## Contents

### Chapter 1. OLAP and Integration Services
- OLAP ............................................................... 5
- Multidimensional Databases .............................................. 6
- SAP BW Data Warehouses ............................................... 6
- User Interactions with Data ............................................. 8
- Sources of Data ........................................................ 8
- Integration Services ..................................................... 9
  - Components of Integration Services .................................. 9
  - Integration Services Console .......................................... 10
  - Integration Server ........................................................ 10
  - Workflow of Integration Services ..................................... 11
- OLAP Models ........................................................ 11
  - Using OLAP Models .................................................. 11
  - Fact Table ........................................................... 12
  - Dimension Tables ..................................................... 12
- About Metaoutlines .................................................... 12
  - Metaoutline Components ............................................. 13
- Using Hierarchies ..................................................... 14
  - Types of Hierarchies .................................................. 14
  - Deploying Hierarchies ................................................ 16
- Duplicate Members .................................................... 17

### Chapter 2. Preparing Data Sources
- Defining User Needs ................................................... 20
- Defining Data Sources .................................................. 20
- Deciding to Create Staging Areas ..................................... 21
- Cleansing Data ......................................................... 22
- Creating Views, Tables, and User-Defined Tables .................... 22
  - Deciding to Create a View or a Table .................................. 23
  - Deciding to Create a User-Defined Table ................................ 23
- Fact Tables in SAP BW .................................................. 23
  - Building Essbase Hierarchies from Recursive Tables ............... 24
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Removing Unions</td>
<td>25</td>
</tr>
<tr>
<td>Transposing Columns and Rows</td>
<td>25</td>
</tr>
<tr>
<td>Denormalizing Source Data Tables</td>
<td>27</td>
</tr>
<tr>
<td>Adding Columns to Tables</td>
<td>28</td>
</tr>
<tr>
<td>Setting Up Columns to Support Member and Measure Properties</td>
<td>28</td>
</tr>
<tr>
<td>Joining Tables</td>
<td>28</td>
</tr>
<tr>
<td>Optimizing Tables for Use with SAP BW</td>
<td>28</td>
</tr>
<tr>
<td>Indexes for Small Tables</td>
<td>29</td>
</tr>
<tr>
<td>Push Down Operations</td>
<td>29</td>
</tr>
<tr>
<td>Creating Indexes</td>
<td>29</td>
</tr>
<tr>
<td>Transforming Data</td>
<td>30</td>
</tr>
<tr>
<td>Deciding Which Tables Are Available to OLAP Model Creators</td>
<td>31</td>
</tr>
<tr>
<td>Selecting Tables for the Fact Table</td>
<td>32</td>
</tr>
<tr>
<td>Selecting Tables for Dimensions</td>
<td>33</td>
</tr>
<tr>
<td>Creating Aliases for Dimensions and Members</td>
<td>33</td>
</tr>
<tr>
<td>Creating Time and Accounts Dimensions</td>
<td>34</td>
</tr>
<tr>
<td>Preparing Data for Time Analysis</td>
<td>34</td>
</tr>
<tr>
<td>Associating Time Data with Measure Data</td>
<td>34</td>
</tr>
<tr>
<td>Working with Summary Data</td>
<td>34</td>
</tr>
<tr>
<td>Accounts Dimensions and SAP BW</td>
<td>36</td>
</tr>
<tr>
<td>Time Dimensions and SAP BW</td>
<td>37</td>
</tr>
<tr>
<td>Slowly Changing Dimensions in SAP BW</td>
<td>37</td>
</tr>
<tr>
<td>Attribute-Enabled Columns in SAP BW</td>
<td>37</td>
</tr>
<tr>
<td>Accessing Tables in OLAP Metadata Catalog</td>
<td>40</td>
</tr>
<tr>
<td>Mapping to DB2 Cube Views</td>
<td>40</td>
</tr>
<tr>
<td>Text Files as Data Sources</td>
<td>41</td>
</tr>
</tbody>
</table>

Index ........................................................................ 43
Oracle’s Essbase® Integration Services provides a suite of graphical tools to create OLAP models and metaoutlines and populate Oracle’s Hyperion® Essbase® – System 9 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 application.

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.

SAP BW Data Warehouses

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

Note:

Support for SAP BW is available through Hyperion® System™ 9 BI+™ Analytic Integration Connector for SAP BW™, a separately purchased and licensed option.

SAP BW data warehouses use different terminology for components that are analogous to multidimensional components. In SAP BW:

- A dimension is a characteristic.
- A dimension denotes a group of related characteristics; for example, a time dimension refers to the characteristics Calendar Year and Fiscal Year.
- A member is a characteristic value.
- A measure is a key figure. It is a quantifiable value; for example, Gross Sales.

Table 1 maps SAP BW data source components to relational and XML targets.
<table>
<thead>
<tr>
<th>Source (BW/ODBO)</th>
<th>Target (Relational Model)</th>
<th>Target (XML Model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Name</td>
<td></td>
<td>ModelDim element, name attribute</td>
</tr>
<tr>
<td>Time dimensions</td>
<td>Time table</td>
<td>ModelDim element, modelDim attribute</td>
</tr>
<tr>
<td></td>
<td>Fact table</td>
<td>ModelDim element, modelDim attribute ModelLogicalJoin element, view1Name attribute</td>
</tr>
<tr>
<td>Level Name</td>
<td></td>
<td>ModelView element, name attribute ModelLogicalJoin element, view1Name attribute</td>
</tr>
<tr>
<td>Level Name</td>
<td>Dimension table</td>
<td></td>
</tr>
<tr>
<td>Hierarchy Name</td>
<td>Parent/child table (for alternate hierarchies)</td>
<td>ModelHierarchy element, name attribute ModelPhysicalJoin element, table1Name attribute, table2Name attribute</td>
</tr>
<tr>
<td>Member Unique Name</td>
<td></td>
<td>ModelHierarchyMember element, viewMemberName attribute</td>
</tr>
<tr>
<td>Member Unique Name</td>
<td>Column in fact, dimension, parent/child tables</td>
<td>ModelViewMember element, name attribute ModelLogicalJoin element, memberName attributes ModelPhysicalJoin element, column1Name attribute</td>
</tr>
<tr>
<td>Member Name</td>
<td>Column in dimension, parent/child tables (used to join parent/child table to dimension table)</td>
<td>ModelViewMember element, name attribute ModelLogicalJoin element, memberName attributes</td>
</tr>
<tr>
<td>Member Caption</td>
<td>Column in dimension, parent/child tables</td>
<td>ModelViewMember element, name attribute</td>
</tr>
<tr>
<td>Parent Unique Name</td>
<td>Column in parent/child tables</td>
<td>ModelViewMember element, name attribute ModelPhysicalJoin element, column2Name attribute</td>
</tr>
<tr>
<td>Property Name (dimension properties)</td>
<td>Columns in dimension, parent/child tables</td>
<td>ModelViewMember element, name attribute</td>
</tr>
<tr>
<td>Property Name (dimension properties)</td>
<td>Columns in dimension, parent/child tables</td>
<td>ModelViewMember element, drill-through type attribute</td>
</tr>
<tr>
<td>Measure Unique Name</td>
<td>Column in fact table</td>
<td>ModelViewMember element, name attribute</td>
</tr>
<tr>
<td>Measure Aggregator</td>
<td></td>
<td>ModelViewMember element, aggregateType attribute</td>
</tr>
</tbody>
</table>
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 Essbase Spreadsheet Add-in or Oracle’s Hyperion® Smart View for Office 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 such as an SAP BW, 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.
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. If your data source is an SAP BW data warehouse, Integration Services extracts characteristic names and characteristic value names.

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

Integration Server includes several subcomponents, as illustrated in Figure 2 on page 10:

- **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 Essbase Integration Services Installation 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 Essbase Integration Services System Administrator’s Guide.

### Workflow of Integration Services

Figure 2 on page 10 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 (or key figure data values in SAP BW) 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 SAP BW data warehouses, facts are key figure values. 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.

  In SAP BW data warehouses, measures are *key figures*. They are always numerical.

  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. In SAP BW data warehouses, dimensions are *characteristics* which contain *characteristic values* (or members).

  In SAP BW data warehouses, *dimension* is used to denote a group of related characteristics; for example, a time dimension refers to the characteristics, Calendar Year and Fiscal Year.

  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 Essbase Spreadsheet Add-in or Hyperion Smart View for Office (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. In SAP BW data warehouses, characteristics have analogous branches.

## 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 2, the primary levels are in the four left columns, and the alternate levels are in the two right columns.

### Table 2  Alternate Hierarchy

<table>
<thead>
<tr>
<th>Prod Code</th>
<th>Prod Alias</th>
<th>Gen_01</th>
<th>Gen_02</th>
<th>Alt_Gen_01</th>
<th>Alt_Gen_02</th>
</tr>
</thead>
<tbody>
<tr>
<td>100-10</td>
<td>Cola</td>
<td>Cola vs. Non-cola</td>
<td>Colas</td>
<td>Regular vs. Diet</td>
<td>Regular</td>
</tr>
<tr>
<td>100-20</td>
<td>Diet Cola</td>
<td>Cola vs. Non-cola</td>
<td>Colas</td>
<td>Regular vs. Diet</td>
<td>Diet</td>
</tr>
<tr>
<td>100-30</td>
<td>Decaf Cola</td>
<td>Cola vs. Non-cola</td>
<td>Colas</td>
<td>Regular vs. Diet</td>
<td>Regular</td>
</tr>
<tr>
<td>200-10</td>
<td>Vanilla</td>
<td>Cola vs. Non-cola</td>
<td>Non-colas</td>
<td>Regular vs. Diet</td>
<td>Regular</td>
</tr>
<tr>
<td>200-20</td>
<td>Diet Vanilla</td>
<td>Cola vs. Non-cola</td>
<td>Non-colas</td>
<td>Regular vs. Diet</td>
<td>Diet</td>
</tr>
<tr>
<td>200-30</td>
<td>Cream</td>
<td>Cola vs. Non-cola</td>
<td>Non-colas</td>
<td>Regular vs. Diet</td>
<td>Regular</td>
</tr>
<tr>
<td>200-40</td>
<td>Diet Cream</td>
<td>Cola vs. Non-cola</td>
<td>Non-colas</td>
<td>Regular vs. Diet</td>
<td>Diet</td>
</tr>
<tr>
<td>400-10</td>
<td>Grape</td>
<td>Cola vs. Non-cola</td>
<td>Fruit Sodas</td>
<td>Regular vs. Diet</td>
<td>Regular</td>
</tr>
<tr>
<td>400-20</td>
<td>Diet Grape</td>
<td>Cola vs. Non-cola</td>
<td>Fruit Sodas</td>
<td>Regular vs. Diet</td>
<td>Diet</td>
</tr>
<tr>
<td>400-30</td>
<td>Orange</td>
<td>Cola vs. Non-cola</td>
<td>Fruit Sodas</td>
<td>Regular vs. Diet</td>
<td>Regular</td>
</tr>
</tbody>
</table>
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 3.

**Note:**

Standard deployment may be used with all hierarchy types.

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Month</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003</td>
<td>1st</td>
<td>Jan</td>
</tr>
<tr>
<td>2003</td>
<td>1st</td>
<td>Feb</td>
</tr>
<tr>
<td>2003</td>
<td>1st</td>
<td>Mar</td>
</tr>
<tr>
<td>2004</td>
<td>1st</td>
<td>Jan</td>
</tr>
<tr>
<td>2004</td>
<td>1st</td>
<td>Feb</td>
</tr>
<tr>
<td>2004</td>
<td>1st</td>
<td>Mar</td>
</tr>
</tbody>
</table>

**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 4.

<table>
<thead>
<tr>
<th>Parent Attribute</th>
<th>Child Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>North America</td>
<td>United States of America</td>
</tr>
<tr>
<td>United States of America</td>
<td>California</td>
</tr>
<tr>
<td>California</td>
<td>San Bernadino</td>
</tr>
</tbody>
</table>
Parent Attribute | Child Attribute
--- | ---
San Bernadino | Los Angeles
Los Angeles | Glendale

**Recursive Hierarchies and SAP BW**

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

**Note:**

We strongly recommend you not edit recursive hierarchies imported from SAP BW data warehouses into Integration Services unless you understand data warehousing and data modeling.

**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.
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 (key figure data values in SAP BW) 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.

Note:

Many tasks discussed here cannot be performed within Integration Services if you are using data from an SAP BW data warehouse. See the Hyperion System 9 BI+ Analytic Integration Connector for SAP BW documentation.
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 Hyperion Essbase - System 9 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. In SAP BW data warehouses, multidimensional dimensions are characteristics.

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. In SAP BW data warehouses, measures are key figures. Examples of measures are quantity sold, cost of goods sold, and profit. Not all numeric values are measures, but all measures are numeric values.

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 22.

- 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 21.

- 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 21. Alternatively, after building the Essbase database outline, use Oracle's Essbase® Administration Services to load data. See Oracle's Essbase® Administration Services Online Help and “Text Files as Data Sources” on page 41.

- **Is the data in SAP BW data warehouses?**

  Note differences between the terminology used in relational databases and SAP BW data warehouses. See “SAP BW Data Warehouses” on page 6.

---

**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.

**Note:**

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

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 22)
- Transform data not consistently described (see “Transforming Data” on page 30)

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.
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 24).
- Create aliases in Essbase databases from data stored in multiple columns in source data tables (see “Creating Aliases for Dimensions and Members” on page 33).

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 25.
● The source tables have columns you want to transpose to rows. See “Transposing Columns and Rows” on page 25.

● 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 21.

● 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 27.

**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 29.

● **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 30.

Create tables and views in the staging area. See “Deciding to Create Staging Areas” on page 21.

**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 23. User-defined tables are created in the Integration Services Console, not in the relational database or data warehouse.

**Fact Tables in SAP BW**

When creating an OLAP model from a relational data source, select one or more relational source tables to use as a fact table, time dimension, accounts dimension, and other dimensions. When working with an SAP BW data source, you do not have the option of individually selecting views to use as components of the OLAP model.

When you import an SAP BW model, Analytic Integration Connector for SAP BW examines the metadata of the selected InfoObject and extracts information to create the fact table. It also
extracts information to create other OLAP model elements such as dimensions, hierarchies, and attributes. Integration Services Console then displays the resulting OLAP model.

Editing the fact table is neither advised nor recommended. Contact Hyperion Technical Support.

You may want to aggregate data before using it in Integration Server. When you aggregate data in views, the DBMS does the aggregation dynamically. When you aggregate data in tables, the aggregation is done ahead of time which may improve performance.

**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 5 and Table 3).

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 5, 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.
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 6, 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 6  Fully Defined Child Column

<table>
<thead>
<tr>
<th>GEO_PARENT</th>
<th>GEO_CHILD</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;NULL&gt;</td>
<td>USA</td>
</tr>
<tr>
<td>USA</td>
<td>East</td>
</tr>
<tr>
<td>East</td>
<td>Maine</td>
</tr>
<tr>
<td>Maine</td>
<td>Bangor</td>
</tr>
<tr>
<td>Bangor</td>
<td>01010</td>
</tr>
</tbody>
</table>

**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>
<thead>
<tr>
<th>PRODID</th>
<th>MKTID</th>
<th>TIMEID</th>
<th>INITSALES</th>
<th>SALESTYPE</th>
<th>PRODID</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>2</td>
<td>01-01-2000</td>
<td>100.00</td>
<td>Sales</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>01-02-2000</td>
<td>10.00</td>
<td>Returns</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>01-03-2000</td>
<td>50.00</td>
<td>Subsequent</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>01-04-2000</td>
<td>20.00</td>
<td>Returns</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>01-01-2000</td>
<td>100.00</td>
<td>Sales</td>
<td>100</td>
</tr>
</tbody>
</table>

You want to create the Essbase outline in the hierarchy illustrated in Figure 4:

![Accounts Hierarchy Diagram]

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>
<thead>
<tr>
<th>PRODID</th>
<th>MKTID</th>
<th>TIMEID</th>
<th>INITSALES</th>
<th>RETURNS</th>
<th>SUBSEQUENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>2</td>
<td>01-01-2000</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>01-02-2000</td>
<td>0.00</td>
<td>10.00</td>
<td>0.00</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>01-03-2000</td>
<td>0.00</td>
<td>0.00</td>
<td>50.00</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>01-04-2000</td>
<td>0.00</td>
<td>20.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

The following example of Oracle SQL defines the view shown in Table 8:

```
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 9** Normalized Product Family Data

<table>
<thead>
<tr>
<th>FAMILYID</th>
<th>FAMILYDESC</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Colas</td>
</tr>
<tr>
<td>200</td>
<td>Root Beer</td>
</tr>
</tbody>
</table>

**Table 10** Normalized Product Data

<table>
<thead>
<tr>
<th>PRODID</th>
<th>FAMILYID</th>
</tr>
</thead>
<tbody>
<tr>
<td>100-10</td>
<td>100</td>
</tr>
<tr>
<td>100-20</td>
<td>100</td>
</tr>
<tr>
<td>100-30</td>
<td>100</td>
</tr>
</tbody>
</table>

**Table 11** Normalized Product Description Data

<table>
<thead>
<tr>
<th>PRODID</th>
<th>PRODDESC</th>
</tr>
</thead>
<tbody>
<tr>
<td>100-10</td>
<td>Cola</td>
</tr>
<tr>
<td>100-20</td>
<td>Diet Cola</td>
</tr>
<tr>
<td>100-30</td>
<td>Caffeine Free Cola</td>
</tr>
</tbody>
</table>

The following shows denormalized data in one table:

**Table 12** Denormalized Product Data

<table>
<thead>
<tr>
<th>FAMILYID</th>
<th>FAMILYDESC</th>
<th>PRODID</th>
<th>PRODDESC</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Colas</td>
<td>100-10</td>
<td>Cola</td>
</tr>
<tr>
<td>100</td>
<td>Colas</td>
<td>100-20</td>
<td>Diet Cola</td>
</tr>
</tbody>
</table>
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 32.
- Tables within each dimension branch. See “Selecting Tables for Dimensions” on page 33.
- Dimension tables joined in the fact table.

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

Optimizing Tables for Use with SAP BW

With SAP BW data warehouses, Integration Services enables you to optimize table access and increase system performance.
Indexes for Small Tables

If you have multiple joins across tables, system performance may degrade because each join requires a full scan through each table for data. Integration Services uses indexes to search on specific rows rather than on entire tables. This works well with small tables.

Push Down Operations

Integration Services uses several “push down” operations to speed access to tables:

Push Down Projections

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

Push Down Filters

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

Push Down Jobs

Integration Services places multi-table SQL statements in a single MDX statement with each table having its own MDX statement, giving more efficient joins.

Push Down Group By

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

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.

Note:

In SAP BW data warehouses, aggregates can be created but not indexes which can be used by Integration Services.

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 33.

● Define bitmapped indexes on numeric data you use to filter the database. For example, if you filter on sales values SELECT Product FROM ProdSales GROUP BY Product ORDER BY 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.

Note:

For information on transforming data in SAP BW data warehouses, see your SAP BW documentation.

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 21) without changing source data.

You can do some data transformations in OLAP models and metatoutlines. (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 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. In SAP BW data warehouses, measures are *key figures* and facts are *key figure values*; dimensions are *characteristics*. (See “SAP BW Data Warehouses” on page 6.)

The type of Essbase database you create determines which source tables to include in the fact table. See “Defining User Needs” on page 20.

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 13**  ORDERS and ORDERDETAILS Tables for the Fact Table

<table>
<thead>
<tr>
<th>Orders</th>
<th>ORDERID</th>
<th>TRANSDATE</th>
<th>SHIPID</th>
<th>EMPTYID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Order Details</td>
<td>ORDERID</td>
<td>PRODID</td>
<td>NUMUNITS</td>
<td>UTPRICE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.


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 28.
Selecting Tables for Dimensions

A *dimension* is a data category. Typical dimensions are Product, Geography, and Time. In SAP BW data warehouses, the components analogous to multidimensional dimensions are *characteristics*, and *dimension* denotes a group of related characteristics. See “SAP BW Data Warehouses” on page 6.

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 20. 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.

**Note:**

In SAP BW data warehouses, an alias is the text for a characteristic.

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.
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.

Note:

In SAP BW data warehouses, multidimensional dimensions are characteristics, and dimension denotes a group of related characteristics. See “SAP BW Data Warehouses” on page 6.

Preparing Data for Time Analysis

If your business requires time-related analysis, include a time dimension in the 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 14  TBC_MD Data in the SALESINVACT Table

<table>
<thead>
<tr>
<th>STATE</th>
<th>PRODCODE</th>
<th>TRANSDATE</th>
<th>SALES</th>
<th>COGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>IL</td>
<td>100-10-C001-S0002-P001</td>
<td>Jan 4 2000 12:00AM</td>
<td>672.00</td>
<td>217.00</td>
</tr>
<tr>
<td>STATE</td>
<td>PRODCODE</td>
<td>TRANSDATE</td>
<td>SALES</td>
<td>COGS</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>IL</td>
<td>100-10-C001-S0002-P001</td>
<td>Feb 4 2000 12:00AM</td>
<td>241.00</td>
<td>70.00</td>
</tr>
<tr>
<td>IL</td>
<td>100-10-C001-S0002-P001</td>
<td>Mar 3 2000 12:00AM</td>
<td>287.00</td>
<td>84.00</td>
</tr>
<tr>
<td>IL</td>
<td>100-10-C001-S0002-P001</td>
<td>Apr 7 2000 12:00AM</td>
<td>295.00</td>
<td>85.00</td>
</tr>
<tr>
<td>IL</td>
<td>100-10-C001-S0002-P001</td>
<td>May 3 2000 12:00AM</td>
<td>702.00</td>
<td>207.00</td>
</tr>
</tbody>
</table>

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 15  Columns Defining Time Periods

<table>
<thead>
<tr>
<th>PRODCODE</th>
<th>STATE</th>
<th>ACCOUNT</th>
<th>JAN</th>
<th>FEB</th>
<th>MAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>100-10-C001-S002-P001</td>
<td>AZ</td>
<td>Sales</td>
<td>672.00</td>
<td>241.00</td>
<td>287.00</td>
</tr>
<tr>
<td>100-10-C001-S002-P001</td>
<td>AZ</td>
<td>COGS</td>
<td>403.00</td>
<td>132.00</td>
<td>177.00</td>
</tr>
<tr>
<td>100-10-C001-S002-P001</td>
<td>CA</td>
<td>Sales</td>
<td>401.00</td>
<td>143.00</td>
<td>378.00</td>
</tr>
<tr>
<td>100-10-C001-S002-P001</td>
<td>CA</td>
<td>COGS</td>
<td>260.00</td>
<td>101.00</td>
<td>226.00</td>
</tr>
</tbody>
</table>

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 16. If data is stored with each month as an individual column, as shown in Table 15, 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 16  Time Periods Stored as Data Values

<table>
<thead>
<tr>
<th>PRODCODE</th>
<th>STATE</th>
<th>MONTH</th>
<th>SALES</th>
<th>COGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>100-10-C001-S002-P001</td>
<td>AZ</td>
<td>1</td>
<td>672.00</td>
<td>403.00</td>
</tr>
<tr>
<td>100-10-C001-S002-P001</td>
<td>AZ</td>
<td>2</td>
<td>241.00</td>
<td>132.00</td>
</tr>
<tr>
<td>100-10-C001-S002-P001</td>
<td>AZ</td>
<td>3</td>
<td>287.00</td>
<td>177.00</td>
</tr>
<tr>
<td>100-10-C001-S002-P001</td>
<td>CA</td>
<td>1</td>
<td>401.00</td>
<td>260.00</td>
</tr>
<tr>
<td>100-10-C001-S002-P001</td>
<td>CA</td>
<td>2</td>
<td>143.00</td>
<td>101.00</td>
</tr>
<tr>
<td>100-10-C001-S002-P001</td>
<td>CA</td>
<td>3</td>
<td>378.00</td>
<td>226.00</td>
</tr>
<tr>
<td>100-10-C001-S002-P001</td>
<td>AZ</td>
<td>1</td>
<td>672.00</td>
<td>403.00</td>
</tr>
</tbody>
</table>

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 25).

**Accounts Dimensions and SAP BW**

When you import an SAP BW model to create an OLAP model, Analytic Integration Connector for SAP BW examines the InfoObject you selected for accounts information. The accounts information is collected and presented in the OLAP model as one coherent accounts dimension containing entities reflecting the structure of the SAP BW database.

If the SAP BW data source does not contain accounts elements, Analytic Integration Connector for SAP BW cannot create an accounts dimension.
It is neither advised nor recommended to edit an accounts dimension. Contact Hyperion Technical Support.

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

**Time Dimensions and SAP BW**
In SAP BW data sources, time entities are generally stored as separate dimensions. When you import an SAP BW model to create an OLAP model, Analytic Integration Connector for SAP BW examines the InfoObject for time information. Various SAP BW time dimensions are collected and presented in the OLAP model as one coherent time dimension.

If the SAP BW data source does not contain time elements, Analytic Integration Connector for SAP BW cannot create an accounts dimension.

It is neither advised nor recommended to create or edit a time dimension. Contact Hyperion Technical Support.

**Slowly Changing Dimensions in SAP BW**
Some metadata and data can be very stable over long periods then experience a change in structure; for example, Seattle is in the Northwest Region for several years before being moved to the Pacific Coast Region. In SAP BW data warehouses, this is a slowly changing dimension.

SAP BW contains a key date, sometimes referred to as the active date range, but this key date is not supported in Integration Services. Therefore, when you access data from a slowly changing dimension in SAP BW, you always see the latest value.

**Attribute-Enabled Columns in SAP BW**
When you import an SAP BW model to create an OLAP model, Hyperion® System™ 9 BI+™ Analytic Integration Connector for SAP BW™ examines the InfoObject for attribute information and marks appropriate columns as attributes.

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

**Consolidation Properties**
The following table defines the values for consolidation properties Integration Services recognizes when loading these properties from database columns.
Table 17  Consolidation Properties Settings

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition—Adds member to result of previous calculations performed on other members (default operator)</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction—Multiplies member by -1 and adds result to sum of previous calculations performed on other members</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication—Multiplies member by result of previous calculations performed on other members</td>
</tr>
<tr>
<td>/</td>
<td>Division—Divides member by result of previous calculations performed on other members</td>
</tr>
<tr>
<td>%</td>
<td>Percent—Divides member by sum of previous calculations performed on other members and multiplies result by 100</td>
</tr>
<tr>
<td>~</td>
<td>Ignore—Does not use member in consolidation to its parent</td>
</tr>
</tbody>
</table>

Two Pass Calculation Properties

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

Table 18  Two Pass Calculation Properties Settings

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Blank)</td>
<td>Two-pass calculations not used on member</td>
</tr>
<tr>
<td>T</td>
<td>Two Pass Calculation—Two-pass calculation used on member</td>
</tr>
</tbody>
</table>

Data Storage Properties

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

Table 19  Data Storage Properties Settings

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Store Data—Member stores data (default storage property)</td>
</tr>
<tr>
<td>V</td>
<td>Dynamic Calc and Store—Data associated with member is not calculated until requested by user; calculated data is then stored</td>
</tr>
<tr>
<td>X</td>
<td>Dynamic Calc—Data associated with member is not calculated until requested by user; calculated data is discarded after the request is completed</td>
</tr>
<tr>
<td>N</td>
<td>Never Share—Data associated with member is duplicated with its parent or child only if an implied shared relationship exists</td>
</tr>
<tr>
<td>O</td>
<td>Label Only—Although member has no data associated with it, it can still display a value</td>
</tr>
</tbody>
</table>
## Time Balance Properties

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Blank)</td>
<td>None—If time balance is set as none, Essbase rolls up parents in time dimension in standard method: value of parent is based on formulas and consolidation properties of children (default setting)</td>
</tr>
<tr>
<td>F</td>
<td>First—Set time balance as first to have parent value represent value of first member in branch (often at beginning of time period)</td>
</tr>
<tr>
<td>L</td>
<td>Last—Set time balance as last to have parent value represent value of last member in branch (often at end of time period)</td>
</tr>
<tr>
<td>A</td>
<td>Average—Set time balance as average to have parent value represent average of children values</td>
</tr>
</tbody>
</table>

## Skip Properties

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Blank)</td>
<td>None—Does not skip data when calculating parent value (default setting). If Oracle's Hyperion® Essbase® – System 9 encounters #MISSING data when calculating an average, it does not divide by total number of members; it divides by number of members with actual values. Therefore, setting the skip property to none or #MISSING does not affect average but does affect first and last.</td>
</tr>
<tr>
<td>M</td>
<td>Missing—Skips #MISSING data when calculating parent value</td>
</tr>
<tr>
<td>Z</td>
<td>Zeros—Skips data equaling zero when calculating parent value</td>
</tr>
<tr>
<td>B</td>
<td>Both Missing and Zeros—Skips both #MISSING data and data equaling zero when calculating the parent value</td>
</tr>
</tbody>
</table>

## Variance Reporting Properties

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

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

Currency Conversion Properties

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Blank)</td>
<td>None—dimension has no relationship to currency conversion (default setting)</td>
</tr>
<tr>
<td></td>
<td>No Conversion—dimension is not converted because it is not a currency value; it can be a value such as a quantity or percentage</td>
</tr>
</tbody>
</table>

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 (Sybase, SQL Server) or synonyms for the tables (DB2, Oracle). See the *Essbase Integration Services Installation Guide*.

Mapping to DB2 Cube Views

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

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

Table 24 lists how Integration Services metadata objects map to DB2 Cube Views metadata objects.

<table>
<thead>
<tr>
<th>DB2 Cube Views metadata object</th>
<th>Integration Services metadata object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cube model</td>
<td>Model</td>
</tr>
<tr>
<td>Facts</td>
<td>Fact</td>
</tr>
<tr>
<td>Measure</td>
<td>Numeric member of fact</td>
</tr>
</tbody>
</table>
Text Files as Data Sources

In Oracle's 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 22). 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 28.)
Index

A
accessing tables in OLAP Metadata Catalog, 40
accounts dimension
  sample hierarchy, 26
  use in Essbase outline, 34
adding columns to tables, 28
aggregation
  choosing level of, 20
  defined, 8
aliases
  concatenating, 33
  creating for dimension or member names, 33
  multiple aliases, using, 33
  pass-through transformations and, 33
  recursive tables and, 24
drill down and, 8
  example, 8
  members, 6
consolidation properties, 37
creating
  Essbase databases, workflow, 11
  hierarchies, 24
  indexes, 29
currency conversion properties, 40
customer comparisons, 5

B
balanced hierarchies, 14, 15
batch processing, Integration Services Shell, 11
bitmapped indexes, defined, 29

C
characteristic values, 6
characteristics, 6
cleansing data, 22
columns
  adding to tables, 28
  availability, aliases, 33
  fully-defined in recursive tables, 24
  indexes, 30
  supporting member and measure properties, 28
  time-related, defining in source data, 34
  transposing with rows, 25
concatenation and aliases, 33
Console. See Integration Services Console
  consolidation
  defined, 8
data
  cleansing, 22
  combining multiple sources, 21
denormalizing, 27
drill down, 8
e nsuring consistency, 22
  pivoting to change view, 8
  preparing for time analysis, 34
  source preparation, 19
  transforming in data source, 30
types of, 20
data sources
  build levels, 24
defining, 20
  parent-child, 24
  spreadsheet files, 8
  SQL, 8
text files, 8
  workflow for transferring, 11
data storage properties, 38
data types and cleansing, 22
data values.  See measures
data warehouse, diagram, 22
data warehouses
  source of data, 8
terminology, 6
databases. See data sources
dates, formatting in source data, 35
DB2, tables displayed in, 31
denormalizing data, 27
deploying hierarchies, 16
deploying hierarchies, 16
detail data
  choosing level of, 20
  retrieving, 8
dimension branches, defined, 33
dimension names, creating aliases for, 33
dimension tables
  defined, 12, 33
  OLAP models, 12
dimensions
  choosing, 20
  choosing tables for, 33
  compared with characteristics, 6
  creating in a metaoutline, 12
  defined, 6, 11, 13, 19, 33
  examples, 6
  frequency of change, 6
  members, 6
  SAP BW, 20
  Scenario, 12
  use in consolidation, 6
  using in metaoutlines, 12
drill down, 8
drill up, 8
drill-through report members, 14
drill-through reports, 14
duplicate members, 17

E
Essbase
  accounts dimension, 34
  databases, workflow, 11
  designing a database for user needs, 20
  Essbase Server, defined, 5
  hierarchies, creating, 24
  members, compared to metaoutline member levels, 13
  users, 8
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, 8

F
fact table
  choosing tables for, 32
  compared with dimensions, 20
  defined, 12, 32
  measures and, 32
  role in OLAP model, 12
  SAP BW, 12
facts, defined, 11, 19
filters, 30
flat files
  as a data source, 20, 41
  combining with SQL data sources, 21
forecasting, 8
fully defined columns, defined, 24

G
general ledger data, 35

H
hierarchies
  balanced, 14, 15
  building from recursive tables, 24
  consolidation and, 6
  creating, 24
  data sources for building to a specific level, 24
  deploying, 16
  ragged, 15
  recursive, 16
  standard, 16
  types, 14
  unbalanced, 15
  using, 14
Hyperion Connect, 6
Hyperion Essbase. See Essbase
Hyperion Integration Server. See Integration Server
Hyperion Integration Server Desktop. See Integration Services Console
I
indenting, 6
indexes
   SAP BW, 29
   working with, 29, 30
indexes, working with, 29, 30
Integration Server, defined, 10
Integration Services
   about, 9
   data warehouse, 21
   defined, 5
   Integration Services Console, defined, 10
   Integration Services Shell, defined, 11
   OLAP Metadata Catalog, defined, 11
   product family, 10, 11
   workflow, 9, 11
Integration Services Console, defined, 10
Integration Services Shell
   batch processing, 11
   defined, 11
   integrity constraints, cleansing data, 22
   intersections, defined, 6
J
joins
   joining tables in your RDBMS, 28
   reducing through denormalization, 27
   unions, removing, 25
K
key figures, 6
keys, columns, transformations and, 31
L
legacy data, preparing, 35
levels, data sources for building, 24
logical model, description, 11
M
measure filters. See filters
measures
   bitmapped indexes and, 30
   compared with key figures, 6
   data, associating with time periods, 34
   defined, 13, 20
   fact table and, 32
   SAP BW, 13, 20
   using with columns, 28
member filters. See filters
member levels, defined, 13
member names, creating aliases, 33
members
   compared with characteristic values, 6
   consolidation, 6
   defined, 6
   dimensions, 6
   examples, 6
   OLAP models, 6
   using with columns, 28
metadata
   accessing OLAP Metadata Catalog, 40
   defined, 10
   metadata repository. See OLAP Metadata Catalog
   metadata, OLAP Metadata Catalog storage, 10
metaoutlines
   advantages of, 12
   components, 13
   creating dimensions, 12
   models. See OLAP models
   multidimensional databases
      defined, 6
      uses, 8
      workflow, 9, 11
   multiple alias tables, using, 33
N
normalized data, denormalizing, 27
numeric data, indexes and, 30
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, 40
   configuring for ODBC, 11
   defined, 10, 11
   OLAP metaoutline. See metaoutlines
   OLAP models
choosing which tables are displayed, 31
components, 11
creating, 10
defined, 12
dimension tables, 12
dimensions, 6
fact table, 12
graphic illustration, 12
members, 6
uses, 5
OLAP Server. See Essbase Server
Online Analytical Processing (OLAP), defined, 5

P
parent-child data source, 24
pass-through transformations, aliases and, 33
performance
  bitmapped indexes and, 29
  increasing through denormalization, 27
permissions, required in the SQL data source, 31
perspective, changing data on a spreadsheet, 8
pivoting, using to change data view, 8
privileges. See permissions
product code, example, 31
product comparisons, 5
properties
  consolidation, 37
  currency conversion, 40
data storage, 38
skip, 39
time balance, 39
two pass calculation, 38
variance reporting, 39
push down operations with SAP BW, 29

R
ragged hierarchies, 15
read-only access, SQL data sources and, 31
recursive hierarchies, 16
recursive tables
  aliases and, 24
  creating Essbase hierarchies from recursive tables, 24
defined, 24
defining for alias or UDAs, 24
fully-defined columns for, 24
rules for, 24
redundant data, denormalizing, 27
relational databases. See data sources
repository. See OLAP Metadata Catalog
roll-ups, defined, 8
rows, transposing with columns, 25

S
sample, hierarchy, accounts dimension, 26
SAP BW
  aliases, 33
data source, 21
dimensions, 20
fact table, 12, 32
indexes, 29
Integration Server, 10
measures, 13, 20
performing tasks, restrictions, 19
push down operations, 29
staging area, 21
star schema, 19
support of Integration Services, 6
tables, 28, 33
terminology, 6
transforming data, 30
scenarios, 8
security
  SQL data sources, 31
  views and, 22
SELECT
  permission, 31
  statements, combining, 25
skip properties, 39
snapshot, staging area, 21
source data
  denormalizing, 27
  transforming, 30
sources. See data sources
spreadsheet files, data sources, 8
SQL data sources
  combining, 21
  combining flat files, 21
  consolidating multiple, 21
  read-only access to, 31
  required permission for, 31
  security, 31
  staging area, 21
SQL Drill-Through. See drill-through reports staging area
  advantages of, 21
deciding whether to create, 21
defined, 21
diagram, 22
standard hierarchies, 16
star schemas. See OLAP models strategic planning, 8
summary source data, associating with time periods, 34

tables
  accessing tables in OLAP Metadata Catalog, 40
  choosing which tables are displayed in OLAP models, 31
  creating to denormalize data, 27
deciding whether to create new tables, 22
denormalizing, 27
  joining, 28
  reasons to create, 23
recursive
  aliases and, 24
defined, 24
  fully-defined columns for, 24
  rules for, 24
  UDAs or aliases, 24
SAP BW optimizing, 28
text file as a data source, 8, 41
time
  analysis, preparing data for, 34
  associating with measures data, 34
  time-related columns, defining in source data, 34
time balance properties, 39
time dimension
  defined, 34
  use in Essbase outline, 34
transformations
  defining in SQL data sources, 30
  example in data source, 31
  indexes and, 30
  key columns and, 31
  transposing columns and rows, 25
  trends, 5
two pass calculation properties, 38

U
UDAs, creating in recursive tables, 24
unbalanced hierarchies, 15
unions, removing, 25
user needs, defining, 20
user privileges. See permissions
user-defined dimensions, defined, 12
user-defined tables, 22
users, Essbase, 8
V
values. See measures
variables, defined, 20
variance reporting properties, 39
views
  deciding whether to create new views, 22
  reasons to create, 23
W
what if scenarios, example, 8
WHERE clause. See filters
workflow
  Integration Services, 9
  Integration Services product family, 11