

# **Oracle® Essbase Integration Services**

## **Data Preparation Guide**

RELEASE 11.1.2.1

**ORACLE®**

---

**ENTERPRISE PERFORMANCE  
MANAGEMENT SYSTEM**

Integration Services Data Preparation Guide, 11.1.2.1

Copyright © 1998, 2011, Oracle and/or its affiliates. All rights reserved.

Authors: EPM Information Development Team

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited. The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS:

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

---

# Contents

---

<b>Chapter 1. OLAP and Integration Services</b> .....	5
OLAP .....	5
Multidimensional Databases .....	6
User Interactions with Data .....	6
Sources of Data .....	7
Integration Services .....	7
Components of Integration Services .....	8
Integration Services Console .....	9
Integration Server .....	9
Workflow of Integration Services .....	10
OLAP Models .....	10
Using OLAP Models .....	10
Fact Table .....	11
Dimension Tables .....	11
About Metaoutlines .....	11
Metaoutline Components .....	12
Using Hierarchies .....	13
Types of Hierarchies .....	13
Balanced .....	13
Unbalanced .....	13
Ragged .....	14
Alternate .....	14
Deploying Hierarchies .....	14
Standard .....	15
Recursive Hierarchies .....	15
Duplicate Members .....	15
<b>Chapter 2. Preparing Data Sources</b> .....	17
Defining User Needs .....	17
Defining Data Sources .....	18
Deciding to Create Staging Areas .....	19
Cleansing Data .....	19

Creating Views, Tables, and User-Defined Tables .....	20
Deciding to Create a View or a Table .....	20
Deciding to Create a User-Defined Table .....	21
Building Essbase Hierarchies from Recursive Tables .....	21
Building a Hierarchy Down to a Specific Level .....	21
Creating Aliases or UDAs for Members in a Recursive Table .....	21
Removing Unions .....	22
Transposing Columns and Rows .....	23
Denormalizing Source Data Tables .....	24
Adding Columns to Tables .....	25
Setting Up Columns to Support Member and Measure Properties .....	25
Joining Tables .....	25
Creating Indexes .....	26
Transforming Data .....	26
Deciding Which Tables Are Available to OLAP Model Creators .....	27
Selecting Tables for the Fact Table .....	28
Selecting Tables for Dimensions .....	28
Creating Aliases for Dimensions and Members .....	29
Creating Time and Accounts Dimensions .....	29
Preparing Data for Time Analysis .....	30
Associating Time Data with Measure Data .....	30
Working with Summary Data .....	30
Formatting Dates .....	31
Working with Data in Similar Time Periods .....	31
Accessing Tables in OLAP Metadata Catalog .....	32
Text Files as Data Sources .....	32
<b>Index .....</b>	<b>35</b>

# 1

## OLAP and Integration Services

### In This Chapter

OLAP .....	5
Multidimensional Databases.....	6
User Interactions with Data .....	6
Sources of Data .....	7
Integration Services.....	7
OLAP Models .....	10
About Metaoutlines .....	11
Using Hierarchies .....	13
Duplicate Members .....	15

Oracle Essbase Integration Services provides a suite of graphical tools to create OLAP models and metaoutlines and populate Oracle Essbase databases. Use data sources to define logical models representing data in an online analytical processing (OLAP) context. Then use the OLAP models to create metaoutlines that serve as templates for Essbase database outlines.

## OLAP

*OLAP* is designed for business managers to analyze consolidated enterprise data in real time. OLAP addresses complex “what if” questions, creating scenarios to test planning strategies.

Essbase supports OLAP, making possible a multidimensional, multiuser database that is accessed with standard retrieval tools.

Essbase Server supports multiple views of data sets so users can analyze relationships between data categories such as:

- How did Product A sell last month? How does this compare to the same month over the last five years? How will it sell next month?
- Did Product A sell better in particular regions?
- Did customers return Product A last year? Were returns due to defects? Did a specific plant manufacture defective products?
- Did commissions and pricing affect how salespeople sold Product A?

You can use Integration Services to create an Essbase database to answer these types of questions quickly. You can use Integration Services Console to create logical data models that represents source databases.

## Multidimensional Databases

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

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

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

Members can be parents of some members and children of other members. The Essbase outline indents members below one another to indicate a consolidation relationship.

## User Interactions with Data

Essbase Server consolidates and calculates data to provide different views of the data. Using a multidimensional database, users can perform several tasks:

### Consolidate (aggregate or roll up) data

In block storage, *consolidation* 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
```

### Create “what if” scenarios

Assume you set a sales goal of ten percent growth on all product lines. You can compare sales forecasts with actual sales data (retrieved from the *online transaction processing* [OLTP] database) to see how close you are to achieving your goals. If actual sales run lower than projected, salespeople can access the forecast data, input new sales scenarios, update the forecast data, and generate revised figures.

### Input strategic planning assumptions

Assume your company is planning 50 percent growth over three years. You know how many new products you need, but how many new people can you hire while still optimizing profits and gross margin?

In block storage, you can input projected sales and expenses and calculate downward to determine the projected cost of goods sold. If the results do not look practical, you can create different scenarios with different mixes of products, people, and expenses until you produce the profit picture that you require.

### Conduct spreadsheet operations

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

For example, you can drill down on the Year dimension to view each quarter. You can drill up from Chicago to view sales totals for the Central region.

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

See the Oracle Essbase Spreadsheet Add-in or Oracle Hyperion Smart View for Office, Fusion Edition documentation.

## Sources of Data

The data in a multidimensional database can originate from a variety of sources, such as OLTP databases, data warehouses, text , and spreadsheet files.

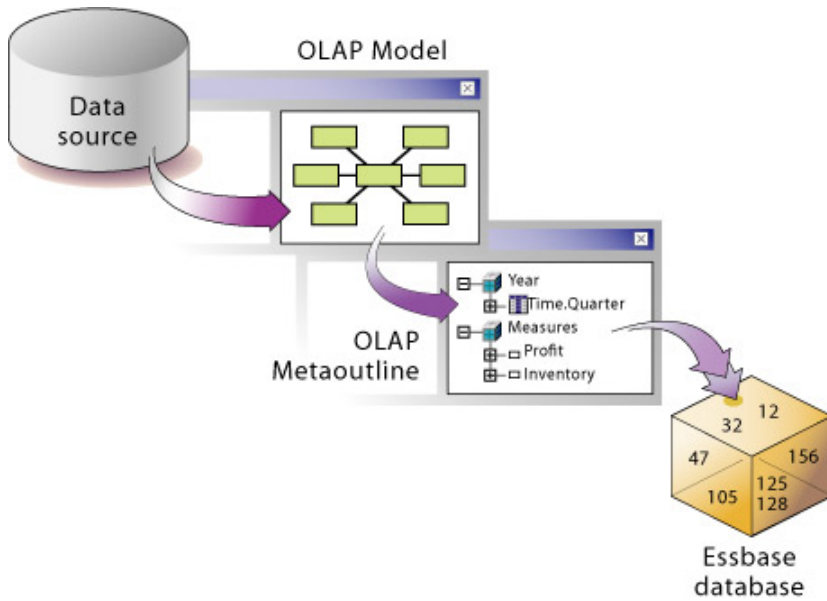
In relational databases, data is stored in rows and columns; in a data warehouse, data is stored in QueryCubes and InfoCubes.

## Integration Services

Integration Services transfers data from data sources to Essbase databases. After you determine which data to transfer, you consolidate it into a form useful for decision-support users. Then you identify the tables, rows, or columns that contain the data and determine how they map to the structure of the multidimensional database.

[Figure 1](#) illustrates the Integration Services workflow:

Figure 1 Workflow for Creating an Essbase Database from a Data Source



Use Integration Services Console for the following tasks:

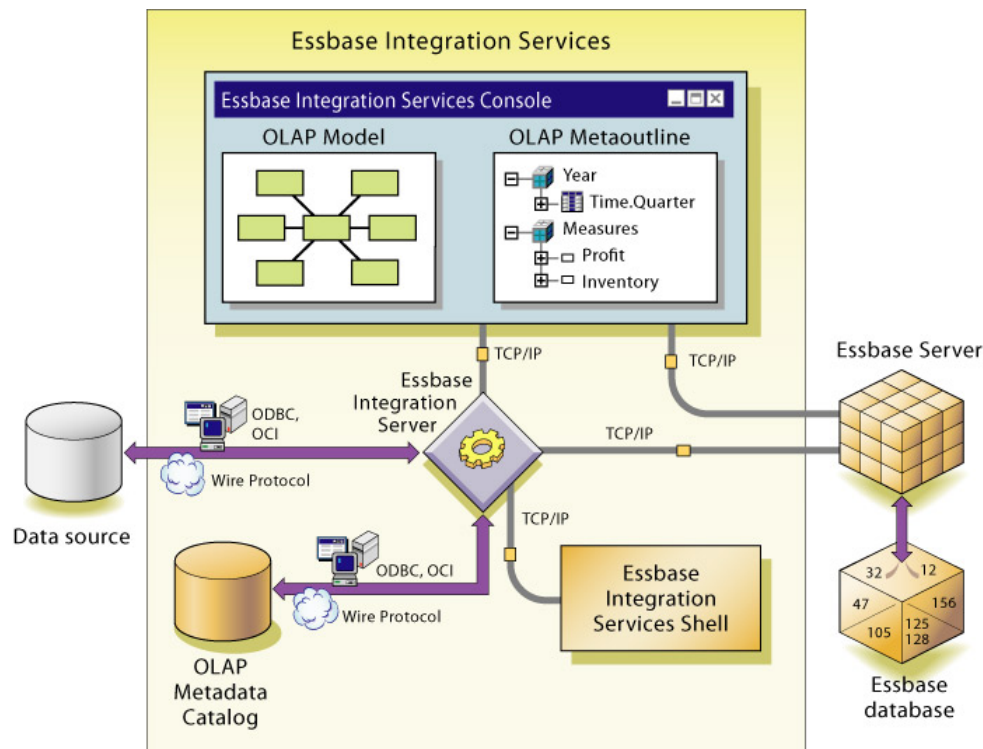
- Use tables, views, and columns in one or more data source databases to create an OLAP model. An *OLAP model* is a logical *star schema* consisting of a fact table surrounded by related dimension tables.
- Use the OLAP model to create a *metaoutline*, an template containing the structure and the rules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

## Components of Integration Services

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



Figure 2 Integration Services



## Integration Services Console

Use Integration Services Console to create OLAP models and metaoutlines and populate Essbase databases.

To create an OLAP model or metaoutline, connect to an OLAP Metadata Catalog and your data sources. See the Integration Services online help.

## Integration Server

Integration Server is the primary component of Integration Services. Integration Server is software that uses the information stored in the OLAP Metadata Catalog to extract from data sources the dimension and member names needed to build an associated Essbase outline.

When the outline is complete, Integration Server extracts data from the data sources, performs the operations specified in the associated metaoutline, and loads the data into the Essbase database. See the *Oracle Essbase Integration Services System Administrator's Guide*.

Integration Server includes several subcomponents, as illustrated in [Figure 2 on page 9](#):

- **OLAP Metadata Catalog:** A structured query language (SQL) relational database containing:
  - Metadata describing the nature, source, location, and type of data to retrieve
  - Metadata describing information required to generate Essbase outlines
  - OLAP models and metaoutlines

You can create multiple OLAP Metadata Catalogs to store OLAP models and metaoutlines. Using XML Import/Export, you can move OLAP models and metaoutlines to a different OLAP Metadata Catalog.

The OLAP Metadata Catalog is a data source that is configured for Open Database Connectivity (ODBC). See the *Oracle Hyperion Enterprise Performance Management System Installation and Configuration Guide* or the ODBC user documentation.

- **Essbase Integration Services Shell:** A command-line tool used to access Integration Server to load members and data into an Essbase database. See the *Oracle Essbase Integration Services System Administrator's Guide*.

## Workflow of Integration Services

Figure 2 on page 9 shows the Integration Services components used to prepare data for OLAP reporting through Essbase Server.

➤ To create Essbase databases from data sources:

- 1 Build an OLAP model based on data sources.**

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

- 2 Create a metaoutline from the OLAP model.**

Integration Server stores the metaoutline in OLAP Metadata Catalog.

- 3 Use the metaoutline to load members and data into the Essbase database.**

You can update the Essbase database with new members and data.

## OLAP Models

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

Integration Services creates an OLAP model that is a logical model, not a physical star schema, and is a logical representation of data values you select from data sources and report in Essbase.

## Using OLAP Models

Use an OLAP model to create metaoutlines. A metaoutline contains the basic structure required to build an Essbase outline and load data into Essbase. You can use one OLAP model as the basis for another by saving the original under a different name and editing it. You can create

any number of OLAP models to build metaoutlines. Each metaoutline, however, is based on one, specific OLAP model.

OLAP models have the following features:

- They can be the basis for multiple metaoutlines.
- They insulate the Essbase database outline from changes in the source database.
- They enable you to create hierarchies to structure and summarize data from a source database. You can use the hierarchies in multiple metaoutlines.

## Fact Table

The *fact table* is a container for all numeric facts (for the measures data values that vary over time). In the sample application provided, the fact table consists of the SALES relational table containing sales figures, cost of goods sold, opening and ending inventory quantities, and other columns of variable measures.

## Dimension Tables

A *dimension table*, such as a Product dimension, is a container for relational tables. Each dimension table contains data related to facts in the fact table.

When a dimension table joins to the fact table, that dimension table and other dimension tables joined to it form a dimension. A dimension in an OLAP model represents a dimension (or characteristic from a data warehouse) you want to report in Essbase. See “[Multidimensional Databases](#)” on page 6.

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

If your source is a relational database, when you build a metaoutline, you can create user-defined dimensions that do not exist in the associated OLAP model.

## About Metaoutlines

A metaoutline is a template containing the structure and rules for creating an Essbase outline. A metaoutline is based on the structure of an OLAP model. Metaoutlines have several features:

- They can be the basis for multiple Essbase outlines.
- They can be defined at a central location and used to create multiple Essbase outlines in multiple locations.
- They enable you to create Essbase databases on demand.
- They enable you to view sample Essbase outlines before building them.

- They automatically generate SQL to retrieve data from an external source.
- They enable you to filter data before you build the associated Essbase outline.
- They enable you to transform data as you create an Essbase outline.

## Metaoutline Components

A metaoutline has several components:

- **One or more measures.** *Measures* are data values and typically include items such as SALES and COGS (cost of goods sold). Every metaoutline used to build an Essbase outline requires at least one measure.

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

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

A dimension in a metaoutline creates a dimension in the associated Essbase outline. Every metaoutline used to create an Essbase outline must include at least two dimensions.

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

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

You can designate user-defined members as shared members. They can share storage space with other members of the same name.

- **Filters.** *Filters* determine which members of a member level that Integration Services adds to the associated Essbase outline. You can define transformation rules to determine what transformations Integration Services performs on the members of a member level as it builds the Essbase outline.
- **Optional attribute dimensions.** Attribute dimensions are based on attribute-enabled columns in the OLAP model. After an attribute dimension and member are created, you can define attribute properties, such as Boolean and numeric ranges, to view business data in finer detail than would otherwise be easily available.
- **Optional Hybrid Analysis low-level members.** Hybrid Analysis integrates source databases with Essbase multidimensional databases so that applications and reporting tools can directly retrieve data within both databases. When Hybrid Analysis is enabled, users of spreadsheets and report writer interfaces can access data contained in the Essbase database and drill down to data accessed directly from the data source.

- **Optional drill-through report members.** A drill-through report is based on an *intersection level* (member combination) that spreadsheet users double-click to start drill-through operations. Spreadsheet users can view or customize pre-defined drill-through reports that retrieve detail columns from data sources. Integration Server returns the drill-through report in the context of the data that spreadsheet users are viewing.
  - **Drill-through report to a URL.** Customize the drill-through SQL template, replacing the data source with an HTML source. Integration Services passes the URL to Oracle Essbase Spreadsheet Add-in or Oracle Hyperion Smart View for Office, Fusion Edition (or another drill-through client).
  - **Drill-through report to a secondary source.**
  - **Drill-through report to attributes and members enabled for Hybrid Analysis.** Select from several options for attribute drill-through reporting:
    - Attribute dimensions and attribute members as OLAP intersections.
    - Hybrid Analysis dimensions as OLAP intersections.
    - Hybrid Analysis member columns as intersections.
    - Attributes associated with members that have been enabled for Hybrid Analysis.

## Using Hierarchies

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

Hierarchies also use attributes to classify members logically within a dimension; for example, a Product dimension with attributes such as Size and Flavor.

You can create hierarchies for a metaoutline while creating OLAP models.

## Types of Hierarchies

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

### Balanced

A balanced hierarchy has multiple branches in a hierarchical tree, and each member is consistent with other members at the same level in each branch. For example, in a relational database, a dimension has branches for 2005 and 2006. In each of these branches, Q1 is at the same level, as are the months Jan, Feb, and Mar. Time dimensions typically have balanced hierarchies.

### Unbalanced

An unbalanced hierarchy contains branches with unequal numbers of member levels although the parent-child relationships are usually consistent from branch to branch. For example, a Sales

Personnel dimension has branches for Sales Manager East and West. Each of these branches has States. The Sales Manager East, however, has four states and the Sales Manager West has two states. Human resource dimensions sometimes have unbalanced hierarchies.

## Ragged

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

The North America branch has United States, Massachusetts, and Boston. The Europe branch has Greece, Athens because Greece does not have individual states like the United States. The State level for Greece is empty, causing a ragged hierarchy.

Geographical dimensions and product dimensions often have ragged hierarchies.

## Alternate

An alternate hierarchy is based upon an existing “primary” hierarchy but has alternate levels in the dimension. In [Table 1](#), the primary levels are in the four left columns, and the alternate levels are in the two right columns.

**Table 1** Alternate Hierarchy

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

## Deploying Hierarchies

A hierarchy is deployed using a standard or recursive method:

## Standard

Each attribute in the hierarchy defines one level. For example, in a Time dimension, the hierarchy is organized so that the attributes Year, Quarter, and Month define different levels, as illustrated in [Table 2](#).

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

**Table 2** Standard Hierarchy Deployment

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

## Recursive Hierarchies

The parent-child relationships of attributes are used to organize the hierarchy. For example, in a Sales Personnel dimension, the hierarchy is organized under the parent-child relationships shown in [Table 3](#).

**Table 3** Recursive Hierarchy Deployment

Parent Attribute	Child Attribute
North America	United States of America
United States of America	California
California	San Bernadino
San Bernadino	Los Angeles
Los Angeles	Glendale

## Duplicate Members

A data source or metaoutline may contain duplicate member names. In Integration Services Console, you can specify a metaoutline that, when loaded into an Essbase database, creates an Essbase outline that supports duplicate member names.

In order to support shared members in a duplicate member Essbase outline, the member column in the metaoutline must be associated with a member key column from the data source. Integration Server makes use of the assigned key to uniquely identify the base member. Internally, Integration Server generates its own unique identifier to differentiate between duplicate members.

When Integration Services performs a member load, Integration Server applies the unique identifier to duplicate names. From Integration Services clients, the member names can be retrieved using a qualified format, for example: [Product] . [300] . [300-30] [Diet] . [300-30]

You may choose to specify a metaoutline that, when loaded into Integration Services, does not create an outline that supports duplicate member names. In this case, when Integration Server performs a member load, it performs one of the following actions:

- It ignores the duplicate members. Integration Server does not load the members into the Integration Services outline.
- It creates the members as shared members. The data associated with a shared member is stored in the real member.

For more information, see Essbase online help.

**Note:** You should not create duplicate members as shared if the actual member or any shared member at the first leaf level has a filter. Instead, individually tag such a member as shared. You can prevent the creation of duplicate member names by transforming data when you create an OLAP model.



# 2

## Preparing Data Sources

### In This Chapter

Defining User Needs .....	17
Defining Data Sources .....	18
Deciding to Create Staging Areas .....	19
Cleansing Data .....	19
Creating Views, Tables, and User-Defined Tables .....	20
Adding Columns to Tables .....	25
Joining Tables .....	25
Creating Indexes .....	26
Transforming Data .....	26
Deciding Which Tables Are Available to OLAP Model Creators .....	27
Creating Time and Accounts Dimensions .....	29
Accessing Tables in OLAP Metadata Catalog .....	32
Text Files as Data Sources .....	32

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

Before creating an OLAP model, understand and define your business needs and your data sources. You may need to modify some data sources to make the transition from relational databases or data warehouses to multidimensional databases easier and more efficient.

For information about creating an OLAP model, see Integration Services Console Help.

## Defining User Needs

When you create an OLAP model, your goal is to create a multidimensional Essbase database. This topic assumes you understand design principles for multidimensional databases. See [“OLAP” on page 5](#) and the *Oracle Essbase Database Administrator's Guide*.

Before designing OLAP models, answer these questions:

**What data do users want to see in the Essbase database?**

Consider your business environment and the available data. Decide whether to create one or multiple Essbase databases.

### What is the level of detail users need in the Essbase database?

Consider your business environment and the performance requirements of the Essbase database. (In general, the less detail stored in the Essbase database, the faster the performance.)

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

### Which dimensions apply to each fact table row?

A dimension is a data category. Typical dimensions are Product, Market, and Time. Consider how users want to view data.

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

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

*Measures* are numeric quantities that vary over time. Examples of measures are quantity sold, cost of goods sold, and profit. Not all numeric values are measures, but all measures are numeric values.

## Defining Data Sources

After defining the data users need in the Essbase database, define the data sources. Consider these questions:

- **Is the data clean?**

Consider the quality and integrity of the source data. See [“Cleansing Data” on page 19](#).

- **Is the data calculated by a procedure and not stored; for example, a discount calculated for a specific product at a specific time?**

Create this information as tables in the source database. Consider using a staging area. See [“Deciding to Create Staging Areas” on page 19](#).

- **Is the data in a single structured query language (SQL) database or in multiple SQL data sources?**

Essbase Integration Server can access multiple SQL sources; however, you can consolidate SQL tables into a single SQL data source for each chosen source database.

- **Is the data in text (flat) files?**

Integration Services supports text files (flat files). If your data is in text files, you need to create text files as tables in the chosen source databases. Consider creating a staging area. See [“Deciding to Create Staging Areas” on page 19](#). Alternatively, after building the Essbase database outline, use Oracle Essbase Administration Services to load data. See Oracle Essbase Administration Services Online Help and [“Text Files as Data Sources” on page 32](#).

## Deciding to Create Staging Areas

A *staging area* is an RDBMS database you create to meet the needs of specific applications. Typically, a staging area is small compared to a data warehouse or an online transaction processing (OLTP) application. As a snapshot of data, it is not constantly updated with new data but is refreshed periodically.

Use a staging area for these tasks:

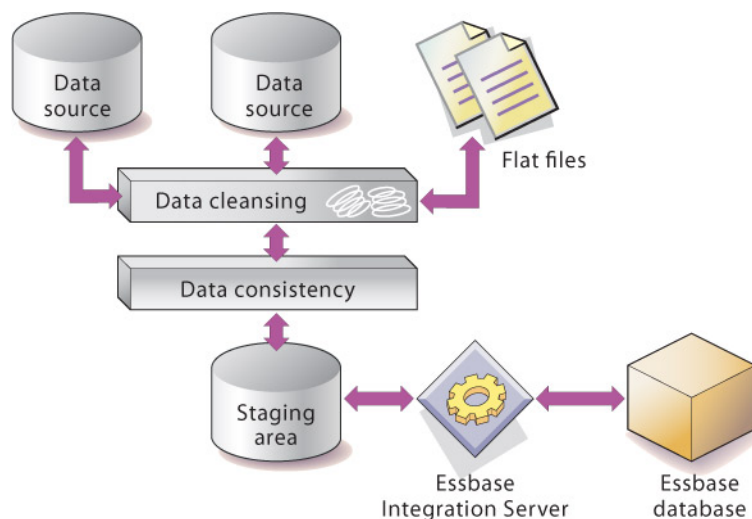
- Combine data from disparate or heterogeneous platforms without changing source data
- Fine-tune data for an application

For example, you can calculate a subset of the data source data and then run faster queries on the precalculated data in the staging area.

- Create tables or views to denormalize the data so that it maps more easily to an OLAP model (see [“Creating Views, Tables, and User-Defined Tables” on page 20](#))
- Transform data not consistently described (see [“Transforming Data” on page 26](#))

[Figure 3](#) shows a staging area as part of a data warehouse. Data is copied into the staging area from the source data and then converted.

**Figure 3** Data Warehouse with Staging Area



## Cleansing Data

Integration Server does not cleanse invalid or inconsistent data. Inconsistent data may include incorrect values, incorrect data types, or non-matching integrity constraints (rows that do not have entries for required key columns). Also, data is inconsistent if the same value is entered in different forms. Inconsistency often occurs when data is drawn from multiple sources or when users enter data incorrectly.

If source data is inconsistent, you cleanse the data. Cleansing data can be a simple process, such as making suspect data into nulls, or a more complex process requiring a data-cleansing tool.

## Creating Views, Tables, and User-Defined Tables

Integration Server does not distinguish between tables and views in data sources. You can use them (and user-defined tables) when building OLAP models.

Creating views, tables, user-defined tables, or synonyms can provide security and make the transition from data sources to multidimensional databases easier. By creating views, tables, or user-defined tables, the structure in the data source remain unchanged.

Create views, new tables, or user-defined tables for these tasks:

- Build an Essbase hierarchy down to a specific level in a recursive table (see [“Building Essbase Hierarchies from Recursive Tables” on page 21](#)).
- Create aliases in Essbase databases from data stored in multiple columns in source data tables (see [“Creating Aliases for Dimensions and Members” on page 29](#)).

Consider creating views, tables, or user-defined tables if source tables meet any of these criteria:

- The source tables contain unions. Integration Server does not generate SQL for unions. See [“Removing Unions” on page 22](#).
- The source tables have columns you want to transpose to rows. See [“Transposing Columns and Rows” on page 23](#).
- The source tables are in a packaged application. You may not know which tables contain the data that you need because the table names provided by the application may not be meaningful in your environment. You may need to ask an application specialist to create the required tables and views (with meaningful names) in a staging area in the target data source. See [“Deciding to Create Staging Areas” on page 19](#).
- The source tables are highly normalized. Normalized data is appropriately grouped and does not include redundant data. Consider denormalizing data. See [“Denormalizing Source Data Tables” on page 24](#).

## Deciding to Create a View or a Table

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

When deciding to create a view or a new table, consider these questions:

- Does the data already exist?  
If the data does not exist, create a table.
- Do you want to index columns that are not indexed in the source table?  
Create a table because you cannot index columns in a view. See [“Creating Indexes” on page 26](#).

- Do you want to index columns that contain data you need to transform?

Many data sources ignore indexes on columns with transformations. You probably need to create a new table. See [“Transforming Data” on page 26](#).

Create tables and views in the staging area. See [“Deciding to Create Staging Areas” on page 19](#).

## Deciding to Create a User-Defined Table

Create user-defined tables for the reasons described in [“Deciding to Create a View or a Table” on page 20](#). User-defined tables are created in the Integration Services Console, not in the relational database or data warehouse.

## Building Essbase Hierarchies from Recursive Tables

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

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

- To associate aliases or user-defined attributes (UDA) with members created from a recursive table, ensure the column with which you associate the alias or UDA is fully defined.
- When creating an OLAP model, join the recursive table to itself.
- When creating a metaoutline, select the parent or child column you want to filter on as a member level in the metaoutline.

## Building a Hierarchy Down to a Specific Level

To build the Essbase outline down to a specific level, create a view containing data for only the desired levels.

## Creating Aliases or UDAs for Members in a Recursive Table

If you want to associate aliases or UDAs with members created from a recursive source table, you must prepare the data:

**If the alias or UDA data is in a separate table**, you must complete specific steps when creating an OLAP model. These steps vary, depending on whether the column, with which you are associating the alias or UDA, is fully defined (see [Table 4](#) and [Table 2](#)).

**If the alias or UDA data is in the recursive source table**, 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 Essbase hierarchy). Thus the

parent column contains the lowest-level value in the hierarchy with a NULL value in the child column. In [Table 4](#), 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 4** Fully Defined Parent Column

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

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

In [Table 5](#), 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 5** Fully Defined Child Column

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

## Removing Unions

Integration Server does not generate SQL for unions. A *union* is join that combines the results of two SELECT statements. It is often used to merge lists of values contained in two tables. If the source tables use unions, you must create views of the data that do not use the unions before you can start to work with the data in Integration Services. See your RDBMS or data warehouse documentation.

## Transposing Columns and Rows

Transpose columns and rows to transition data from various sources to an Essbase database. Transpose the columns before you start to work with the data in Integration Server.

Consider the following example in which you create multiple Essbase measures (Init\_Sales, Subsequent\_Sales, and Return\_Sales) from a single database column (SALESTYPE).

The SALESACTUALS table contains the following columns:

**Table 6** SALESACTUALS Table

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

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

**Figure 4** Accounts Hierarchy



You want each SALESTYPE value to form the Essbase member: SALES to form the Init\_Sales member, RETURNS to form the Return\_Sales member, and SUBSEQUENT to form the Subsequent\_Sales member. Create a view or table that transposes row values into column values, for example:

**Table 7** 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 7](#):

```

)
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
  )

```

After defining 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 OLAP models. Normalized data is grouped and has no redundant data. You can use normalized source tables to create an OLAP model by specifying joins, but it may be more efficient to create a new table of denormalized data in the RDBMS.

If you use the denormalized data source, Integration Server does not need to compute the joins when building OLAP models.

In this example, in the first three tables, data is highly normalized, minimizing redundant data:

**Table 8** Normalized Product Family Data

FAMILYID	FAMILYDESC
100	Colas
200	Root Beer

**Table 9** Normalized Product Data

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

**Table 10** Normalized Product Description Data

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

The following shows denormalized data in one table:



**Table 11** Denormalized Product Data

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

## Adding Columns to Tables

Adding columns to tables in relational source databases eliminates several manual tasks required by Integration Services.

## Setting Up Columns to Support Member and Measure Properties

To add columns in relational database tables to use the member and measure properties in Integration Services:

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

**Note:** For purposes of explaining concepts, this chapter uses the column names of the Integration Services sample application.

## Joining Tables

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

- Tables that form the fact table. See [“Selecting Tables for the Fact Table” on page 28](#).
- Tables within each dimension branch. See [“Selecting Tables for Dimensions” on page 28](#).
- Dimension tables joined in the fact table.

When you join tables in the RDBMS, Integration Server detects joins while building OLAP models.

## Creating Indexes

Integration Server detects indexes (including bitmapped indexes) you have defined on source tables and uses them to create an Essbase outline and to load data. *Indexes* are pointers 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.

To improve performance:

- Define indexes on columns you use to filter data in the source database or in the OLAP model. For example, if the source database contains columns for city and state, and you filter on city or state (`SELECT * FROM Region WHERE State = Ca%`), then index the columns which you are filtering (here, State).
- If you are creating and filtering on alias names, index the column containing alias names. See [“Creating Aliases for Dimensions and Members” on page 29](#).
- Define bitmapped indexes on numeric data you use to filter the database. For example, if you filter on sales values `SELECT Product FROM ProdSales GROUPBY Product ORDERBY ProdId HAVING SUM(Sales)>15000`, then consider defining a bitmapped index on sales values. Most source databases support bitmapped indexes. See the documentation for the RDBMS or data warehouse that you are using.

## Transforming Data

You may need to transform data, for example, to change date formats.

Assume you want to create an OLAP model, combining data from sales and financial databases. If the sales database specifies New York as New\_York and the financial database specifies New York as NY, you can transform NY to New\_York in the staging area (see [“Deciding to Create Staging Areas” on page 19](#)) without changing source data.

You can do some data transformations in OLAP models and metatayoutlines. (For a list of available transformations, see Integration Services online help.) If you must do significant transformations on the data, consider using a data transformation tool before you use the data in Integration Server.

Transformations you must perform in source databases, and not in Integration Services, include these:

- Most frequently accessed transformations

For example, you want the Essbase database outline to include members for Year, Qtr, and Month, and 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 source table that contains the columns YEAR, QTR, and MONTH. The columns contain data transformed from the TRANSDATE column. You can index the YEAR, QTR, and MONTH columns to improve performance.

**Note:** Many data source databases ignore indexes on columns with transformations. 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 you want to split out before using the data in Integration Server. For example, if the source table has a PRODID column containing product identification information, transform the data in the data source so that it maps to the data categories you want to display in the Essbase outline. In [Figure 5](#), for example, the categories are CUSTID, STATE, and PRODNUM.

**Figure 5** Transforming a Product Code Stored in a PRODID Column



## Deciding Which Tables Are Available to OLAP Model Creators

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

For all source databases, if you have partial access to a table, Integration Server displays the whole table. In the OLAP model, Integration Server cannot filter, load data, or load members in a column for which you do not have SELECT permission.

**Note:** For IBM DB2, if you do not have access (SELECT permission) to a table, you cannot view columns or use the table to build OLAP models.

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

For security and administration, define access:

- Create specific RDBMS user IDs for users who create OLAP models.
- Create views of data and provide users who create OLAP models with access to the views rather than the original tables.
- Set conditions so that users access only a subset of the view. See the documentation for your RDBMS or data warehouse.

**Note:** When you connect to the source database, Integration Server provides read-only access to it, regardless of the access provided by the source database user ID.

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

## Selecting Tables for the Fact Table

A fact table is a logical container for the relational tables defining the data values (measures) for each dimension intersection in an OLAP model.

The type of Essbase database you create determines which source tables to include in the fact table. See [“Defining User Needs” on page 17](#).

**Note:** If you configure ODBC data sources for Teradata, invoke the UserXViews option.

When selecting source tables for the fact table, consider data distribution in tables and how the data relates to dimensions in the Essbase database.

Select tables containing measure data that Essbase users want to see. For example, if the source database contains ORDERS and ORDERDETAILS tables, include both tables.

**Table 12** ORDERS and ORDERDETAILS Tables for the Fact Table

Orders	ORDERID	TRANSDATE	SHIPID	EMPTYID
Order Details	ORDERID	PRODID	NUMUNITS	UTPRICE

Select tables that represent dimensions (data categories) on which users want to analyze data. For example, if users need to analyze product sales by time and sales channel, select tables containing key columns for these dimensions.

You may improve performance by creating aggregate data in tables instead of in views. See [“Creating Views, Tables, and User-Defined Tables” on page 20](#). See [“Deciding to Create Staging Areas” on page 19](#).

If the Essbase users need to only aggregate data, then include tables or views of aggregated data. You can add more detail as you build the OLAP model.

**Note:** If your data source is a relational database, you can improve performance by setting up primary and foreign keys, and joining the tables that form the fact table. See [“Adding Columns to Tables” on page 25](#).

## 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 defining a potential Essbase dimension. A dimension table can join to other dimension tables, forming a *dimension branch*.

The data needed for analysis in Essbase applications determines which source tables used to create dimensions. See [“Defining User Needs” on page 17](#). When selecting source tables for dimensions, select tables that include the maximum amount of information about the data category (dimension).

## Creating Aliases for Dimensions and Members

An alias often provides an easily identifiable label, such as a product name or product description, for columns in relational data sources. For example, the 200-20 member could have the alias Root Beer.

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

If the source data contains information you need to create aliases in an Essbase outline, include the relevant data as columns in the associated OLAP model. Then, when you create a metaoutline, you can retrieve alias information from columns of the OLAP model.

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

If alias information for a single alias is in multiple columns, complete one of these procedures:

- In the source database, create a view or table to concatenate (and, if necessary, transform) alias information into a single column. Creating a view or table is preferred if alias information is stored in three or more columns.
- In the OLAP model, concatenate (and, if necessary, transform) columns containing alias information. See the Integration Services Console online help.
- In the OLAP model, use the pass-through feature to run a procedure that uses relational logic to retrieve alias information from the columns you specify. If a field in one column is empty, the procedure tells the source database to retrieve alias information for that field from a different column.

For information on pass-through transformations, see Integration Services Console online help.

## Creating Time and Accounts Dimensions

In an OLAP model, you can create time and accounts dimensions that relate directly to the Essbase outline. The *time dimension* table contains date-related columns from the relational data source. The *accounts dimension* table contains measures columns that duplicate columns in the fact table.

## Preparing Data for Time Analysis

If your business requires time-related analysis, include a time dimension in the Oracle Essbase outline. A time dimension includes members for the time periods that you report on; for example, months.

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

- One or more time-related columns associated with measure data.
- One or more tables containing user-defined calendars. A user-defined calendar maps the time-related column in the fact table to a specific business calendar.

## Associating Time Data with Measure Data

For Integration Server to map data that you are measuring to specific time periods, the source data for the fact table must contain one or more time-related columns.

## 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 13** 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	

STATE	PRODCODE	TRANSDATE	SALES	COGS	...
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. A row can also summarize data by other periods; for example, by week.

## Formatting Dates

Integration Server supports three *data types*: string, numeric, and datetime. The data type of the source data column containing 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 can be included in the source data as separate string or numeric columns.

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

## Working with Data in Similar Time Periods

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

**Table 14** Columns Defining Time Periods

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

To create a dimension of the type time, Integration Server requires that all time identifiers be in one column and that accounts identifiers be in separate columns, as shown in [Table 15](#). If data

is stored with each month as an individual column, as shown in [Table 14](#), restructure the data. Individual months should be data values in a MONTH column, and the measures categories, such as SALES, should be separate columns.

**Table 15** Time Periods 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	<b>1</b>	<b>401.00</b>	<b>260.00</b>
100-10-C001-S002-P001	CA	<b>2</b>	<b>143.00</b>	<b>101.00</b>
100-10-C001-S002-P001	CA	<b>3</b>	<b>378.00</b>	<b>226.00</b>
100-10-C001-S002-P001	AZ	1	672.00	403.00

You can write a program to convert the data to a table that uses the appropriate format or use the source database to transpose the columns (see [“Transposing Columns and Rows” on page 23](#)).

## Accessing Tables in OLAP Metadata Catalog

If 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 (Microsoft SQL Server) or synonyms for the tables (Oracle, IBM DB2). See the *Oracle Essbase Integration Services System Administrator's Guide*.

## Text Files as Data Sources

In Oracle Essbase Integration Services, you can use text files (*flat files*) as data sources, but note these conditions:

- Open Database Connectivity (ODBC) drivers cannot extract month names from date fields in text files, for example, “January” cannot be extracted from “01/30/04.” During a member load, you can apply a transformation so that the numerical value is converted into the name of the month.
- If the data source consists of several text files that resemble or consist of tables, the importing of data will be much like that of importing relational data. If the data source is one large, contiguous text file with several different columns, it is likely that you will need to “cleanse” your data beforehand (see [“Cleansing Data” on page 19](#)). Or you may drop some tables



multiple times to represent dimensions. You may also need to define your joins explicitly for optimal performance (see [“Joining Tables” on page 25](#)).



# Index

## A

accessing tables in OLAP Metadata Catalog, [32](#)  
 accounts dimension  
   sample hierarchy, [23](#)  
   use in Essbase outline, [29](#)  
 adding columns to tables, [25](#)  
 aggregation  
   choosing level of, [18](#)  
   defined, [6](#)  
 aliases  
   concatenating, [29](#)  
   creating for dimension or member names, [29](#)  
   multiple aliases, using, [29](#)  
   pass-through transformations and, [29](#)  
   recursive tables and, [21](#)

## B

balanced hierarchies, [13](#)  
 batch processing, Integration Services Shell, [10](#)  
 bitmapped indexes, defined, [26](#)

## C

cleansing data, [19](#)  
 columns  
   adding to tables, [25](#)  
   availability, aliases, [29](#)  
   fully-defined in recursive tables, [21](#)  
   indexes, [26](#)  
   supporting member and measure properties, [25](#)  
   time-related, defining in source data, [30](#)  
   transposing with rows, [23](#)  
 concatenation and aliases, [29](#)  
 Console. *See* Integration Services Console  
 consolidation  
   defined, [6](#)  
   drill down and, [7](#)  
   example, [6](#)

members, [6](#)  
 creating  
   Essbase databases, workflow, [10](#)  
   hierarchies, [21](#)  
   indexes, [26](#)  
 customer comparisons, [5](#)

## D

data  
   cleansing, [19](#)  
   combining multiple sources, [19](#)  
   denormalizing, [24](#)  
   drill down, [7](#)  
   ensuring consistency, [19](#)  
   pivoting to change view, [7](#)  
   preparing for time analysis, [30](#)  
   source preparation, [17](#)  
   transforming in data source, [26](#)  
   types of, [17](#)  
 data sources  
   build levels, [21](#)  
   defining, [18](#)  
   parent-child, [21](#)  
   spreadsheet files, [7](#)  
   SQL, [7](#)  
   text files, [7](#)  
   workflow for transferring, [10](#)  
 data types and cleansing, [19](#)  
 data warehouse, diagram, [19](#)  
 data warehouses  
   source of data, [7](#)  
 databases. *See* data sources  
 dates, formatting in source data, [31](#)  
 denormalizing data, [24](#)  
 deploying hierarchies, [15](#)  
 deploying hierarchies, [14](#), [15](#)  
 detail data

- choosing level of, 18
- retrieving, 7
- dimension branches, defined, 28
- dimension names, creating aliases for, 29
- dimension tables
  - defined, 11, 28
  - OLAP models, 11
- dimensions
  - choosing, 18
  - choosing tables for, 28
  - creating in a metaoutline, 11
  - defined, 6, 10, 12, 17, 28
  - examples, 6
  - frequency of change, 6
  - members, 6
  - Scenario, 11
  - use in consolidation, 6
  - using in metaoutlines, 11
- drill down, 7
- drill up, 7
- drill-through report members, 13
- drill-through reports, 13
- duplicate members, 15

## E

- Essbase
  - accounts dimension, 29
  - databases, workflow, 10
  - designing a database for user needs, 17
  - Essbase Server, defined, 5
  - hierarchies, creating, 21
  - members, compared to metaoutline member levels, 12
  - users, 6
- Essbase Integration Server. *See* Integration Server
- Essbase Integration Services. *See* Integration Services
- Essbase Integration Services Shell. *See* Integration Services Shell
- Essbase Server, defined, 5
- Essbase Server, overview of, 6

## F

- fact table
  - choosing tables for, 28
  - compared with dimensions, 18
  - defined, 11, 28

- measures and, 28
- role in OLAP model, 11
- facts, defined, 10, 17
- filters, 26
- flat files
  - as a data source, 18, 32
  - combining with SQL data sources, 19
- forecasting, 6
- fully defined columns, defined, 21

## G

- general ledger data, 31

## H

- hierarchies
  - balanced, 13
  - building from recursive tables, 21
  - consolidation and, 6
  - creating, 21
  - data sources for building to a specific level, 21
  - deploying, 14, 15
  - ragged, 14
  - recursive, 15
  - standard, 14, 15
  - types, 13
  - unbalanced, 13
  - using, 13
- Hyperion Essbase. *See* Essbase
- Hyperion Integration Server. *See* Integration Server
- Hyperion Integration Server Desktop. *See* Integration Services Console

## I

- IBM DB2, tables displayed in, 27
- indenting, 6
- indexes
  - working with, 26
- indexes, working with, 26
- Integration Server, defined, 9
- Integration Services
  - about, 7
  - data warehouse, 19
  - defined, 5
  - Integration Services Console, defined, 9
  - Integration Services Shell, defined, 10
  - OLAP Metadata Catalog, defined, 10

- product family, [9](#), [10](#)
- workflow, [8](#), [10](#)
- Integration Services Console, defined, [9](#)
- Integration Services Shell
  - batch processing, [10](#)
  - defined, [10](#)
- integrity constraints, cleansing data, [19](#)
- intersections, defined, [6](#)

## J

- joins
  - joining tables in your RDBMS, [25](#)
  - reducing through denormalization, [24](#)
  - unions, removing, [22](#)

## K

- keys, columns, transformations and, [27](#)

## L

- legacy data, preparing, [31](#)
- levels, data sources for building, [21](#)
- logical model, description, [10](#)

## M

- measure filters. *See* filters
- measures
  - bitmapped indexes and, [26](#)
  - data, associating with time periods, [30](#)
  - defined, [12](#), [18](#)
  - fact table and, [28](#)
  - using with columns, [25](#)
- member filters. *See* filters
- member levels, defined, [12](#)
- member names, creating aliases, [29](#)
- members
  - consolidation, [6](#)
  - defined, [6](#)
  - dimensions, [6](#)
  - examples, [6](#)
  - OLAP models, [6](#)
  - using with columns, [25](#)
- metadata
  - accessing OLAP Metadata Catalog, [32](#)
  - defined, [9](#)
- metadata repository. *See* OLAP Metadata Catalog

- metadata, OLAP Metadata Catalog storage, [9](#)
- metaoutlines
  - advantages of, [11](#)
  - components, [12](#)
  - creating dimensions, [11](#)
- models. *See* OLAP models
- multidimensional databases
  - defined, [6](#)
  - uses, [6](#)
  - workflow, [7](#), [10](#)
- multiple alias tables, using, [29](#)

## N

- normalized data, denormalizing, [24](#)
- numeric data, indexes and, [26](#)

## O

- OLAP Builder. *See* metaoutlines
- OLAP Catalog. *See* OLAP Metadata Catalog
- OLAP Command Interface. *See* Integration Services Shell
- OLAP Integration Server. *See* Integration Server
- OLAP Metadata Catalog
  - accessing tables, [32](#)
  - configuring for ODBC, [10](#)
  - defined, [9](#), [10](#)
- OLAP metaoutline. *See* metaoutlines
- OLAP models
  - choosing which tables are displayed, [27](#)
  - components, [10](#)
  - creating, [9](#)
  - defined, [11](#)
  - dimension tables, [11](#)
  - dimensions, [6](#)
  - fact table, [11](#)
  - graphic illustration, [11](#)
  - members, [6](#)
  - uses, [5](#)
- OLAP Server. *See* Essbase Server
- Online Analytical Processing (OLAP), defined, [5](#)

## P

- parent-child data source, [21](#)
- pass-through transformations, aliases and, [29](#)
- performance
  - bitmapped indexes and, [26](#)

- increasing through denormalization, [24](#)
- permissions, required in the SQL data source, [27](#)
- perspective, changing data on a spreadsheet, [7](#)
- pivoting, using to change data view, [7](#)
- privileges. *See* permissions
- product code, example, [27](#)
- product comparisons, [5](#)

## R

- ragged hierarchies, [14](#)
- read-only access, SQL data sources and, [27](#)
- recursive hierarchies, [15](#)
- recursive tables
  - aliases and, [21](#)
  - creating Essbase hierarchies from recursive tables, [21](#)
  - defined, [21](#)
  - defining for alias or UDAs, [21](#)
  - fully-defined columns for, [21](#)
  - rules for, [21](#)
- redundant data, denormalizing, [24](#)
- relational databases. *See* data sources
- repository. *See* OLAP Metadata Catalog
- roll-ups, defined, [6](#)
- rows, transposing with columns, [23](#)

## S

- sample, hierarchy, accounts dimension, [23](#)
- scenarios, [7](#)
- security
  - SQL data sources, [27](#)
  - views and, [20](#)
- SELECT
  - permission, [27](#)
  - statements, combining, [22](#)
- snapshot, staging area, [19](#)
- source data
  - denormalizing, [24](#)
  - transforming, [26](#)
- sources. *See* data sources
- spreadsheet files, data sources, [7](#)
- SQL data sources
  - combining, [19](#)
  - combining flat files, [19](#)
  - consolidating multiple, [18](#)
  - read-only access to, [27](#)

- required permission for, [27](#)
- security, [27](#)
- staging area, [19](#)
- SQL Drill-Through. *See* drill-through reports
- staging area
  - advantages of, [19](#)
  - deciding whether to create, [19](#)
  - defined, [19](#)
  - diagram, [19](#)

- standard hierarchies, [14](#), [15](#)
- star schemas. *See* OLAP models
- strategic planning, [7](#)
- summary source data, associating with time periods, [30](#)

## T

- tables
  - accessing tables in OLAP Metadata Catalog, [32](#)
  - choosing which tables are displayed in OLAP models, [27](#)
  - creating to denormalize data, [24](#)
  - deciding whether to create new tables, [20](#)
  - denormalizing, [24](#)
  - joining, [25](#)
  - reasons to create, [20](#)
  - recursive
    - aliases and, [21](#)
    - defined, [21](#)
    - fully-defined columns for, [21](#)
    - rules for, [21](#)
    - UDAs or aliases, [21](#)
- text file as a data source, [7](#), [32](#)
- time
  - analysis, preparing data for, [30](#)
  - associating with measures data, [30](#)
  - time-related columns, defining in source data, [30](#)
- time dimension
  - defined, [30](#)
  - use in Essbase outline, [29](#)
- transformations
  - defining in SQL data sources, [26](#)
  - example in data source, [27](#)
  - indexes and, [26](#)
  - key columns and, [27](#)
- transposing columns and rows, [23](#)
- trends, [5](#)

**U**

- UDAs, creating in recursive tables, [21](#)
- unbalanced hierarchies, [13](#)
- unions, removing, [22](#)
- user needs, defining, [17](#)
- user privileges. *See* permissions
- user-defined dimensions, defined, [11](#)
- user-defined tables, [20](#)
- users, Essbase, [6](#)

**V**

- variables, defined, [18](#)
- views
  - deciding whether to create new views, [20](#)
  - reasons to create, [20](#)

**W**

- what if scenarios, example, [6](#)
- WHERE clause. *See* filters
- workflow
  - Integration Services, [8](#)
  - Integration Services product family, [10](#)

A B C D E F G H I J K L M N O P R S T U V W