolbar to add joins between tables in an OLAP model or to move objects in an OLAP model. The default position for the Join tool is *Move mode*. To switch between Move mode and Add Joins mode, click the Join tool.

Use the toolbar to perform commonly used functions. For more information on individual toolbar buttons, click Help. To hide the toolbar, deselect View > Toolbar. In the menu, the check mark next to Toolbar is removed, and the toolbar disappears. To redisplay the toolbar, select View > Toolbar.

The toolbar is not attached to the edge of the program window. You can move it by selecting it and dragging it to a new location. You can also change the shape and size of the toolbar.

## Menus

Use the menu bar to perform commonly used commands.

Use the pop-up menu that opens when you right-click an object to perform operations on the selected object.

## Status Bar

The status bar identifies the data source and the OLAP Metadata Catalog to which you are connected and displays other status information. To hide the status bar, deselect View > Status Bar. In the View menu, the check mark next to Status Bar is removed, and the status bar disappears. To redisplay the status bar, select View > Status Bar.

## OLAP Model Dimension Table Graphics

The following graphics identify dimension tables in the OLAP model:


identifies the fact table as the table containing measures values for the OLAP model.


identifies a time dimension.


identifies an accounts dimension.


identifies a standard dimension table.

For illustrations and explanations of the graphics that identify relational data source columns, see “Viewing Column Properties” on page 6-4.

## Display of Column Names

Normally, the tables in the relational data source are displayed in a collapsed view in the left frame of the OLAP Model main window, as shown in Figure 3-4.

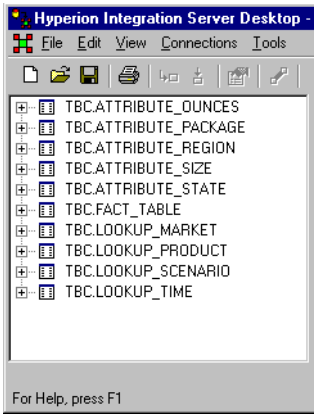



*Figure 3-4: Collapsed Table View in Left Frame*

As you create a fact table or a dimension table, you can display the names of the source columns that the table contains. You can use this information to ensure that you have selected the correct table from the relational data source. Display the contents of the table by selecting the table and then selecting View > View Table Data or by clicking the plus symbol, ⊞, to the left of the table name.

**Note:** When you display columns in the left frame, OLAP Integration Server displays only certain ODBC data types. For a complete list, see "About Column Data Types" on page 6-3.

The following topics describe the methods that you can use to display column names and contents:

- "Displaying Column Names in the Left Frame" on page 3-16 describes how to display table columns as they are displayed in the original relational data source.
- "Viewing Table Data" on page 3-17 describes how to display the data of the selected relational table.
- "Displaying Column Names in the Right Frame" on page 3-18 describes how to display table columns after you have added the tables to an OLAP model.

## Displaying Column Names in the Left Frame

➤ To display the column names of a relational table:

1. Expand the table by clicking the plus sign, ⊞, in front of the table.

   The table expands to include an alphabetical list of columns, as shown in Figure 3-5. A graphic identifies each column as a column, , or as a primary key, . In this example, the TBC.LOOKUP_MARKET table is expanded. You can expand as many tables as you want and scroll through each table to display column and key names.

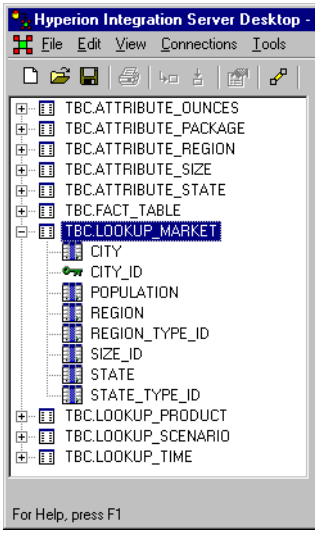



*Figure 3-5: Expanded Columns in the Left Frame*

2. In the left frame, right-click and select Show Views, Show Tables, or Show Synonyms to display views, tables, or synonyms (aliases).

   - A *view* is a logical table created by combining columns from one or more tables in the relational data source (see "Creating Views and Tables" on page 2-6).
   - Synonyms are displayed only for RDBMSs that support synonyms (aliases).

- Although you can display tables, views, synonyms, or any combination of tables, views, or synonyms in the left frame, this guide refers to tables. Whenever tables in the left frame are discussed, synonyms and views are implied.

3. Click the minus sign, ⊟, in front of the table to collapse it.

## Viewing Table Data

➤ To display the contents of a table that is listed in the left frame:

1. Select the table, right-click, and select View Table Data.

   Hyperion Integration Server displays the **View Table Data** dialog box. The **View Table Data** dialog box displays the first 100 rows of the selected source table.

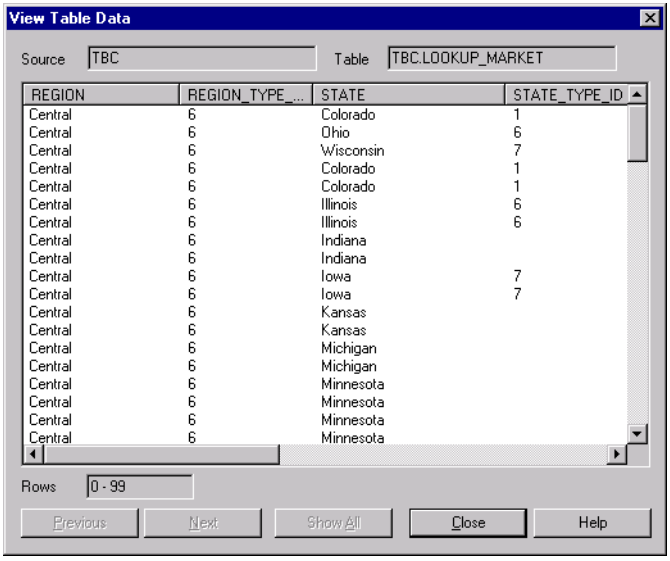



*Figure 3-6: Viewing Table Data*

2. To view the next 100 rows, click Next.
3. To display the previous 100 rows, click Previous.
4. To display all rows in the table, click Show All. Retrieval may take some time, depending on the number of rows the table contains.

### Displaying Column Names in the Right Frame

To display column names after dragging a table to the right frame, use one of the following methods:

- Drag the corner of the dimension table or fact table down until the table is large enough to display its columns.
- Right-click the dimension table or fact table and select View Columns. To collapse the column list, right-click and select View Columns again.
- Select the dimension table or fact table and select View > View Columns. To collapse the column list, Select View > View Columns again.
- To display all columns of all dimension tables (including the fact table) currently in the right frame, select View > View All Columns.

  A scrollable list of column names is displayed for each dimension table, as shown in Figure 3-7.

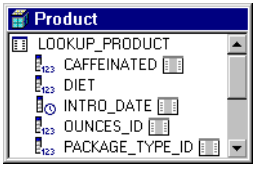



*Figure 3-7: View Columns Option*

For illustrations and explanations of the graphics that identify columns, see "Viewing Column Properties" on page 6-4.

## Validating OLAP Models Manually

When you save an OLAP model, OLAP Integration Server automatically *validates* it and writes it to OLAP Metadata Catalog. If you prefer, you can manually validate an OLAP model before saving it. You can save an OLAP model that does not pass validation, but you cannot use it to create a metaoutline.

To validate an OLAP model manually, select File > Validate. If the OLAP model is valid, a dialog box informs you that the OLAP model contains no errors.

If the OLAP model is invalid, Hyperion Integration Server displays a dialog box, similar to the one in Figure 3-8, that details the errors in the OLAP model.

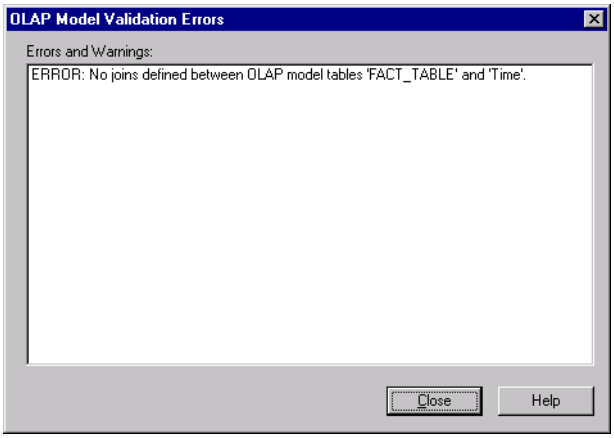



*Figure 3-8: Invalid OLAP Model Validation Error Message Listing*

If the OLAP model is invalid, check to make sure that the following conditions are met:

- The OLAP model contains a fact table. See Chapter 4, "Creating the Fact Table, Accounts Dimension, and Time Dimension."
- The OLAP model contains at least one measures value; for example, sales amounts, quantities sold, or commissions. (Measures can be included in the fact table and do not necessarily need to be in an accounts dimension. If the business is a non-financial application, for example, you do not need to create an accounts dimension.) See Chapter 4, "Creating the Fact Table, Accounts Dimension, and Time Dimension."
- The OLAP model contains at least one dimension.
- Each dimension table in the OLAP model is joined to the fact table or to another dimension table.
- If the OLAP model contains a single dimension, the dimension contains at least one column.

- Dimensions in the OLAP model that are made up of two or more physical tables have joins defined. See Chapter 8, "Joining Dimension Tables."
- The OLAP model contains no dimensions with duplicate names.

To save a model that does not pass validation, see "Saving an OLAP Model Manually" on page 3-22.

# Saving OLAP Models

When you save an OLAP model, Hyperion Integration Server automatically validates it and writes it to OLAP Metadata Catalog. You can save an OLAP model if it does not pass validation, but you cannot use an invalid OLAP model to create a metaoutline.

This topic describes how to save an OLAP model. This topic contains the following subtopics:

## Understanding Exclusive Access

You can save an OLAP model even if multiple users are reading it by selecting **Exclusive Access** on the **Welcome** screen. Opening an existing OLAP model with **Exclusive Access** guarantees that no other user can modify the OLAP model.

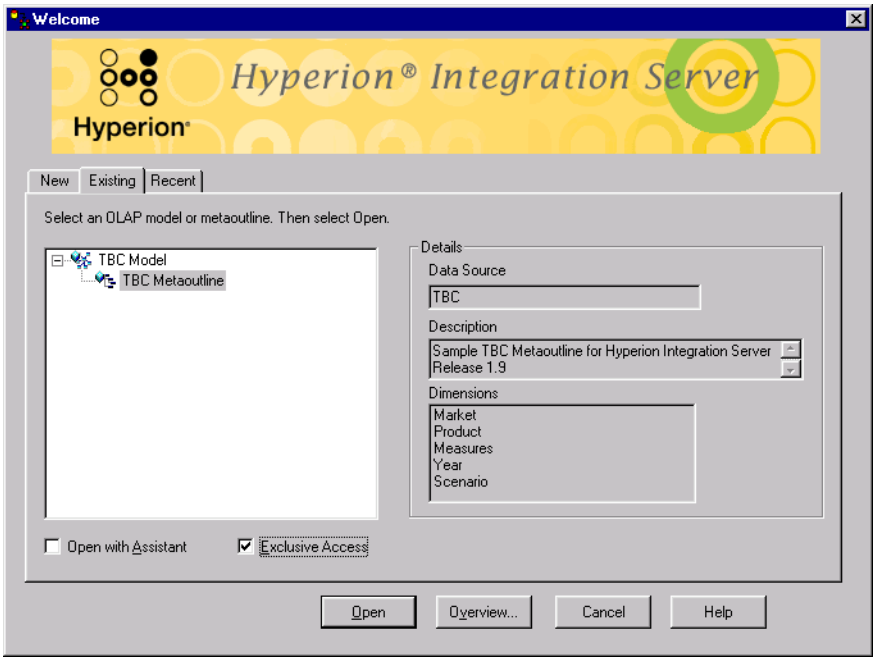



*Figure 3-9: Selecting the Exclusive Access Option*

When opening an existing OLAP model, select **Exclusive Access** to place write-lock privileges on the model. This option allows you to do both of the following:

- You can save an existing OLAP model.
- You can modify and save an existing OLAP model. If you open a read-only OLAP model and no other user has **Exclusive Access**, you can save the OLAP model to the same name.

If **Exclusive Access** is not selectable, another user has write-lock privileges, and you have read-only privileges.

If you want to modify and save an OLAP model but have read-only privileges, use File > Save As to save the OLAP model to a different name. See "Saving an OLAP Model to a Different Name" on page 3-27. Also see the *Hyperion Integration Server System Administrator's Guide* for information on unlocking OLAP models.

**Note:** You cannot save an OLAP model if other users are reading it. Save it later or use File > Save As to save the model with a different name. See "Saving an OLAP Model to a Different Name" on page 3-27.

## Saving an OLAP Model Manually

If the OLAP model that you want to save has been previously saved and you do not want to give the OLAP model a different name, select File > Save.

OLAP Integration Server writes the changes to the OLAP Metadata Catalog to which you are connected.

If you have not previously saved the OLAP model or if you want to save the OLAP model to a different name, you must use the Save New OLAP Model dialog box.

➤ To use the **Save New OLAP Model** dialog box to save an outline manually:

1. Select one of the following menu options:

   - If you are saving a an OLAP model for the first time, select File > Save.
   - If you are saving a previously saved an OLAP model to a new name, select File > Save As.

   The **Save New OLAP Model** dialog box, as shown in Figure 3-10, is displayed.

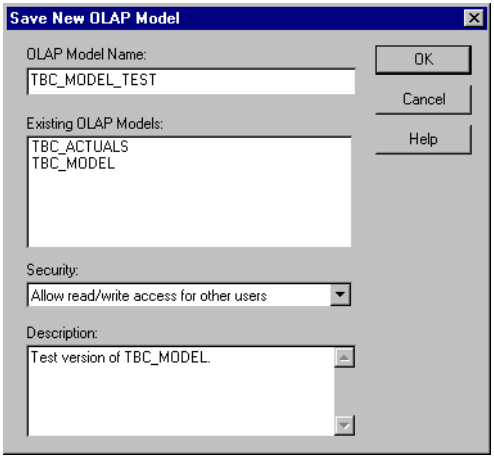



*Figure 3-10: Saving an OLAP Model*

**Note:** If you save an existing model, the Save New OLAP Model dialog box is not displayed

2. In the **OLAP Model Name** text box, enter the name of the OLAP model.

   See "Observing Naming Rules" on page 3-28.

3. From the **Security** drop-down list box, select one of the following options to determine who can access the OLAP model:

*Table 3-2: Read/Write Settings*

| Option | Permitted Activity |
|---|---|
| Allow read/write access for other users. | Other users can read and write to the OLAP model. This setting is the default. |
| Allow read access for other users. | Other users can read the OLAP model but not write to the OLAP model; that is, another user cannot save changes to the OLAP model. |
| Disallow all access for other users. | Other users can neither read nor write to the OLAP model. Only you can read or write to the OLAP model. |

**Note:** To change the permissions on an OLAP model after you create it, you must use the tools provided with the relational database that contains the OLAP Metadata Catalog or use File > Save As to save the OLAP model to a different name. See the *Hyperion Integration Server System Administrator's Guide*.

4. If you want to describe the OLAP model, in the **Description** text box, enter a brief description.

   The description can include up to 255 characters. Hyperion Integration Server displays the description in the following places:

   - The **Save New OLAP Model** dialog box when you select File > Save As (see "Saving an OLAP Model to a Different Name" on page 3-27)
   - The **OLAP Model Properties** dialog box (see "Changing Read/Write Access" on page 3-25)
   - The **Existing** tab of the **Welcome** dialog box (see "Working with OLAP Models" on page 3-3)

5. Click OK.

   If the OLAP model is invalid, Hyperion Integration Server displays an error message. You can save the OLAP model without correcting the error, but you cannot use an invalid OLAP model to create a metaoutline.

**Note:** If you attempt to save an OLAP model that another user currently has open, a dialog box informs you that the OLAP model is currently locked or open. You must wait to save until all other users close the OLAP model.



## Changing Read/Write Access

At any time after you save an OLAP model, you can change the access rights that you originally assigned by using the OLAP Model Properties dialog box.

➤ To change read/write access to an OLAP model:

1. In a blank space of the right frame, right-click and select Properties.

   Hyperion Integration Server displays the **General** tab of the **OLAP Model Properties** dialog box.

2. In the **Security** drop-down list box, change the user access defined for the model.

   For more information about security settings, see Table 3-2.

3. Click OK.

## Specifying Retrieval Optimization Preferences

Oracle users have the option of specifying data retrieval optimization preferences in the Optimization tab of the OLAP Model Properties dialog box. Optimization preferences determine how OLAP Integration Server handles data queries for individual users. This option exists for Oracle users because Oracle is the only RDBMS that offers performance options for overall database access. Overall database information is stored with the OLAP model.

➤ To specify optimization preferences for Oracle users:

1. In a blank space of the right frame, right-click and select Properties.

   The **OLAP Model Properties** dialog box is displayed.

   **Tip:** You can also open the OLAP Model Properties dialog box by selecting View > Properties > OLAP Model.

2. Select the **Optimization** tab, shown in Figure 3-11.

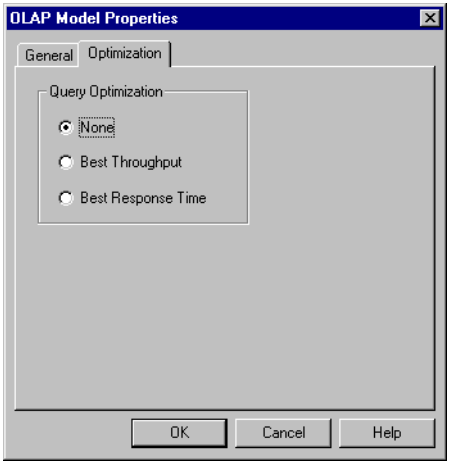



*Figure 3-11: Optimization Tab of the OLAP Model Properties Dialog Box*

3. Select one of the following preferences:
   - **None** to have no parameters included in user data queries
   - **Best Throughput** to have all users treated equally, with no high-priority data retrieval
   - **Best Response Time** to have some queries (queries from users who have this setting) receive a higher priority response than other, concurrent queries
4. Click OK.

## Saving While Closing

If you try to close an OLAP model without saving it, a dialog box prompts you to save your changes.

Click Yes to save the OLAP model or No to close the OLAP model without saving the changes. See "Saving an OLAP Model Manually" on page 3-22.

---

**CAUTION:** If the OLAP model that you are closing is a new OLAP model that has never been saved, you will lose it if you click No.

---



## Saving an OLAP Model to a Different Name

Save an OLAP model to a different name if you want to take either of the following actions:

- Change the name of an OLAP model. (The original OLAP model retains the original name.)
- Make a copy of an OLAP model.

  You cannot copy an OLAP model using the Copy function that the operating system provides because the OLAP model resides in an OLAP Metadata Catalog. Instead, you need to save the OLAP model to a different name.

To save an OLAP model to a different name, select File > Save As. Hyperion Integration Server displays the **Save New OLAP Model** dialog box. For more information, see "Saving an OLAP Model Manually" on page 3-22.

## Observing Naming Rules

When naming objects in an OLAP model, follow these rules:

- Do not use more than 80 characters.
- Do use mixed uppercase and lowercase characters. Names are not case-sensitive in Hyperion Essbase unless there is a check mark next to the Settings > Case Sensitive Members menu item in Hyperion Essbase.
- Do not use the following characters anywhere in a name:

| | | | |
|---|---|---|---|
| `&` | (ampersand) | `<` | (less than sign) |
| `*` | (asterisk) | `( )` | (parenthesis) |
| `@` | (at sign) | `.` | (period) |
| `\` | (backslash) | `+` | (plus sign) |
| `{ }` | (brace) | `‘` | (single quotation mark) |
| `,` | (comma) | | tab |
| `-` | (dash, hyphen, or minus sign) | `_` | (underscore) |
| `“` | (double quotation mark) | `\|` | (vertical bar) |
| `=` | (equal sign) | | |

- Do not place spaces at the beginning or end of a name. Hyperion Essbase ignores spaces at the beginning or end of a name.
- Do not use the following words as dimension or member names:
  - `$$$UNIVERSE$$$`
  - `#MISSING` or `#MI`
  - Calc script commands, operators, and keywords

    For a list of commands, see the online *Quick Technical Reference* in the `DOCS` directory.
  - Report script commands

    For a list of commands, see the online *Quick Technical Reference* in the `DOCS` directory.

- Function names and function arguments

  For a list of functions, see the online *Quick Technical Reference* in the `DOCS` directory.

- Names that are already in the database: names of dimensions and members (unless the member is shared), names of generations and levels, and aliases and combinational aliases.

**Note:** If you tag members as Dynamic Time Series, do not use the associated generation names: History, Year, Season, Period, Quarter, Month, Week, or Day. See the *Hyperion Essbase Database OLAP Server Administrator's Guide* for more information.



# Printing OLAP Models

You can print the OLAP models contained in the right frame of the OLAP Model main window to use as a hard-copy reference or to retain on file. The relational tables in the left frame of the window do not print when you print the OLAP model.

➤ To print an OLAP model:

1. Click in the right frame.
2. Select File > Print or click the Print toolbar button,  to open the standard **Print** dialog box.
3. Select the settings that you want.
4. Click OK.

# Deleting OLAP Models

When you delete an OLAP model, Hyperion Integration Server removes the OLAP model from OLAP Metadata Catalog.

➤ To delete one or more OLAP models:

1. Make sure that you are connected to the OLAP Metadata Catalog in which the OLAP model is stored.

   If you are connected, the lower-right corner of the OLAP Model main window contains the appropriate OLAP Metadata Catalog name (for example, TBC_MD in the sample TBC database).

   If you are not connected, select Connections > OLAP Metadata Catalog > Connect and log on to the correct OLAP Metadata Catalog.

2. Select File > Delete to display the **Delete an OLAP Model/Metaoutline** dialog box, shown in Figure 3-12.

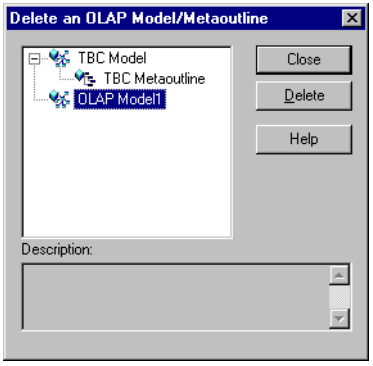



*Figure 3-12: Deleting an OLAP Model*

3. Select the OLAP model to delete; for example, **OLAP Model1**, and click Delete.

   **Note:** To select multiple OLAP models, select one OLAP model, hold down the Ctrl key, and select each additional OLAP model.

   Hyperion Integration Server displays a dialog box that prompts you to confirm that you want to delete the OLAP model.



---

**CAUTION:** Use caution when deleting. You cannot undo the deletion of an OLAP model.

---

4. Click Yes.

   OLAP Integration Server deletes the selected model from OLAP Metadata Catalog.

**Note:** If you attempt to delete an OLAP model that you or another user currently has open, a dialog box informs you that the OLAP model is currently locked or open. If you have it open, close the OLAP model and select Connections > OLAP Metadata Catalog > Delete Locks. If other users have the OLAP model open, you must wait until they close it.

Chapter 4

# Creating the Fact Table, Accounts Dimension, and Time Dimension

Every OLAP model contains a fact table. The fact table serves as a container for all numeric (measures) data values that you want to track. Measures are items such as unit price, units sold, and sales amounts. If you track more than one measure, you must create an accounts dimension to hold the measures. If an application requires analysis to be performed on a time basis, you should also create a time dimension.

The topics of this chapter describe the role of the fact table in an OLAP model and provide procedures for creating a fact table, an accounts dimension, and a time dimension:

# Understanding the OLAP Model Workflow

Figure 4-1 illustrates the workflow for creating an OLAP model. This workflow includes creating and working with an OLAP model; creating a fact table, an accounts dimension, and a time dimension; creating and working with dimensions; editing and transforming the columns of dimension tables; joining dimension tables, and creating and working with hierarchies. This chapter focuses on how to create a fact table, an accounts dimension, and a time dimension.

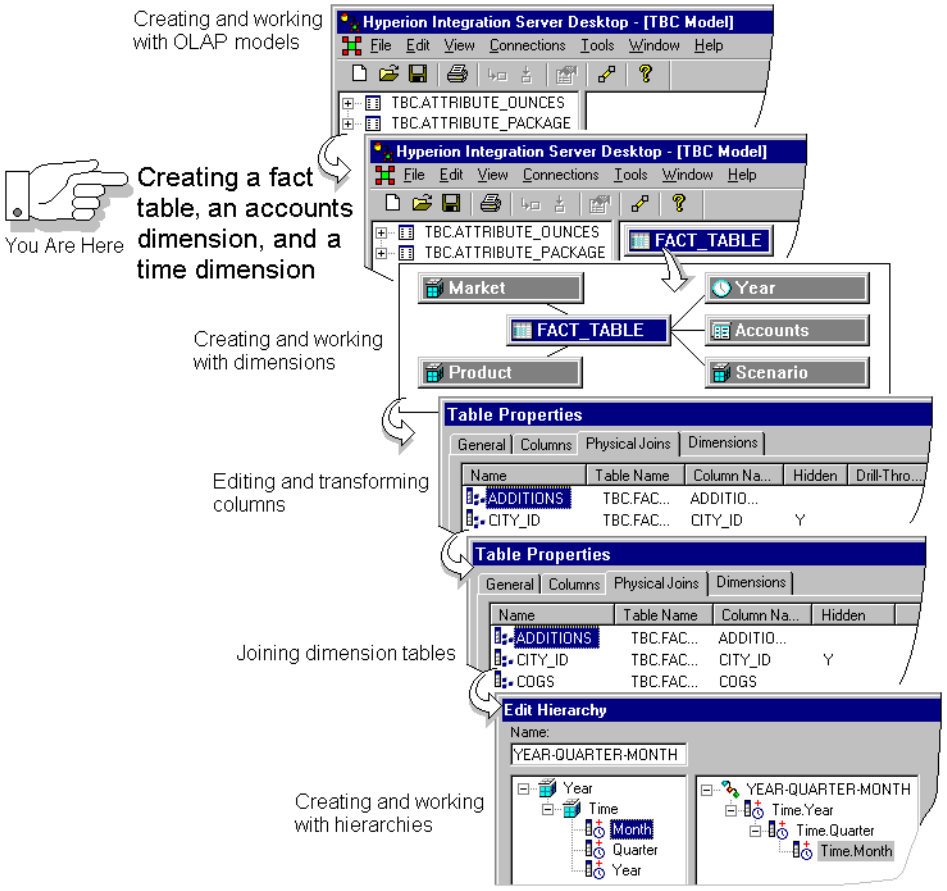



*Figure 4-1: OLAP Model Workflow*

# About the Fact Table

A fact table contains detailed measures (data values) for each dimension intersection of its OLAP model. Measures include items such as price or units sold. For example, if the OLAP model contains Product, Geography, and Time dimensions, the fact table may contain the number of units of Product A sold in New York in January.

## What Components Are Included in the Fact Table

The fact table is made up of one or more source tables and contains a compound key that uniquely identifies the records that it contains. The compound key consists of primary and foreign keys and links to each dimension table of the OLAP model, as illustrated in Figure 4-2. The dimension tables contain columns of related information from the source tables.



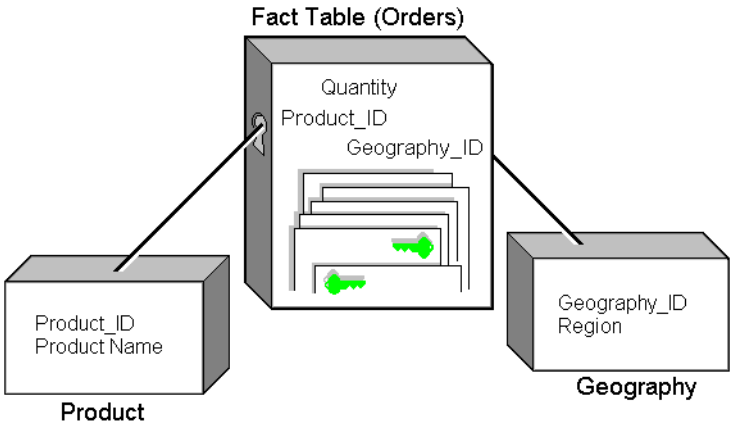



*Figure 4-2: Dimension Tables Joined to the Fact Table with Primary and Foreign Keys*

For example, you might store product ID and geographic information in two or more tables in the relational database. You can use the relational tables to create Product and Geography dimensions in the OLAP model. You join the dimensions to the fact table, either manually or through automatic joins. When you create a new dimension, OLAP Integration Server creates an automatic join whenever it detects a natural join relationship. A natural join exists between two tables when a primary key of one table is identified as a foreign key in the other table.

Bear in mind that OLAP Integration Server does not create additional structures for the relational data that you use when creating an OLAP model. Rather, it reorganizes the existing data into a format that you can use to create a metaoutline.

## Why a Fact Table is Required

Every OLAP model that you create must contain a fact table. The fact table contains the values that are later aggregated and reported by Hyperion Essbase. The first source table that you drag to the right frame automatically becomes the fact table. Because of its relationship to each dimension that you define, the fact table becomes the hub (the center) of the OLAP model. After you create the fact table, you can add other source tables to it.

Any columns that you add to the fact table are added automatically to the accounts dimension. Likewise, columns that you add to the accounts dimension are added automatically to the fact table. All measures values that you want reported are referenced through the fact table and through the accounts dimension, which is an exact copy of the fact table.

If, in the relational data source, you have only one column containing measures, you can store the column in the fact table, and you do not need to create an accounts dimension. For more information about the accounts dimension, see "About the Accounts Dimension" on page 4-7.

# Creating the Fact Table

The first table that you drag to the right frame of the OLAP Model main window becomes the fact table. When you create the fact table, you also choose whether to create the time and accounts dimensions automatically.

---

**CAUTION:** If the relational database stores accounts measures values and time values in the same columns, you must use a non-standard method to create the OLAP model accounts dimension. This situation is common in some applications, such as general ledger and legacy systems. See "Switching the Time and Accounts Dimensions" on page 4-14.

---

For more information on the fact table, see "About the Fact Table" on page 4-3. Before creating an OLAP model, be sure to read and follow the procedures in Chapter 2, "Preparing Relational Data Sources."

➤ To create a fact table:

1. Drag the appropriate source table from the left frame to the right frame, as shown in Figure 4-3.

   If you are not sure which table to use, view the data stored in each source table by selecting View > View Table Data. For more information, see "Viewing Table Data" on page 3-17.

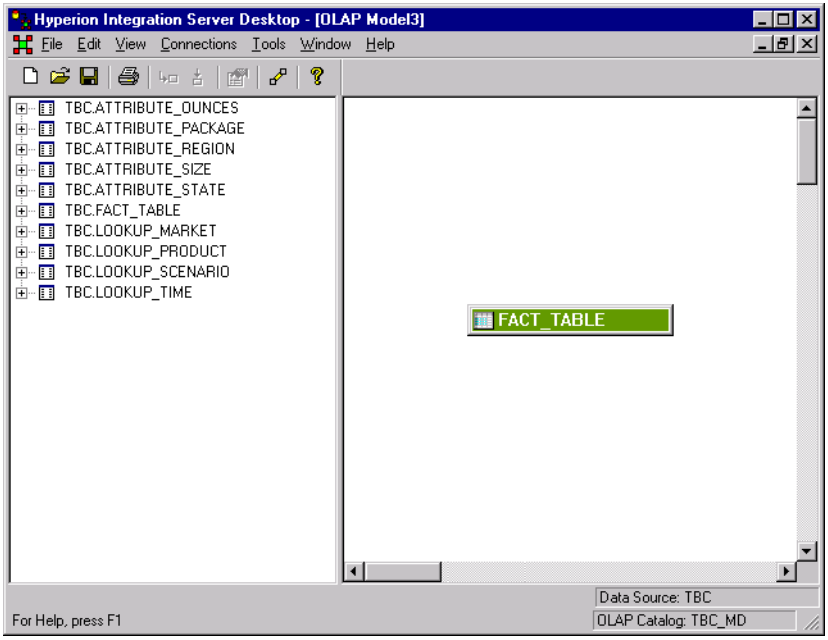



*Figure 4-3: Creating a Fact Table*

The fact table uses the name of the source table that you select to create it. To rename the fact table, see "Renaming Dimensions and the Fact Table" on page 5-14.

2. Decide whether to create the time dimension first, and then the accounts dimension, by responding to the following prompts:

   - Would you like a Time dimension using a value from the fact table? Click Yes or No.
   - Would you like to add an Accounts dimension? Click Yes or No.

| When to Create the Time Dimension Automatically | When to Create the Accounts Dimension Automatically |
|---|---|
| The table that you select to create the fact table contains *at least one column with a time value* (a shipping date, for example) and you want to use that column to create a time dimension.<br>If you select this option, the time dimension initially contains the same columns as the fact table. However, any columns that you later add to the time dimension are not added automatically to the fact table; likewise, if you add columns to the fact table, they are not added automatically to the time dimension. | The source table that you select to create the fact table contains *at least one column with a measures value* (units sold, for example) and you want to use that column to create an accounts dimension.<br>The accounts dimension inherits all of the columns that the fact table contains. In an OLAP model, the fact table and the accounts dimension are mirror images of each other. The accounts dimension reflects changes made in the fact table; likewise, the fact table reflects changes made in the accounts dimension. |

After you respond to the time and accounts dimension prompts, the fact table is displayed in the right frame.

If you create an accounts dimension automatically, an object is displayed in the right frame. The object is identified by the "Accounts" label and the  graphic. A join line connects the fact table and the accounts dimension.

If you create a time dimension automatically, an object is displayed in the right frame. The object is identified by the "Time" label and the  graphic. A join line connects the fact table and the time dimension.

3. After you create the fact table, proceed with the following options:

   - To add tables from the relational data source to the fact table, see "Adding Tables to the Fact Table" on page 5-12.
   - To add time and accounts dimensions manually, if you opted not to create time and accounts dimensions automatically along with the fact table, see "Creating the Time Dimension Manually" on page 4-9 and "Creating the Accounts Dimension Manually" on page 4-8.
   - To create standard dimensions, see Chapter 5, "Creating and Working with Dimensions."



# About the Accounts Dimension

The accounts dimension consists of columns that contain measures. Measures values are numeric data values that you track over time, such as sales, discounts, and commissions. In Hyperion Essbase, accounts is a dimension type that makes accounting intelligence available during online analytical processing.

An OLAP model is not complete without at least one column from the relational data source that contains a measure. If you add more than a single measure to the fact table, you *must* create an accounts dimension in the OLAP model.

If you have a non-financial application or for any reason have only a single measure to be analyzed and reported, you do not need to create an accounts dimension. Add the appropriate measure to the fact table. You can use the measure from the fact table to add a single measure to the metaoutline that you create from the OLAP model, as described in the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*.

---

**CAUTION:** If the relational database stores accounts measures values and time values in the same columns, you must use a non-standard method to create the OLAP model accounts dimension. This situation is common in some applications, such as general ledger and legacy systems. See "Switching the Time and Accounts Dimensions" on page 4-14.

---

# Creating the Accounts Dimension Manually

If you need an accounts dimension and do not create it automatically, you can create it manually. To add an accounts dimension manually:

1. Right-click the fact table, and select Add Accounts Dimension.

   An accounts dimension is displayed in the right frame. The dimension is identified by the "Accounts" label and the  graphic. A join line connects the accounts dimension to the fact table.

2. If needed, add one or more source tables to the accounts dimension.

   You must add source tables directly to the accounts dimension; you cannot create a dimension branch. For more information, see "Adding Source Tables Directly to Dimension Tables" on page 5-7.

3. If needed, add one or more user-defined columns to the accounts dimension.

   Any columns that you add to the accounts dimension are also added to the fact table. For instructions on how to add new columns to the accounts dimension, see "Creating User-Defined Columns" on page 6-14.

# About the Time Dimension

In a Hyperion Essbase database, you use the dimension tagged as time to describe how often you collect and update data. You can use time dimension members, such as Year and Quarter, to report yearly and quarterly totals for members of other dimensions; for example, total sales of Diet Cola in Boston for the first quarter.

In an OLAP model, you create a dimension tagged as time to provide a time structure from which to create a specific time dimension in an associated metaoutline. You cannot create a time dimension in a metaoutline if you did not create, in the associated OLAP model, a dimension that contains a time value.

It is not mandatory to include a time dimension in an OLAP model. However, if an application requires analysis to be performed on a time basis, such as examining trends over time or making seasonal sales comparisons, you should include a time dimension. You can create only one time dimension in an OLAP model, but you can create as many time dimension hierarchies, within that single time dimension, as are necessary to meet reporting requirements.

In an OLAP model, the time dimension includes the following elements:

- One or more time-related columns joined to the fact table.
- A time-related column for every time dimension member that you want to report on in Hyperion Essbase; for example, a separate column for year, for quarter, and for month. You use these time-related columns in an associated metaoutline to define members of the time dimension.
- As needed, one or more hierarchies that define how to roll up member data in the time dimension. For example, a time hierarchy can include Month, Quarter, and Year members whereby monthly totals roll up to quarterly totals and quarterly totals roll up to yearly totals. Creating hierarchies in an OLAP model provides a multilevel structure that you can view when creating dimensions in metaoutlines.



You may have specific reporting and analysis requirements that are based on specialized calendar years, such as in a merchant calendar. For more information on how to prepare the relational data source to meet such requirements, see "Creating Time and Accounts Dimensions" on page 2-20.

# Creating the Time Dimension Manually

You can create a time dimension either automatically or manually. The procedure for automatically creating a time dimension is described in "Creating the Fact Table" on page 4-4. After automatically creating the time dimension, you can edit the columns it contains, as described in Chapter 6, "Editing Columns in Dimension Tables and the Fact Table."

If you do not create a time dimension automatically when you create the fact table, you can add the time dimension manually. You create the time dimension manually in one of two ways:

- Based on the fact table, whereby the time dimension table inherits all of the columns in the fact table. See "Manually Creating a Time Dimension that Is Based on the Fact Table" on page 4-10.
- Based on another table in the relational data source, such as a user-defined calendar. Follow the same procedure that you use for creating standard dimensions, with some exceptions that are noted in the procedure. See "Creating Dimensions" on page 5-3.

## Manually Creating a Time Dimension that Is Based on the Fact Table

➤ To add a time dimension that is based on the fact table:

1. Right-click the fact table, and select Add Time Dimension.

   A time dimension is displayed in the right frame. The dimension is identified by the "Time" label and the  graphic. A broken join line connects the time dimension to the fact table. The time dimension table includes all of the columns in the fact table. The broken join line indicates that you need to select time-related columns in both tables to define the join relationship between the two tables.

2. Define the join between the two tables, as described in Chapter 8, "Joining Dimension Tables."

3. If needed, add one or more source tables to the time dimension.

   You must add source tables directly to the time dimension; you cannot create a dimension branch. For more information, see "Adding Source Tables Directly to Dimension Tables" on page 5-7.

4. If needed, add one or more user-defined columns to the time dimension.

   Add new columns to the time dimension to use the columns as levels in a time hierarchy. If you want monthly and quarterly aggregations in the Hyperion Essbase database, you need a monthly and a quarterly column in the OLAP model time dimension.

   If the time dimension does not include the needed columns, you must add new, user-defined columns. Add a column for each level of aggregation that the OLAP model does not currently include. You derive the values of the new columns from date columns currently existing in the data; for example, a transaction or sales date.

   For instructions on how to add new columns to the time dimension table, see "Creating User-Defined Columns" on page 6-14.

## Determining if a User-Defined Calendar Is Needed

OLAP Integration Server can map a date in a source column to a day, week, month, quarter, or year time period in the time dimension.

If you need to analyze data by other time periods or if you use a calendar different from the standard Gregorian calendar, create a user-defined calendar in the source data. A user-defined calendar is a source table in a relational database that maps a business calendar to the date information associated with the measures data. If you answer "Yes" to any of the following questions, create a relational source table containing your user-defined calendar to perform the required mapping.

- Does the fiscal year start on a date other than January 1?
- Does the time dimension include a member for a period that has no matching column in the source data for the fact table? For example, do you want to measure by sales season, but no sales season column exists in the source data for the fact table?
- Do you measure business cycles in periods other than the standard Gregorian calendar months? For example, instead of the standard 12 months to the year, is your year measured by 13 four-week periods?



## Designing a User-Defined Calendar

Assume that a user-defined calendar is located in a table in the source database that you are using to create an OLAP model. This table requires both of the following:

- A column for each potential time period (member) in the time dimension.
- A time-related key column that maps to the lowest-level, time-related column associated with the measures data. See "Associating Time Data with Measures Data" on page 2-20.

Table 4-1 is an example of a user-defined calendar.

*Table 4-1: A User-Defined Calendar*

| TIMESTAMP | YEAR | QTR | MONTH | WK_OF YEAR | DAY_OF YEAR | DAY_OF WEEK | SEASON |
|---|---|---|---|---|---|---|---|
| Jul 1 2000 12:00AM | 2000 | 1 | 1 | 1 | 1 | Thurs | Summer |
| Jul 2 2000 12:00AM | 2000 | 1 | 1 | 1 | 2 | Fri | Summer |
| Jul 5 2000 12:00AM | 2000 | 1 | 1 | 2 | 5 | Mon | Summer |
| Jul 6 2000 12:00AM | 2000 | 1 | 1 | 2 | 6 | Tues | Summer |
| Jul 7 2000 12:00AM | 2000 | 1 | 1 | 2 | 7 | Wed | Summer |
| Jul 8 2000 12:00AM | 2000 | 1 | 1 | 2 | 8 | Thurs | Summer |
| Jul 9 2000 12:00AM | 2000 | 1 | 1 | 2 | 9 | Fri | Summer |
| : | : | : | : | : | : | : | : |
| Sep 1 2000 12:00AM | 2000 | 1 | 3 | 10 | 63 | Wed | Autumn |
| Sep 2 2000 12:00AM | 2000 | 1 | 3 | 10 | 64 | Thurs | Autumn |
| Sep 3 2000 12:00AM | 2000 | 1 | 3 | 10 | 65 | Fri | Autumn |
| Sep 6 2000 12:00AM | 2000 | 1 | 3 | 11 | 66 | Mon | Autumn |
| | | | | | | | |

Table 4-1 illustrates several points to consider when creating a user-defined calendar:

- The first column, TIMESTAMP, is the key column in the user-defined calendar table. This column is joined to a time-related column in the fact table. As joined columns, the columns must have the same data type. The key column in a user-defined calendar defines the lowest level of time analysis that the source data can provide to the fact table.

- In order for the measures data (such as sales) to map to the time members, each value in the date column of the source data for the fact table *must* match a value in the key column of the user-defined calendar table.

  TIMESTAMP is a column with a datetime data type. Its values include the time rounded in minutes to 12:00 AM; for example, `Jul 2 2000 12:00AM`. TIMESTAMP is the join column from the user-defined table to the fact table. Therefore, not only must the join column in the fact table have a datetime data type, but also each time value must be rounded to `12:00AM`. For example, if a sale occurred on July 2, 2000, at 8:30 AM, you must prepare the transaction date in the source data so that the fact table and the user-defined calendar show the same value, `Jul 2 2000 12:00AM`.

- Representing a fiscal calendar starting with July 1, the calendar in Table 4-1 shows the `July 1` timestamp value mapping to a `1` value for the quarter, month, week, and day of the year.

- Table 4-1 shows only a few rows for July and September. Actually, a table must contain a row for every potential date in the source data. For example, Table 4-1 does not include rows for the dates that are Saturdays and Sundays. Unless you are certain that no data is posted with a Saturday or Sunday date, you should include rows for those days.

- Table 4-1 includes two columns related to the day: the day of the year and the day of the week. If you want to analyze whether sales are better on Mondays than on Wednesdays, you need a column in the user-defined calendar and a corresponding member in the time dimension for the day of the week.

- Table 4-1 includes a SEASON column. Although Autumn traditionally starts in mid-September, the example shows Autumn starting on September first. Set the values in the calendar to correspond to the needs of your company.

A user-defined calendar is as unique as the business requirements of your company.

# Switching the Time and Accounts Dimensions

Most databases store time values in rows and measures values in columns, as shown in Table 4-2. For this type of data, you do not need to switch the time and accounts dimensions.

*Table 4-2: Time Data Stored in Rows*

| | COGS | Sales |
|---|---|---|
| January | 105 | 127 |
| February | 111 | 214 |
| March | 120 | 189 |

Some applications, such as general ledger applications, store time values in columns and measures values in rows, as shown in Table 4-3. For these databases, you must switch the time and accounts dimensions in the OLAP model and then set the dimension types correctly in the associated metaoutline. For more information, see "Switching the Time and Accounts Dimensions in the OLAP Model" on page 4-15 and "Setting Dimension Properties in the Metaoutline" on page 4-17.

*Table 4-3: Time Data Stored in Columns*

| | January | February | March |
|---|---|---|---|
| COGS | 105 | 111 | 120 |
| Sales | 127 | 214 | 189 |

## Switching the Time and Accounts Dimensions in the OLAP Model

➤ To switch time and accounts dimensions in the OLAP model:

1. Create and name the fact table using the procedures in "Creating the Fact Table" on page 4-4, with the following exceptions:
   - To the prompt for creating the time dimension automatically, click No.
   - To the prompt for creating the accounts dimension automatically, click Yes.
   - Right-click the Accounts dimension table and select Properties to display the **Dimension Properties** dialog box, change the name of the Accounts dimension to Time, and click OK.
   - Right-click the newly created Time dimension table and select View Columns to display the column names.
   - Right-click the Accounts table in the column view, select Properties to display the **Table Properties** dialog box, change the name of the Accounts dimension to Time, and click OK.
2. In the OLAP Model main window, drag the source table that you used for the fact table; for example, FACT_TABLE, to the right frame to create a new dimension.
3. When a dialog box prompts you to rename the table (because that name already exists), click Yes.

   Hyperion Integration Server displays the **Create New Table** dialog box, as shown in Figure 4-4.

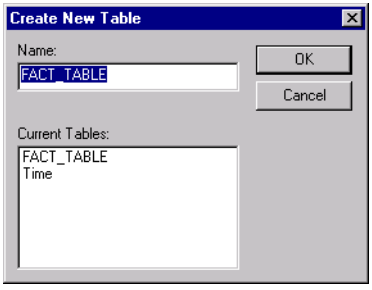



*Figure 4-4: Create New Table Dialog Box*

4. In the **Name** text box, type **Accounts** and click OK to close the dialog box.
5. In the OLAP Model main window, use the join tool to join the new dimension to the fact table.

   Click the join tool (or right-click in a blank space of the right frame and select Add Joins Mode) to activate Add Joins Mode, and drag the tool between the two tables.

   Hyperion Integration Server displays the **Create New Dimension** dialog box with the dimension name Accounts in the **Name** text box, as shown in Figure 4-5.

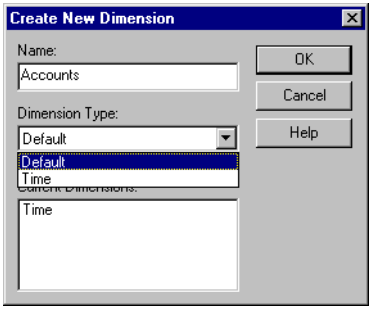



*Figure 4-5: Create New Dimension Dialog Box*

6. From the **Dimension Type** drop-down list, select Default, and click OK to close the dialog box.
7. Define the joins completely, using all necessary keys.
8. Create any other dimensions that you need (see "Creating Dimensions" on page 5-3).
9. Save the OLAP model (see "Saving OLAP Models" on page 3-20).

Complete the OLAP model before using it to create a metaoutline.

## Setting Dimension Properties in the Metaoutline

➤ To set properties for the time and accounts dimensions in the metaoutline:

1. From the newly created OLAP model, create a metaoutline (see the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*).
2. After you add the time dimension to the metaoutline, right-click the time dimension and select Properties.

   Hyperion Integration Server displays the **General** tab of the **Dimension Properties** dialog box.

   a. Select the **Member Info** tab.

   b. From the **Dimension type** options, select Time.

   c. Complete information on the other tabs as needed, and click OK to close the dialog box.
3. As needed, right-click a member level name within the time dimension, select the Properties option, complete information on the tabs.
4. Click OK to close the dialog box.
5. Right-click the accounts dimension and select Properties.

   Hyperion Integration Server displays the **General** tab of the **Dimension Properties** dialog box.

   a. Select the **Member Info** tab.

   b. From the **Dimension type** options, select Accounts.

   c. Complete information on the other tabs as needed, and click OK to close the dialog box.
6. As needed, right-click a member level name within the accounts dimension, select the Properties option, and specify on each tab the information required to define the selected member level.
7. Click OK to close the dialog box.
8. Continue with the workflow as described in the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*.

Chapter 5

# Creating and Working with Dimensions

After you create a fact table, continue building an OLAP model by adding dimensions. You can create a time dimension, an accounts dimension, and various other dimensions in an OLAP model. If you do not create time and accounts dimensions when you create the fact table, you can add them manually as described in Chapter 4, "Creating the Fact Table, Accounts Dimension, and Time Dimension."

You may also need to create additional dimensions to provide Hyperion Essbase with enough information to meet your business analysis requirements. This chapter describes how to create dimensions other than time and accounts and how to edit dimensions and the fact table.

This chapter contains the following topics:

# Understanding the OLAP Model Workflow

Figure 5-1 illustrates the workflow for creating an OLAP model. This workflow includes creating and working with an OLAP model; creating a fact table, an accounts dimension, and a time dimension; creating and working with dimensions; editing and transforming the columns of dimension tables; joining dimension tables; and creating and working with hierarchies. This chapter focuses on how to create dimensions other than time and accounts and how to edit dimensions and the fact table.

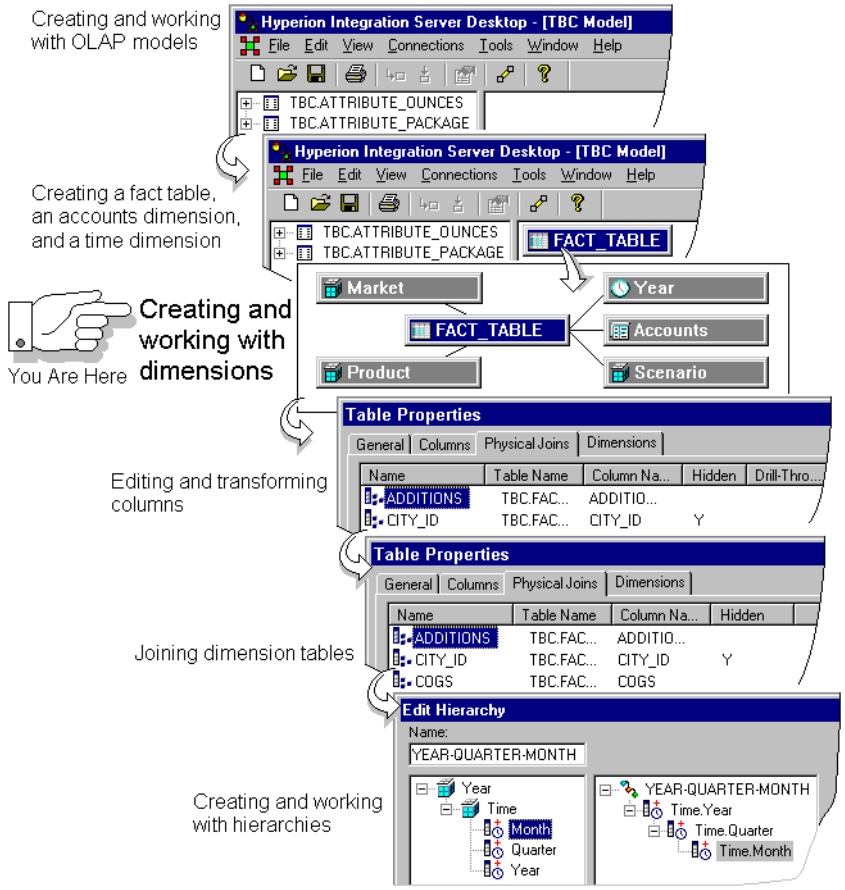



*Figure 5-1: OLAP Model Workflow*

# Creating Dimensions

This topic describes how to create a dimension that joins directly with the fact table. If you have not yet created a fact table, see "Creating the Fact Table" on page 4-4.

**Tip:** You can create two or more dimensions from a single relational database table. Each dimension must have a different name. You should hide all columns that do not contain values that you want to be reported for the dimension (see "Hiding and Showing Columns" on page 6-6).

➤ To create dimensions:

1. Drag the appropriate source table from the left frame to the right frame.

   If you are not sure which table to use, view the data stored in the table by selecting a table, right-clicking, and then selecting View Table Data. For information, see "Viewing Table Data" on page 3-17.

   - If a predefined primary-foreign key relationship exists between the table that you are adding and the fact table, Hyperion Integration Server joins the dimension table to the fact table. Because the dimension table joins directly to the fact table, the dimension table is also a dimension.
   - If a predefined primary-foreign key relationship does not exist between the table that you are adding and the fact table, Hyperion Integration Server does not join the dimension table to the fact table. To create the dimension, join the dimension table to the fact table manually. See "Joining Dimension Tables Directly to the Fact Table" on page 8-6.

   Hyperion Integration Server displays the **Create New Dimension** dialog box, as shown in Figure 5-2. By default, the new dimension inherits the name of the source table you select to create it.

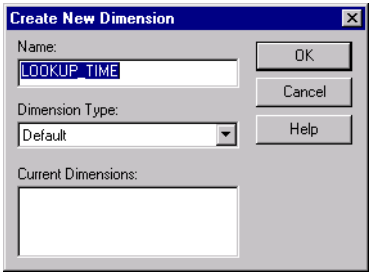



*Figure 5-2: Creating a New Dimension*

2. If you want to change the dimension name, in the **Name** text box, enter a new name.

   For naming rules, see "Observing Naming Rules" on page 3-28.

3. From the **Dimension Type** drop-down list, select Default.
4. Click OK.

# Adding Tables to Dimension Tables or to the Fact Table

You can add tables to existing dimension tables or to the fact table by using either of two methods:

- You can add a source table to an existing dimension table to create a dimension branch by adding the new source table to the OLAP model and joining it to the existing dimension table.
- You can add the new source table directly to an existing dimension table without creating a branch.

The method used to add a table to a dimension affects both the OLAP model and the metaoutlines that you create from the OLAP model.

This topic describes how to add source tables to dimension tables and to the fact table and explains the reasons to add tables to dimension tables or dimension branches. This topic contains the following subtopics:

## About Dimension Branches

A dimension branch contains a dimension table that joins directly to the fact table (known as the *primary dimension table*) and to one or more related dimension tables. In Figure 5-3, the LOOKUP_PRODUCT and ATTRIBUTE_OUNCES dimension tables form a dimension branch for the Product dimension.

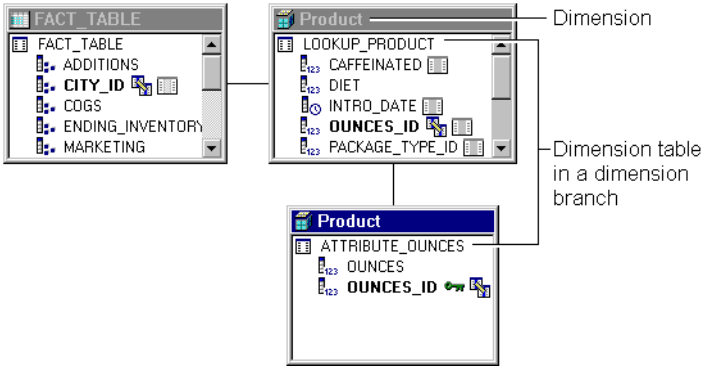



*Figure 5-3: A Dimension Branch*

Notice that both dimension tables are located in the Product dimension and that both dimension tables display the name Product in their title bars.

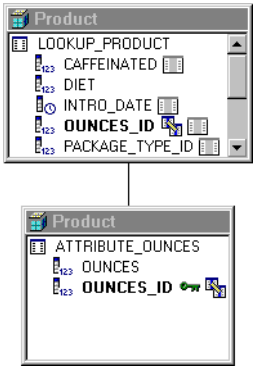
The hierarchical structure of dimension branches is similar to the consolidation levels reported in Hyperion Essbase. The fact table in an OLAP model is considered level 0 (zero). A dimension table joined directly to the fact table is level 1. A dimension table joined to another dimension table that is joined directly to the fact table is level 2. A dimension table joined to a level 2 table is level 3, and so on.

Level numbers are assigned to dimension tables and branches to provide a hierarchical structure to work with as you create a metaoutline. Viewing a hierarchical structure in an OLAP model helps you to define the correct consolidation levels you require. Level numbers do not carry over to the metaoutline or to the subsequent Hyperion Essbase outline.

## Adding Source Tables to Create Dimension Branches

The use of dimension branches offers some advantages, as illustrated in Figure 5-4. In both the OLAP model and the metaoutline, it is easy to tell to which source table each column belongs. In the metaoutline, it is easy to add columns in the required order—the hierarchical order defined by the dimension branch in the OLAP model. For example, in the metaoutline, the columns in the ATTRIBUTE_OUNCES table must be added before and positioned above the columns from the LOOKUP_PRODUCT table.

**In an OLAP model**




**In a metaoutline**

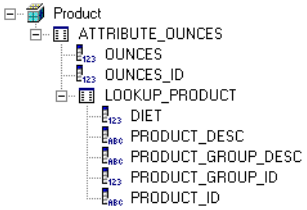



*Figure 5-4: Displaying Table and Column Information in a Dimension Branch*

When you join a source table to an existing dimension table to create a dimension branch, all columns of the table are added to the dimension. Each dimension table in the dimension branch displays the original dimension table name.

You cannot create dimension branches for the fact table, for the time dimension, or for the accounts dimension. To use multiple tables for the time and accounts dimensions and for the fact table, add the tables directly (see "Adding Source Tables Directly to Dimension Tables" on page 5-7).

➤ To add a table and its columns to form a dimension branch:

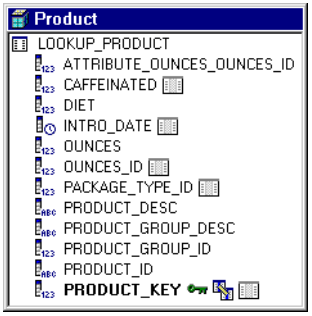
1. Drag the source table from the left frame to the right frame.
2. Join the table to the dimension table as described in "Joining Dimension Tables to Create or Expand a Dimension Branch" on page 8-8.

## Adding Source Tables Directly to Dimension Tables

When you add a new table directly to a dimension table, you combine the two tables. Combining tables offers both advantages and disadvantages, as illustrated in Figure 5-5.

In both the OLAP model and the metaoutline, the original sources of the columns are not as clear as they would be if the new table were part of a dimension branch, but, in the OLAP model, there are fewer tables. In the metaoutline, you can add columns in any order (that is, you are not required to duplicate the OLAP model hierarchy).

**In an OLAP model**




**In a metaoutline**

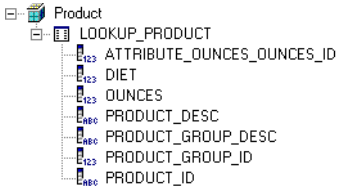



*Figure 5-5: Displaying Table and Column Information from Multiple Tables Combined in a Single Dimension Table*

When you add a new source table directly to an existing dimension table, you must join columns between the two tables. If a column name is used in both tables, the name of the added table is prefixed to the duplicate name from the added table. For example, if in Figure 5-5, both the LOOKUP_PRODUCT and ATTRIBUTES_OUNCES tables contain a column named DIET, the DIET column of the ATTRIBUTES_OUNCES table is renamed as ATTRIBUTE_OUNCES_DIET.

**Note:** To add tables to the fact table, the time dimension, and the accounts dimension, you must combine tables; you cannot create dimension branches.

➤ To add a source table to a dimension table:

1. Drag the source table from the left frame to the right frame, position it directly over the dimension table to which you want to add it, and release the mouse button.

   A dialog box prompts you to make sure that you want to add the columns to the existing dimension table.

2. To add all columns of the new dimension table to the existing dimension table, click Yes.

   Hyperion Integration Server displays the **Physical Joins** tab in the **Table Properties** dialog box, as shown in Figure 5-6.

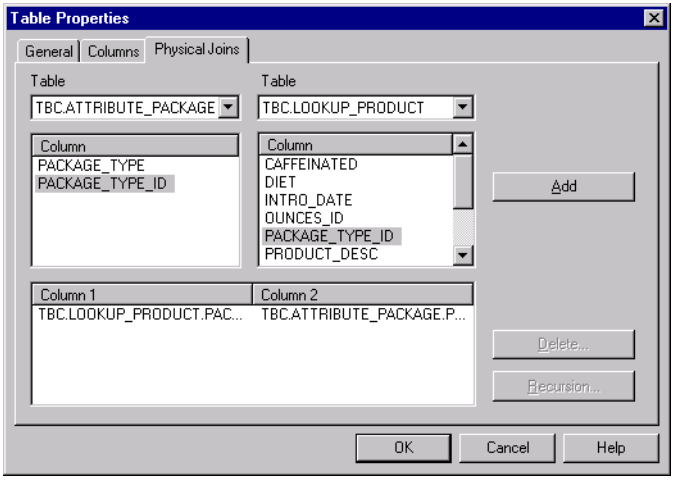



*Figure 5-6: Adding a Table to a Dimension Table*

The left **Table** drop-down list contains the name of the table that you just added. The right **Table** drop-down list contains the names of all tables currently included within the dimension table, including the name of the newly added table. The left and right **Column** lists contain the names of the columns associated with the two selected tables. If a predefined primary-foreign key relationship exists between the tables, the join columns are displayed under **Column 1** and **Column 2** at the bottom of the dialog box.

3. If the column joins listed in the join information box are correct and no joins are missing, proceed to step 5.
4. If you need to join additional columns or if the column joins listed in the join information are not correct, complete the following steps:
   a. In the right **Table** list, select the table with which the newly added table has a relationship.
   b. From the left and right **Column** lists, select from each table the columns that you want to join; for example, PACKAGE_TYPE_ID.

      **Tip:** If you double-click a column in one list, the column with the same name is selected in the other list.

   c. To join the selected columns, click Add.

      The columns that you select are displayed in the box at the bottom of the dialog box.

   d. As needed, repeat steps a. through c. to join additional columns from the selected table.
5. When you finish specifying join information for the selected table, click OK to return to the right frame of the OLAP Model main window.

   If you select the original dimension table to which you have added the new dimension table and then right-click and select View Columns, all of the columns that the new table contains are displayed in the original dimension table.

Now is a good time to hide or delete unwanted columns that the dimension table received from the newly added table and to mark columns as Drill-Through. See "Hiding and Showing Columns" on page 6-6, "Deleting Columns" on page 6-9, and "Defining Drill-Through Columns" on page 6-12.

**Note:** Deleted columns are permanently removed from the OLAP model. If you later decide to include the deleted columns, you must re-create the OLAP model.

## About Adding Tables to the Time Dimension

When adding source tables to the time dimension, remember the following:

- The table that you add must have at least one column with date or time information; for example, a shipping date. The column can be any data type.
- The columns are not added to the fact table. Likewise, when you add a table to the fact table, the columns are not added to the time dimension.

For instructions on how to add tables to the time dimension, see "Adding Source Tables Directly to Dimension Tables" on page 5-7.

## About Adding Tables to the Accounts Dimension

When adding source tables to the accounts dimension, remember the following:

- The source table that you are adding must have at least one measures data column; for example, COGS (cost of goods sold).
- The columns are added to the fact table. Likewise, when you add a table to the fact table, the columns it contains are added to the accounts dimension.
- In the accounts dimension, you can hide or delete non-measures columns after you add a table (see "Hiding and Showing Columns" on page 6-6 and "Deleting Columns" on page 6-9).
- Columns that are hidden or deleted in the accounts dimension do not appear in the fact table; likewise, columns that are hidden or deleted in the fact table do not appear in the accounts dimension.

**Note:** Deleted columns are permanently removed from the OLAP model. If you later decide to include the deleted columns, you must re-create the OLAP model.

For instructions on how to add tables to the accounts dimension, see "Adding Source Tables Directly to Dimension Tables" on page 5-7.

## About Adding Tables to the Fact Table

After you create a fact table, you can add other tables from the relational data source to the fact table. For example, assume that you created the fact table from a sales table in the relational data source. If you want to track inventory and keep inventory data values in a separate table, you need to add to the fact table the necessary columns from the relational data source inventory table.

When adding source tables to the fact table, remember the following:

- When you add a source table to the fact table, the source table columns are added to the accounts dimension. Likewise, when you add a table to the accounts dimension, the source table columns are added to the fact table.
- You can hide or delete columns in the fact table after you add a table (see "Hiding and Showing Columns" on page 6-6 and "Deleting Columns" on page 6-9).
- Columns that are hidden or deleted in the fact table do not appear in the accounts dimension; likewise, columns that are hidden or deleted in the accounts dimension do not appear in the fact table.
- When you add a source table to the time dimension, the source table columns are not added to the fact table. Likewise, when you add a source table to the fact table, the source table columns are not added to the time dimension.

To ensure that you select the correct relational source tables to add to the fact table, view the column names within the tables as described in "Display of Column Names" on page 3-15. If you decide to hide columns after adding a table, check column names after completing the process to make sure that you made the correct selections.

**Note:** Deleted columns are permanently removed from the OLAP model. If you later decide to include those columns, you must re-create the OLAP model.

## Adding Tables to the Fact Table

➤ To add tables to the fact table:

1. Drag the source table from the left frame to the right frame, position it directly over the fact table, and release the mouse button.

   A dialog box prompts you to make sure that you want to add the columns to the fact table.

2. To add the new table to the fact table, click Yes.

   Hyperion Integration Server displays the **Physical Joins** tab in the **Table Properties** dialog box.

3. Join the new table to the fact table as described in "Joining Dimension Tables Directly to the Fact Table" on page 8-6.

Now is a good time to hide or delete unwanted columns that the fact table received from the newly added table. See "Hiding and Showing Columns" on page 6-6 and "Deleting Columns" on page 6-9.

**Note:** Deleted columns are permanently removed from the OLAP model. If you later decide to include the deleted columns, you must re-create the OLAP model.

# Viewing Dimension Table and Fact Table Information

The right frame of the OLAP Model main window contains the fact table and the dimension tables that the OLAP model contains. This topic describes how to view information about the fact table and related dimension tables.

➤ To view properties of a dimension table or of the fact table:

1. Select the dimension table or the fact table, and then select View > Properties > Table.

   Hyperion Integration Server displays the **General** tab of the **Table Properties** dialog box, as shown in Figure 5-7. The **General** tab describes the dimension table—its name, its dimension, its level, and the names of its relational source tables. The fact table is level 0. Dimension tables joined to the fact table are level 1. In a dimension branch, dimension tables joined to level 1 dimension tables are level 2; dimension tables joined to level 2 tables are level 3, and so on.

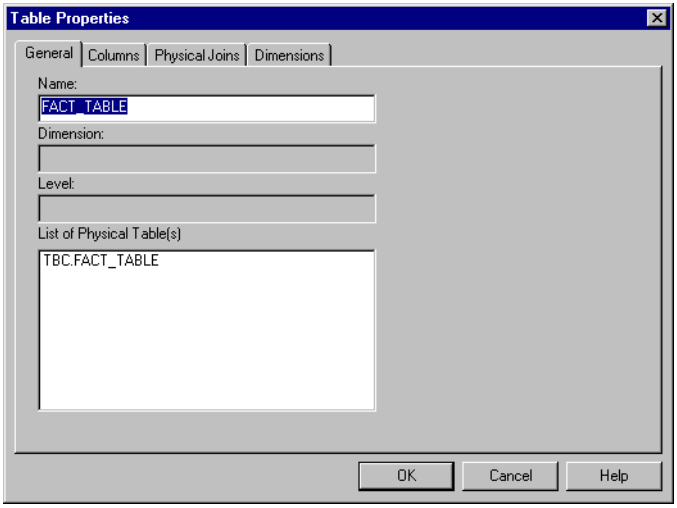



*Figure 5-7: General Tab of the Table Properties Dialog Box*

2. To view the table properties for the fact table, select the **Dimensions** tab to see a list of all dimensions in the OLAP model, as shown in Figure 5-8.

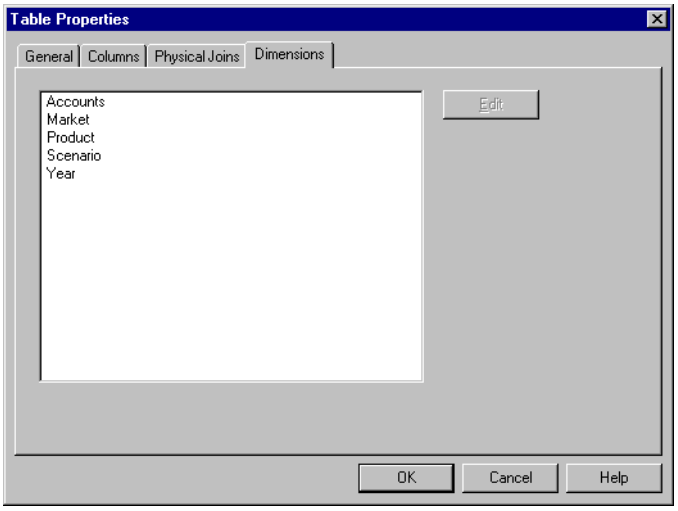



*Figure 5-8: Viewing the Names of Dimensions in an OLAP Model*

# Renaming Dimensions and the Fact Table

When you create an OLAP model fact table or a dimension, the source table name is used by default. This topic describes how to change the name of a dimension or the name of the fact table. For information on how to change a dimension table name, see "Renaming Dimension Tables" on page 5-15.

➤ To rename a dimension or to rename the fact table:

1. In the right frame of the OLAP Model main window, select the dimension or the fact table, right-click, and select Rename.
2. Type the new name for the dimension or the fact table; for example, GEOGRAPHY.

   Follow the rules described in "Observing Naming Rules" on page 3-28.
3. Press Enter to record the new name.

   Hyperion Integration Server displays the new name in the right frame.

# Renaming Dimension Tables

When you create a dimension table, the source table name becomes the dimension table name by default. This topic describes how to change the name of a dimension table. For information on how to change the name of a dimension or of the fact table, see "Renaming Dimensions and the Fact Table" on page 5-14.

➤ To rename a dimension table:

1. In the right frame of the OLAP Model main window, select the dimension table.

   If the dimension table is not visible, drag the corner of the dimension box down until the dimension table is visible.

2. Click the dimension table.

3. Type the new name.

   Follow the rules described in "Observing Naming Rules" on page 3-28.

4. Press Enter to record the new name.

   Hyperion Integration Server displays the new name in the right frame.



# Moving Tables

You can rearrange dimension tables and the fact table in the right frame. For example, you can move a table to allow room to display all columns of the table in the right frame. You can also move tables to display the tables more clearly in the right frame of the OLAP Model main window.

➤ To move a table:

1. Make sure that the join tool is in Move mode, not in Add Joins mode.

   In Move mode, the join tool is not indented,  .

   In Add Joins mode, the join tool is indented,  .

   If the join tool is in Add Joins mode, click the join tool to change it to Move mode.

2. Select the dimension table and drag it to the new location.

   If the dimension table that you move is joined to another dimension table, the join line also moves.

# Deleting Dimension Tables or the Fact Table

Remember these rules when deleting dimension tables or the fact table:

- If you delete a table, its columns are not available for metaoutlines.
- Deleting a table in the OLAP model does not affect the tables of the relational data source.
- When you delete a table, you also delete the joins associated with it.
- If you delete a table that is the primary dimension table of a dimension branch, the other dimension tables of the branch remain in the OLAP model as unjoined dimension tables. When you validate or save an OLAP model with unjoined tables, you receive an error message and you cannot use the model to create a metaoutline.
- You cannot delete tables that are used in hierarchies or metaoutlines.
- If you delete the fact table, the accounts dimension is deleted.
- If you delete the fact table and you created the time dimension from the fact table, the time dimension is deleted.
- If you delete the fact table, the next source table that you drag to the right frame is used to create a new fact table.

➤ To delete a dimension table or the fact table:

1. In the right frame of the OLAP model main window, right-click the dimension table or the fact table, and select Delete.

   Hyperion Integration Server displays a delete confirmation dialog box.

2. Click **Yes** to confirm the deletion.

   The dimension table or fact table is removed from the OLAP model.

Chapter 6

# Editing Columns in Dimension Tables and the Fact Table

Each source table that you drag to the right frame to create a fact table or dimension table contains columns. You can perform operations on columns contained in the tables you create. For example, you can rename, hide, and delete columns and use them to create time hierarchies. This chapter describes how to edit columns and how to create user-defined columns. For information on changing column values, see Chapter 7, "Transforming Columns."

**Note:** For examples of how to enable columns to be attributes, refer to the procedures in Appendix A, "Creating a Sample OLAP Model with Attribute Dimensions." The relational data source, OLAP Metadata Catalog, OLAP model, metaoutline, and data on the example screens in this appendix are based on the sample TBC application provided with Hyperion Integration Server.

This chapter contains the following topics:

# Understanding the OLAP Model Workflow

Figure 6-1 illustrates the workflow for creating an OLAP model. This workflow includes creating and working with an OLAP model; creating a fact table, an accounts dimension, and a time dimension; creating and working with dimensions; editing and transforming the columns of dimension tables; joining dimension tables; and creating and working with hierarchies. This chapter focuses on editing columns in dimension tables and the fact table.

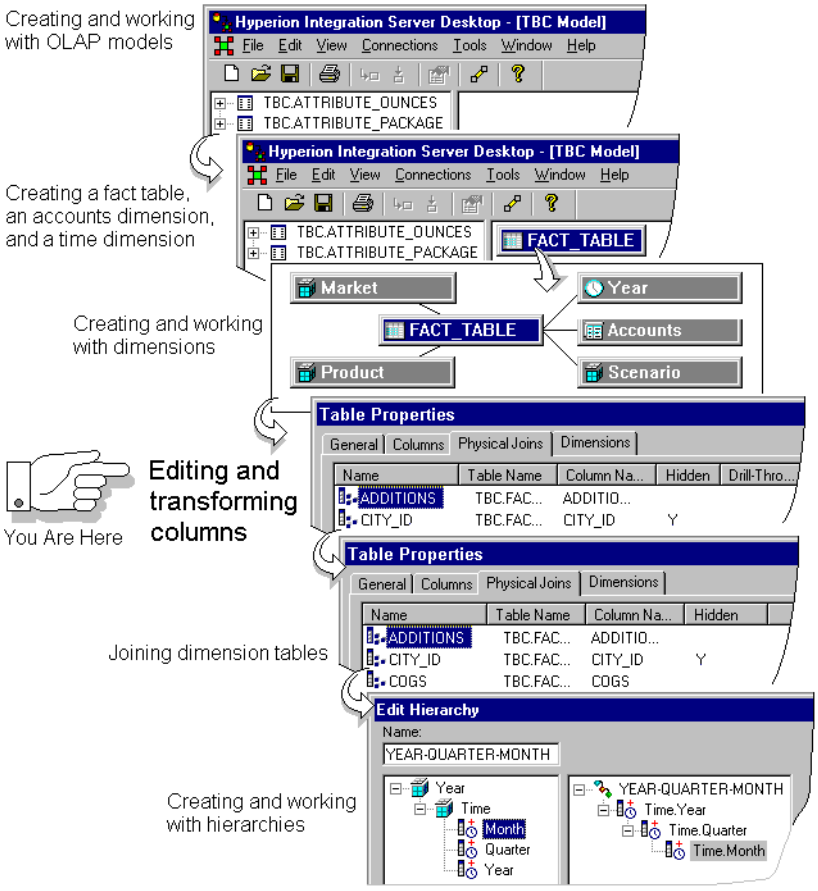



*Figure 6-1: OLAP Model Workflow*

# About Column Data Types

When you drag a source table to the right frame to create a fact table or a dimension table, OLAP Integration Server assigns a data type to each column of the table based on the Open Database Connectivity (ODBC) data type of the column in the source table. Table 6-1 shows commonly used names for these data types. Some relational database management systems (RDBMSs) use different names. Some RDBMSs do not list all types; see the SQL documentation for the RDBMS that you are using.

An OLAP model refers to columns as having numeric, string, or datetime data types.

*Table 6-1: Mapping ODBC Data Types to Hyperion Integration Server Data Types*

| ODBC Data Type | Hyperion Integration Server Data Type |
|---|---|
| CHAR | String. OLAP Integration Server handles CHAR columns that include up to 80 characters. For a CHAR column exceeding 80 characters, add a string column that is based on the CHAR column and use a substring transformation rule to extract 80 or fewer characters (see "Creating User-Defined Columns" on page 6-14). |
| VARCHAR | String |
| DECIMAL | Numeric |
| NUMERIC | Numeric |
| TINYINT | Numeric |
| SMALLINT | Numeric |
| BIGINT | Numeric |
| INTEGER | Numeric |
| REAL | Numeric |
| FLOAT | Numeric |
| DOUBLE | Numeric |

*Table 6-1: Mapping ODBC Data Types to Hyperion Integration Server Data Types (Continued)*

| ODBC Data Type | Hyperion Integration Server Data Type |
|---|---|
| DATE | DateTime |
| TIME | DateTime |
| TIMESTAMP | DateTime |

OLAP Integration Server does not include columns with the following data types:

- LONGVARCHAR
- BIT
- BINARY
- VARBINARY
- LONGVARBINARY

# Viewing Column Properties

In the OLAP Model main window, you can view the name and properties of any column at any time.

➤ To display the names and properties of columns in a dimension table or in the fact table:

1. To display a scrollable list of column names, as shown in Figure 6-2, right-click the dimension and select View Columns.

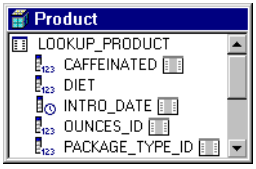



*Figure 6-2: View Columns Option*

The following graphics identify column properties:


identifies a string column.


identifies a measures column in the fact table.


identifies a numeric column in any dimension table other than the fact table.


identifies a datetime column.


identifies a hidden column.


identifies a Drill-Through column.


identifies a join column.


identifies a primary key column.


identifies a column on which a hierarchy filter is defined.


identifies a column on which a column transformation is defined.


identifies a column on which a hierarchy transformation is defined.



2. If you want to turn off the display of all graphics next to columns, deselect View > Model Icons > All.

   You can also choose to turn off the display of individual column graphics; for example, to turn off the display of all primary key graphics in the OLAP model, deselect View > Model Icons > Primary Key.

3. To close the column list, right-click and select View Columns.

# Editing Columns

You can perform a variety of basic editing procedures on columns. You can rename columns, hide and show columns, and delete columns. When you edit columns, you can select an individual column, or you can select multiple columns on which you want to perform the same editing procedure.

**Note:** To learn how to attribute-enable columns in an OLAP model, see Appendix A, "Creating a Sample OLAP Model with Attribute Dimensions."

## Renaming Columns

You can rename any column in a dimension table or in the fact table. Renaming a column in the OLAP model does not affect the relational data source.

➤ To rename a column:

1. In the right frame of the OLAP Model main window, select the column.

   If the column is not visible, drag down the corner of the dimension table containing the column until the column is visible.

2. Select the column name.
3. Type the new column name.

   Follow the rules described in "Observing Naming Rules" on page 3-28.

4. Press Enter to record the new name.

   Hyperion Integration Server displays the new name in the selected table.

## Hiding and Showing Columns

Typically, relational data source tables contain some columns that you do not want to include in a Hyperion Essbase database. Hiding a column removes it from an OLAP model in the sense that the column is not available to a metaoutline as a member level, a measure, an alias, or a Drill-Through column.

You can join dimension tables on hidden columns. You can also show a previously hidden column if you change your mind and want to make the column available to a metaoutline. Hiding columns does not affect the relational data source.

**Note:** In an OLAP model, you cannot hide a column that is used in an associated metaoutline.

➤ To hide or show a column:

1. Choose one of the following options:
   - Select the fact table or dimension table, and then select View > Properties > Table.
   - Drag the corner of a dimension table until its columns are visible, right-click the table name within the column view, and select Properties.

   Hyperion Integration Server displays the **General** tab of the **Table Properties** dialog box.
2. Select the **Columns** tab.

   The **Columns** tab displays all columns available in the OLAP model. For each hidden column, the **Hidden** field contains a Y (Yes), as shown in Figure 6-3.

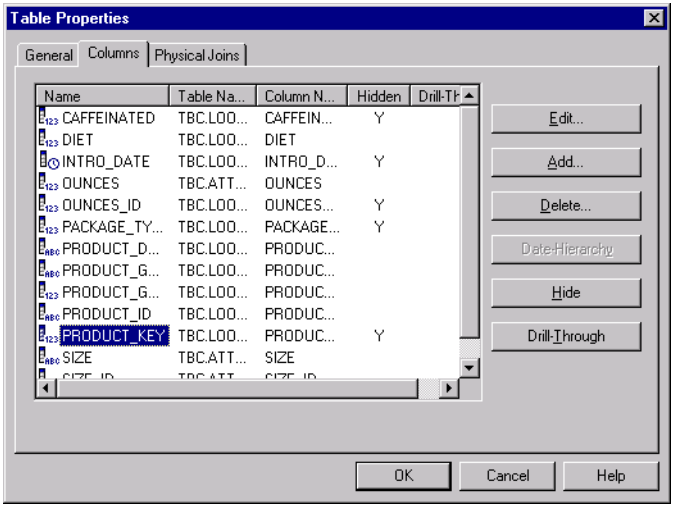



*Figure 6-3: Viewing Hidden Columns on the Columns Tab*

3. Choose one of the following settings:

   - To hide one or more available columns (columns without a Y in the **Hidden** column), select each appropriate column, and click Hide.
   - To show one or more hidden columns (columns with a Y in the **Hidden** column), select each appropriate column, and click Hide to reverse the setting.

   To select multiple columns, see "Working with Multiple Columns" on page 6-8.

   If you select a combination of columns (some with and some without the Y in the **Hidden** column) and then click Hide, all selected columns are hidden.

   If you select a column with a Y in the **Drill-Through** column, the Y is removed from the **Drill-Through** column and placed in the **Hidden** column because columns cannot be both hidden and Drill-Through.

## Working with Multiple Columns

You can apply a procedure to an individual column or to multiple columns. If you want to apply a procedure to multiple columns, use the **Table Properties** dialog box to select the members.

➤ To select multiple, contiguous columns in the **Table Properties** dialog box:

1. Click the name of the first column.
2. Shift-click the name of the last column.

   The first, last, and all columns between are selected.

➤ To select multiple, non-contiguous columns in the **Table Properties** dialog box:

1. Click the name of the first column.
2. Hold down Ctrl and click the individual names of the other columns.

## Deleting Columns

You can delete unwanted columns from dimension tables and from the fact table. Deleting columns is different from hiding columns; deleted columns are permanently removed from the OLAP model. If you later decide to include the deleted columns, you must re-create the OLAP model.

Deleting columns from an OLAP model does not affect the relational data source.

**Note:** You cannot delete columns that are currently used in joins, hierarchies, or associated metaoutlines.

➤ To delete a column from a dimension table or from the fact table:

1. Select the dimension table or the fact table, and then select View > Properties > Table.

   Hyperion Integration Server displays the **Table Properties** dialog box.

   **Tip:** You can also delete a column by selecting it in the OLAP Model main window, right-clicking, and selecting Delete.

2. Select the **Columns** tab.
3. Select the column to delete.

   To select multiple columns, see "Working with Multiple Columns" on page 6-8.

4. Click Delete.

   Hyperion Integration Server displays a delete confirmation dialog box.

   **CAUTION:** Use caution when deleting. You cannot undo the deletion of a column.

5. Click **Yes** to confirm the deletion.

   The selected column name is removed from the list of names in the **Columns** tab.

# Using a Column to Create Time Hierarchies

You can create predefined time hierarchies automatically from a single column. For example, you can select a column containing a timestamp, such as `Dec 29 2000 0733AM` and use the values of the timestamp to create a hierarchy that includes the year (2000), quarter (Qtr_4) and month (December).

➤ To transform a date column into a hierarchy:

1. Select the dimension table that contains the column that you want to use to create a hierarchy and then select View > Properties > Table to display the **Table Properties** dialog box.
2. In the **Table Properties** dialog box, select the **Columns** tab.

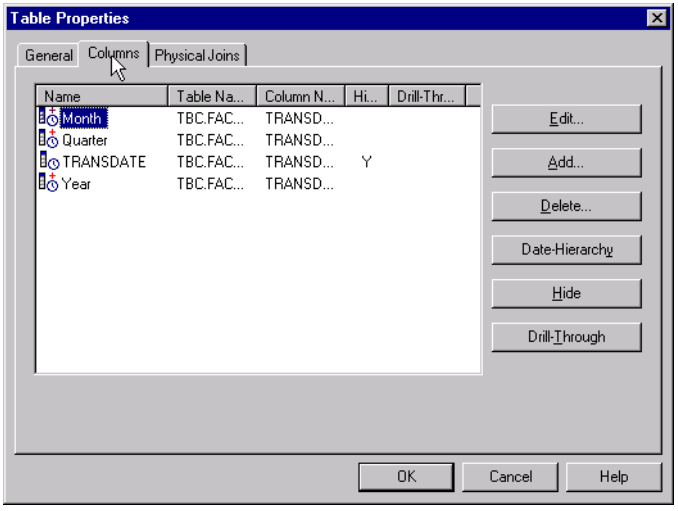



*Figure 6-4: Columns Tab, Table Properties Dialog Box*

3. Select the appropriate date or time column, and click Date-Hierarchy.

   The **Date Hierarchy** dialog box, as shown in Figure 6-5, is displayed.

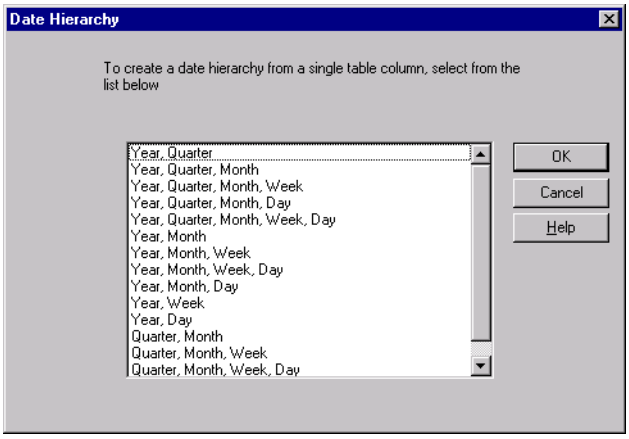



*Figure 6-5: Date Hierarchy Dialog Box*

4. Select the hierarchy to create from the selected column; for example, **Year, Quarter, Month**.

   If you attempt to create a hierarchy that includes a column that already exists within the OLAP model, Hyperion Integration Server displays an error message. You cannot create a hierarchy with a duplicate column name, whether the duplicate column is contained in the source data or in an OLAP model fact table or dimension table (including user-defined columns).

5. To return to the OLAP Model main window, click OK twice.

# Defining Drill-Through Columns

Drill-Through columns are not available as member levels, aliases, or measures in metaoutlines. Therefore, Drill-Through columns are not available to Hyperion Essbase databases. However, Drill-Through columns are available to Hyperion Essbase database users as additional information in reports.

For example, if you set the PRODDESC column as Drill-Through, PRODDESC is not available to the Hyperion Essbase database. However, when Hyperion Essbase Spreadsheet Add-in users drill through to a report that contains the PRODDESC column, they see the product descriptions as a part of the detail. For more information about Drill-Through reports, see the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*.

Use the following guidelines when creating Drill-Through columns:

- You cannot define columns in the accounts dimension table and the fact table as Drill-Through.
- You cannot define a column as Drill-Through if the column is used in an associated metaoutline.
- You can define hidden columns as Drill-Through.

---

**CAUTION:** Do not define columns as Drill-Through if you want to use the columns as aliases, member levels, or measures in a metaoutline.

---

➤ To change the Drill-Through property of a column:

1. Choose one of the following methods to display the **General** tab of the **Table Properties** dialog box:
   - Select the dimension table, and select View > Properties > Table.
   - Drag the corner of a dimension table until its columns are visible, right-click the table name within the column view, and select Properties.
2. Select the **Columns** tab.

   The **Columns** tab displays all columns of the OLAP model, as shown in Figure 6-6. Each Drill-Through column displays a Y (Yes) in the **Drill-Through** field. All numeric columns in the accounts dimension table and the fact table show NA under Drill-Through. An NA indicates that a column is not available as a Drill-Through column.

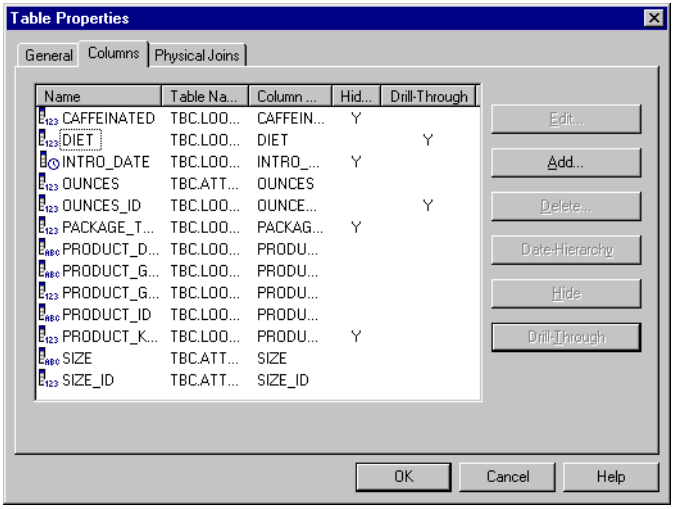



*Figure 6-6: Viewing Drill-Through Columns on the Columns Tab*

3. Select one or more columns for which you want to add or remove the Drill-Through property.

   To select multiple columns, see "Working with Multiple Columns" on page 6-8.

4. Click Drill-Through.

   - If you select only columns that do not have a Y (Yes) in the **Drill-Through** column, each selected column receives a Y.
   - If you select only columns that have a Y in the **Drill-Through** column, the Y of each selected column is removed.
   - If you select a combination of columns that do and do not have a Y in the **Drill-Through** column, each selected column receives or retains a Y (as determined by its previous status).
   - If you select a column that has a Y in the **Hidden** column, the Y is removed from the **Hidden** column and placed in the **Drill-Through** column because columns cannot be both hidden and Drill-Through.

# Creating User-Defined Columns

*User-defined* columns are columns that you create, based on existing columns in the relational database. You may want to add user-defined columns to dimension tables for several reasons. For example, assume that you want to split out a specific portion of the PRODCODE column. To accomplish this, create a SHORTPROD column from the first six characters of the PRODCODE column. As a result, when the value of PRODCODE is 100-10-C001-S0003, the value of SHORTPROD is 100-10.

Another reason to add user-defined columns is to use the columns as levels in a time hierarchy. If you want monthly and quarterly aggregations in the Hyperion Essbase database, you need monthly and quarterly columns in the dimension table for the time dimension. You can add the columns as user-defined columns and derive their values from a single transaction date column contained in the same dimension table.

You must base a user-defined column on existing columns in a dimension table. A user-defined column assumes a numeric, string, or datetime data type, based on the data type of the source column, as discussed in "About Column Data Types" on page 6-3. You cannot change the data type of a column.

Adding user-defined columns to a dimension table includes two major steps:

- Selecting an existing column as the source for the new column and naming the new column
- Applying transformation rules that determine how the data values of the user-defined column are derived from the source column

➤ To add a user-defined column to a dimension table:

1. Select the table that contains the source column on which you want to base a new column.
2. Choose one of the following methods to display the **General** tab of the **Table Properties** dialog box:
   - Select the fact table or dimension table and then select View > Properties > Table.
   - Drag the corner of a dimension table until its columns are visible, right-click the table name within the column view, and select Properties.
3. Select the **Columns** tab, shown in Figure 6-7.

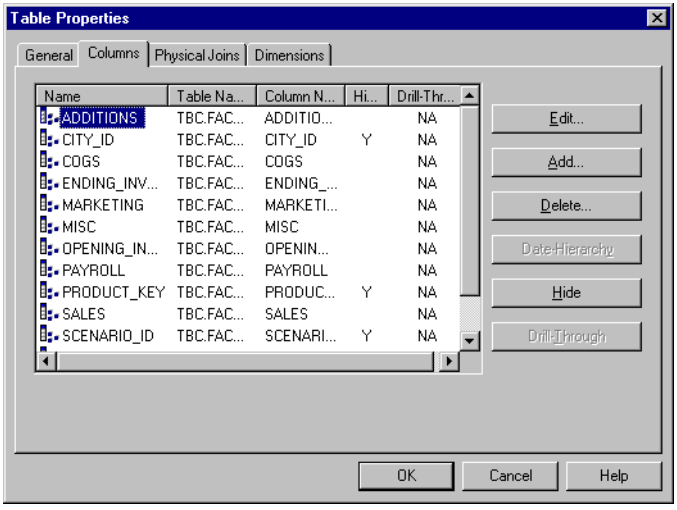



*Figure 6-7: Table Properties Dialog Box, Columns Tab*

4. Click Add.

   Hyperion Integration Server displays the **Add New Column to Table** dialog box, as shown in Figure 6-8.

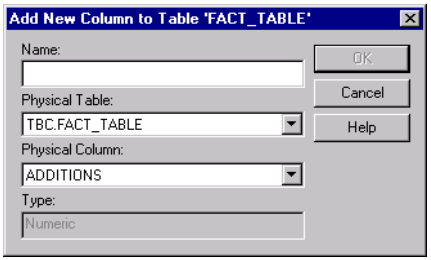



*Figure 6-8: Adding a User-defined Column to a Dimension Table*

5. Enter the name of the user-defined column; for example, NETSALES.

   Follow the rules described in "Observing Naming Rules" on page 3-28.

6. From the **Physical Table** list, select the source table that contains the column on which you are basing the user-defined column; for example, TBC.FACT_TABLE.

7. From the **Physical Column** list, select the source column that contains the data values from which you want to derive the values of the new column; for example, ADDITIONS.

8. Click OK.

Hyperion Integration Server displays the **General** tab of the **Column Properties** dialog box, as shown in Figure 6-9. The tab lists information about the column that you are creating.

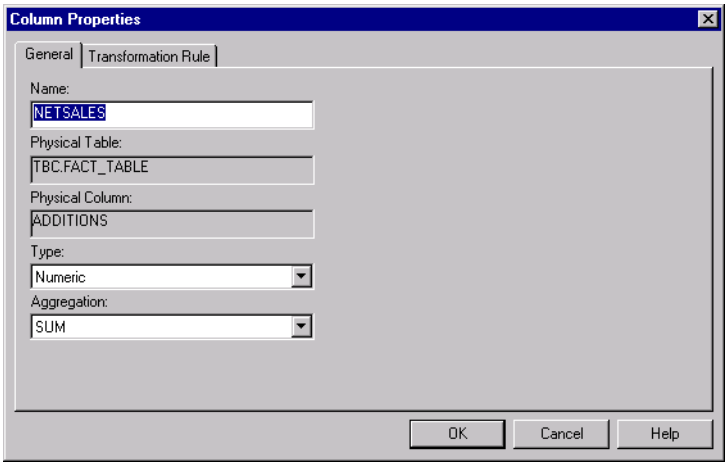



*Figure 6-9: Specifying General Properties for a User-defined Column*

The **General** tab displays the name that you just entered, the names of the source (physical) table and column, and the data type of the column.

If the user-defined column is in the accounts dimension and the column has a numeric data type, the dialog box also contains an **Aggregation** drop-down list. *Aggregation* options are methods for consolidating column data.

9. If the user-defined column belongs to the accounts dimension table, from the **Aggregation** drop-down list, select an aggregation option.

   - SUM: adds the data for each reporting level. SUM is the default selection.
   - <NONE>: prevents a column from being available as measures to the metaoutline. Typically, <NONE> is the aggregation option specified for a join column.
   - AVG, MIN, or MAX: reports, respectively, the average, minimum, or maximum value computed from the data in the column.
   - COUNT: reports, for each reporting level, a count of the source data cells that are not NULL.

10. To define the transformation rule for the user-defined column, select the **Transformation Rule** tab.

    Hyperion Integration Server displays the **Transformation Rule** tab appropriate to the data type of the column with which you are working. For more information about transformations for each data type, see one or more of the following subtopics:

    - "Defining Transformations that Use Substrings to Create String Columns" on page 7-3
    - "Defining Transformations that Use Concatenations to Create String Columns" on page 7-5
    - "Defining Transformations for Numeric Columns" on page 7-8
    - "Defining Transformations for Datetime Columns" on page 7-12
    - "Defining Pass-Through Transformations" on page 7-14

Chapter 7

# Transforming Columns

Use *transformation rules* to define how the data in columns is derived. You can perform transformations that are common to most relational database management systems (RDBMSs) and define transformations that are specific to a particular RDBMS. OLAP Integration Server performs column transformations when it loads members into a Hyperion Essbase outline.

This chapter describes how to transform columns in the right frame of the OLAP Model main window. For information on editing columns, see Chapter 6, "Editing Columns in Dimension Tables and the Fact Table." For information on transforming columns that are part of a hierarchy, see Chapter 9, "Creating and Working with Hierarchies."

This chapter contains the following topics:

# Understanding the OLAP Model Workflow

Figure 7-1 illustrates the workflow for creating an OLAP model. This workflow includes creating and working with an OLAP model; creating a fact table, an accounts dimension, and a time dimension; creating and working with dimensions; editing and transforming the columns of dimension tables; joining dimension tables; and creating and working with hierarchies. This chapter focuses on transforming columns in dimension tables and the fact table.

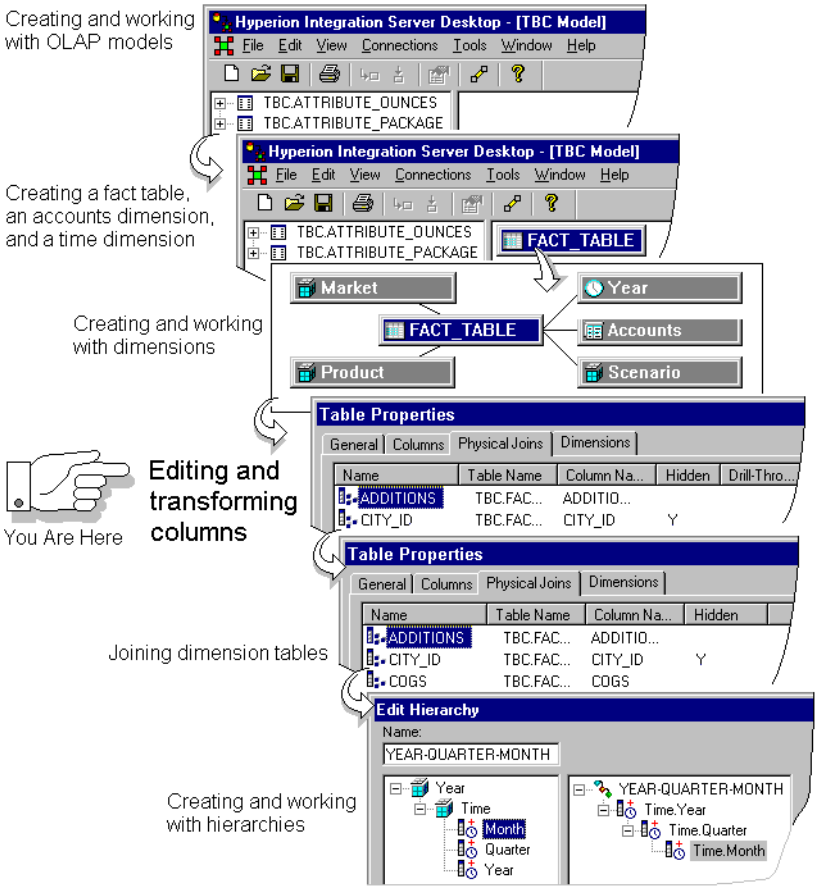



*Figure 7-1: OLAP Model Workflow*

# Defining Transformations that Use Substrings to Create String Columns

You can define a string column as a substring of another column, provided that the two columns are in the same dimension. For example, if only the six characters at the beginning of the PRODCODE column are meaningful to your analysis, you can create a SHORTPROD column that contains only the first six characters. As a result, when the value of PRODCODE is 100-10-C001-S0003, the value of SHORTPROD is 100-10.

To define a transformation rule for a user-defined column, see "Creating User-Defined Columns" on page 6-14.

To use a pass-through transformation, see "Defining Pass-Through Transformations" on page 7-14.

➤ To define a transformation rule that uses a substring to create a string column:

1. Select the appropriate dimension table and then select View > Properties > Table.
2. In the **Table Properties** dialog box, select the **Columns** tab.
3. Double-click the column that you want to transform to display the **Column Properties** dialog box.
4. In the **Column Properties** dialog box, select the **Transformation Rule** tab.

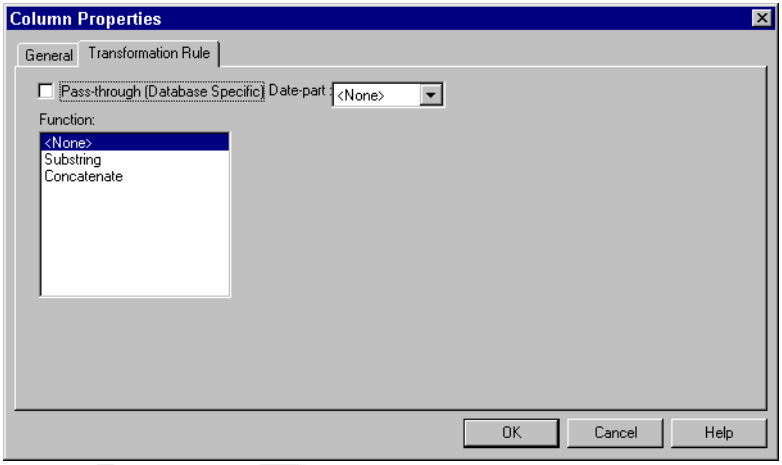



*Figure 7-2: Transformation Rule Tab for a String Column*

The **Function** list displays the options that you can use to derive the new column from the source data.

For numeric and string columns in the time dimension, the dialog box contains a **Date-part** drop-down list. Use the options in the **Date-part** list to use a column to define a Dynamic Time Series member and to report to-date totals, such as week-to-date totals, in the Hyperion Essbase database. For more information about defining Dynamic Time Series members, see the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*.

5. If the column is a time-related column of the time dimension, in the **Date-part** drop-down list, select the part of the date that you want the column to represent.

   For example, if you want the substring to contain the year, select Years.

6. In the **Function** list, select Substring.

   Two text boxes, **Start** and **Length**, are displayed to the right of the **Function** list, as shown in Figure 7-3.

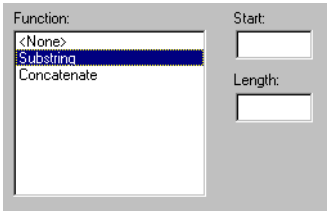



*Figure 7-3: Adding a Column that Is a Substring of Another Column*

7. In the **Start** text box, enter a number that identifies the position in the source column at which you want to start the substring.

   For example, to start at the first position of the product code string, type **1**.

8. In the **Length** text box, enter a number that identifies how many characters from the source column are needed in the new column.

   For example, to use the first six positions of the product code, type **6**.

9. Click OK.

# Defining Transformations that Use Concatenations to Create String Columns

You can define the data in a string column by concatenating one or more string columns provided that all columns are from the same dimension. For example, assume that you want to create a single column, COLA_ID-DESCRIPTION, that contains values from both the PRODUCT_ID and the PRODUCT_DESC columns. When the value of the PRODUCT_ID column is 100-10 and the value of the PRODUCT_DESC column is KoolKola, the value of the concatenated column becomes 100-10-KoolKola.

To define the transformation rule for a user-defined column, see "Creating User-Defined Columns" on page 6-14.

To use a pass-through transformation, see "Defining Pass-Through Transformations" on page 7-14.

➤ To define a transformation rule that uses a concatenation to create a string column:

1. Select the appropriate dimension table and then select View > Properties > Table.
2. In the **Table Properties** dialog box, select the **Columns** tab.
3. Double-click the column that you want to transform to display the **Column Properties** dialog box.

4. In the **Column Properties** dialog box, select the **Transformation Rule** tab.

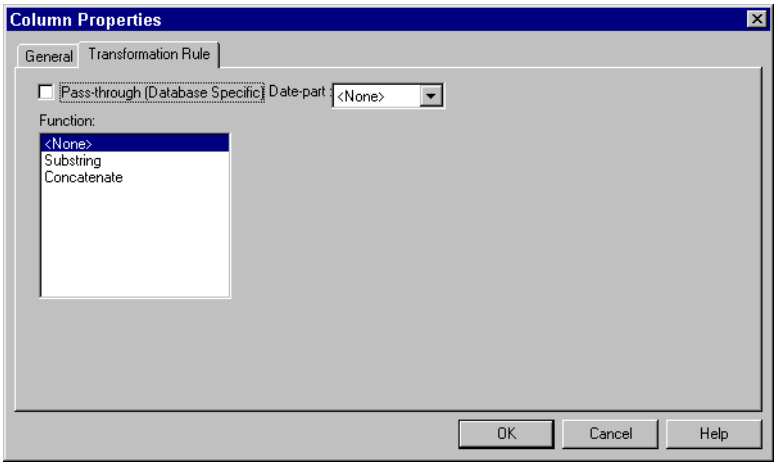



*Figure 7-4: Transformation Rule Tab for a String Column*

The **Function** list displays the options that you can use to derive the new column from the source data.

For numeric and string columns in the time dimension, the dialog box contains a **Date-part** drop-down list. Use the options in the **Date-part** list to use the column to define a Dynamic Time Series member in the metaoutline and thus to report to-date totals, such as week-to-date totals, from the Hyperion Essbase database. For more information about defining Dynamic Time Series members, see the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*.

5. If the column is a time-related column of the time dimension, in the **Date-part** drop-down list, select the part of the date that you want the column to represent.

   For example, if you want the column to contain the year, select Years.

6. In the **Function** list, select Concatenate.

   As shown in Figure 7-5, two list boxes, **Column Names** and **Concate Columns**, are displayed to the right of the **Function** list box.

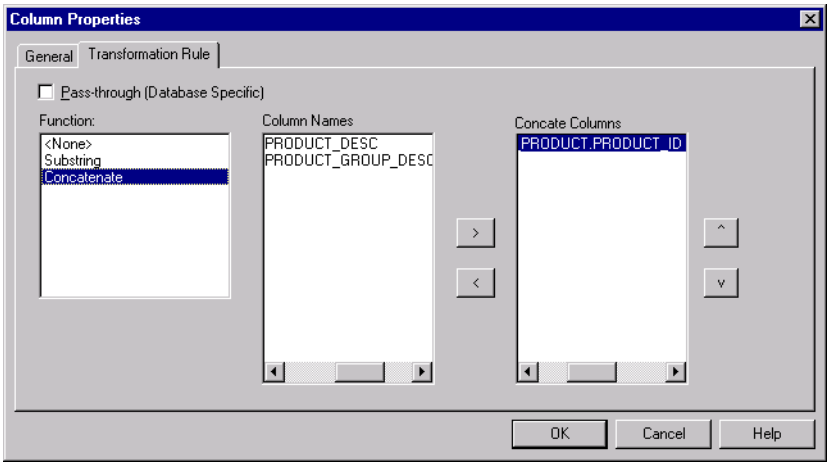



*Figure 7-5: Concatenating Columns*

The **Column Names** list displays the name of the source table (PRODUCT) as a column name prefix and the names of the columns that you can include in the concatenated column (DESC and GROUP_DESC). The **Concate Columns** list initially displays the source column you selected to transform (in this case, PRODUCT_ID).

7. In the **Column Names** list, select the name of the next column to include in the concatenated column (PRODUCT_GROUP_DESC in this example), and click the > button.

   The new column is placed at the end of the **Concate Columns** list.

   **Tip:** To move more than one column at a time, select multiple columns. See "Working with Multiple Columns" on page 6-8. As the columns move to the **Concate Columns** list, they retain the same sequence of the **Column Names** list.

8. To include additional columns, repeat step 7 for each column.

9. As necessary, complete and repeat one or both of the following actions:
   - To remove a column from the **Concate Columns** list, select the column and click the  button. The selected column returns to the **Column Names** list.
   - To change the sequence of the column names in the **Concate Columns** list, select a column name and then click the  button or the  button until the column is in the desired location.
10. When the **Concate Columns** list displays all columns to be concatenated and they are sequenced in the correct order, click OK.

    In the example shown, which produces a concatenated value of 100-10-KoolKola from the sample database OLAP model columns, the **Concate Columns** list contain the following two columns in the order that they are listed:

    TBC.LOOKUP_PRODUCT.PRODUCT_ID
    TBC.LOOKUP_PRODUCT.PRODUCT_DESC

# Defining Transformations for Numeric Columns

Numeric column transformations are the most common transformations for columns of the accounts dimension table. A numeric column transformation is the result of a calculation that involves a source column and a numeric column that is located in the same dimension. For example, you can add, subtract, multiply, or divide the value of a numeric source column from the accounts dimension table by the value of another numeric column from the accounts dimension table to derive the value that you want to report.

For example, assume that you want to define a transformation rule for a new column named NETPRICE. NETPRICE can be derived from the SALES and DISCOUNTS columns of the TBC sample fact table called TBC.FACT_TABLE. For the transformation rule, OLAP Integration Server uses the source column, in this case, TBC.FACT_TABLE.SALES, as the first operand in the calculation. You specify the subtract operator (-) and the other operand, the TBC.FACT_TABLE.DISCOUNTS column. Thus, the calculation for NETPRICE is as follows:

```
TBC.FACT_TABLE.SALES - TBC.FACT_TABLE.DISCOUNTS
```

As a result, for a row with a value of `15.80` for SALES and `3.50` for DISCOUNTS, the value of NETPRICE is `12.30`.

To define the transformation rule for a user-defined column, see "Creating User-Defined Columns" on page 6-14.

To use a pass-through transformation, see "Defining Pass-Through Transformations" on page 7-14.

➤ To define a transformation rule for a numeric column (a column with a numeric data type):

1. Select the appropriate dimension table and then select View > Properties > Table.
2. In the **Table Properties** dialog box, select the **Columns** tab.
3. Double-click the column that you want to transform to display the **Column Properties** dialog box.
4. In the **Column Properties** dialog box, click the **Transformation Rule** tab.

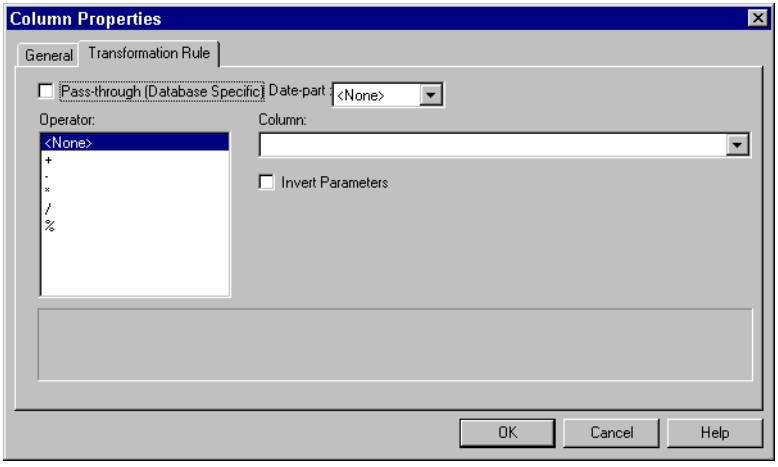



*Figure 7-6: Transformation Rule Tab for a Numeric Column*

7

The **Operator** list displays the types of operations available. The **Column** drop-down list displays the names of all other numeric columns in the dimension table. You can use any one of the numeric columns as the second operand in the calculation.

For numeric and string columns in the time dimension, the dialog box contains a **Date-part** drop-down list. Use the options in the **Date-part** list to use a column to define a Dynamic Time Series member in a metaoutline and thus to report to-date totals, such as week-to-date totals, in the Hyperion Essbase database. For more information about defining Dynamic Time Series members, see the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*.

5. If the column is a time-related column of the time dimension, in the **Date-part** drop-down list, select the part of the date that you want the column to represent.

   For example, if you want the column to contain the year, select Years.

6. In the **Operator** list, select an operator; for example, minus (-).

   By default, the source column is the first operand in the calculation.

   **Note:** In the list of available operators, the % operator is the modulus function. The % operator returns the remainder of a division calculation; the % operator does not stand for percentage.

7. In the **Column** drop-down list, select a column to be the other operand; for example, TBC.FACT_TABLE.COGS.

   The calculation is displayed in the box below the **Operator** box, as shown in Figure 7-7; for example:

   ```
   TBC.FACT_TABLE.SALES - TBC.FACT_TABLE.COGS
   ```

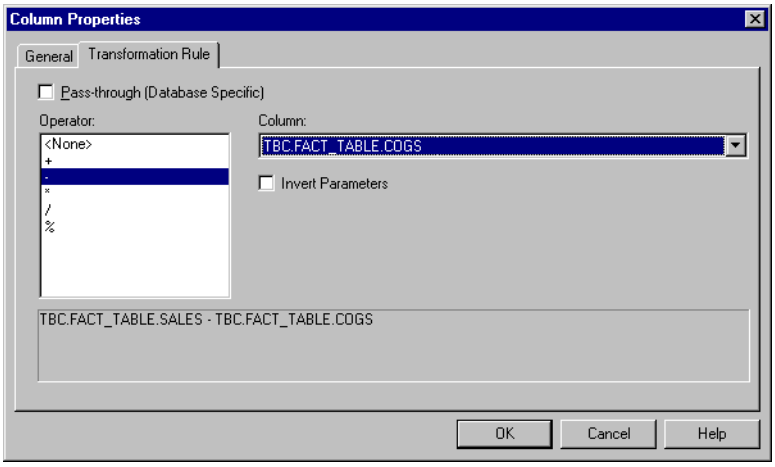



*Figure 7-7: Defining a Transformation Rule for a Numeric Column*

8. As necessary, complete either or both of the following actions:

   - To change the operator, from the **Operator** list, select a different operator.

     The calculation reflects the change.

   - To reverse the sequence of the operands, select **Invert Parameters**.

     The operands switch their positions.

     ```
     hyperion.SALESINVACT.SALES - hyperion.SALESINVACT.COGS
     ```

     changes to

     ```
     hyperion.SALESINVACT.COGS - hyperion.SALESINVACT.SALES
     ```

9. When the calculation is correct, click OK.

# Defining Transformations for Datetime Columns

For a time-related column, select a transformation rule that maps the content of the new column to the appropriate portion of a date, such as the year or month. For more information about columns in the time dimension, see "Creating Time and Accounts Dimensions" on page 2-20.

Typically, time-related columns have a datetime data type. Time-related data can also be stored with a string data type, with or without formatting. For example, a string column can contain the date formatted with slashes, `03/24/2000`, or without slashes, `03242000`. Time-related data can also be stored with a numeric data type, without formatting; for example, `20000324`.

In most cases, you work with columns with a datetime data type in the time dimension table and in the fact table. If you include a column with a datetime data type in another type of dimension, you can use the column as a join column or as a column to be used in Drill-Through spreadsheet reports (see "About Joins" on page 8-3 and "Defining Drill-Through Columns" on page 6-12).

If your fiscal year starts at any month other than January, you must use a user-defined calendar to tie measures data to time-related columns such as month or quarter. For information, see "Creating Time and Accounts Dimensions" on page 2-20.

To define the transformation rule for a user-defined column, see "Creating User-Defined Columns" on page 6-14.

To use a pass-through transformation, see "Defining Pass-Through Transformations" on page 7-14.

➤ To define a transformation rule for a datetime column:

1. Select the appropriate dimension table and then select View > Properties > Table.
2. In the **Table Properties** dialog box, select the **Columns** tab.
3. Double-click the column that you want to transform to display the **Column Properties** dialog box.

4. In the **Column Properties** dialog box, select the **Transformation Rule** tab.

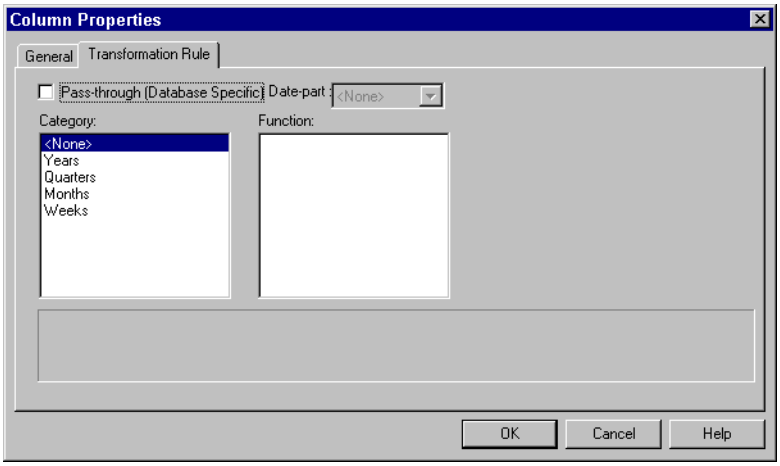



*Figure 7-8: Transformation Rule Tab for a Datetime Column*

5. In the **Category** list, select the time category for the column.

   Select the part of the date that you want the new column to contain. For example, if you select the Quarters category, OLAP Integration Server inserts the quarter of the year into the new column. If the value in the source column is `04/21/2000`, which is in the second quarter, OLAP Integration Server inserts in the new column the value `2`.

   To copy the full date into the column, select <None>.

6. In the **Function** list, select how to represent the date.

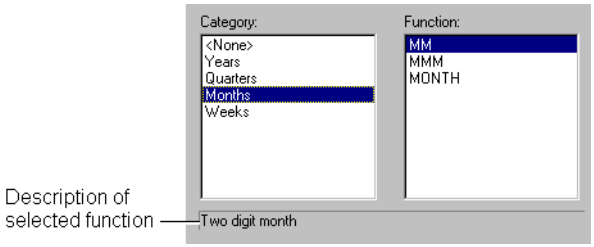



*Figure 7-9: Selecting the Transformation for Month Values*

As shown in Figure 7-9, if you select Months from the **Category** list, the **Function** list displays three possible formats: MM, MMM, and MONTH. If the date value in the source column is `April 15, 2000`, and you select MM, OLAP Integration Server inserts in the new column the value `04`. Select MMM for a three-character abbreviation; for example, `APR`. Select MONTH for a full name; for example, `April`. The text box at the bottom of the **Transformation Rule** tab describes the selected function.

7. Click OK.

# Defining Pass-Through Transformations

If you need to extract data values for a column in ways other than those provided in the Hyperion Integration Server Desktop standard user interface, use SQL functions specific to the source RDBMS. When you define a *pass-through transformation* rule, you provide a statement that OLAP Integration Server passes through as a part of the SQL SELECT statement that it uses to build and load the Hyperion Essbase outline and database.

For example, assume that a new column, SALESVAR, is based on the SALES column. You want the value of the new column to equal the variance of the values in the SALES column. The Hyperion Integration Server Desktop standard user interface does not provide a variance transformation rule because the variance function is not common to all RDBMSs supported by Hyperion Integration Server. However, if the RDBMS you are using supports the variance function, you can use the pass-through transformation rule to calculate the variance of the values in the SALES column.

You can use any function or stored procedure that is valid for the source RDBMS. For example, you can include a stored procedure that determines, based on certain parameters that you include in the procedure, which of two columns to use as an alias column.

You can also use stored procedures to perform calculations related to the database as a whole. Such a stored procedure might reduce the number of rows included in member load and data load. For example, the standard procedure might restrict the load to those products whose profit is greater than 30 percent.

**Note:** If you define a pass-through transformation in an OLAP model, you must use the OLAP model with the RDBMS that you were using when you defined the pass-through transformation.

➤ To define a pass-through transformation:

1. Select the appropriate dimension table and then select View > Properties > Table.
2. In the **Table Properties** dialog box, select the **Columns** tab.
3. Double-click the column that you want to transform to display the **Column Properties** dialog box.
4. In the **Column Properties** dialog box, select the **Transformation Rule** tab.
5. Click **Pass-through (Database Specific)**.

   The **Transformation Rule** tab changes to display a text box, shown in Figure 7-10, for specifying the pass-through transformation.
6. In the text box, enter the command or the information needed to perform the pass-through function.

   You can enter anything that you can validly insert in the SELECT clause of an SQL statement in the source RDBMS. Make sure that the text that you enter is *not* enclosed in quotation marks.

   **Note:** For column names, always enter the user ID, the dimension table name, and the column name. Place periods between the elements; for example, TBC.FACT_TABLE.SALES.

   In the statement that you enter, do not type the word *SELECT*; enter only the text that you want to add to the SELECT clause.

The performance of the pass-through function produces the value of the new column. Here are two examples of using the pass-through feature:

- As shown in Figure 7-10, to derive the variance of the SALES column, type **`VARIANCE(TBC.FACT_TABLE.SALES)`**

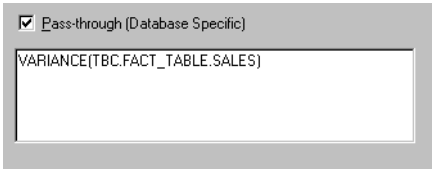



*Figure 7-10: Using a Variance Pass-through Transformation*

- As shown in Figure 7-11, to derive the month portion of a date that is stored in a numeric column (if the source RDBMS supports the % operator as a modulus operator), type

  **`(TBC.LOOKUP_TIME.TRANSDATE%10000)/100)`**

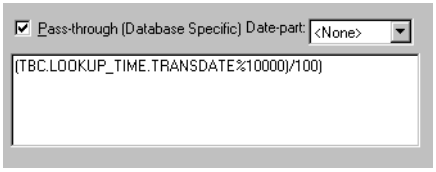



*Figure 7-11: Using a Calculation as a Pass-through Transformation*

**Note:** For some RDBMSs, use the MOD or MODULUS operators instead of the % operator.

7. Click OK.

# Editing Column Transformation Rules

Column transformation rules can be changed. The rules that are available for use vary, depending upon the data type of the selected column. For more information on data types, see "About Column Data Types" on page 6-3.

➤ To change the transformation rule for a column:

1. Select the appropriate dimension table, right-click, and select View Columns.
2. Select the column to edit, right-click, and select Properties.
3. In the **Column Properties** dialog box, select the **Transformation Rule** tab and make the appropriate changes.

   For more information, see one of the following subtopics:

   - "Defining Transformations that Use Substrings to Create String Columns" on page 7-3
   - "Defining Transformations that Use Concatenations to Create String Columns" on page 7-5
   - "Defining Transformations for Numeric Columns" on page 7-8
   - "Defining Transformations for Datetime Columns" on page 7-12
   - "Defining Pass-Through Transformations" on page 7-14

Chapter

8

# Joining Dimension Tables

This chapter describes tasks that you can perform on joins between tables that an OLAP model contains. A *join* establishes a relationship between the data of one table and the data of another table. The join is based on column values common to each table. The two tables can be one fact table and a dimension table, two dimension tables, two source tables consolidated within a dimension table, or a table joined to itself. Join relationships provide information that OLAP Integration Server uses to retrieve data from relational source tables. OLAP Integration Server then uses the data to build Hyperion Essbase outlines and load Hyperion Essbase databases.

**Note:** Tasks that you perform on the joins of an OLAP model do not affect the relational data source.

This chapter contains the following topics:

# Understanding the OLAP Model Workflow

Figure 8-1 illustrates the workflow for creating an OLAP model. This workflow includes creating and working with an OLAP model; creating a fact table, an accounts dimension, and a time dimension; creating and working with dimensions; editing and transforming the columns of dimension tables; joining dimension tables; and creating and working with hierarchies. This chapter focuses on defining relationships between tables through joins.

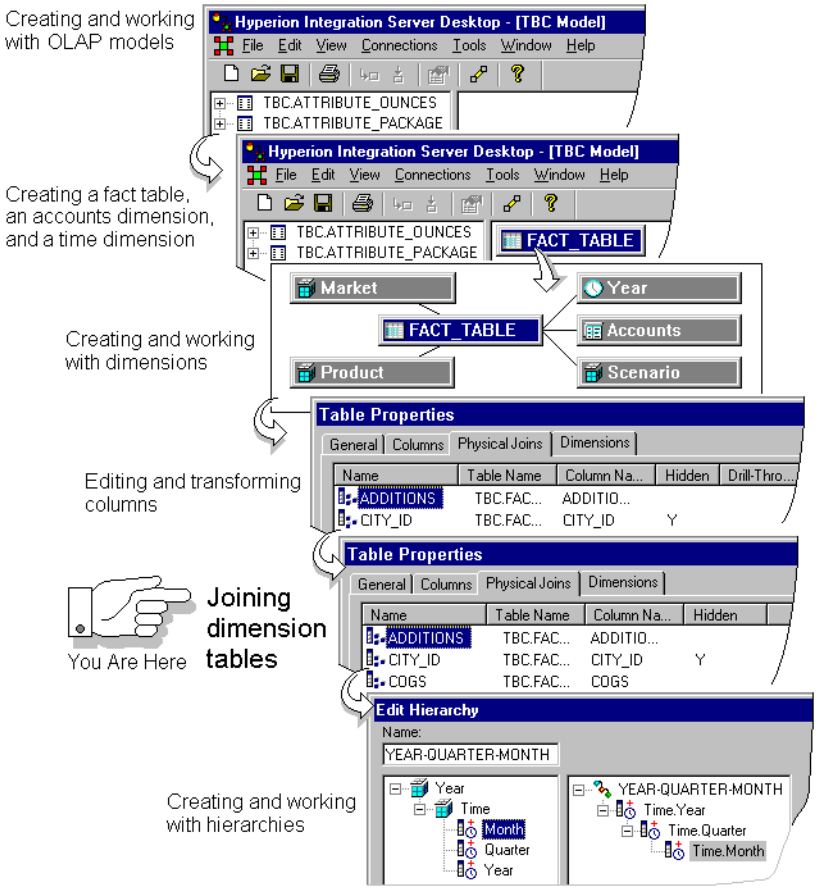



*Figure 8-1: OLAP Model Workflow*

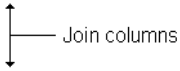
# About Joins

Joins between two tables are formed through related columns called j*oin columns*. The join columns of the two tables represent the same entity. For example, assume that two tables contain information about products. One table stores information about product suppliers. The other table stores information about the number of product units in the warehouse. Because both tables have a column that contains product numbers, you can join the tables by using the product number columns as join columns.

When a data value of a join column matches the data value of the corresponding join column, the information contained in the associated rows of both tables is related. For example, in Figure 8-2, the data values in join columns PRODUCTID and PRODNO match, so all information in the KC-100 row of the PRODUCTS table applies to the KC-100 row of the ORDERS table. Thus, you know that the supplier, Beverage Distrib. supplies the product Kool Cola at `1.10` per case.

**PRODUCTS**

| PRODUCTID | PRODDESC | SUPPLIERID | UNITOFMEASURE |
|---|---|---|---|
| KC-100 | Kool Cola | SUP440 | case |




**ORDERS**

| PRODNO | SUPPLIERID | SUPPLIERDESC | CONTRACTPRICE |
|---|---|---|---|
| KC-100 | SUP440 | Beverage Distrib. | 1.10 |

*Figure 8-2: Join Columns Between Two Tables*

**Note:** The names of the joined columns do not have to match.

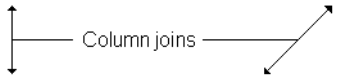
- If the dimension table automatically joins to the fact table, and you want to join the dimension table to a dimension table in a dimension branch, delete the join line between the dimension table and the fact table. You can now join the dimension table to whatever table you select (see "Joining Dimension Tables to Create or Expand a Dimension Branch" on page 8-8).

## Why You May Need to Define Multiple Joins

You may need more than one pair of join columns to define a join relationship. For example, if a product number, as used by two suppliers, indicates two different products, you need to create two pairs of join columns, as shown in Figure 8-3. Each pair of join columns is called a column join. One column join includes the product number columns of both tables. The other column join includes the SUPPLIERID columns of both tables. The second column join ensures unique joins.

**PRODUCTS**

| PRODUCTID | PRODDESC | SUPPLIERID | UNITOFMEASURE |
|---|---|---|---|
| KC-100 | Kool Cola | SUP440 | Case |




**ORDERS**

| PRODNO | SUPPLIERID | SUPPLIERDESC | CONTRACTPRICE |
|---|---|---|---|
| KC-100 | SUP440 | Beverage Distrib. | 1.10 |

*Figure 8-3: Using Multiple Sets of Join Columns*

In this example, for product KC-100 and supplier SUP440, the data of the two rows is related. If more than one supplier supplies Kool Cola, the join on SUPPLIERID ensures that the price and supplier name retrieved from the PRODUCTS table are correct for each supplier.

## How Joins Are Displayed in an OLAP Model

A join line between two tables indicates a join relationship. As shown in Figure 8-4, a solid join line indicates a fully-defined join; that is, the join columns of both dimension tables are specified (see "About Joins" on page 8-3). A broken line indicates that the dimension tables are joined but that the join columns are not specified.

**Note:** If every dimension table is not joined either to another dimension table or to the fact table, the OLAP model is invalid. You can save an invalid OLAP model, but you cannot use an invalid OLAP model to create a metaoutline.

Any dimension table that you join to the fact table becomes a dimension in the OLAP model. Any dimension table that you join to another dimension table becomes part of the dimension of that dimension table. For more information, see "Adding Source Tables to Create Dimension Branches" on page 5-6.

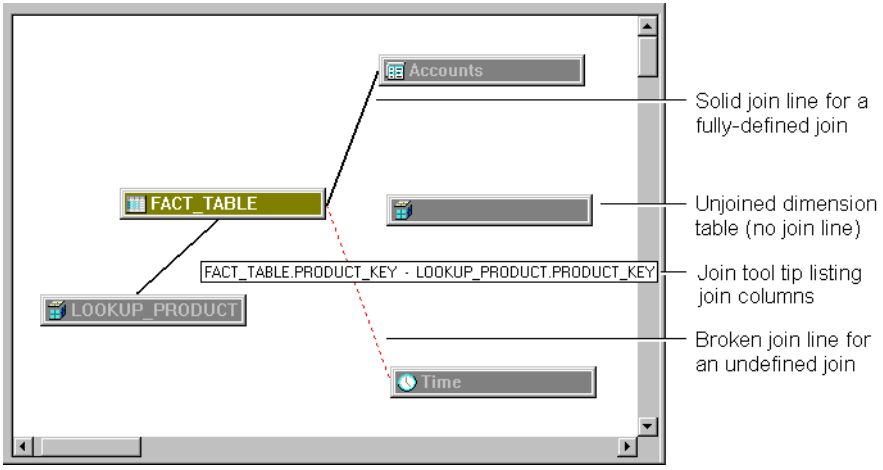



*Figure 8-4: Working with Joins in the OLAP Model*

To see the properties of a join, select the join line, right-click, and select Properties.

To see the join columns in a join, move the cursor over the join line. A tool tip displays the names of the table and join columns. To hide the tool tip, deselect View > Enable Join Tool Tips.

**Note:** If a join relationship exists between tables in the relational data source, the columns join automatically when you add a new source table to an existing dimension table in the OLAP model.

8

# Joining Dimension Tables Directly to the Fact Table

When you join a dimension table to the fact table, the joined table is called the primary dimension table. The primary dimension table becomes a dimension to form a dimension branch. In a dimension branch, only the primary dimension joins directly to the fact table.

➤ To join a dimension table to the fact table:

1. Click the join tool,  , (in the upper right of the OLAP Model main window) to enter Add Joins mode, and draw a line between the dimension table and the fact table.

   **Tip:** Another way to switch to Add Joins mode is to right-click in a blank space of the right frame and select Add Joins Mode.

   Hyperion Integration Server displays the **Create New Dimension** dialog box.

**Note:** If the predefined primary-foreign relationship exists between the table that you drag to the right frame and the fact table, Hyperion Integration Server creates a join automatically and displays the **Create New Dimension** dialog box.

2. Specify the dimension name and dimension type and click OK.

   For more information, see "Creating Dimensions" on page 5-3.

   Hyperion Integration Server displays the **Edit Join Info** dialog box, as shown in Figure 8-5.

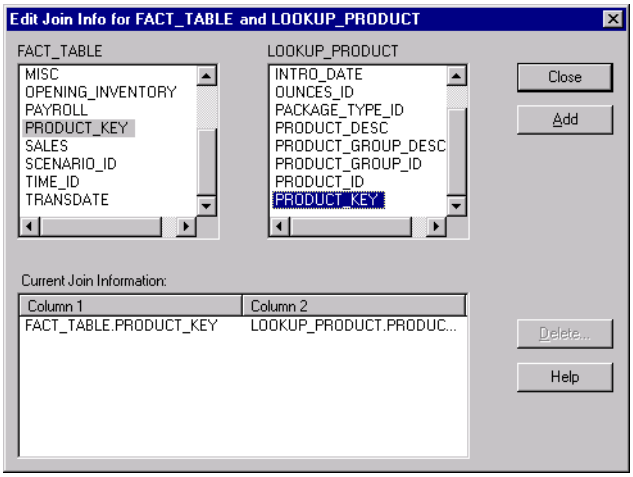



*Figure 8-5: Specifying Column Joins Between the Fact Table and a New Dimension*

Above the left list box is the name of the fact table (in this case, FACT_TABLE). Above the right list box is the name of the table that you joined to the fact table (in this case, LOOKUP_PRODUCT).

Each box lists the columns of its table. If a predefined primary-foreign key relationship exists between the tables that you are joining, the **Current Join Information** list box contains one entry for each column join between the two tables.

For information about deleting joins, see "Deleting Joins Between Dimension Tables" on page 8-14 or "Deleting Joins Within Dimension Tables" on page 8-14.

3. From each table list, select a join column, and click Add.

   For example, select PRODUCT_KEY in both tables. A solid line between the selected columns connects the two tables.

   The columns that you select are displayed in the **Current Join Information** box.

   **Tip:** Double-click a column in one table to select its corresponding column in the other table.

4. For each column join that you want to add to the join definition, repeat Step 3.
5. When the **Current Join Information** box lists all required column joins, click Close.

   **Note:** If you plan to create a recursive hierarchy from a newly-created dimension, the table that you use to create the dimension must have a self-join defined (see "Defining a Recursive Table" on page 8-15).

   If you close the dialog box without defining a column join, the join line displays as a broken line. The OLAP model is invalid. You can save the OLAP model, but you cannot use it to create a metaoutline until all joins are defined. If you see a broken line, create a valid join. See "Working with Joins" on page 8-13.

# Joining Dimension Tables to Create or Expand a Dimension Branch

You can join a dimension table to a primary dimension table, thus forming a dimension branch (see "About Dimension Branches" on page 5-5.) You can also join a dimension table to any table of a dimension branch, thus expanding the branch.

➤ To join two dimension tables:

1. Click the join tool,  , (in the upper right of the OLAP Model main window) to enter Add Joins mode, and drag the tool between the two tables.

   **Tip:** Another way to switch to Add Joins mode is to right-click in a blank space of the right frame and select Add Joins Mode.

   Hyperion Integration Server displays the **Edit Join Info** dialog box for the join, as shown in Figure 8-6.

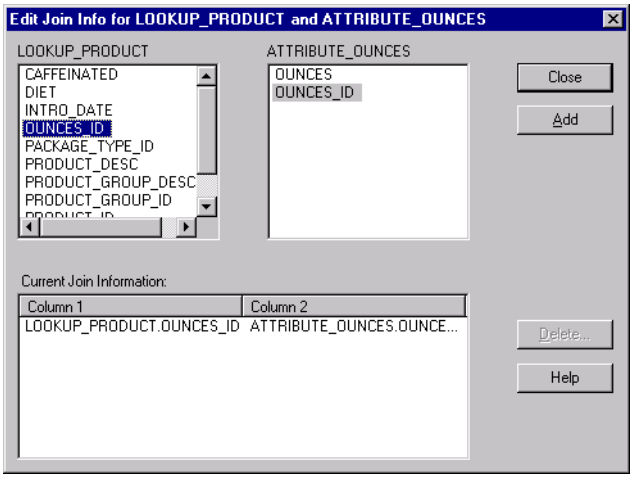



*Figure 8-6: Specifying or Verifying Join Information for a New Join Between Dimension Tables*

At the top of the dialog box are the names of the original dimension table and the table that you are joining to the original table (in this case, LOOKUP_PRODUCT and ATTRIBUTE_OUNCES). The list boxes contain the columns of their respective tables. If a predefined primary-foreign key relationship exists between the tables that you are joining, the **Current Join Information** list box contains one entry for each column join.

For example, in Figure 8-6, the **Current Join Information** list box shows that LOOKUP_PRODUCT table and the ATTRIBUTE_OUNCES table are joined on the OUNCES_ID column in both tables.

2. From each table list, select a column to join, and click Add.

   For example, select OUNCES_ID in both tables.

   **Tip:** Double-click a column in one table to select its corresponding column in the other table.

   The columns that you select are displayed in the **Current Join Information** list box.

3. For each column join that you want to add to the join definition, repeat Step 2.

   You can create column joins in any sequence.

4. When the **Current Join Information** list displays all required column joins, click Close.

   In the right frame of the OLAP Model main window, a solid join line connects the two joined tables.

   **Note:** If you plan to create a recursive hierarchy from a newly-created dimension, the table that you use to create the dimension must have a self-join defined (see "Defining a Recursive Table" on page 8-15).

   If you close the dialog box without defining a column join, the join line displays as a broken line. An OLAP model that contains unjoined tables is invalid. You can save the OLAP model, but you cannot use it to create a metaoutline until all joins are defined. If you see a broken line, create a valid join. See "Working with Joins" on page 8-13.

# Joining Dimension Tables to Combine Table Columns

You can add a source table to a dimension by dragging the source table directly onto an existing dimension table. When you use this process, the columns from the second table are combined with the columns of the existing dimension table. When you drop a dimension table directly onto the fact table or onto another dimension table, you must join the two tables. For more information, see "Adding Source Tables to Create Dimension Branches" on page 5-6.

➤ To add and join a dimension table to another dimension table or to the fact table:

1. In the left frame of the OLAP Model main window, select a source table.
2. Drag the selected source table to the right frame, and drop it on the dimension table to which you want to join it.

   When you add a source table to an existing dimension table or to the fact table, Hyperion Integration Server displays the **Physical Joins** tab of the **Table Properties** dialog box, as shown in Figure 8-7.

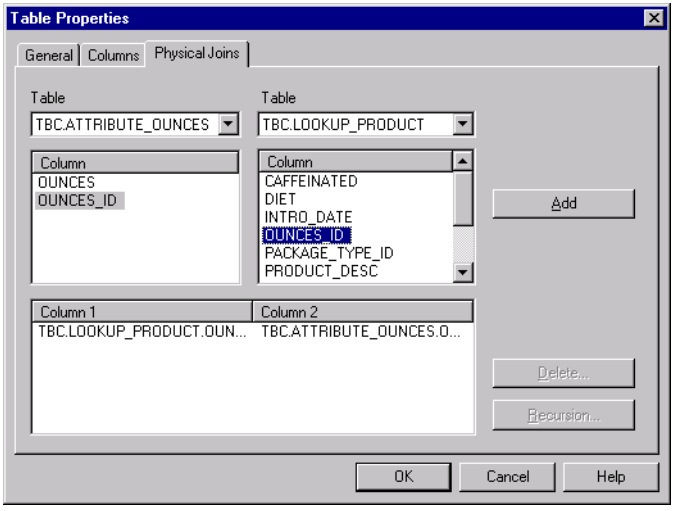



*Figure 8-7: Table Properties Dialog Box, Physical Joins Tab*

Two drop-down lists at the top of the dialog box contain the names of the joined tables. The **Column** lists display the names of the columns of their respective tables. The join information box below lists the column joins.

8

3. If you want to find and correct columns that are incorrectly joined and shown at the bottom of the dialog box or if you want to join columns that are not already joined, complete steps a. and b., as appropriate; otherwise, proceed to step 4.

   a. If the tables to be joined are not already displayed, from the drop-down **Table** lists, select the appropriate tables.

   b. From each **Column** list, select a column to join, and click Add.

   For example, in Figure 8-7, the **Current Join Information** list box shows that the TBC.ATTRIBUTE_OUNCES table and the TBC.LOOKUP_PRODUCT table are joined on the OUNCES_ID column in both tables.

   **Tip:** Double-click a column in one table to select its corresponding column in the other table.

   The columns that you select to join are displayed in the join information box at the bottom of the dialog box.

   If a primary-foreign key relationship exists between one or more columns of the selected tables, the join information box contains one entry for each column join. In this example, the OUNCES_ID columns from the two tables are joined.

4. Repeat Step 3 for each column join at the bottom of the dialog box that you want to add to the join definition.

   You can create column joins in any sequence.

5. When the join information box lists all required column joins, click OK.

# Working with Joins

You can view existing joins and edit them, as needed, to add new joins to existing dimension tables and to the fact table and to delete existing joins.

This topic contains the following subtopics:



## Viewing Joins Information

To view join information for joins between tables, right-click the join line between two dimension tables or between the fact table and a dimension table. Hyperion Integration Server displays the Edit Join Information dialog box. (A different dialog box is displayed if you click the dimension table rather than the join line.)

When the Edit Join Information dialog box opens, you can review the following join property information:

- Names of joined tables
- Names of columns within the joined tables
- Names of joined columns

If you want to add column joins between tables, follow the same procedure as described in Step 2 of "Joining Dimension Tables to Create or Expand a Dimension Branch" on page 8-8.

If you want to delete column joins, see "Deleting Joins Between Dimension Tables" on page 8-14 and "Deleting Joins Within Dimension Tables" on page 8-14.

## Deleting Joins Between Dimension Tables

Deleting a join between two dimension tables or between a dimension table and the fact table results in an unjoined dimension table or dimension branch. An unjoined dimension table or dimension branch makes an OLAP model invalid. You cannot create a metaoutline from an OLAP model that contains a dimension table that is not joined either directly to the fact table or to a dimension table that joins, in series, to the fact table.

➤ To delete a join between two dimension tables or between a dimension table and the fact table:

1. Right-click the join line and select Delete.

   Hyperion Integration Server displays a delete confirmation dialog box.

2. Click **Yes** to confirm the deletion.

   The join line disappears.

   **Tip:** Another way to delete a join between dimension tables is to select the join line and press Delete.

## Deleting Joins Within Dimension Tables

➤ To delete joins within a dimension table:

1. Select the appropriate dimension, right-click, and select View Columns.
2. In the column view, select and double-click the dimension table name.

   Hyperion Integration Server displays the **General** tab in the **Table Properties** dialog box.

3. Select the **Physical Joins** tab.
4. Click the column join that you want to delete; for example, FACT_TABLE.PRODUCT_KEY, and click Delete.

   In the OLAP model, the column join is removed from the table.

   **Note:** If you delete all column joins, an OLAP model becomes invalid. To delete a table from within a dimension table, delete all the columns from the dimension table, as described in "Deleting Columns" on page 6-9.

# Defining a Recursive Table

A recursive table describes the hierarchical relationship of a Hyperion Essbase dimension in just two columns. One column functions as a parent column and the other column functions as a child column. Information in one row relates to information in another row.

For example, assume that a payroll database includes an EMPLOYEES table that contains a child column, EMPLOYEE_ID, and a parent column, REPORTS_TO_ID. As shown in Figure 8-8, the data value (100101) is included in two rows, one row in the child column and another row in the parent column.

| | Parent Column | Child Column |
|---|---|---|
| **EMPLOYEES Table** | **REPORTS_TO_ID** | **EMPLOYEE_ID** |
| | <NULL> | 100101 |
| | 100101 | 273645 |
| | 273645 | 978663 |

*Figure 8-8: Example of Parent and Child Columns in a Recursive Table*

To define a recursive table, you need to join the table to itself. In relational databases, this process is called a self-join. See "Joining a Table to Itself" on page 8-19.

For more information about recursive tables, see "Building Hyperion Essbase Hierarchies from Recursive Tables" on page 2-8.

## Preparing Recursive Tables to Be Used with Aliases and User-Defined Attributes

A recursive table can contain additional information that relates to the parent column or child column values. In Figure 8-9, the additional information provided in the FULL_NAME column and the TITLE column relates to the employees named in the child column, EMPLOYEE_ID.

| | Parent Column | Child Column | Additional Information | |
|---|---|---|---|---|
| **EMPLOYEES Table** | **REPORTS_TO_ID** | **EMPLOYEE_ID** | **FULL_NAME** | **TITLE** |
| | <NULL> | 100101 | John Smith | CEO |
| | 100101 | 273645 | Mary Lee | VP Sales |
| | 273645 | 978663 | James White | Dir Sales, Eastern Region |

*Figure 8-9: Example of a Recursive Table Containing Additional Information*

OLAP Integration Server can use the values of the columns that provide the additional information as aliases or as user-defined attributes (UDAs). A *UDA* is a word or phrase that describes a member level.

In a metaoutline, you can assign an alias column to a parent or child column. OLAP Integration Server then substitutes the value of the alias column for the value of the parent or child column. Aliases improve the readability of Hyperion Essbase outlines or reports. For example, if you use the product description column (PRODUCT_DESC) as an alias for the product code column (PRODUCT_ID), users see the description, Root Beer, instead of the product code, 200-10, on their reports.

In a metaoutline, you can create a UDA for a member level. For example, you might create a UDA called Under Forecast for regions whose sales are below the forecast.

Whether you can define aliases or create UDAs for columns in a recursive table depends on the nature of the data and on how you define the joins for the recursive table. To use aliases and UDAs, you must define joins as follows:

- The information of an alias or a UDA column must be related either to the value in the parent column or to the value in the child column. (You cannot apply an alias or a UDA column to both the parent column and the child column.)
- In most cases, the data in the column for which you are defining the alias or UDA must be fully defined. For example, in Figure 8-9, the child column is fully defined and the parent column is not.

  For more information about fully defined columns in recursive tables, see "Building Hyperion Essbase Hierarchies from Recursive Tables" on page 2-8.

In some cases, the parent and child columns and the columns with additional information are not in the same table. For example, in Figure 8-10, the EMPLOYEE RELATIONSHIPS table and the EMPLOYEE DETAILS table show such a differentiation.

| | Parent Column | Child Column |
|---|---|---|
| **EMPLOYEE RELATIONSHIPS Table** | **REPORTS_TO_ID** | **EMPLOYEE_ID** |
| | <NULL> | 100101 |
| | 100101 | 273645 |
| | 273645 | 978663 |

| | | Additional Information | |
|---|---|---|---|
| **EMPLOYEE DETAILS Table** | **EMPLOYEE_ID** | **FULL_NAME** | **TITLE** |
| | 100101 | Jean Smith | CEO |
| | 273645 | Mary Lee | VP Sales |
| | 978663 | James White | Dir Sales, Eastern Region |

*Figure 8-10: Example of a Recursive Table with Additional Information in a Separate Table*

Table 8-1 describes how to define a recursive table that can be used with aliases or UDAs. In Table 8-1, the Data value characteristics column includes only those situations for which OLAP Integration Server can work with aliases or UDAs.

For more information about aliases and UDAs, see the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*.

*Table 8-1: Defining a Recursive Table Associated with Additional Information in Alias or UDA Columns*

| Data Value Characteristics | When the alias or UDA columns and the parent and child columns are in one recursive table | When the alias or UDA columns and the parent and child columns are in separate tables |
|---|---|---|
| The additional information applies to the parent column *and* the parent column is fully defined. | 1. Define the recursive table with a self-join.<br>2. Hide the child column. | 1. Define the recursive table with a self-join.<br>2. Join the foreign key column of the table containing the alias or UDA column to the parent column of the recursive table.<br>3. Hide the child column. |
| The additional information applies to the child column *and* the child column is fully defined. | 1. Define the recursive table with a self-join.<br>2. Hide the parent column. | 1. Define the recursive table with a self-join.<br>2. Join the foreign key column of the table containing the alias or UDA column to the child column of the recursive table.<br>3. Hide the parent column. |
| The additional information applies to either the parent column or to the child column *and* neither column is fully defined. | You cannot use an alias or a UDA in this situation. | 1. Define the recursive table with a self-join.<br>2. Join the foreign key column of the table containing the alias or UDA column to the parent column of the recursive table.<br>3. Join the foreign key column of the table containing the alias or UDA column to the child column of the recursive table.<br>•Do not hide the parent column.<br>•Do not hide the child column. |

For further details, see:

- "Joining a Table to Itself" on page 8-19
- "Hiding and Showing Columns" on page 6-6

## Joining a Table to Itself

➤ To join a table to itself:

1. In the right frame of the OLAP Model main window, select the dimension table to be joined to itself, and select View > Properties > Table.

   Hyperion Integration Server displays the **Table Properties** dialog box.

2. Select the **Physical Joins** tab, shown in Figure 8-11.

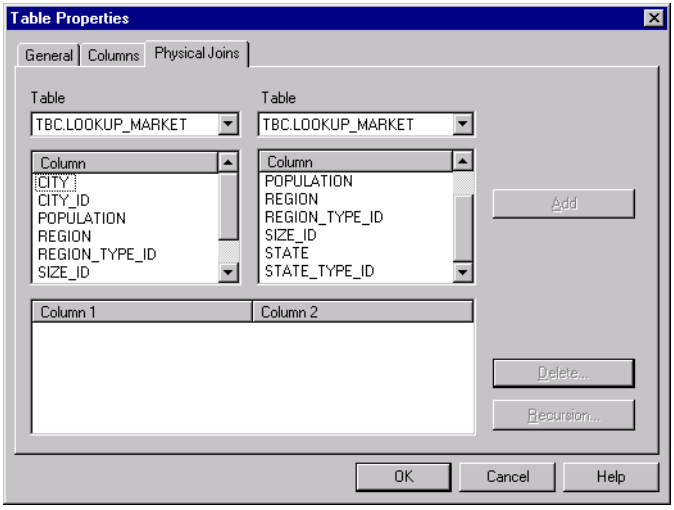



*Figure 8-11: Joining a Table to Itself*

On the **Physical Joins** tab, two **Table** drop-down lists display the name of the physical table that you are joining to itself. Two **Column** lists display the columns in the table.

3. In the **Column** list on the left, select the name of the column representing the lower level of the relationship; for example, as shown in Figure 8-12, CITY.
4. In the **Column** list on the right, select the name of the column representing the higher level of the relationship; for example, as shown in Figure 8-12, STATE.

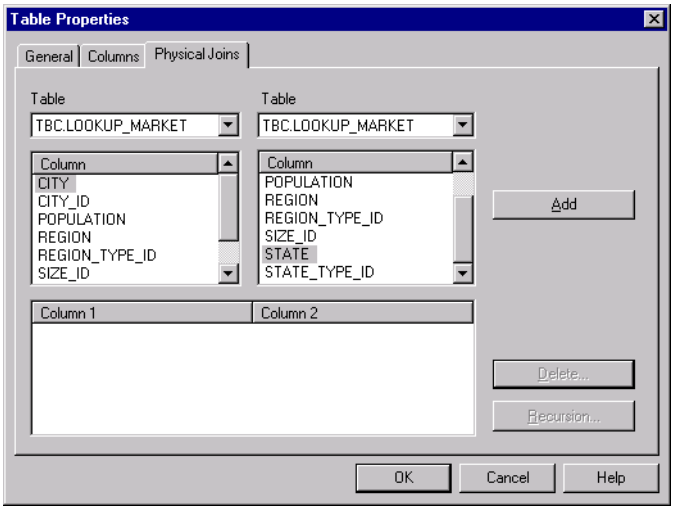



*Figure 8-12: Selecting Join Columns in a Recursive Table*

5. Click Add.

The two join columns are displayed in the **Recursion** dialog box.

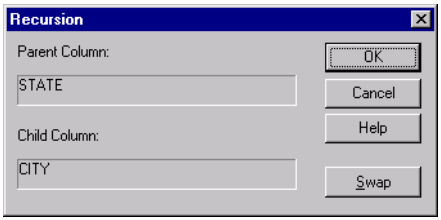



*Figure 8-13: Defining Parent and Child Relationships in a Recursive Table*

If the wrong columns are displayed in the **Parent Column** and **Child Column** list boxes, click Swap to switch the columns.

If you make a mistake, click Swap to switch the columns again.

Hyperion Integration Server displays the **Physical Joins** tab of the **Table Properties** dialog box, with the table and column names in the Current Join Information box, as shown in Figure 8-14.

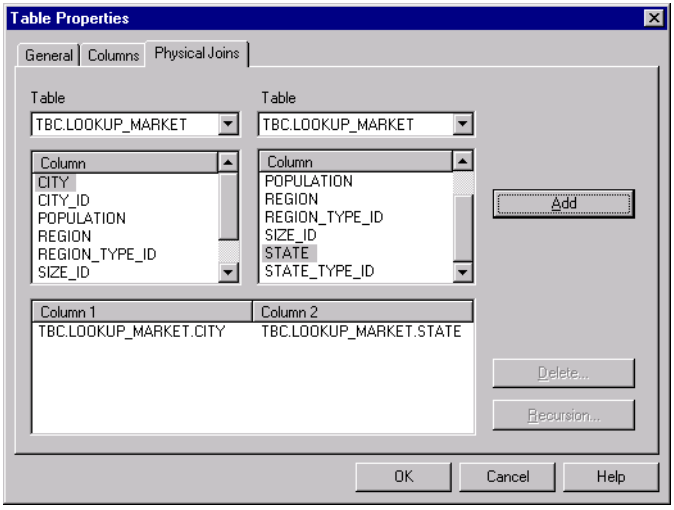



*Figure 8-14: Joined Columns in a Self-Joined Table*

6. Click Close.

**Note:** In most cases, either the Parent Column or the Child Column is the significant column in the table. After defining a self-join for a recursive table, hide the column that is not significant (see "Hiding and Showing Columns" on page 6-6).

---

**CAUTION:** Be sure that the joins are correctly defined. Otherwise, the Hyperion Essbase outline built from the table may not load correctly.

---

Chapter 9

# Creating and Working with Hierarchies

Within a dimension in an OLAP model, you can arrange columns into hierarchies. In an OLAP model, hierarchies define a tree-structure relationship of the data in columns. In a metaoutline, this structure determines the member levels of a dimension and the children of those member levels. You can specify column filters, sorts, and transformations on columns in hierarchies to define how data appears in the Hyperion Essbase outline.

This chapter contains the following topics:

# Understanding the OLAP Model Workflow

Figure 9-1 illustrates the workflow for creating an OLAP model. This workflow includes creating and working with an OLAP model; creating a fact table, an accounts dimension, and a time dimension; creating and working with dimensions; editing and transforming columns of the dimension tables; joining dimension tables; and creating and working with hierarchies. This chapter focuses on defining hierarchies, including creating filters, sorts, and data transformations.

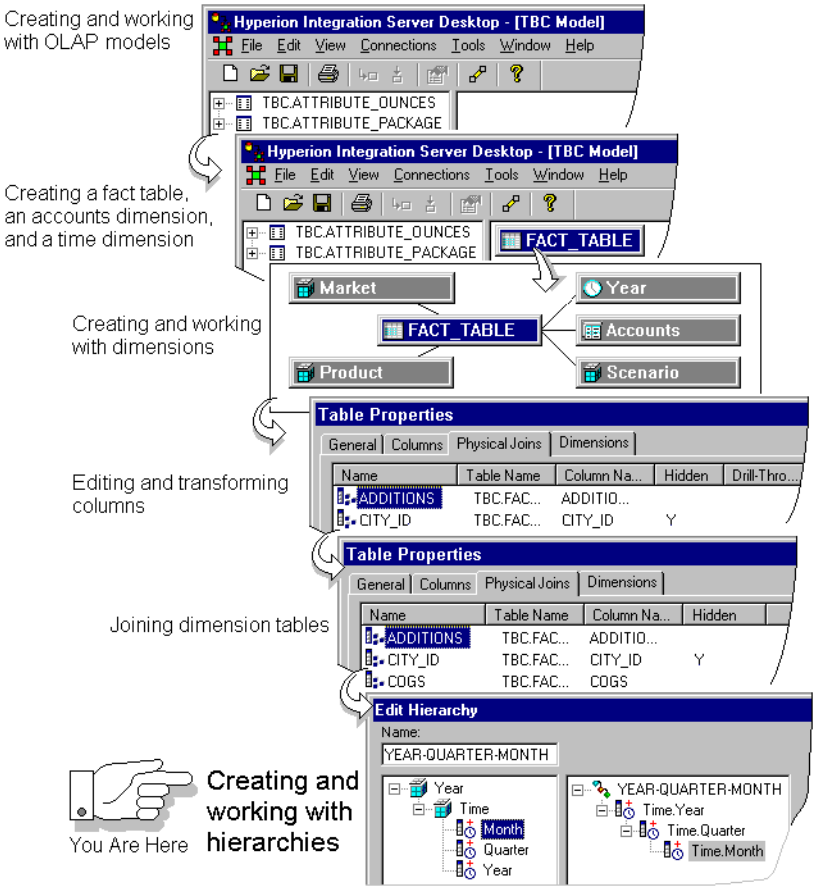



*Figure 9-1: OLAP Model Workflow*

# About Hierarchies

In Hyperion Essbase, hierarchies determine how data is consolidated. For example, many businesses summarize their data monthly, roll up the monthly data to get quarterly figures, and then roll up the quarterly data to get annual figures. Some businesses may also summarize data by zip code, and then by city, state, and country. Any dimension can be used to consolidate data for reporting purposes.

For example, a hierarchy for a GEOGRAPHY dimension in an OLAP model might look like the hierarchy shown in Figure 9-2. In this hierarchy, MARKETS.REGION is a child of the dimension GEOGRAPHY. MARKETS.STATE is a child of MARKETS.REGION, and MARKETS.CITY is a child of MARKETS.STATE.

The structure of the hierarchy carries over to the Hyperion Essbase outline where you can report on sales for individual cities, consolidate city figures to report on sales for states, and consolidate state figures to report on regional sales.


*Figure 9-2: Sample Hierarchy*

Hierarchies provide named structures that contain:

- The hierarchical structure itself—a level-by-level sequence for consolidating data. For example, sales totals by CITY roll up to sales totals by STATE, and sales totals by STATE roll up to sales totals by REGION.
- The data filters that are placed on selected columns within the hierarchy—a way to select specific categories of information. For example, you can filter the REGION column to include only information on sales in the USA.
- The organizational sequence of the data—the sort sequence for a column. For example, you can sort the MONTH column in descending sequence to see the most recent totals first.
- Transformations—a way to control column data values, which become Hyperion Essbase member names. For example, to ensure unique member names, you can prefix each CITY value with an appropriate value from the STATE column. You can then differentiate between NYALBANY and GAALBANY.

After a hierarchy is created in an OLAP model, it can be dragged directly onto a metaoutline. By creating hierarchies in OLAP models, you can customize an OLAP model for each user group. For example, assume that you use hierarchical filters in the OLAP model that you provide to users from a specific corporate division. With such an OLAP model, the users can more easily create metaoutlines that contain only the data relevant to their specific division. For information about how to use hierarchies to create metaoutlines, see the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*.

## About Recursive Tables

A recursive table (parent and child table) contains information in one row that is related to information in another row. For example, assume that you have a recursive table named GEOGRAPHY that contains the columns GEO_CHILD and GEO_PARENT. The fact that the table is recursive indicates that, within each row, the child geographical area is part of the parent geographical area. Any value can serve as both parent and child, a parent in one row and a child in another row.

For example, in Figure 9-3, zip code 01010 is within Bangor, which is within Maine, which is within East. A <NULL> parent value defines the corresponding child value as the highest level in the branch, in this case, USA.

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

*Figure 9-3: Example Recursive Table*

Whenever OLAP Integration Server finds a <NULL> parent, the child value is considered the parent of a new branch. Thus, the data in Figure 9-3 generates the hierarchy shown in Figure 9-4:

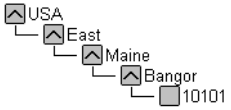



*Figure 9-4: Hierarchy Generated from a Recursive Table*

## About Shared Members in Recursive Tables

When OLAP Integration Server finds a child with two parents, one parent in each of two records, it creates a *shared member*. In Hyperion Essbase, a shared member is a member that shares storage space with another member of the same name. The use of shared members prevents Hyperion Essbase from making extra calculations on members that are in more than one location in the outline.

For example, the relational table shown in Figure 9-5 generates a Hyperion Essbase outline hierarchy where 100-20 (Diet Cola) is a shared member of Diet because it is already a child of 100 (Colas), and 200-20 is a shared member of Diet because it is already a child of 200 (Fruit Drinks).

| Child Column | Child Description | Parent Column |
|---|---|---|
| 100-10 | Regular Cola | 100 |
| 100-20 | Diet Cola | 100 |
| 100-30 | Caffeine-free Cola | 100 |
| 200-10 | Regular Fruit-Orange | 200 |
| 200-20 | Diet Fruit-Orange | 200 |
| 200-30 | Regular Fruit Lime | 200 |
| 100 | Colas | <NULL> |
| 200 | Fruit Drinks | <NULL> |
| Diet | Diet Drinks | <NULL> |
| 100-20 | | Diet |
| 200-20 | | Diet |

*Figure 9-5: Sample Recursive Table Resulting in Shared Members*

The outline hierarchy based on Figure 9-5 includes the shared members 100-20 and 200-20 under Diet, as shown in Figure 9-6:

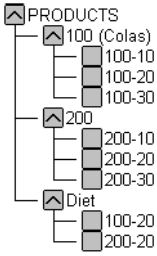



*Figure 9-6: Shared Member Hierarchy Generated from a Recursive Table*

If a relational database contains recursive tables, you need to join each recursive table to itself (self-join) to access any related information and to consolidate the data properly. Detailed procedures for creating a self-join are provided in "Defining a Recursive Table" on page 8-15.

## About Ragged Hierarchies

Ragged (asymmetrical) hierarchies occur when branches within a dimension have different numbers of levels. Figure 9-7 illustrates a ragged hierarchy where the East branch contains four levels, and the West branch contains five levels:

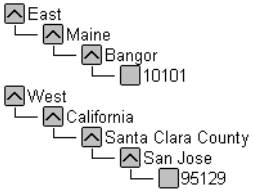



*Figure 9-7: Sample Ragged Hierarchy*

OLAP Integration Server can handle tables that create ragged hierarchies.

# Creating Hierarchies

➤ To create a hierarchy:

1. Double-click the dimension for which you want to create a hierarchy; for example, MARKET.

   Hyperion Integration Server displays the **Dimension Properties** dialog box.

2. In the **Dimension Properties** dialog box, select the **Hierarchies** tab.

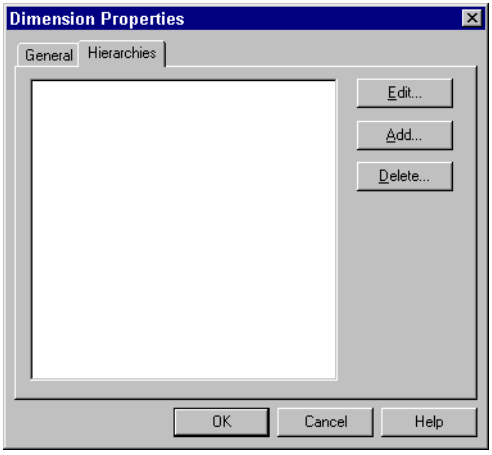



*Figure 9-8: Dimension Properties Dialog Box, Hierarchies Tab*

The box lists any hierarchies that have been defined for the selected dimension.

3. To open the **Edit Hierarchy** dialog box, click Add.

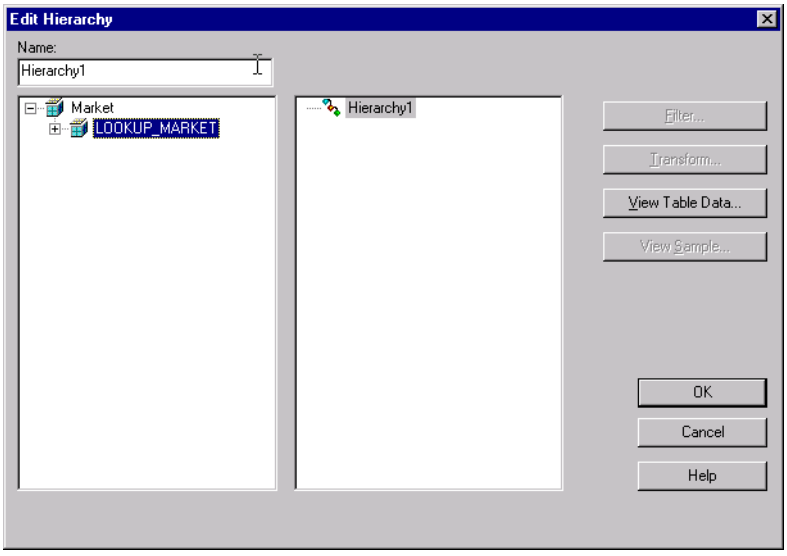



*Figure 9-9: Edit Hierarchy Dialog Box*

The box on the left displays the dimension name above the name of the dimension table. The hierarchy graphic in the right box contains the default name for the hierarchy.

4. In the **Name** text box, enter the name for the hierarchy; for example, Geography.

   The hierarchy name that you enter replaces the default name (Hierarchy1).

5. To display the columns of the selected dimension, in the left box, click the plus sign, ⊞, in front of the table.

   To view the data in a column, in the left box, select a column and click View Table Data.

   For more information, see "Viewing Table Data" on page 3-17.

6. To place a column at the top of the hierarchy, in the left box, double-click the column name.

   The name of the selected column is displayed in the right box immediately below the name of the hierarchy. For example, double-click the REGION column to place it at the top of the hierarchy. Column names in a hierarchy are prefaced with the name of the dimension table to which they belong; for example, MARKET.REGION.

7. To place a column at the next level in the hierarchy, double-click the column.

   The column is displayed in the right box immediately below the previously selected column. In the example shown in Figure 9-10, STATE was selected for the next level.

   Your hierarchy should look similar to the one shown in Figure 9-10.

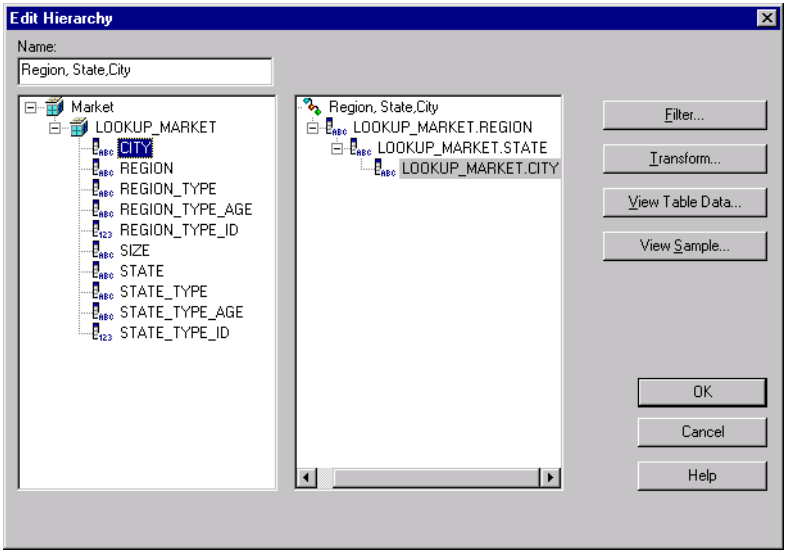



*Figure 9-10: Edit Hierarchy Dialog Box with Completed Geography Hierarchy*

8. To view or perform additional operations on a completed hierarchy, proceed with any of the following options:
   - To create a filter or sort on the selected column, click Filter (see "About Hierarchy Filters" on page 9-14 or "Sorting Data in Hierarchies" on page 9-22).
   - To transform the data in the selected column, click Transform (see "Transforming Data in Hierarchies" on page 9-23).
   - To view the hierarchy as it will appear in the Hyperion Essbase database, click View Sample. For more information, see "Previewing the Hyperion Essbase Outline" on page 9-28.
9. Click OK twice to return to the OLAP Model main window.

# Editing and Deleting Hierarchies

You can delete or edit any existing hierarchy.

➤ To delete a hierarchy:

1. Double-click the dimension that contains the hierarchy; for example, MARKET.
2. When the **Dimension Properties** dialog box opens, select the **Hierarchies** tab.
3. Select the name of the hierarchy.
4. Click Delete to delete the selected hierarchy.

➤ To edit a hierarchy:

1. Complete steps 1 through 3 of the delete a hierarchy procedure.
2. In the **Hierarchies** tab of the **Dimension Properties** dialog box, click **Edit**.

   The **Edit Hierarchy** dialog box opens.

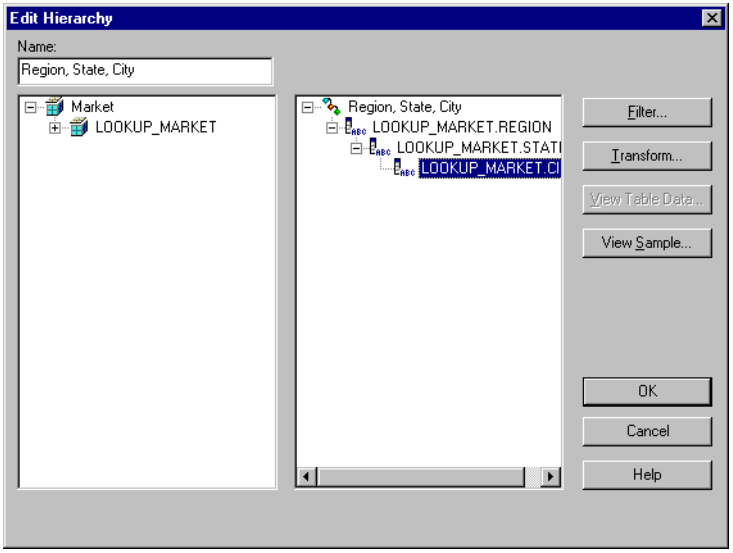



*Figure 9-11: Edit Hierarchy Dialog Box*

   The hierarchy graphic in the right box contains the name of the hierarchy that you are editing; for example, Geography.

3. To display the columns of a dimension table in the left box, click the plus sign, ⊞, in front of the table to expand the view.
4. To view the data in a column, in the left box, select the column and click View Table Data.

   Hyperion Integration Server displays the **View Table Data** dialog box showing the first 100 rows of data in the selected column.

   For more information, see "Viewing Table Data" on page 3-17.

5. To make changes to an existing hierarchy, proceed with any of the following options:
   - To delete a column from the hierarchy, right click the column in the hierarchy and select Delete.
   - To add a column or to move a column to some level above the lowest level in the hierarchy, delete the columns located above the level where you want to add or move a column and then add the columns in the order that you want them listed.
   - To create or edit a filter or sort on the selected column, click Filter (see "About Hierarchy Filters" on page 9-14 or "Sorting Data in Hierarchies" on page 9-22).
   - To create or edit data transformations on the selected column, click Transform (see "Transforming Data in Hierarchies" on page 9-23).
   - To view the hierarchy as it will appear in the Hyperion Essbase database, click View Sample. For more information, see "Previewing the Hyperion Essbase Outline" on page 9-28.
6. Click OK to close the dialog box.

# Filtering Data in Hierarchies

This topic describes how to create filters on columns in hierarchies. You use filters to determine what data values are returned. When transforming data values, OLAP Integration Server performs filters before it performs sorts and transformations.

This topic contains the following subtopics:

## About Hierarchy Filters

You can apply filters to the columns in a hierarchy to restrict the data values that are reported. For example, you can create a filter for each regional office in the United States so that regional managers can view sales totals only for their respective regions. At headquarters, you can view sales for each region and for the total USA.

A filter contains the following components:

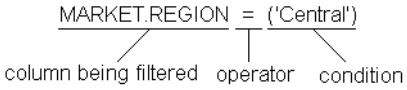



*Figure 9-12: Components of a Filter*

In an OLAP model, you can define a filter only at the column level of a hierarchy. Any filter that you define on a parent value applies to the children of that parent value; for example, if you define a REGION column filter that selects values from the Central region, values from the cities (children) of the Central region are automatically selected.

The filters that you define in an OLAP model are used to create related metaoutlines and are then used to create related Hyperion Essbase outlines.

In a metaoutline, you can define filters for member levels in dimension hierarchies, for member levels created from columns contained in an OLAP model, or for measures contained in an OLAP model. For details, see the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*.

**Note:** If you make changes to a filter in a hierarchy in an OLAP model, the changes do not carry through to metaoutlines previously created from that OLAP model.

You can also define the sequence in which you want to see information. To define the sort order of a column, see "Sorting Data in Hierarchies" on page 9-22.

## About Recursive Table Filters

You can most easily define a filter for a column of a recursive table in the process of defining a metaoutline. For more information, see the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*.

However, one advantage of an OLAP model is the fact that you can select and reuse its predefined dimensions and hierarchies in more than one metaoutline. If you plan to use a filter in more than one metaoutline, you may want to consider creating the filter in an OLAP model. Remember that, to create a filter on a column of a recursive table in a reusable OLAP model, you must first create a hierarchy. You can define filters in an OLAP model only through hierarchies.

When you create a hierarchy from a recursive table, filter on *only* the parent column or the child column. For more information on creating filters, see "Filtering Data in Hierarchies" on page 9-13.

---

**CAUTION:** If you place both the parent and child columns of a hierarchy into an OLAP model, you lose the recursive tie between the rows of the table. Consequently, the table is no longer recursive.

---

To understand what to expect when you filter from a recursive table, consider the following example based on data shown in Table 9-1.

*Table 9-1: Filtering on Columns in a Recursive Table*

| Row Number | GEO_CHILD | GEO_PARENT |
|---|---|---|
| 1 | USA | <NULL> |
| 2 | East | USA |
| 3 | Maine | East |
| 4 | Bangor | Maine |
| 5 | 01010 | Bangor |
| 6 | New York | East |
| 7 | Buffalo | New York |
| 8 | 30092 | Buffalo |

When you filter on a value in a parent column, OLAP Integration Server retrieves only records that include the filter value as the parent. For example, on the GEO_PARENT column, filtering for the value East retrieves the two rows that include East as the parent, rows 3 and 6, which contain Maine and New York in the GEO_CHILD column. No other rows are retrieved or reported on in the Hyperion Essbase database.

When you filter on a value in a child column, OLAP Integration Server retrieves only records that include the filter value as the child. For example, on the GEO_CHILD column, filtering for the value East retrieves only row 2, the row that includes East in GEO_CHILD and USA in GEO_PARENT.

If you filter on the parent column, hide the child column as described in "Hiding and Showing Columns" on page 6-6. If you filter on the child column, hide the parent column.

For more information about working with parent and child relationships in recursive tables, see "About Recursive Tables" on page 9-4.

## Creating Filters in Hierarchies

➤ To create a filter for a column in a hierarchy:

1. In the **Dimension Properties** dialog box, select the **Hierarchies** tab.
   a. Select the hierarchy to filter and click Edit.

      The **Edit Hierarchy** dialog box is displayed.

   b. In the right-hand box, click the plus sign, ⊞, in front of a hierarchy name; for example, Geography.

      The columns in the hierarchy are displayed, as shown in Figure 9-11.

2. From the hierarchy, select the column to filter (for example, MARKET.REGION) and click Filter.

   Hyperion Integration Server displays the **Edit Filter** dialog box, as shown in Figure 9-13. Use the dialog box to specify conditions and operators for the selected conditions. A condition that you apply to a filter is like an SQL WHERE clause.

   If you enter a condition value directly into the **Filters** text box, you must use the syntax rules of the SQL for the relational database management system (RDBMS) that you use.

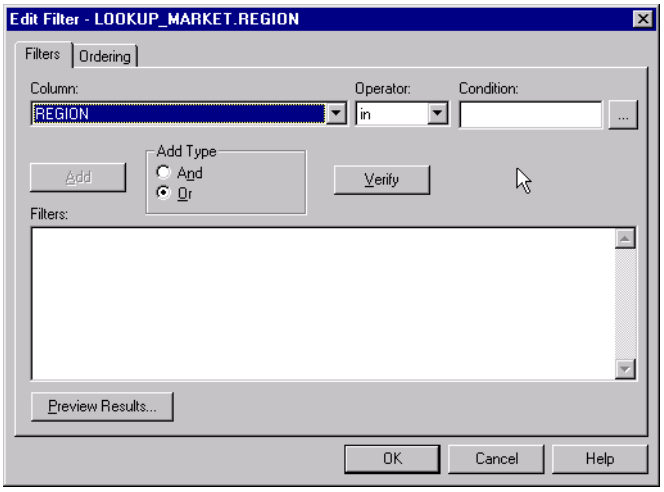



*Figure 9-13: Edit Filter Dialog Box*

The **Column** drop-down box displays the selected column. Notice that the Add button is not selectable; you cannot add a filter until you have selected the condition to use to create the filter, as described in steps 3 and 4.

3. Click the browse button  .

   Hyperion Integration Server displays the **Select Values** dialog box, as shown in Figure 9-14. The **Select Values** dialog box lists the values that you can select.

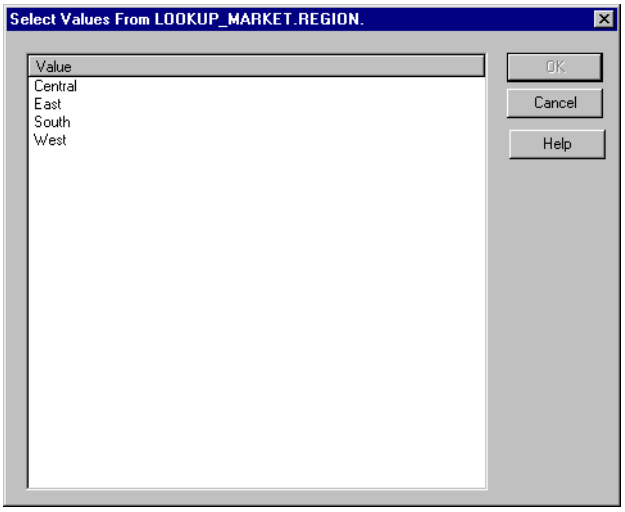



*Figure 9-14: Select Values Dialog Box*

4. Select one or more values to include in the filter; for example, Central, and click **OK**.

   The number of values (conditions) that you can select depends on the type of operator that you want to apply. You can select more than one value (condition) when applying the `in` or `not in` operator. You can select only a single value (condition) when applying any other operator. Available operators are described in Table 9-2.

*Table 9-2: Condition Filter Operators*

| Operator | Description |
|---|---|
| `in` | Use to select one or more values; for example, `Region in United States or Europe` selects regions in both the United States and Europe. |
| `not in` | Use to eliminate one or more values (see `in` operator). |
| `like` | Selects values with similar characters. For example, to select all products that begin with P without specifying all characters in the name, set the condition `Like "P%"` (the wildcard character may vary with the RDBMS). |
| `not like` | Selects values without similar characters (see `Like` operator). |
| `=` | Equal to. |
| `<>` or `!=` | Not equal to. |
| `>` | Greater than. |
| `>=` | Greater than or equal to. |
| `<` | Less than. |
| `<=` | Less than or equal to. |

5. Click **OK**.

   The values (conditions) that you selected are displayed in the **Condition** text box of the **Edit Filter** dialog box.

   The Add button of the **Edit Filter** dialog box is now selectable, as shown in Figure 9-15.

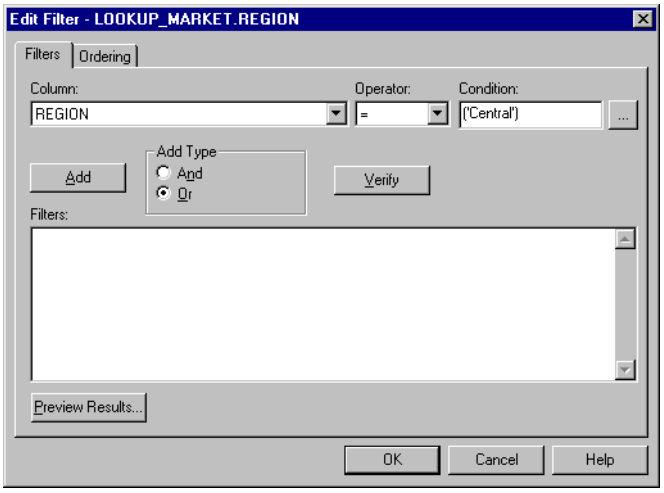



*Figure 9-15: Defining Filters*

6. From the **Operator** drop-down list, select an operator; for example, = (equal to).

   For a description of available Operator options, see Table 9-2.

7. Click **Add**.

   The condition and the operator that you selected are displayed in the **Filters** list box, as shown in Figure 9-16.

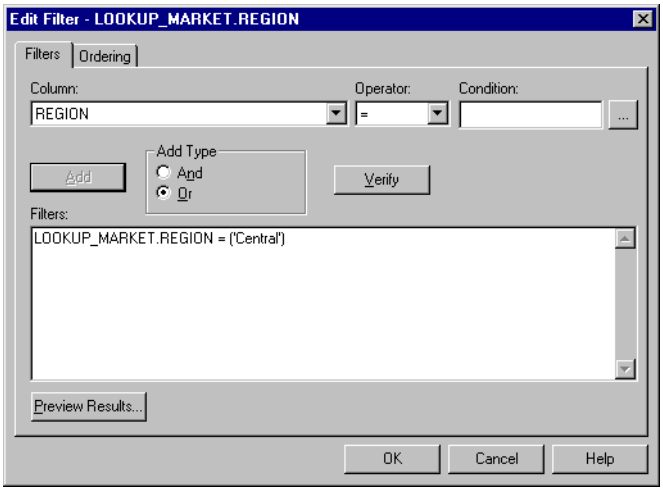



*Figure 9-16: Selected Values Displayed in Filters List Box*

8. To add another condition to the filter, in the **Add Type** box, click **And** or **Or** to determine how the filters are combined, and repeat steps 3 through 7.

   You can group **And** and **Or** clauses by typing them directly into the **Filters** text box. Be sure to use complete SQL WHERE clauses with the valid SQL syntax for the RDBMS that you use.

9. To view the data that will be returned by the filter, click Preview Results.

   For more information, see "Previewing the Hyperion Essbase Outline" on page 9-28.

10. To apply the filter to the column and return to the **Edit Hierarchy** dialog box, click **OK**.

# Sorting Data in Hierarchies

You can specify a sequence for sorting the values of any column. You can select ascending, descending, or none (no sort order). The default sort order is none. If you select ascending or descending, data is retrieved faster than if you select none for no sort order. If you make a change to a sort in an OLAP model hierarchy, the changes do not carry through to metaoutlines previously created from that OLAP model.

**Note:** While retrieving data, OLAP Integration Server sorts columns after it filters and before it performs transformations. For example, if the MONTH column contains numeric values for the month (01, 02, 03, and so on) and you specify a transformation that substitutes month abbreviations, (Jan, Feb, Mar, and so on), OLAP Integration Server sorts columns by their pre-transformation numeric values. April totals are displayed after March totals because `04` sorts after `03`.

➤ To define a sort order for column values:

1. Open the **Hierarchies** tab of the **Dimension Properties** dialog box.
   a. Select a hierarchy and click Edit.

      The **Edit Hierarchy** dialog box is displayed.

   b. To expand the view, in the right box, click the plus sign,  , in front of the hierarchy (for example, Geography).
2. To display the **Edit Filter** dialog box, select the column for which you want to sort values, and then click Filter.
3. Select the **Ordering** tab.
4. From the **Sort Order** drop-down list, select Ascending, Descending, or None.
5. Click **OK** three times to return to the OLAP Model main window.

   To change the way data values are represented when reported by Hyperion Essbase, see "Replacing Data" on page 9-23.

# Transforming Data in Hierarchies

After you create a dimension hierarchy, you can change the way the data values of the hierarchy are represented when reported by Hyperion Essbase. For example, perhaps the date values of the OLAP model hierarchy are numeric and you want the corresponding Hyperion Essbase members to be alphabetic.

A transformation that you define on a hierarchy in an OLAP model is used to create first a metaoutline and then a Hyperion Essbase outline. If you change a transformation in an OLAP model, the change does not carry through to metaoutlines previously created from the OLAP model. OLAP Integration Server performs filters and sorts before transformations.

For more information about transforming data in hierarchies, see:

- "Replacing Data" on page 9-23
- "Prefixing and Suffixing Column Values" on page 9-26

## Replacing Data

You can change column data by replacing current values with new values, changing capitalization, and removing leading and trailing spaces.

➤ To transform data values in a hierarchy:

1. Double-click the dimension that contains the hierarchy with which you want to work.
2. To display the **Edit Hierarchy** dialog box, select the **Hierarchies** tab, then select the appropriate hierarchy, and click Edit.

3. In the **Edit Hierarchy** dialog box, select from the hierarchy the column for which you want to transform data values (for example, MARKET.CITY), and click Transform.

   Hyperion Integration Server displays the **Replace** tab of the **Edit OLAP Model Transformations** dialog box, as shown in Figure 9-17.

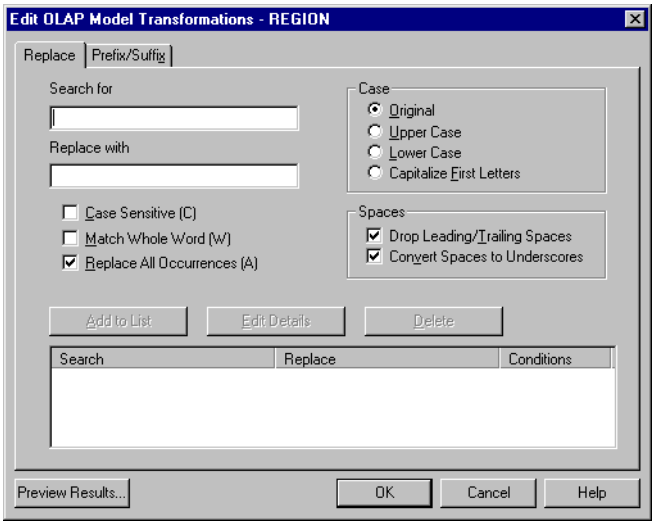



*Figure 9-17: Replace Tab of the Edit OLAP Model Transformations Dialog Box*

4. To replace text, change the capitalization of the text, or replace spaces, follow the steps in Table 9-3, and click **OK**.

*Table 9-3: Editing Text in Hierarchy Data*

| Replacement | Actions |
|---|---|
| Text | 1. In the **Replace** text box, enter the original value and, in the **With** text box, enter the text to replace the value.<br>Specify the conditions for making the replacement by checking one or more of the following options:<br>**Case Sensitive (C)**. Replaces only text strings that match the capitalization of the string in the **Replace** text box.<br>**Replace All Occurrences (A)**. Replaces every occurrence of the text string. For example, if you replace all occurrences of `10` in the string `100 10 1` with an `A`, the string changes to `A0 A 1`.<br>**Match Whole Word (W)**. Replaces the text string only when it occurs as a whole word. For example, when replacing the `10` in the string `100 10 1` with an `A`, if you set the **Match Whole Word** option, the string changes to `100 A 1`.<br>2. Click Add to add the transformation to the value specified. The entries move to the **Replace** box. |
| Capitalization | To change the case of the data value, select one of the following options:<br>• **Original**. Does not change the case. This setting is the default.<br>• **Upper Case**. Changes the value to uppercase; for example, january or January to JANUARY.<br>• **Lower Case**. Changes the value to lowercase; for example, JANUARY or January to january.<br>• **Capitalize First Letters**. Changes the first letter of the value and the first letter following a space or an underscore to uppercase; for example, qtr1_january changes to Qtr1_January. |
| Spaces | • To omit spaces before or after the data value, select **Drop Leading/Trailing Spaces**.<br>• To convert spaces that exist in the original data value to underscores ( _ ), select **Convert Spaces to Underscores**. |

5. To view the data as it will appear in the Hyperion Essbase outline, click **Preview Results**.

   For more information, see "Previewing the Hyperion Essbase Outline" on page 9-28.

6. Click **OK** three times to return to the OLAP Model main window.

## Prefixing and Suffixing Column Values

You can use a prefix or suffix to generate unique member names from the columns of a hierarchy. Hyperion Essbase requires unique member names to report data values accurately. For example, if you use the same month names for multiple years, you can add a year suffix to the month column names of each hierarchy to ensure that Hyperion Essbase reports the correct value for each month of each year.

**Note:** The default setting for columns in a time hierarchy is parent name.

➤ To define prefixes or suffixes for data values in a hierarchy:

1. Double-click the dimension that contains the hierarchy with which you want to work.
2. To display the **Edit Hierarchy** dialog box, select the **Hierarchies** tab, then select the appropriate hierarchy, and click **Edit**.
3. In the **Edit Hierarchy** dialog box, select the hierarchy column that you want to transform (for example, MARKET.CITY), and click Transform.

   The **Edit OLAP Model Transformations** dialog box is displayed.

4. Select the **Prefix/Suffix** tab.

   Hyperion Integration Server displays the **Prefix/Suffix** tab, as shown in Figure 9-18.

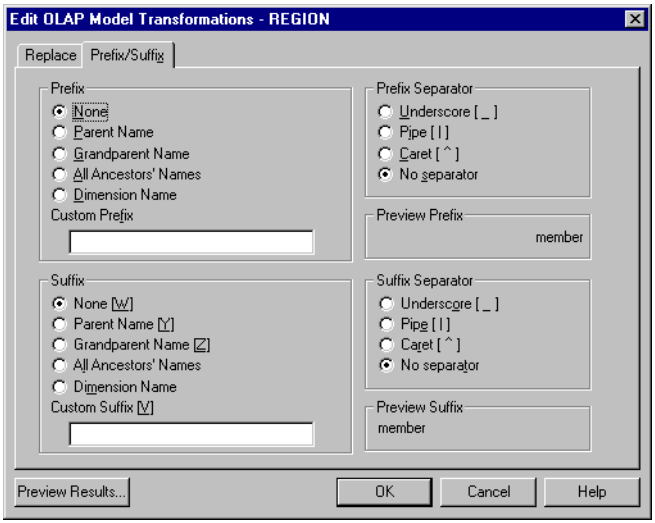



*Figure 9-18: Prefix/Suffix Tab*

5. To add a prefix or suffix to a data value, select the value to prefix or suffix, as listed in Table 9-4.

*Table 9-4: Prefix and Suffix Values*

| Prefix/Suffix | Definition |
|---|---|
| None | Uses the column value as the full name. Nothing is attached. |
| Parent Name | Attaches the immediate parent; for example, Florida_Miami. |
| Grandparent Name | Attaches the parent and the parent of the parent; for example, Central_Florida_Miami. |
| All Ancestors' Names | Attaches all parents; for example, Market_Central_Florida_Miami. |

*Table 9-4: Prefix and Suffix Values (Continued)*

| Prefix/Suffix | Definition |
|---|---|
| Dimension Name | Attaches the dimension name; for example, Market_Miami. |
| Custom Prefix<br>Custom Suffix | Attaches your own prefix or suffix. Type the prefix or suffix in the text box; for example, **SKU_**. |

6. To insert a character between the prefix or suffix and the member value, in the **Prefix Separator** or **Suffix Separator** box, select the character to insert.

   For example, to prefix the value for Year to the quarter members and insert an underscore between the prefix and the quarter member, select **Parent Name** and **Underscore**. This procedure creates member names like 2000_Quarter4.

7. To view the data as it will appear in the Hyperion Essbase outline, click **Preview Results**.

   For more information, see "Previewing the Hyperion Essbase Outline" on page 9-28.

8. Click **OK** three times to return to the main window.

# Previewing the Hyperion Essbase Outline

You can preview the Hyperion Essbase outline that will be created from a hierarchy. The preview includes the results of filtering and transformation operations. For example, you can create a filter and then view the column values that will be returned by the filter.

➤ To preview the Hyperion Essbase outline:

1. Choose one of the following options:

   - If you are creating a filter in the **Edit Filter** dialog box, click **Preview Results**.
   - If you are creating a transformation in the **Edit OLAP Model Transformation** dialog box, click **Preview Results**.
   - If you are creating a hierarchy in the **Edit Hierarchy** dialog box, click View Sample.

Hyperion Integration Server displays a dialog box similar to the one in Figure 9-19. You can click the plus sign, ⊞, in front of each member to expand and view the full member hierarchy within the sample outline.

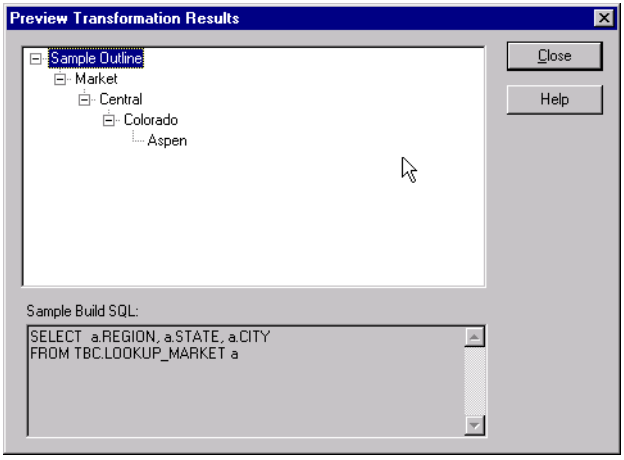



*Figure 9-19: Viewing a Sample Hyperion Essbase Outline*

The top list box displays the members that will show up in the Hyperion Essbase outline. The **Sample Build SQL** list box displays the SQL that OLAP Integration Server uses to retrieve the members from the data source.

2. When you finish viewing the members and the SQL, click **Close**.

Appendix A

# Creating a Sample OLAP Model with Attribute Dimensions

This appendix provides procedures for creating an OLAP model that contains attribute dimensions. The sample login information, sample screens, OLAP Model name, OLAP metaoutline name, and relational data source are all based on the Hyperion Integration Server sample application.

The sample application is based on a fictitious company named The Beverage Company (TBC) and contains the following components:

- A relational data source (TBC)
- An OLAP Metadata Catalog (TBC_MD)
- A sample OLAP model (TBC Model)
- A sample metaoutline (TBC Metaoutline)

For information about installing the sample application, see the *Hyperion Integration Server Installation Guide*. For information on creating a metaoutline, see the *Hyperion Integration Server Desktop OLAP Metaoutline User's Guide*.

Refer to the following topics in this appendix for information about creating a sample OLAP model containing attribute dimensions:



To create the sample OLAP model described in the following procedures, you must have the sample application installed. If you do not have the sample application installed, you can use these procedures as a guide for creating an OLAP model with your own relational data source and Metadata Catalog.

# Sample TBC Model Dimensions

The TBC sample OLAP model contains both standard and attribute dimensions. The procedures and screen examples in this appendix illustrate how to create an OLAP model that contains the standard, base, and attribute dimensions defined in Table A-1 and in Table A-2.

*Table A-1: Sample TBC OLAP Model Standard Dimensions*

| Dimension | Description |
|---|---|
| Year | Year is a hierarchy based on the transformation of a date/time stamp column in the fact table. This hierarchy contains Quarters and Months. |
| Measures | A hierarchy containing members from the fact table and containing derived (user-defined) members such as Profit, Margin, and Total Expenses. |
| Scenario | A hierarchy based on the SCENARIO fact column, containing any combination of Actual, Budget, Forecast, and Plan. This dimension also contains derived (user-defined) members for variances and percentage variances. |
| Market | A hierarchy based on the REGION and STATE where products are sold. This is the base dimension for the Population attribute. |
| Product | A hierarchy based on the FAMILY and SKU (Stock Keeping Unit) columns in the PRODUCTDIM table. PRODUCTDIM is an additional table added to the star schema to facilitate the building of a Hyperion Essbase dimension containing alternative hierarchies (repeated SKU). Repeated SKU is a business analysis requirement. Product is the base dimension for the Caffeinated, Ounces, Pkgtype, and Introdate attribute dimensions. |

*Table A-2: Sample TBC OLAP Model Attribute Dimensions*

| Attribute Dimension | Description |
|---|---|
| Caffeinated | This attribute describes a product SKU as caffeinated; it is a Boolean attribute dimension associated with level 0 Product members or SKU level products. |
| Ounces | This attribute describes the container size of the product; it is a numeric attribute dimension associated with level 0 Product members or SKU level products. |
| Pkgtype | This attribute describes the type of container in which the product is sold; it is a text attribute dimension associated with level 0 Product members or SKU level products. |
| Population | This attribute describes the population range of a state; it is a numeric attribute dimension associated with level 0 Market members or a STATE metaoutline member. |
| Introdate | This attribute describes the product introduction date; it is a date attribute dimension associated with level 0 product members or SKU level products. |

# Starting Up and Logging On

To begin the OLAP model creation process, you must first start Hyperion Integration Server, and then start and log on to Hyperion Integration Server Desktop. For details on starting up and logging on, see "Starting Hyperion Integration Server Desktop" on page 3-3.

After you have logged on, Hyperion Integration Server Desktop displays the Welcome dialog box, as shown in Figure A-1.

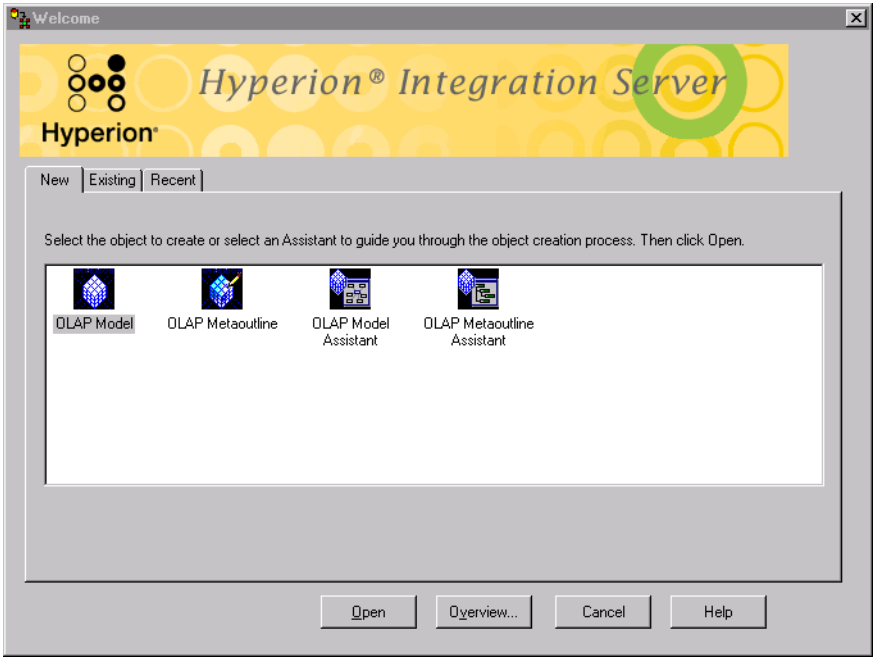



*Figure A-1: Hyperion Integration Server Desktop Welcome Dialog Box*

# Opening the Standard User Interface to Create a New OLAP Model

To create a new OLAP model containing attribute dimensions, you must use the standard user interface, rather than the OLAP Model Assistant.

➤ To open the OLAP Model standard user interface:

1. In the **Welcome** dialog box, select the **New** tab.
2. Select the **OLAP Model** icon and click **Open**.

   The **Data Source** dialog box opens, as shown in Figure A-2.

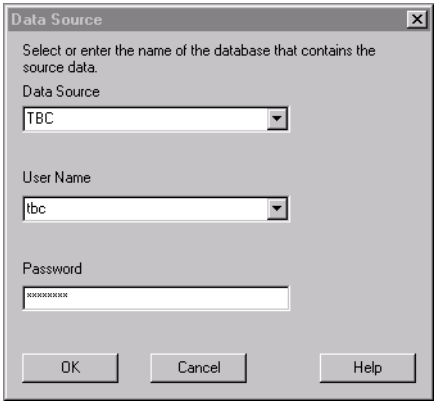



*Figure A-2: Data Source Dialog Box*

3. From the **Data Source** drop-down list, select the relational data source to use for creating the OLAP model (TBC for the sample application).
4. From the **User Name** drop-down list, select the user name for creating the OLAP model (TBC for the sample application).

5. In the **Password** text box, type your password (**password** for the sample application, all lowercase letters) and click **OK**.

   Hyperion Integration Server Desktop displays the OLAP Model main window, as shown in Figure A-3.

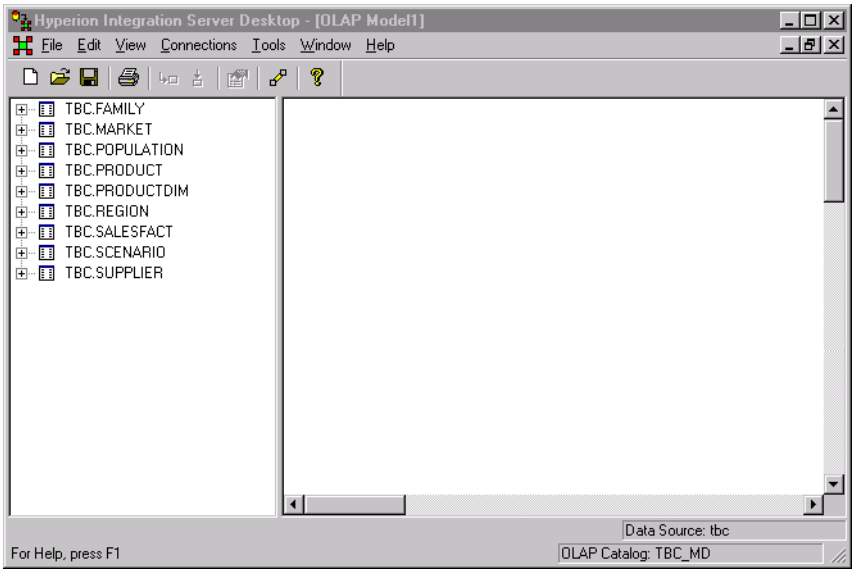



*Figure A-3: OLAP Model Main Window*

The OLAP Model main window consists of two frames. The left frame contains information about the relational data source, and the right frame, which is blank, is the area where you create the OLAP model.

# Working in the OLAP Model Main Window

This topic contains the following subtopics that describe how to create the sample OLAP model:



## Creating the Fact Table

➤ To create the fact table:

1. Drag the fact table to the right frame and release the mouse button. In this example, SALESFACT is the fact table.
2. The system displays the following prompts:
   a. "Would you like a Time dimension using a value from the fact table?" In this exercise, the answer is Yes.

      **Note:** Usually in a fact table that contains a DATETIME field related to the metrics being measured, the answer is Yes.

   b. "Would you like to add an Accounts dimension?" In this exercise, the answer is Yes.

      In some cases, either or both of the Time and Accounts dimensions are separate from the fact table. This is especially true in the financial sector, where an accounts dimension may include thousands of members. The four possible combinations for the Time and Accounts dimensions are:

      - Time, yes; Accounts, yes.
      - Time, yes; Accounts, no.
      - Time, no; Accounts, yes.
      - Time, no; Accounts, no.

3. After you have dragged the SALESFACT table to the right frame and have answered both the Time and Accounts questions, the OLAP Model main window looks like the one shown in Figure A-4.

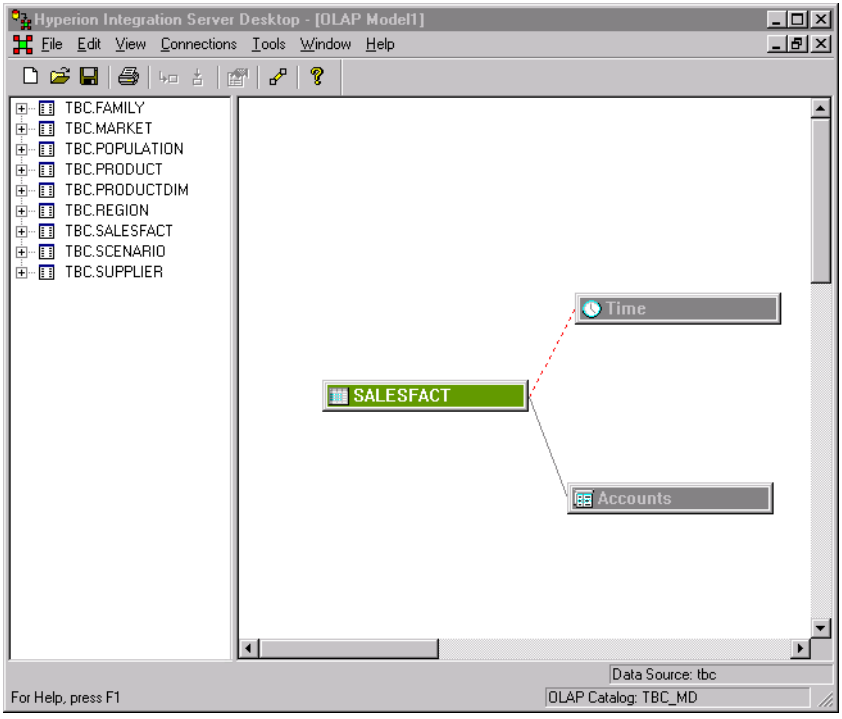



*Figure A-4: Sample OLAP Model with Fact Table, Time Dimension, and Accounts Dimension*

The dotted line between the Time dimension and the SALESFACT table indicates that the join information is incomplete.

4. To complete the join, right-click the join line and select Properties from the pop-up menu.

   Hyperion Integration Server displays the **Edit Join** dialog box.

5. In the **Edit Join** dialog box, select one or more column pairs to create a valid join between the two tables.

   In this example, both tables are joined by the TRANSDATE column pairs.

6. Click **Add** and then click **Close**.

## Creating the Product, Family, and Productdim Tables

In addition to the fact table, other tables required to create dimensions must be dragged to the right frame to create the OLAP model. The first group of these tables are PRODUCT, FAMILY, and PRODUCTDIM.

### The Product Table

➤ To create the PRODUCT table:

1. In the OLAP Model main window, drag the PRODUCT table to the right frame and release the mouse button.

   Depending on the type of relationship existing between the dragged table (here, the PRODUCT table) and the fact table, one of the following scenarios occurs:

- If a primary or foreign key relationship exists between the dragged table and the fact table, Hyperion Integration Server Desktop displays the **Create New Dimension** dialog box, as shown in Figure A-5. Click **OK** to automatically create the join.

  **Note:** In this example, a primary or foreign key relationship exists between the dragged table and the fact table.

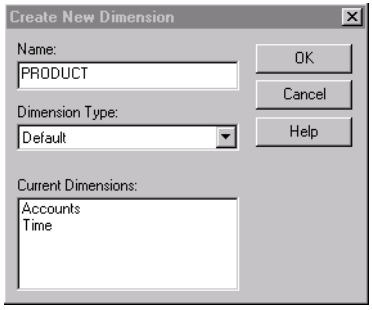



*Figure A-5: Create New Dimension Dialog Box*

- If no relationship exists in the relational database management system (RDBMS) between the dragged table and the fact table, the dialog box to create a new dimension is not automatically displayed and the tables need to be joined manually. See "Joining Tables Manually" on page A-11.

## Joining Tables Manually

➤ To join tables manually:

1. Click the **Join** button, which changes the pointer behavior from drag to join selection.


*Figure A-6: OLAP Model Toolbar*

The **Join** button is on the right of the toolbar and looks like this:

*Figure A-7: Join Button*

2. Drag the desired table to the right frame and place the table near the table you wish to join.
3. Place the cursor over the table. The cursor changes to a hand symbol.
4. Use the left mouse button to drag the hand symbol to the table you wish to join.

   A dotted line trails the hand symbol.
5. Release the left mouse button.

   Hyperion Integration Server Desktop displays the **Edit Join** dialog box.
6. In the **Edit Join** dialog box, select one or more column pairs to create a valid join between the two tables.

   **Note:** When double-clicking a column name in either table list box, the identical column name in the other list box is automatically selected. If both column names to be joined are not identical, they must be selected individually.
7. Click **Add** to add the join condition to the **Current Join Information** section.
8. Click **Close** to accept the join.

## The Family Table

➤ To create the FAMILY table:

1. In the OLAP Model main window, drag the FAMILY table to the right frame.

   Because no join exists between this table and the fact table, the join is not created automatically. See "Joining Tables Manually" on page A-11.

   **Note:** FAMILY is joined to SALESFACT through the PRODUCT table.

   Hyperion Integration Server Desktop displays the **Edit Join** dialog box, as shown in Figure A-8.

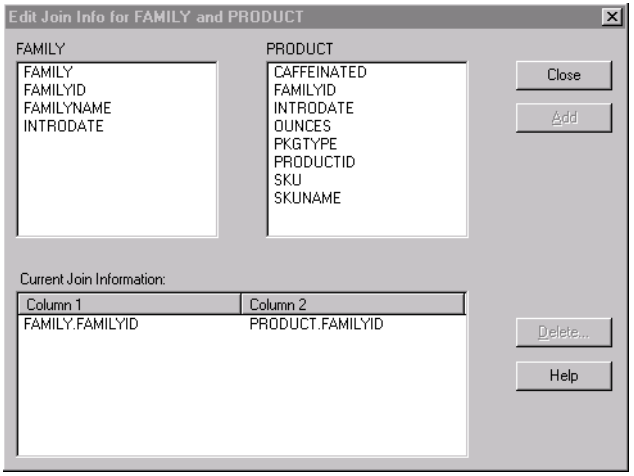



*Figure A-8: Edit Join Dialog Box for the FAMILY and PRODUCT Tables*

   Because a primary or foreign key exists between the FAMILY and the PRODUCT tables, the join from FAMILY.FAMILYID to PRODUCT.FAMILYID is suggested as the join condition between the two tables. If no primary or foreign key existed between the tables, the join condition would have to be selected manually.

2. Click **Close** to accept this join.

## The Productdim Table

➤ To create the PRODUCTDIM table:

1. In the OLAP Model main window, drag the PRODUCTDIM table to the right frame.

   Because no join exists between this table and the fact table, the join is not created automatically. See "Joining Tables Manually" on page A-11.

   **Note:** PRODUCTDIM is joined to SALESFACT through the PRODUCT table.

   Hyperion Integration Server Desktop displays the **Edit Join** dialog box, as shown in Figure A-9.

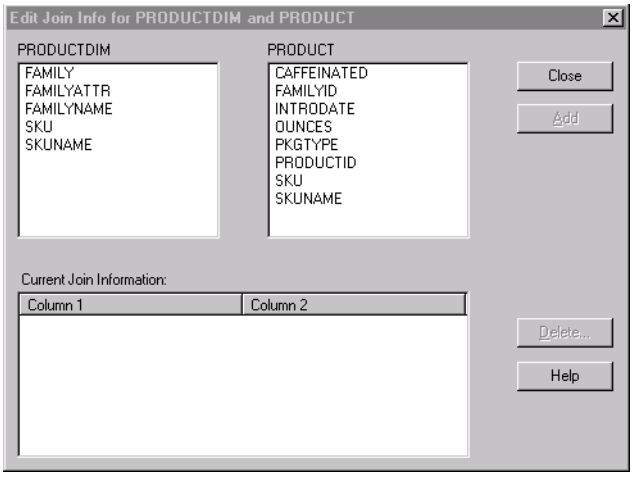



*Figure A-9: Edit Join Dialog Box for the PRODUCTDIM and PRODUCT Tables*

   Because no primary or foreign key exists between the two tables, no join information is supplied. Use SKU to join the PRODUCTDIM and PRODUCT tables.

2. Click **Close** to accept this join.

## Creating Additional Dimension Tables

To complete the OLAP model, you must drag additional relational tables to the right frame to create the necessary dimension tables. Use the same procedures as you did to drag the PRODUCT, FAMILY, and PRODUCTDIM tables to the right frame. See "Creating the Product, Family, and Productdim Tables" on page A-9.

➤ To create the required additional dimension tables:

1. In the OLAP Model main window, drag the REGION table to the right frame.
2. Because REGION is joined to SALESFACT through the MARKET table, drag the REGION table to the MARKET table.

   Hyperion Integration Server Desktop displays the **Edit Join** dialog box.

   Because a primary or foreign key exists between the REGION and the MARKET tables, the join from REGION.REGIONID to MARKET.REGIONID is suggested as the join condition between the two tables.
3. Click **Close** to accept this join.
4. In the OLAP Model main window, drag the POPULATION table to the right frame.
5. Because POPULATION is joined to SALESFACT through the MARKET table, drag the POPULATION table to the MARKET table.

   Hyperion Integration Server displays the **Edit Join** dialog box.

   Because a primary or foreign key exists between the POPULATION and the MARKET tables, the join from POPULATION.POPULATIONID to MARKET.POPULATIONID is suggested as the join condition between the two tables.

6. Click **Close** to accept this join.
7. In the OLAP Model main window, drag the SCENARIO table to the right frame.

   A primary or foreign key relationship to the fact table exists, so the join is created automatically.
8. Click **OK**.
9. In the OLAP Model main window, drag the SUPPLIER table to the right frame.

   A primary or foreign key relationship to the fact table exists so the join is created automatically.
10. Click **OK**.

**Note:** In this exercise, the SUPPLIER table is not used to create a dimension. SUPPLIER is used only for drill-through purposes, using the Hyperion Essbase Spreadsheet Add-in.

When all tables have been dragged to the right frame, the OLAP model looks like the one shown in Figure A-10.

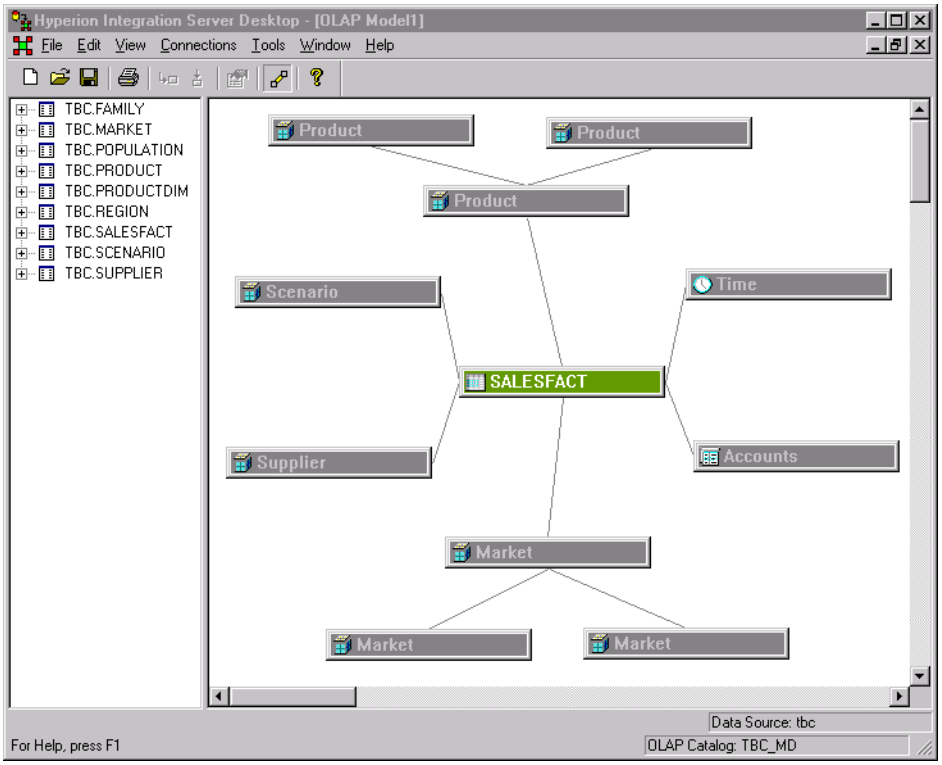



*Figure A-10: Dimension Tables in Sample OLAP Model*

## Adding Hierarchies to the OLAP Model

After the dimension tables have been created, the hierarchies for the Hyperion Essbase standard and base dimensions can be created. These include hierarchies for the Market, Product and Time dimensions.

**Note:** Attribute dimension hierarchies are created later when you build the metaoutline.

This topic contains the following subtopics that describe how to create hierarchies:


- "Creating the Market Hierarchy" on page A-17
- "Creating the Product Hierarchy" on page A-19
- "Creating the Time Year Hierarchy" on page A-20


### Creating the Market Hierarchy

➤ To create the Market hierarchy:

1. In the OLAP Model main window, right-click any Market dimension table and select Properties from the pop-up menu.

2. Select the **Hierarchies** tab and click **Add**.

   Hyperion Integration Server Desktop displays the **Edit Hierarchy** dialog box, as shown in Figure A-11.

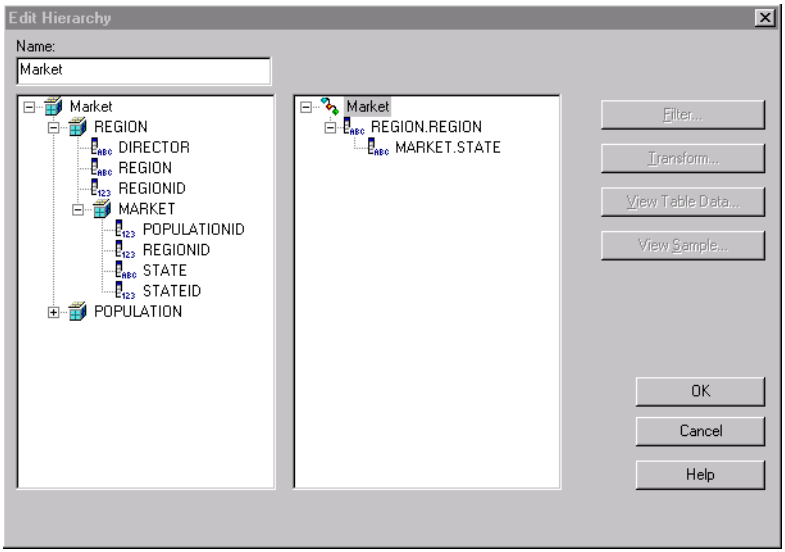



*Figure A-11: Edit Hierarchy Dialog Box*

3. In the **Name** text box of the **Edit Hierarchy** dialog box, type **Market** to change the name from "Hierarchy1" to "Market."
4. Click the plus (+) symbol in front of the REGION table to expand the view.
5. Click the plus (+) symbol in front of the MARKET table to expand the view.
6. Under the REGION table, double-click the REGION column and then under the MARKET table, double-click the STATE column.

   **Note:** You can achieve the same result by dragging the columns from the left side to the right side.

7. Click **OK**, then click **OK** again.

   **Note:** In this exercise, all dimensions are built by generation. However, Hyperion Integration Server Desktop can also build dimensions recursively (that is, by parent/child).

## Creating the Product Hierarchy

➤ To create the Product hierarchy:

1. In the OLAP Model main window, right-click any Product dimension table and select Properties from the pop-up menu.
2. Select the **Hierarchies** tab and click **Add**.

   Hyperion Integration Server Desktop displays the **Edit Hierarchy** dialog box.
3. In the **Name** text box of the In the **Edit Hierarchy** dialog box, type **Product** to change the name from "Hierarchy1" to "Product."
4. Click the plus (+) symbol in front of the PRODUCTDIM table to expand the view.
5. Under the PRODUCTDIM table, double-click the FAMILY column and then under the PRODUCTDIM table, double-click the SKU column.
6. Click **OK**, then click **OK** again.

   In this exercise, the PRODUCTDIM table illustrates that, at times, a table can be created separately from the star schema and still be used in the creation of a hierarchy. Specifically, the FAMILYATTR column is used later in the metaoutline to create the consolidation attribute.

   Also, in this example, duplicate SKUs are in the PRODUCTDIM table. By creating a separate table, you can create an alternate hierarchy called Diet.

## Creating the Time Year Hierarchy

➤ To create the Time Year hierarchy:

1. In the OLAP Model main window, right-click the Time dimension table and click **View Columns**.

   Hyperion Integration Server Desktop expands the Time table and displays the columns that make up the Time dimension, as shown in Figure A-12.


*Figure A-12: Time Dimension Columns*

2. In the Time dimension box, right-click the Time table and select Properties from the pop-up menu.

   Hyperion Integration Server Desktop displays the **Table Properties** dialog box, as shown in Figure A-13.

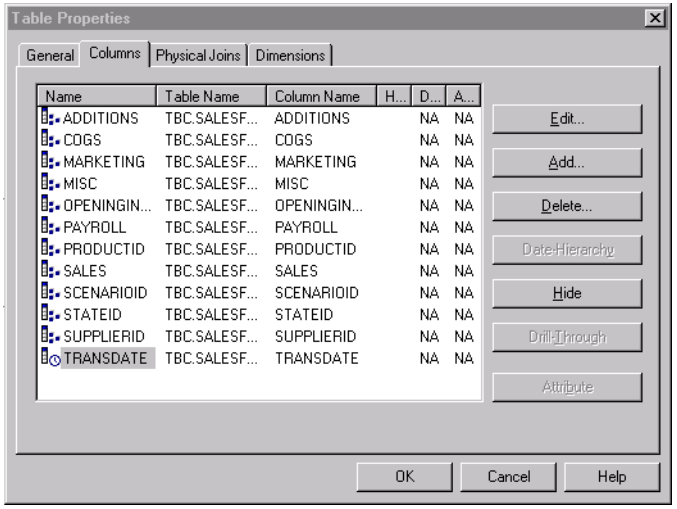



*Figure A-13: Table Properties Dialog Box*

3. In the **Table Properties** dialog box, select the **Columns** tab.

4. Because both of the new columns are based on a transformation of the TRANSDATE column, select TRANSDATE and then click **Add**.

   Hyperion Integration Server Desktop displays the **Add New Column to Table** dialog box, as shown in Figure A-14.

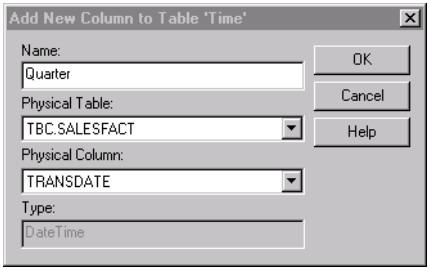



*Figure A-14: Add New column to Table Dialog Box*

5. In the **Name** text box of the **Add New Column to Table** dialog box, name the new column "Quarter" and click **OK**.

   Hyperion Integration Server Desktop displays the **Column Properties** dialog box, as shown in Figure A-15.

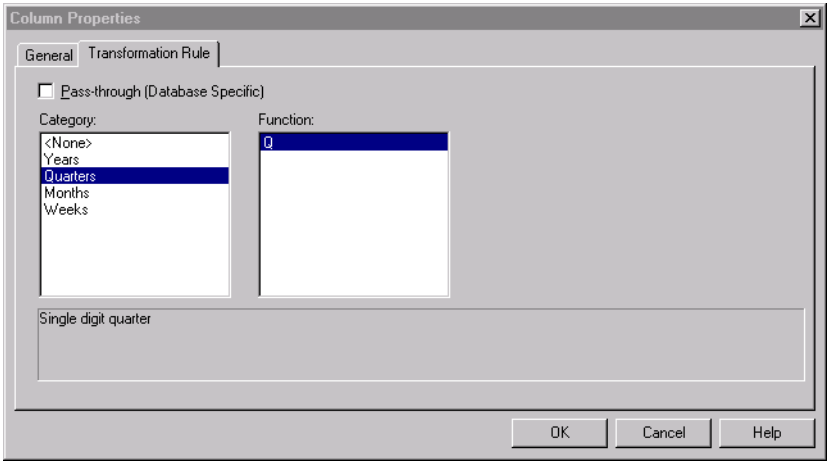



*Figure A-15: Column Properties Dialog Box*

6. In the **Column Properties** dialog box, select the **Transformation Rule** tab.

7. In the **Categories** list box, select Quarters and in the **Function** list box, select Q.
8. Click **OK**.

   Hyperion Integration Server Desktop displays the **Table Properties** dialog box.

   You have created the column Quarters as the extraction of the Quarter Date-Part from the TRANSDATE column.
9. In the **Name** column of the **Table Properties** dialog box, select TRANSDATE and click **Add**.

   Hyperion Integration Server Desktop displays the **Add New Column to Table** dialog box.
10. In the **Name** text box of the **Add New Column to Table** dialog box, name the new column “Month” and click **OK**.

    Hyperion Integration Server Desktop displays the **Column Properties** dialog box.
11. In the **Column Properties** dialog box, select the **Transformation Rule** tab.
12. In the **Category** list box, select Months and then in the **Function** list box, select MM.
13. Click **OK**, then click **OK** again.

    You have created the logical column Month as the extraction of the Month Date-Part from the TRANSDATE column. The available Categories and Functions vary, based on the data type of source column (in this example, TRANSDATE, which has a DATETIME data type in the source column).

### Renaming the Time Hierarchy

The Time dimension must be renamed before the Year hierarchy can be created.

➤ To rename the Time dimension:

1. In the OLAP Model main window, right-click the Time dimension table and select Rename from the pop-up menu.
2. Change the Dimension name from “Time” to “Year.”
3. Press the Enter key.

## Creating the Year Hierarchy

After you have created the logical columns Quarter and Month, and the Time dimension has been renamed to Year, you can create the Year Hierarchy.

➤ To create the Year hierarchy:

1. In the OLAP Model main window, right-click the Year dimension table and select Properties from the pop-up menu.
2. Select the **Hierarchies** tab and click **Add**.

   Hyperion Integration Server Desktop displays the **Edit Hierarchy** dialog box:
3. In the **Name** text box of the **Edit Hierarchy** dialog box, type **Year** to change the name from “Hierarchy1” to “Year.”
4. Click the plus (+) symbol in front of the Time dimension to expand the view.
5. Under the TIME table, double-click the Quarter column, and then under the TIME table, double-click the Month column.

6. Click the Time.Quarter member and select **Transformation**.

   Hyperion Integration Server Desktop displays the **Edit Transformations** dialog box, as shown in Figure A-16.

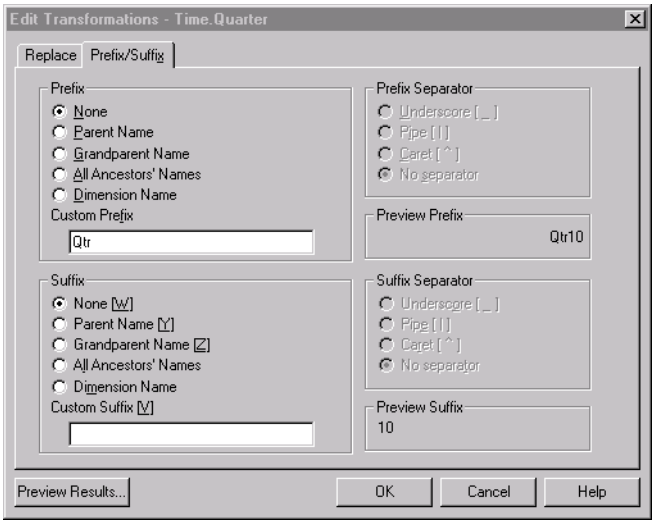



*Figure A-16: Edit Transformations Dialog Box*

7. Select the **Prefix/Suffix** tab.
8. In the **Custom Prefix** text box, type **Qtr** and click **OK**.

   **Note:** Many Prefix/Suffix parameters can be selected to ensure that Hyperion Essbase members are unique.

**9.** In the **Edit Hierarchy** dialog box, select the Time.Month member and click **Transformation**.

Hyperion Integration Server Desktop displays the **Edit Transformations** dialog box, as shown in Figure A-17.

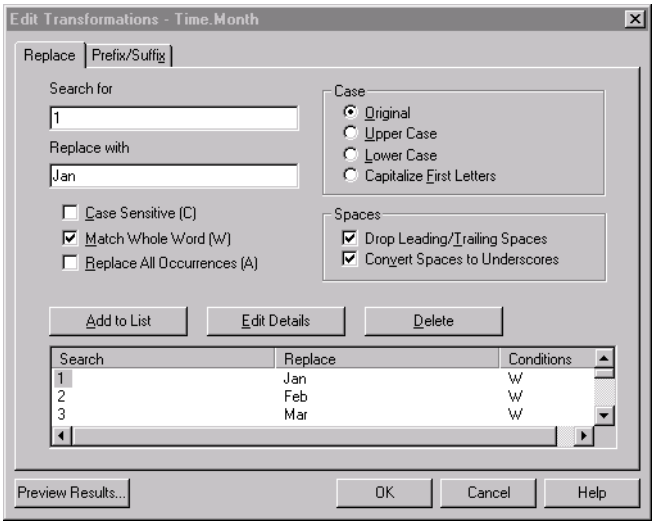



*Figure A-17: Edit Transformations Dialog Box*

### Replacing the Year Hierarchy Month Numbers

You can change the format of year hierarchy months from numeric digits to three-character month abbreviations.

➤ To replace month numbers:

1. In the **Search for** text box, enter the Month number to be replaced.
2. In the **Replace with** text box, enter the text label which will replace the numeric month value.
3. The default replacement option is **Replace All Occurrences (A)**. Change this selection to **Match Whole Word (W)**, first by clicking **Replace All Occurrences (A)** to deselect it and then clicking **Match Whole Word (W)**.
4. Click **Add to List** to add this Search and Replace operation to the list.

   **Note:** Repeat this process for the other eleven months. Each numeric-digit month must be replaced individually.

5. Click **OK** three times to exit the dialog box.

   **Note:** You can apply other transformations in the **Edit Transformations** dialog box; for example, you can change the case of the member names and omit leading or tailing spaces.

## Setting Columns to Attribute-Enabled

The final step in setting up the OLAP model so that you can create attribute dimensions in the subsequent metaoutline is to set columns to attribute-enabled.

➤ To set columns to attribute-enabled:

1. In the OLAP Model main window, right-click all the Product dimensions and select View Columns from the pull-down menu.

   **Note:** In this example, there are three Product dimensions.

2. Select the Product dimension with PRODUCTDIM as the table name.

3. Right-click PRODUCTDIM and select Properties from the pop-up menu.

   Hyperion Integration Server Desktop displays the **Table Properties** dialog box, as shown in Figure A-18.

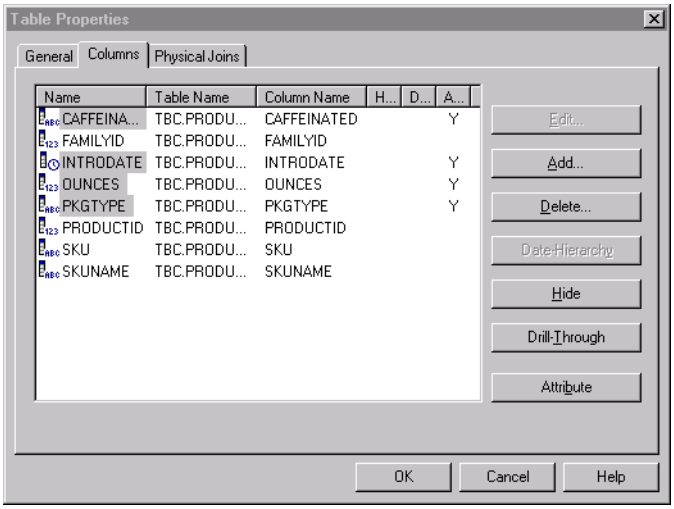



*Figure A-18: Table Properties Dialog Box*

4. Select the **Columns** tab.
5. Select all column names that are to be set as attribute-enabled (in this example, CAFFEINATED, INTRODATE, OUNCES, and PKGTYPE).
6. Click **Attribute**, and then click **OK**.
7. In the OLAP Model main window, right-click all the Market dimension tables and select View Columns from the pull-down menu.

   **Note:** In this example, there are three Market dimensions.

8. Select the Market dimension with POPULATION as the table name.
9. Right-click POPULATION and select Properties from the pop-up menu.

   Hyperion Integration Server Desktop displays the **Table Properties** dialog box.

10. Select the **Columns** tab.
11. Select all column names that are to be set as attribute-enabled (in this example, POPGROUP and POPULATION).
12. Click **Attribute**, and then click **OK**.

## Renaming the Accounts Dimension

Before the OLAP model is saved, the Accounts dimension must be renamed to "Measures."

➤ To rename the Accounts dimension:

1. In the OLAP Model main window, right-click the Accounts dimension and click **Rename**.
2. Type **Measures** to change the Dimension name from "Accounts" to "Measures."
3. Press the Enter key.

## Saving the OLAP Model

➤ To save the OLAP model:

1. Select File > Save.
2. Enter the OLAP model name and click **OK**.

# Glossary

**#MISSING.** *See* missing data (#MISSING).

**accounts dimension.** A dimension type that makes accounting intelligence available. You can tag only one dimension as accounts; you do not have to have an accounts dimension.

**Add Joins mode.** In Hyperion Integration Server, a state in which you can draw lines to define joins between objects in the OLAP model.

**agent log file.** A record of actions performed by the agent (server).

**aggregate.** *See* consolidate.

**aggregation level.** *See* consolidation level.

**alias.** An alternative name for a dimension, member, or description.

**alias column.** In Hyperion Integration Server, a column in the data source that contains the aliases for a member level in the metaoutline.

**alternate name.** *See* alias.

**ancestor.** A branch member that has members below it. For example, in a dimension that includes years, quarters, and months, the members Qtr2 and 2001 are ancestors of the member April.

**application.** In Hyperion Essbase, a container for one or more Hyperion Essbase databases and files, such as calc scripts and data load rules, that are related to the databases.

**ARBORPATH.** An environment variable that specifies the Hyperion Essbase root directory.

**Architect.** *See* Hyperion Integration Server Desktop.

**attribute.** A classification of a member in a dimension. You can specify an attribute to select and group members with a specified attribute and to perform calculations and application-specific functions. For example, a Product dimension can have several attributes, such as Size and Flavor. A specific member of the Product dimension can have the Size attribute 8 and the Flavor attribute Cola.

**base dimension.** A standard dimension that is associated with one or more attribute dimensions. To classify a member of a base dimension, you associate it with a member of one or more attribute dimensions that describes the classification, such as a specific flavor. For example, assuming products have flavors, the Product dimension is the base dimension for the Flavors attribute dimension.

**block.** The primary storage unit within Hyperion Essbase. A data block is a multidimensional array representing the cells of all dense dimensions.

**branch.** A member of a hierarchy that may or may not contain leaf members.

**Builder.** *See* Hyperion Integration Server Desktop.

**calc script.** *See* calculation script.

**calculation script.** A text file containing a set of instructions telling Hyperion Essbase how to calculate a database.

**Catalog.** *See* OLAP Metadata Catalog.

**cell.** A unit of data representing the intersection of dimensions in a multidimensional database; the intersection of a row and a column in a worksheet.

**child.** A member that has a parent above it in the database outline. A child may have peers (siblings) that exist at the same layer of the database outline.

**column.** In relational databases, a vertical part of a table. Also known as a field. A column contains all the values for a specific type of information. Every column has a name and a particular data type. For example, a column may contain the name or employee number for each employee.

**Command Interface.** *See* OLAP Command Interface.

**concatenation.** An operation that joins two characters or strings in the order specified, forming one string whose length is equal to the sum of the lengths of the two characters or strings. For example, the strings “New York “ and “Library”, when concatenated, become “New York Library”.

**condition.** In relational databases, a data extraction criterion. For example, you can apply a condition to extract only the data that begins with the letter A.

**consolidate.** The process of gathering data from dependent entities and aggregating the data to parent entities. Once you enter or load data into child entities, you perform a consolidation to aggregate the data through the organization. As data consolidates, intercompany processing, conversion methods, equity adjustments, and minority ownerships perform calculations on the data.

**consolidation level.** The top of an aggregation hierarchy or any branch or sub-branch below the top, including the input (leaf) portion of the hierarchy.

**data block.** *See* block.

**data cleansing.** The process of making inconsistent data consistent. Examples of inconsistent data are data in which some values are incorrect or not of the correct data type.

**data file.** A file containing data blocks; Hyperion Essbase generates the data file during a data load and stores it on disk.

**data load.** The process of populating a Hyperion Essbase database with data. Loading data establishes actual values for the cells defined by the structural outline of the database.

**data load rules.** A set of criteria or rules that Hyperion Essbase uses to determine how to load data from a text-based file or a relational data set into a Hyperion Essbase database.

**data point.** *See* cell.

**data source.** External data, such as a text file, spreadsheet file, or SQL database that will be loaded into a Hyperion Essbase database.

**data type.** Defines the kind of data that a column can contain. For example, columns with the Numeric data type contain numbers.

**data value.** *See* cell.

**database.** A collection of related information. Each unit (record) of the database is typically organized in a fixed format to make it easy to retrieve selected portions of the data on demand. Each record is made up of one or more data fields, and each data field can hold one piece of data (known as a value).

**database administrator.** An individual who administers database servers, such as Hyperion Essbase, and who may also design, maintain, and create databases.

**database outline.** *See* outline.

**DateTime transformation.** A set of instructions that defines how to change or reformat a relational database DateTime data type to your choice of date format.

**DBMS.** *See* relational database management system (RDBMS, DBMS).

**denormalization.** The process of adding redundancy to data in a database, typically by joining tables to form more complete sets of data in the individual tables. This process is performed for the purpose of increasing data retrieval performance. *Contrast with* normalization.

**dense dimension.** A dimension likely to contain data for every combination of dimension members. For example, a time dimension is typically a dense dimension because it contains all combinations of all members. Contrast with sparse dimension.

**descendant.** Any member below a parent in the database outline. For example, in a dimension that includes years, quarters, and months, the members Qtr2 and April are descendants of the member Year.

**detail member.** *See* leaf member.

**dimension.** A data category that is used to organize business data for retrieval and preservation of values. Each dimension usually contains a hierarchy of related members grouped within it. For example, a Year dimension often includes members for each time period, such as quarters and months. Other common business dimensions may be measures, natural accounts, products, and markets.

**dimension branch.** A collection of dimension tables organized in a hierarchical structure, with one of the dimension tables joined directly to the fact table. A dimension branch defines a single, potential Hyperion Essbase dimension in a Hyperion Integration Server metaoutline.

**dimension build rules.** In Hyperion Essbase, a set of operations similar to data load rules. Instead of loading data, the dimension build rules modify the outline based on data in the external data source file.

**dimension table.** A container in the OLAP model for one or more relational tables that define a potential Hyperion Essbase dimension. When one dimension table joins to another dimension table, forming a dimension branch, the dimension is composed of the columns of all of the dimension tables in the dimension branch.

**dimension type.** A property in the OLAP model and in the metaoutline that defines which tables form the Hyperion Essbase accounts, time, and standard (default) dimensions.

**duplicate member.** The second occurrence of a member name in a data source. Users can determine whether Hyperion Integration Server ignores duplicate members or adds them as shared members. *See also* shared member.

**Dynamic Calc And Store members.** A member that the Hyperion Essbase server calculates only upon the first retrieval of the value. The server then stores the calculated value in the database. Subsequent retrievals of a Dynamic Calc And Store member do not require calculating.

**Dynamic Calc members.** A member that the Hyperion Essbase server calculates only at retrieval time. The server discards calculated values after the retrieval request is complete.

**dynamic calculation.** In Hyperion Essbase, a calculation that occurs only when you retrieve data on a member that has been tagged as Dynamic Calc or Dynamic Calc And Store. The member's values are calculated at retrieval time instead of being precalculated during batch calculation.

**Dynamic Time Series.** A process that is used to perform dynamic period-to-date reporting.

**Dynamic Time Series members.** Predefined members that are used to perform Dynamic Time Series reporting.

**Essbase database.** A repository of data within Hyperion Essbase that contains a multidimensional data storage array. Each database consists of a defined storage structure (a database outline), data, security definitions, and other associated files, such as calc scripts or data load rules. *See also* application.

**Essbase OLAP server.** A database server that locates and accesses multidimensional data.

**Essbase outline.** *See* outline.

**fact table.** In Hyperion Integration Server, a container for one or more relational tables that define the data values for each dimension intersection in the OLAP model. For example, if the OLAP model contains Products, Region, and Year dimensions, the fact table might include data values for the number of units of Product A sold in New York in January.

**field.** (1) In Hyperion Essbase, a value or item in a data source file that is retrieved from a Hyperion Essbase database. (2) In relational databases, a space allocated for a particular item of information. Fields are the smallest units of information you can access. Most fields have certain characteristics associated with them. For example, some fields are numeric, whereas others are textual. Every field has a name.

**file delimiter.** One or more characters, such as a comma (,), separating fields in a data source.

**filter.** (1) In Hyperion Essbase, a method for controlling access to database cells. A filter is the most detailed level of security, allowing you to define the varying access levels that users can have to data. (2) In Hyperion Integration Server and in relational databases, a method for controlling which data you retrieve from a relational database. For example, you can choose to retrieve products only in the Product A product family. A filter contains one or more conditions.

**foreign key.** In relational databases, a column whose data values correspond to the values of a key column in another relational table. *See also* key column and primary key.

**formula.** In Hyperion Essbase, a combination of operators and functions as well as dimension names, member names, and numeric constants. Formulas are used to calculate relationships between members of a Hyperion Essbase database. @VAR(Actual, Budget) is an example of a formula.

**generation.** A layer in a hierarchical tree structure that defines member relationships in a Hyperion Essbase database. Hyperion Essbase orders generations incrementally from the dimension (generation 1) down to the leaf members.

**generation name.** A unique name that describes a generation.

**hierarchy.** A set of multidimensional relationships in an outline, often created in a tree formation. For example, parents, children, and generations represent a hierarchy.

**Hyperion Essbase kernel.** A layer of the Hyperion Essbase server that provides the foundation for a variety of functionality, including data loading, calculations, spreadsheet lock&send, partitioning, and restructuring. The Hyperion Essbase kernel reads, caches, and writes data; manages transactions; and enforces transaction semantics to ensure data consistency and data integrity.

**Hyperion Integration Server Desktop.** The client component of the Hyperion Integration Server product family. This tool is used to create OLAP models and metaoutlines, and load data into Hyperion Essbase databases.

**index.** (1) In Hyperion Essbase, a method of retrieving data based on sparse dimensions. Also refers to the index files, collectively. (2) In relational databases, pointers that are logically arranged by the values of a key. Indexes optimize access to relational data.

**index cache.** In Hyperion Essbase, a buffer in memory that holds index pages.

**index entry.** In Hyperion Essbase, a pointer to an intersection of sparse dimensions. Each index entry points to a block on disk and locates a particular cell within the block by means of an offset.

**index file.** In Hyperion Essbase, a file used to store data retrieval information. It resides on disk and contains index pages.

**index page.** In Hyperion Essbase, a subdivision of an index file containing entries that point to data blocks.

**input block.** A type of data block that has at least one loaded data value.

**input data.** Any data that is loaded from a data source and is not generated by calculation.

**Integration Server.** *See* OLAP Integration Server.

**integrity constraint.** In relational databases, a rule stating that each row should have an entry for each required key column.

**intersection level.** In the Hyperion Essbase Spreadsheet Add-in, a Hyperion Essbase member combination that defines a specific value. For example, the member combination Actual, Root Beer, Sales, Jan, East represents the actual January sales value for root beer in the Eastern region.

**join.** In relational databases, a relationship between two tables based on matching key column values.

**join columns.** In Hyperion Integration Server, two relational table columns that are joined from one table to another.

**key column.** In relational databases, a column or columns that form a unique identifier for each row. For example, EMPLOYEE_ID might be a key column.

**leaf member.** A member that has no children.

**left frame.** (1) In the Hyperion Integration Server Desktop OLAP Metaoutline main window, the area on the left that enables you to view a list of dimensions previously defined in the OLAP model. (2) In the OLAP Model main window, the area on the left that displays a list of the tables and views available in your source relational database.

**level.** (1) In Hyperion Essbase, a layer of a hierarchical tree structure that defines database member relationships. Hyperion Essbase numbers levels incrementally from the leaf member (level 0) up to the dimension member. (2) In Hyperion Integration Server, the distance of a dimension table from the fact table in the OLAP model. For example, if a table is level 2, it is separated from the fact table by two links.

**level 0 block.** A data block that is created for sparse member combinations when all of the members of the sparse combination are level 0 members.

**level 0 member.** *See* leaf member.

**level name.** A unique name that describes a level.

**load member.** In Hyperion Integration Server, a member in a user-defined dimension into which data is loaded. Only user-defined dimensions require load members. For all non user-defined dimensions, Hyperion Integration Server knows how to load members and data into the Hyperion Essbase database.

**load properties.** In Hyperion Integration Server, a set of rules that determine what actions the product performs on member level names and data as they are loaded.

**logical column.** In Hyperion Integration Server, a column created by manipulating the data in one or more physical columns. *See also* column. *Contrast with* physical column.

**logical table.** In relational databases, a table created by manipulating columns from one or more physical tables. The logical table is only a view of the data; the columns remain stored in the original tables and are not physically duplicated in the logical table. *See also* view. *Contrast with* physical table.

**mathematical operator.** A symbol that defines how data is calculated. A mathematical operator can be any of the standard mathematical or Boolean operators, for example, +, -, *, /, and %. Mathematical operators are used in formulas, abd outlines.

**MDDB.** *See* multidimensional database (MDDB).

**measures.** Data values that a user wants to track, such as Unit_Price and Discount. By default, measures values map to the accounts dimension in the OLAP model, which maps to the measure dimension in the OLAP metaoutline, which in turn maps to the accounts dimension in the Hyperion Essbase outline.

**measures dimension.** In Hyperion Integration Server, a dimension that, by default, maps to the accounts dimension in the Hyperion Essbase outline.

**member.** A discrete component within a dimension. For example, a time dimension might include members such as Jan, Feb, and Qtr1.

**member combination.** In Hyperion Essbase, a list of member names used to specify a set of data at the intersection of two or more dimensions. A member combination is specified by using the cross-dimensional operator -> (a hyphen followed by a right-angle bracket). For example, the actual sales data for the month of January in Sample Basic is Sales->Jan->Actual.

**member level.** A hierarchical level of detail within a dimension. For example, in a dimension that defines geographic areas by nation, which are then subdivided into provinces, the nation and province categories each represent a member level. A member level corresponds to a level in a Hyperion Essbase outline. The measures dimension contains actual members that are also member levels.

**member load.** The process of adding new dimensions or members (without data) to a Hyperion Essbase outline. *Contrast with* data load.

**metadata.** The components of a database outline that describe and hold data. Examples of metadata are dimension names, member names, properties, time periods, and security.

**metaoutline.** In Hyperion Integration Server, a template containing the structure and rules for creating a Hyperion Essbase outline from an OLAP model.

**missing data (#MISSING).** A marker indicating that data in the labeled location does not exist, contains no meaningful value, or was never entered or loaded. For example, missing data exists when an account contains data for a previous or a future period but not for the current period.

**Move mode.** In Hyperion Integration Server, a state in which you can pick up, move, and drop objects in the OLAP Model main window.

**multidimensional.** Describes a method of referencing data through three or more dimensions. An individual data value is the intersection of one member from each dimension.

**multidimensional database (MDDB).** A method of referencing data through three or more dimensions. An individual record is the intersection of a point for a set of dimensions. *Contrast with* relational database.

**normalization.** The process of grouping and removing redundancy from data so that each entity is in its appropriate place in the database and only in its appropriate place. *Contrast with* denormalization.

**numeric transformation.** In Hyperion Integration Server, a set of instructions that define how to change or reformat a relational database numeric field type. For example, you may choose to divide numeric data by 100.

**ODBC.** *See* Open Database Connectivity (ODBC).

**ODBC data source.** Location of the data that you are accessing and the information necessary to access the data using ODBC. For example, an ODBC data source that connects to an SQL Server might require a server name, a database name, a user name, and a password. See also Open Database Connectivity.

**OLAP.** *See* online analytical processing (OLAP).

**OLAP Architect.** *See* Hyperion Integration Server Desktop.

**OLAP Builder.** *See* Hyperion Integration Server Desktop.

**OLAP Catalog.** *See* OLAP Metadata Catalog.

**OLAP Command Interface.** In Hyperion Integration Server, a command-line tool that you can use to perform common operations on the Hyperion Essbase outline and the data in the Hyperion Essbase database. For example, you can use the LOADDATA command to load data.

**OLAP Integration Server.** The server component of the Hyperion Integration Server product family. OLAP Integration Server uses the information stored in the OLAP Metadata Catalog to extract the dimension names and member names needed to build a Hyperion Essbase outline from the data source. When the Hyperion Essbase outline is complete, OLAP Integration Server extracts data from the data source, performs the operations specified in the metaoutline, and loads the data into the Hyperion Essbase database.

**OLAP Metadata Catalog.** In Hyperion Integration Server, a relational database containing metadata describing the nature, source, location, and type of data that you pull from the relational data source. OLAP Integration Server accesses the OLAP Metadata Catalog to generate the SQL statements and the information required to generate a Hyperion Essbase database outline.

**OLAP model.** In Hyperion Integration Server, a logical model (star schema) that you create from tables and columns in a relational database. You can then use the OLAP model to generate the structure of a multidimensional database.

**OLTP.** *See* online transaction processing (OLTP).

**online analytical processing (OLAP).** A multidimensional, multi-user, client-server computing environment for users who need to analyze consolidated enterprise data in real time. OLAP systems feature drilling down, data pivoting, complex calculations, trend analyses, and modeling.

**online transaction processing (OLTP).** OLTP applications are commonly referred to as data capture, data entry, or data collection applications. OLTP applications enable an organization to capture the large amounts of data resulting from its daily activities but provide limited capability for reporting on the data.

**Open Database Connectivity (ODBC).** Standardized application programming interface (API) technology that allows applications to access multiple third-party databases.

**outline.** The structure that defines all elements of a database within Hyperion Essbase. For example, an outline contains definitions of dimensions, members, and formulas.

**parent.** A member that has an aggregated branch below it.

**pass-through transformations.** In Hyperion Integration Server, a feature that allows you to use functions specific to your relational database management system (RDBMS) to extract data values for columns. You can provide a statement that Hyperion Integration Server passes through to your RDBMS as a part of the SQL SELECT statement. You provide the statement as a property of a column and the pass-through feature returns a value for the column.

**permission.** A special privilege that must be assigned to users or groups to enable them to access or modify secure data. Permissions include Read, Read/Write and None.

**physical column.** A column that is stored in a relational database. *See also* column. *Contrast with* logical column.

**physical table.** A combination of rows and columns stored in a relational database. *Contrast with* logical table.

**pointer.** In relational databases, a data element that indicates the location of data in storage.

**primary dimension table.** A dimension table that joins directly to the fact table. Additional dimension tables may join to the primary dimension table to create a dimension branch.

**primary key.** In relational databases, a column (key) that uniquely identifies a row. For example, Employee_ID.

**query governor.** In Hyperion Integration Server, a parameter that you set to control the time and size of the queries made to the data source.

**RDBMS.** *See* relational database management system (RDBMS, DBMS).

**record.** A set of information in a data source. Records are composed of fields, each of which contains one item of information. A set of records constitutes a table. A single record constitutes a row in the table. For example, a table containing personnel information might contain records (rows) that have three fields: a NAME field, an ADDRESS field, and a PHONE_NUMBER field.

**recursive table.** A source relational table that contains information in one row that is a parent or child of information in another row. For example, in a relational source table containing the columns EMPLOYEE_ID, NAME, and MANAGER_ID, the columns EMPLOYEE_ID and MANAGER_ID are recursive because MANAGER_ID refers back to the EMPLOYEE_ID. Using Hyperion Integration Server, you can build a Hyperion Essbase outline hierarchy from a recursive source table.

**relational database.** A type of database that stores data in the form of related tables. A single database can be spread across several tables, and can be viewed in many different ways. *Contrast with* multidimensional database (MDDB).

**relational database management system (RDBMS, DBMS).** A system for accessing data in a relational database. This term is often used to describe systems for management of multiple relational databases or tables.

**restructure.** In Hyperion Essbase, an operation to regenerate or rebuild the database index and, in some cases, the data files.

**right frame.** In the Hyperion Integration Server Desktop OLAP Metaoutline main window, the area on the right, in which you build a metaoutline; In the OLAP Model main window, the area on the right, in which you build an OLAP model.

**roll-up.** *See* consolidate.

**schema.** In relational databases, a logical model that represents the data and the relationships between the data.

**server.** *See* Essbase OLAP server and OLAP Integration Server.

**shared member.** A member that shares storage space with another member of the same name. The shared member has a property that designates it as shared. The use of shared members prevents duplicate calculation of members that appear more than once in a Hyperion Essbase outline.

**sibling.** A child member within a dimension, having the same parent as another child member. For example, the members East and West are both children of the Markets dimension and siblings of each other.

**sparse dimension.** A dimension unlikely to contain data for all combinations of dimension members. For example, product and market dimensions are sparse if not all products are sold in all markets. Contrast with dense dimension.

**SQL.** *See* Structured Query Language (SQL).

**staging area.** A database that you create to meet the needs of a specific application. A staging area is a snapshot or restructured version of one or more RDBMSs.

**standard dimension.** A dimension that is not an attribute dimension.

**star schema.** In Hyperion Integration Server, a logical model that represents your relational data in a form that mirrors OLAP. A star schema contains a fact table and one or more dimension tables.

**storage manager.** *See* Hyperion Essbase kernel.

**string.** A sequence of characters treated as a unit.

**Structured Query Language (SQL).** A computer language used to access data in relational databases.

**supervisor.** A defined type of user who has full access to all applications, databases, related files, and security mechanisms for a server.

**system administrator.** A person who maintains the hardware, software, disk space distribution, and configurations for running software applications such as Hyperion Essbase.

**table.** In relational databases, a form of data storage in which data is stored in rows and columns.

**time dimension.** A dimension type that defines how often data is collected and updated, such as fiscal or calendar periods. Only one dimension may be tagged as time; a time dimension is not required.

**transformation rules.** In Hyperion Integration Server, a set of instructions that define how to change or reformat the member names and data you extract from the source relational database.

**two-pass calculation.** A Hyperion Essbase property that is used to recalculate members that are dependent on the calculated values of other members. Two-pass members are calculated during a second pass through the database outline.

**UDA.** Formerly called user-defined attribute. A UDA is a term associated with members of an outline to describe a particular characteristic of the members. Users can specify UDAs within calc scripts and reports to return lists of members that have the specified UDA associated with them. UDAs can be applied to dense as well as sparse dimensions.

**unary operator.** A group of mathematical indicators (+, -, *, /, %) that define how roll-ups take place on the database outline.

**union.** In relational databases, a type of join that combines the results of two SELECT statements. A union is often used to merge lists of values contained in two tables.

**upper-level block.** A type of data block that is created for sparse member combinations when at least one of the sparse members is a parent-level member.

**user-defined attribute.** *See* UDA.

**user-defined dimension.** Dimensions that you explicitly create in Hyperion Integration Server, rather than dimensions obtained or built from the relational data source.

**user-defined member.** Members that you explicitly create in Hyperion Integration Server, rather than obtaining and building them from the relational data source.

**validation.** (1) In Hyperion Essbase, a process of checking a rules file against the outline to make sure the rules file is valid. (2) In Hyperion Integration Server, a process of checking the OLAP model and metaoutine.

**view.** In relational databases, a logical table created by combining columns from one or more tables.

# Index

## Symbols



## A

## B



## C

## D

## E



## F

## G



## H

## I



## J

## K



## L



## M

## N



## O

## P

## Q



## R

# S

## T

## U



## V

## W



## Y



## Z