
PeopleSoft Enterprise EPM 9.1 Campus Solutions Warehouse PeopleBook

April 2010

Copyright © 1999, 2010, Oracle and/or its affiliates. All rights reserved.

Trademark Notice

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

License Restrictions Warranty/Consequential Damages Disclaimer

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.

Warranty Disclaimer

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.

Restricted Rights Notice

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.

Hazardous Applications Notice

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.

Third Party Content, Products, and Services Disclaimer

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

Preface

PeopleSoft Enterprise Campus Solutions Warehouse Preface	ix
Oracle's PeopleSoft Products	ix
PeopleSoft Application Fundamentals	ix
PeopleBooks and the Online PeopleSoft Library	ix

Chapter 1

Getting Started with PeopleSoft Enterprise Campus Solutions Warehouse	1
PeopleSoft Enterprise Campus Solutions Warehouse Overview	1
Other Sources of Information	1

Chapter 2

Understanding the EPM Warehouses	3
Overview of PeopleSoft EPM Warehouses	3
CRM Warehouse	3
Campus Solutions Warehouse	4
FMS Warehouse	4
Financials Warehouse for Public Sector and Higher Education	5
HCM Warehouse	6
SCM Warehouse	6
Components of PeopleSoft EPM Warehouses	7
Extract Transform and Load (ETL) Component	7
Infrastructure Tables and Tools	7
Security Tables	8
Staging Tables	8
Multidimensional Warehouse Fact Tables	8
Multidimensional Warehouse Dimension Tables	9
Data Models	12
Measures	12
EPM Architecture and Data Flow	13
Operational Warehouse - Staging (OWS)	14
Operational Warehouse - Enriched (OWE)	14
Multidimensional Warehouse (MDW)	15

Reporting on the EPM Warehouses	15
Chapter 3	
Understanding the Campus Solutions Warehouse	17
Campus Solutions Warehouse Overview	17
Campus Solutions Warehouse Structure	18
Admissions and Recruiting Data Mart	19
Admissions and Recruiting Data Mart Delivered Fact and Dimension Tables	19
Campus Community Data Mart	28
Campus Community Data Mart Delivered Fact and Dimension Tables	28
Student Records Data Mart	34
Student Records Data Mart Delivered Fact and Dimension Tables	34
Student Financial Services Data Mart	43
Student Financial Services Data Mart Delivered Fact and Dimension Tables	44
Common Dimensions in the Campus Solutions Warehouse	53
Shared Dimensions	55
Chapter 4	
Running Campus Solutions Warehouse Implementation Jobs	61
Prerequisites	61
Running Campus Solutions Warehouse Implementation Jobs	62
Running Campus Solutions Warehouse - OWS Jobs	62
Running Global Dimension Jobs for Campus Solutions Warehouse	63
Running Local Dimension Jobs for Campus Solutions Warehouse	64
Running Campus Solutions Warehouse SKU Jobs	65
Chapter 5	
Configuring Slowly Changing Dimensions	69
Understanding Slowly Changing Dimensions	69
Type 1 Slowly Changing Dimensions	69
Type 2 Slowly Changing Dimensions	70
Type 3 Slowly Changing Dimensions	71
Understanding Slowly Changing Dimensions in EPM	72
Valid Date Range Subrecord	72
Design Differences Between Type 1 and Type 2 Slowly Changing Dimension Jobs	73
Fact Table Jobs and Slowly Changing Dimensions	75
Converting Type 1 Slowly Changing Dimension Jobs to Type 2	76
Overview	76

Method 1: Converting a Type 1 Slowly Changing Dimension Job Using the Effective Date and Effective Sequence	76
Method 2: Converting a Type 1 Slowly Changing Dimension Job Without Using the Effective Date . 88	
Converting a Hash File Lookup to a Dynamic DRS Lookup in the Related Fact Table Job	89
Chapter 6	
Implementing Audit Jobs	91
Understanding Audit Jobs	91
Understanding Audit Job Implementation	92
Creating Audit Triggers and Running Trigger Scripts	94
Pages Used to Create Audit Triggers and Run Trigger Scripts	94
Creating Audit Triggers	94
Creating and Running the Trigger Script	96
Chapter 7	
Managing Source System Deletes and Archiving	97
Understanding Source System Deletes	97
Identifying Source Record Deletes with CRC Staging Jobs	98
Identifying Source Record Deletes with Standard Staging Jobs	100
Understanding Source System Archives	101
Enabling the Source-Delete Diagnostic Feature	103
Implementing the Source-Delete Diagnostic Feature in Staging Jobs	103
Adjusting the Source-Delete Diagnostic Option after Implementation	104
Adjusting the Source-Delete Diagnostic Option for CRC Staging Jobs	105
Adjusting the Source-Delete Diagnostic Option for Standard Staging Jobs	108
Implementing the Source-Archiving Diagnostic Feature	108
Chapter 8	
Implementing Currency Conversion for Multiple Currencies	111
Understanding Currency Conversion	111
Understanding Currency Conversion Methodology	114
Understanding Currency Conversion Rules	119
Setting Up Currency Conversion	125
Pages Used to Set Up the Schema Definition and Currency Conversion Rules	125
Setting Up the Schema Definition	126
Setting Up Currency Conversion Rules	127
Setting up the Conversion Schema Rule	128

Running the ETL Currency Conversion Process 129

Chapter 9

Setting Up Multilanguage Processing and Running the Language Swap Utility 131

Understanding Multilanguage Processing 131
 Understanding the Language Swap Utility 133
 Running the Language Swap Jobs 137

Chapter 10

Processing Trees and Recursive Hierarchies 139

Understanding Tree and Recursive Hierarchy Processing 139
 Trees and Recursive Hierarchies 139
 OWE Tree Flattener Versus MDW Tree Denormalizer 140
 Hierarchies Supported by the Tree and Recursive Hierarchy Process 141
 Denormalized Tree Result Balancing 143
 Skip Levels 144
 Tree and Recursive Hierarchy Source Tables 146
 Multilanguage Support for Relationship and Hierarchy Tables 147
 Understanding Tree and Recursive Hierarchy Process Results 147
 Tree Flattener and Tree Denormalizer Output Tables 147
 Tree Flattener and Denormalizer Results 157
 Setting Up Parameters for Tree and Recursive Hierarchy Processing 163
 Defining Parameters for the Tree and Recursive Hierarchy Process 164
 Pages Used to Run the Tree and Recursive Hierarchy Process 164
 Defining the Target and Language Tables for Tree Flattening 165
 Defining the Target and Language Tables for Tree Denormalizing 165
 Creating the Hierarchy Group Definition 166
 Running the Tree and Recursive Hierarchy ETL Process 169
 Running Hash File Hierarchy Jobs 169
 Running OWS Hierarchy Jobs 170
 Running Hierarchy Utility Jobs 170

Chapter 11

Extending the Multidimensional Warehouse Data Model 173

Considerations for Modifying an EPM Warehouse 173
 Adding a Fact or Dimension Table to the Multidimensional Warehouse Data Model 174
 Extending a Fact Table in the Multidimensional Warehouse Data Model 179
 Adding a New Measure to a Fact Table 179

Adding a New Surrogate Key to a Fact Table	183
Extending a Dimension Table in the Multidimensional Warehouse Data Model	186

Appendix A

Frequently Asked Questions for Campus Solutions Warehouse Implementation	197
Implementation Scenarios	197
ETL Load Sequence	198
Environmental Parameters Configuration	199

Appendix B

Using the PeopleSoft EPM Lineage Spreadsheets	201
Understanding the EPM Lineage Spreadsheets	201
Viewing Lineage Information	204
Finding Lineage Information for a Server Job	204
Identifying the List of Jobs to be Run for a Data Mart	208
Generating Lineage Information for a Job	209

Index	211
--------------------	------------

PeopleSoft Enterprise Campus Solutions Warehouse Preface

This preface discusses:

- Oracle's PeopleSoft Products.
- PeopleSoft Application Fundamentals.

Oracle's PeopleSoft Products

This PeopleBook refers to these products:

- PeopleSoft Enterprise Customer Relations Management (CRM) Warehouse.
- PeopleSoft Enterprise Financial Management Solutions (FMS) Warehouse.
- PeopleSoft Enterprise Financials Warehouse for Public Sector and Higher Education.
- PeopleSoft Enterprise Human Capital Management (HCM) Warehouse.
- PeopleSoft Enterprise Supply Chain Management (SCM) Warehouse.

PeopleSoft Application Fundamentals

Additional, essential information describing the setup and design of your system appears in a companion volume of documentation called *PeopleSoft Enterprise Performance Management Fundamentals PeopleBook*.

PeopleBooks and the Online PeopleSoft Library

A companion PeopleBook called PeopleBooks and the Online PeopleSoft Library contains general information, including:

- Understanding the PeopleSoft online library and related documentation.
- How to send PeopleSoft documentation comments and suggestions to Oracle.
- How to access hosted PeopleBooks, downloadable HTML PeopleBooks, and downloadable PDF PeopleBooks as well as documentation updates.
- Understanding PeopleBook structure.
- Typographical conventions and visual cues used in PeopleBooks.

- ISO country codes and currency codes.
- PeopleBooks that are common across multiple applications.
- Common elements used in PeopleBooks.
- Navigating the PeopleBooks interface and searching the PeopleSoft online library.
- Displaying and printing screen shots and graphics in PeopleBooks.
- How to manage the PeopleSoft online library including full-text searching and configuring a reverse proxy server.
- Understanding documentation integration and how to integrate customized documentation into the library.
- Glossary of useful PeopleSoft terms that are used in PeopleBooks.

You can find this companion PeopleBook in your PeopleSoft online library.

Chapter 1

Getting Started with PeopleSoft Enterprise Campus Solutions Warehouse

This chapter provides an overview of PeopleSoft Enterprise Campus Solutions Warehouse and discusses other sources of information.

PeopleSoft Enterprise Campus Solutions Warehouse Overview

PeopleSoft Enterprise Campus Solutions Warehouse serves both as a repository of information and as the foundation for business intelligence reporting. It provides higher education institutions with an integrated suite of reporting and analytic solutions and provides you with a framework to develop custom reporting, analytics, and dashboard components for a variety of end-users. The Campus Solutions Warehouse sources data from the PeopleSoft Enterprise Campus Solutions transaction system, and can be extended to draw data from other PeopleSoft and legacy systems, to stage, store and enrich information for reporting.

Other Sources of Information

In the planning phase of your implementation, take advantage of all PeopleSoft sources of information, including the installation guides and troubleshooting information. A complete list of these resources appears in the preface in *PeopleSoft Enterprise Performance Management Fundamentals PeopleBook*, which also provides overview information about EPM, the Multidimensional Warehouse, and required setup tasks for the EPM warehouses.

See Also

Enterprise PeopleTools PeopleBook: PeopleSoft Setup Manager

Chapter 2

Understanding the EPM Warehouses

This chapter provides an overview of the EPM Warehouses.

Overview of PeopleSoft EPM Warehouses

PeopleSoft delivers six EPM warehouses that provide you with the tools and technology to manage your organization's information that is used for reporting and analysis. Each warehouse is divided into multiple subject areas, or data marts. Each data mart is aligned with a business process, which enables you to answer strategic questions essential to your organization's bottom line.

The following sections describe these PeopleSoft EPM Warehouses:

- PeopleSoft Enterprise Customer Relations Management (CRM) Warehouse.
- PeopleSoft Enterprise Campus Solutions (CS) Warehouse
- PeopleSoft Enterprise Financial Management Solutions (FMS) Warehouse.
- PeopleSoft Enterprise Financials Warehouse for Public Sector and Higher Education.
- PeopleSoft Enterprise Human Capital Management (HCM) Warehouse.
- PeopleSoft Enterprise Supply Chain Management (SCM) Warehouse.

For detailed information about EPM, the Multidimensional Warehouse, and required setup tasks for the EPM warehouses, please refer to the *PeopleSoft Enterprise Performance Management Fundamentals PeopleBook*.

See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Oracle's PeopleSoft Enterprise Performance Management Fundamentals 9.1 Preface."

CRM Warehouse

The CRM warehouse enables you to create reports related to these business processes.

- Marketing
- Support
- Sales

The CRM Warehouse consists of these data marts:

- Sales

- Service
- Marketing
- Customer Segment

See *PeopleSoft Enterprise Customer Relationship Management Warehouse 9.1 PeopleBook*, "PeopleSoft Enterprise Customer Relationship Management Warehouse Preface."

Campus Solutions Warehouse

The Campus Solutions warehouse enable you to create reports related to these business processes:

- Student Recruiting
- Student Admission Application
- Application Evaluations
- Student Responses
- External Test Scores
- External Education
- Class Meeting Patterns
- Course Catalog, Class Scheduling, Instructor Workload
- Program Activation & Management
- Student Career Term Record Management
- Enrollment
- Award
- Student Payments

The Campus Solutions warehouse consists of the following data marts:

- Admissions and Recruiting
- Campus Community
- Student Records
- Student Financials

See [Chapter 3, "Understanding the Campus Solutions Warehouse," page 17.](#)

FMS Warehouse

The Financial Management Solutions Warehouse enables you to create reports related to these business processes:

- Order Fulfillment
- Procurement
- Financial Control and Reporting
- Commitment Control
- Grant Analytics
- Project Management
- Asset Lifecycle Management

The Financial Management Solutions Warehouse consists of these data marts:

- Receivables
- Payables
- General Ledger and Profitability
- Enterprise Services Automation (ESA)

See *PeopleSoft Enterprise Financial Management Solutions Warehouse 9.1 PeopleBook*, "PeopleSoft Enterprise Financial Management Solutions Warehouse Preface."

Financials Warehouse for Public Sector and Higher Education

The Financials Warehouse for Public Sector and Higher Education enables you to create reports related to these business processes:

- Procurement
- Spend
- Order Fulfillment
- Financial Control and Reporting
- Commitment Control
- Grant Analytics
- Project Management
- Asset Lifecycle Management

The Financials Warehouse for Public Sector and Higher Education consists of these data marts:

- Procurement (from the Supply Chain Management Warehouse)
- Spend (from the Supply Chain Management Warehouse)
- Receivables (from the Financial Solutions Management Warehouse)
- Payables (from the Financial Solutions Management Warehouse)

- General Ledger and Profitability (from the Financial Solutions Management Warehouse)
- Enterprise Services Automation (from the Financial Solutions Management Warehouse)

See *PeopleSoft Enterprise Financials Warehouse for Public Sector and Higher Education 9.1 PeopleBook*, "PeopleSoft Enterprise Financials Warehouse for Public Sector and Higher Education Preface."

HCM Warehouse

The HCM Warehouse enables you to create reports related to these business processes:

- Deployment
- Reward
- Development
- Recruiting

The HCM Warehouse consists of these data marts:

- Workforce Profile
- Compensation
- Learning and Development
- Recruiting

See *PeopleSoft Enterprise Human Capital Management Warehouse 9.1 PeopleBook*, "PeopleSoft Enterprise Human Capital Management Warehouse Preface."

SCM Warehouse

The Supply Chain Warehouse enables you to create reports related to these business processes:

- Order Fulfillment
- Procurement
- Production

The Supply Chain Warehouse consists of these data marts:

- Fulfillment and Billing
- Procurement
- Spend
- Inventory
- Manufacturing
- Supply Chain Planning

See *PeopleSoft Enterprise Supply Chain Management Warehouse 9.1 PeopleBook*, "PeopleSoft Enterprise Supply Chain Management Warehouse Preface."

Components of PeopleSoft EPM Warehouses

PeopleSoft delivers the following content with an EPM warehouse:

- Extract Transform and Load (ETL) component
- Infrastructure tables and tools
- Security tables
- Staging tables
- Multidimensional Warehouse tables
- Data Models
- Measures

Each bullet is discussed in more detail below.

Extract Transform and Load (ETL) Component

PeopleSoft EPM warehouses are delivered with the IBM WebSphere DataStage ETL tool and prepackaged ETL jobs. Together they enable you to extract data from PeopleSoft source transaction systems, integrate your data into a single database, and populate prepackaged data models which optimize your data for analysis and reporting.

There are also several ETL objects that support the ETL process, such as routines, environment parameters, and hashed files.

See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Preparing to Load Source Data Into EPM."

Infrastructure Tables and Tools

PeopleSoft EPM warehouses are delivered with infrastructure tables and tools, which serve as the underlying framework that supports the EPM Warehouses. Some examples of core infrastructure tables include the Currency Code (CURRENCY_CD_TB) table, which enables you to manage financial information in multiple currencies, and the Unit of Measure (PS_UNITS_TBL) table, which determine how specific resources are quantified.

Some examples of infrastructure tools provided by PeopleSoft include the Country and State Information component and the Business Unit Wizard, which automates the steps required to set up warehouse business units and set IDs

See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Implementing PeopleSoft Enterprise Performance Management," EPM Core Infrastructure and ETL Setup Tasks.

Security Tables

EPM security controls access to specific data within the EPM database and enables you to grant user-access to specific rows, columns, fields, or dimensions in the multidimensional warehouse. An example of the security tables delivered with an EPM warehouse is the Security Join Table, which stores the security profiles for users and the corresponding dimension values for which they have access.

See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Setting Up EPM Security."

Staging Tables

The Operational Warehouse - Staging tables act as an entry-point for your PeopleSoft source transaction data into EPM, and provide a platform to offload, consolidate, and stage your source transaction data in preparation for migration to prepackaged data models.

See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Understanding PeopleSoft Enterprise Performance Management," Operational Warehouse - Staging (OWS).

Multidimensional Warehouse Fact Tables

In an EPM warehouse, fact tables typically consist of numerical values, such as quantity, sales, and revenue, that relate to elements of your business. Fact tables help to quantify a organization's activities. In addition, fact tables usually contain an additive business performance measurement. That is, you can usually perform arithmetic functions on facts. EPM multidimensional fact tables contain numeric performance measurement information that is used in multidimensional reports that categorize your business.

Multidimensional warehouse fact tables can contain either transactional data or snapshot data:

- *Transactional data:* A transaction-dated fact source stores data by tracking individual events and when they occurred. To select the data for a particular date range, you retrieve all rows of data that have transaction dates between the start and end date in the desired range. For example, assume that you are measuring the number of units sold and you track the information using a transaction-dated structure. A row of data exists for each time a unit is sold, and each row has a date, or timestamp. To measure how many units sold in a week, you add all of the transactions—that is, the number of units sold—each day in that week.

In some situations, the application adds these events together over time to calculate an aggregated value.

- *Snapshot data:* An as of dated fact source stores the data based as a snapshot of the data at a given point in time. This snapshot often represents events across multiple time periods. It reduces the amount of data stored on a system, because each individual transaction is not stored. For example, to track organization head count by month, you can determine how many employees you have on the last day of every month. You store that information instead of storing every new hire transaction and attempting to aggregate each one to the month.

Because this information is typically aggregated, this type of data is usually not additive across multiple as of dated snapshots. To aggregate this type of data, you typically use the last snapshot taken for the specific time period that you want to aggregate.

In some EPM warehouses there are *factless fact tables*, a fact table that does not have an amount field that you sum to derive the value that you want. Instead, it allows you to do counts based on the key relationships. For example, a question such as "How many employees participate in the 401(K) program?" could likely be answered by querying a factless fact table. Factless fact tables are not empty, rather, they are another type of fact table commonly used in data modeling.

Note. MDW fact tables use the following naming convention: F_*[table name]*.

See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Understanding PeopleSoft Enterprise Performance Management," MDW Fact Tables.

Multidimensional Warehouse Dimension Tables

In an EPM warehouse, dimension tables are sets of related attributes that you use to group or constrain fact-based information when reporting. Dimension tables are descriptive, usually text (in character data type), non-additive (that is, they cannot be used for arithmetic computations), and often hierarchical. In terms of data analysis, dimensions can be thought of as criteria, such as time, product, and location, used to locate a particular piece of data.

For example, in higher education a set of dimensions could be Student, Academic Career, Instructor, and Courses. The Career dimension might include Career, Term, and session attributes. Business intelligence reporting typically makes use of dimension values to filter criteria. For example, the department head of the School of Engineering might filter the data so that a report only displays information relating to that specific school. Dimension table data can originate from a PeopleSoft source system or a flat file.

Note. MDW dimension tables use the following naming convention: D_*[table name]*.

See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Understanding PeopleSoft Enterprise Performance Management," MDW Dimension Tables.

Shared and Global Dimensions

Certain dimensions, such as Account, Customer, Department, Item dimensions, or Person are used across all EPM warehouses. Conformity of structure in these dimensions is essential to provide a consistent view of data and to easily integrate business measurements between functional warehouses. Therefore, these dimensions are identical in structure and content across all EPM warehouses.

Commonly Used Dimensions

The following table describes dimension tables that are commonly used across EPM warehouses:

Commonly Used Dimension	Description
Business Unit Dimension	<p>Business units are generally defined as distinct operational or organizational entities that maintain their own sets of books or transactional data. You can associate one source system with various types of business units, such as a general ledger business unit, an inventory business unit, and a manufacturing business unit. To facilitate EPM application and EPM foundation processing, a performance business unit (PFBU) is associated with each source business unit. PFBU is used for analytical and reporting purposes and has no equivalent in the source system. Each business unit must belong to one and only one PFBU. All business units that are members of the same PFBU must have the same fiscal calendar and default currency.</p> <p>A business unit can be associated with one or more business functions, as defined by its Business Unit Type attribute. Examples of Business Unit Type are Inventory business unit and General Ledger business unit. The multifunctional business unit can be associated with more than one business function. For example, business units with either Inventory Business Unit Type or Multifunctional Business Unit Type can be associated with the Inventory business function.</p> <p>You can relate one or more business units to a general ledger business unit. If a general ledger business unit has one or more business units associated to it, in the MDW that general ledger business unit is captured as a composite business unit, in addition to being a regular business unit in the Business Unit dimension table.</p> <p>Note. Business units that come from different source systems are different business units, even if they have the same name and the same BUSINESS_UNIT value.</p>

Commonly Used Dimension	Description
Calendar Dimension	<p>The Calendar dimension stores date-related attributes that are associated with a measure on a specific date. The Calendar dimension has a granularity of one day. In the MDW, the Calendar dimension accommodates storage of one regular, or Gregorian, calendar, plus any number of standard or custom calendars, such as fiscal, manufacturing, and sales calendars.</p> <p>In addition to having a granularity of <i>day</i>, the Gregorian calendar also provides hierarchies of <i>week</i>, <i>month</i>, <i>quarter</i>, and <i>year</i>. Because the application cannot consolidate calendar dates and fiscal patterns in the same hierarchy, the Calendar dimension is in the form of a snowflake dimensional structure. This is necessary because weeks do not roll up into the same hierarchy as months, and therefore require a separate hierarchy.</p> <p>For user-defined calendars, the lowest granularity is also a <i>day</i>, which can be rolled up into a user-defined period, such as fiscal period. User-defined calendars support the concept of detail and summary periods. A detail period consists of one or more days. A summary period consists of one or more detail periods. The user-defined calendar also supports fiscal calendars, which are limited to a specific fiscal year, as well as budget calendars, which can span multiple fiscal years.</p>
Currency Dimension	<p>Because transactional data can exist in any currency in which a organization does business, companies transacting business in multiple countries often must deal with data in multiple currencies. The Currency dimension enables you to present a unified view of your enterprise data.</p>
Language	<p>Companies that do business in different geographic areas often process data in different languages. The Language dimension contains a language ID, a two-letter language code, a three-letter language code, and a description. The two and three-letter language codes are based on International Organization for Standardization (ISO) codes. These ISO two and three-letter language codes are not abbreviations for the language, but they do identify a given language or group of languages.</p>
Time Dimension	<p>The Time dimension enables you to properly define a time of day attribute outside of the context of a specific date. This supports situations in which the time-only portion of a calendar is captured—as opposed to date and time. The granularity of the Time dimension is one minute.</p> <p>The Time dimension includes a textural Time Period attribute. This attribute refers to specific periods of time, such as AM or PM.</p>

Commonly Used Dimension	Description
Unit of Measure Dimension	<p>Measurements, particularly those that relate to the supply chain, can be complicated. For example, manufacturing might measure product in carload lots or pallets. Distribution might want to see everything in shipment cases, while retail can only process items in individual scan units. To satisfy reporting requirements for the various entities that use unit of measure (UOM), the PeopleSoft application presents the measured facts in a single, standard unit of measure, with conversion factors to all of the other possible units of measure in a separate conversion table.</p> <p>Because some units of measure are different when used for different products or items, a unit of measure relationship table used to facilitate a multi-tier hierarchy exists for the Unit of Measure dimension. This multi-tier system helps categorize a unit of measure and its conversion rate by role and conversion type, both of which are attributes of the Unit of Measure relationship table.</p> <p>Some conversions of UOM are standard and are independent from the subject of measurement, such as from meters to feet. However, some conversions depend on a set of attributes, such as shipping vendor, business unit, a particular item, and so on. The Unit of Measure table facilitates this conversion process.</p> <p>Note. You must populate this relationship table according to your particular requirements.</p>
Time Zone Dimension	<p>The Time Zone dimension component of date and time is required if your organization tracks events in different geographical locations situated in different time zones. In this situation, recording the time zone component of date and time is crucial.</p>

Data Models

Each EPM warehouse is delivered with its own set of *data models*, which are abstract models that define your data and the relationships among the data. Specifically, EPM warehouse data models dimensionalize your data, grouping it into facts and dimensions in a star-schema format based on specific business processes.

See the PeopleSoft Enterprise Performance Management Entity Relationship Diagrams located on My Oracle Support.

Measures

PeopleSoft EPM warehouses are delivered with prepackaged *measures*, which are numerical fact table values that have calculations (such as SUM, COUNT, or AVERAGE) applied to them. For example, the measure *SUM(SALES)* uses the Sales fact value and applies the SUM calculation to it.

Derived measures are also delivered with EPM warehouses. A derived measure includes a fact value and applies an arithmetic operator to it. Arithmetic operators are ADD, SUBTRACT, MULTIPLY, and DIVIDE. An example of a derived measure is $SUM(SALES*QTY)$ where SALES and QTY are each separate fact values and * signifies the arithmetic operator multiply.

EPM Architecture and Data Flow

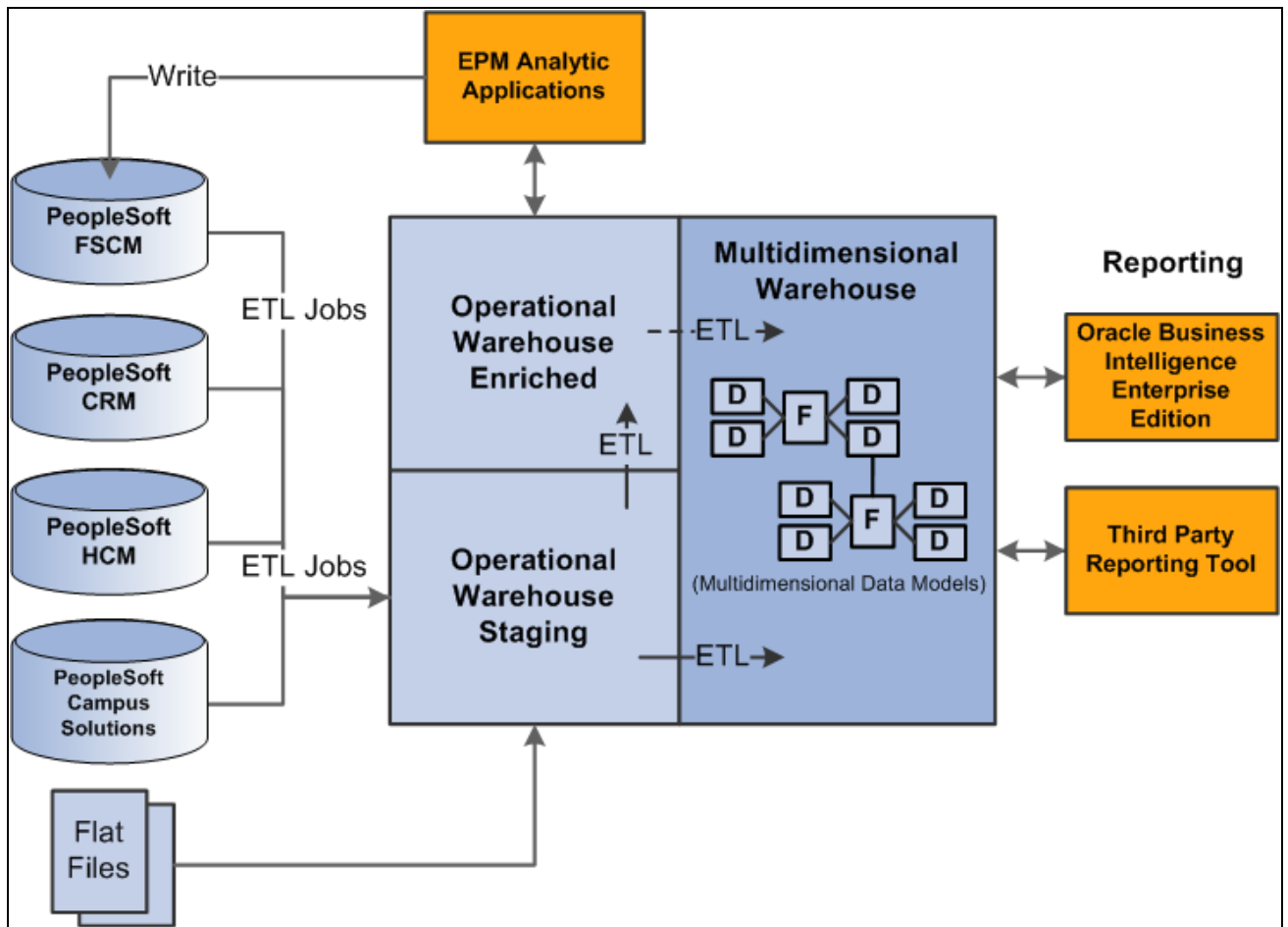
PeopleSoft EPM warehouses are built on a foundation of infrastructure tables and tools, ETL platform, and staging/multidimensional tables, all of which enable the warehouses to bring together data from different PeopleSoft source systems. Prepackaged data models enable complex analysis and reporting of your data.

To bring source data into an EPM warehouse and prepare your data for reporting, you must run prepackaged ETL jobs that extract information contained in PeopleSoft source systems and load it into multidimensional warehouse data models:

1. Use the ETL process to load your source data into the OWS.
2. Use the ETL utility to move data from the OWS to the MDW.
3. Complete setup of the Multidimensional Warehouse.
4. Review the chapters that describe the specific data marts that you are licensed to use and complete any additional setup that is necessary. Each data mart might have additional setup or processing steps that you must perform before creating the data mart. Review these steps in the chapter for that data mart in this PeopleBook.

See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Implementing PeopleSoft Enterprise Performance Management."

This graphic illustrates the various components comprising the EPM architecture and how data flows from source systems to the multidimensional warehouses via the ETL process:



EPM Data Flow

Operational Warehouse - Staging (OWS)

The first step in preparing your data for multidimensional reporting is to load source data from your PeopleSoft source transaction system into the OWS layer. You use PeopleSoft delivered ETL jobs to extract and load the data into the OWS. For Enterprise sources the ETL process does not transform the source data brought into the OWS, all table and field names and key structures are the same in the OWS as in the corresponding source table.

The ETL process brings dimension records, such as data for business units, calendars, and related language tables, from the source system, as well. In addition to the fields on the OWS tables that match those on the source tables, EPM adds additional fields to facilitate incremental loading (date stamps), and source and error tracking. These can typically be found in the LOAD_OWS_SBR subrecord.

See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Understanding PeopleSoft Enterprise Performance Management," Operational Warehouse - Staging (OWS).

Operational Warehouse - Enriched (OWE)

If you use the PeopleSoft EPM Analytic Applications in conjunction with the FMS Warehouse, you can use the prepackaged ETL jobs to move OWE data to the MDW layer:

- Profitability data (PS_PF_LEDGER_F00) is generated in the Analytic Applications and is stored in the OWE.
- Global Consolidation data (GC_CLED_MGT_F00) is generated in the Global Consolidations Analytic Application and stored in the OWE.

Note. Even if you do not use the Analytic Applications or the FMS Warehouse, you still use ETL jobs to move HCM Warehouse external survey data to the OWE before moving it to the MDW.

See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Understanding PeopleSoft Enterprise Performance Management," Operational Warehouse - Enriched (OWE).

Multidimensional Warehouse (MDW)

After you use ETL jobs to move your source data into the OWS, you use another set of ETL jobs to move your data into the MDW. The MDW is built on the principles of dimensional modeling—that is, logically modeling data for query performance starting from a set of base measurement events. Data in the MDW is grouped as it is related to one or more business processes. Data is in a star schema format—a fact table surrounded by one or more dimension tables. Generally, the star schema is in a denormalized form, which enables more efficient query processing.

In general, the MDW contains data at the most granular level—that is, the lowest level—found in the source system. This provides the most flexible choice regarding how report data is rolled up. The MDW data is based on surrogate keys rather than business keys, as this provides more efficient joining of tables. Values of surrogate keys contain no semantic content and are used specifically to join structures.

See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Understanding PeopleSoft Enterprise Performance Management," Multidimensional Warehouse (MDW).

Reporting on the EPM Warehouses

In order to leverage your data, the EPM warehouses are delivered with an open reporting platform (open data models), which enable you to add the Oracle Business Intelligence Enterprise Edition reporting tool or another third party reporting tool. Because the PeopleSoft open reporting solution stores the data mart data in relational tables, virtually any reporting tool that has connectivity to the database is able to use them.

See the Oracle Business Intelligence Enterprise Edition (OBIEE) suite of products and documentation.

Chapter 3

Understanding the Campus Solutions Warehouse

This chapter provides an overview of the Campus Solutions Warehouse, its components, and delivered fact and dimension tables.

Campus Solutions Warehouse Overview

The PeopleSoft Campus Solutions Warehouse is a comprehensive business intelligence platform for campus solutions analytics. At the core of the Campus Solutions Warehouse are prepackaged dimensional data models, which optimize the arrangement, accessibility, and reportability of your admissions, student records, student financial, and campus community data. With these data models you can review data against organizational metrics and perform strategic analyses using prepackaged measures such as Admissions Rate, Enrollment Rate, Graduation Rate, and Applicant, Admit, and Prospect Counts. You can slice and dice that data using various dimensions including Academic Career, Academic Program, Academic Plan, Academic Level, Term, Admit Type, Recruiting Category, Referral Source, Application Method, Ethnic Group, and Gender.

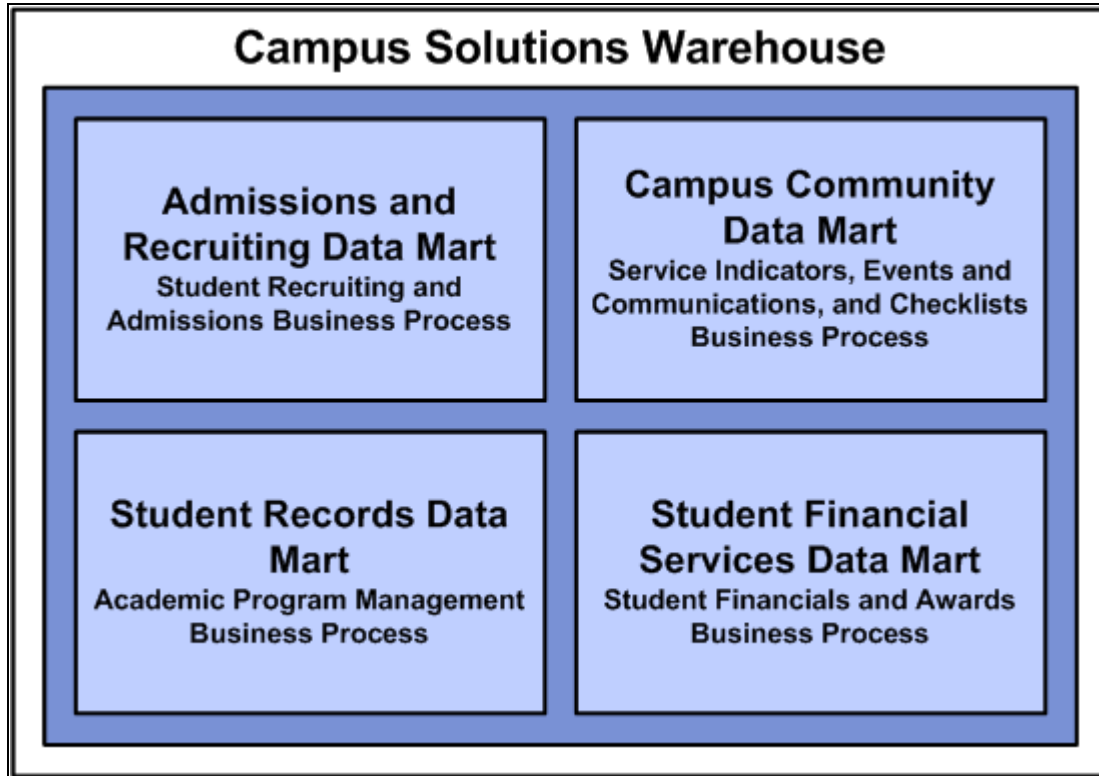
The Campus Solutions Warehouse enables you to identify key performance drivers, trends, and opportunities, and obtain actionable-insight into your institution so you can make informed strategic and operational decisions. With the Campus Solutions Warehouse you can better understand:

- Recruiting trends and recruiting effectiveness.
- Efficiencies and inefficiencies of the recruiting and admissions process.
- Applicant-progress within the admissions process.
- Student finances and financial aid.
- Class and term enrollments trends.
- Curricula offered and its effectiveness.

For more information about EPM, the Multidimensional Warehouse, and required setup tasks for the EPM warehouses, please refer to the *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*.

Campus Solutions Warehouse Structure

Data marts are logical divisions within the Campus Solutions Warehouse and are comprised of subject-specific dimensional data models designed around a specific institutional process. The Campus Solutions Warehouse includes the Admissions and Recruiting Data Mart, Campus Community Data Mart, Student Records Data Mart, and Student Financial Services Data Mart.



Campus Solutions Warehouse data marts and business processes

This section discusses:

- Admissions and Recruiting Data Mart
- Campus Community Data Mart
- Student Records Data Mart
- Student Financial Services Data Mart

Admissions and Recruiting Data Mart

With the Admissions and Recruiting Data Mart you can analyze the recruiting and admissions lifecycle. Applicant rates can be compared year over year by academic program to offer insight on the conversion percentage of applicants to students and the programs in which they are enrolled. External GPA and class percentile ranking can be tracked by institution and degree program to measure applicant academic excellence. Application evaluation and student responses can be tracked in detail to obtain insight into the admission process and cycle. The Admissions and Recruiting Data Mart provides a complete Admissions Funnel subject area that enables you to follow prospects and students through the admissions life cycle, analyzing their respective yields.

With the Admissions and Recruiting Data Mart you can answer questions such as:

- How many prospects are there for a particular academic program?
- How many applicants are admitted and enrolled in classes for a particular academic program in a particular admit term?

Admissions and Recruiting Data Mart Delivered Fact and Dimension Tables

This section discusses the delivered fact and dimension tables for the Admissions and Recruiting Data Mart.

Admissions and Recruiting Data Mart Fact Tables

The following table describes the delivered Admissions and Recruiting Data Mart fact tables.

Note. In the table, the *Helps Answer* column includes examples of the type of answers a fact table can provide; it does not contain the complete list of answers.

<i>Fact Name</i>	<i>Fact Record Name</i>	<i>PeopleSoft Source</i>	<i>Description</i>	<i>Helps Answer</i>
Student Admission Application	F_ADM_APPL	PS_ADM_APPL_SBP LAN PS_ADM_APPL_PLA N PS_ADM_APPL_DA TA PS_ADM_APP_PRO G	Contains current information of student applications (based on the warehouse load date) and provides information on applicants and their applications, and associated academic institutions, academic careers, programs, plans and subplans. There is one row of data per applicant, Institution, Career, career number , application, program, plan and subplan.	Current pool of applicants (specific admit term) by ethnicity, gender, applicant type, academic level. Applications by program action and program action reason (when populated) to analyze for example students that rejected applications and reasons, as well as withdrew students and other program actions. Number of waitlisted students.

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Application Evaluation	F_ADM_APPL_EVAL	PS_ADM_APPL_DATA PS_ADM_APP_CAR_SEQ PS_ADM_APPL_PROGRAM PS_ADM_APPL_EVAL PS_ADM_APPL_CMP	<p>Contains information about the application evaluation process, which are used to evaluate applicants on specific criteria for the academic career and program to which they are applying.</p> <p>This fact table provides insight into evaluation information, ratings, committees and additional information related to the application evaluation.</p> <p>CAREER_SID provides institution and career information and RATING_CMP_SID provides rating scheme and rating comp info.</p> <p>Institution and rating scheme are indirectly part of the key since they are derived from academic career and rating component respectively.</p> <p>There is one row of data per applicant, academic career, rating component, student career number, applicant number, application program number, application evaluation number and evaluation code.</p>	<p>Number of application evaluations by status (committee, final, on-hold statuses)</p> <p>Number of evaluations are still in progress by Institution, academic career and program.</p> <p>Ranking of application centers by number of application evaluations in progress.</p> <p>Analysis of application evaluations overall rating by Institution, academic career, program and admit type.</p> <p>Report number of application evaluations for a given rating component above a certain rating value.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Student Admission Applications Status	F_ADM_APPL_STAT	PS_ADM_APPL_DATA PS_ADM_APPL_PLAN PS_ADM_APPL_PROGRAM PS_ADM_APPL_SBP_PLAN	Contains application historical data and status changes, and provides information on applicants and their applications, and associated academic institutions, academic careers, programs, plans and subplans. There is one row of data per applicant, institution, career, career number, application, program, plan, subplan, action date and effective sequence.	Number of students went through a specific status during the admission process for a specific career, institution, and so forth. Number of students waitlisted during the admissions process. Number of changes in status did an application have.
Admissions Funnel	F_ADM_FUNNEL	PS_ADM_PRSPCT_CAR PS_ADM_PRSPCT_PROG PS_ADM_PRSPCT_PLAN PS_ADM_APPL_DATA	Contains information on number of prospects, applicants, admitted, confirmed and enrolled students. This table enables funnel report analysis by applicant type, academic level, academic load, last school attended, as well as by term, institution, career, program, campus, and so forth. The <i>admit term</i> is used as the date of capture for the snapshot data. There is one row of data per person, academic career, institution, admit term, academic program and academic plan.	Number of prospects, applicants, admitted, confirmed and enrolled students by admit term, institution, campus, career, and so forth. Funnel information by ethnicity, gender, academic level, academic type, and so forth. Yield trends by academic year, admit term, and so forth.

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
External Academic Summary	F_EXT_ACAD_SUM M	PS_EXT_ACAD_SU M	<p>Contains information on the external academic information of prospects and applicants, and an external academic summary entry of a prospect/applicant for each particular external academic career at a particular external organization, derived with a particular summary type.</p> <p>There is one row of data per person, external organization, external academic career, external data number (sequence number) and external summary type.</p>	<p>Average GPA of the prospect/applicant in a particular external organization.</p> <p>Number of course units the prospect/applicant attempted in a particular external organization.</p> <p>Number of course units the prospect/applicant completed in a particular external organization.</p> <p>Converted GPA of the prospect/applicant.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
External Test Scores	F_EXT_TESTSCORE	PS_STDNT_TEST_C OMP	<p>Contains information on all internal and external test scores of prospects and applicants, and contains every test score entry of a prospect/applicant for a particular test and the related components taken on a particular date, obtained from a particular data source.</p> <p>Test ID is indirectly part of the key since it is derived from test component. EXT_TST_CMPNT_S ID performs a lookup to D_EXT_TST_CMPNT by Test ID and Test Component.</p> <p>There is one row of data per person, test component, test date and test data source.</p> <p>To link test score information to admissions applications you need to use this fact table in conjunction with another bridge table provided in the warehouse (R_TESTSCORE_APPL).</p>	<p>List of students with ACT/math rating greater than average.</p> <p>Min, Max, Average test value by test component, test and percentile.</p> <p>Number of students with self reported test scores.</p> <p>Number of students that have taken specific test component/test more than once.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Student Response	F_STDNT_RESP	PS_STDNT_RESPON SE PS_ADM_APPL_DA TA PS_ADM_APP_CAR_ SEQ PS_ADM_APPL_PRO G	<p>Contains information about student responses to your institution and enables you to analyze, for example, why students chose not attend your institution and where they attended instead. You can also analyze why students did select your school.</p> <p>There is one row of data per applicant, career, academic career number, admission application number, academic program, effective date and effective sequence.</p> <p>Institution is indirectly part of they key since it is derived from academic career.</p>	<p>Number of students offered acceptance that have not responded.</p> <p>Reasons why students declined admission to your institution.</p> <p>Number of negative/positive student responses by reason and by admit term, institution, program, and admit type.</p>
Student Recruiting	F_STU_RECRT	PS_ADM_PRSPCT_C AR PS_PRSPCT_RECRT ER PS_PRSPCT_RCR_C AT PS_ADM_PRSPCT_P LAN PS_ADM_PRSPCT_P ROG PS_ADM_PRSPCT_S BPL	<p>Contains data on prospective students, associated academic institutions, academic careers, academic programs, plans and subplans.</p> <p>This fact also contains data on the recruiting categories, regions and the recruiters the prospective students are assigned to.</p> <p>There is one row of data per prospect, institution, academic career, academic program, plan, subplan, recruiter category and recruiter.</p> <p>The measures provided in this table enable you to analyze the number of prospects by recruiter, recruiter category, referral source, and so forth.</p>	<p>Number of prospects by recruiter, recruiter category, referral source (and so forth) by institution, campus, program, plan for a specific admit term.</p> <p>Top high schools from where students originate.</p>

Admissions and Recruiting Data Mart Dimension Tables

The following table describes the delivered Admissions and Recruiting Data Mart dimension tables.

Dimension Name	Dimension Record Name	Description
Academic Rank Type	D_ACAD_RANK_TYP	Defines various rank types.
Academic Unit Type	D_ACAD_UNIT_TYP	Defines various academic unit types.
Application Center	D_APPL_CNTR	Defines centers where admissions applications are processed.
Application Method	D_APPL_MTHD	Indicates how or in what form an application was received.
Student Ethnicity	D_ETHNICTY	Stores information about the ethnic group to which a student belongs. This dimension is snowflaked to the Person (D_PERSON) dimension.
Evaluation Code	D_EVAL_CODE	Defines evaluation codes for admission applications.
Evaluation Status	D_EVAL_STATUS	Defines the status of admission application evaluations.
External Academic Career	D_EXT_ACAD_CAR	Defines academic careers from external organizations.
External Academic Level	D_EXT_ACAD_LVL	Defines academic levels from external organizations.
External Academic Summary Type	D_EXT_SUMM_TYP	Defines the types of academic summary data from external organizations that is captured for prospects and applicants.
External Term	D_EXT_TERM	Defines the terms from external organizations.
External Test Component	D_EXT_TST_CMPNT	Defines the components of a test, such as a quantitative or verbal election.
GPA Type	D_GPA_TYPE	Defines the different grade point average types.
Rating Component	D_RATING_CMP	Defines the rating components for application evaluations.
Rating Scheme	D_RATING_SCH	Defines the rating schemes for application evaluations.

Dimension Name	Dimension Record Name	Description
Recruiting Application Center	D_REC_APPL_CNTR	Stores information about the recruiting application center associated with the recruiter. This dimension is snowflaked to the Recruiter (D_RECRTR) dimension.
Recruiting Center	D_RECRT_CNTR	Contains a list of recruiting centers which identify the prospects and recruiters who belong to a particular recruiting office.
Recruitment Category	D_RECRT_CTGRY	Defines special categories of prospective students to be targeted for recruiting and admissions purposes.
Recruiting Status	D_RECRT_STAT	Indicates the level of interest, at the academic career level, that the institution has in the prospective student.
Recruiter	D_RECRTR	Stores recruiter information, such as recruiter name, ID, type, and institution.
Recruiter Category	D_RECRTR_CAT	Stores information about the category associated with the recruiter. This dimension is snowflaked to the Recruiter (D_RECRTR) dimension.
Recruiting Center	D_RECRTR_CNTR	Stores information about the recruiting center associated with the recruiter. This dimension is snowflaked to the Recruiter (D_RECRTR) dimension.
Recruiter External Org	D_RECRTR_EXT	Stores information about the external organization associated with the recruiter. This dimension is snowflaked to the Recruiter (D_RECRTR) dimension.
Recruiter Academic Plan	D_RECRTR_PLAN	Stores information about the academic plan associated with the recruiter. This dimension is snowflaked to the Recruiter (D_RECRTR) dimension.

Dimension Name	Dimension Record Name	Description
Recruiter Academic Program	D_RECRTTR_PROG	Stores information about the academic program associated with the recruiter. This dimension is snowflaked to the Recruiter (D_RECRTTR) dimension.
Recruiter Region	D_RECRTTR_REG	Stores information about the region associated with the recruiter. This dimension is snowflaked to the Recruiter (D_RECRTTR) dimension.
Recruiter Role	D_RECRTTR_ROLE	Stores information about the role associated with the recruiter. This dimension is snowflaked to the Recruiter (D_RECRTTR) dimension.
Student Response	D_RESP_RSN	Stores information about the detailed reasons given with a student's responses for an admission application.
Referral Source	D_RFRL_SRC	Tracks how prospects or applicants learned of the institution and indicates why they were originally added to the system.
Test Data Source	D_TST_DATA_SRC	Indicates how the test scores were provided to the institution.
Admissions Recruiter (<i>Relationship Table</i>)	R_ADM_RECRTTR	Links the one to many relationships between the Student Admission Application fact (F_ADM_APPL) and the Recruiter dimension (D_RECRTTR).
External Degree (<i>Relationship Table</i>)	R_EXT_DEG	Lists the degrees a prospect or applicant has received from external organizations.
Prospect Recruiter (<i>Relationship Table</i>)	R_PRSPCT_RECRTTR	Links the one to many relationships between the Student Recruiting fact (F_STU_RECRT) and the Recruiter dimension (D_RECRTTR).
Test Score Application (<i>Relationship Table</i>)	R_TESTSCORE_APPL	Links the one to many relationships between the External Test Scores fact (F_EXT_TESTSCORE) and the Student Admission Application fact (F_ADM_APPL).

Campus Community Data Mart

The Campus Community data mart enables you to analyze student information, service indicators, communications and checklists.

With the Campus Community data mart you can track and analyze:

- service indicators, which enables you to identify which services should be available to a student.
- communications to parties associated with your institution, which enables you to better understand the impact of recruitment events on student enrollment.
- events, meetings, and attendees that are associated with an event.
- checklists for students and external organizations, and enables you to analyze the relationship of checklists between the university admissions office and prospective or accepted students.

With the Campus Community Data Mart you can answer questions such as:

- How many service indicators are outstanding for a particular service reason?
- How many person communications have been created for an institution?
- How many events for a given Campus event type?
- How many organization checklists have been created for an institution?
- How many service indicators are outstanding for a student?
- How many organization communications have been created for a department?

Campus Community Data Mart Delivered Fact and Dimension Tables

This section discusses the delivered fact and dimension tables for the Campus Community Data Mart.

Campus Community Data Mart Fact Tables

The following table describes the delivered Campus Community Data Mart fact tables.

Note. In the table, the *Helps Answer* column includes examples of the type of answers a fact table can provide; it does not contain the complete list of answers.

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Campus Events	F_CAMPUS_EVENT	PS_CAMPUS_EVENT	<p>Contains information about events, such as when an event occurred or when it is planned, the number of meetings associated with an event, expected attendance, number of attendees, and so forth.</p> <p>The information can be analyzed by institution, event type, event manager, and so forth.</p> <p>There is one row of data per campus event.</p>	<p>Number of events an institution offered during the last academic year, by event type.</p> <p>Number of attendees an per event.</p> <p>Average number of meetings, by event, for an institution.</p> <p>Average number of attendees versus invited.</p> <p>Top 10 events by number of attendees.</p>
Campus Event Meeting	F_CAMP_EVNT_MTG	PS_EVENT_MTG PS_CAMPUS_EVENT	<p>This fact table enables you to analyze current credit history information and credit history trends for a given business unit, account type, student, external org, and so forth.</p> <p>Provides analysis of events, event meetings, and event attendees. Also provides lower level of granularity than the Campus Event fact table, since the Campus Event Meeting fact table provides information at the event meeting level, and it allows analysis of events and event meetings by institution, event type, coordinator, external organization, and department.</p> <p>There is one row of data per campus event and event meeting number.</p>	<p>Number of event meetings that occurred in the campus facilities in the last academic year.</p> <p>Number of event meetings that occurred outside the campus facilities in the last academic year.</p> <p>Number of campus events by institution, by campus event type.</p> <p>Number of meeting events occurring on specific dates, detailing projected attendance, number of people invited and number of attendees.</p> <p>Top 10 meeting events ranked by attendance and/or percentage of attendance attendees/invitations. And location of event and day of week it occurred.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Organization Check List	F_CHKLST_ORG	PS_PERSON_CHECK LST	<p>Provides information about external organization checklists. Checklists may be lists of steps that must be performed, or documents that must be provided, or communications that are planned to occur, and so on.</p> <p>This table enables you to analyze external organization checklists by organization, admin function, institution, and related checklist items.</p> <p>There is one row of data per Organization ID and sequence number.</p>	<p>Number of external organizations with pending (incomplete) checklists for a specific admin function.</p> <p>Number of external organizations with past due checklists, by institution and admin function.</p> <p>Number of external organizations with specific checklist items that are still incomplete. And how many of them were created by a given user.</p> <p>Number of incomplete checklists for a given department.</p>
Person Check List	F_CHKLST_PERSON	PS_PERSON_CHECK LST	<p>Provides information about person (student) checklists. Checklists may be lists of steps that must be performed, or documents that must be provided, or communications that are planned to occur, and so on.</p> <p>This table enable you to analyze person organization checklists by student, admin function, institution, and related checklist items.</p> <p>There is one row of data per Person ID and sequence number.</p>	<p>Number of students with pending (incomplete) checklists for a specific admin function (such as admissions application).</p> <p>Number of students with past due checklists, by institution and admin function.</p> <p>Number of students with specific checklist items that are still incomplete. And how many of them were created by a given user.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Organization Communications	F_COMM_EXT_ORG	PS_COMMUNICATION	<p>Provides analysis of external organization communications. This fact table enables you to analyze External Organization Communications by organization, administrative function, and related variable data, institution, and outcome reason.</p> <p>There is one row of data per external organization and sequence number.</p>	<p>Number of communications created for a given external organization during a given time period. Number of those that have been completed.</p> <p>Number of communications with enclosures for a given external organization.</p> <p>Top 10 external organizations ranked by total number of communications (both complete and incomplete) for a given time period.</p>
Person Communications	F_COMM_PERSON	PS_COMMUNICATION	<p>Provides analysis of person communications by person, administrative function, related variable data, institution and outcome reason.</p> <p>There is one row of data per person and sequence number.</p>	<p>Number of communications created for a given person for a given time period. Number that have been completed.</p> <p>Number of communications with enclosures for a given person.</p> <p>Top 10 people ranked by total number of communications (both complete and incomplete) for a given time period.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Service Indicators	F_SRVC_INDCTRS	PS_SRVC_IND_DATA	<p>Provides analysis of service indicators and related impact values by student or external organization.</p> <p>Negative service indicators can be holds that prevent an individual or organization from receiving certain services and positive indicators can designate special services to be provided. Service indicators consist of one or more impact values that identify the types of specific services that are restricted or provided.</p> <p>There is one row of data per person and service indicator date/time stamp.</p>	<p>Number of outstanding service indicators for a student.</p> <p>Type of service indicator attached to a student.</p> <p>Number of outstanding service indicators for an institution.</p> <p>Number of outstanding service indicators for a particular service code or service reason.</p> <p>Users that have created a given service indicator and when.</p> <p>Which service impacts are attached to a given service indicator.</p> <p>Number of students with service indicators that are for a given service impact.</p> <p>Which service indicators have been released (deleted).</p>

Campus Community Data Mart Dimension Tables

The following table describes the delivered Campus Community Data Mart dimension tables.

Dimension Name	Dimension Record Name	Description
Admin Function	D_ADMIN_FUNC	Lists administrative functions.
Aging Category	D_AGING_SF	Contains a list of aging set and aging category descriptions.
Campus Event	D_CAMPUS_EVENT	Lists campus events.
Checklist Code	D_CHKLST_CD	Contains a list of available checklist codes.
Checklist Status	D_CHKLST_STAT	Contains a list of checklist statuses.
Campus Event Attendee	D_CMP_EVNT_ATND	Contains a lists campus event attendees.

Dimension Name	Dimension Record Name	Description
Communication Category	D_COMM_CATGRY	Contains a list of available communication categories.
Communication Context	D_COMM_CTXT	Contains a list of communication context.
Communication Direction	D_COMM_DIR	Contains a list of available communication directions.
Communication Method	D_COMM_MTHD	Contains a list of available communication methods.
Event Meeting	D_EVENT_MTG	Contains details of campus event meetings.
Checklist Item	D_ITEM_CD	Contains a list of checklist item codes.
Letter Code	D_LTR_CODE	Contains a list of available letter codes.
Outcome Reason	D_OUTCM_RSN	Contains a list of available communication outcome reasons.
Service Impact	D_SRVC_IMPACT	Maintains information about the impact entities in a campus community and includes several attributes about a service impact like description, institution code, and service impact code fields.
Service Indicator	D_SRVC_IND	Stores information about service indicator codes by institution.
Service Indicator Reason	D_SRVC_IND_RSN	Stores service indicator reasons by institution and service indicator code.
Variable Data	D_VAR_DATA	Provides details of variable data based on administrative function.
Checklist Item (<i>Relationship Table</i>)	R_CHKLST_ITEM	Bridges the one to many relationships of person and organization checklist facts to the corresponding checklist item code dimension.
Campus Event Attendee (<i>Relationship Table</i>)	R_CMP_EVNT_ATND	Links the one to many relationships between one-to-many relationships between event meetings and the event attendees.

Dimension Name	Dimension Record Name	Description
Service Impact (<i>Relationship Table</i>)	R_SRVC_IMPACT	Links the one to many relationship between the Service Indicator (D_SRVC_IND) fact and Service Impact (D_SRVC_IMPACT) dimension.

Student Records Data Mart

With the Student Records Data Mart you can review and manage items such as enrollment and registration metrics, count of student and faculty by registered courses, available courses for catalog building and schedule building, graduation rates and student career fulfillment of requirements, student academic standing, and faculty and student profiles.

Enrollment reports include areas such as monitoring average class sizes, prerequisites not being met, grade distributions, and graduation eligibility. Reports in this category include aggregated statistical reports with year-to-year comparisons in several subject areas.

With the Student Records Data Mart you can answer questions such as:

- What are the enrollment metrics for this term?
- Can I get the Student Retention and Graduation rates year over year by Cohort, Gender and Ethnicity?
- What is average GPA by Institution, Career and Program?
- What is average time to graduate by Institution, Career and Program?
- What classes are scheduled for this term?
- Can I track and analyze the workload of the faculty?
- What are the Class enrollment details?
- What are the honors/awards details for the enrolled students?

Academic Program Activation and Management Business Process

Student Records Data Mart is related to the Academic Program Activation and Management business process. This business process includes Course Catalog, Class Scheduling, Student Career Term records and Enrollment business processes. These processes fulfill the institutions need to track course delivery, student participation through enrollments to those classes. The Academic Program activation and Management processes help manage class size, a student's enrollment in a class and track the resulting grades from the class.

Student Records Data Mart Delivered Fact and Dimension Tables

This section discusses the delivered fact and dimension tables for the Student Records Data Mart.

Student Records Data Mart Fact Tables

The following table describes the delivered Student Records Data Mart fact tables.

Note. In the table, the *Helps Answer* column includes examples of the type of answers a fact table can provide; it does not contain the complete list of answers.

<i>Fact Name</i>	<i>Fact Record Name</i>	<i>PeopleSoft Source</i>	<i>Description</i>	<i>Helps Answer</i>
Academic Program Detail	F_ACAD_PROG_DTL	PS_ACAD_PROG	<p>Contains information about an individual program action entry for a particular student for a given academic program.</p> <p>This fact table contains all students that enrolled, matriculated, withdrew, or completed an academic career/program at an institution.</p> <p>There is one row of data per student (person) surrogate ID, student career number, program surrogate ID, effective date/sequence and program action surrogate ID.</p> <p>The table is not keyed by term. Institution and career are indirectly part of the key since they are derived from the Academic Program dimension through the Academic Program Surrogate ID.</p>	<p>Number of active students in a program by institution, career, or program.</p> <p>Number of students that are active in more than one program, by institution or campus.</p> <p>Number of students with transfer status who were transferred during a specific time period.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Academic Plan Summary	F_ACADPLAN_SUM M	PS_ACAD_PROG PS_ACAD_PLAN	<p>Contains student summary entry for a given academic plan and related academic program.</p> <p>Each row of this table contains the most current information about an individual student and a particular academic plan. This fact table also contains all students that enrolled, matriculated, withdrew, or completed the academic career/program and plan in the institution.</p> <p>The table is not keyed by term. Institution, career and program are indirectly part of the key since they are derived from the Academic Plan dimension.</p> <p>There is one row of data per student (person) academic plan surrogate ID, and student career number.</p> <p>Records are updated frequently, whenever the student program status is changed, such as activated, admitted, discontinued or completed.</p>	<p>Number of enrollments by program/plan.</p> <p>Number of drop-outs.</p> <p>Number of students in different program action and program action reasons, by career, program, or plan.</p> <p>Number of students that completed the program/plan.</p> <p>Number of students that cancelled the program/plan.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Class Enrollment	F_CLASS_ENRLMT	PS_TERM_TBL PS_STDNT_ENRL	<p>Contains information about student class enrollment, such as class enrollment status, units taken, units in progress, grades, grade points, and units earned.</p> <p>There is one row of data per Term, Session, Person (Student), Class and Class Number.</p> <p>Institution and career are indirectly part of the key since they are derived from the Session dimension table (SESSION_SID is populated from PS_D_SESSION based on the unique combination of Institution, Career, Term and Session).</p>	<p>Number of students enrolled in a class.</p> <p>Number of dropout for a specific class.</p> <p>Number of students that passed or failed a class.</p> <p>Class roster for a particular department.</p>
Class Instructor	F_CLASS_INSTRCT	PS_INSTR_TERM_D TL PS_INSTRUCTOR_T ERM	<p>Contains information on instructor workload such as classes/class sections taught by instructor, instruction mode, instructor role, location, campus, facility and also indicates the assignment percentage.</p> <p>There is one row of data per instructor (person), instructor assignment sequence, term and session.</p> <p>Institution and career are indirectly part of the key since they are derived from the Session dimension.</p> <p>Note. The assumption is that you use instructor workload process in the Campus Solutions application.</p>	<p>Total FTE per instructor.</p> <p>Workload by instructor.</p> <p>Instructors with a workload that exceeds 90%.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Class Meeting Pattern	F_CLASS_MTG_PAT	PS_CLASS_TBL PS_CLASS_MTG_PAT PS_CLASS_INSTR	<p>Contains schedule information for a course including start and end dates, days of the week, and times. The data represents the superset of all classes offered, regardless of any student enrollment.</p> <p>This fact table enables you to analyze class meeting pattern and instructor information (when that information is available for the class). If the information regarding meeting pattern and instructor does not exist, it will also show a row for the class, and the class meeting pattern number and instructor assignment sequence will be set to zero.</p> <p>There is one row of data per Session Surrogate ID, Course Surrogate ID, Class Section Code, Class Meeting Number, and instructor assignment number.</p> <p>Institution, Career, and Term are indirectly part of the key since they are derived from the Session Surrogate ID, as well as by Course ID and Course offer number that are derived from the Course Surrogate ID.</p>	<p>Number of courses taught, by subject.</p> <p>Average number of instructors, by course and specific subject.</p> <p>Classes taught on Friday afternoon.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Degrees and Honors	F_DEGREES	PS_ACAD_DEGR PS_ACAD_DEGR_H ONS PS_ACAD_DEGR_PL AN PS_ACAD_DEGR_SP LN	Provides analysis of student degrees as well as related honors. It provides measures on degrees and honors conferred by student, term, institution, career, program, academic plan, academic sub plan, and so forth. There is one row of data per student, degree number, honors number, plan and subplan.	Number of students graduating with honors by term/institution/career. Number of those students who are athletes. Honors awarded by major.
Enrollment Requests	F_ENRL_REQ	PS_ENRL_REQ_DET AIL PS_ENRL_REQ_HEA DER	Provides analysis of student enrollment requests. This table enables you to analyze enrollment request metrics by student, academic career, institution, term, status, action, reason, an so forth. This table can provides insight into the classes that are more popular, which classes have the longest waitlists, substitute classes, an so forth. There is one row of data per person, career, institution, term, enrollment request ID, enrollment request header status and enroll request sequence number.	Number of classes a student enrolled or attempted to enroll for a given term. Popular classes, by enrollment and student. Popular instructors by enrollment and student. Number of units by student a for a given term. Classes with a high number of repeat attempts by students. Substitute classes requested if students cannot enroll in a particular class. Students enrolled in a class without having met the class prerequisite.

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Institution Summary	F_INST_SUMM	PS_X_INST_SUMM (from PS_ACAD_PLAN and PS_F_ACAD_PLAN_SUMM) PS_R_STDNT_COHORT (from PS_STDNT_ATTR_D TL)	<p>Contains information about the retention and graduation counts of an institution. Each row has calculated metrics for every academic program, admit year, admit term, admit type, student gender, ethnicity, and student cohort.</p> <p>This table is an aggregate fact and each row in the table has calculated measures for every academic plan, admit term, admit type, student gender, and ethnicity.</p> <p>This table is designed to store data that corresponds to a list of Academic Careers (Parameter), Full time or Part time academic load (Parameter) for different actions and action reasons (like discontinuation of the course to join armed forces) (Parameter).</p>	<p>Institution 4-year retention rate by cohort, gender and ethnicity.</p> <p>Institution 4-year graduation rate by cohort, gender and ethnicity.</p> <p>Graduation and student trends.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Term Enrollment	F_TERM_ENRLMT	PS_STDNT_CAR_T ERM PS_TERM_TBL	Provides term statistics and cumulative statistics by term, student, institution and career. This fact also provides measures such as units in progress, GPA, number of courses, and so forth. Institution and career are indirectly part of the key since they are derived from the Term surrogate ID. There is one row of data per term surrogate ID and student.	Top student academic standings. Number of units in progress versus passed, by institution, career, program, or plan. Average number of courses a student takes per term, by career, program, or plan. Number of units a student is enrolled in. Number of full time and part time students, by term, institution, campus, or primary program. Number of students taking GPA units. Enrollment analysis, by primary program and plan, term, institution, campus, career, and program.

Student Records Data Mart Dimension Tables

The following table describes the delivered Student Records Data Mart dimension tables.

Dimension Name	Dimension Record Name	Description
Academic Group	D_ACAD_GRP	Defines all the academic groups of an institution.
Academic Organization	D_ACAD_ORG	Defines the organizational entities in an academic organization.
Academic Standing	D_ACAD_STNDNG	Contains a list of all the academic standings of an Institution.
Class	D_CLASS	Stores course information such as course ID and section, location, and academic group.
Course	D_CRSE	Defines all the course offerings for an Institution.
Degree	D_DEG	Contains the list of degrees conferred by the institution.

Dimension Name	Dimension Record Name	Description
Degree Honors	D_DEG_HONORS	Defines the degree honor codes.
Degree Status	D_DEG_STAT	Indicates the status of the degree.
Enrollment Action	D_ENRL_ACTION	Contains a list of enrollment request actions such as enroll, drop, add grade, drop grade, and change wait-list position.
Enrollment Detail Status	D_ENRL_DTL_STAT	Contains a list of enrollment detail statuses, such as pending, submitted, and successful.
Enrollment Header Status	D_ENRL_HDR_STAT	Contains a list of enrollment header statuses such as pending, submitted, and successful.
Enrollment Message	D_ENRL_MSG_LOG	Contains a list of enrollment messages such as class full and minimum requirements not met.
Enrollment Reason	D_ENRL_RSN	Contains a list of enrollment request reasons such as student withdrawal, term cancellation, and student request drop.
Enrollment Status	D_ENRLMT_STAT	Contains a list of enrollment statuses.
Ethnic Group	D_ETHNIC_GRP	Contains a list of all the ethnic groups.
Facility	D_FCLTY	Defines the facilities in various campuses of an Institution.
Gender	D_GNDR	Provides student gender detail.
Grade	D_GRADE	Defines the valid grade codes.
Instruction Mode	D_INSTRCTN_MODE	Defines all the course mode of instructions, such as distant and classroom.
Instructor Load	D_INSTRCTR_LOAD	Denotes whether an instructor is full-time or part-time.
Instructor Class Role	D_INSTRCTR_ROLE	Defines the different roles that can be played by a faculty/instructor.
Repeat	D_REPEAT	Contains a list of all the course repeat codes.

Dimension Name	Dimension Record Name	Description
Session	D_SESSION	Contains a list of all the academic sessions of a term.
SSR Component	D_SSR_COMP	Defines all the SSR components; for example it can specify the section of a course.
Student Cohort	D_STDNT_COHORT	Contains a list of all the student cohorts (student attributes) of an Institution.
Student Group	D_STDNT_GRP	Contains a list of all the student groups of an Institution.
Relate Academic Subplan <i>(Relationship Table)</i>	R_ACAD_SPLAN	Represents the relationship between an academic plan to its subplans.
Relate Award <i>(Relationship Table)</i>	R_AWD	Represents the relationship between a student to the awards he/she has received.
Relate Degree Honors <i>(Relationship Table)</i>	R_DEG_HONORS	Represents the relationship between a student degree to the honors he/she has received.
Enrollment Request Message <i>(Relationship Table)</i>	R_ENRL_MSG_LOG	Links the one to many relationship between the Enrollment Request (F_ENRL_REQ) fact and the Enrollment Message (D_ENRL_MSG_LOG) dimension.

Student Financial Services Data Mart

The Student Financial Services Data Mart is comprised of two subject areas: Student Financial Services and Financial Aid.

Student Financial Services is used by institutions to manage student receivables, billing, and collections, as well as the transactions that are supported by reports, such as Receipts Per Day and Refund Customers Trial Balance by Students. This subject area of the Campus Solutions Warehouse links student financial information to dimensions such as Person, Account, and Product, enabling the analysis of student financial aid information from various viewpoints.

The Financial Aid subject area automates federal and institutional aid processing and helps you to manage student financial aid activity for students and applicants. Reports for financial aid application provide details such as Year to Date Disbursement, Overawards, and Pell Originations. This subject area links financial aid information to dimensions such as Person, Account and Product, enabling you to perform analysis of aid information from various viewpoints.

With the Student Financial Services Data Mart you can answer questions such as:

- What is the total amount of bills for a student or an external organization?

- What is the total amount of tuition charged for a particular course?

Award Business Process

Student Financial Services Data Mart is related to the Award business process. The Award Business Process fulfills an institutions need to track and report student's payments for classes taken as well as financial aid awards given to students for an academic year.

Student Financial Services Data Mart Delivered Fact and Dimension Tables

This section discusses the delivered fact and dimension tables for the Student Financial Services Data Mart.

Student Financial Services Data Mart Fact Tables

The following table describes the delivered Student Financial Services Data Mart fact tables.

Note. In the table, the *Helps Answer* column includes examples of the type of answers a fact table can provide; it does not contain the complete list of answers.

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Award Disbursement	F_AWD_DISB	PS_STDNT_AWARD S PS_STDNT_AWRD_ DISB	<p>Contains term-based detail information associated with a student award such as amount, status, disbursement, aid type, and so forth.</p> <p>There is one row of data per person, institution, academic career, financial aid item type, aid year and disbursement ID.</p>	<p>Number and amount of awards accepted by source (federal, state, or institutional, for example) and type (grants, scholarships, loans, or work, for example) or for an individual item type.</p> <p>Average award accepted.</p> <p>Number and amount of awards declined by source (e.g., federal, state, institutional) and type (e.g., grants, scholarships, loans, work) or for an individual item type</p> <p>Declined awards as a percentage of total awards offered.</p> <p>Students with undisbursed awards for a particular term.</p> <p>Students with pending disbursements for a particular time period.</p> <p>Students with unaccepted awards.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Award Summary	F_AWD_SNPSHT	Award Disbursement (F_AWD_DISB) Fact Table	<p>This snapshot summary table provides Award Amount and Award Count issued to a person for an academic year. Award amount represents the amount of a financial aid award to a student. This can represent the offered, disbursed, accepted or authorized amount depending on the award status. Award Count represents the count of financial aid award to a student. This can represent the offered, disbursed, accepted or authorized counts depending on the award status.</p> <p>The frequency at which this table is loaded determines the snapshot period. The snapshot period is determined by the last run of the ETL process and the new run of the ETL process.</p> <p>There is one row of data per snapshot date, academic career, financial aid item type, institution, person and aid year.</p>	<p>Total award offered, accepted, disbursed, and authorized by institution, academic career and aid year.</p> <p>Top 10 careers with highest award offered by aid year and institution.</p> <p>Number of awards offered by institution, academic career, and aid year.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Bill Summary	F_BIL_SNPSHT	PS_BI_BILLING_LIN E PS_BI_BILL_HEADE R	<p>Contains accumulated snapshot billing information for a student or external organization between a period of time. The information the table provides is the total summary billing information (billed and paid amount) for a student or external organization for a specific snapshot period.</p> <p>There is one row of data per snapshot date, business unit, person, external organization and student/organization type.</p> <p>Source aggregated data is defined using time interval parameters, namely, FROM_DATE and TO_DATE. The snapshot data is identified by a SNAP_DT_SID. Data that is present in the source between these dates is picked up by the source query and is aggregated over the total amount billed to a student or external organization per month. Then it is populated along with the total amount of payments made by a student or external organization, the number of bills unpaid, the number of bills paid late.</p>	<p>Total billed and paid amount by business unit, person or external organization in a specific time frame (snapshot period).</p> <p>Total billed/paid amount by career, program, plan for a specific time frame (snapshot period).</p> <p>Top billed career in the snapshot period.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Credit History	F_CREDIT_HIST	PS_CREDIT_HISTOR Y	<p>This table allows you to analyze current credit history information and credit history trends for a given business unit, account type, student, external org, and so forth.</p> <p>Contains information on student and external organization accounts by aging set and aging category.</p> <p>This fact table also enables you to track whether the credit history process has assigned any service indicators to a particular student.</p> <p>There is one row of data per business unit, person or external organization, student/organization type, account number, account term, effective date (date when process credit history was run) and aging set category.</p>	<p>Students that are regularly delinquent on payments based on credit history trends.</p> <p>Total balance past due per external organization/student when process credit history is run for a given effective date.</p> <p>Number of student accounts with service indicators placed on them based on credit history process.</p> <p>Number of accounts with cleared service indicators.</p> <p>Student accounts with service indicators placed by process credit history (with detailed service indicator code and reasons).</p> <p>Balance past due for a particular aging category and aging set for a given student and effective date when process credit history is run.</p>
Pending Payments	F_PYMNT_PENDIN G	PS_S_PAYMENT_TB L PS_ITEM_SF	<p>Contains summary information of payment and charges. This table aggregates payment amount and charge amount by business unit, person/external organization, student/organization type, academic year and term.</p>	<p>Total amount of payment/charges by SFBU, academic year and term.</p> <p>Percentage of total amount that corresponds to student accounts and percentage that corresponds to external organization.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
SF General Ledger Accounting Entries	F_SF_ACCOUNT_LN	PS_SF_ACCTG_LN PS_JRNL_HEADER	<p>Contains the accounting entries produced by the Student Financials system and enables you to analyze the accounting transactions generated (by business unit, ledger, fiscal year, accounting period and so forth) and whether such transactions have been transferred to the general ledger.</p> <p>There is one row of data per run date, sequence number and student financials line number.</p>	<p>Transactions passed/not passed to the general ledger for a specified date range. Also, number of transactions and transactions amount.</p> <p>Transactions passed to the general ledger analysis, by department, chartfield1, chartfield2, chartfield3 and/or operating unit.</p>
Payment Detail	F_SF_PAYMENT	PS_S_PAYMENT_TBL PS_ITEM_SF	<p>Contains information about payments at a detail level and enables you to analyze payments by business unit, payment method, item type, term, academic year, and so forth.</p> <p>There is one row of data per business unit, person or external organization, student/organization type and payment ID.</p>	<p>Number of payments from self-service and lockbox by SF business unit, term and academic/calendar year.</p> <p>Total amount of payment (checks or credit card, for example) applied to a student ' account in this SFBU, by term, academic or calendar year, career, program, or college.</p> <p>Students with unapplied payments.</p> <p>Students with an overall credit balance and unapplied payments.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Student Financials Transactions	F_SF_TRAN	PS_ITEM_SF	<p>Contains information about student financial transactions and enables you to analyze number of transactions, transaction amounts, paid amounts and encumbered amounts by academic year, calendar year, accounting term, account type, billing date, item type and so forth.</p> <p>This table serves as a roll-up, or summarization, of the Student Financial Transaction Detail fact table.</p> <p>There is one row of data per business unit, person/external organization, student/organization type and transaction number (item number).</p>	<p>Total tuition charged for a particular academic/calendar year, institution, or career.</p> <p>Top 10 external organizations with higher waiver amounts in a particular academic or calendar year.</p> <p>Total tuition/encumbered amount for this term, career, or program.</p> <p>Average tuition (tuition and fees) charged for students in this SFBU, this term, academic or calendar year, career, program or college.</p> <p>Total amount of waivers applied against tuition (tuition and fees) in this SFBU, term, academic or calendar year, career, program, or college.</p> <p>Number of enrolled students who have not had tuition calculated.</p>

Fact Name	Fact Record Name	PeopleSoft Source	Description	Helps Answer
Transaction Detail	F_SF_TRAN_DTL	PS_ITEM_LINE_SF PS_ITEM_SF	<p>Contains information about student financial transactions at a detail level and enables you to analyze line items by business unit, item type, account type, term, academic year, and so forth.</p> <p>There is one row of data per business unit, person/external organization, student/organization type, transaction number and line sequence number.</p>	<p>Number of transactions not posted to the general ledger, by business unit, career, or program.</p> <p>Number of transactions/transaction amount set to "aging transactions" in the last month, by business unit, career, or program.</p> <p>Number of transaction with a refund and amount, by calendar or academic year.</p> <p>Amount of disputed transaction by career, program, or college.</p> <p>Top 10 students with higher dispute amounts.</p>
Payments and Charges Cross Reference	F_PYMNT_CHARGES	PS_ITEM_XREF PS_ITEM_SF	<p>Provides information about payments applied to charges within student financials and enables you to analyze payments applied to charges by business unit, item type, account type, term, academic year, and so forth.</p> <p>There is one row of data per business unit, person/external organization, student/organization type, item number charge and item number payment.</p>	<p>Payment method used to apply payments to specific items.</p> <p>Total amount of payments applied to a given item charge by business unit, account term, item term, academic career, academic year, program, and so forth.</p> <p>Federal Financial Aid audits to provide evidence that financial aid was used to pay for eligible charges or ensure tuition scholarships were used to pay for tuition and course/class fees.</p>

Student Financial Services Data Mart Dimension Tables

The following table describes the delivered Student Financial Services Data Mart dimension tables.

Dimension Name	Dimension Record Name	Description
Account Term	D_ACCT_TERM	Contains term value data that relates to an institution quarter or semester definition regardless of academic unit.
Account Type	D_ACCT_TYPE	Contains account types used for student financial transactions.
Federal Aid Year	D_AID_YR	Contains details about the federal aid year.
Award	D_AWD	Contains a list of all the awards which an Institution confers to its students.
Award Period	D_AWD_PRD	Contains the different award periods that can be used for financial aid awards.
Award Status	D_AWD_STAT	Contains details of the status of a financial aid award.
Contract	D_CONTRACT	Provides a list of contracts.
Disbursement	D_DISB	Contains the disbursement plan and split code values established for each financial aid item type, by career.
Dependency Status	D_DPNDCY_STAT	Defines the dependency information of the student applying for financial aid.
Financial Aid Item Type	D_FA_ITEM_TYPE	Contains attributes that may be associated with financial aid item types used to define the type of financial aid award.
Financial Aid Application	D_FED_AID_APP	Contains attributes necessary to maintain the FAFSA (Free Application for Federal Student Aid) application and changes to Institutional Student Information Record (ISIR) data.
Financial Aid Type	D_FIN_AID_TYPE	Contains the category or type of financial aid offered by a financial aid item type.
Item Code	D_ITEM_CODE	Provides a list of item codes.
Item Type	D_ITEM_TYPE	Contains details associated with item types, which are used to define the type of financial transactions.

<i>Dimension Name</i>	<i>Dimension Record Name</i>	<i>Description</i>
Line Reason	D_LINE_REAS	Provides a list of line item reason codes.
Line Action	D_LINE_ACTN	Provides a list of line item actions.
Line Status	D_LINE_STAT	Provides a list of line item status values
Payment Method	D_PYMNT_MTD	Contains a list of payment methods.
Tuition Group	D_SEL_GRP	Provides a list of tuition groups.
Session Code	D_SESSION_CODE	Contains a list of all of the session codes.
Student Aid Attributes	D_STU_AID_ATTR	Defines records for students applying for aid and contains the attributes, high level information relating to the application, awarding and processing of financial aid for the year. The status fields indicate if aid-processing steps such as SSNMatch, Verification Status, Scholarship status etc., have been completed for a particular application.
Student Financial Aid Terms	D_STU_FA_TERM	Defines a consolidated view of student term data for Financial Aid processing purposes.

Common Dimensions in the Campus Solutions Warehouse

The following table describes the common dimension tables that are shared across the Campus Solutions Warehouse.

<i>Dimension Name</i>	<i>Dimension Record Name</i>	<i>Description</i>
Academic Career	D_ACAD_CAR	Defines all the academic careers that an institution offers.
Academic Load	D_ACAD_LOAD	Defines the academic loads offered; for example, full time or part time.
Academic Level	D_ACAD_LVL	Defines the academic level associated with a student; for example first year, second year, freshman, or sophomore.
Academic Plan	D_ACAD_PLAN	Contains all the academic plans within an academic program of an institution.

Dimension Name	Dimension Record Name	Description
Academic Program	D_ACAD_PROG	Contains a list of all the academic programs of an institution.
Academic Subplan	D_ACAD_SPLAN	Contains a list of all the academic sub plans with an academic plan of an institution.
Admit Type	D_ADMIT_TYPE	Defines the admit types that are assigned to prospects and applications to clarify the type of prospect or applicant, such as first year, readmit, or transfer.
Campus	D_CAMPUS	Defines all the campuses for an Institution.
External Organization	D_EXT_ORG	Defines the external organizations for posting external education data and the last school attended for admissions applicant data.
Institution	D_INSTITUTION	Contains a list of all academic institutions.
Person Additional Attributes	D_PERSON_ATTR	Defines all the additional attributes of a person which are specific to the student role.
Program Action	D_PROG_ACN	Contains a list of all the academic program actions.
Program Action Reason	D_PROG_ACN_RSN	Contains a list of all the academic program action reasons.
Program Status	D_PROG_STAT	Contains a list of all the academic program statuses.
Campus Region	D_REGION_CS	Defines the geographical groupings that the institution uses for recruiting and admissions; for example, a region can be a continent, a country, a state, a portion of a state, or a mix of states.
Residency	D_RSDNCY	Defines the residency of a prospect of applicant to the institution.
Term	D_TERM	Contains a list of all the academic terms of an institution.
Relate Student Cohort (<i>Relationship table</i>)	R_STDNT_COHORT	Represents the relationship between a student and the cohorts (attributes).

Dimension Name	Dimension Record Name	Description
Relate Student Group (<i>Relationship table</i>)	R_STDNT_GRP	Represents the relationship between a student and the student groups.

Shared Dimensions

Certain dimensions, such as Account or Department are used across all EPM warehouses. These dimensions are identical in structure and content across all EPM warehouses. The following table describes the delivered shared dimension tables.

Dimension Name	Dimension Record Name	Description
Account	D_ACCOUNT	Stores details of an account that represents a ChartField.
AP Document Type	D_AP_DOC_TYPE	Stores details about AP document types, such as Payables Payments, Payables Adjustments, Payables Accruals, and so on.
Association Type	D_ASSOC_TYPE	Defines the association type for Case, Interaction and Order association.
Bank Account	D_BANK_ACCT	Store details about banks and bank accounts.
Book Code	D_BOOK_CODE	Stores details about book codes, which represent an account attribute and a balancing ChartField.
Budget Reference	D_BUDGET_REF	Stores budget descriptions.
Buyer	D_BUYER	Stores information on buyers, including information related to a buyer's employee ID and address.
Contract	D_CA	Stores the details of the contract information entered with customers. A contract contains the agreement information and obligations for the products and services licensed in the contract and is grouped by contract type.
Carrier	D_CARRIER	Stores information on carriers.
Certification Source	D_CERTSRC	Stores information on certification sources for suppliers.
Channel	D_CHANNEL	Stores channel information related to sales and procurement.

Dimension Name	Dimension Record Name	Description
Chartfield1	D_CHARTFIELD1	Stores user defined ChartField details.
Chartfield2	D_CHARTFIELD	Stores user defined ChartField details.
Chartfield3	D_CHARTFIELD3	Stores user defined ChartField details.
Channel Partners	D_CHNL_PARTNER	Stores information about channel partners involved in the sales process.
Expenses Classifications	D_CLASS_FIELD	Stores expenses classification codes and descriptions, such as wages, benefits, health, and office supplies.
Company	D_CMPNY	Stores company-related information.
Credit Risk	D_CREDIT_RISK	Classifies credit risk values as High, Low, and Medium.
Customer Contact Person	D_CUST_CNTCT	Stores information about the customer contact person, which includes contacts and partners.
Customer Organization	D_CUST_ORG	Stores information related to customer organizations (companies). A customer organization is a company that purchases, leases, or contracts for products or services. The customer organization (company) is a subset of the Customer dimension.
Customer Person	D_CUST_PERSON	Stores information about individuals that purchase, lease, and contract for products or services. The Customer Person is a subset of the Customer dimension.
Customer Site	D_CUST_SITE	Stores information about organizations that purchase, lease, and contract for product or services located at a particular site or location. Sites can be an organization site or an individual site. Site is also a subset of the Customer dimension.
Customer Master	D_CUSTOMER	Stores information for entities that can participate in business relationships.

Dimension Name	Dimension Record Name	Description
Department	D_DEPT	Stores information about the entities in an organization. This dimension includes attributes about a department, such as description, company code, location, and budget fields.
Employee Job Code	D_EMPL_JOB	Stores employee job history data, such as actions taken, department, job code, location, and salary history. Multiple records can be created for an employee.
Establishment	D_ESTAB	Stores distinct physical places of business (establishments) within a company and its address, and is used for regulatory reporting purposes.
Frequency	D_FREQ	Stores the payment and hours reporting frequency for time and payroll data. You can use a frequency to indicate how many times per year an event occurs.
Fund	D_FUND	Stores details about fund codes and their description.
GL Adjustment types	D_GL_ADJ_TYPE	Stores types of general ledger (GL) adjustments.
GL Offset	D_GL_OFFSET	Stores information on GL offset. This dimension groups billing information, such as office rent and retail rent.
Industry Group	D_INDUSTRY_GRP	Stores customer industry group information.
Inventory Item	D_INV_ITEM	Stores information about Inventory Item, which includes all attributes of item, including simple hierarchy information, such as category or group, as well as Make or Buy flag
Inventory Location	D_INV_LOCATION	Stores information about the storage location from which goods will be moved.
Jobcode	D_JOBCODE	Stores information about the job assignments in an organization. This dimension represents the categorization of jobs into types, such as executive, technical, and administrative services.

Dimension Name	Dimension Record Name	Description
Journal Line Source	D_JRNL_SOURCE	Stores the details about source of journal entries created in GL.
Sales Lead	D_LEAD	Stores sales leads generated by marketing campaign waves.
Ledger	D_LEDGER	Stores the ID and description of ledgers that are defined based on templates.
Line Type	D_LN_TYP	Stores information on line types.
Location	D_LOCATION	Stores a list of work sites for an organization. Location is used to establish physical locations in an organization, such as corporate headquarters, branch offices, and remote sales offices.
Lot	D_LOT	Stores information on lot (a group of items with similar characteristics).
Operating Unit	D_OPER_UNIT	Stores details about operating units, such as a plant, office, physical location, branch, and building.
Sales Opportunity	D_OPPORTUNITY	Stores information about a sales opportunity.
Order Capture	D_ORD_CAPTURE	Stores order capture information for the sales order process.
Sales Order Status	D_ORD_STAT	Stores information on order status.
Partner	D_PARTNER	Stores partner information. The dimension has the following hierarchy: Partner, Partner Status.
Pay Group	D_PAYGRP	Groups employees by how they are paid.
Person	D_PERSON	Stores the most current personal information of both employees and non-employees of an organization.
AR Specialist	D_PERSON_ARSPL	Stores details, such name and contact, about the accounts receivable (AR) specialist involved in handling the disputes and deductions in the AR module.

Dimension Name	Dimension Record Name	Description
AR Collector	D_PERSON_COLTR	Stores details, such name and contact, about the AR collector involved in collecting the receivables amount in the AR module.
AR Credit Analyst	D_PERSON_CRNYST	Stores details, such name and details, about the AR credit analyst involved in handling the credits given to customers.
AR Deduction Manager	D_PERSON_DEDMGR	Stores AR deduction manager name and contact information.
Position	D_POS	Stores information on all job positions available, whether an employee fills the position or no, and helps with data analysis based on salary or standard hours.
Product Group	D_PROD_GROUP	Stores information on product groups.
Product	D_PRODUCT	Stores information on products.
Program	D_PROGRAM_FDM	Keeps track of programs, such as public works, social services, fire, and public safety, that are tracked in General Ledger.
Project	D_PROJECT	Stores information about projects. A project is a vehicle for identifying an initiative that has a specified start and end date.
Partner Contact	D_PRTR_CNTCT	Stores partner contact data.
Payment Method	D_PYMNT_MTHD	Stores methods of payment, such as check, cash, and credit card.
Receive Line Status	D_RECLN_STATUS	Stores information on all receive line statuses.
Regulatory Region	D_REG_RGN	Stores the codes for regulatory and regional edit purposes. A regulatory region is any region where there are specific laws and regulations that are used for transactional purposes.
Geographic Region	D_REGION	Contains geography information for customers.

Dimension Name	Dimension Record Name	Description
Salary Plan	D_SALPLN	Stores unique salary categories that are defined in an organization. These categories are set up according to an employee's compensation structure.
Scenario	D_SCENARIO	Stores details of historical, budgeting, and forecast scenarios.
Customer Segment	D_SEGMENT	Stores customer segment information.
Statistics Code	D_STAT_CODE	Stores details about statistical information, such as floor space, full-time equivalent workdays, and shipment size.
Subledger	D_SUBLEDGER	Stores information on subledger, which groups the accounting information.
Supplier	D_SUPPLIER	Stores information on suppliers, such as remit to supplier and corporate supplier.
Sales Territory	D_TERRITORY	Stores sales territory information. Sales territories are user defined sales regions independent of geography or proximity.
Unit	D_UNIT	Stores detail information on real estate properties.
Unit of Measure	D_UOM	Indicates the quantity in which an inventory item is expressed, such as case (CS) or box (BX).

Chapter 4

Running Campus Solutions Warehouse Implementation Jobs

This chapter discusses how to run the jobs required to implement the Campus Solutions Warehouse.

Note. For detailed information about delivered Campus Solutions Warehouse jobs and the order in which you should run them, please see the *Using the ETL Lineage Spreadsheets appendix*.

Prerequisites

After you have configured IBM WebSphere DataStage for EPM, and before you run Campus Solutions Warehouse implementation jobs, you must:

- Import all of the appropriate *.dsx files containing your ETL jobs.
- Compile all jobs.
- Run hashed file setup jobs.
- Run initial OWS setup jobs to bring source data into the Campus Solutions Warehouse.
- Run the Dimension Mapper setup jobs.
- Run shared lookup jobs.
- Run the Setup - OWE jobs.
- Run the Common Dimensions jobs.

For more information about these prerequisites, see the *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*.

Also note that because the following IBM WebSphere DataStage folders contain implementation jobs for all EPM warehouses, you must identify which jobs relate to your warehouse and, optionally, delete any unwanted jobs.

- OWS
- GLOBAL_DIMENSION
- LOCAL_DIMENSION
- Global_D00

- OWE_E

You can also create your own master sequencer job, which you can use to drag and drop only those jobs relating to your warehouse and then run the master sequencer; or you can use the master sequencer utility to automate this activity. The delivered ETL lineage spreadsheet can help you identify which jobs apply to the EPM product you purchased.

See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Preparing to Load Source Data Into EPM"; *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Running Initial Setup Jobs"; *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Importing Source Business Units into EPM to Create Warehouse Business Units" and *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "ETL Configurations," Using the Master Sequencer Utility to Create Master Sequencer Jobs.

See Appendix B, "Using the PeopleSoft EPM Lineage Spreadsheets," page 201.

Running Campus Solutions Warehouse Implementation Jobs

This section discusses how to run all the ETL implementation jobs for the Campus Solutions Warehouse, in the following order:

1. Campus Solutions Warehouse - OWS jobs.
2. Global dimension jobs for Campus Solutions Warehouse.
3. Local dimension jobs for Campus Solutions Warehouse.
4. Campus Solutions Warehouse - SKU jobs.

Running Campus Solutions Warehouse - OWS Jobs

The first step in implementing the Campus Solutions Warehouse is to run the Campus Solutions Warehouse - OWS jobs. These jobs consist of Campus Solutions Warehouse-specific hash file jobs and OWS jobs. Run the hash file jobs first, as the tables that they load are required to run your standard OWS jobs.

As with most prepackaged jobs, you can use the Master Run Utility to automatically run a set of jobs located in a flat file on the IBM WebSphere DataStage Server. When you use the Master Run Utility, it reads a list of jobs that are present in a specified flat file and triggers the jobs to run in serial mode, using the dependency logic specified in the Input flat file.

Campus Solutions Warehouse - OWS Hash File Jobs

Perform the following steps to run the Campus Solutions Warehouse - OWS hash file jobs:

1. In IBM WebSphere DataStage Director, navigate to the hash file jobs by expanding the nodes in the left navigation panel using the following path: *CS_E, OWS, Base, Load_Hash_Files, Server*.
2. Select each CS Warehouse - OWS hash file job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time and the job's status is updated to *Running*.

Campus Solutions Warehouse - OWS Jobs

Perform the following steps to run the Campus Solutions Warehouse - OWS jobs:

1. In IBM WebSphere DataStage Director, navigate to the CS Warehouse - OWS jobs by expanding the nodes in the left navigation panel using the following path: *CS_E, OWS, Base, Load_Tables, Sequence*.
2. Select each CS Warehouse - OWS job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time and the job's status is updated to *Running*.

Running Global Dimension Jobs for Campus Solutions Warehouse

The second step in implementing the Campus Solutions Warehouse is to run the global dimension jobs for Campus Solutions Warehouse. These jobs consist of global dimension hash file jobs and global dimension jobs. Run the hash file jobs first, as the tables that they load are required to run your standard global dimension jobs.

Note. You can run global dimension jobs individually or together using the master sequence job.

Global Dimension Hash File Jobs

Perform the following steps to run the global dimension hash file jobs individually:

1. In IBM WebSphere DataStage Director, navigate to the global dimension hash file jobs by expanding the nodes in the left navigation panel using the following path: *Global_Dimensions_E, OWS_To_MDW, Base, Load_Hash_Files, Server*.
2. Select each global dimension hash file job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time, and the job's status is updated to *Running*.

Perform the following steps to run the global dimension hash file jobs together using the master sequence job:

1. In IBM WebSphere DataStage Director, navigate to the master sequence job by expanding the nodes in the left navigation panel using the following path: *Global_Dimensions_E, Master_Sequence*.
2. Select the global dimension master sequence job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time, and the job's status is updated to *Running*.

Global Dimension Jobs

Perform the following steps to run the global dimension jobs individually:

1. In IBM WebSphere DataStage Director, navigate to the global dimension hash file jobs by expanding the nodes in the left navigation panel using the following path: *Global_Dimensions_E, OWS_To_MDW, Base, Load_Hash_Files, Sequence*.

2. Select a global dimension job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time, and the job's status is updated to *Running*.

Perform the following steps to run the global dimension jobs together using the master sequence job:

1. In IBM WebSphere DataStage Director, navigate to the master sequence job by expanding the nodes in the left navigation panel using the following path: *Global_Dimensions_E, Master_Sequence*.

2. Select the master sequence job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time, and the job's status is updated to *Running*.

Running Local Dimension Jobs for Campus Solutions Warehouse

The third step in implementing the Campus Solutions Warehouse is to run the local dimension jobs for Campus Solutions Warehouse. These jobs consist of local dimension hash file jobs and local dimension jobs. Run the hash file jobs first, as the tables they load are required to run your standard global dimension jobs.

Note. You can run local dimension jobs individually or together using the master sequence job.

Local Dimension Hash File Jobs

Perform the following steps to run the local dimension hash file jobs individually:

1. In IBM WebSphere DataStage Director, navigate to the global dimension hash file jobs by expanding the nodes in the left navigation panel using the following path: *CS_E, Local_Dimensions, OWS_To_MDW, Base, Load_Hash_Files, Server*.

2. Select each local dimension hash file job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time and the job's status is updated to *Running*.

Perform the following steps to run the local dimension hash file jobs together using the master sequence job:

1. In IBM WebSphere DataStage Director, navigate to the master sequence job by expanding the nodes in the left navigation panel using the following path: *CS_E, Local_Dimensions, Master_Sequence*.
2. Select the master sequence job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time and the job's status is updated to *Running*.

Local Dimension Jobs

Perform the following steps to run the local dimension jobs individually:

1. In IBM WebSphere DataStage Director, navigate to the local dimension jobs by expanding the nodes in the left navigation panel using the following path: *CS_E, Local_Dimensions, OWS_To_MDW, Dimensions, Base, Load_Tables, Sequence*.
2. Select each local dimension job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time and the job's status is updated to *Running*.

Perform the following steps to run the local dimension jobs together, using the master sequence job:

1. In IBM WebSphere DataStage Director, navigate to the master sequence job by expanding the nodes in the left navigation panel using the following path: *CS_E, Local_Dimensions, Master_Sequence*.
2. Select the master sequence job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time, and the job's status is updated to *Running*.

Running Campus Solutions Warehouse SKU Jobs

The fourth and final step in implementing the Campus Solutions Warehouse is to run the Campus Solutions Warehouse SKU jobs. These jobs consist of hash file jobs, dimension jobs, and fact jobs. Run the hash file jobs first, as the tables that they load are required to run your dimension and fact jobs.

Note. You can run Campus Solutions Warehouse SKU jobs individually or together using the master sequence job.

Campus Solutions Warehouse SKU Hash File Jobs

Perform the following steps to run the Campus Solutions Warehouse SKU hash file jobs individually:

1. In IBM WebSphere DataStage Director, navigate to the CS Warehouse SKU hash file jobs by expanding the nodes in the left navigation panel using the following path: *CS_E, [SKU/Data Mart Name], [Business Process], OWS_To_MDW, Dimensions, Base, Load_Hash_Files, Server.*

2. Select each hash file job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time and the job's status is updated to *Running*.

Perform the following steps to run the CS Warehouse SKU hash file jobs together using the master sequence job:

1. In IBM WebSphere DataStage Director, navigate to the master sequence job by expanding the nodes in the left navigation panel using the following path: *CS_E, [SKU/Data Mart Name], [Business Process], Master_Sequence.*

2. Select the master sequence job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time and the job's status is updated to *Running*.

Campus Solutions Warehouse Dimension Jobs

Perform the following steps to run the Campus Solutions Warehouse dimension jobs individually:

1. In IBM WebSphere DataStage Director, navigate to the CS Warehouse dimension jobs by expanding the nodes in the left navigation panel using the following path: *CS_E, [SKU/Data Mart Name], [Business Process], OWS_To_MDW, Dimensions, Base, Load_Tables, Sequence.*

2. Select each dimension job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time and the job's status is updated to *Running*.

Perform the following steps to run the CS Warehouse dimension jobs together using the master sequence job:

1. In IBM WebSphere DataStage Director, navigate to the master sequence job by expanding the nodes in the left navigation panel using the following path: *CS_E, [SKU/Data Mart Name], [Business Process], Master_Sequence..*

2. Select the master sequence job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time, and the job's status is updated to *Running*.

Campus Solutions Warehouse Fact Jobs

Perform the following steps to run the Campus Solutions Warehouse fact jobs individually:

1. In IBM WebSphere DataStage Director, navigate to the CS Warehouse fact jobs by expanding the nodes in the left navigation panel using the following path: *CS_E, [SKU/Data Mart Name], [Business Process], OWS_To_MDW, Facts, Base, Load_Tables, Sequence*.
2. Select each fact job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time and the job's status is updated to *Running*.

Perform the following steps to run the fact jobs together using the master sequence job:

1. In IBM WebSphere DataStage Director, navigate to the master sequence job by expanding the nodes in the left navigation panel using the following path: *CS_E, [SKU/Data Mart Name], [Business Process], Master_Sequence*.
2. Select the master sequence job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time and the job's status is updated to *Running*.

Chapter 5

Configuring Slowly Changing Dimensions

This chapter provides an overview of slowly changing dimensions and discusses how to convert type 1 slowly changing dimension jobs to type 2 slowly changing dimension jobs.

Understanding Slowly Changing Dimensions

Data warehouses store historical data from an online transaction processing (OLTP) system. As new data is extracted into the data warehouse from the source OLTP system, some records may change. When the attributes of a given dimension table change, this is called a *slowly changing dimension*.

For example, an organization may use its Product dimension table to store product descriptions. The description lists the ingredients of the product. If there is a change to the ingredient list, the description in the OLTP is updated to reflect this change. When the changed record (the slowly changing dimension) is extracted into the data warehouse, the data warehouse updates the appropriate record with the new data. How that change is reflected in the data warehouse depends on how slowly changing dimensions has been implemented in the warehouse.

There are three types of slowly changing dimensions:

- *Type 1 Slowly Changing Dimension:* This method overwrites the existing value with the new value and does not retain history.
- *Type 2 Slowly Changing Dimension:* This method adds a new row for the new value and maintains the existing row for historical and reporting purposes.
- *Type 3 Slowly Changing Dimension:* This method creates a new *current value* column in the existing record but also retains the original column.

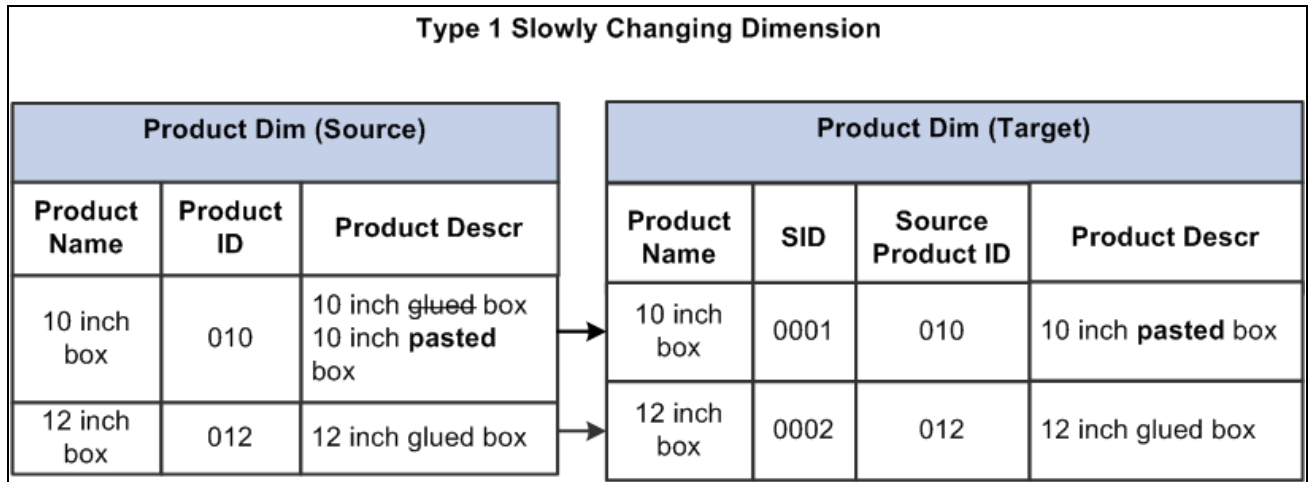
Note. PeopleSoft does not support type 3 slowly changing dimensions.

Type 1 Slowly Changing Dimensions

A type 1 slowly changing dimension overwrites the existing data warehouse value with the new value coming from the OLTP system. Although the type I does not maintain history, it is the simplest and fastest way to load dimension data. Type I is used when the old value of the changed dimension is not deemed important for tracking or is an historically insignificant attribute.

For example, a company that manufactures cardboard boxes might have a Product dimension table that tracks the product ID, product name, and product description. Similar columns would be present in the *warehouse* Product dimension, with the addition of a surrogate ID (primary key) to track each unique record.

If one of the product descriptions were to change from *glued box* to *pasted box* in the OLTP system, it would trigger a slowly changing dimension event in the warehouse Product dimension. If you want to overwrite the former description without saving history, you would use type 1 slowly changing dimension:



Type 1 slowly changing dimension

Note. After overwriting an existing dimension value, you may find that some of your reports that depended on the value will not return the same information as before.

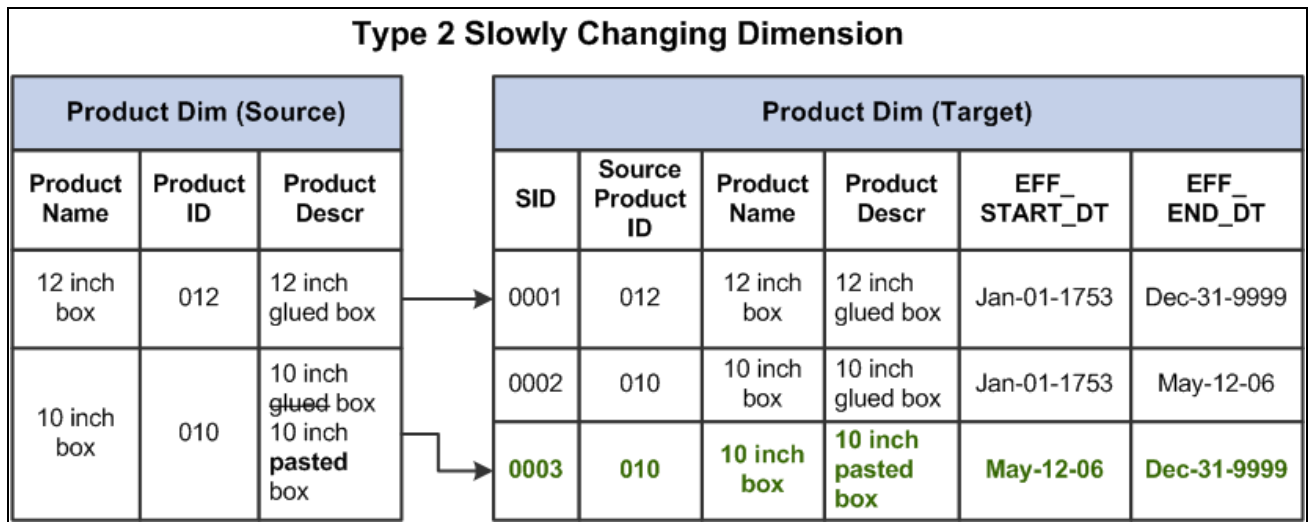
Type 2 Slowly Changing Dimensions

A type 2 slowly changing dimension enables you to track the history of updates to your dimension records. When a changed record enters the warehouse, it creates a new record to store the changed data and leaves the old record intact. Type 2 is the most common type of slowly changing dimension because it enables you to track historically significant attributes. The old records point to all history prior to the latest change, and the new record maintains the most current information.

Each change to a dimension generates a new dimension record, and each record partitions history. This is done by a combination of:

- Effective dating both the new and old record (the old record is assigned a non-active effective date and the new record is assigned an active effective date).
- Assigning the new record a new (and unique) surrogate key.

Using the same cardboard manufacturing company as an example from the previous section, and assuming one of the product descriptions changed from *glued box* to *pasted box* in the OLTP system, type 2 slowly changing dimension would be used to retain the former description while incorporating the new. Instead of overwriting the existing value in the product description column, a new record is added, and a new surrogate ID (primary key) is assigned to the record. The original record with the description *glued box* remains. The following graphic demonstrates this type 2 slowly changing dimension scenario:



Type 2 slowly changing dimension

Note that the values for source product ID and source product name columns remain unchanged, but the surrogate key values are unique for each record and the effective start and end dates indicate the current record. This distinguishes the past and current records and enables you to report on historical and current data alike.

The main drawback of type 2 slowly changing dimensions is the need to generalize the dimension key and the growth of the dimension table itself. The dimension table could become quite large in cases where there are a number of changes to the dimensional attributes that are tracked.

Type 3 Slowly Changing Dimensions

A type 3 slowly changing dimension creates a new current value column in the existing record but retains the original column as well. The new current value column holds the new dimension data coming from the OLTP system. This type of slowly changing dimension is used when a change in a dimension value must be tracked but the old value must be retained as part of the record, usually for reporting.

For example, a type 3 slowly changing dimension might be useful in a sales force realignment. When the names of the sales regions have changed but there is a need to state today's sales in terms of the past region names for comparison, a new field in the sales dimension table named *current_region* is added. The old field can be renamed to *previous_region* and no changes are made to the sales dimension record keys or to the number of sales team records. These two fields now enable an application to group all sales fact records by either the old sales assignments (*previous region*) or the new sales assignments (*current region*).

Type 3 slowly changing dimensions handle only the two most recent changes. If many changes take place and they must all be tracked, type 2 slowly changing dimensions should probably be used.

Note. PeopleSoft does not support type 3 slowly changing dimensions.

Understanding Slowly Changing Dimensions in EPM

EPM is designed to support both type 1 and type 2 slowly changing dimensions, while type 3 are not supported. The majority of prepackaged EPM dimensions are set to type 1 with a smaller number set to type 2 (for example, D_EMPL_JOB).

Because the EPM data model supports both type 1 and type 2 slowly changing dimensions, there is no need to modify the data model should you wish to change a dimension from a type 1 to a type 2. You need only modify the ETL job that loads the dimension and, in some instances, the fact job that uses the dimension as a lookup. Instructions for modifying these jobs are discussed in latter sections of this chapter.

Every EPM dimension table includes a *Valid Date Range* subrecord to help facilitate the process of converting a type 1 slowly changing dimension to a type 2. The subrecord tracks the date range for which a version of a dimension entity was valid. The subrecord is discussed in further detail below.

Valid Date Range Subrecord

All EPM dimension tables have a Valid Date Range subrecord added to them to facilitate implementation of type 1 and type 2 slowly changing dimensions. The following table displays the structure of the Valid Date Range subrecord:

Column	Data Type
EFF_START_DT	Date
EFF_END_DT	Date
CURRENT_IND	CHAR(1)

EFF_START_DT and EFF_END_DT

For type 1 slowly changing dimensions, the EFF_START_DT and EFF_END_DT columns are assigned default values. EFF_START_DT is set to *Jan-01-1753* and EFF_END_DT is set to *Dec-31-9999*.

For type 2 slowly changing dimensions, the EFF_START_DT and EFF_END_DT columns serve to partition the related dimension records and indicate which version is active. When a changed record is extracted into an MDW dimension table, the new record is assigned an EFF_START_DT value, which is derived from the EFFDT column. The old record is assigned an EFF_END_DT value equal to the new EFFDT minus one day ($\text{EFFDT} - 1 \text{ day} = \text{EFF_END_DT}$ of old record), and the new record is assigned an EFF_END_DT value equal to *Dec-31-9999*.

Note. The EFF_START_DT and EFF_END_DT columns are populated during ETL process.

CURRENT_IND

When a control (dimension) table in the source system has multiple records with the same business keys and different effective dates, the corresponding tables in the MDW also have multiple records with the same business keys and different `EFF_START_DT` and `EFF_END_DT` values. The applications only use the row that is currently valid (when the system date falls between the `EFF_START_DT` and `EFF_END_DT` values).

To help determine which records are valid and active, the `CURRENT_IND` column has been added to dimension tables and it indicates whether a row is active for a given system date. The `CURRENT_IND` column uses two-valued logic. Current rows are marked with `CURRENT_IND = 'Y'` and past and future dated rows are marked with `CURRENT_IND = 'N'`.

For a type 1 slowly changing dimension implementation, this column will have a default value of 'Y'.

Design Differences Between Type 1 and Type 2 Slowly Changing Dimension Jobs

This section describes the differences between type 1 and type 2 slowly changing dimension jobs in EPM and is divided into the following topics:

- Source Query
- Target DRS Stage
- Target Lookup Stage

Source Query

The source query for a type 1 slowly changing dimension has a correlated sub query to take the latest effective dated row from the source table in the source DRS (Dynamic Relational Stage).

The source query for the type 2 slowly changing dimension does not have a correlated sub query; instead it uses an *ORDER BY* clause based on the effective date from the source table in the source DRS (Dynamic Relational Stage).

Target DRS Stage

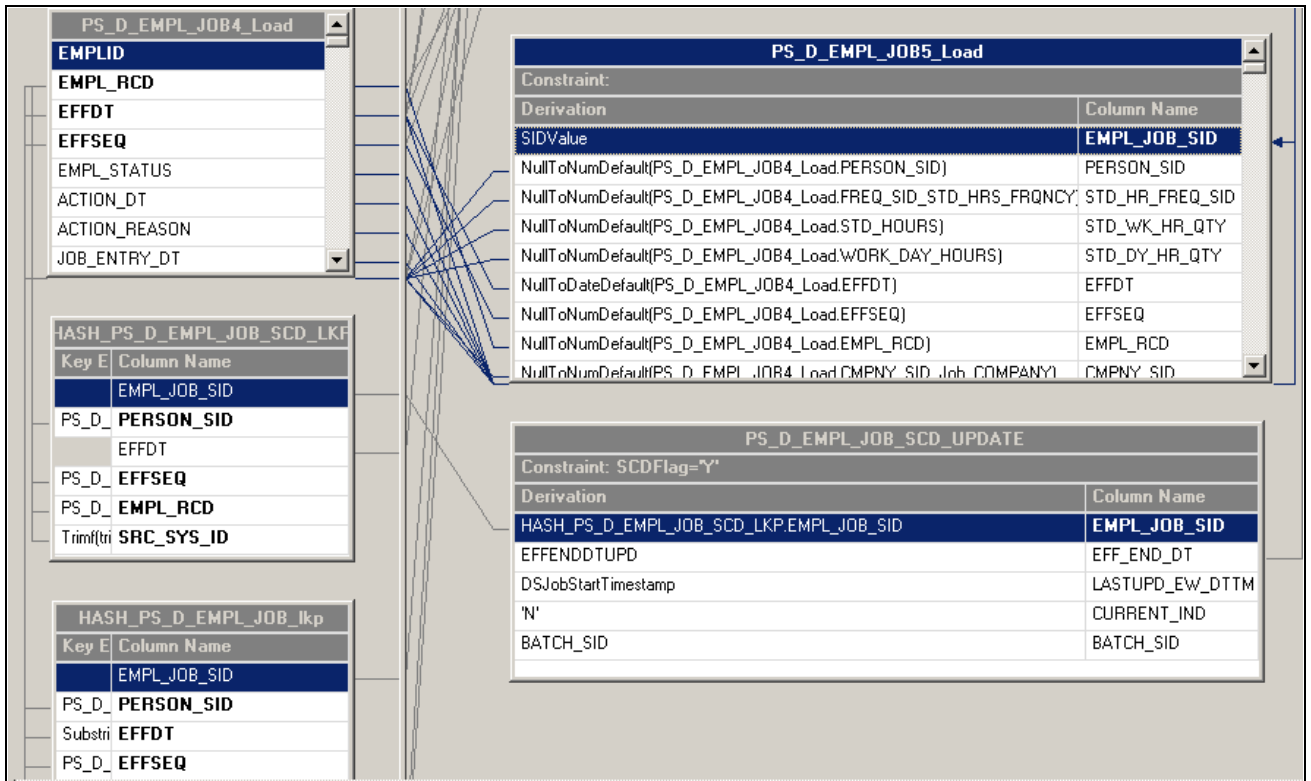
There is only one target DRS stage for a type 1 slowly changing dimension and it uses an *update existing rows* or *insert new rows* logic for its loading strategy.

There are two target DRS stages for the type 2 slowly changing dimension:

- The first target DRS stage uses an *update existing rows only* logic for its loading strategy.

The link with *update existing rows only* has the constraint `SCDFlag='Y'` so that it will update the `EFF_END_DT` and `CURRENT_IND` columns of the old dimension record.

- The second target DRS stage uses an *update existing rows* or *insert new rows* logic for its loading strategy.



Slowly changing dimension and target DRS stage

Target Lookup Stage

If the incoming rows already exist in the dimension table, the type 1 slowly changing dimension lookup stage retrieves the SID value using a lookup on the target dimension that matches the business keys from the incoming row with those of the target table. If the keys match, the existing SID is extracted.

There are two target lookup stages for the type 2 slowly changing dimension:

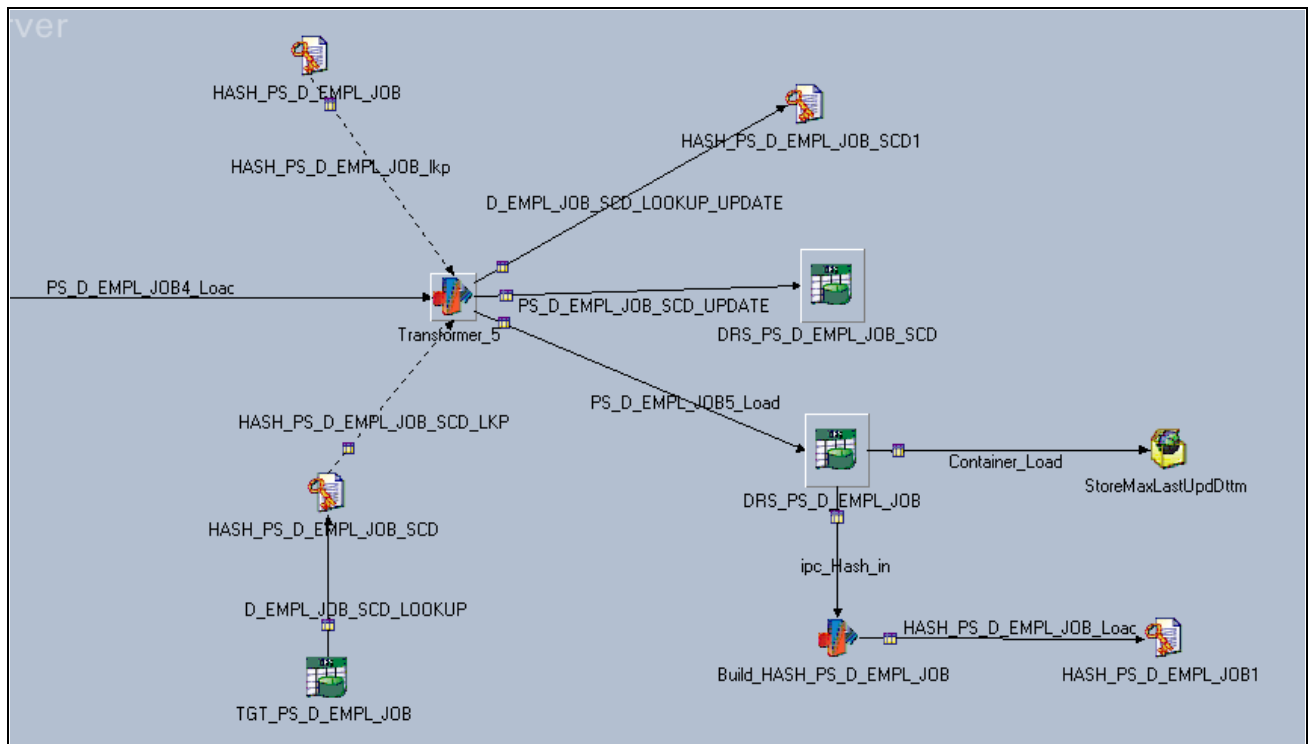
- The first target lookup stage retrieves the latest SID in case the incoming rows are already there in the target dimension table.

The first lookup should have EFFDT as key column and it must be joined with incoming row to get the SID value if the incoming dimensional row is already there in the target table.

- The second target lookup stage indicates whether the incoming row falls under slowly changing dimension logic.

It is loaded in the same job with the latest EFFDT and SID value to compare it with incoming data. If the incoming row falls under slowly changing dimension logic, the SID is retrieved and the EFF_END_DT column is updated for the old dimension row. As soon as we processed the SCD logic, we will update the second lookup in the same job.

Please refer the figure below to get the information about the lookups (HASH_PS_D_EMPL_JOB_SCD1 and HASH_PS_D_EMPL_JOB_SCD) that are used to determine the SCD logic. The following Stage Variables are added to determine the SCD logic: SCDFlag, EFFENDDTUPD and EFFENDDT.



Slowly changing dimension and target lookup stage

Fact Table Jobs and Slowly Changing Dimensions

Most EPM fact table jobs contain dimension lookups. Dimension lookups in fact table jobs use either *hash file* or *dynamic DRS* lookups. EPM fact table jobs with a lookup to a type 1 slowly changing dimension use hash file lookups. This type of lookup does not use the effective date (EFFDT) and performs faster than a dynamic DRS lookup.

EPM fact table jobs with a lookup to a type 2 slowly changing dimension use dynamic DRS lookups. This type of lookup is based on user defined SQL with the following effective date (EFFDT) range criteria:

```
EFF_START_DT<=%DateTimeIn(?) AND EFF_END_DT>=%DateTimeIn(?)
```

Due to the relationship between dimension lookups in a fact table job and the corresponding dimension used in the lookup, if you convert a type 1 slowly changing dimension job to a type 2 slowly changing dimension job, this may impact the related fact table job. Thus, if you want to convert a type 1 slowly changing dimension job to a type 2, you might also have to modify the dimension lookup in the related fact table job. If the related fact table job uses a hash file lookup, you must convert the hash file lookup to a dynamic DRS lookup. However, if the related fact table job uses a dynamic DRS lookup, you do not need to convert the lookup.

See [Chapter 5, "Configuring Slowly Changing Dimensions," Converting a Hash File Lookup to a Dynamic DRS Lookup in the Related Fact Table Job, page 89.](#)

Converting Type 1 Slowly Changing Dimension Jobs to Type 2

This section provides a brief overview of the methods used to convert type 1 slowly changing dimension jobs and discusses how to:

- Convert a type 1 slowly changing dimension job to a type 2 slowly changing dimension job using the effective date.
- Convert a type 1 slowly changing dimension job to a type 2 slowly changing dimension job without the effective date.
- Convert a hash file lookup to a Dynamic DRS lookup in a fact table job.

Overview

You can use one of two methods to convert a type 1 slowly changing dimension job to a type 2 slowly changing dimension job. If the type 1 slowly changing dimension job you are converting contains a source transaction table that uses the effective date (EFFDT) as part of its operational key, use method 1: converting a type 1 slowly changing dimension job using the effective date (described in the following section). If the source transaction table does not use the effective date (EFFDT) as part of its operational key, use method 2: converting a type 1 slowly changing dimension job without the effective date (also described in the following section).

Converting a Hash File Lookup in a Related Fact Table Job

Due to the relationship between dimension lookups in a fact table job and the corresponding dimension used in the lookup, if you convert a type 1 slowly changing dimension job to a type 2 slowly changing dimension job, this may impact the related fact table job. Thus, if you want to convert a type 1 slowly changing dimension job to a type 2, you might also have to modify the dimension lookup in the related fact table job.

If the fact table job related to the modified dimension uses a hash file lookup, you must convert the hash file lookup to a dynamic DRS lookup. However, if the fact table job related to the dimension uses a dynamic DRS lookup, you do not need to convert the lookup. Instructions on how to convert a hash file lookup to a dynamic DRS lookup are described in the following section.

Note. This step is only necessary if the fact table job related to the modified (type 2) dimension uses a hash file lookup!

See [Chapter 5, "Configuring Slowly Changing Dimensions," Converting a Hash File Lookup to a Dynamic DRS Lookup in the Related Fact Table Job, page 89.](#)

Method 1: Converting a Type 1 Slowly Changing Dimension Job Using the Effective Date and Effective Sequence

The following steps are required to convert your type 1 slowly changing dimension jobs to type 2 using the Effective Date and EFFSEQ:

1. Modify the Source Query

2. Modify the Target Hash Lookup Stage
3. Add Lookup stages to identify SCD logic
4. Add the WHERE clause to the newly added Lookup DRS stage
5. Add a new Hash File stage to refresh the Lookup data
6. Add a target DRS stage to update the old dimension record
7. Verify the number of links in the Job design
8. Add stage variables to perform slowly changing dimension logic
9. Modify column expressions to perform slowly changing dimension logic
10. Compile the job

These steps are discussed in further detail below.

Note. The EFFSEQ field is not available in all source tables and should only be used when it exists in the source table. If the EFFSEQ field does not exist in the source table, you should only use the Effective Date (EFFDT) field in the conversion steps.

Step 1: Modifying the Source Query

Perform the following steps to modify the source query:

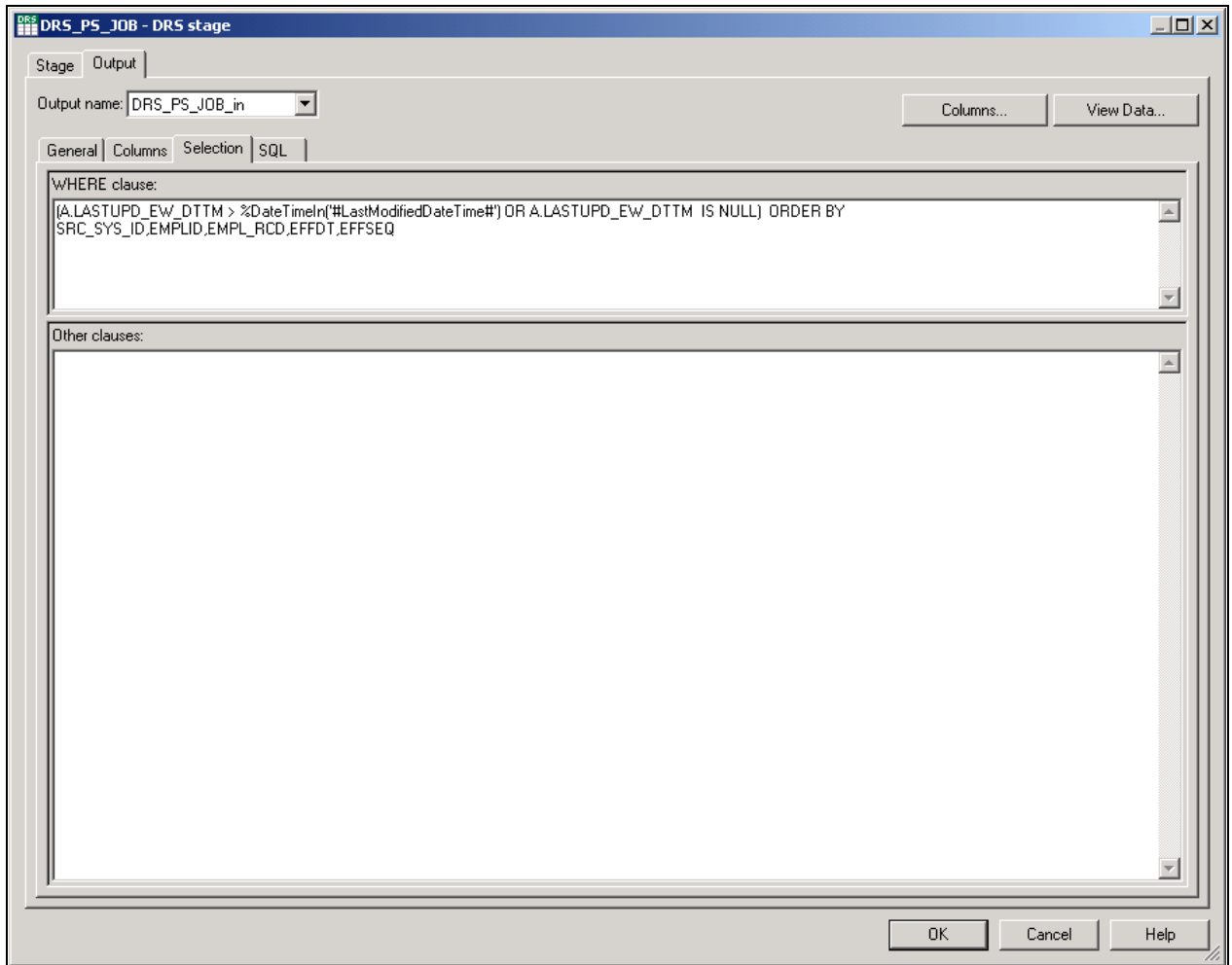
1. In IBM WebSphere DataStage Designer, navigate to the type 1 slowly changing dimension job you would like to convert by expanding the nodes in the left navigation panel; then open the job.
2. Locate the source DRS stage within the job and open it.
3. In the Output tab, select the Selection sub-tab to edit the WHERE clause of your source table.

Note. Most of the Type 1 dimension jobs have the correlating sub-query to get the latest effective dated dimensional record.

- Remove the correlating sub-query.

You should be left with the following SQL statement:

```
(LASTUPD_EW_DTTM > %DateTimeIn('#LastModifiedDate#') OR LASTUPD_EW_DTTM IS NULL) ORDER BY EFFDT, EFFSEQ
```



DRS stage - Output tab

Note. There is an ORDER BY clause to sort the dimensions that are changed over a period of time.

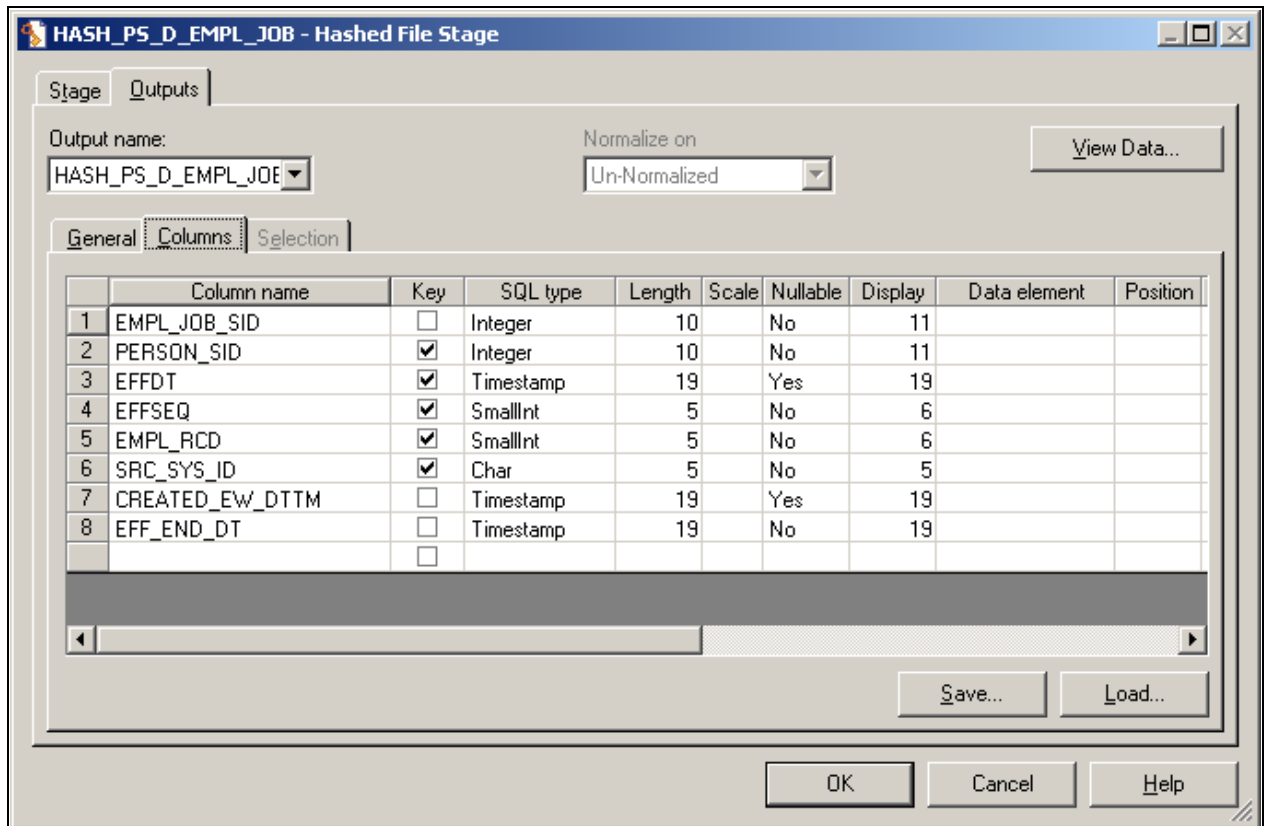
- Click OK.

Step 2: Modifying the Target Hash Lookup Stage

Perform the following steps to modify the target hash lookup stage:

- Locate the target hash lookup stage within the job and open it.
- In the Output tab, select the Columns sub-tab and add EFFDT and EFFSEQ as a key columns, and EFF_END_DT as a non-key column with *Timestamp(19)* as the datatype.

3. Select the General sub-tab and change the *Hash File* and *Hash Stage* names by adding the suffix *_TGT* to them.
4. Locate the transformer stage that defines the lookup transformation between this hash file and the incoming row and open it.
5. In the Output tab, select the Columns sub-tab and map the EFFDT and EFFSEQ columns from the incoming link to the newly added EFFDT and EFFSEQ columns of this Hash File.



Transformer stage - Output tab

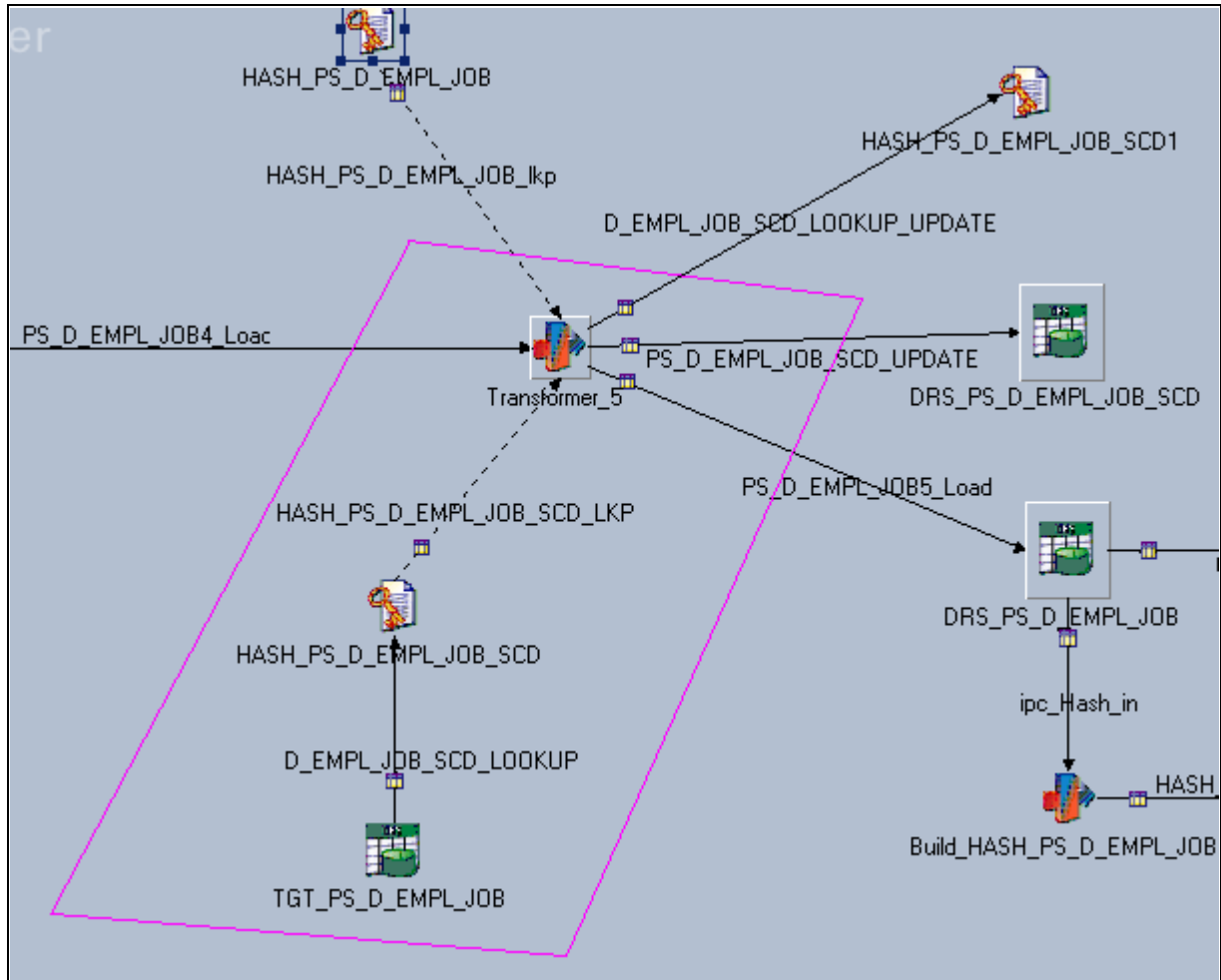
6. Click OK.

Step 3: Adding Lookup Stages to Identify Slowly Changing Dimension Logic

Perform the following steps to add lookup stages:

1. Add new *DRS* and *Hash File* stages to the job, placing them next to the transformer that loads the target table.
2. Link the *DRS* stage to the newly added *Hash File* stage.

3. Link the new Hash File stage to the aforementioned transformer.



Adding lookup stages

Note. The new DRS stage should refer to that target dimension table name with target database connection parameters. The DRS stage should have all the alternate key columns and primary key columns (SID column) in the columns metadata. The alternate key columns should be enabled as key columns in the Input and Output of Hashed File columns metadata.

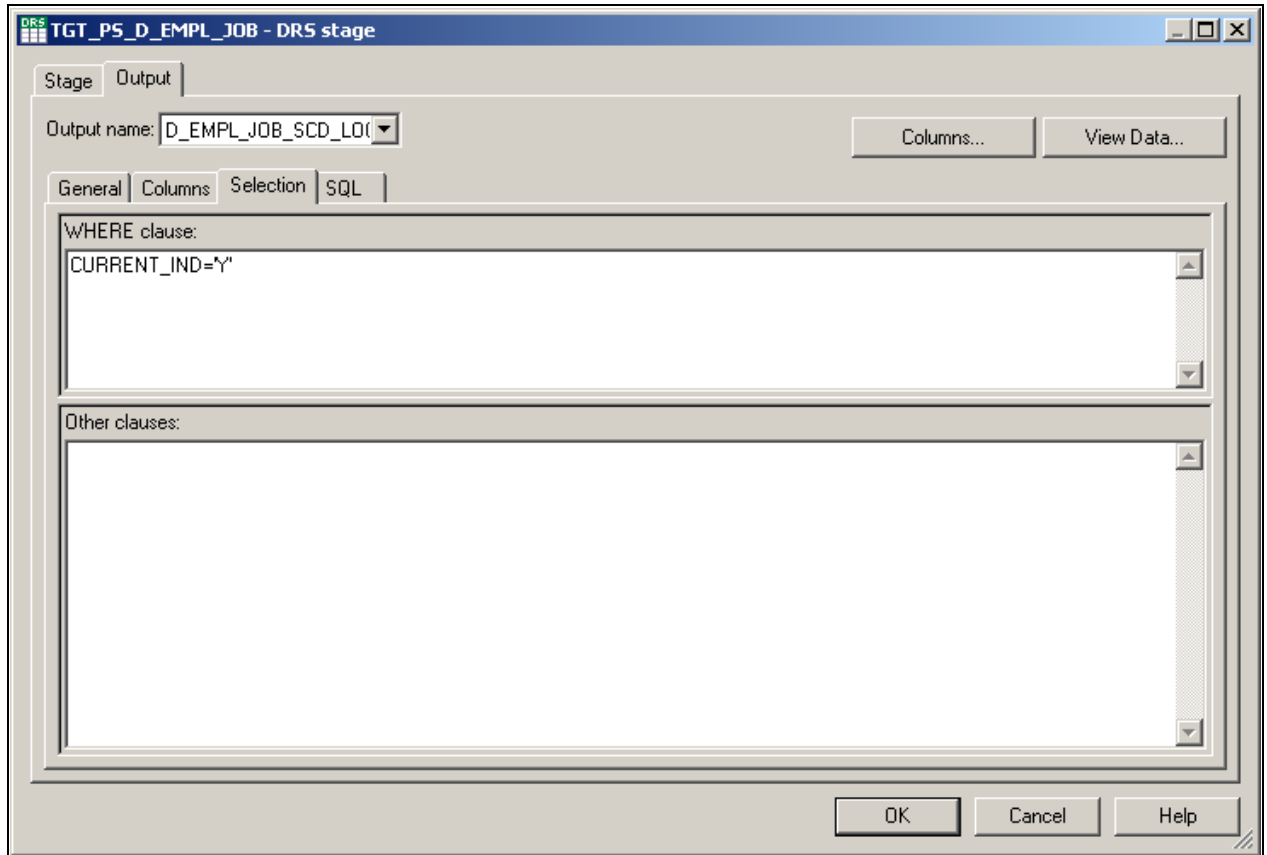
4. Change the *DRS stage*, *DRS link*, *Hash File stage*, and *Hash File link* names by adding the suffix *_SCD* to them.
5. Click OK.

Step 4: Adding a WHERE Clause to the Lookup DRS Stage

Perform the following steps to add a WHERE clause to the lookup DRS stage:

1. Locate the new DRS stage and open it.

- In the Output tab, select the Selection sub-tab to edit the WHERE clause.



DRS stage - Output tab

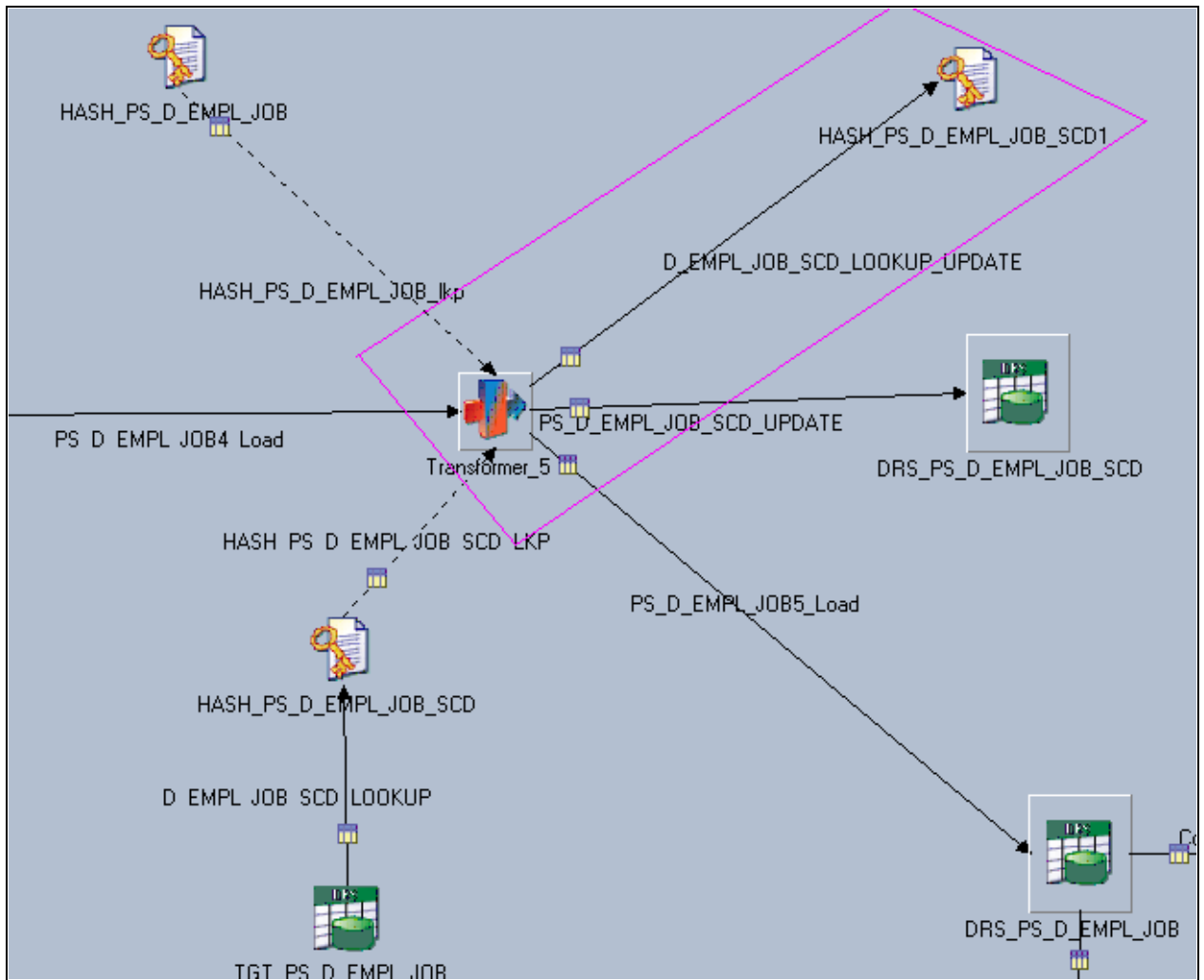
- Add the following WHERE condition to get the most recent SID value:
CURRENT_IND = 'Y'
- Specify the database connection parameters in the General tab of the DRS stage.
- Click OK.

Step 5: Adding a New Hash File Stage to Refresh the Lookup Data

Perform the following steps to add a new hash file stage:

- Add a new *hash file* stage to the job.

2. Link the new hash file stage to the target transformer such that the target transformer loads the has file stage.



Adding a new hash file stage

Note. The new DRS stage should refer to that target dimension table name with target database connection parameters. The DRS stage should have all the alternate key columns and primary key columns (SID column) in the columns metadata. The alternate key columns should be enabled as key columns in the Input and Output of Hashed File columns metadata.

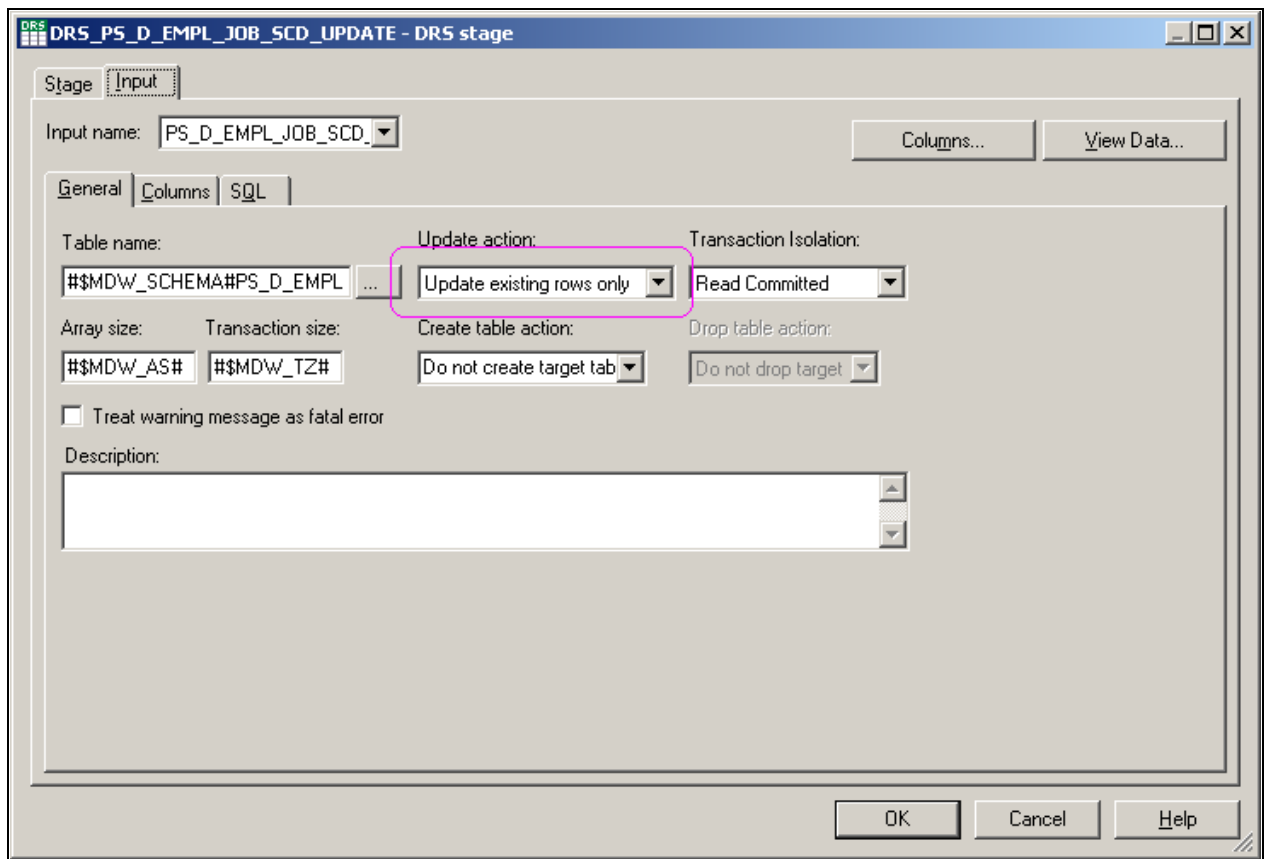
3. Change the hash file name by adding the suffix `_SCD` to it.
4. Click OK.

Step 6: Adding a Target DRS Stage to Update the Old Dimension Record

Perform the following steps to add a target DRS stage:

1. Copy the DRS stage from where the data is loaded and paste it into the same job.
2. Change the new *DRS stage* and *DRS link* names by adding the suffix `_SCD_UPDATE` to them.

3. Link the target transformer stage with the new target DRS stage.
4. Open the new DRS stage and select the Input tab.
5. Select the General sub-tab and change the Update action value to *Update existing rows only*.

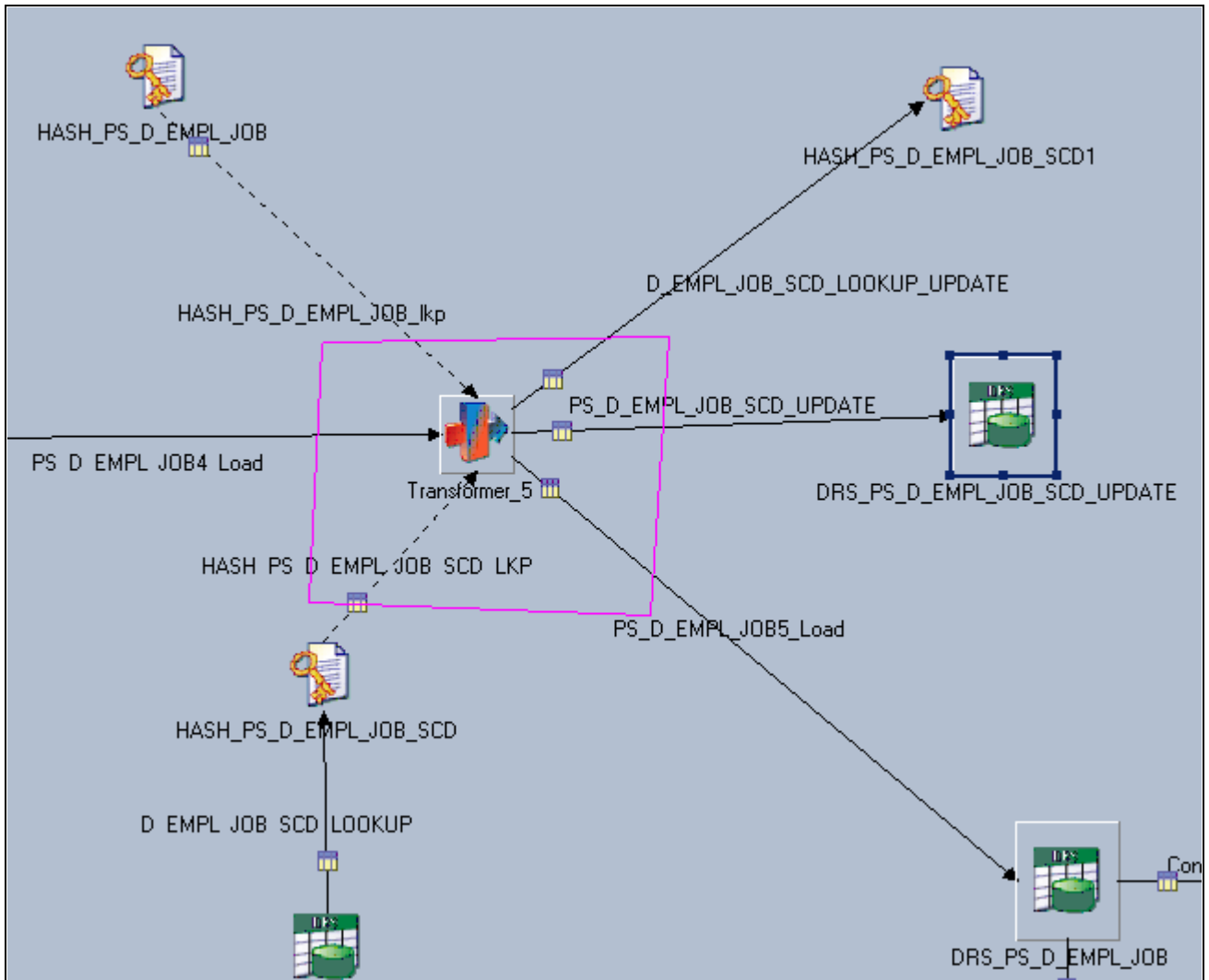


DRS stage - Input tab

6. Click OK.

Step 7: Verifying the Number of Links in the Job Design

Examine the entire job and verify that there are a total of six links in the job. There should be three input links to the target transformer stage and three output links from target transformer. All the links should be connected as follows:



Verifying links

Step 8: Adding Stage Variables to Perform Slowly Changing Dimension Logic

Perform the following steps to add stage variables:

1. Locate the target transformer and open it.
2. Verify the lookup join between input link and the new lookup stage (for example, _TGT) .
3. Link the key columns of the input link to those in the [hash file name]_SCD lookup link.
4. Add a new SCDFlag stage variable to the transformer, using the following expression:

```
If NOT(HASH_PS_D_EMPL_JOB_SCD_LKP.NOTFOUND) AND
(Substrings(PS_D_EMPL_JOB4_Load.EFFDT, 1, 19) <>
Substrings(HASH_PS_D_EMPL_JOB_SCD_LKP.EFFDT, 1, 19) or
(HASH_PS_D_EMPL_JOB_SCD_LKP.EFFSEQ <> PS_D_EMPL_JOB4_Load.EFFSEQ))Then 'Y' Else
'N'
```

5. Add a new EFFENDDTUPD stage variable to the transformer, using the following expression:

```
AddToDate(<INPUT_LINK_NAME>.EFFDFT, 'DD', -1)
```

6. Add a new EFFENDDTUPD stage variable to the transformer, using the following expression:

```
If Len(<TGT_LOOKUP_LINK_NAME>.EFF_END_DT)= 0 Then MaxDate Else  
<TGT_LOOKUP_LINK_NAME>.EFF_END_DT
```

7. Click OK.

The screenshot shows the 'Transformer_5 - Transformer Stage' configuration window. On the left, a list of source columns is shown, including 'EMPLID', 'EMPL_RCD', 'EFFDFT', 'EFFSEQ', 'EMPL_STATUS', 'ACTION_DT', 'ACTION_REASON', and 'JOB_ENTRY_DT'. The 'Stage Variables' table on the right lists the following variables and their derivations:

Derivation	Stage Variable
If HASH_PS_D_EMPL_JOB_TGT.NOTFOUND Then DSJobStartTimestamp Else HASH_PS_D_EM	CreatedDTTM
If @INROWNUM = 1 Then (If \$SID_UNIQUENESS = "W" Then "EPM" Else "D_EMPL_JOB") Else SIDParam	SIDParam
If (HASH_PS_D_EMPL_JOB_TGT.NOTFOUND) then KeyMgtGetNextValueConcurrent(SIDParam).E	SIDValue
If NOT(HASH_PS_D_EMPL_JOB_SCD_LKP.NOTFOUND) AND Substrings(PS_D_EMPL_JOB4_Ld	SCDFlag
AddToDate(PS_D_EMPL_JOB4_Load.EFFDFT, 'DD', -1)	EFFENDDTUPD
If len(HASH_PS_D_EMPL_JOB_TGT.EFF_END_DT)= 0 Then MaxDate Else HASH_PS_D_EMPL	EFFENDDT

Below the stage variables, the 'D_EMPL_JOB_SCD_LOOKUP_UPDATE' table is shown with the following columns and constraints:

Constraint	Derivation	Column Name
	SIDValue	EMPL_JOB_SID
	TRIMF(TRIMB(PS_D_EMPL_JOB4_Load.PERSON_SID))	PERSON_SID
	if len(PS_D_EMPL_JOB4_Load.EFFDFT) = 0 then GetDateDefault("") else PS_D_EMPL_JOB4_Load.E	EFFDFT
	NullToNumDefault(PS_D_EMPL_JOB4_Load.EFFSEQ)	EFFSEQ
	TRIMF(TRIMB(PS_D_EMPL_JOB4_Load.EMPL_RCD))	EMPL_RCD
	Trim(trim(PS_D_EMPL_JOB4_Load.SRC_SYS_ID))	SRC_SYS_ID

At the bottom, two data tables are shown side-by-side. The left table is 'PS_D_EMPL_JOB4_Load' and the right table is 'D_EMPL_JOB_SCD_LOOKUP_UPDATE'. Both tables have columns for Column name, Key, SQL type, Length, Scale, Nullable, Display, and Data element.

Adding stage variables

Step 9: Modifying Column Expressions to Perform Slowly Changing Dimension Logic

Perform the following steps to modify column expressions:

1. Locate the target transformer and open it.
2. Locate the output link that loads the target with *Update existing rows or Insert new rows*.
3. Open the link for editing and modify the expression for the EFF_START_DT column as follows:

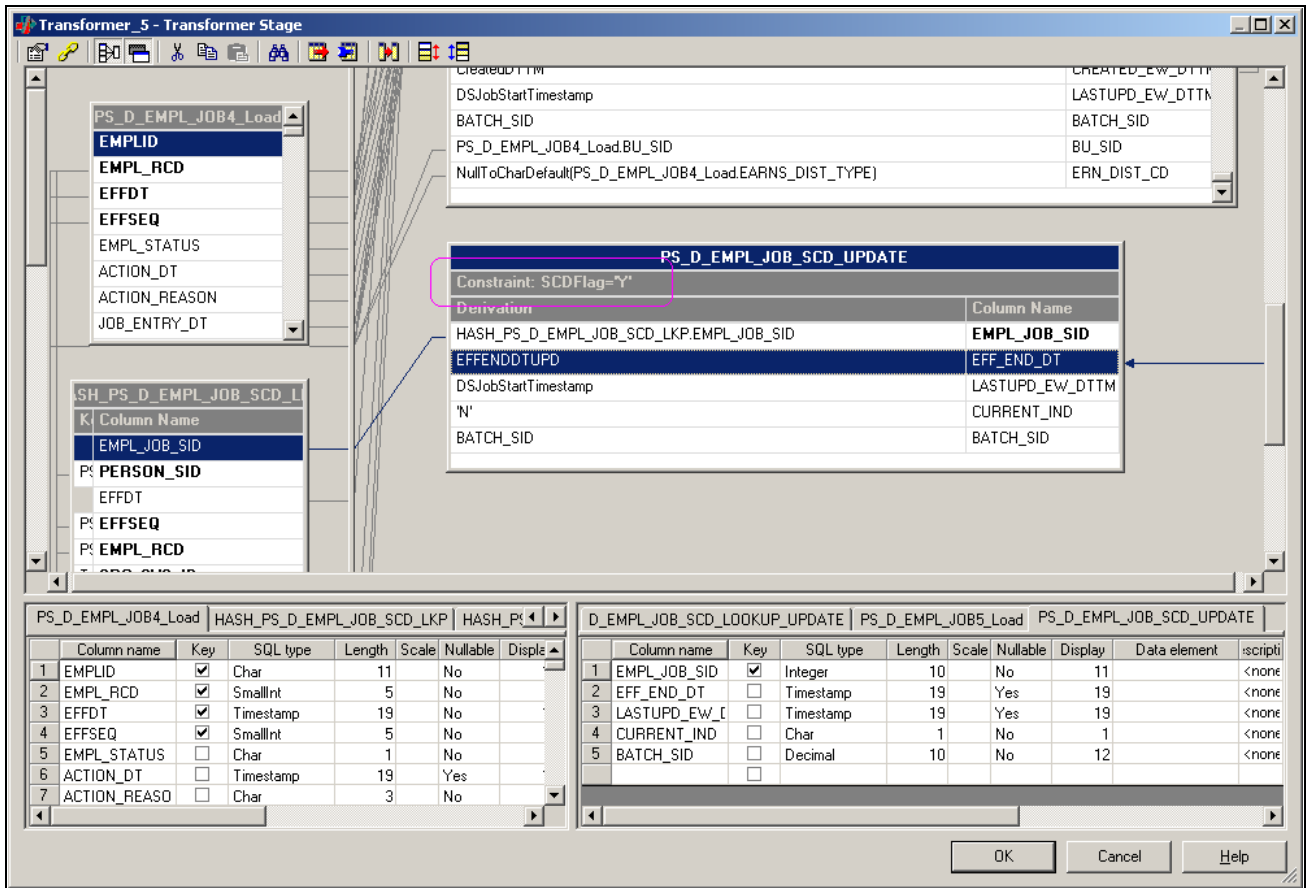
```
EFF_START_DT = <INPUT_LINK_NAME>.EFFDFT
```

4. Modify the expression for the EFF_END_DT column as follows:

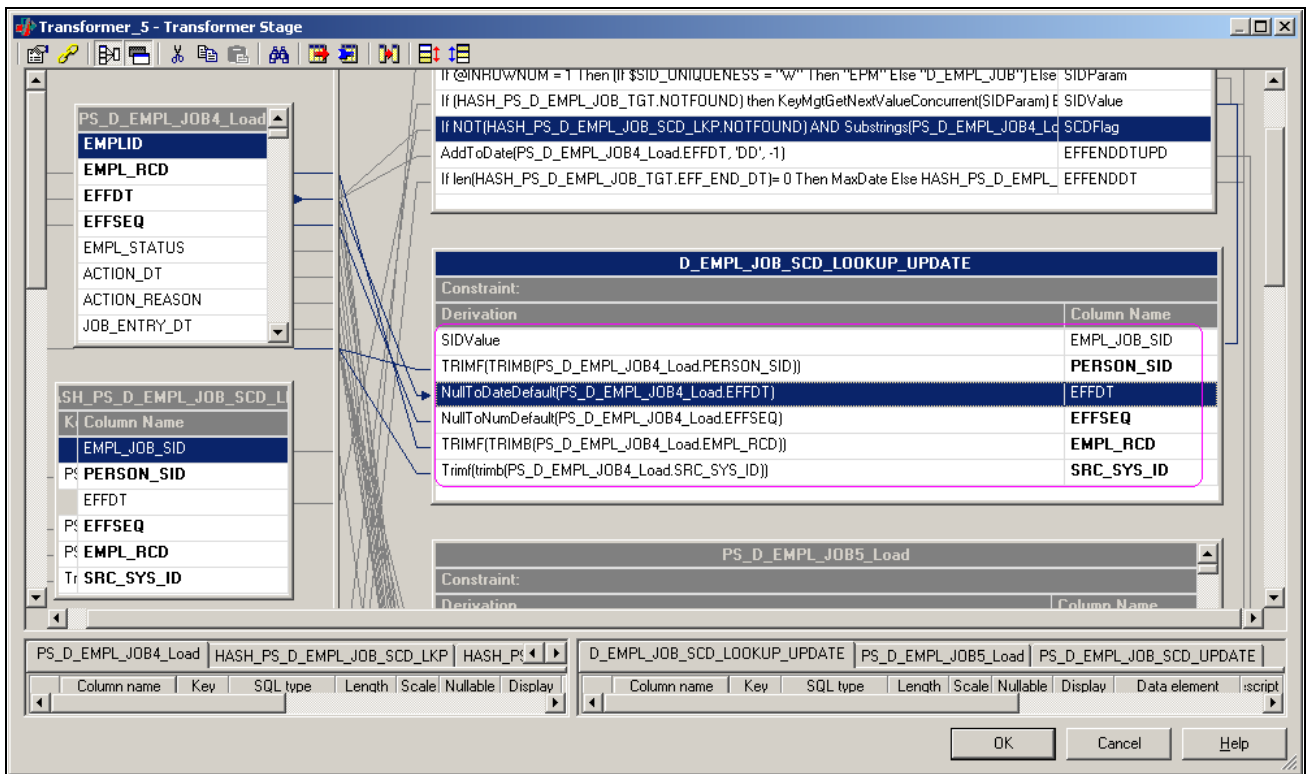
```
EFF_END_DT = EFFENDDT (it is a stage variable)
```

5. Locate the output link that updates the target with *Update existing rows only*.
6. Open the link for editing and delete all columns from the table except the primary key column (SID column), EFF_END_DT, LASTUPD_EW_DTTM, CURRENT_IND and BATCH_SID.
7. Modify the expression for the SID column as follows:
`SID column = <SCD_LOOKUP_LINK_NAME>.<PRIMARY_SID_COLUMN_NAME>`
8. Modify the expression for the EFF_END_DT column as follows:
`EFF_END_DT = EFFENDDTUPD`
9. Modify the expression for the LASTUPD_EW_DTTM column as follows:
`LASTUPD_EW_DTTM = DSJobStartTimestamp`
10. Modify the expression for the CURRENT_IND column as follows:
`CURRENT_IND = 'N'`
11. Modify the expression for the BATCH_SID column as follows:
`BATCH_SID = BATCH_SID`
12. Add the following constraint to the link so that the EFF_END_DT of the old dimension record is updated:
`SCDFlag = 'Y'`
13. Locate the output link that updates the new Hash file (for example, [*hash file name*]_{SCD}).
14. Map the alternate key columns from the input link to alternate keys in the lookup table.

Although the column mapping is one-to-one, there should be proper NULL handling based on the column data type.
15. Modify the expression for the primary key (SID) column so that the expression uses the stage variable *SIDValue*.
16. Click OK.



Modifying column expressions, 1 of 2



Modifying column expressions, 2 of 2

Step 10: Compiling the job

Perform the following steps to compile the job:

1. Select *File, Save* from the menu to save the job.
2. Select *File, Compile* from the menu to compile the job.

If your mapping is correct, the Compilation Status window displays the *Job successfully compiled with no errors* message. If your mapping is incorrect, the Compilation Status window displays an error message.

3. If your job successfully compiles, select Close.

If you job does not compile successfully, you must return to the job and troubleshoot the errors.

Method 2: Converting a Type 1 Slowly Changing Dimension Job Without Using the Effective Date

To convert your type 1 slowly changing dimension jobs to type 2 without using the Effective Date, follow the steps as described in the previous section (*Converting Type 1 Slowly Changing Dimension Jobs Using the Effective Date and EFFSEQ*) noting the variation in steps three, five, and eight, outlined below.

Note. The column you choose to convert your slowly changing dimension jobs is referred to generically in the steps below as: *COLUMN_X*.

Step 3: Adding Lookup Stages to Identify Slowly Changing Dimension Logic

COLUMN_X should not be enabled as a key in the input and output links of the hash file stage (for example, [hash file name]_SCD).

Step 5: Adding a New Hash File Stage to Refresh the Lookup Data

COLUMN_X should not be enabled as a key in the input links of the hash file stage (for example, [hash file name]_SCD).

Step 8: Adding Stage Variables to Perform Slowly Changing Dimension Logic

Add a new SCDFlag stage variable to the transformer, using the following expression:

```
If NOT(<SCD_LOOKUP_LINK_NAME>.NOTFOUND) AND <INPUT_LINK_NAME>.<COLUMN_X>
<> <SCD_LOOKUP_LINK_NAME>.<COLUMN_X> Then 'Y' Else 'N'
```

Converting a Hash File Lookup to a Dynamic DRS Lookup in the Related Fact Table Job

The following steps are required to convert a hash file lookup to a dynamic DRS lookup:

1. In IBM WebSphere DataStage Designer, navigate to the fact job containing the hash file lookup by expanding the nodes in the left navigation panel; then open the job.
2. Locate the hash file lookup within the job.
3. If the hash file is populated in a separate job (for example, initial hash loading job or the job which loads the dimension), replace the existing hash file stage with the DRS Stage.

If your hash file is populated by the DRS stage in the same job, delete the DRS stage (including the link) and replace the hash file stage with a DRS stage.

4. Open the DRS stage for editing.
5. In the Stage tab, select the General sub-tab and specify the database connection parameters.
6. In the Output tab, select the General sub-tab and specify the corresponding table name (the table name should always include the schema name as its prefix).

Specify the appropriate job parameter for array size and change query type to *User-defined SQL query*.

7. Select the Columns sub-tab and specify parameters for EFF_START_DT and EFF_END_DT.

8. In the SQL, User-Defined sub-tabs, input your user-defined query.

For example,

```
SELECT
INSTITUTION_SID,
LTRIM(RTRIM(INSTITUTION_CD)),
%DateTimeOut(EFF_START_DT),
%DateTimeOut(EFF_END_DT),
LTRIM(RTRIM(SRC_SYS_ID)),
LTRIM(RTRIM(INSTITUTION_SD)),
LTRIM(RTRIM(INSTITUTION_LD))
FROM #MDW_SCHEMA#PS_D_INSTITUTION
WHERE
INSTITUTION_CD=?
AND EFF_START_DT<= %DateTimeIn(?)
AND EFF_END_DT >= %DateTimeIn(?)
AND SRC_SYS_ID =?
```

Note. All the columns specified in the selection criteria of the SQL user defined query should match the columns defined in the Columns sub-tab; the same is true of column order. Also, those columns defined as keys must be used in the WHERE clause and their order must match the order defined in the Columns sub-tab.

Chapter 6

Implementing Audit Jobs

This chapter provides an overview of audit jobs and audit job implementation, and discusses:

- Creating Audit Triggers
- Creating Trigger Scripts

Note. Currently audit jobs are only available for the FMS Warehouse product.

Understanding Audit Jobs

Some PeopleSoft Enterprise Campus Solutions source table records do not have a date time stamp field. When source table records lack a date time stamp, a cyclical redundancy check (CRC) must be performed during the ETL process to determine new or changed records. Unlike the traditional date time lookup process which targets the DTTM column for each record, the CRC process reads the entire record for each record in the source table and generates a CRC value to compare against the target warehouse record. Because the CRC process is so extensive it can create performance issues with the OWS jobs that use the logic. However, if you want faster processing you can use PeopleSoft delivered audit jobs as an alternative.

Audit jobs have the same functionality as the OWS jobs but employ a different incremental load strategy that allow them to process faster than their counterparts. Audit jobs use audit records created in the source table. Audit records are associated with a date time stamp field and are created when you enter a new source record or modify an existing source record. The audit jobs join the parent record and audit record to make use of the date time stamp field and determine new or changed source records. Since the audit jobs only process modified or added records, they process much faster than the OWS jobs which use the CRC logic.

Audit tables include only the key fields of the base table plus three additional audit-specific fields:

- **AUDIT_OPRID:** Identifies the user who causes the system to trigger the audits, either by performing an add, change, or delete to an audited field.
- **AUDIT_STAMP:** Identifies the date and time that the audit is triggered.
- **AUDIT_ACTN:** Indicates the type of action the system audited. Possible action values include:
 - A – Row inserted.
 - D – Row deleted.
 - K – Row updated, snapshot before update.
 - N – Row updated, snapshot after update.

For example the audit job *J_STAGE_PS_CA_STATUS_TBL_AUDIT* contains the base table *PS_CA_STATUS_TBL* and the audit table *PS_AUDIT_CA_STATUS_TBL*. The following table compares the fields in the base table with the fields in the audit table:

Base Table (PS_CA_STATUS_TBL)	Audit Table (PS_AUDIT_CA_STATUS_TBL)
	AUDIT_OPRID
	AUDIT_STAMP
	AUDIT_ACTN
SETID (Key field)	SETID
CA_STATUS (Key field)	CA_STATUS
SRC_SYS_ID	
CA_PROC_STATUS	
DEFAULT_FLAG	
DESCR	

When bringing your source data into EPM using ETL, you can use either the audit jobs or the existing OWS jobs that use the CRC logic. You can also make use of both; for example, if you discover that only five jobs process slowly due to the CRC logic, you can implement the audit jobs for these five jobs only and continue to run the remaining OWS jobs with CRC logic. If you want to use audit jobs, you must import the appropriate .dsx file and project, build the records related to the audit jobs, and define audit triggers for the audit records you build. These steps are documented in detail below.

You can access all PeopleSoft delivered audit jobs from the following IBM WebSphere DataStage location:

- FMS Warehouse - FMS_E, OWS, Base, AuditJobs, Server.
- CS Warehouse - CS_E, OWS, Base, AuditJobs, Server.

Understanding Audit Job Implementation

You must perform the following implementation tasks prior to running the audit jobs:

1. Import the audit .dsx file to your IBM WebSphere DataStage project.
 - For the FMS Warehouse:

Customers using FSCM 8.8 should import the *WFN_OWS_AUDIT_E.dsx* file.

Customers using FSCM 8.9 should import the *WFN_OWS_AUDIT_E_FSCM89_EPM9_IU.dsx* file.

Customers using FSCM 9.0 should import the *WFN_OWS_AUDIT_E_FSCM9_EPM9_IU.dsx* file.

Customers using FSCM 9.1 should import the *WFN_OWS_AUDIT.dsx* file.
 - For the Campus Solutions Warehouse:

Customers using Campus Solutions 8.9 should import the *WCS_OWS_AUDIT_E.dsx* file.

Customers using Campus Solutions 9.0 should import the *WCS_OWS_AUDIT_E_CS9_EPM9_IU.dsx* file.
2. In the PeopleSoft source transaction database, use PeopleSoft Application Designer to import the audit project.
 - For the FMS Warehouse:

Customers using FSCM 8.8 should import the *AUDIT_FMS* project.

Customers using FSCM 8.9 should import the *AUDIT_FMS_89* project.

Customers using FSCM 9.0 should import the *AUDIT_FMS_9* project.

The audit project contains all the related audit records for the source tables.
 - For the Campus Solutions Warehouse:

Customers using Campus Solutions 8.9 should import the *AUDIT_CS_89* project.

Customers using Campus Solutions 9.0 should import the *AUDIT_CS_9* project.
3. In the source transaction system database, use Application Designer to build the audit tables for which you want to use with the audit jobs.
4. Define audit triggers for each audit record you build using the Audit Trigger page.

You must create an audit trigger for each audit record. A trigger is a database level object that the system initiates based on a specified event occurring on a table. The audit trigger ensures that auditing is triggered whenever there is a change to the source record.

You must choose the record to hold the auditing data, the audit record, using the Audit Triggers page.
5. Create and run trigger scripts.

Note. Before you enable and run audit jobs, you must first load your source data into EPM using the delivered OWS jobs that use CRC logic. This ensures that all subsequent changes in the source records are captured in the audit record.

Creating Audit Triggers and Running Trigger Scripts

This section discusses how to:

- Create audit triggers.
- Create and run trigger scripts.

Pages Used to Create Audit Triggers and Run Trigger Scripts

<i>Page Name</i>	<i>Definition Name</i>	<i>Navigation</i>	<i>Usage</i>
Audit Triggers	TRIGAUDPNL	PeopleTools, Utilities, Audit, Update Database Level Auditing, Audit Triggers	Create audit triggers and trigger statements.
Run Audtrgs (Run Audit Triggers)	RUN_AUDTRGS	PeopleTools, Utilities, Audit, Perform Database Level Audit, Run Audtrgs (Run Audit Triggers)	Run trigger scripts.

Creating Audit Triggers

Access the Audit Triggers page (PeopleTools, Utilities, Audit, Update Database Level Auditing, Audit Triggers).

Audit Triggers page

Record (Table) Name Displays the base table you selected and intend to associate with an audit table.

Audit Record Name Select an audit table you want to associate with the base table.
You must specify an audit table before you can create a trigger.

Trigger Name Displays the name of the audit trigger.
The system automatically names the audit trigger using the following naming convention: *[base record name]_TR*.

Create Trigger Statement Displays the trigger statement (code).
The code is automatically populated when you click Generate Code.
You can edit the script as needed.

Generate Code Click this button to generate the SQL that creates the trigger.
Clicking this button also populates the Create Trigger Statement field with the trigger statement (code).

Audit Options

Add Select this option if you want the audit table to track newly added records.

Change Select this option if you want the audit table to track changed records.

Delete Select this option if you want the audit table to track deleted records.

After you create your trigger statements, you can create and run the trigger script against the database to physically create the triggers. See the following section for more information.

Creating and Running the Trigger Script

Access the Run Audtrgs (Run Audit Triggers) page (PeopleTools, Utilities, Audit, Perform Database Level Audit, Run Audtrgs (Run Audit Triggers)).

Run Audtrgs (Run Audit Triggers) page

- Create All Triggers** Select this check box if you want all the triggers you defined in the Trigger Definition (PSTRIGGERDEFN) table included in the script.
- The application engine writes a 'create trigger' statement to a file for every row in the Trigger Definition table.
- Create Triggers On** Specify the table for which the trigger statement should be created.
- Run** Click to run the trigger script.
- This process writes a 'create trigger' statement to a file for the triggers you specified. The system writes the file to the location that is determined by the run location of the process. If it is run on the server, the file is created in the PS_SRVRDIR directory. If it is run on a Windows workstation, the file is created in the directory that the %TEMP% environment variable specifies. The file name is *TRGCODEX.SQL*, where X represents a digit that is determined by the number of files by the same name that already exist in the output directory.

Note. After you create the SQL script, use a SQL utility to run the script against the database. If you encounter an issue while running SQL due to OPRID, then OPRID (V_AUDIT_OPRID) generated from the script can be hardcoded with the username used for sign on.

Chapter 7

Managing Source System Deletes and Archiving

This chapter provides an overview of source system deletes and source system archives, and describes how to:

- Enable the Source-Delete Diagnostic Feature.
- Implement the source-delete diagnostic feature in staging jobs.
- Adjust the source-delete diagnostic option after implementation.
- Implement the source-archiving diagnostic feature.

Understanding Source System Deletes

Depending on your business practices, it can sometimes become necessary to delete records in your PeopleSoft source transaction system. Because EPM delivers ETL jobs that perform incremental loads, a deleted source record would not be reflected in the Campus Solutions Warehouse, which causes the two systems to be unsynchronized. When the two systems are unsynchronized, the Campus Solutions Warehouse may produce reports with incorrect results. Therefore, to ensure synchronicity between the Campus Solution Warehouse and your source system, it is important to track deleted source records in the Campus Solutions Warehouse.

EPM provides staging jobs that can identify which source records have been physically deleted from your transaction system and flag those records in the OWS table. When the flagged records pass to the MDW table during the fact load job, they are physically deleted from the Campus Solutions Warehouse schema. This process varies slightly depending on whether it is executed in a CRC staging job or a standard (date time stamp dependent) staging job. Both processes are discussed in more detail below.

If you wish to retain deleted source records in the Campus Solutions Warehouse, you can disable the *source-delete diagnostic* feature in the staging jobs. This process is discussed in more detail below.

Note. MDW dimension load jobs do not use the source-delete diagnostic feature because doing so can cause linked facts to become orphaned and generate incorrect results in reports.

Identifying Source Record Deletes with CRC Staging Jobs

Unlike standard staging jobs, which only target the Date Time (DTTM) column for each record during an incremental load, a CRC staging job reads the entire record for every record in a source table, and then generates a CRC hashed file with a CRC value assigned for each record. The source-delete diagnostic uses the CRC hashed file to identify deleted source records.

The first time you implement the source-delete diagnostic for the Campus Solutions Warehouse, you run a setup job which populates a Delete hashed file with all records from the corresponding CRC hashed file. During this process, every record in the Delete hashed file is assigned a default value of 'E' for the Delete Flag (DELFLAG) column, where 'E' represents a delete.

When you run the CRC staging job all source records are extracted and written to the Delete hashed file. During this process, these source records are assigned a new value of 'EX' for the Delete Flag column, where 'EX' represents a valid record. Hence, when an incoming source record matches the existing record in the Delete hashed file, the existing record is overwritten and the delete flag value changes from 'E' to 'EX.' If a source record has been deleted, there is no matching record in the Delete hashed file and that record retains a Delete Flag value equal to 'E,' or delete.

The process later updates Delete Flag values from 'E' to 'DEL' and writes the flagged records to the OWS table. In the OWS table the flagged record is assigned a value of 'D' for the Data Origin (DATA_ORIGIN) column, thereby officially flagging the record for deletion.

Once records are flagged in the OWS table, a MDW fact load job passes those flagged records to a target DRS stage that contains a *delete* target update action. The flagged records are then deleted from the Campus Solutions Warehouse schema.

Example

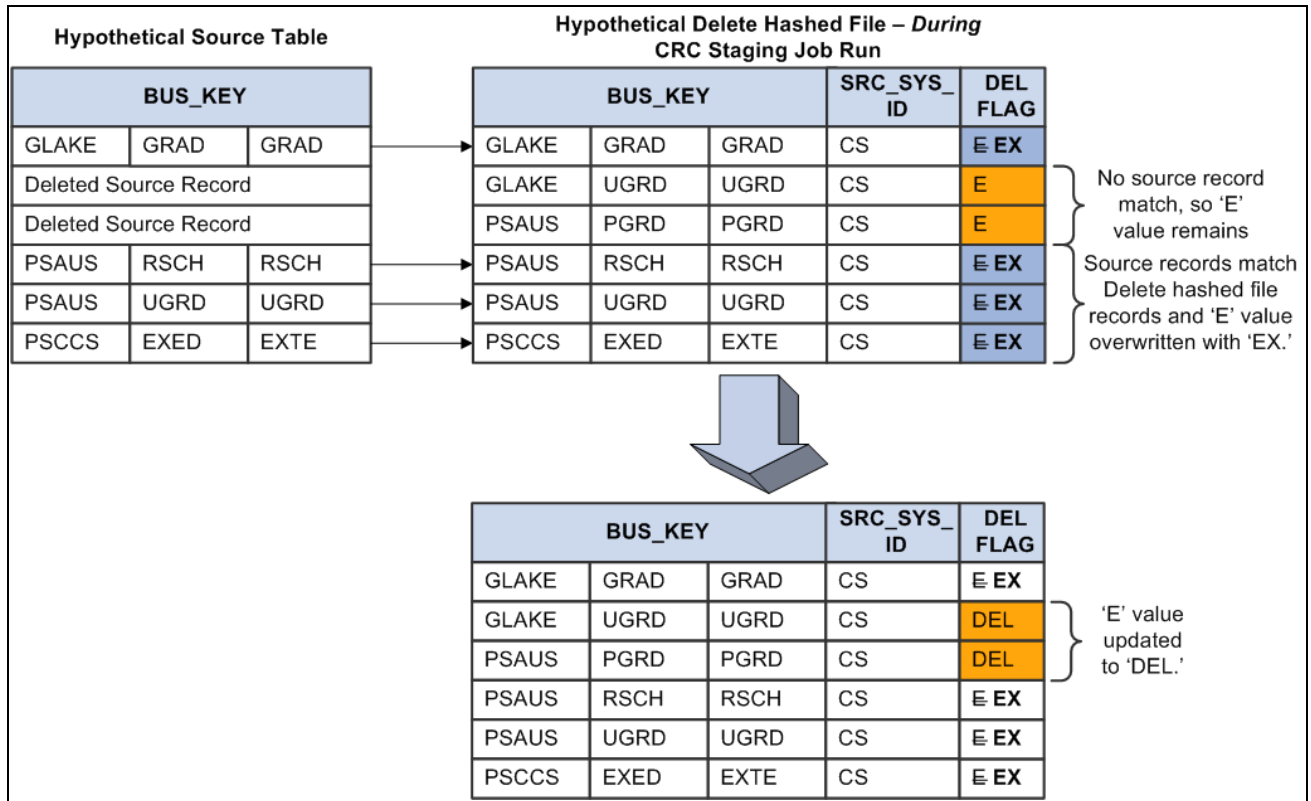
The following graphic depicts a hypothetical Delete hashed file after it has been populated with all the records from its corresponding CRC hashed file. Note that during this process, every record in the Delete hashed file is assigned a default value of 'E' for the Delete Flag (DELFLAG) column, where 'E' represents a delete.

BUS_KEY			SRC_SYS_ID	DELFLAG
GLAKE	GRAD	GRAD	CS	E
GLAKE	UGRD	UGRD	CS	E
PSAUS	PGRD	PGRD	CS	E
PSAUS	RSCH	RSCH	CS	E
PSAUS	UGRD	UGRD	CS	E
PSCCS	EXED	EXTE	CS	E

} Defaulted to 'E'

Hypothetical Delete hashed file before the CRC staging job is run

Assume that two source records are later deleted from the source table; in this case *GLAKE* and *PSAUS*. When you run the CRC staging job, all available source records are extracted and written to the Delete hashed file.

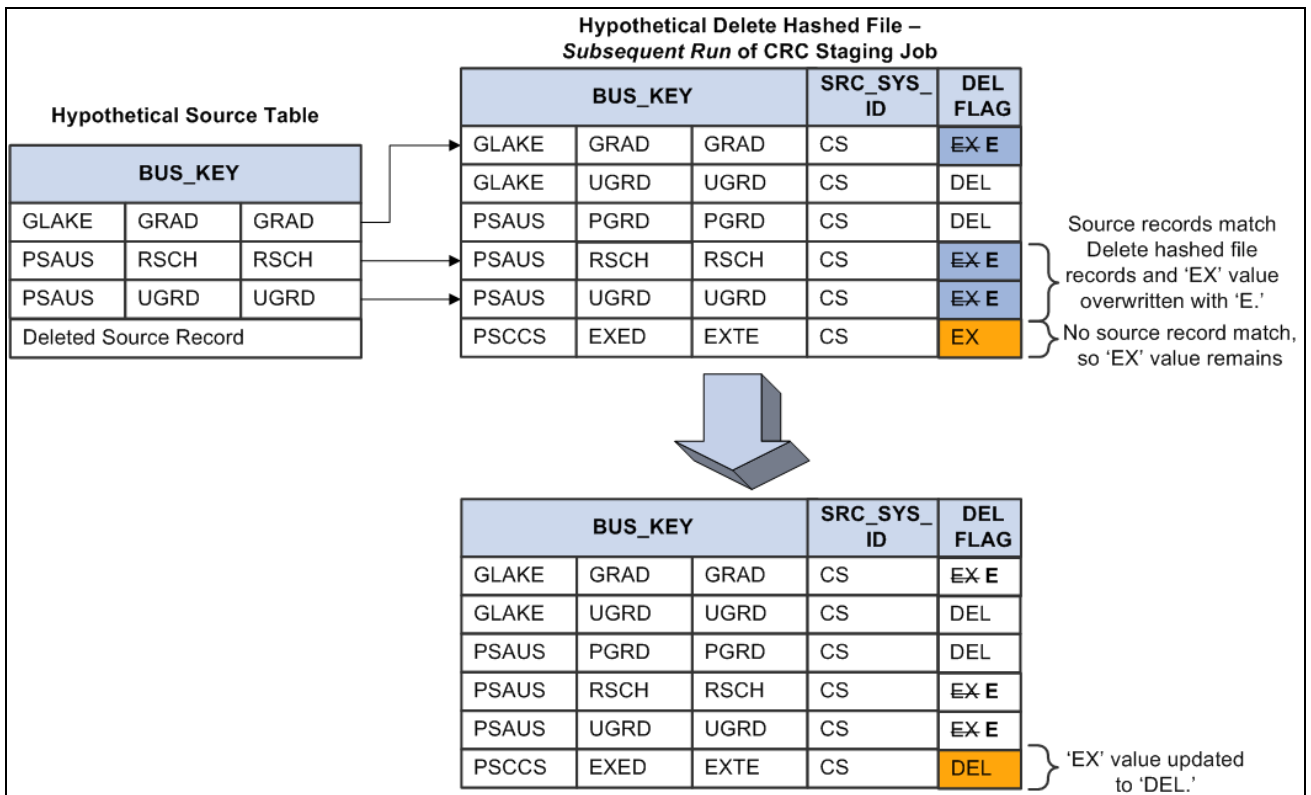


Hypothetical Delete hashed file during the CRC staging job run

Note that all Delete hashed file records that match the source records are overwritten, and the delete flag value for those records changes from 'E' to 'EX' (shown in blue). There are no matching records for *GLAKE* and *PSAUS* in the Delete hashed file since those records were deleted in the source. Hence, those records retain a Delete Flag value equal to 'E,' or delete, in the Delete hashed file (shown in red). The process then updates Delete Flag values from 'E' to 'DEL.'

Bear in mind that all *valid* records in the Delete hashed file now retain a Delete Flag value equal to 'EX.'

When you run the CRC staging job again at a later date, all available source records are extracted and written to the Delete hashed file, as expected. But this time when matching Delete hashed file records are overwritten, the Delete Flag value is changed from 'EX' to 'E' (shown in blue in the diagram below). This time, non-matching records retain a Delete Flag value equal to 'EX,' or delete (as shown in red in the diagram below).



Hypothetical Delete hashed file during a subsequent run of the CRC staging job

Again the process updates Delete Flag values from 'EX' to 'DEL.'

This time all valid records in the Delete hashed file now retain a Delete Flag value equal to 'E' and the next time you run the CRC staging job the process repeats, swapping 'E' and 'EX' to denote valid records. This cycle continues with each run of the staging job.

Identifying Source Record Deletes with Standard Staging Jobs

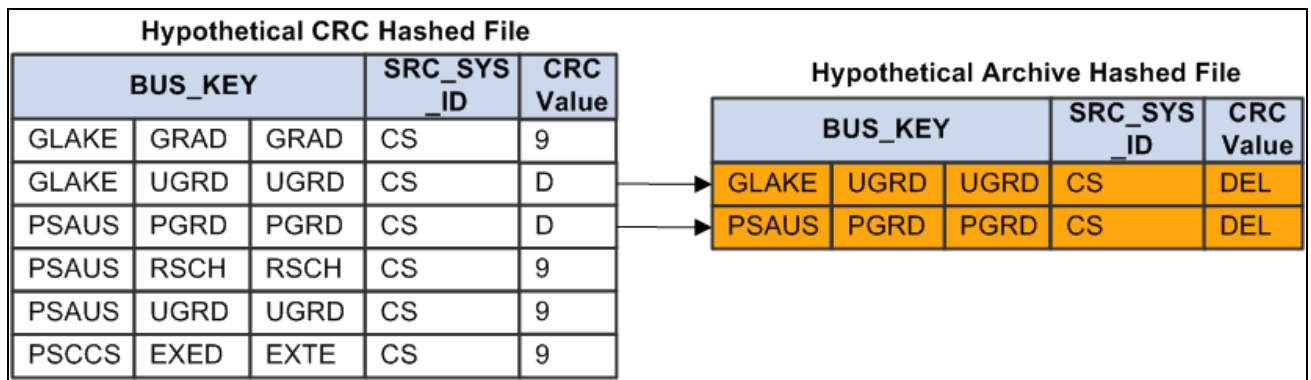
Like CRC staging jobs, standard staging jobs rely on hashed files to identify deleted source records. Standard staging jobs contain a hashed file that stores the row count of the source record. Thus, each time you run a staging job, it extracts the most recent row count from the source record and stores it in the target table. The row count in the target table is then compared against the row count in the hashed file. If the row count in the hashed file is less than the row count in the target table, a source record has been deleted and a separate *delete* job is called to deleted the flagged records from the Campus Solutions Warehouse schema.

Note. Delete jobs are suffixed with *_DEL*.

Understanding Source System Archives

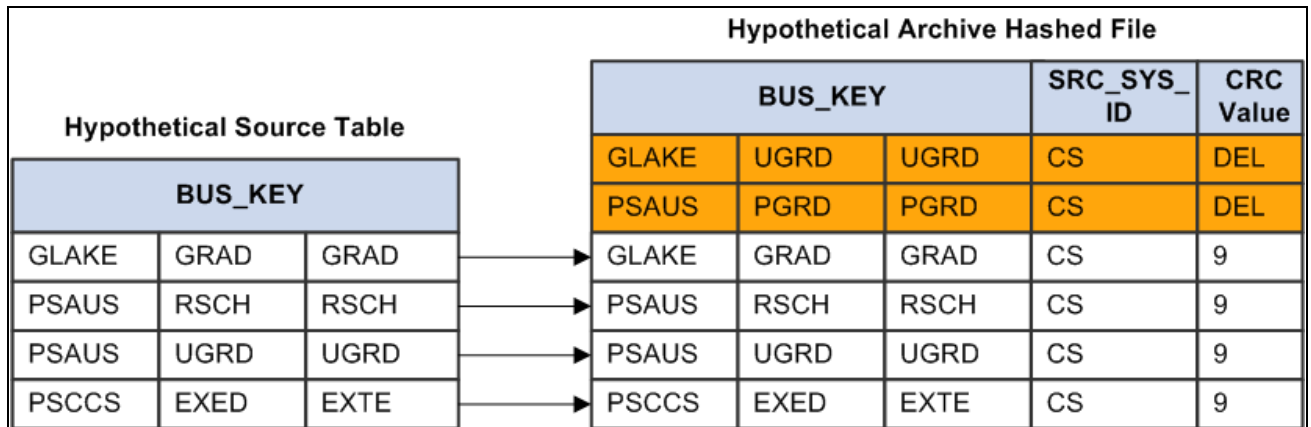
As discussed in the Source System Delete overview above, when you delete a record from your source system without that change being reflected in the Campus Solutions Warehouse, the source system and the warehouse become unsynchronized and reporting errors can occur. Because archiving records in the source system moves records from the active transactional system to an archive system apart from the source, archiving source records can produce reporting errors similar to source system deletes. Thus, it is also important to identify archived source records in the Campus Solutions Warehouse. To identify archived source records, PeopleSoft provides you with the *source-archiving diagnostic* feature, which you should always run immediately after archiving a record in your source system.

To identify archived source records, the source-archiving diagnostic first examines the appropriate CRC hashed file and locates records that have the value *D* for the CRC value column. The source-archiving diagnostic then copies the flagged records to an intermediate Archive hashed file.



Source-archiving diagnostic, step 1

Next, all source table records are copied to the same intermediate Archive hashed file.



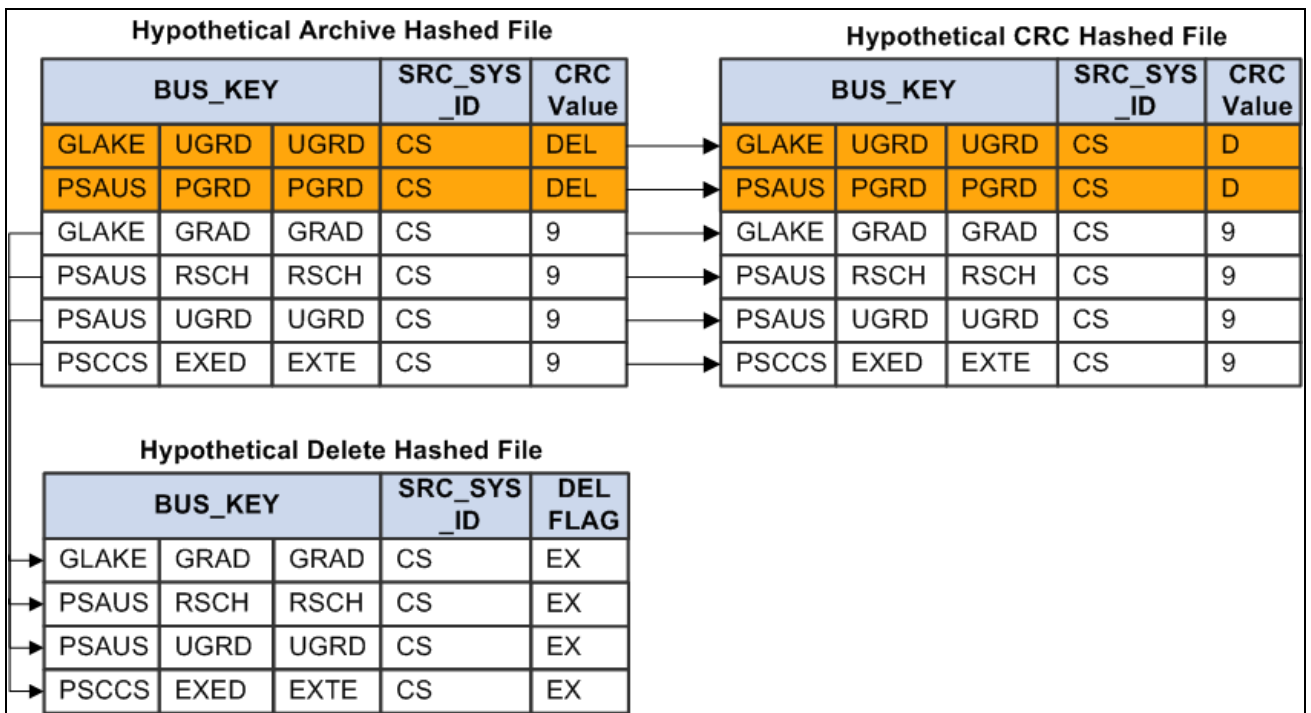
Source-archiving diagnostic, step 2

In the third step, the source-archiving diagnostic clears all data from the CRC and Delete hashed files.

Hypothetical CRC and Delete Hashed Files				
BUS_KEY			SRC_SYS_ID	CRC Value
BUS_KEY			SRC_SYS_ID	DEL FLAG

Source-archiving diagnostic, step 3

In the final step, the source-archiving diagnostic copies data from the intermediate Archive hashed file back to the CRC and Delete hashed files. You should note, however, that this process does not copy records marked as deleted to the Delete hashed file; only the CRC hashed file contains those records.



Source-archiving diagnostic, step 4

The source table and delete hashed file are synchronized upon completion of the source-archiving diagnostic process.

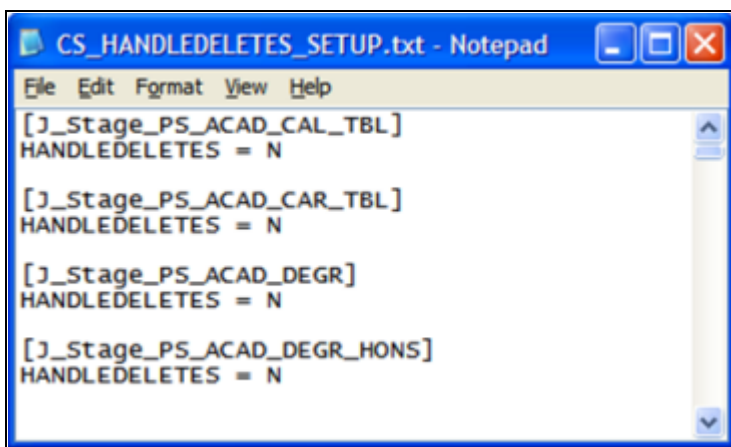
The archive hashed file permanently stores the archived records so you can maintain a history of those records.

Warning! You should never delete source records at the same time you are archiving source records, as doing so can cause the source-delete and source-archiving diagnostic processes to produce incorrect results. Furthermore, you must always run the source-archiving diagnostic after archiving source data and before deleting source data.

Enabling the Source-Delete Diagnostic Feature

The source-delete diagnostic feature is designed as optional, and you can activate it or deactivate it for each staging job in the `CS_HANDLEDELETES_SETUP.txt` parameter file.

Each staging job in the parameter file has the `HANDLEDELETES` value defaulted to 'N' which means the source-delete diagnostic logic will not execute when you run the staging jobs. You can set the `HANDLEDELETES` value equal to 'Y' if you want your staging job to identify source deletes.



CS_HANDLEDELETES_SETUP environment parameter file

The `HANDLEDELETES` value is read every time the staging job runs and will execute according to your setting.

Note. The `CS_HANDLEDELETES_SETUP.txt` parameter file must be placed in the same directory as the `$PARAM_FILE_DIR` environment variable.

If a staging job is not present in the parameter file, the `HANDLEDELETES` value is treated as 'N' at runtime.

Implementing the Source-Delete Diagnostic Feature in Staging Jobs

The following steps are required to implement and configure the source-delete diagnostic feature in the EPM Campus Solutions Warehouse staging jobs.

These steps must be completed even if you do not plan to use the source-diagnostic feature.

1. In IBM WebSphere DataStage Administrator, select the Projects tab.
2. Select the appropriate project and click the Properties button.

3. On the General tab, click the Environment button.
4. Select User-Defined.
5. Add the following environment parameter to the dsparams file:

<i>Name</i>	<i>Type</i>	<i>Prompt</i>	<i>Value</i>
<i>HASHED_FILE_DIRECTORY</i>	<i>String</i>	<i>Hashed Files Directory Path</i>	<i>[User defined path]</i>

The hash file directory path parameter stores all the delete hashed files required to use the source-delete diagnostic feature.

6. Copy the delivered parameter file *CS_HANDLEDELETES_SETUP.txt* to the same directory as the *\$PARAM_FILE_DIR* environment variable.
7. Open the *CS_HANDLEDELETES_SETUP.txt* file and set the *HANDLEDELETES* value equal to 'Y' if you want the staging job to identify source deletes or 'N' if you do not want the staging job to identify source deletes.
8. In IBM WebSphere DataStage Designer, navigate to the job *JC_DeleteStrat_SequentialRun* under the *ReusableJobs_Parallel*, *DeleteStrategy* nodes.
9. Click the Run button on the toolbar, set the *SourceCategory* job parameter equal to *CS*, and click the Run button.

This job performs the initial setup required to implement the source-delete diagnostic feature, and only needs to be run once.

If you have activated the source-delete diagnostic feature for staging jobs, this job populates delete hashed files with data from their related CRC hashed files.

If you have deactivated the source-delete diagnostic feature for staging jobs, this job stores the job name and the job Start Date time in the hashed file. This data is used when a staging job is aborted.

Adjusting the Source-Delete Diagnostic Option after Implementation

After you have enabled the source-delete diagnostic feature in the *CS_HANDLEDELETES_SETUP.txt* parameter file and run the staging jobs accordingly, you may later wish to disable the feature for a particular staging job or set of staging jobs. Or you may wish to enable the feature for a particular staging job that previously did not have the feature enabled. In both cases you can modify the *HANDLEDELETES* value for the staging jobs in the *CS_HANDLEDELETES_SETUP.txt* parameter file (as discussed in the preceding sections of this chapter).

However, once you have modified the *HANDLEDELETES* value for a staging job, you must run a specific initial load job to update that staging job. The initial load job required to populate the Delete hashed file depends on whether you have changed the setting for a CRC staging job or standard staging job.

Adjusting the Source-Delete Diagnostic Option for CRC Staging Jobs

The following sections discuss the steps required when you enable or disable the source-delete diagnostic feature for CRC staging jobs.

Enabling the Source-Delete Diagnostic Feature

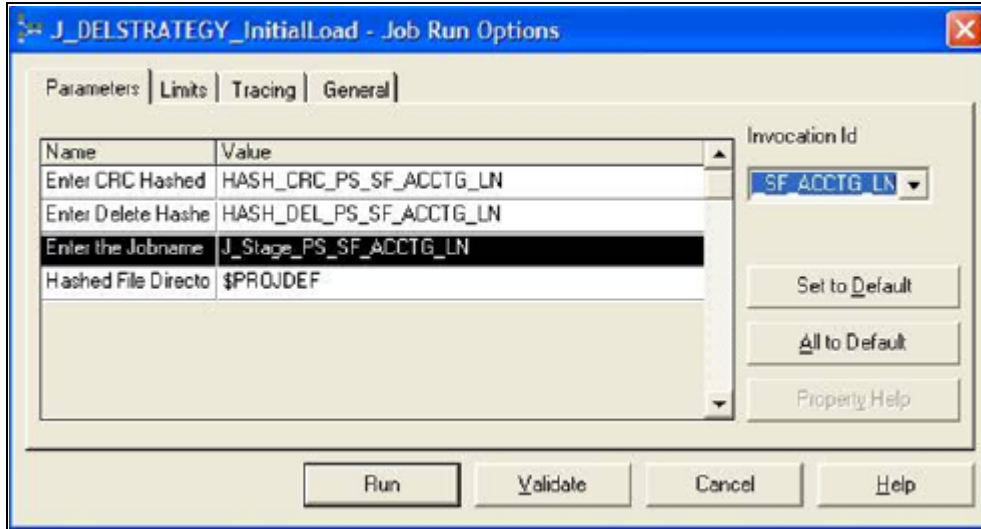
Perform the following steps if you changed the HANDLEDELETES value from 'N' to 'Y' (*enabling* the source-delete diagnostic feature):

1. In IBM WebSphere DataStage Designer, navigate to the job *J_DELSTRATEGY_InitialLoad* under the ReusableJobs_Parallel, DeleteStrategy nodes.
2. Click the Run button on the toolbar and complete the Parameters tab as follows:

Name	Value
[CRC hashed file name]	[CRC hashed file name] Note. You enter the name of the CRC hashed file associated with the CRC staging job being modified. For example, HASH_CRC_PS_SF_ACCTG_LN.
[Delete hashed file name]	[Delete hashed file name] Note. You enter the name of the Delete hashed file associated with the CRC staging job being modified. For example, HASH_DEL_PS_SF_ACCTG_LN.
[Staging job name]	[Staging job name] Note. You enter the name of the CRC staging job being modified. For example, J_Stage_PS_SF_ACCTG_LN.

Note. These fields are case sensitive.

- Enter the name of the staging job being modified for the Invocation ID.



Job run parameters for the Invocation ID

Note. This field is case sensitive.

- Click the Run button.

Note. This job should only be run once.

Disabling the Source-Delete Diagnostic Feature

Perform the following steps if you changed the HANDLEDELETES value from 'Y' to 'N' (*disabling* the source-delete diagnostic feature):

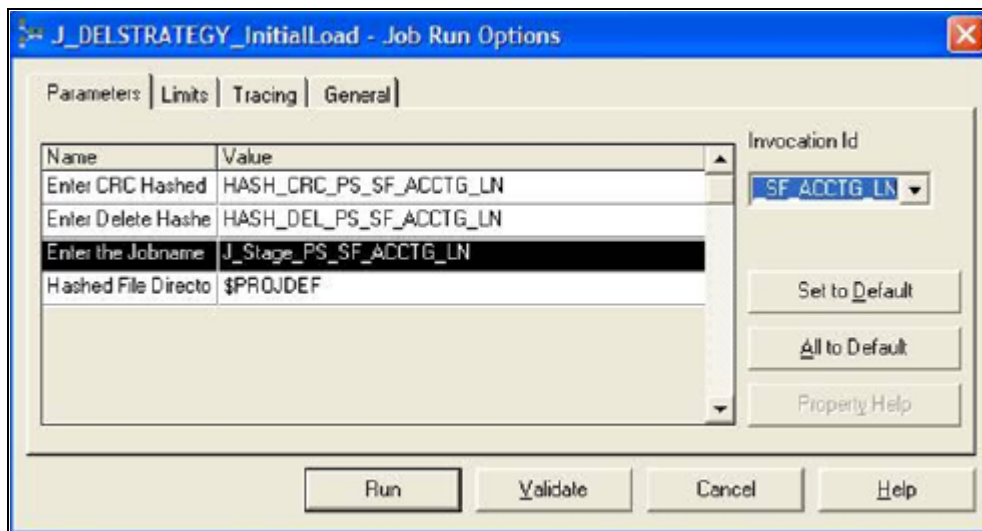
- In IBM WebSphere DataStage Designer, navigate to the job *J_DELSTRATEGY_InitialLoad_NonDeletes* under the ReusableJobs_Parallel, DeleteStrategy nodes.

2. Click the Run button on the toolbar and complete the Parameters tab as follows:

Name	Value
[CRC hashed file name]	[CRC hashed file name] Note. You enter the name of the CRC hashed file associated with the CRC staging job being modified. For example, HASH_CRC_PS_SF_ACCTG_LN.
[Delete hashed file name]	[Delete hashed file name] Note. You enter the name of the Delete hashed file associated with the CRC staging job being modified. For example, HASH_DEL_PS_SF_ACCTG_LN.
[Staging job name]	[Staging job name] Note. You enter the name of the CRC staging job being modified. For example, J_Stage_PS_SF_ACCTG_LN.

Note. These fields are case sensitive.

3. Enter the name of the staging job being modified for the Invocation ID.



Job run parameters for the Invocation ID

Note. This field is case sensitive.

4. Click the Run button.

Note. This job should only be run once.

Adjusting the Source-Delete Diagnostic Option for Standard Staging Jobs

If you have enabled the source-delete diagnostic option for a standard staging job, you must run an initial load job to update the related *Date Time hashed file*. However, if you have disabled the source-delete diagnostic for a standard staging job you need not run an initial load job; no updates to the staging job is necessary because the Date Time hashed file will continue to be used to identify updates and inserts, as normal.

Enabling the Source-Delete Diagnostic Feature

Perform the following steps if you changed the HANDLEDELETES value from 'N' to 'Y' (*enabling* the source-delete diagnostic feature):

1. In IBM WebSphere DataStage Designer, navigate to the job [*Staging_Job_Name*]*_PRV_DEL* under the Handle_DTTM_Previous_Deletes, CS nodes.

For example, if you are modifying the J_Stage_PS_CREDIT_HISTORY staging job, the job you select under the CS node will be *J_Stage_PS_CREDIT_HISTORY_PRV_DEL*.

2. Click the Run button on the toolbar, accept the default run options, and click the Run button.

Note. This job should only be run once.

Implementing the Source-Archiving Diagnostic Feature

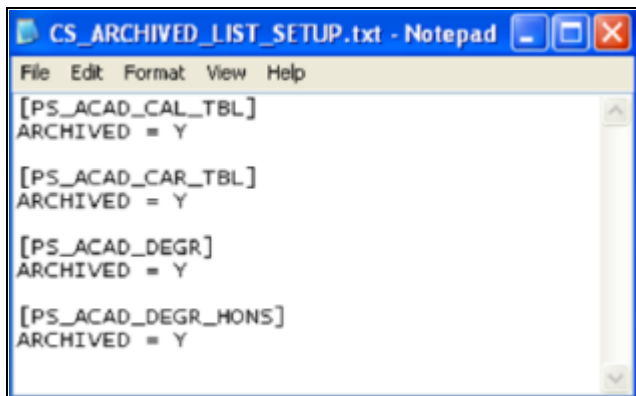
Perform the following three steps to implement the source-archiving diagnostic feature:

1. Copy and paste the parameter file, *CS_ARCHIVED_LIST_SETUP.txt*, to the same directory you use for the \$PARAM_FILE_DIR environment variable.

Note. PeopleSoft delivers the *CS_ARCHIVED_LIST_SETUP.txt* parameter file.

2. Set the input parameters for each staging table in the `CS_ARCHIVED_LIST_SETUP.txt` file.

This text file contains a list of all source tables related to EPM staging and a *Y* or *N* archived setting for each table. The archived setting indicates whether records in the table have been archived.



CS_ARCHIVED_LIST_SETUP input parameter file

The default archived setting value is 'Y' but you can change the setting to 'N' if a table does not have archived records. This reflects the fact that archiving may occur for all tables or only for a set of tables in your source system. Only tables marked 'Y' are processed during the source-archiving diagnostic process.

3. In IBM WebSphere DataStage Designer, navigate to the `JC_Handle_Source_Archiving` job control under the ReusableJobs_Parallel, Archiving nodes.

Click the Run button on the toolbar, set the SourceCategory job parameter equal to `CS`, and click the Run button.

This job control calls the individual jobs necessary to perform the source-archiving diagnostic process.

Chapter 8

Implementing Currency Conversion for Multiple Currencies

To set up and run the currency conversion process, use the Currency Conversion Schema Definition (CCU_SCHEMA_DEFN) and Currency Conversion Definition (CCU_CONV_DEFN) components, as well as the ETL currency conversion process.

This section provides overviews of currency conversion, the methodology behind the currency conversion process, and currency conversion rules, and discusses how to:

- Set up currency conversion.
- Set up the schema definition.
- Set up currency conversion rules.
- Set up the conversion schema rule.
- Run the ETL currency conversion process.

Understanding Currency Conversion

Companies spanning national boundaries often experience problems handling multiple currencies, as well as problems providing a unified view of their enterprise data. This can occur because transactional data can be recorded in any currency in which the company transacts business.

To overcome this disparity, information in MDW fact tables is kept in more than one currency: the source currency and up to two additional currencies. To process data in more than one currency, you must potentially convert transactions from one currency to another currency. You do this using the PeopleSoft MDW Currency Conversion utility, an ETL process that you run after you populate the MDW.

For data analysis and simulation in EPM, for proper engine processing to occur, you must convert monetary amounts to a single currency for each business unit. For reporting in EPM, you must convert the amounts to a single currency, sometimes regardless of the business unit, to provide a unified view of enterprise data. For these reasons, the MDW currency conversion utility has been created.

Note. The MDW currency conversion process discussed in this chapter populates MDW tables only. Do not confuse this process with the currency conversion application engine process that populates the OWE and is used with the Analytic Applications.

MDW Table Structure Used to Support Currency Conversion

Every source amount that is stored in an MDW fact table must have a corresponding source currency code field in that fact table. Additionally, because each fact table can carry the source currency code and up to two additional currencies, each fact table can have up to two additional currency codes. Therefore, each source amount in a fact table has a corresponding reporting1 amount and a reporting2 amount. Because all currency amount columns must have a corresponding currency code, each reporting1 and reporting2 amount must have a respective reporting currency code1 and reporting currency code2. Following is an example of currency and currency code fields in a fact table showing the transaction currency (AMOUNT column), its currency code (AMT_CD) and reporting1 and 2 amount columns, and their respective currency code columns:

AMOUNT	AMT_CD	RPT_AMT1	RPT_AMT1_CD	RPT_AMT2	RPT_AMT2_CD
100	USD	517	FFR	79	EUR

Base amount and *base currency code* fields can also exist in the MDW fact table. However, they exist only if the source table has the corresponding base amount and base currency code fields, and only if a currency conversion process was run on that database. The base amount and transaction amount are considered source amounts.

The ETL process that populates the MDW fact tables does not populate the reporting1 and reporting2 amount fields, nor their corresponding reporting currency code fields. The reporting amounts (RPT_AMT1 and RPT_AMT2 in the fact table example) are populated as a result of the ETL currency conversion process that you run after populating the MDW. Their values do not exist in the source system. The report amounts can represent amounts in any currency that you choose.

Assuming the source currency amount is <ABC>_AMT, where "<ABC>" represents the name of the field, this table lists the currency field naming convention for MDW fact tables:

Field Type	Field Name
Source Amount	<ABC>_AMT
Source Currency Code	CURRENCY_CD
Base Amount	<ABC>_BCE_AMT
Base Currency Code	CURRENCY_BCE_CD
Reporting1 Amount	<ABC>_R1_AMT
Reporting1 Currency Code	CURRENCY_R1_CD
Reporting2 Amount	<ABC>_R2_AMT
Reporting2 Currency Code	CURRENCY_R2_CD

In summary, MDW table structures use the following rules to support currency conversion in the MDW:

- Each source amount that is stored in an MDW table must have a corresponding source currency code field in the MDW fact table.

If multiple source amounts from the same source tables are stored in the MDW tables, they may share the same source currency code field.

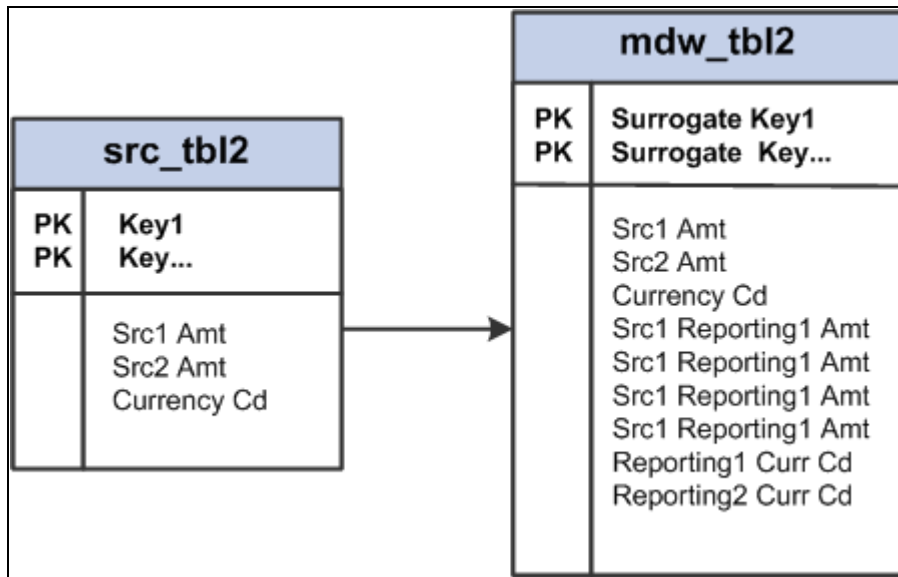
- Base amount and base currency code fields can exist in the MDW table only if the source table has the corresponding base amount and base currency code fields.

That is, base amount and base currency code fields are source database fields. If multiple base amounts from the same source tables are stored in the MDW tables, they may share the same base currency code field.

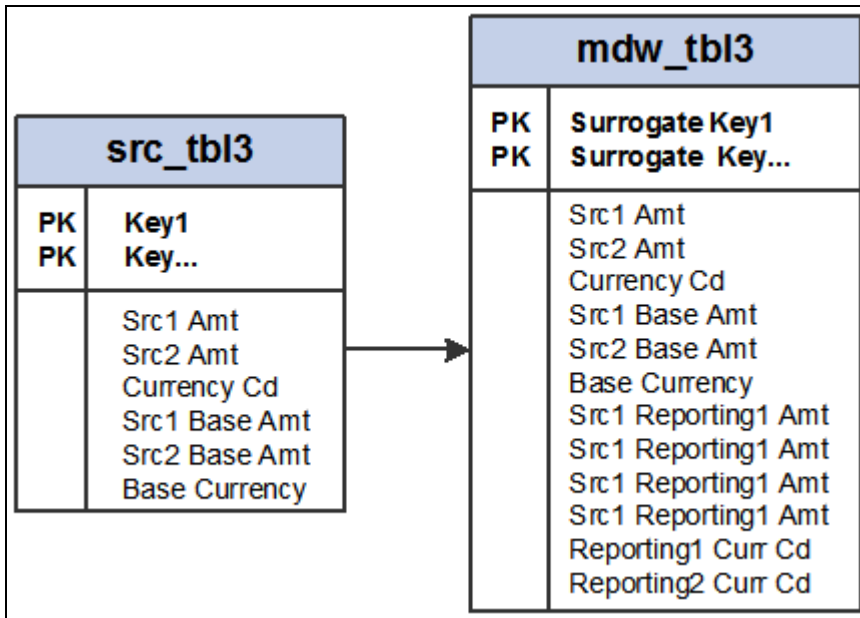
- Source amounts in MDW tables must have corresponding reporting1 amount and reporting2 amount fields, if that amount requires currency conversion.
- MDW tables must have only one reporting1 currency code and reporting2 currency code fields that serve as the currency codes for all reporting1 amounts and reporting2 amounts in that MDW table.

Note. The target columns for the MDW Currency Conversion utility are the reporting1 and reporting2 columns. The columns are named as reporting amount or currency code because the converted amount and currency code are usually used for trend or analysis reporting in the MDW.

The following examples describe the rules for MDW fact table structures:



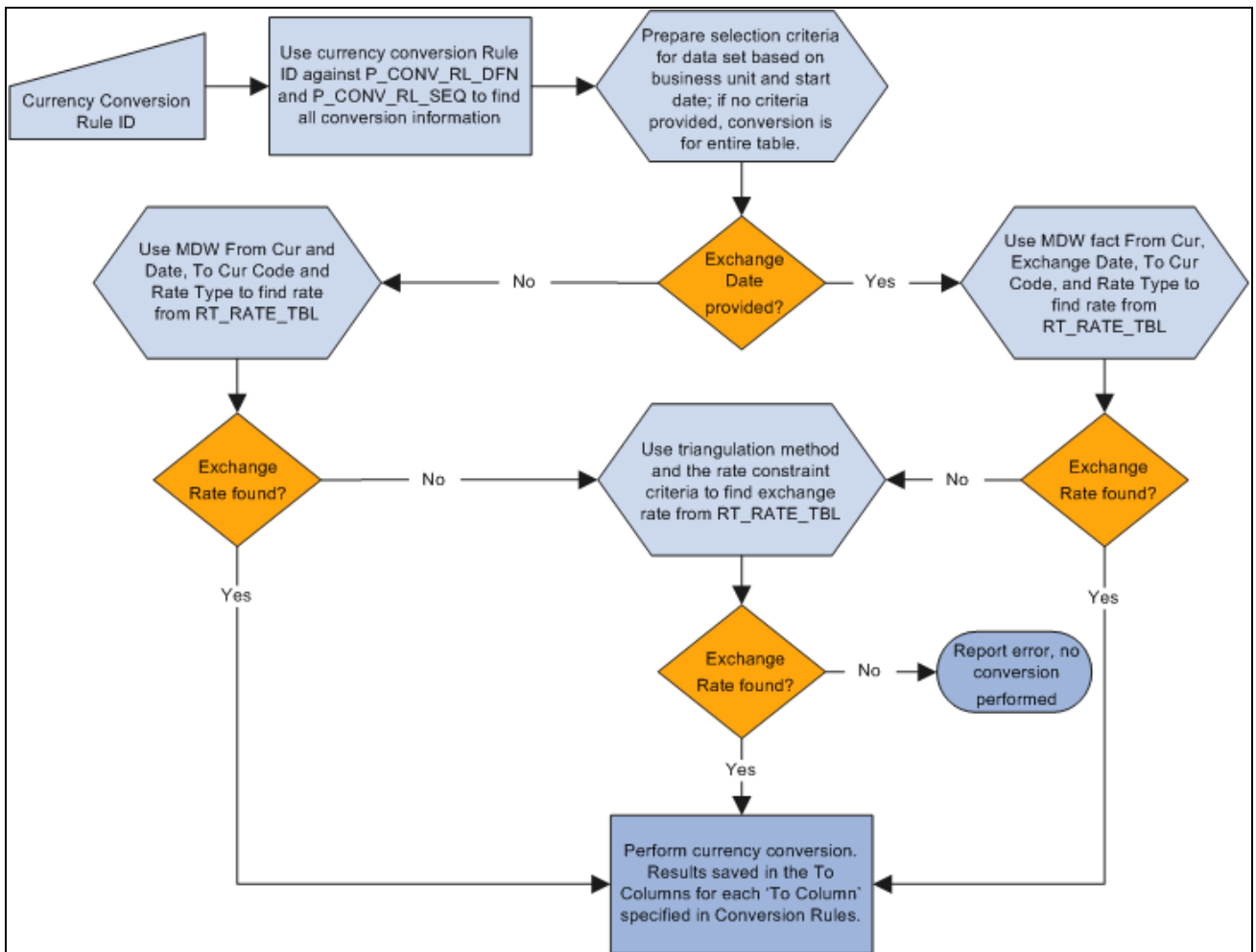
Carrying multiple source amounts into the MDW fact table



Carrying multiple source and base amounts into the MDW fact table

Understanding Currency Conversion Methodology

The following diagram represents the currency conversion process:



Currency conversion methodology flow

The following sections provide additional technical details regarding the currency conversion process.

Identifying the Data Set from a Conversion Rule Business Unit

Although the business unit specified in the Currency Conversion Rule is a PF business unit, different MDW fact tables can occur at different business unit granularity levels. Therefore, the process of identifying the data set for the currency conversion process must be aware of the business unit granularity level of the fact table. There are three levels of business unit granularity: source business unit, composite business unit, and PF business unit.

Based on the granularity of the business unit, you use the following rules to determine the surrogate IDs used to constrain the fact data:

- If the granularity level is the PF Business Unit, then `SELECT PBU_SID FROM PS_D_BUSINESS_UNIT WHERE BUSINESS_UNIT = <Conversion Rule's BU>`.
- If the granularity level is the Composite Business Unit, then `SELECT CBU_SID FROM PS_D_BUSINESS_UNIT WHERE BUSINESS_UNIT = <Conversion Rule's BU>`.
- If the granularity level is the Source Business Unit, then `SELECT BU_SID FROM PS_D_BUSINESS_UNIT WHERE BUSINESS_UNIT = <Conversion Rule's BU>`.

Identifying the Exchange Date from Date Columns

When the Currency Conversion Rule does not specify an exchange date to identify the currency conversion rate, then the Exchange Date column from the Schema Rule is used to determine the exchange date. Because an MDW fact table may have data in different date/period granularity, the Exchange Date column is used with the date/period dimension record name to determine the date.

Based on the value of date/period dimension record name, use the following rules to determine the date: If the date/period dimension record name is:

- D_DAY


```
SELECT DAY_DT FROM PS_D_DAY WHERE DAY_SID = <Exchange Date Col Value>
```
- D_MONTH


```
SELECT MAX(DAY_DT) FROM PS_D_DAY WHERE MONTH_SID = <Exchange Date Col Value>
```
- D_WEEK


```
SELECT MAX(DAY_DT) FROM PS_D_DAY WHERE WEEK_SID = <Exchange Date Col Value>
```
- D_QUARTER


```
SELECT MAX(DAY_DT) FROM PS_D_DAY WHERE QUARTER_SID = <Exchange Date Col Value>
```
- D_YEAR


```
SELECT MAX(DAY_DT) FROM PS_D_DAY WHERE YEAR_SID = <Exchange Date Col Value>
```
- D_PATTERN_DAY


```
SELECT DAY_DT FROM PS_D_CAL_DAY WHERE PDAY_SID = <Exchange Date Col Value>
```
- D_DET_PERIOD


```
SELECT PPERIOD_END_DT FROM PS_D_DET_PERIOD WHERE PPERIOD_SID = <Exchange Date Col Value>
```
- D_SUM_PERIOD


```
SELECT MAX(A.PPERIOD_END_DT) FROM PS_D_DET_PERIOD A, PS_R_DET_SUM_PRD B, PS_D_SUM_PERIOD C WHERE C.PPERIOD_SUM_SID = <Exchange Date Col Value> AND B.PPERIOD_SUM_SID = C.PPERIOD_SUM_SID AND B.PPERIOD_SID = A.PPERIOD_SID
```
- D_DET_BUDGET


```
SELECT BPERIOD_END_DT FROM PS_D_DET_BUDGET WHERE BPERIOD_SID = <Exchange Date Col Value>
```
- D_SUM_BUDGET


```
SELECT MAX(A.BPERIOD_END_DT) FROM PS_D_DET_BUDGET A, PS_R_DET_SUM_BPRD B, PS_D_SUM_BUDGET C WHERE C.BPERIOD_SUM_SID = <Exchange Date Col Value> AND B.BPERIOD_SUM_SID = C.BPERIOD_SUM_SID AND B.BPERIOD_SID = A.BPERIOD_SID
```

- D_PATTERN_YEAR

```
SELECT MAX(DAY_DT) FROM PS_D_PATTERN_DAY WHERE PYEAR_SID = <Exchange Date Col Value>
```

- D_PATTERN_WEEK

```
SELECT MAX(DAY_DT) FROM PS_D_PATTERN_DAY WHERE PWEK_SID = <Exchange Date Col Value>
```

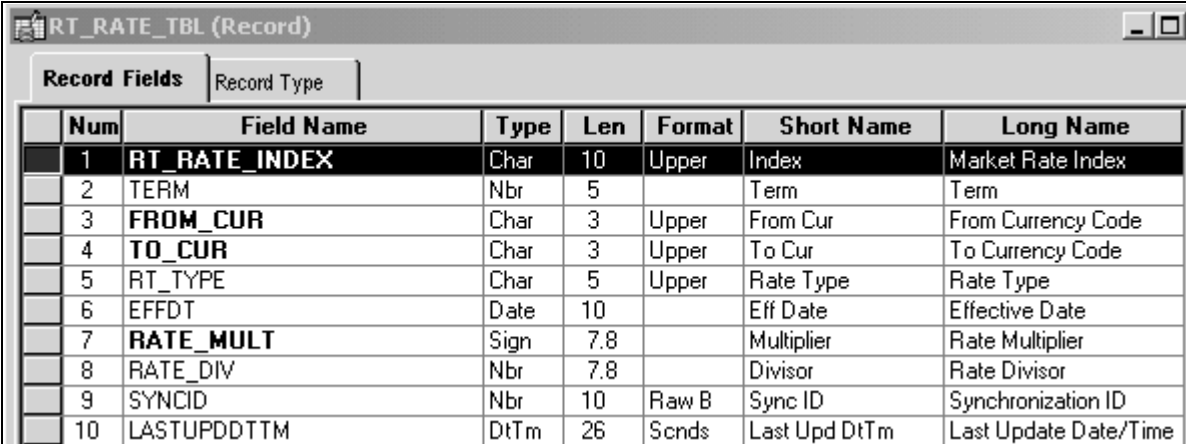
- Other tables: ETL checks to see whether the column represents a date or datetime field. If the field is not a date or datetime field, then the process terminates with error in the log.

You should never use D_DT_PATTERN or other common time dimension tables as the date/period dimension record name for an MDW fact table because they contain only period attributes, rather than a list of periods.

Currency Conversion Logic Using the Enterprise Rate Table

The Enterprise rate table that is used for currency conversion is the RT_RATE_TBL. When the triangulation method is used, CURR_QUOTE_TBL is also used to provide the triangulation rule.

The RT_RATE_TBL has the following structure:



Num	Field Name	Type	Len	Format	Short Name	Long Name
1	RT_RATE_INDEX	Char	10	Upper	Index	Market Rate Index
2	TERM	Nbr	5		Term	Term
3	FROM_CUR	Char	3	Upper	From Cur	From Currency Code
4	TO_CUR	Char	3	Upper	To Cur	To Currency Code
5	RT_TYPE	Char	5	Upper	Rate Type	Rate Type
6	EFFDT	Date	10		Eff Date	Effective Date
7	RATE_MULT	Sign	7.8		Multiplier	Rate Multiplier
8	RATE_DIV	Nbr	7.8		Divisor	Rate Divisor
9	SYNCID	Nbr	10	Raw B	Sync ID	Synchronization ID
10	LASTUPDDTTM	DtTm	26	Scnds	Last Upd DtTm	Last Update Date/Time

RT_RATE_TBL structure

The CURR_QUOTE_TBL has the following structure:

CURR_QUOTE_TBL (Record)							
Record Fields		Record Type					
	Num	Field Name	Type	Len	Format	Short Name	Long Name
	1	FROM_CUR	Char	3	Upper	From Cur	From Currency Code
	2	TO_CUR	Char	3	Upper	To Cur	To Currency Code
	3	EFFDT	Date	10		Eff Date	Effective Date
	4	EFF_STATUS	Char	1	Upper	Status	Status as of Effective D
	5	RATE_DECIMALS	Nbr	1		Decimals	Rate Decimal Positions
	6	QUOTE_UNITS	Nbr	4		Units	Quote Units
	7	RATE_DIRECT	Char	1	Upper	Quotation	Rate Quotation Basis
	8	AUTO_RECIPROCATE	Char	1	Upper	Reciprocate	Auto Reciprocate
	9	RATE_TRIANGULATE	Char	1	Upper	Triangulate	Rate Triangulate
	10	REF_CUR	Char	3	Upper	Ref Cur	Reference Currency
	11	PRIMARY_VISUAL	Char	2	Upper	Visual	Primary Visual Rate
	12	XRATE_OVERRIDE	Char	1	Upper	Allow Override	Allow Cross-Rate Overrid
	13	XRATE_RECALC	Char	2	Upper	Recalculate	Cross-Rate Recalculate

CURR_QUOTE_TBL structure

Using the From Currency, To Currency, Rate Type, and Date from the Currency Conversion Rules:

Condition	Result
If exchange rate is found in RT_RATE_TBL	then $Converted\ Amount = Source\ Amount * (RATE_MULT/RATE_DIV)$
If exchange rate is not found in RT_RATE_TBL	then verify using the From and To Currencies and Exchange Date when the currency quotation method is defined in CURR_QUOTE_TBL
If Currency Quotation method is defined and RATE_TRIANGULATE = 'Y'	then get REF_CUR field from CURR_QUOTE_TBL Leg 1: Find in the RT_RATE_TBL the exchange rates (RATE_MULT and RATE_DIV) between the From Currency and REF_CUR for the given Rate Type and Date. Leg 2: Find in the RT_RATE_TBL the exchange rates (RATE_MULT and RATE_DIV) between REF_CUR and the To Currency for the given Rate Type and Date. $Converted\ Amount = Source\ Amount * (RATE_MULT\ of\ leg\ 1 * RATE_MULT\ of\ leg\ 2) / (RATE_DIV\ of\ leg\ 1 * RATE_DIV\ of\ leg\ 2).$
If Currency Quotation method is not defined, or if RATE_TRIANGULATE = 'N'	then = error (because no conversion rate is found)

The Exchange Date is used to identify an effective-dated exchange rate. This means that if the exchange rate is not available, the most recent entry that matches all the other exchange rate selection criteria (that is, To and From Currencies and Rate Type) is used.

Error Logging

There are three situations in which error can occur in the currency conversion process:

- `RATE_MULT = 0`, when `RATE_MULT` is used for currency conversion.
- `RATE_DIV = 0`, when `RATE_DIV` is used for currency conversion.
- No exchange rate between from and to currency codes is found for a given exchange rate type.

Errors for MDW currency conversion are written to an error table `PS_E_CCU_ERROR`.

The following information is made available in this table:

- Currency conversion rule ID
- Fact table name
- From Amount column
- To Amount column
- From Currency column
- To Currency column
- Rate table name
- Rate type
- From currency
- To currency
- `RATE_MULT` value
- `RATE_DIV` value

Understanding Currency Conversion Rules

You must specify the following parameters for the ETL currency conversion process:

- **Schema Rules:** Specify the structure of MDW tables upon which currency conversion process are performed.
- **Conversion Rules:** Specify how the currency conversion should be performed by indicating the rate type, target currency code, and the effective rate to use for the currency conversion process.
- **Chunking Rules:** Identify the subset of data in MDW tables that are affected by the conversion process.

Together these three rules are referred to as *currency conversion rules*.

Schema Rules

Schema rules specify on what table the currency conversion is performed, the source amount and currency code column, and target amount and currency code column that is populated by the conversion result. Schema rules also include the table name where the source and target columns are found. Essentially, the schema rules abstract the interrelationship between source and target columns for currency conversion. The schema rules also specify the granularity of data for the conversion process. This is done by specifying the relevant date/period dimension and business unit column for both resolving the exchange date and chunking the data set for currency conversion.

You set up the schema rules based on the *schema definition*, which is system data and is delivered as part of the PeopleSoft EPM product. The predefined schema definition associates source amounts and currency code fields to their target amount and currency code fields. Schema rules contain the list of all target columns for currency conversion, along with their associated information, such as the record name, *Source Amount* column, the *Source Currency Code* column, and the *To Currency Code* column for a particular fact table.

It is important to understand the difference between the *schema definition* and the *schema rules*. The *schema definition* only records the relationship of columns in delivered tables. A *schema rule* is customer-specific because it depends on what rate and date to use for a currency conversion. For example, the schema definition only indicates the currency code column for a particular amount column in a stated table.

Schema rules consist of:

- Table name.

- Business unit column.

An MDW fact table may have multiple business unit surrogate ID (SID) columns, but only one is used to drive the currency conversion. Because not all MDW tables have a business unit SID, this column is optional.

- Business unit grain level.

Although all of the data in the same MDW fact table has the same granularity level, different MDW facts can be at different business unit granularity levels, whether it is a source business unit, composite business unit, or performance business unit (PF BU.) The business unit grain level is required only if the Business Unit column exists.

- Exchange date/period column for the table.

Only one date/period column per table is used to drive the exchange rate. Some MDW tables do not have a date/period SID column; therefore, the column is optional. When no date/period SID column is specified, the exchange date resolution always uses the current date/prespecified date in the conversion rule.

- The date/period dimension record for the exchange date/period column.

This is required only if the exchange date/period column is specified.

- *From* amount column name.
- *To* amount column name.
- *From* currency code column name.

The utility assumes that the from currency code column is always populated with the source currency code from the source table.

- To currency code column name.

The to currency code is populated by the To Currency Code parameter when the currency conversion has completed.

The following example shows the schema definitions for the F_AP_TRAN fact table:

The screenshot shows the 'Schema Definition' tab for the F_AP_TRAN fact table. The record name is F_AP_TRAN. The following fields are defined:

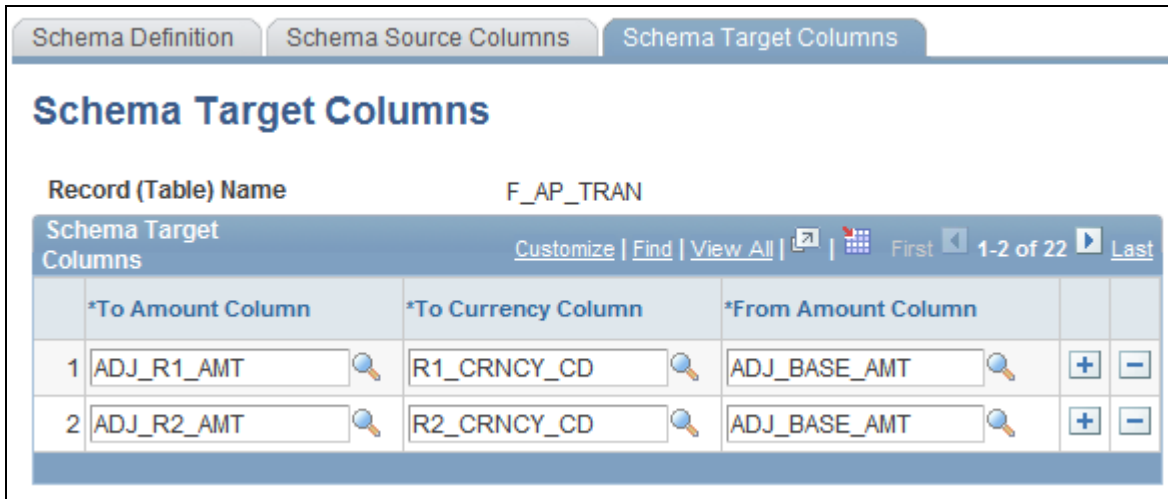
Business Unit Column	AP_BU_SID
Business Unit Granularity Lvl.	Source
MDW Fact Exchange Dt Column	ACCT_DAY_SID
Date/Period Dimension Record	D_DAY

Schema Definition page for F_AP_TRAN fact table

The screenshot shows the 'Schema Source Columns' tab for the F_AP_TRAN fact table. It displays a table with columns for *From Amount Column and *From Currency Column. The table lists two source columns:

	*From Amount Column	*From Currency Column		
1	ADJ_BASE_AMT	BASE_CRNCY_CD	+	-
2	DILS_BASE_AMT	BASE_CRNCY_CD	+	-

Schema Source Columns page for F_AP_TRAN fact table



Schema Target Columns page for F_AP_TRAN fact table

Table F_AP_TRAN is an MDW fact table that has data granularity at day level. It also tracks its data with AP Business Unit, one type of business unit originating from the source database. Therefore, in the schema definition page for F_AP_TRAN table, we identify the Business Unit column (AP_BU_SID), as well as the business unit granularity (that is, the source). In addition, we also specify the Date column in the table (ACCT_DAY_SID), as well as the date/period granularity, by specifying the date/period dimension table, in this case, D_DAY.

Table F_AP_TRAN has several data columns that require currency conversion. Among others, they are ADJ_BASE_AMT and DILS_BASE_AMT. Therefore, we specify these columns on the Schema Source Columns page for F_AP_TRAN. You must associate the current currency code for each of the columns upon which currency conversion can be performed.

You must associate each reporting amount and currency code column to the source amount and currency column. You do this on the Target Schema Columns page for F_AP_TRAN.

Note. Schema rules are prepackaged with EPM, but if you add a new MDW dimension or fact table upon which a currency conversion process must be performed, you must add schema rules for the new tables.

See [Chapter 10, "Processing Trees and Recursive Hierarchies," Setting Up Parameters for Tree and Recursive Hierarchy Processing, page 163.](#)

Conversion Rules

Conversion rules specify how the currency conversion should be carried out, such as which rate and exchange date to use, or whether to perform the conversion for a subset of data (constrained by business unit, date range, or both) or for the entire table. Conversion rules specify the rate type, the target currency code, and the data that determines the effective rate to use for the currency conversion process. The rules also specify which specific amount columns in the table to convert.

Conversion rules are user-defined data and are not delivered as part of the EPM product. Because exchange rate rules are specific to user requirements, and therefore they are treated as user data, PeopleSoft provides only sample data with your EPM product.

Exchange rate rules consist of:

- Rate Type.

- To currency code.
- Conversion date.

This rule is optional. The conversion date can be a specified date or current date. Otherwise, the exchange rate is determined by the date of the transaction.

You must choose either:

- To give the specific exchange date that the currency conversion process will use to identify the exchange rate.
- To have the currency conversion process use the processing date.
- To have the currency conversion process use the exchange date column as specified in the schema rule.

The following provides an example of how to create a currency conversion rule:

MDW Currency Conversion Rule		MDW Conversion Schema Rule	
MDW Currency Conversion Rule			
Currency Conversion Rule	AP_TRANS		
*Description	AP Transaction		
Conversion Rule Definition			
*Rate Type	AVG		
*To Currency Code	USD		
Specify conversion date	Recent dt. ▼		
Chunking Rule Definition			
Warehouse BU			
Start Date		End Date	
Notes			

MDW Currency Conversion Rule page for AP_TRANS fact table

MDW Currency Conversion Rule		MDW Conversion Schema Rule	
MDW Conversion Schema Rule		AP_TRANS	
Currency Conversion Rule		AP_TRANS	
MDW Currency Conversion Rule		Customize Find View All First 1 of 11 Last	
*Record (Table) Name	*To Amount Column	From Amount Column	
1 F_AP_TRAN	ADJ_R1_AMT	ADJ_BASE_AMT	+ -

MDW Conversion Schema Rule page for AP_TRANS fact table

The AP_TRANS conversion rule tells the MDW currency conversion process what criteria to use to perform the currency conversion on the amount columns listed in the MDW Conversion Schema Rule.

From Example 1 in Section 3.1, we find that table F_AP_TRAN has a date/period column that is used to determine the exchange date. If you want to have the exchange date follow the transaction date, then you specify conversion date as "date column." This will force the currency conversion utility to take the date from the F_AP_TRAN table, as specified in the Schema Rules. In this example, Recent Date (Recent Dt.) is used for the conversion date. Therefore, you are overriding the exchange date used for currency conversion with the recent date (that is, processing date). You can also override the exchange date with a pre-specified date. This feature is useful when your company has a policy of using a particular date as a standard exchange date for a subset of data.

In this example, chunking rules are not specified. Therefore, you are telling the MDW Currency Conversion utility to perform currency conversion for all of the rows in F_AP_TRAN table.

Chunking Rules

Chunking rules are parameters that the currency conversion process uses to identify the subset of data in MDW tables that are affected by the conversion process. Chunking rules are considered part of the currency conversion rules. They are specific to user requirements and are treated as user data. Therefore, PeopleSoft provides only sample data with your EPM product.

Chunking rules consist of performance (PF) business unit, start date, and end date. These parameters are optional. If you do not provide business unit and chunking date parameters, the process performs conversion on the entire table.

Chunking rules identify the subset of data in MDW tables that experience currency conversion. These rules consist of:

- **Business Unit:** This rule is optional. If it is provided, only MDW fact data that belongs to this business unit will experience currency conversion. Otherwise, data for all business units experience currency conversion. The business unit in the currency conversion rule is always a PF business unit. If the MDW table has different business unit granularity levels (source or composite), refer to section 4.2 on how this PF business unit is translated into the equivalent source or composite business unit.
- **Start Date:** This rule is optional. If it is provided, any date greater than or equal to this parameter experiences currency conversion.

- **End Date:** This rule is optional. If it is provided, any date less than or equal to this parameter value experiences currency conversion.

Setting Up Currency Conversion

You run the currency conversion process from a prepackaged ETL job, which uses the *Currency Conversion Rule ID* as an input and is a composite of the conversion rules whose parameters you have set up before you run the currency conversion process. The Currency Conversion Rule ID provides the necessary information to perform the currency conversion, such as the rule to obtain the appropriate exchange rate, the rule to obtain the subset of data on which the currency conversion is performed, and the source and target columns for currency conversion.

To set up currency conversion, use the Schema Definition (CCU_SCHEMA_DEFN) component and the Currency Conversion (CCU_CONV_DEFN) component.

Before you run the ETL currency conversion process, you must define the required parameters. This section discusses how to:

1. Set up the schema definition and columns.
2. Define schema source columns.
3. Define schema target columns.
4. Define rules for currency conversion.

Pages Used to Set Up the Schema Definition and Currency Conversion Rules

<i>Page Name</i>	<i>Definition Name</i>	<i>Navigation</i>	<i>Usage</i>
Schema Definition	CCU_SCHEMA_DEFN	EPM Foundation, EPM Setup, Common Definitions, MDW Currency Conversions, Schema Definition	Set up the schema definition.
Schema Source Columns	CCU_SCHEMA_SRC	EPM Foundation, EPM Setup, Common Definitions, MDW Currency Conversions, Schema Definition	View or modify source monetary amount and currency code columns.
Schema Target Columns	CCU_SCHEMA_TGT	EPM Foundation, EPM Setup, Common Definitions, MDW Currency Conversions, Schema Definition	View or modify target monetary amount and currency code columns.

Page Name	Definition Name	Navigation	Usage
MDW Currency Conversion Rule	CCU_CONV_DEFN	EPM Foundation, EPM Setup, Common Definitions, MDW Currency Conversions, MDW Currency Conversion Rules	Set up currency rule definition and chunking rule definition.
MDW Conversion Schema Rule	CCU_CONV_SEQ	EPM Foundation, EPM Setup, Common Definitions, MDW Currency Conversions, MDW Conversion Schema Rules	Set up conversion schema rule.

Setting Up the Schema Definition

Access the Schema Definition page (EPM Foundation, EPM Setup, Common Definitions, MDW Currency Conversions, Schema Definition).

Schema Definition page

Business Unit Column Enter the name of the PF (performance) business unit column.

MDW Fact Exchange Dt Column Enter the date column used to determine the exchange date or conversion date. (MDW fact exchange date column)

Date/Period Dimension Record Enter the level (year, month, day, and so on) for the Fact Exchange Date column.

Setting Up Currency Conversion Rules

Access the MDW Currency Conversion Rule page (EPM Foundation, EPM Setup, Common Definitions, MDW Currency Conversions, MDW Currency Conversion Rules).

The screenshot shows the 'MDW Currency Conversion Rule' page. At the top, there are two tabs: 'MDW Currency Conversion Rule' (selected) and 'MDW Conversion Schema Rule'. Below the tabs is the main title 'MDW Currency Conversion Rule'. The page is divided into several sections:

- Currency Conversion Rule:** CRM_CASE
- *Description:** CRM Case Fact
- Conversion Rule Definition:**
 - *Rate Type:** AVG
 - *To Currency Code:** USD
 - Specify conversion date:** Dt. column
- Chunking Rule Definition:**
 - Warehouse BU:** CORP1
 - Start Date:** 10/01/2004
 - End Date:** 10/20/2004
- Notes:** Currency Conversion Rule for CRM Case

MDW Currency Conversion Rule page

Currency Conversion Rule The name of the currency conversion rule.

Conversion Rule Definition

Rate Type Select the rate type for this rule.

To Currency Code Select the currency code for the converted value.

Specify conversion date Select the date as of which the conversion rate should be applied. Values are:

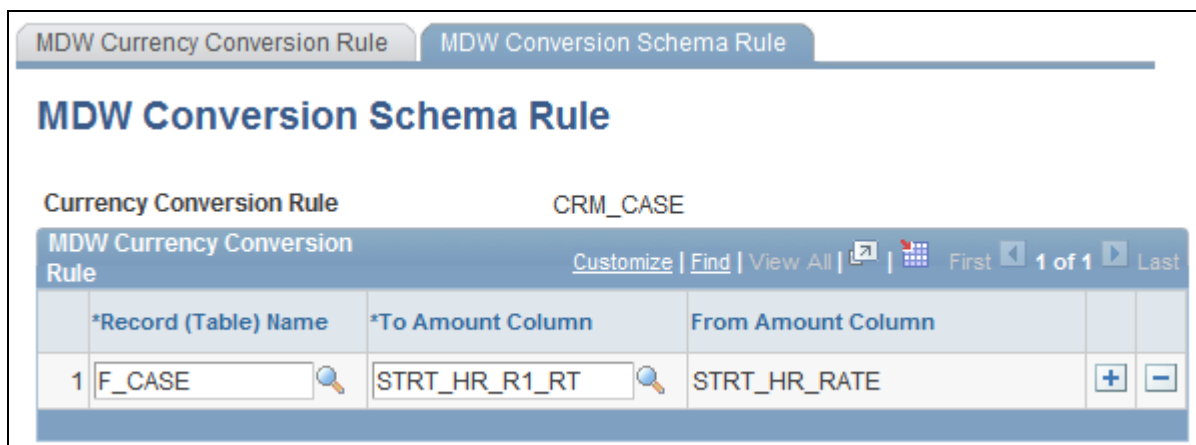
- Dt. column* (date column): From the transaction record.
- Exchg Dt.* (exchange date): User defined for the entire data set on which currency conversion is performed.
- Recent Dt.* (recent date): The most recent conversion rate that exists in the exchange rate table.

Chunking Rule Definition

- Warehouse BU** (warehouse business unit) (Optional) Select the warehouse business unit (PF BU) for the currency conversion. If you supply a business unit parameter, only MDW fact data that belongs to that business unit will experience currency conversion. If you do not supply a business unit parameter, data for all business units will experience currency conversion. Because all source business units have an associated PF BU in the EPM MDW, the business unit in the conversion rule refers to the PF business unit.
- Start Date** (Optional) Enter the date for which the currency conversion should begin. If you supply a start date parameter, any date greater than or equal to this parameter value will experience currency conversion. If you do not supply a start date parameter, then the currency conversion process uses 01-01-1900 in place of a start date.
- End Date** (Optional) Enter the date for which the currency conversion should end. If you supply an end date, any date less than or equal to this parameter value will experience currency conversion. If you do not supply an end date parameter, then the currency conversion process uses the present (or process) date in place of an end date.
- Notes** (Optional) Enter notes about this currency conversion rule.

Setting up the Conversion Schema Rule

Access the MDW Conversion Schema Rule page (EPM Foundation, EPM Setup, Common Definitions, MDW Currency Conversions, MDW Conversion Schema Rules).



MDW Conversion Schema Rule page

- Record (Table) Name** Select the record on which the conversion will be performed. The record name must exist in the schema definition.

To Amount Column	Select the column for the conversion result. The options are obtained from the schema definition.
From Amount Column	The column to be converted. This field is automatically populated from the schema definition when you select the To Amount Column.

Running the ETL Currency Conversion Process

The process that actually converts monetary amounts stored in your MDW tables is an ETL utility. Currency conversion is a post process that must be applied after initial data load of all Fact tables. Therefore, before running this utility, make sure to run all the fact jobs in your project.

You can perform the ETL currency conversion process using both a direct exchange rate and a triangulated exchange rate. (A triangulated exchange rate conversion takes place when no direct exchange rate between a *from currency* and a *to currency* exists, but the exchange rates exist between the *from currency* and a reference currency, and from the reference currency the *to currency*. Using the triangulation method, the currency conversion process indirectly establishes the exchange rates between a from currency and a to currency using the intermediary reference currency. You set up triangulation parameters with other EPM setup functions.

If you provide an optional exchange rate date parameter, the currency conversion process searches for an exchange rate for a given exchange rate date. If you do not provide an exchange rate date parameter, the conversion utility uses the date of the transaction to determine the exchange rate.

Note. The MDW currency conversion process does not perform balancing; that is, when there are parent and child tables and rounding occurs, the process does not ensure that the sum of the child tables equals the value of the parent table.

If you map multiple source business units into one warehouse business unit (WBU), the default currency of the source business units must be the same as the default currency of the warehouse business unit.

Note. The Currency Conversion Rule ID exists on the MDW Conversion Schema Rule and the MDW Conversion Schema Rule pages. The Currency Conversion Rule ID provides the MDW Currency Conversion process with the necessary information to perform the currency conversion that you have previously identified, such as the rule to obtain the appropriate exchange rate, the rule to obtain the subset of dates within which the currency conversion will take place, and the source and target columns for the currency conversion.

Steps Required to Run the MDW ETL Currency Conversion Utility

Perform the following steps to run the MDW ETL currency conversion process:

1. In IBM WebSphere DataStage Director, navigate to the MDW currency conversion job by expanding the nodes in the left navigation panel using the following path: *EPM90_Uutilities, Sequence, SEQ_J_Run_CurrencyConversion*.
2. Select the sequence job *SEQ_J_Run_CurrencyConversion* in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters by entering the required currency conversion rule for a specified fact table, and click Run.

The job is scheduled to run with the current date and time, and the job's status is updated to *Running*.

Currency conversion occurs for all the facts grouped under this rule.

4. Repeat steps one through three for each currency conversion rule that you want to run.

See Also

PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook, "Preparing to Load Source Data Into EPM"

Chapter 9

Setting Up Multilanguage Processing and Running the Language Swap Utility

This section provides an overview of multilanguage processing and the language swap utility, and discusses how to run the language swap utility.

Understanding Multilanguage Processing

Many organizations conduct business globally and deploy their PeopleSoft source systems in various locations throughout the world. PeopleTools can store multiple translations of application data and PeopleTools objects in a single database. Each PeopleSoft database has a single base language. The base language of a PeopleSoft database is usually the language most commonly used by application users, and is the language in which data is stored in the core PeopleSoft tables known as base language tables.

All PeopleTools objects (such as pages, fields and queries) can be maintained in multiple languages. Descriptions of application data elements (such as departments, locations and job codes) can also be maintained in multiple languages. The key to maintaining this data in multiple languages is the use of special tables known as *related language tables*.

Related language tables store descriptions and other language-sensitive elements in all languages other than the base language of the database. In this way, while any table in the database can store data in the base language of that database, only tables that have related language tables can maintain the same data in multiple languages simultaneously. For example, it is unlikely that you would maintain the descriptions of your general ledger journal lines in multiple languages—the sheer volume of the journal lines in most systems would preclude any effort to maintain translations of their descriptions. The cost of hiring a translator to translate each journal line would be prohibitive, and in most cases only the person entering the journal line, and possibly that person's supervisor, would be likely to want to view that information again. However, for frequently used values, such as a chart of accounts, many users across your entire organization would often need to refer to this data. Therefore, you would most likely maintain the descriptions of each ChartField entry in each language spoken by your users. In this case, you would not need a related language table for your Journal Lines table, as you would be maintaining journal line descriptions in a single language, which would be in the base table. However, you would need a related language table for each of your ChartField tables.

When the system displays a language-sensitive field value, it retrieves the text from either the base table or the related language table, depending on the following:

- The current language preference.
- Whether any translated rows for the field exist in the related language table.

The language preference refers either to the PeopleSoft PIA sign-in language, or in the case of PeopleSoft Application Designer, to the language preference as determined by the PeopleSoft Configuration Manager language setting. If the current language preference is the system's base language, the text is retrieved from the base table. If the language preference is a non-base language, then the system looks for a translation of the text in the related language table. If it finds a translation, it displays the translated text; if no translation exists, the system uses the text in the base table. This enables developers to selectively translate portions of the data, while keeping the system fully functional at all times, even if not all rows have been translated.

EPM also uses related language tables to support multilanguage processing. In each of the three data warehouse layers (the OWS, OWE, and MDW), all records that have translatable description fields have corresponding related language tables. Related language tables are defined for every OWS, DIM, and D00 table that contain translatable values. For example, the table CUSTOMER_D00 has a corresponding related language table CUSTOMER_LNG. Related language tables have key structures identical to the related DIM and D00 table plus one additional key field called language code (LANGUAGE_CD). The language code field holds the source language value. Prepackaged ETL jobs extract this data from a PeopleSoft source system and populate the field with your source language value.

EPM extracts data from PeopleSoft source systems, which have their own base languages and supported foreign languages. Multilanguage infrastructure in PeopleSoft source systems store the base language in the base table and the foreign language descriptions in the related language table. If the base language of the source database and that of the EPM database are not the same (but the source database's base language is one of EPM warehouse's supported foreign languages), the description from the base table in source database must be stored in the related-language table in EPM to ensure consistency. If a supported foreign language in the source database is the EPM warehouse's base language, then that foreign language description must be stored in the base table in the EPM database. We achieve this consistency through use of the Language Swap Utility.

The Language Swap Utility and multilanguage processing enables you to:

- Import descriptions for any language into EPM target warehouse tables.
- Exchange descriptions in source tables with the related-language tables, when source defined language is different than the EPM defined language.
- Report in different languages.

The Language Swap utility abstracts the process of language swapping from all of the ETL maps that load data into the EPM database. As a result, the utility reduces the complexity and increases the maintainability of the ETL maps.

Understanding Multilanguage Setup

You must enable multilanguage processing before you can view your data in a different language or run multilanguage reports. Setting up multilanguage processing requires three simple steps:

- Define the base language of your source systems: Use the Define Warehouse Source page to perform this task.

See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Setting Up EPM Business Rules," Specifying Your EPM Sources.

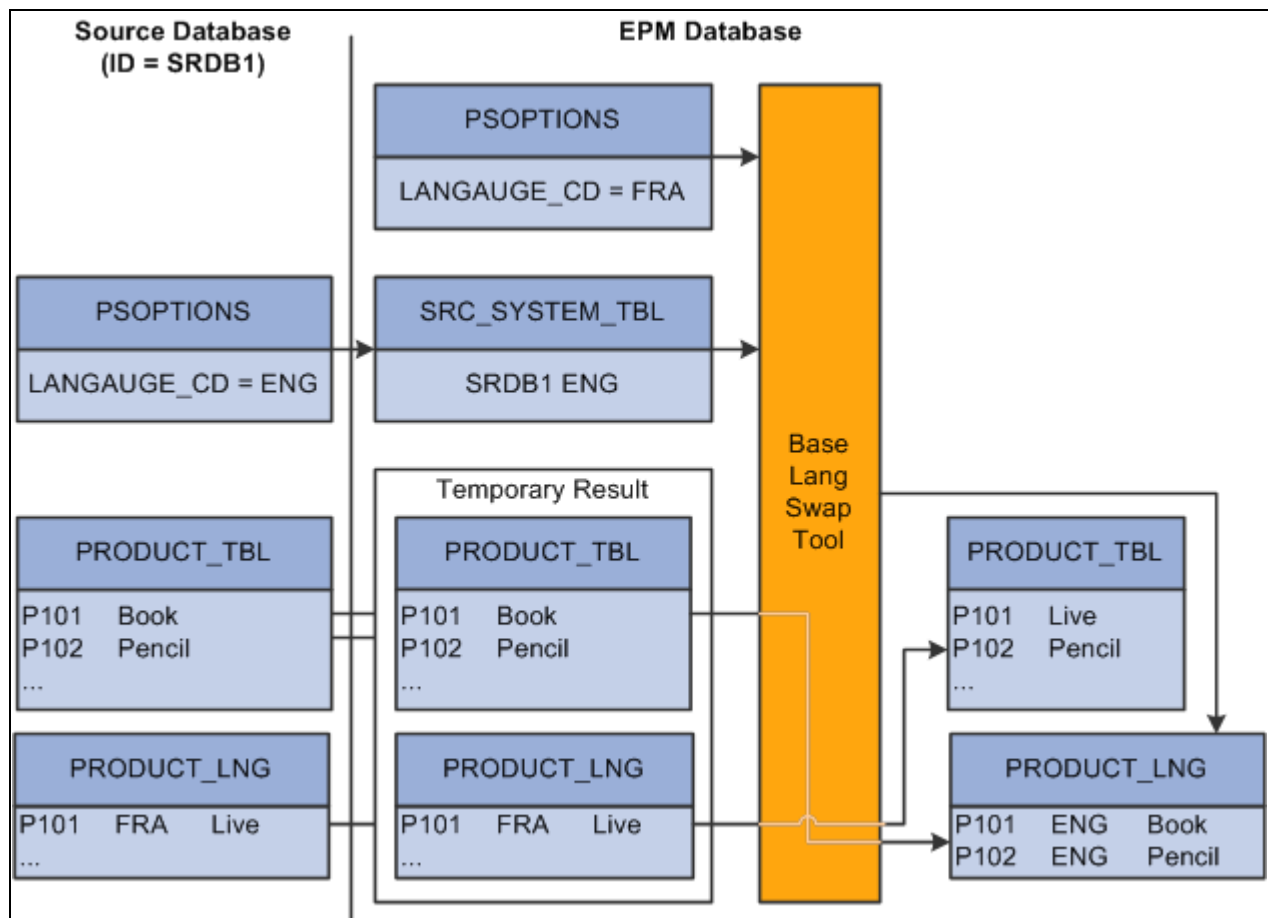
- Run the language swap utility.

Note. Language Swap is OWS post process and so before running this utility, make sure to run all the staging jobs for base and language tables.

Understanding the Language Swap Utility

The language swap utility automatically detects the mismatched languages between the language defined in the source and the language defined in the EPM database, and sets the correct base language for incoming data. The utility compares the base language of the source database (as it is stored in the SRC_SYSTEM_TBL) and the base language of the EPM database. If they are different, the utility swaps the descriptions that are found in the base table and the related-language tables whenever possible. Once the data are in OWS and the base language swap utility has been performed, the reference data and their related language data in OWS are conformed to the PeopleSoft infrastructure for related language. This ensures proper synchronization and enables you to process and report in multiple languages. Note, however, that this process requires that descriptions be available in the source record. This process cannot be performed if the related language record doesn't have any description fields, or any other fields that are translatable, from the base table.

The following graphic depicts the language swap process.



Language swap process flow

The swapping process works as follows:

1. Check if the base language of the source database is the same as the base language of the EPM database.
If the languages are the same, then continue to the last step in this process. Otherwise, proceed to the next step.
2. Check if the base language of the source database is a supported foreign language in the EPM database.
If the language is the same, then create a corresponding entry in the related-language table for every new (not yet swapped) row in the base table. The LANGUAGE_CD used for tagging the new entry in the related-language table is the LANGUAGE_CD for the source database as it is found in the SRC_SYSTEM_TBL table.
3. Locate the description in the related language table where LANGUAGE_CD for the row is the same as the base language code of the EPM database. Once identified, perform the swap between the description in the base table and the related-language table.
4. Delete from the related-language table any rows where the LANGUAGE_CD is the base language code of the EPM database or is not any of the supported foreign languages in the EPM

The language swap utility is embedded in prepackaged ETL jobs and should be run only when your source database language is different from the language defined for the EPM database. Run the Language Swap utility *after* your source data is completely extracted into the OWS, but *before* you run any subsequent ETL jobs to transform OWS data into OWE or MDW data.

Outrigger Tables

An outrigger table represents foreign language descriptions for data in the base dimension table. The structure of the outrigger table is the same as the structure of an MDW dimension related language table. It has the base table's keys, an additional key to represent the language code (LANGUAGE_CD), and as many columns as there are translatable columns in the base table.

The difference between an outrigger table and a related language table lies in the content. Outrigger tables contain not only the foreign language descriptions of the data, but also the base language descriptions for every row of data found in the base table, even though some data do not have foreign language translations. For example, if there are ten entries in the base table and there are three supported languages, there will be 30 entries in the corresponding outrigger table. If, however, there is no corresponding description (that is, translation) for a particular entry for one of the languages in the related language, the description defaults to the value in the base table.

The advantage of an outrigger table over a related language table for reporting in third-party tools is that the outrigger table contains descriptions in the base language, as well as any supported foreign language, for all data in the base table. The completeness of the content in the outrigger table simplifies the logic for displaying the foreign language description in the third-party reporting tool, which does not have the built-in multilanguage infrastructure like PeopleSoft applications do.

Sample Outcome of the Language Swap Process

Assume you have one source database, SRC01, whose base language is Spanish and supported foreign languages are English and French. In addition, assume your EPM database has English as the base language, and Spanish as the only supported foreign language.

Here is the Product table from the source database:

ProductID	Description
P101	Libro
P102	Lápiz
P103	Pluma

Here is the Product Language table from the source database:

ProductID	Lang CD	Description
P101	ENG	Book
P102	ENG	Pencil
P101	FRA	Livre
P102	FRA	Crayon
P103	FRA	Stylo

After the data is extracted into the OWS and the Language Swap utility is run, the following changes result in the tables:

Here is the OWS Product table from the EPM database:

ProductID	Description
P101	Book
P102	Pencil
P103	Pluma

Here is the OWS Product Language table from the source database:

ProductID	Lang CD	Description
P101	SPA	Libro
P102	SPA	Lápiz
P103	SPA	Pluma

Notice that the French translations that are available in the source database are no longer found in the EPM OWS because French is neither the EPM base language nor its supported foreign language. The Spanish descriptions that are originally in the base table are now in the related language table, while the English descriptions are now in the base table. Product P103 does not have an English description and retains its original description from the source database "pluma" – that is, "pen" in English.

This table shows an example of a Sales fact table:

Time Key	Product Key	Store Key	Quantity	Amount
1	1	1	5	10
1	2	1	1	3
1	3	2	2	3

This table shows an example of the related Product dimension table:

Product Key	SKU	Description
1	A123	<i>Bread</i>
2	B234	<i>Marmalade</i>
3	C345	<i>Milk</i>

Typically, if a dimension table is used in conjunction with an outrigger table, the dimension table does not have any attributes that are country-specific. This is to prevent duplicate attributes in the dimension table and the outrigger table.

This table shows an example of the related Product outrigger table, assuming the base language is English (ENG) and the supported languages are English (ENG), German (GER), and Italian (ITA):

Product Key	Language Code	Description
1	ENG	Bread
1	GER	Brot
1	ITA	Pane
2	ENG	Marmalade
2	GER	Marmelade
2	ITA	Marmellata di agrumi
3	ENG	Milk
3	GER	Milch
3	ITA	Latte

When you constrain language code to a single value, your reporting tool uses the attributes from the outrigger table (Product Description) and the Product dimension table (SKU and Description) to qualify the metrics (Amount and Quantity) and produce the description in the selected language.

Note. In this example, if the related language table did not contain a German description for *Bread*, the description for Bread in German in this outrigger table would contain *Bread*, rather than *Brot*.

You use ETL to populate outrigger tables at the time that you populate the MDW layer.

Running the Language Swap Jobs

The language swap process is a part of the PeopleSoft ETL sequencer job *SEQ_J_Run_LangSwap*. Simply run the job to initiate the language swap process. Please note, however, that this process should not be run until you run all staging jobs that populate base tables and language tables.

Perform the following steps to run the language swap jobs:

1. In IBM WebSphere DataStage Director navigate to the language swap jobs by expanding the nodes in the left navigation panel using the following path: *EPM90_Uilities, Sequence, SEQ_J_Run_LangSwap*.
2. Highlight the jobs and click the "Run" button.

If you want the job to use the values that are defined in the IBM WebSphere DataStage Administrator, then click "Run" button. If you want to override the values then type the appropriate values and then click "Run" button.

Chapter 10

Processing Trees and Recursive Hierarchies

This chapter provides an overview of tree and recursive hierarchy processing and process results, and discusses how to set up and run the Tree and Recursive Hierarchy process.

Understanding Tree and Recursive Hierarchy Processing

This section discusses:

- Trees and recursive hierarchies.
- OWE tree flattener versus MDW tree denormalizer.
- Hierarchies that are supported by the tree and recursive hierarchy process.
- Denormalized tree result balancing.
- Skip levels.
- Tree and recursive hierarchy source tables.
- Multilanguage Support for Relationship and Hierarchy Tables.

Trees and Recursive Hierarchies

PeopleSoft transaction applications store hierarchical structures in the form of trees and recursive hierarchies. In PeopleSoft applications, a recursive hierarchy is a data hierarchy in which all levels of data are from the same data table, and the parent-child relationships between levels are defined in the same source table. That is, recursive hierarchies are generic two-column tables, with the columns representing parent and child.

However, in the MDW, PeopleSoft hierarchical structures, such as trees, recursive hierarchies must be in denormalized form. This enables efficient data query, as well as integration with third-party business intelligence tools. PeopleSoft's tree and recursive hierarchy processing provides the functionality to denormalize trees and recursive hierarchies for multidimensional reporting.

The Tree and Recursive Hierarchy process populates existing relationship and hierarchy tables, which are the source for business intelligence reporting. Unlike the original hierarchy structure—such as tree or recursive hierarchy—that the utility processes, the relationship table contains parent-child relationships within the structure not only to the direct children, but also to the indirect children of a node in the hierarchy. The denormalized structure enables you to use one simple join to access all lower-level entities within a hierarchy that are related directly or indirectly to a particular entity. For this reason, a relationship table is frequently used to facilitate further processing of a fact table, such as aggregation, or to integrate with a third-party reporting tool.

The extract, transform, and load (ETL) process that you use to create input tables for business intelligence reporting combines with the ETL Tree and Recursive Hierarchy process, enabling you to flatten and denormalize your data in a single process. You run the Tree and Recursive Hierarchy process at the same time you that run the ETL process to populate the MDW.

Note. The Tree and Recursive Hierarchy process cannot process some invalid trees. Specifically, it cannot process a tree that refers to a node that does not exist in the node table, as specified in the tree structure definition, and a tree that refers to a leaf that does not exist in the detail table, as specified in the tree structure definition.

See Also

Enterprise PeopleTools PeopleBook: PeopleSoft Tree Manager

OWE Tree Flattener Versus MDW Tree Denormalizer

This section details the differences between the MDW Tree and Recursive Hierarchy ETL utility and the OWE Tree Flattener utility. Understanding the differences in how these two utilities are used can help you understand why two separate tree processing utilities are necessary in EPM.

Subject	MDW Tree Denormalizer	OWE Tree Flattener
Technology Platform	Based on ETL technology.	Based on Application Engine technology for seamless integration with application processing that is also based on the Application Engine.
Supported Types of Hierarchical Structures	EPM and source database trees, recursive hierarchies.	Only EPM trees.
Usage	Preparing hierarchical data for MDW reports, as well as facilitating data transformation by ETL maps. Warehouse ETL maps use the Tree and Recursive Hierarchy ETL utility to enable seamless integration.	Used by Application Engine-based applications to facilitate further data processing. Application Engine-based applications use Application Engine-based tree flattener to enable seamless integration.

The PeopleSoft Tree and Recursive Hierarchy process has two parts: tree flattener and tree denormalizer. First, the process flattens a tree or recursive hierarchy into a relationship table. Next, the process denormalizes the data further into a hierarchy table. Although the processes are sequential, not all tree or recursive hierarchy tables must be denormalized into a hierarchy table. Thus, this step is optional. For example, you may not need to denormalize a hierarchy if you are not using it for business intelligence reporting, but only to facilitate fact data processing, as in aggregating data.

PeopleSoft trees and recursive hierarchies relate each node in a hierarchy only to its direct parent or child. Data stored in this way makes it difficult to access non-subsequent child nodes (the "grandchildren," or further removed generations) of a hierarchy. The relationship table, which is the result of the flattening part of the Tree and Recursive Hierarchy process, makes all generations related to a specific node easily accessible by associating each node in a hierarchy to any of its descendants, direct or indirect.

The output of the denormalization part of the Tree and Recursive Hierarchy process is a hierarchy table. A hierarchy table format associates the lowest level nodes to all of its parents, direct or indirect, in a row of data. That is, the data in a hierarchy table is denormalized such that a node relationship for a particular path within a tree or recursive hierarchy is represented in one row.

The Tree denormalizer process converts trees into a multicolumn data format so that they can be used by your selected business intelligence reporting tool. The output of the tree flattener portion of the process is the input to the tree denormalizer portion of the process. When you process a dimension, you must run the tree flattener *and* the tree denormalizer in sequential order. When you process a fact, if the fact uses a tree as its source, usually only the tree flattener is required.

You can control the Tree and Recursive Hierarchy process by specifying the hierarchy output table name for each tree or recursive hierarchy. If you do not specify a hierarchy output table name (Hierarchy Record Name), the denormalization process does not run, and the tree or recursive hierarchy is not denormalized.

Note. PeopleSoft Analytic Applications use a different ETL process for flattening hierarchical data. Do not confuse that process with the ETL process for business intelligence reporting described here.

Hierarchies Supported by the Tree and Recursive Hierarchy Process

This section reviews the hierarchies supported by the tree and recursive hierarchy process.

Source Database Tree

Source database trees are trees that exist in the source databases that supply data to the EPM warehouses. The source database tree is different from EPM tree, such that tree processing for a source database tree must consistently use the tree definition and underlying data from the source database that has been mirrored in the EPM OWS layer.

The following table provides a list of source tables in the OWS that contain source database tree definitions:

OWS Table Name	Source Table in Source Database	Description
PS_S_TREESTRCT	PSTREESTRCT	Tree Structure table
PS_S_TREEDEFN	PSTREEDEFN	Tree Definition table
PS_S_TREENODE	PSTREENODE	Tree Node table

OWS Table Name	Source Table in Source Database	Description
PS_S_TREELEAF	PSTREELEAF	Tree Leaf table
PS_S_TREE_NODE_TBL	PS_TREE_NODE_TBL	Tree Node Definition table

In addition to the source database tree definition tables that are listed in this list, the underlying data tables for trees are also used as the source for the source database tree processing. You must retrieve the name of the underlying data tables from the tree structure definition table; you will be asked to associate the data table for nodes and leaves when you create your trees. These data tables must already exist in the EPM OWS.

Sometimes the OWS data table name is not the same as the original data table name in the source database. You must refer to metadata console tables PS_MDC_JOB_SRC_REC and PS_MDC_JOB_TGT_REC to associate the OWS table name to its original name as it is found in the source database.

Source Database Recursive Hierarchy

The source table for a relationship or hierarchy table that is based on a recursive hierarchy of the source database data is the OWS table that is the mirror of the source database recursive hierarchy table. One example of the source database recursive hierarchy is the OWS Campaign table: PS_RA_CAMPAIGN.

EPM Tree

EPM trees are typical PeopleSoft trees. They are created within the EPM database and are viewable through the PeopleSoft Tree Manager. The following table provides a list of EPM tables that contain tree definitions:

Table Name	Description
PSTREESTRCT	Tree Structure table
PSTREEDEFN	Tree Definition table
PSTREENODE	Tree Node table
PSTREELEAF	Tree Leaf table
PS_TREE_NODE_TBL	Node Definition table

In addition to the EPM tree definition tables, the underlying data tables for the trees are also used as the source for the EPM tree processing. The name of the underlying data tables can be found in the tree structure definition table.

EPM Recursive Hierarchy

The difference between the EPM recursive hierarchy and the source database recursive hierarchy is that EPM recursive hierarchy stores its recursive hierarchy data in EPM OWE tables, rather than the copy of the source database recursive hierarchy table in the OWS.

Denormalized Tree Result Balancing

A tree is balanced if all of its branches, or paths, are the same length. For example, if one path of a balanced tree is three levels deep, then all of the paths in the tree must be three levels deep. An unbalanced tree has paths of varying length.

Some business intelligence tools, especially ROLAP tools, require that the denormalized dimension tables in the MDW be balanced to use data effectively. If you use a denormalized table for certain third-party business intelligence reporting, you must balance the hierarchy such that no columns contain blanks in the denormalized table. Because not all business intelligence tools require denormalized data to be balanced, the balancing process is optional. Because balancing occurs during denormalization, it has no impact on the tree flattening process.

If you choose to perform balancing, you can select *up-balancing* or *down-balancing*. Up-balancing is replicating detail data to a higher level. Down-balancing is propagating the lowest level nodes in a tree down to the node level next to the detail.

As a result of balancing an unbalanced tree, the description field for the newly created nodes contains a specific notation. This notation is $\langle dd \rangle \sim$, where $\langle dd \rangle$ is the two-digit level number, for example *03* for level three, and \sim , which is the special character that you select for the Hierarchy Balancing Infix field on the Hierarchy Group Definition page.

Note. The balanced node IDs remain the same as their original values.

The balancing process requires up to two parameters on the Hierarchy Group Definition:

- The flag to indicate that up-balancing, down-balancing, or no balancing process is to be performed.
- The special character to indicate that a node is introduced as a result of the balancing process.

If no balancing is required, you do not populate this field.

Balancing Example

To provide an example of the balancing process, consider a simple tree with two levels: a parent node named *auto* and a child node named *car*. If the selected special character is \sim , then these are the balancing results.

Balancing up:

<i>E_ID</i>	<i>E_Desc</i>	<i>L31_ID</i>	<i>L31_Desc</i>	...	<i>L2_ID</i>	<i>L2_Desc</i>	<i>L1_ID</i>	<i>L1_Desc</i>
C	Car	C	31~Car	-	C	02~Car	A	Auto

Balancing down:

<i>E_ID</i>	<i>E_Desc</i>	<i>L31_ID</i>	<i>L31_Desc</i>	...	<i>L2_ID</i>	<i>L2_Desc</i>	<i>L1_ID</i>	<i>L1_Desc</i>
C	Car	A	31~Auto	-	A	02~Auto	A	Auto

No balancing:

<i>E_ID</i>	<i>E_Desc</i>	<i>L31_ID</i>	<i>L31_Desc</i>	<i>...</i>	<i>L2_ID</i>	<i>L2_Desc</i>	<i>L1_ID</i>	<i>L1_Desc</i>
C	Car	-	-	-	-	-	A	Auto

Skip Levels

Trees with strictly enforced levels require that each path of the tree has the same depth. You can skip a level if a portion of the hierarchy does not have nodes at that level. For example, one path in a tree may have levels A, B, C, and D, and another path may have levels A, C, and D (skipping level B).

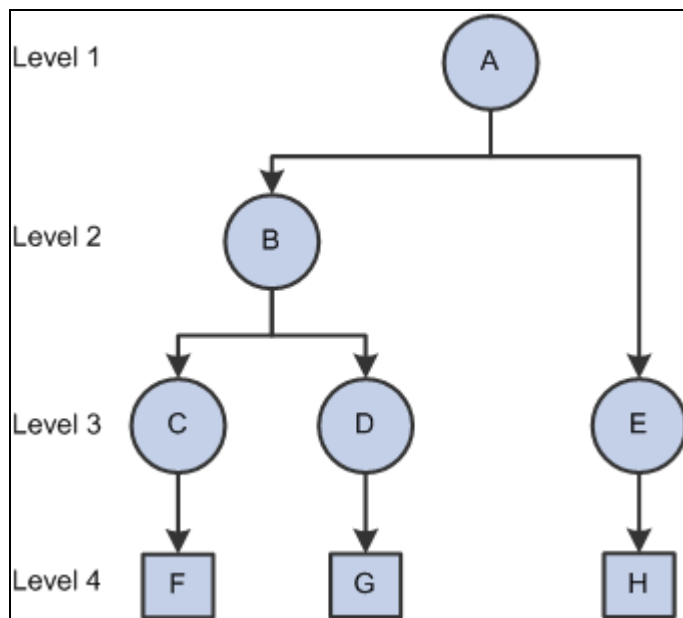
Similar to tree balancing, skip level handling produces synthetic, or artificial, node entries to fill the gap in a denormalized table. For some business intelligence tools, especially ROLAP tools, you must close the gap produced by a skipped level to use the denormalized table effectively.

Because not all business intelligence tools require a skipped level to be closed, skipped level handling is optional. You use the same flag that indicates up-balancing or down-balancing for tree balancing to indicate processing of skip levels. The special mark for nodes that result from skip level processing is applied to the description field. The special mark is `<dds>`, where `<dd>` is the two-digit level number and `<s>` is the special character that you enter in the Skip Level Infix field on the Hierarchy Group Definition page.

Note. The special mark templates that you use for skip level balancing and for regular balancing can be the same or different than the other. For example, you can use `<ddb>` to refer to the result of balancing and `<dds>` to refer to the result of resolving a skip level, where `dd` refers to the level number, such as `03`, `b` refers to the balancing infix character, such as `~`, and `s` refers to the skip level infix character, such as `#`.

Skipping Levels Example

To provide an example of the skip-level process, consider the following summer tree:



Skip level summer tree

The following table represents the tree flattener result:

Tree Node	Entity ID	Parent Level Number	Child Level Number
A	F	1	0
A	G	1	0
A	H	1	0
B	F	2	0
B	G	2	0
C	F	3	0
D	G	3	0
E	H	3	0

If the selected special character for balancing is ~, and the selected special character for skip-level handling is #, and the option is down balancing, then the skip level result the following result occurs:

-	Row 1	Row 2	Row 3
Ent_ID	F	G	H
Ent_Desc	F Description	G Description	H Description
L31_ID	C	D	E
L31_Desc	31~C Description	31~D Description	31~E Description
...	-	-	-
L4_ID	C	D	E
L4_Desc	04~C Description	04~D Description	04~E Description
L3_ID	C	D	E
L3_Desc	C Description	D Description	E Description
L2_ID	B	B	A
L2_Desc	B Description	B Description	02#A Description
L1_ID	A	A	A
L1_Desc	A Description	A Description	A Description

Note. Due to space limitation, this example is rotated 90 degrees, with the table columns appearing in the rows.

Tree and Recursive Hierarchy Source Tables

The Tree and Recursive Hierarchy process can use as its source:

- A tree or recursive hierarchy originating in the source database and mirrored in the OWS.
- A tree or recursive hierarchy originating from the EPM database OWE.

Note. The Tree and Recursive Hierarchy process cannot process trees that contain a combination of dynamic details and range details. This combination may yield incorrect reporting results when it is used with business intelligence tools.

Enterprise Tree and Recursive Hierarchy Source Tables

Enterprise recursive hierarchy tables are typically data tables; therefore, bringing them into the EPM OWS is similar to the process of bringing any source data table into the OWS using the source to OWS ETL jobs. Before you run the tree denormalizer part of the Tree and Recursive Hierarchy process, if you are using the source database's trees or recursive hierarchies, you must either first bring your source database tree and recursive hierarchy definition and structure tables into EPM OWS, or you must ensure that your EPM trees and recursive hierarchies exist in the EPM database.

This table lists the PeopleSoft Enterprise tree source tables that you bring into the OWS before running the Tree and Recursive Hierarchy process:

<i>Source Database Tree Metadata Records</i>	<i>EPM OWS Tree Metadata Records</i>
PSTREESTRCT	S_TREESTRCT
PSTREESTRCTLANG	S_TREESTRCTLANG
PSTREEDEFN	S_TREEDEFN
PSTREEDEFNLANG	S_TREEDEFNLANG
PSTREENODE	S_TREENODE
PSTREELEAF	S_TREELEAF
TREE_NODE_TBL	S_TREE_NODE_TBL
TREE_NODE_LANG	S_TREE_NODE_LNG

Multilanguage Support for Relationship and Hierarchy Tables

If your EPM database supports multiple languages, then you may need to apply multilanguage capability in your relationship and hierarchy tables. You must have the language tables and also the corresponding outrigger tables. Language table names have an *L* prefix. Therefore, the name of your relationship language record is always prefixed with *LR_*, while the name of your hierarchy language record is always prefixed with *LH_*. The naming standard for outrigger tables is *O* prefix; therefore, *OR_* is the prefix for your relationship outrigger record, and *OH_* is the prefix for your hierarchy outrigger table.

The multilanguage support for relationship and hierarchy tables are not built-in within the predelivered EPM warehouses. You must extend the warehouse by creating maps that populate the language and outrigger tables. You must also modify your reports to use the relationship and hierarchy tables, as well as their language and outrigger tables.

See [Chapter 9, "Setting Up Multilanguage Processing and Running the Language Swap Utility," Understanding Multilanguage Processing, page 131.](#)

Understanding Tree and Recursive Hierarchy Process Results

This section discusses tree flattener and tree denormalizer output tables and tree flattener and tree denormalizer results.

Tree Flattener and Tree Denormalizer Output Tables

Running the Tree and Recursive Hierarchy process creates:

- A relationship table created by the flattening portion of the Tree and Recursive Hierarchy process.
- A hierarchy table (if you run the denormalization portion of the process).

In the hierarchy table:

- The Tree and Recursive Hierarchy process denormalizes each node of a winter tree to the detail level.
- The Tree and Recursive Hierarchy process denormalizes only the leaves in a summer tree to the detail level.
- An unbalanced level, skip level, or both, has a special character infix, which you specified on the Hierarchy Group Definition page, for example ~ , concatenated with the tree node ID in the denormalizer output table *TRDN*.

Output Relationship Tables

The output table structures of the MDW flattened tree or recursive hierarchy relationship data are similar for all relationship tables, except for the keys, which indicate whether the trees are setID, business unit, or user-defined based.

Relationship tables capture the parent-child relationship between an entity and its direct or indirect children in a hierarchy. For this reason, relationship tables always have a parent column and a child column. In addition, because the EPM tree and recursive hierarchy process also handles source database trees and recursive hierarchy and all setID, business unit, and user-defined based trees, the key sets are adjusted according the tree or recursive hierarchy that is being flattened. Also, certain keys are not required because they may not be relevant, depending on the source type, tree, or recursive hierarchy. The flag, `NODE_DET_FLAG`, is used to indicate whether the entry is a node or a detail in the tree or recursive hierarchy. This field has translate values, where *D* indicates that the entry is a detail and *N* indicates that the entry is a node.

The MDW relationship table is also effective-dated. When the source is a tree, the effective date of the relationship data is the tree effective date. When the source is a recursive hierarchy, the effective date of the relationship data is the recursive hierarchy process date.

Relationship tables for recursive hierarchies also store the driver record description. The driver record is the recursive hierarchy table or the underlying data table, in the case of recursive hierarchies that are based on the F0150 table. The record description is obtained from the PeopleSoft record definition table (PSRECDEFN).

Because relationship tables have a description field, the hierarchy processing creates a related language table for the relationship table. This related language table has all of the keys of the base relationship table, plus one additional key, `LANGUAGE_CD`. (The non-key field that is in the related language table for the relationship table is the description field.)

Note. Relationship tables always have a prefix *R_* to identify them.

The following table represents an output relationship table for *setID* based trees:

<i>Field Name</i>	<i>Type</i>	<i>Length</i>	<i>Key</i>	<i>Required</i>	<i>Edit</i>	<i>Prompt</i>	<i>Default</i>
SETID	Char	5	K	Y	N	N	None
SRC_SETID	Char	5	K	N	N	N	None
TREE_NAME	Char	18	K	Y	N	N	None
SRC_SYS_ID	Char	5	K	Y	N	N	None
EFFDT	Date	10	K	Y	N	N	None
TREE_NODE	Char	30	K	Y	N	N	None
CHILD_NODE_DTL	Char	30	K	Y	N	N	None
TREE_NODE_DESCR	Char	50	-	N	N	N	None
NODE_DETAIL_FLAG	Char	1	-	N	XLAT	N	None

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
RANGE_FR OM	Char	30	-	N	N	N	None
RANGE_T O	Char	50	-	N	N	N	None
TREE_LEV EL_NUM	Number	3	-	N	N	N	None
CHILD_LE VEL_NUM	Number	3	-	N	N	N	None
DATA_SRC _SYS_ID	Char	5	-	Y	N	N	None

The following table represents an output relationship table for *business unit* based trees:

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
BUSINESS _UNIT	Char	5	K	Y	N	N	None
SRC_BUSI NESS_UNI T	Char	5	K	N	N	N	None
TREE_NA ME	Char	18	K	Y	N	N	None
SRC_SYS_I D	Char	5	K	Y	N	N	None
EFFDT	Date	10	K	Y	N	N	None
TREE_NOD E	Char	30	K	Y	N	N	None
CHILD_NO D_DTL	Char	30	K	Y	N	N	None
TREE_NOD E_DESCR	Char	50	-	N	N	N	None
NODE_DE T_FLAG	Char	1	-	N	XLAT	N	None
RANGE_FR OM	Char	30	-	N	N	N	None
RANGE_T O	Char	30	-	N	N	N	None

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
TREE_LEV EL_NUM	Number	3	-	N	N	N	None
CHILD_LE VEL_NUM	Number	3	-	N	N	N	None
DATA_SRC _SYS_ID	Char	5	-	Y	N	N	None

The following table represents an output relationship table for *user-defined* (no key) trees:

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
SETCNTRL VALUE	Char	5	K	N	N	N	None
TREE_NA ME	Char	18	K	Y	N	N	None
SRC_SYS_I D	Char	5	K	Y	N	N	None
EFFDT	Date	10	K	Y	N	N	None
TREE_NOD E	Char	30	K	Y	N	N	None
CHILD_NO D_DTL	Char	30	K	Y	N	N	None
TREE_NOD E_DESCR	Char	50	-	N	N	N	None
NODE_DE T_FLAG	Char	1	-	N	XLAT	N	None
RANGE_FR OM	Char	30	-	N	N	N	None
RANGE_T O	Char	30	-	N	N	N	None
TREE_LEV EL_NUM	Number	3	-	N	N	N	None
CHILD_LE VEL_NUM	Number	3	-	N	N	N	None
DATA_SRC _SYS_ID	Char	5	-	Y	N	N	None

The following table represents an output relationship table for *setID* based *recursive hierarchy* trees:

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
SETID	Char	5	K	Y	N	N	None
SRC_SETID	Char	5	K	N	N	N	None
RECORD	Char	30	K	Y	N	N	None
SRC_SYS_ID	Char	5	K	Y	N	N	None
EFFDT	Date	10	K	Y	N	N	None
NODE	Char	30	K	Y	N	N	None
CHILD_NODE_DTL	Char	30	K	Y	N	N	None
NODE_DESCRIPTOR	Char	50	-	N	N	N	None
NODE_DELET_FLAG	Char	1	-	N	XLAT	N	None
LEVEL_NUMBER	Number	3	-	N	N	N	None
CHILD_LEVEL_NUM	Number	3	-	N	N	N	None
DATA_SRC_SYS_ID	Char	5	-	Y	N	N	None

The following represents an output relationship table for *business unit* based *recursive hierarchy* trees:

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
BUSINESS_UNIT	Char	5	K	Y	N	N	None
SRC_BUSINESS_UNIT	Char	5	K	N	N	N	None
RECORD	Char	30	K	Y	N	N	None
SRC_SYS_ID	Char	5	K	Y	N	N	None

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
EFFDT	Date	10	K	Y	N	N	None
NODE	Char	30	K	Y	N	N	None
CHILD_NO D_DTL	Char	30	K	Y	N	N	None
NODE_DES CR	Char	50	-	N	N	N	None
NODE_DE T_FLAG	Char	1	-	N	XLAT	N	None
LEVEL_NU M	Number	3	-	N	N	N	None
CHILD_LE VEL_NUM	Number	3	-	N	N	N	None
DATA_SRC _SYS_ID	Char	5	-	Y	N	N	None

The following table represents an output relationship table for *non business unit* and *non setID* based recursive hierarchy trees:

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
RECORD	Char	30	K	Y	N	N	None
SRC_SYS_I D	Char	5	K	Y	N	N	None
EFFDT	Date	10	K	Y	N	N	None
NODE	Char	30	K	Y	N	N	None
CHILD_NO D_DTL	Char	30	K	Y	N	N	None
NODE_DES CR	Char	50	-	N	N	N	None
NODE_DE T_FLAG	Char	1	-	N	XLAT	N	None
LEVEL_NU M	Number	3	-	N	N	N	None

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
CHILD_LE VEL_NUM	Number	3	-	N	N	N	None
DATA_SRC _SYS_ID	Char	5	-	Y	N	N	None

Output Hierarchy Tables

For hierarchy tables, the keys match the keys of the trees or recursive hierarchy, either setID, business unit, or user-defined key based. Hierarchy tables capture all of the parent IDs and descriptions of a detailed entity in a hierarchy. For this reason, they always have an entity ID as a key. The prepackaged Tree and Recursive Hierarchy ETL utility supports data processing for a hierarchy that is up to 32 levels deep, including one level for details. Therefore, it has 32 ID and description columns. The ID column is named L<n>_ID, where *n* is the hierarchy level. The description column is named L<n>_DESC.

Hierarchy tables have a description field for each of the supported levels for denormalization. (PeopleSoft supports 32 levels for a table.) Except for the entity ID and description, which are the lowest level, the ID and description are named L<n>_ID and L<n>_DESCR, where *n* is between 1 and 31, which is one less than the number of supported levels.

The hierarchical data is also effective-dated. When the source is a tree, the effective date of the hierarchy data is the tree effective date. When the source is a recursive hierarchy, the effective date of the hierarchy data is the recursive hierarchy process date. Like the relationship tables, hierarchy tables for recursive hierarchy also have the record description as a key. The record is the recursive hierarchy driver record. The description is obtained from the PeopleSoft record definition table (PSRECDEFN).

Because a hierarchy table has a description field, the process creates the related language table for the hierarchy table. The related language table for the hierarchy table has all of the keys of the base hierarchy table, plus one additional key called *LANGUAGE_CD*. (The non-key fields that exist in the related language table of the hierarchy table are the description fields.)

Note. Hierarchy tables are always prefixed with *H_* to identify them.

The following represents an output hierarchy table for *setID* based trees:

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
SETID	Char	5	K	Y	N	N	None
SRC_SETI D	Char	5	K	N	N	N	None
TREE_NA ME	Char	18	K	Y	N	N	None
SRC_SYS_I D	Char	5	K	Y	N	N	None
EFFDT	Date	10	K	Y	N	N	None

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
DTL_ID	Char	30	K	Y	N	N	None
DTL_DESC	Char	50	-	N	N	N	None
L<n>_ID	Char	30	-	N	N	N	None
L<n>_DESC	Char	50	-	N	N	N	None
...	-	-	-	-	-	-	-
DATA_SRC_SYS_ID	Char	5	-	Y	N	N	None

The following table represents an output hierarchy table for *business unit* based trees:

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
BUSINESS_UNIT	Char	5	K	Y	N	N	None
SRC_BUSINESS_UNIT	Char	5	K	N	N	N	None
TREE_NAME	Char	18	K	Y	N	N	None
SRC_SYS_ID	Char	5	K	Y	N	N	None
EFFDT	Date	10	K	Y	N	N	None
DTL_ID	Char	30	K	Y	N	N	None
DTL_DESC	Char	50	-	N	N	N	None
L<n>_ID	Char	30	-	N	N	N	None
L<n>_DESC	Char	50	-	N	N	N	None
...	-	-	-	-	-	-	-
DATA_SRC_SYS_ID	Char	5	-	Y	N	N	None

The following table represents an output hierarchy table for *user defined* (no key) trees:

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
SETCNTRL VALUE	Char	5	K	N	N	N	None
TREE_NA ME	Char	18	K	Y	N	N	None
SRC_SYS_I D	Char	5	K	Y	N	N	None
EFFDT	Date	10	K	Y	N	N	None
DTL_ID	Char	30	K	Y	N	N	None
DTL_DESC	Char	50	-	N	N	N	None
L<n>_ID	Char	30	-	N	N	N	None
L<n>_DES C	Char	50	-	N	N	N	None
...	-	-	-	-	-	-	-
DATA_SRC _SYS_ID	Char	5	-	Y	N	N	None

The following table represents an output hierarchy table for *setID* based *recursive hierarchy* trees:

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
SETID	Char	5	K	Y	N	N	None
SRC_SETI D	Char	5	K	N	N	N	None
RECORD	Char	30	K	Y	N	N	None
SRC_SYS_I D	Char	5	K	Y	N	N	None
EFFDT	Date	10	K	Y	N	N	None
DTL_ID	Char	30	K	Y	N	N	None
DTL_DESC	Char	50	-	N	N	N	None
L<n>_ID	Char	30	-	N	N	N	None
L<n>_DES C	Char	50	-	N	N	N	None

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
...	-	-	-	-	-	-	-
DATA_SRC _SYS_ID	Char	5	-	Y	N	N	None

The following table represents an output hierarchy table for *business unit* based *recursive hierarchy* trees:

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
BUSINESS _UNIT	Char	5	K	Y	N	N	None
SRC_BUSI NESS_UNI T	Char	5	K	N	N	N	None
RECORD	Char	30	K	Y	N	N	None
SRC_SYS_I D	Char	5	K	Y	N	N	None
EFFDT	Date	10	K	Y	N	N	None
DTL_ID	Char	30	K	Y	N	N	None
DTL_DESC	Char	50	-	N	N	N	None
L<n>_ID	Char	30	-	N	N	N	None
L<n>_DES C	Char	50	-	N	N	N	None
...	-	-	-	-	-	-	-
DATA_SRC _SYS_ID	Char	5	-	Y	N	N	None

The following table represents an output hierarchy table for *non business unit* and *non setID* based *recursive hierarchy* trees:

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
RECORD	Char	30	K	Y	N	N	None
SRC_SYS_I D	Char	5	K	Y	N	N	None
EFFDT	Date	10	K	Y	N	N	None

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
DTL_ID	Char	30	K	Y	N	N	None
DTL_DESC	Char	50	-	N	N	N	None
L<n>_ID	Char	30	-	N	N	N	None
L<n>_DESC	Char	50	-	N	N	N	None
...	-	-	-	-	-	-	-
DATA_SRC _SYS_ID	Char	5	-	Y	N	N	None

Related Language Tables

You use flattened (relationship) tables and denormalized (hierarchy) tables for business intelligence reporting; therefore, the tables have description fields. Thus, both types of tables have related language tables for multilanguage reporting. If your company does not require multilanguage processing, you do not have to populate the related language tables on the Relationship Record Definition and Hierarchy Record Definition pages. However, you must still define the relationship record name on the Relationship Record Definition page, and if you are running the denormalization part of the process, you must also define the hierarchy record name on the Hierarchy Record Definition page.

The Tree and Recursive Hierarchy process populates related language tables only if you specify a relationship language and outrigger record on the Relationship Record Definition page or a hierarchy language and outrigger record on the Hierarchy Record Definition page.

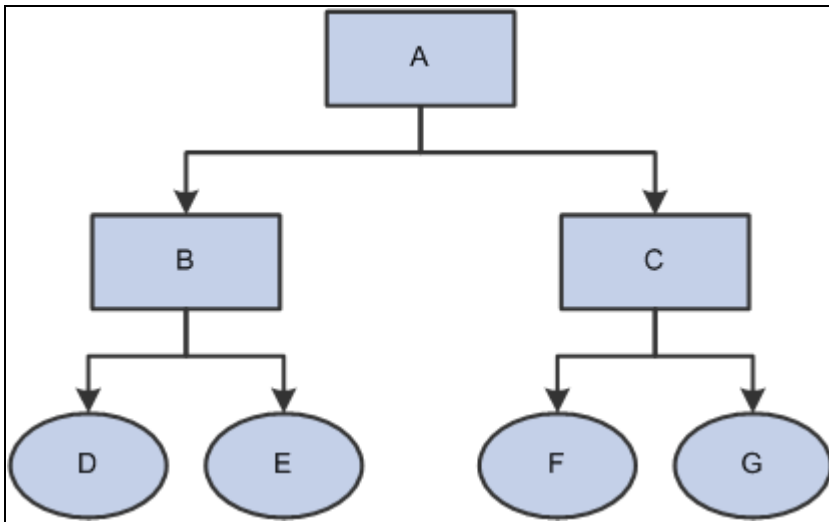
See [Chapter 9, "Setting Up Multilanguage Processing and Running the Language Swap Utility," Understanding Multilanguage Processing, page 131.](#)

Tree Flattener and Denormalizer Results

The output of flattening and denormalizing trees depends on the type of tree: summer or winter, balanced or unbalanced, skip level, and so on.

Summer Tree

This graphic shows an example of a summer tree before processing and without any balancing:



Example of a summer tree before processing

Processing of this tree results in the following relationship table:

<i>Tree Node</i>	<i>Entity ID</i>	<i>Parent Level Number</i>	<i>Child Level Number</i>
A	D	1	0
A	E	1	0
A	F	1	0
A	G	1	0
B	D	2	0
B	E	2	0
C	F	3	0
C	G	3	0

In a summer tree, the relationship table does not contain a tree node to itself or other node. Tree nodes only relate to the leaves. In addition, the child level numbers are always set to 0.

The following table shows the hierarchy table, without balancing:

<i>Entity</i>	<i>L31</i>	<i>L30</i>	<i>...</i>	<i>L2</i>	<i>L1</i>
D	-	-	-	B	A
E	-	-	-	B	A
F	-	-	-	C	A

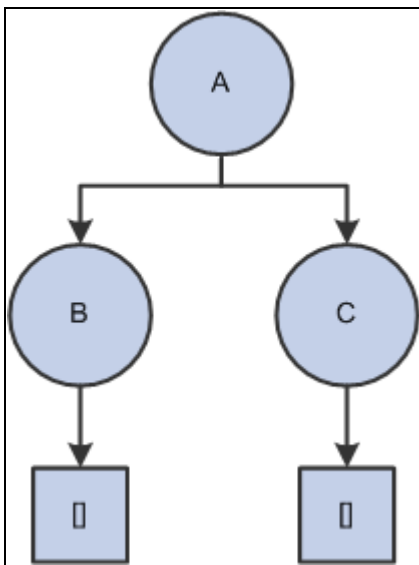
Entity	L31	L30	...	L2	L1
G	-	-	-	C	A

Note. In the previous example, the columns for levels 3 through 31 are not populated because the balancing option is turned off. If the balancing option were turned on, levels 3 through 31 would also be populated.

If your relationship or hierarchy tables require multilanguage support, then you must create the outrigger tables for the relationship and hierarchy tables.

Dynamic Summer Tree

This graphic shows an example of a dynamic summer tree (with relationships between departments and employees) before processing:



Dynamic summer tree

The following table represents the Department database table:

Department ID	Department Name
A	HR
B	Benefit
C	Payroll

The following table represents the Employee table:

Employee ID	Department ID	Employee Name
D	B	Jane Doe

<i>Employee ID</i>	<i>Department ID</i>	<i>Employee Name</i>
E	B	Joe Bloe
F	C	John Who

Processing of this tree results in the following relationship table:

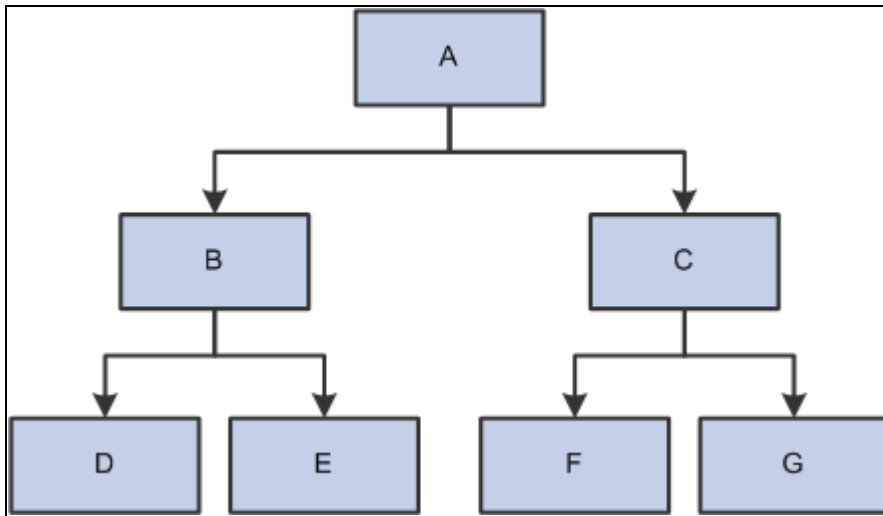
<i>Tree Node</i>	<i>Entity ID</i>	<i>Parent Level Number</i>	<i>Child Level Number</i>
A	D	1	0
A	E	1	0
A	F	1	0
B	D	2	0
B	E	2	0
C	F	2	0

The following table shows the hierarchy table, without balancing:

<i>Entity</i>	<i>L31</i>	<i>L30</i>	<i>...</i>	<i>L2</i>	<i>L1</i>
D	-	-	-	B	A
E	-	-	-	B	A
F	-	-	-	C	A

Winter Tree

This graphic shows an example of a winter tree before processing:



Example of a winter tree before processing

This table shows the winter tree relationship table after tree flattening:

<i>Tree Node</i>	<i>Entity ID</i>	<i>Parent Level Number</i>	<i>Child Level Number</i>
A	A	1	1
A	B	1	2
A	C	1	2
A	D	1	3
A	E	1	3
A	F	1	3
A	G	1	3
B	B	2	2
B	D	2	3
B	E	2	3
C	C	2	3
C	F	2	3
C	G	2	3
D	D	3	3
E	E	3	3

<i>Tree Node</i>	<i>Entity ID</i>	<i>Parent Level Number</i>	<i>Child Level Number</i>
F	F	3	3
G	G	3	3

In the relationship table, every node in a winter tree is associated with itself, as well as any nodes that are directly or indirectly under it.

This table shows the winter tree hierarchy after tree denormalizing:

<i>Entity</i>	<i>L31</i>	<i>L30</i>	<i>...</i>	<i>L3</i>	<i>L2</i>	<i>L1</i>
A						A
B					B	A
C					C	A
D				D	B	A
E				E	B	A
F				F	C	A
G				G	C	A

Recursive Hierarchy

The following table provides an example of a recursive hierarchy table:

<i>Entity ID</i>	<i>Entity Parent ID</i>
1	0
2	1
3	1
4	2
5	2

In a PeopleSoft recursive hierarchy, an Entity Parent ID = 0 implies that the entity is at the top of the hierarchy, if the entity is of numeric field. If the entity is a character field, then the highest-level entity will have Entity Parent ID = blank (that is, a space).

Because the lowest level entities in the hierarchy are of the same type as the parents, we can think of a recursive hierarchy like a winter tree. Therefore, the relationship and hierarchy table output from the Tree and Recursive Hierarchy ETL utility resembles the output format for a winter tree. That is, any node in the recursive hierarchy is associated with itself, as well as any nodes directly or indirectly under it.

<i>Parent Node</i>	<i>Entity</i>	<i>Parent Level Number</i>	<i>Child Level Number</i>
1	1	1	1
1	2	1	2
1	3	1	2
1	4	1	3
1	5	1	3
2	2	2	2
2	4	2	3
2	5	2	3
3	3	2	2
4	4	3	3
5	5	3	3

The following table shows the hierarchy table without balancing:

<i>Entity</i>	<i>L31</i>	<i>L30</i>	<i>...</i>	<i>L3</i>	<i>L2</i>	<i>L1</i>
1	-	-	-	-	-	1
2	-	-	-	-	2	1
3	-	-	-	-	3	1
4	-	-	-	4	2	1
5	-	-	-	5	3	1

Setting Up Parameters for Tree and Recursive Hierarchy Processing

Before you can run the actual Tree and Recursive Hierarchy ETL process, you must first define the parameters for process.

This section provides an overview of parameters for the Tree and Recursive Hierarchy process and discusses how to:

- Define the target and language tables for tree flattening.
- Define the target and language tables for tree denormalizing.

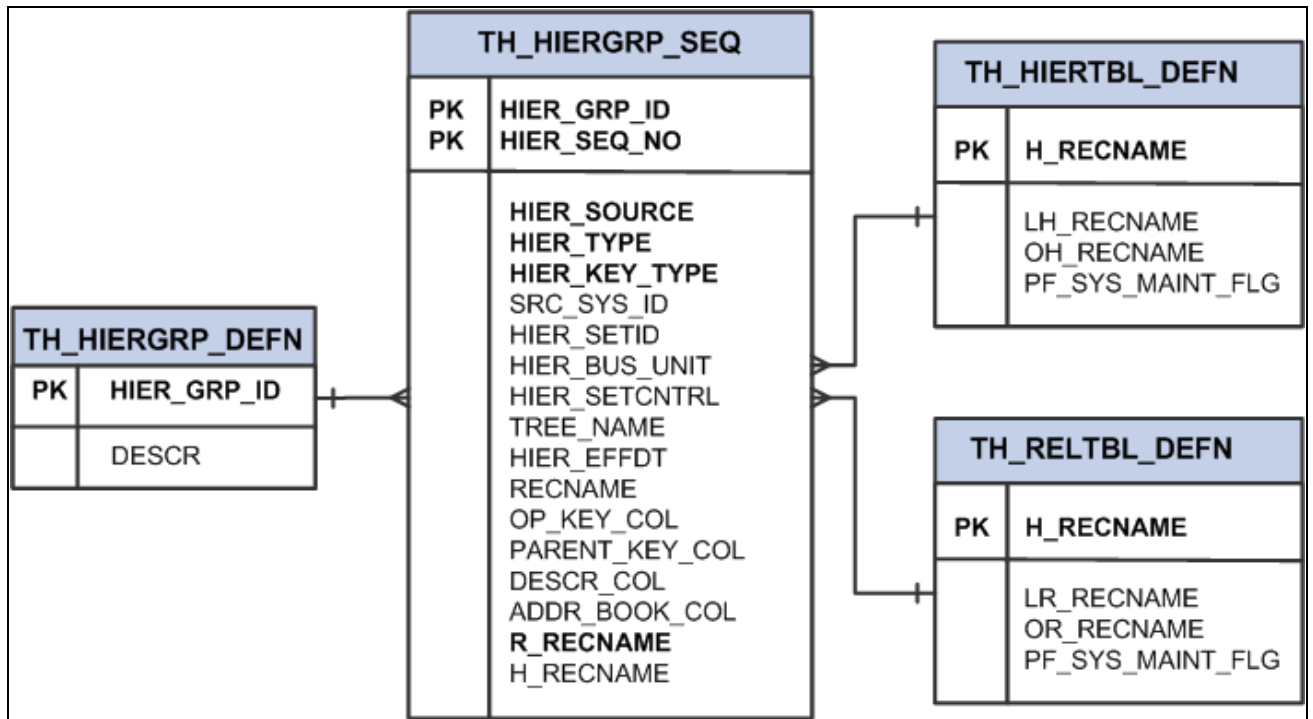
- Create the hierarchy group definition.

Defining Parameters for the Tree and Recursive Hierarchy Process

To run the Tree and Recursive Hierarchy process, use the Tree Hierarchy-Relational Table (TH_RELTBL_DEFN) component, Tree Hierarchy-Hierarchy Table (TH_HIERTBL_DEFN) component, and Tree Hierarchy-Hierarchy Group Definition (TH_HIERGRP_DEFN) component.

Because you must first flatten all hierarchies that are processed, you must define the *relationship table* that is the target for the flattening process. Because the denormalization process is optional, you must define the *hierarchy table* only if you intend to denormalize the flattened table.

The following diagram shows the Tree and Recursive Hierarchy processing setup pages:



Tree and Recursive Hierarchy processing setup pages

Pages Used to Run the Tree and Recursive Hierarchy Process

Page Name	Definition Name	Navigation	Usage
Relationship Record Definition	TH_RELTBL_DEFN	EPM Foundation, EPM Setup, Common Definitions, Hierarchy Group Definition, Relationship Table Definition	Define the target and language tables for tree flattening.

Page Name	Definition Name	Navigation	Usage
Hierarchy Record Definition	TH_HIERTBL_DEFN	EPM Foundation, EPM Setup, Common Definitions, Hierarchy Group Definition, Hierarchy Table Definition	Define the target and language tables for tree and hierarchy denormalizing.
Hierarchy Group Definition	TH_HIERGRP_DEFN	EPM Foundation, EPM Setup, Common Definitions, Hierarchy Group Definition, Hierarchy Group Definition	Enter parameters for the Tree and Recursive Hierarchy process.

Defining the Target and Language Tables for Tree Flattening

Access the Relationship Record Definition page (EPM Foundation, EPM Setup, Common Definitions, Relationship Group Definition, Relationship Record Definition).

Relationship Record Definition

Relationship record name R_ACCOUNT

Language & Outrigger Records

Relationship language record



Relationship outrigger record

Relationship Record Definition page

- Relationship record name** Displays the target record for the flattening portion of the Tree and Recursive Hierarchy process.
- Relationship language record** Enter the language record for this relationship table. This value is required only if this table is used for multilanguage processing.
- Relationship outrigger record** Enter the outrigger record for this relationship table. This value is required only if this table is used for multilanguage processing.

Defining the Target and Language Tables for Tree Denormalizing

Access the Hierarchy Record Definition page (EPM Foundation, EPM Setup, Common Definitions, Hierarchy Group Definition, Hierarchy Table Definition).

Hierarchy Record Definition	
Hierarchy record name	H_COMPCD
Language & Outtrigger Records	
Hierarchy language record	<input type="text" value="LH_COMPCD"/> 
Hierarchy outrigger record	<input type="text" value="OH_COMPCD"/> 

Hierarchy Record Definition page

Hierarchy record name Displays the target record for the denormalizing portion of the Tree and Recursive Hierarchy processing.

Hierarchy language record Enter the language record for this hierarchy table. You must enter this value only if this table is used for multilanguage processing.

Hierarchy outrigger record Enter the outrigger record for this hierarchy table. You must enter this value only if this table is used for multilanguage processing.

Creating the Hierarchy Group Definition

Access the Hierarchy Group Definition page (EPM Foundation, EPM Setup, Common Definitions, Hierarchy Group Definition, Hierarchy Group Definition).

Hierarchy Group Definition

Hierarchy Group ID	CRMH1
Description	CRM Recursive Hierarchy Group
*Hierarchy Balancing Rule	Down Balancing
Skip Level Infix	#
	Hierarchy Balancing Infix ~

Hierarchy Definition		Find View All	First	1 of 4	Last
*Hierarchy Sequence Number	1				
*Hierarchy Source	Non Current EPM Database				
*Hierarchy Type	Recursive hierarchy				
*Hierarchy Key Type	Business Unit based				
*Relationship record name	R_MKT_PROGRAM				
Hierarchy record name	H_MKT_PROGRAM				
Source System Identification	CRM				
Hierarchy Business Unit	APP01				
Record (Table) Name	RA_CAMPAIGN				
Operational Key Column	RA_CAMPAIGN_ID				
Parent Key Column	RA_ROLLUP_CMPGN_I				
Description column	RA_CMPGN_NAME				

Hierarchy Group Definition page

The Hierarchy Group Definition page contains a list of trees, recursive hierarchies, or both, that are related to a particular business process. For example, when you perform a workforce composition analysis, you must analyze data along organization, jobcode, and compensation code hierarchies. In this case, you can define the organization tree, jobcode tree, and compensation tree in one hierarchy group on the Hierarchy Group Definition page. Then, to perform workforce composition analysis, you need only to run tree processing using that hierarchy group ID as the parameter. When you run the Tree and Recursive Hierarchy process for that hierarchy group ID, the trees and recursive hierarchies that are associated with that ID are processed into either relational or hierarchical tables.

Note. The Hierarchy Group Definition page shown above is an example of this page using certain field values. If your field values differ, the fields that are available may be different. The following table of terms includes a list of all possible fields and the situations under which they display on this page.

Hierarchy Group ID	Displays the identifier for a group of trees, recursive hierarchies, or both, that relate to a specific business process that you intend to process into relationship or hierarchical tables. You can add a new hierarchy group or modify an existing hierarchy group for this hierarchy group ID.
---------------------------	--

Hierarchy Balancing Rule	Enter the balancing rule if the hierarchy is unbalanced. The options are: <ul style="list-style-type: none"> • <i>Up Balancing</i> • <i>Down Balancing</i> • <i>No Balancing</i>
Skip Level Infix	Enter the special character to indicate skip level nodes that result from the balancing process. You must enter this character only if you have selected <i>Up Balancing</i> or <i>Down Balancing</i> in the Hierarchy Balancing Rule field.
Hierarchy Balancing Infix	Enter the special character to indicate balancing nodes that result from the balancing process. You must enter this character only if you have selected <i>Up Balancing</i> or <i>Down Balancing</i> in the Hierarchy Balancing Rule field.
Hierarchy Sequence Number	Enter the sequence number within the hierarchy group ID for this tree or recursive hierarchy.
Hierarchy Source	Select the source database ID for this hierarchy. The options are: <p><i>Current EPM Database</i> (for OWE).</p> <p><i>Non Current EPM Database</i> (for OWS). This value refers to the PeopleSoft source database, as exists on the OWS.</p>
Hierarchy Type	Select the type of hierarchy. <p>The options are:</p> <p><i>Tree hierarchy.</i> When this option is selected, the Hierarchy effective date field displays.</p> <p><i>Recursive hierarchy.</i></p>
Hierarchy Key Type	Select the additional key type for this hierarchy. Options are: <ul style="list-style-type: none"> • <i>Business Unit based</i> • <i>No Additional Key Defined</i> • <i>SetID based</i> • <i>User defined</i>
Relationship record name	Enter the target table for the flattening part of the process that you identified on the Relationship Record Definition page.
Hierarchy record name	Enter the target table for the denormalizing part of the process that you identified on the Hierarchy Record Definition page. You must enter this value only if you are denormalizing the hierarchy.
Hierarchy SetID	Enter the setID for this hierarchy. This field is available only if the value in the Hierarchy Key Type field is <i>SetID</i> .

Hierarchy Business Unit	Enter the business unit for this hierarchy. This field is available only if the value in the Hierarchy Key Type field is <i>Business Unit based</i> .
User Defined Value	Enter the user-defined key value for this hierarchy. This field is available only if the value in the Hierarchy Key Type field is <i>User defined</i> .
Record (Table) Name	Enter the recursive hierarchy source table name for this hierarchy. This field is available only if the value in the Hierarchy Type field is <i>Recursive hierarchy</i> .
Tree Name	Enter the name of the tree for this process. This field is available only if the value in the Hierarchy Type field is <i>Tree hierarchy</i> .
Hierarchy effective date	Enter the effective date for the tree in Tree Manager.
Operational Key Column	Enter the column for the operational key for this hierarchy. This field is available only if the value in the Hierarchy Type field is <i>Recursive Hierarchy</i> .
Parent Key Column	Enter the column for the parent key for this hierarchy. This field is available only if the value in the Hierarchy Type field is <i>Recursive Hierarchy</i> .
Description Column	Enter the column for the description for this hierarchy. This field is available only if the value in the Hierarchy Type field is <i>Recursive Hierarchy</i> .
Source System Identification	Enter the name of the source for this hierarchy. This field is available only if the value in the Hierarchy Source field is <i>Non Concurrent EPM Database</i> .

Note. You can process multiple hierarchy definitions in one process. Use the + and – boxes on this page to add or subtract hierarchy definitions.

Running the Tree and Recursive Hierarchy ETL Process

After setting up the hierarchy parameters using the appropriate PeopleSoft Internet Architecture pages, you are ready to run the Tree and Recursive Hierarchy ETL process.

This section discusses how to run the tree and recursive hierarchy ETL process, which consist of the following steps:

1. Running OWS hierarchy jobs.
2. Running hash file hierarchy jobs.
3. Running hierarchy utility jobs.

Running Hash File Hierarchy Jobs

Perform the following steps to run the hash file hierarchy jobs:

1. In IBM WebSphere DataStage Director, navigate to the hash file hierarchy jobs by expanding the nodes in the left navigation panel, using the following path: *EPM90_Uilities, Tree_Recursive_Hierarchy, Load_Hash_Files*.
2. Select all the jobs in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time, and the job's status is updated to *Running*.

Running OWS Hierarchy Jobs

This step is required only when there are source database tree hierarchies.

Perform the following steps to run the OWS hierarchy jobs:

1. In IBM WebSphere DataStage Director, navigate to the OWS hierarchy jobs by expanding the nodes in the left navigation panel, using the following path: *EPM[release number]_Uilities, Tree_Recursive_Hierarchy, EsourceTree, StagingTreeMetadata, [warehouse name], Sequence*.
2. Select all jobs in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the job parameters if necessary and click Run.

The job is scheduled to run with the current date and time, and the job's status is updated to *Running*.

Running Hierarchy Utility Jobs

Perform the following steps to run the hierarchy utility jobs:

1. In IBM WebSphere DataStage Director, navigate to the *J_Hierarchy_Startup_Process* job by expanding the nodes in the left navigation panel, using the following path: *EPM90_Uilities, Init_Process, Init_Process*.
2. Select the job in the Job Status view and select *Job, Run Now...* from the menu.

The Job Run Options box appears.

3. Update the Hierarchy Group ID and Hierarchy Sequence Number job parameters with the appropriate values.

To process all the hierarchies under a single group, provide the appropriate numerical value for the group ID. The sequence number can be left blank.

To process a single hierarchy, both the Hierarchy Group ID and Hierarchy Sequence Number should be given while running the job.

4. Use the Populate Language Data field to specify whether you want to insert related language data in hierarchy and relationship related language tables.

5. Click Run.

The job is scheduled to run with the current date and time, and the job's status is updated to *Running*.

Chapter 11

Extending the Multidimensional Warehouse Data Model

This chapter provides considerations for modifying an EPM warehouse and discusses:

- Add a fact or dimension table to the MDW data model.
- Extend a fact table in the MDW data model.
- Extend a dimension table in the MDW data model.

Note. The procedures discussed in this chapter are not supported by PeopleSoft, they are merely a guide for extending the MDW data model.

Considerations for Modifying an EPM Warehouse

The prepackaged MDW data model is designed to facilitate the introduction of a new dimension, a new attribute on an existing dimension table, a new dimension and/or a new fact based on an existing fact table. Any modifications that you make to the overall solution, including data mart content, will affect the reporting results. Consider the following questions when modifying the data model and developing reports.

Before modifying the data model, determine:

- What business intelligence reporting tool you will use— relational online analytic processing (ROLAP) or multidimensional online analytic processing (MOLAP).
- Whether you must build aggregate fact tables (if using ROLAP).

If so, determine what aggregate levels of each dimension to use for this fact table.

- The typical profile of your end-users.

Evaluate Dimension Requirements

To ensure that the dimensions meet the needs of your business, determine:

- The dimensions in the delivered data mart that you want to keep and the ones that you can eliminate.
- The dimensions that you must create that are not part of the delivered data mart.
- The changes that you must make to delivered dimensions.

If you alter an existing dimension or add a new dimension, determine:

- Whether this is a dimension or merely an attribute of an existing dimension.
- The hierarchies in this dimension.
- The hierarchy levels in each hierarchy of this dimension.
- The attributes of this dimension.
- Whether you are altering the lowest level of an existing dimension.
- Whether you have facts at this level.
- Whether your multidimensional analysis will still work.

Evaluate Measure Requirements

To ensure that the measures you use meet the needs of your business, determine:

- The measures in the delivered data mart to keep and those to eliminate.
- The measures that you must create that are not part of the delivered data mart.
- What changes you must make to delivered measures, such as changing calculations or populating required fields.

If you alter an existing measure or add a new measure, determine:

- Whether they are base measures that you can store on a row by row basis, or are runtime calculations that you must define in the reporting tool.
- What dimensions qualify this measure.
- Whether this measure is an addition to an existing fact table, or must be part of a new fact table.
- Whether you will use this measure along with other measures on the same report.

Adding a Fact or Dimension Table to the Multidimensional Warehouse Data Model

You can extend the analytic capabilities of the prepackaged Multidimensional Warehouse (MDW) data model by adding new fact and dimension tables. New fact tables can become necessary when you introduce new sources of data to the data warehouse and/or new business processes are desired for analytic analysis.

Steps Required to Add a New Fact Table

The following steps are required to add a new fact table to the MDW data model:

1. Select the business process to be modeled.
2. Decide the grain for the new fact table.
3. Choose the dimensionality of the new fact table.

4. Identify the facts to be represented on the new fact table.
5. Identify the source of the new content and its corresponding error table.
6. Design the table structure required and apply it to the database.
7. Create new fact ETL job.
8. Declare the category or mart to which the fact applies.
9. Modify the applicable master sequence to include the new ETL fact job.

Note. This step is discussed in more detail below.

Use the following path in the left navigation panel of IBM WebSphere DataStage Director to locate the fact master sequence referenced in step nine above: *E, [business process], [data mart], Master_Sequence*.

Steps Required to Add a New Dimension Table

The following steps are required to add a new dimension table to the MDW data model:

1. Choose the new dimensionality desired and determine whether there are any corresponding related or outrigger language requirements.
2. Identify the source of the new content and its corresponding error table.
3. Design the table structure required and apply to the database.
4. Create new dimension ETL jobs (including any corresponding related or outrigger language jobs).
5. Declare the category or mart to which the dimension applies.
6. Modify the master sequence for this mart to include the new ETL dimension job as described in this section.
7. Modify the applicable master sequence to include the new ETL dimension job.

If required, the Language dimension sequence must be modified, as well.

Note. This step is discussed in more detail below.

There are three types of dimension master sequences that must be modified, as referenced in step seven above:

- Global Dimension Master Sequence.
- Local Dimension Master Sequence
- Data Mart Business Process Dimension Master Sequence

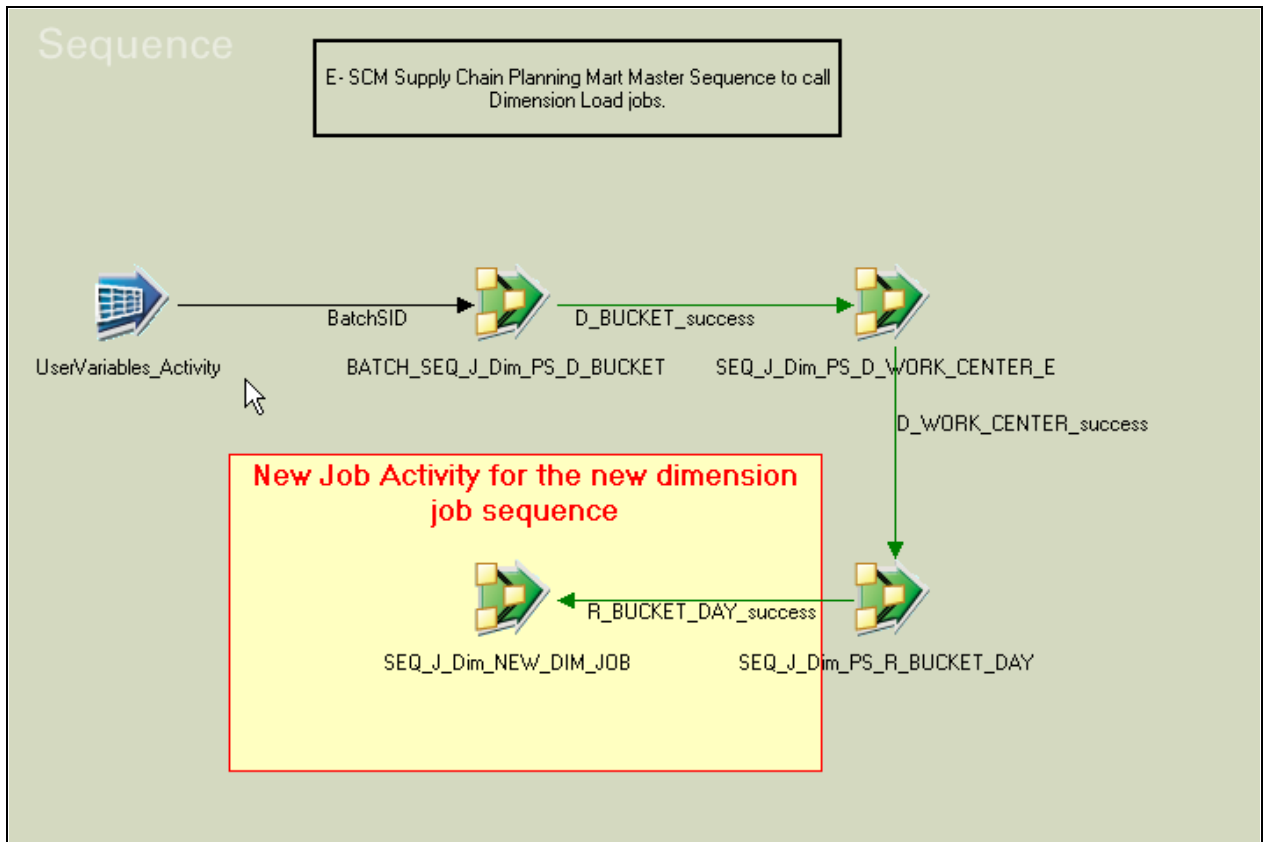
Use the following path in the left navigation panel of IBM WebSphere DataStage Director to locate the applicable dimension master sequence: *E, [business process], [data mart], Master_Sequence*.

Adding a New Fact or Dimension Job to a Master Sequence

Perform the following steps to add a new fact or dimension job to a master sequence:

1. In IBM WebSphere DataStage Designer, locate the appropriate fact or dimension master sequence using the navigation provided in the preceding sections.
2. Open the master sequence for editing.
3. Add your new job (as a *job activity*) to the master sequence.

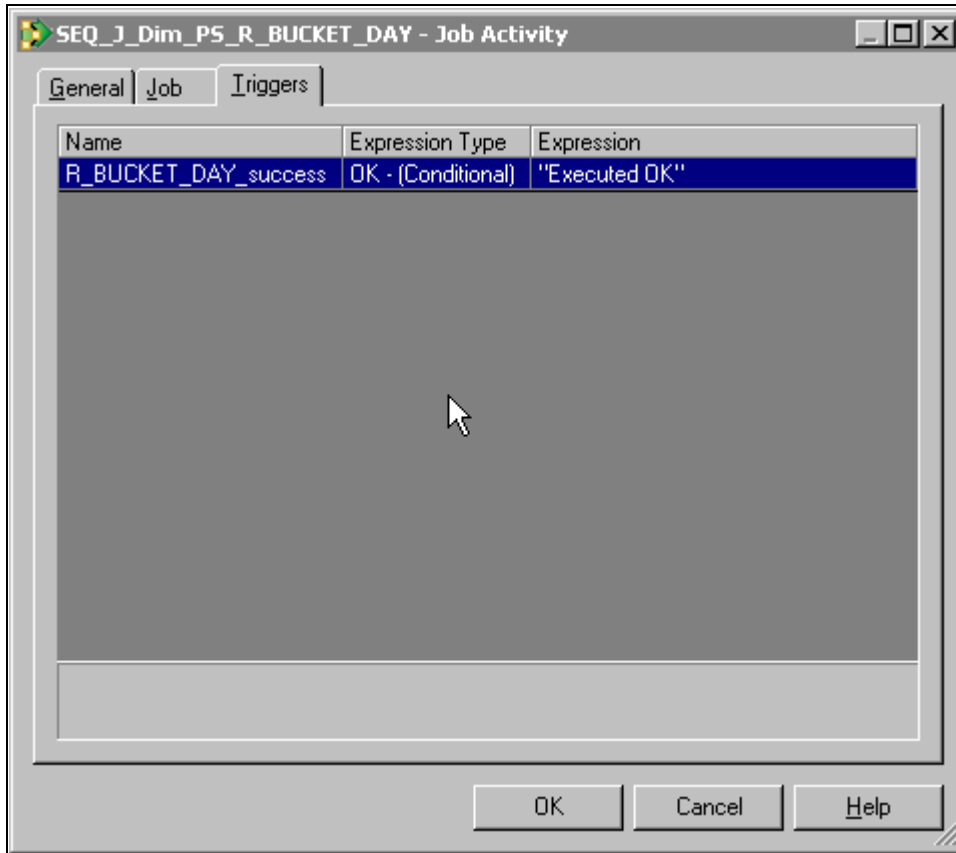
Drag the new job from the IBM WebSphere DataStage Designer Repository window and drop it in the Diagram window. The job appears as an activity in the Diagram window.



Add new job to master sequence

4. Connect the new job activity to the existing activity using a *trigger*.

5. In the *Triggers* tab of the properties box, edit the expression of the output trigger as appropriate.



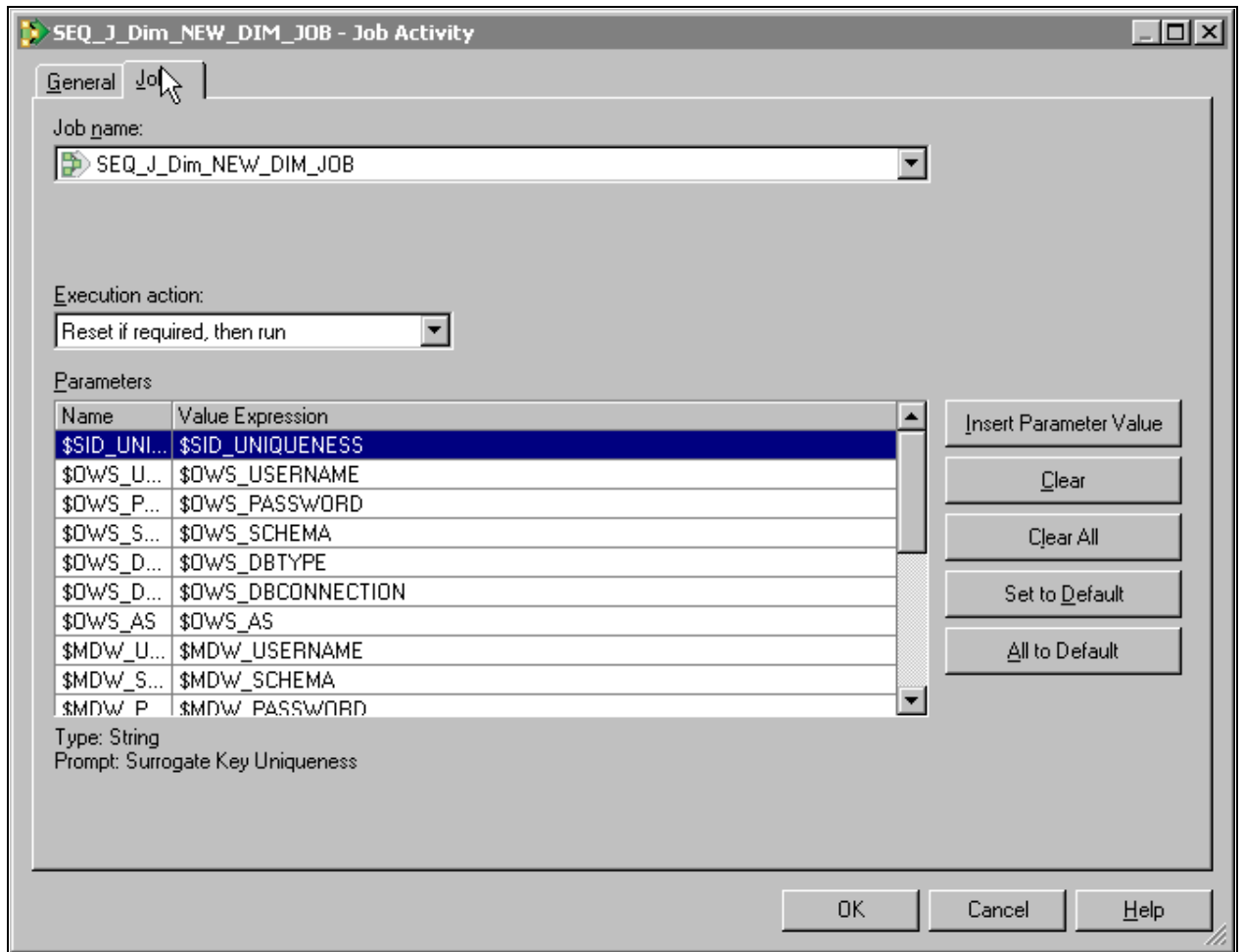
Job Activity box with Triggers tab selected

By default, the expression type on the triggers is set to *OK – (Conditional)*. This condition can be modified if you have a different business requirement.

6. Open the new job activity for editing.

The job activity property box appears.

- In the Jobs tab, change the job name to reflect the name of the new dimension sequence job and modify the value expression in the parameters as appropriate.



Job Activity box with Job tab selected

- Select *File, Save* from the menu to save the job.
- Select *File, Compile* from the menu to compile the job.

If your mapping is correct, the Compilation Status window displays the *Job successfully compiled with no errors* message. If your mapping is incorrect, the Compilation Status window displays an error message.

- If your job successfully compiles, select Close.

If your job does not compile successfully, you must return to the job and troubleshoot the errors.

- You should perform technical unit testing and regression testing on the master sequence to ensure that each job activity is executed properly.

Extending a Fact Table in the Multidimensional Warehouse Data Model

This sections discusses how to add a new measure and surrogate key to a fact table.

Adding a New Measure to a Fact Table

You can extend the functionality of a fact table by introducing new measures to it. To load a new measure into a fact table, you must extract a new field from either an existing source table or a new source table.

When the new measure is available at the same granularity as the existing measures, the new measure is populated in the same manner. If the new measure occurs at a granularity higher than is represented by the existing fact table, it must be allocated to the appropriate level of detail represented by the existing fact table. If there is no business logic that can be applied as an allocation rule, you must create a new fact table.

The following steps are required to add a new measure to a fact table:

1. Define the new measure desired.
2. Identify the source of the new content and its corresponding error table.
3. Assess the impact to the granularity with respect to the existing fact table being considered for extension.
4. Design the table structure modifications required and apply them to the database.
5. Update the fact ETL job to include the new measure.

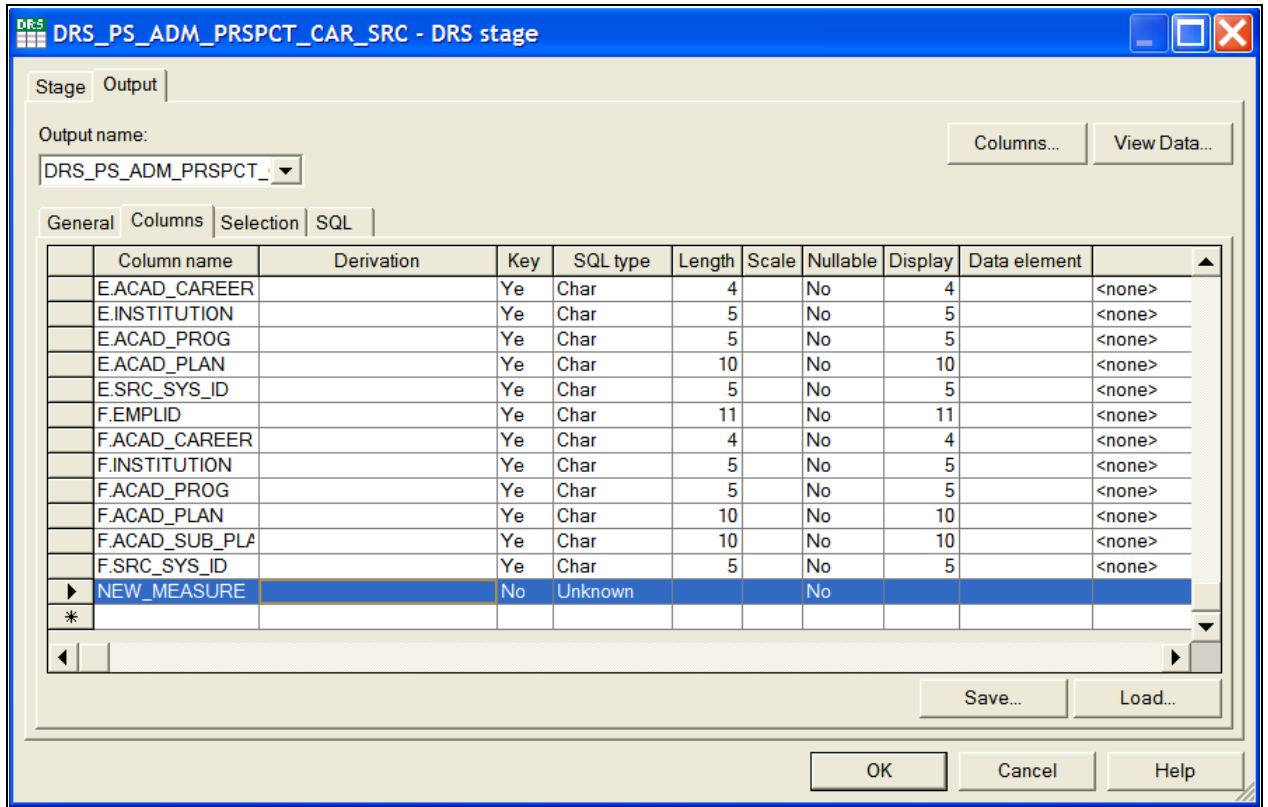
Note. This step is discussed in more detail below.

Updating a Fact Job with a New Measure That Originates From the Same Source Table

Perform the following steps to update a fact job with a new measure that originates from the same source table:

1. In IBM WebSphere DataStage Designer, locate the appropriate fact job and open it for editing.
2. Open the *source DRS stage* and select the Output tab.

- In the Columns sub-tab, add a new row for the new measure.



Adding a new row for the new measure

- Input the appropriate values for the derivation, key, SQL type, and other applicable properties of the new measure.
- Repeat steps two through four for all the stages between the source DRS and the target DRS, but provide information for the Input tab as well as the Output tab.

Once the new attribute is defined in the IPC stage, it becomes available on the Transformer Stage - Input Links window.

- In the Transformer Stage - Input Links window, apply any transformation logic, such as any string or number functions, as necessary.

The logic is defined in the derivations field of the output link for the target table.

- Link all ports as necessary.
- Open the target DRS stage and select the Input tab.
- In the Columns sub-tab ensure that the new measure column is present and properly defined.
- Select *File, Save* from the menu to save the job.
- Select *File, Compile* from the menu to compile the job.

If your mapping is correct, the Compilation Status window displays the *Job successfully compiled with no errors* message. If your mapping is incorrect, the Compilation Status window displays an error message.

12. If your job successfully compiles, select Close.

If your job does not compile successfully, you must return to the job and troubleshoot the errors.

13. You should perform technical unit testing and regression testing on the server job to ensure that the new measure is populated properly.

Updating a Fact Job with a New Measure That Originates From a New Source Table

Perform the following steps to update a fact job with a new measure that originates from a new source table:

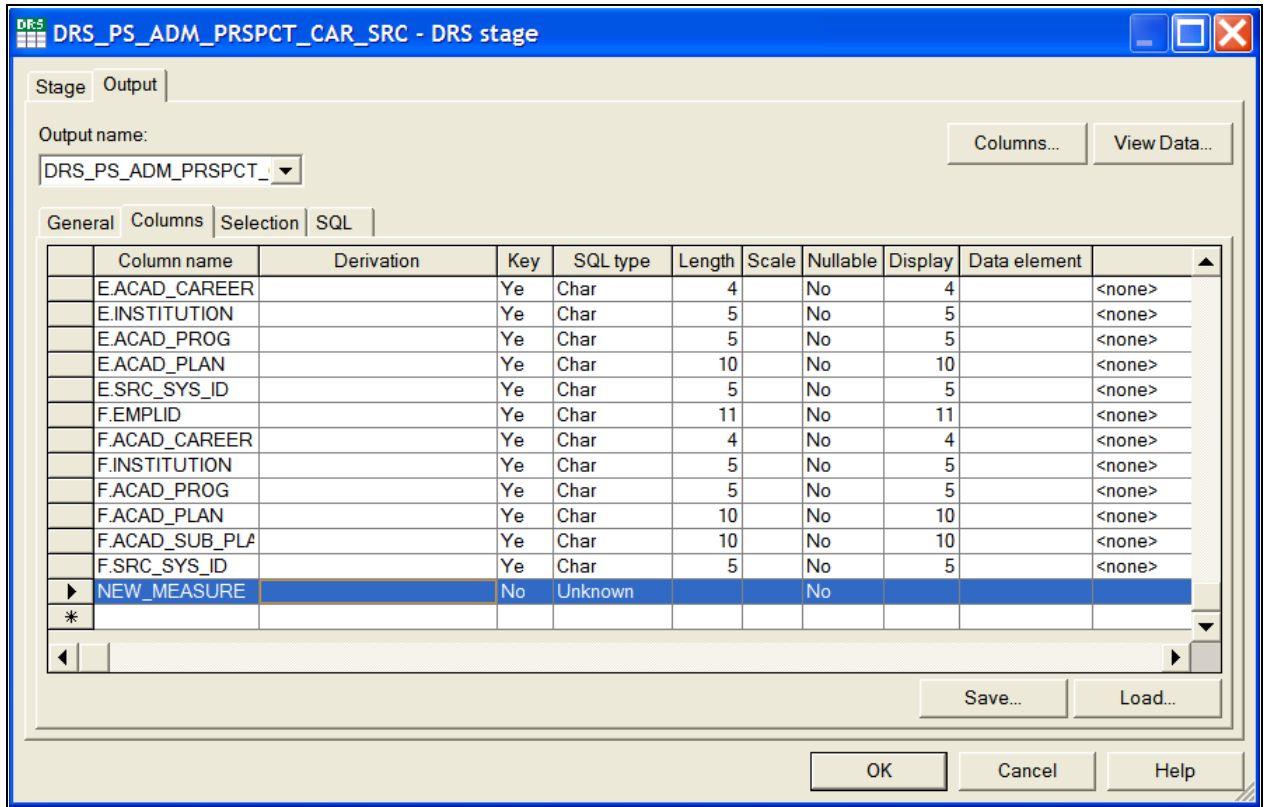
1. In IBM WebSphere DataStage Designer, locate the appropriate fact job and open it for editing.
2. Open the *source DRS stage* and select the Output tab.
3. In the General sub-tab, define the new source table.

Input the appropriate values for the table name, transaction isolation, array size, and query type. You can give table aliases to the source tables for use in defining join conditions and column derivations, and the query type can be user defined or generated by SQL.

The screenshot shows the 'Output' tab of the DataStage Designer interface. The 'Output name' is 'DRS_PS_EN_WORK_CI'. Below it, the 'General' sub-tab is active. The 'Table names' field contains 'INTER A, #OWS# SCHEMA#PS_NEW_SOURCE_TBL B'. The 'Transaction Isolation' is set to 'Read Committed'. The 'Array size' is '#OWS_AS#' and the 'Query type' is 'Generated SQL query'. There is also a 'Description' field which is currently empty.

General sub-tab with new source table information

4. Select the Columns sub-tab and add a new row for the new measure.



Adding a new row for the new measure

5. Input the appropriate values for the derivation, key, SQL type, length, scale, and other applicable properties of the new measure.

The derivations of the columns must indicate the source table alias for each field.

6. Repeat steps two through four for all the stages between the source DRS and the target DRS, but provide information for the Input tab as well as the Output tab.

Once the new attribute is defined in the IPC stage, it becomes available on the Transformer Stage - Input Links window.

7. In the Transformer Stage - Input Links window, apply any transformation logic, such as any string or number functions, as necessary.

The logic is defined in the derivations field of the output link for the target table.

8. Link all ports as necessary.
9. Open the target DRS stage and select the Input tab.
10. In the Columns sub-tab ensure that the new measure column is present and properly defined.
11. Select *File, Save* from the menu to save the job.

12. Select *File, Compile* from the menu to compile the job.

If your mapping is correct, the Compilation Status window displays the *Job successfully compiled with no errors* message. If your mapping is incorrect, the Compilation Status window displays an error message.

13. If your job successfully compiles, select Close.

If your job does not compile successfully, you must return to the job and troubleshoot the errors.

14. You should perform technical unit testing and regression testing on the server job to ensure that the new measure is populated properly.

Adding a New Surrogate Key to a Fact Table

The MDW data model enables you to add new dimension tables to it. If you add a new dimension table, you must update the corresponding fact table with the primary/foreign key relationship.

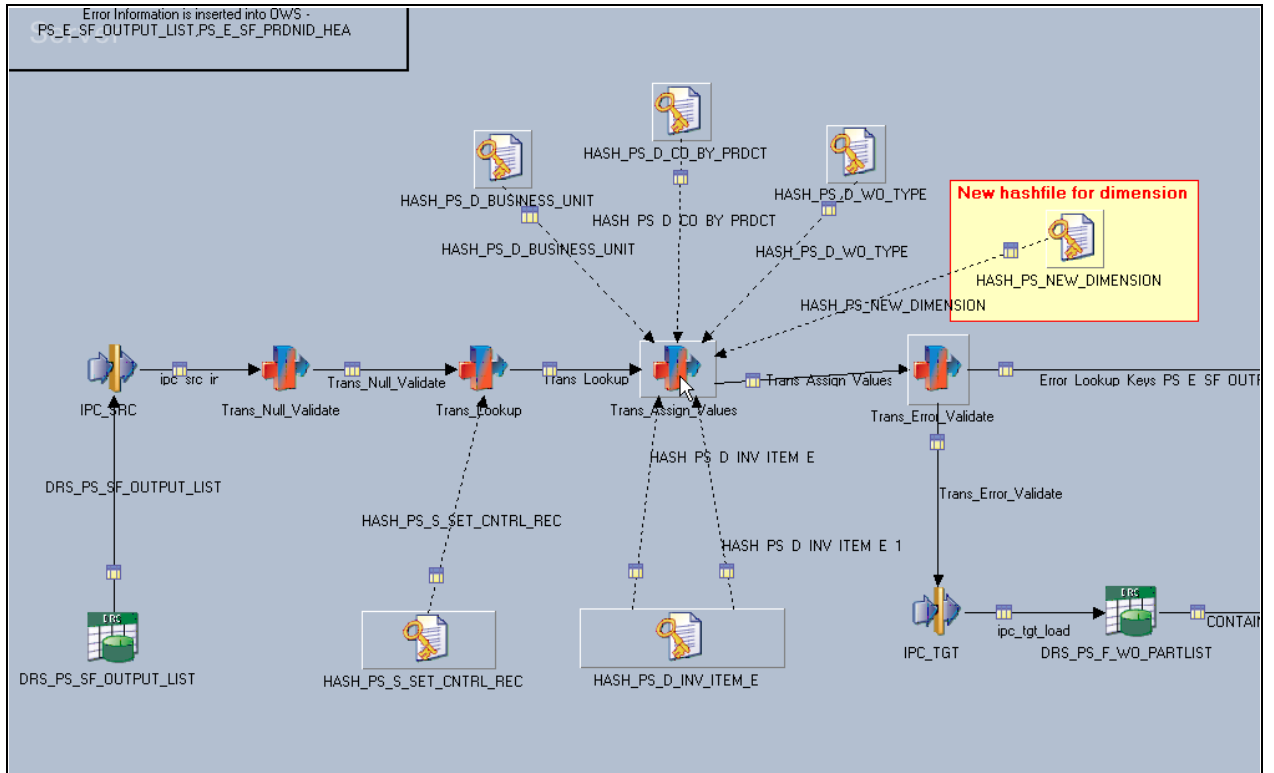
The dimension can be associated with the fact table by adding a new surrogate ID (SID) field (the foreign key field) and populating it appropriately with values of the primary key from the associated dimension. To populate a new SID field in a fact table, a new lookup must be performed on a dimension table hash file. Performing a lookup on a dimension table requires a field from the source to be joined with the key fields of the dimension hash file. Existing fields from the source can be used for the join or a new field must be extracted from the source for the join.

Steps to Add a New Surrogate Key to a Fact Table

Perform the following steps to add a new surrogate key:

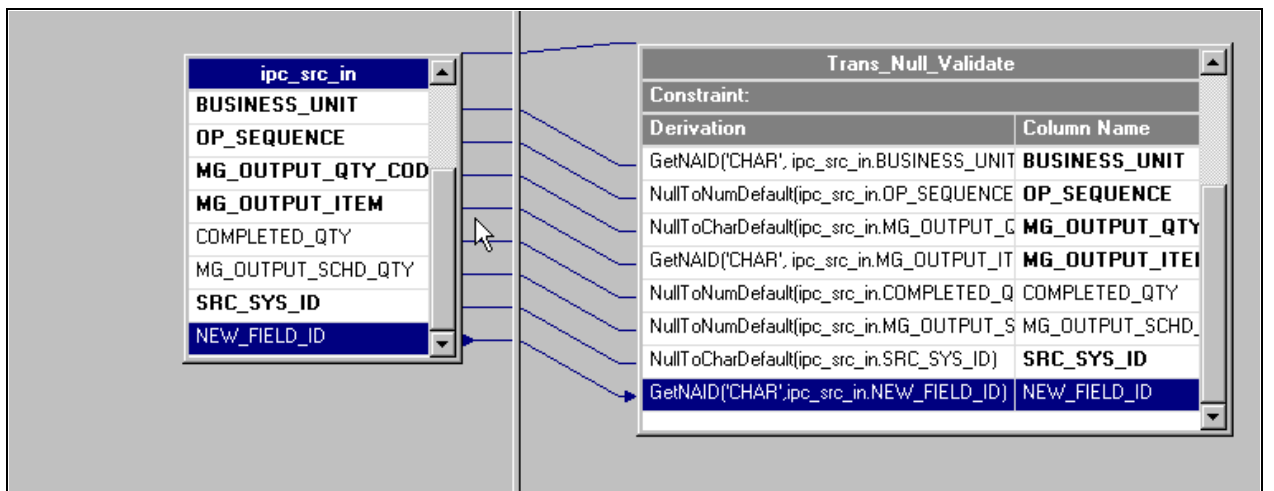
1. In IBM WebSphere DataStage Designer, locate the appropriate fact job and open it for editing.

2. Add a new *dimension table hash file* for the new dimension you have added to the data model.



Dimension table hash file added

3. Open the first transformer stage that follows the source (usually the *Trans_Null_Validate* stage).

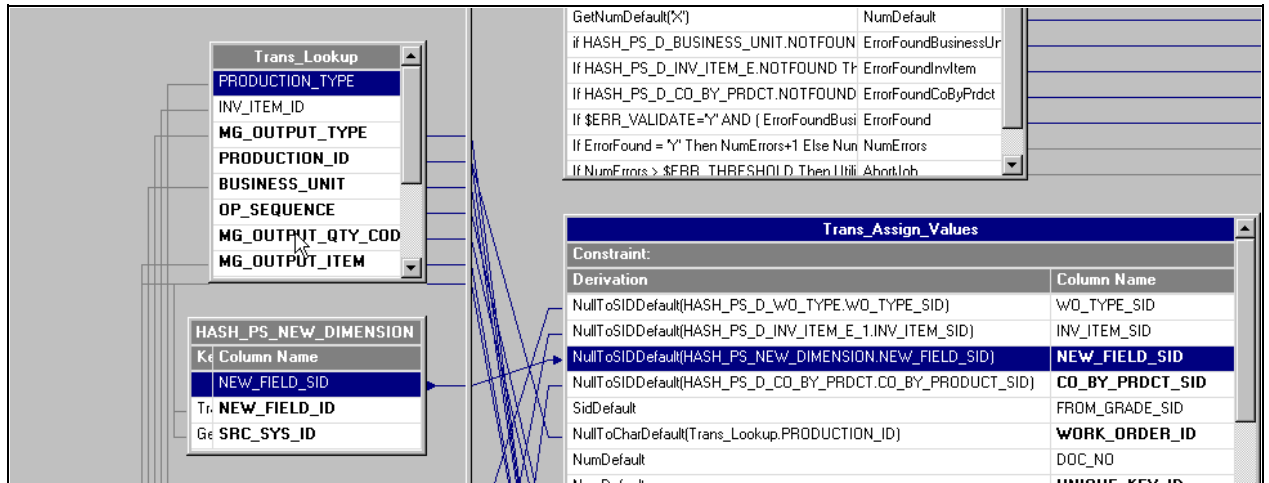


Trans_Null_Validate Stage

4. The new field must be processed by the *GetNAID* routine.

- Once the key fields of the dimension table are matched with the input fields, the SID is extracted to the target SID field in the fact table.

This field must be validated by the *NullToSIDDefault* routine in case the lookup results in a null value.



SID extract

- Select *File, Save* from the menu to save the job.
- Select *File, Compile* from the menu to compile the job.

If your mapping is correct, the Compilation Status window displays the *Job successfully compiled with no errors* message. If your mapping is incorrect, the Compilation Status window displays an error message.

- If your job successfully compiles, select Close.

If you job does not compile successfully, you must return to the job and troubleshoot the errors.

- You should perform technical unit testing and regression testing on the server job to ensure that the new measure is populated properly.

Extending a Dimension Table in the Multidimensional Warehouse Data Model

You can extend the functionality of a dimension table by introducing new attributes to it. To load a new attribute into a dimension table, you must extract a new field from either a source table or a new lookup table. This new field can then go through any required transformation before loading to the dimension table.

The following steps are required to extend a dimension table:

- Define the new attribute desired and determine whether there are any corresponding related or outrigger language requirements for that attribute.
- Identify the source of the new content and its corresponding error table.

3. Assess the impact to the granularity with respect to the existing dimension table being considered for extension.
4. Design the table structure modifications required and apply them to the database.
5. Update the dimension job to include the new attribute.

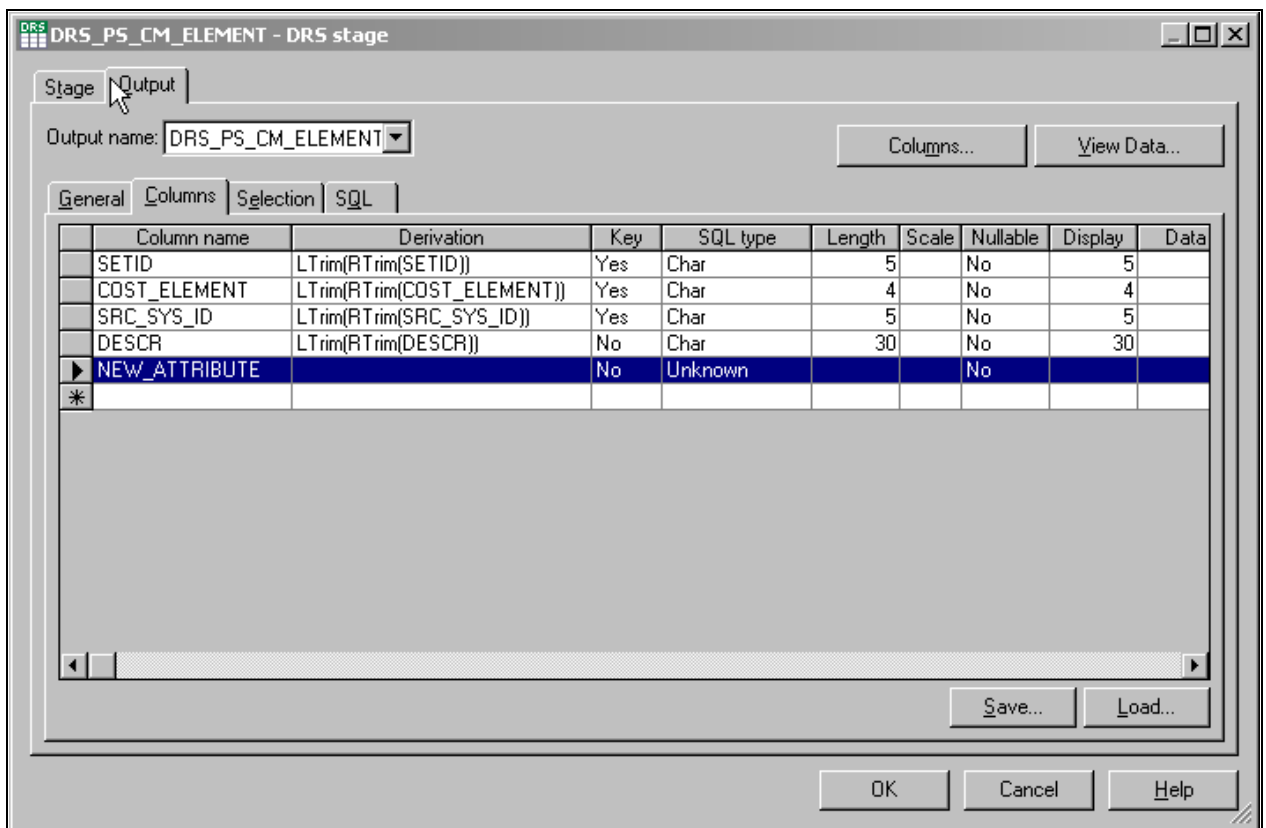
Note. This step is discussed in more detail below.

6. Update corresponding related language or outrigger language jobs, as necessary.

Updating a Dimension Job with a New Attribute That Originates from a Source Table

Perform the following steps to update a dimension job with an attribute that originates from a source table:

1. In IBM WebSphere DataStage Designer, locate the appropriate dimension job and open it for editing.
2. Open the *DRS stage* and select the Output tab.



DRS Stage with Columns sub-tab selected

3. In the Columns sub-tab, add a new row for the new attribute.
4. Input the appropriate values for the derivation, data type, data size, and other applicable properties of the new attribute.

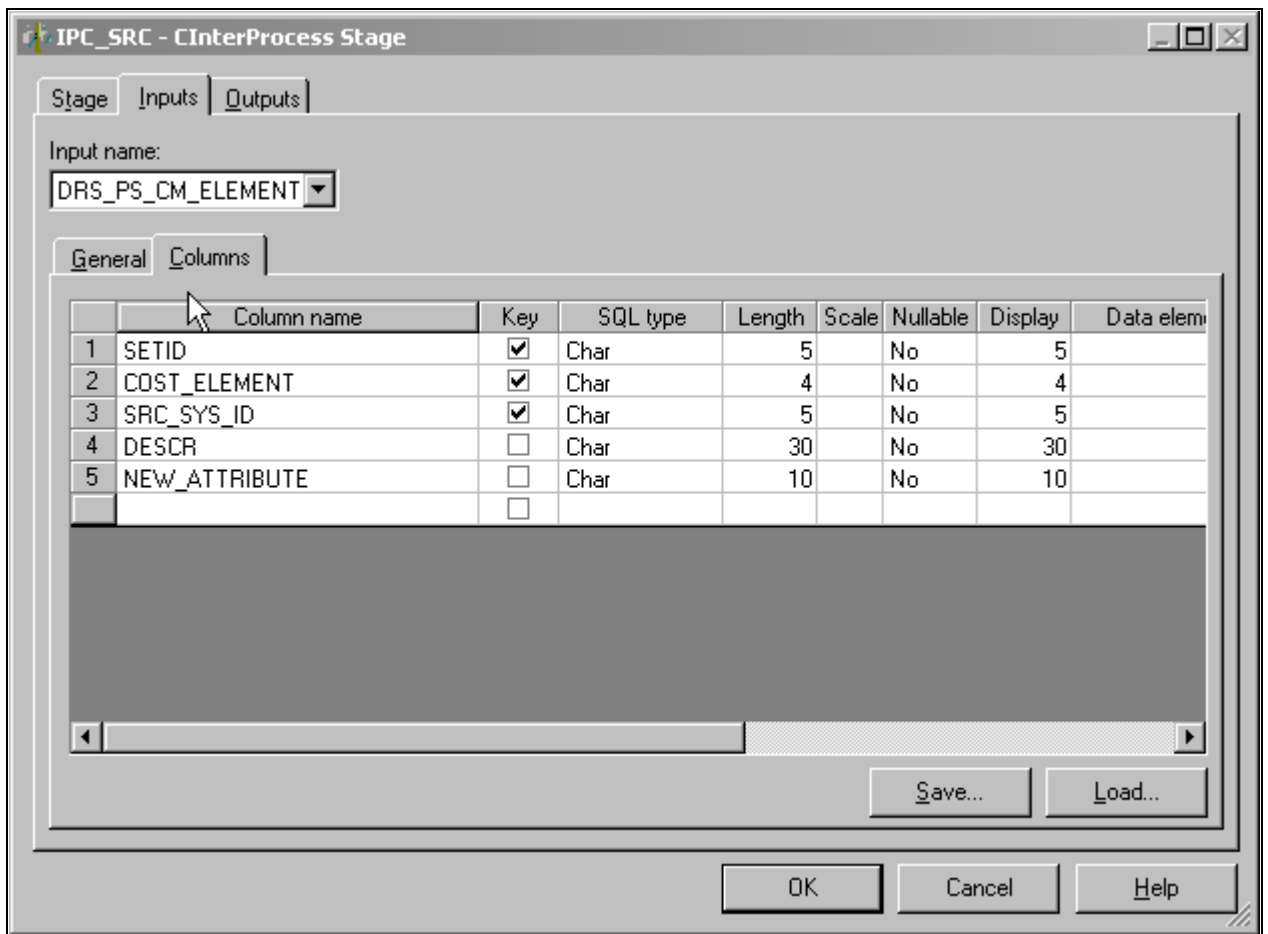
- Repeat steps two through four for the *IPC stage* but provide information for the Input tab as well as the Output tab.

Once the new attribute is defined in the IPC stage, it becomes available on the Transformer Stage - Input Links window.

- In the Transformer Stage - Input Links window, apply any transformation logic, such as any string or number functions, as necessary.

The logic is defined in the derivations field of the output link for the target table.

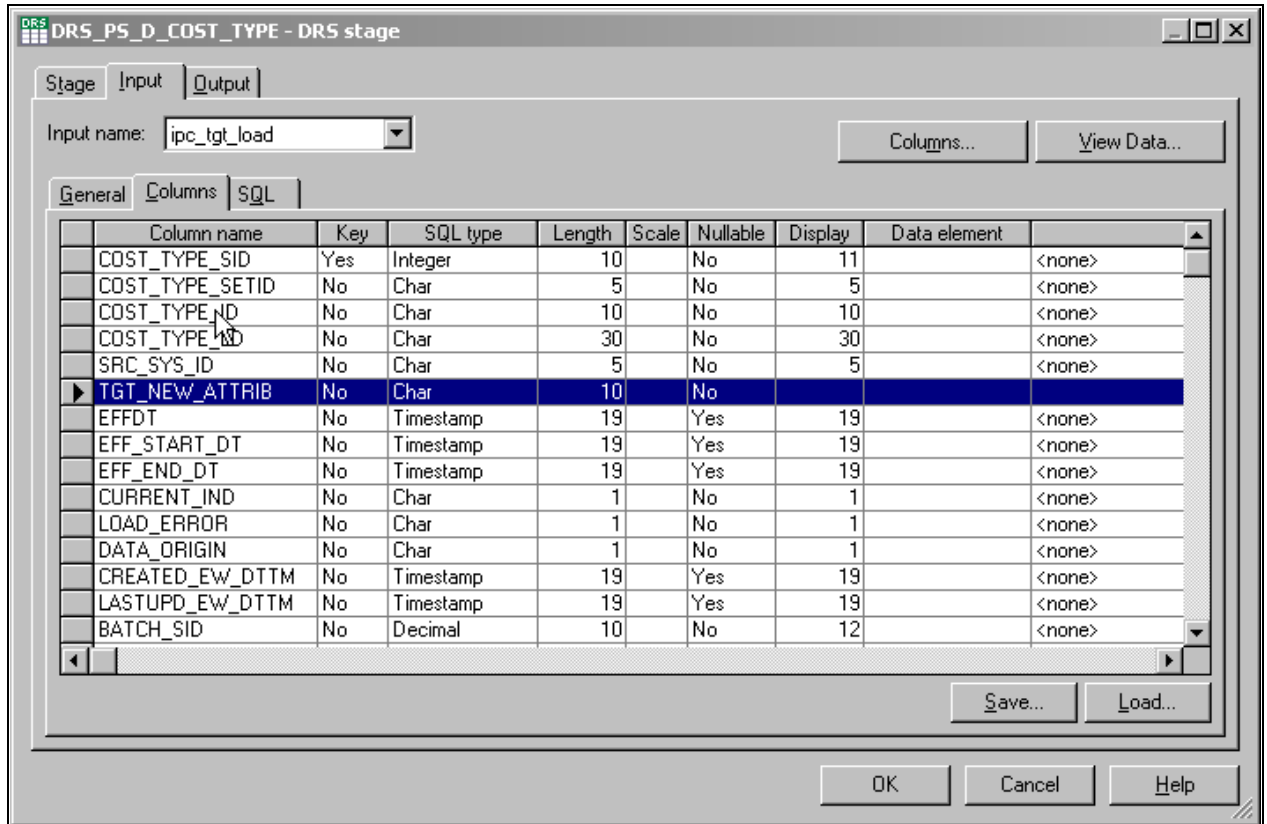
- Connect the output link of the transformer stage to the target dimension table.
- Open the IPC stage and select the Inputs tab.



IPC_SRC Stage

- In the Columns sub-tab ensure that the new attribute column is present and properly defined.
- Open the target DRS stage and select the Input tab.

11. In the Columns sub-tab ensure that the new attribute column is present and properly defined.



Target DRS stage with new attribute row

12. Select *File, Save* from the menu to save the job.
13. Select *File, Compile* from the menu to compile the job.

If your mapping is correct, the Compilation Status window displays the *Job successfully compiled with no errors* message. If your mapping is incorrect, the Compilation Status window displays an error message.

14. If your job successfully compiles, select Close.

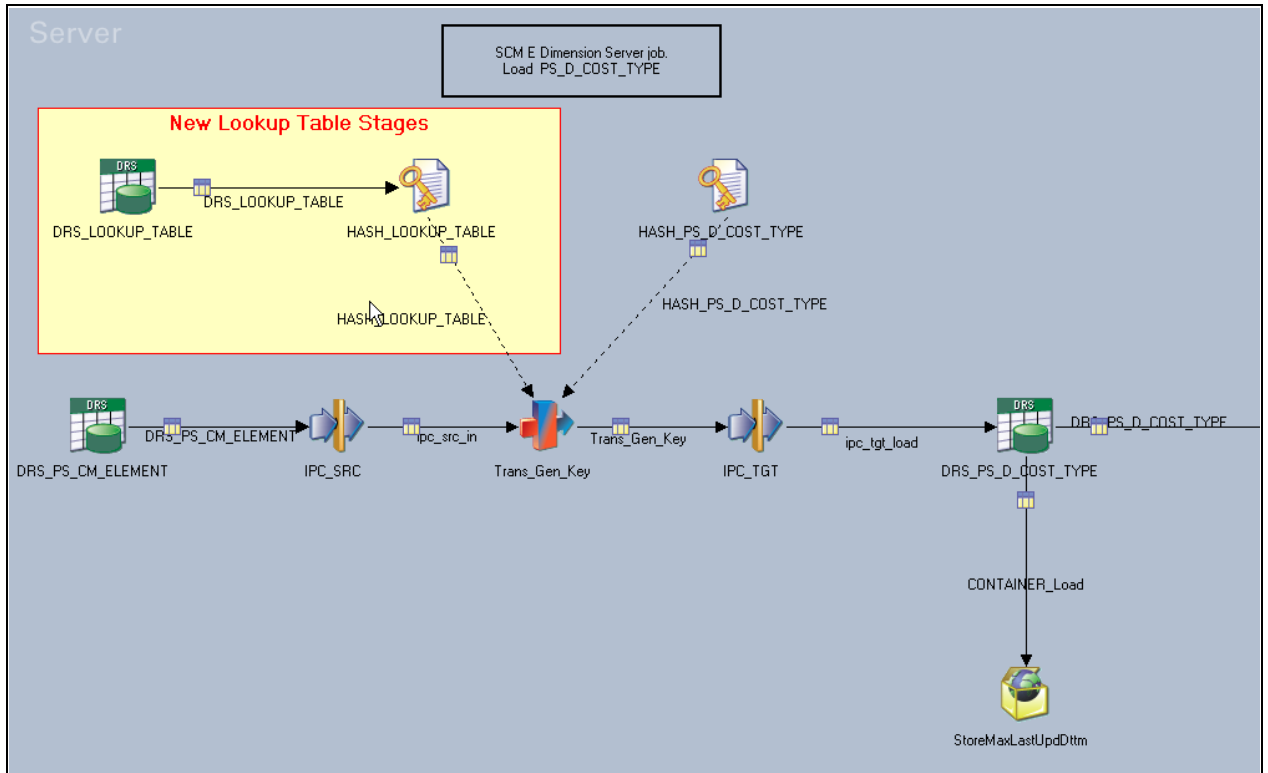
If your job does not compile successfully, you must return to the job and troubleshoot the errors.
15. You should perform technical unit testing and regression testing on the server job to ensure that the new attribute is populated properly.

Updating a Dimension Job with a New Attribute That Originates from a Lookup Table

Perform the following steps to update a dimension job with an attribute that originates from a lookup table:

1. In IBM WebSphere DataStage Designer, locate the appropriate dimension job and open it for editing.

2. Create a new DRS lookup stage and open it for editing.



New lookup table stages

3. Select the main Stage tab and input the appropriate values for database type, connection name, user ID and user password in the General sub-tab.

DRS_LOOKUP_TABLE - DRS stage

Stage **Output**

Stage name: DRS_LOOKUP_TABLE

General NLS

Database type:
#\$OWS_DBTYPE# DBMS Type ▾

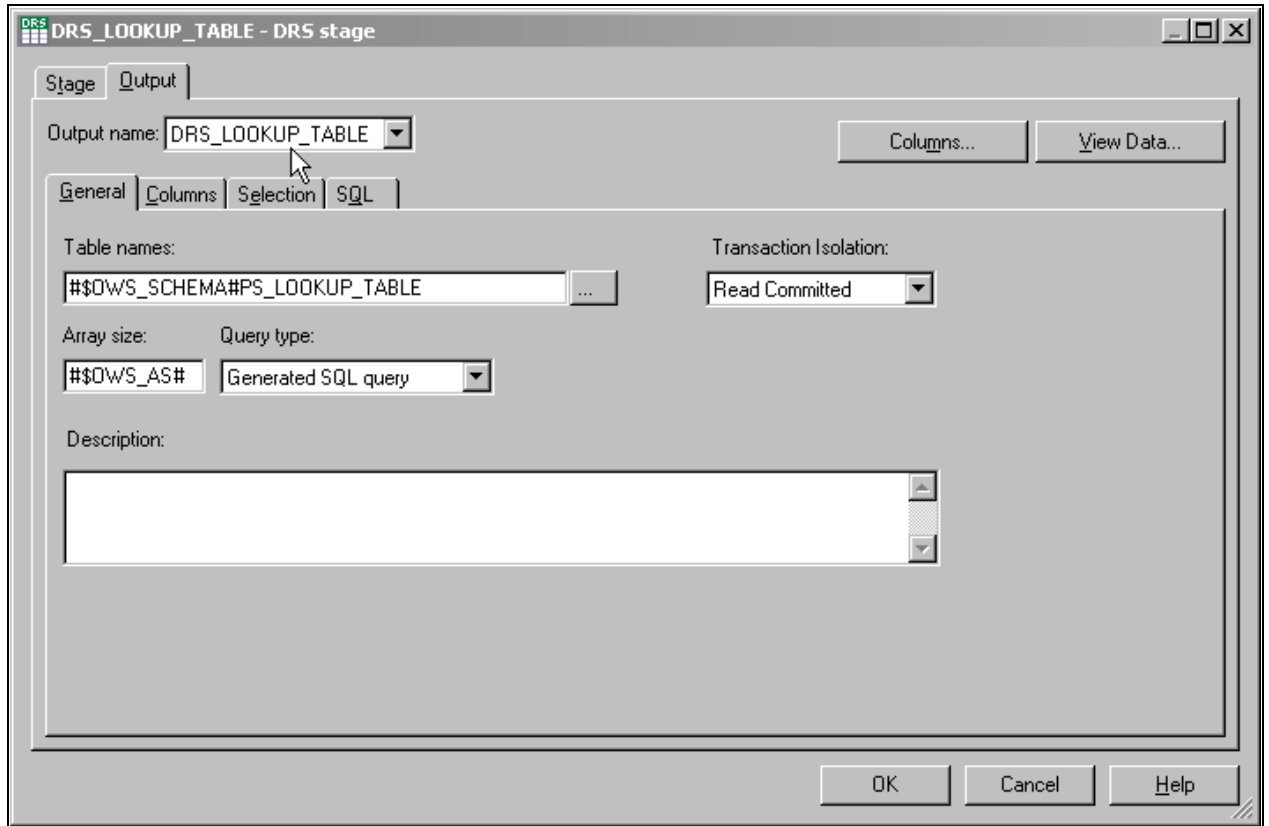
Connection name: User ID: Password:
#\$OWS_DBCONNECTION# \$OWS_USERNAME# *****

Description:
This stage is used to extract the incremental data from the source table PS_LOOKUP_TABLE

OK Cancel Help

DRS Lookup with Stage tab and General sub-tab selected

4. Select the Output tab and then the General sub-tab.



DRS Lookup with Output tab and General sub-tab selected

5. Input the appropriate values for the table names, transaction isolation, array size, and query type.
The query type can be user defined or generated by SQL.

6. Select the Columns sub-tab and add a new row for the new attribute.

Column name	Derivation	Key	SQL type	Length	Scale	Nullable	Display	Data
KEY1	LTrim(RTrim(KEY1))	Yes	Char	5		No	5	
KEY2	LTrim(RTrim(KEY2))	Yes	Char	4		No	4	
KEY3	LTrim(RTrim(KEY3))	Yes	Char	5		No	5	
NEW_ATTRIBUTE	LTrim(RTrim(NEW_ATTRIBUTE))	No	Char	30		No	30	
*								

Buttons: Save... Load...

DRS Lookup with Output tab and Columns sub-tab selected

7. Input the appropriate values for the derivation, key, SQL type, length, scale, and other applicable properties of the new attribute.

The key fields must be marked appropriately as they are used to extract the value for the new attribute.

8. Select the Selection sub-tab and input any selection criteria for the attribute.

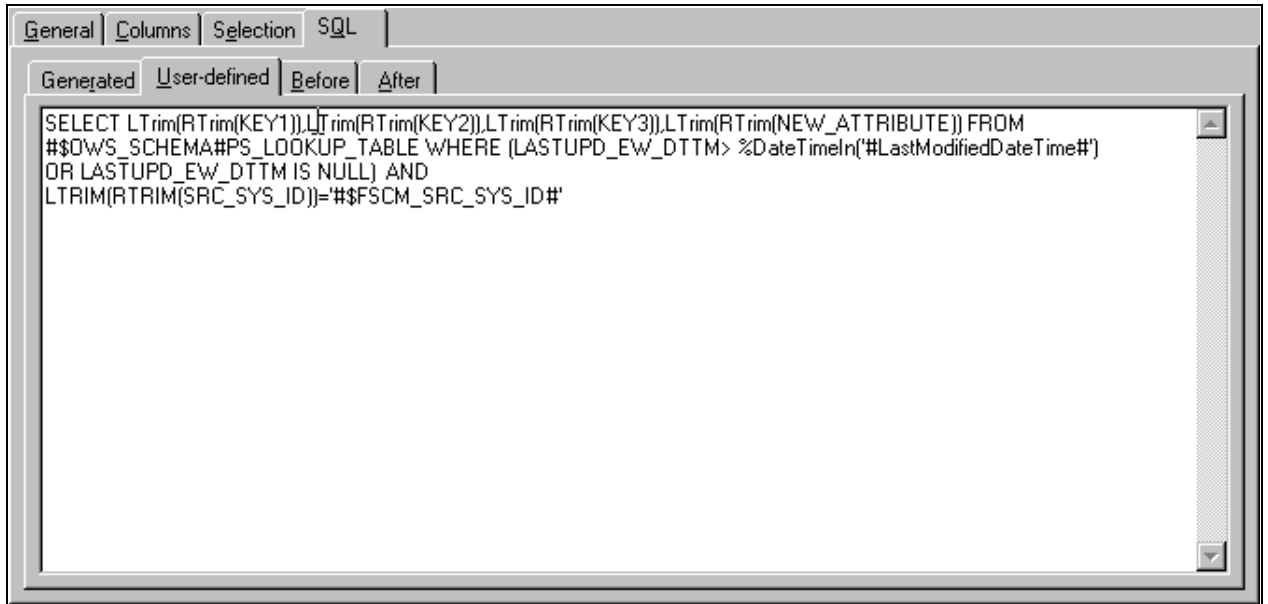
WHERE clause:

```
(LASTUPD_EW_DTTM > %DateTimeIn('#LastModifiedDate#')
OR LASTUPD_EW_DTTM IS NULL) AND
LTRIM(RTRIM(SRC_SYS_ID)) = '#FSCM_SRC_SYS_ID#'
```

Other clauses:

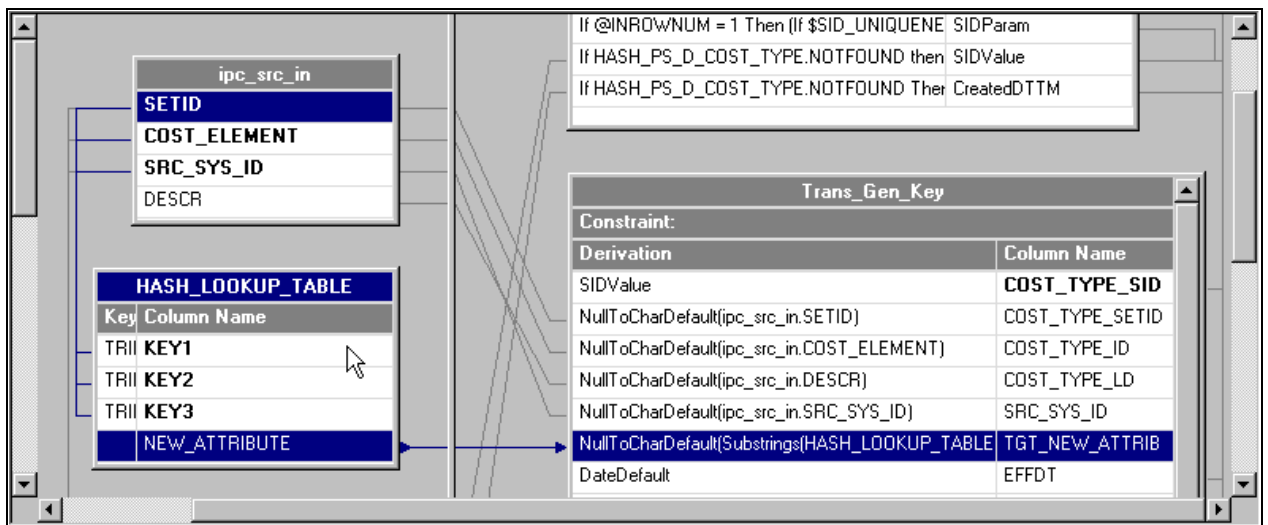
DRS Lookup with Output tab and Selection sub-tab selected

9. Select the SQL sub-tab and input any user-defined query for the attribute in the User-Defined tab.



DRS Lookup with user defined SQL

10. Connect an output link from the new DRS lookup stage to the applicable hash file stage.



Attribute to target mapping

11. Open the aforementioned hash file stage for editing and select the Inputs tab.
12. On the Inputs tab, input the appropriate file name and description, and select the options that are applicable to the attribute.
13. Select the Columns sub-tab and add a new row for the new attribute.

14. Input the appropriate values for the key, SQL type, length, scale, and other applicable properties of the new attribute.

Note. The column definitions must match those defined on the Output tab in the DRS lookup stage.

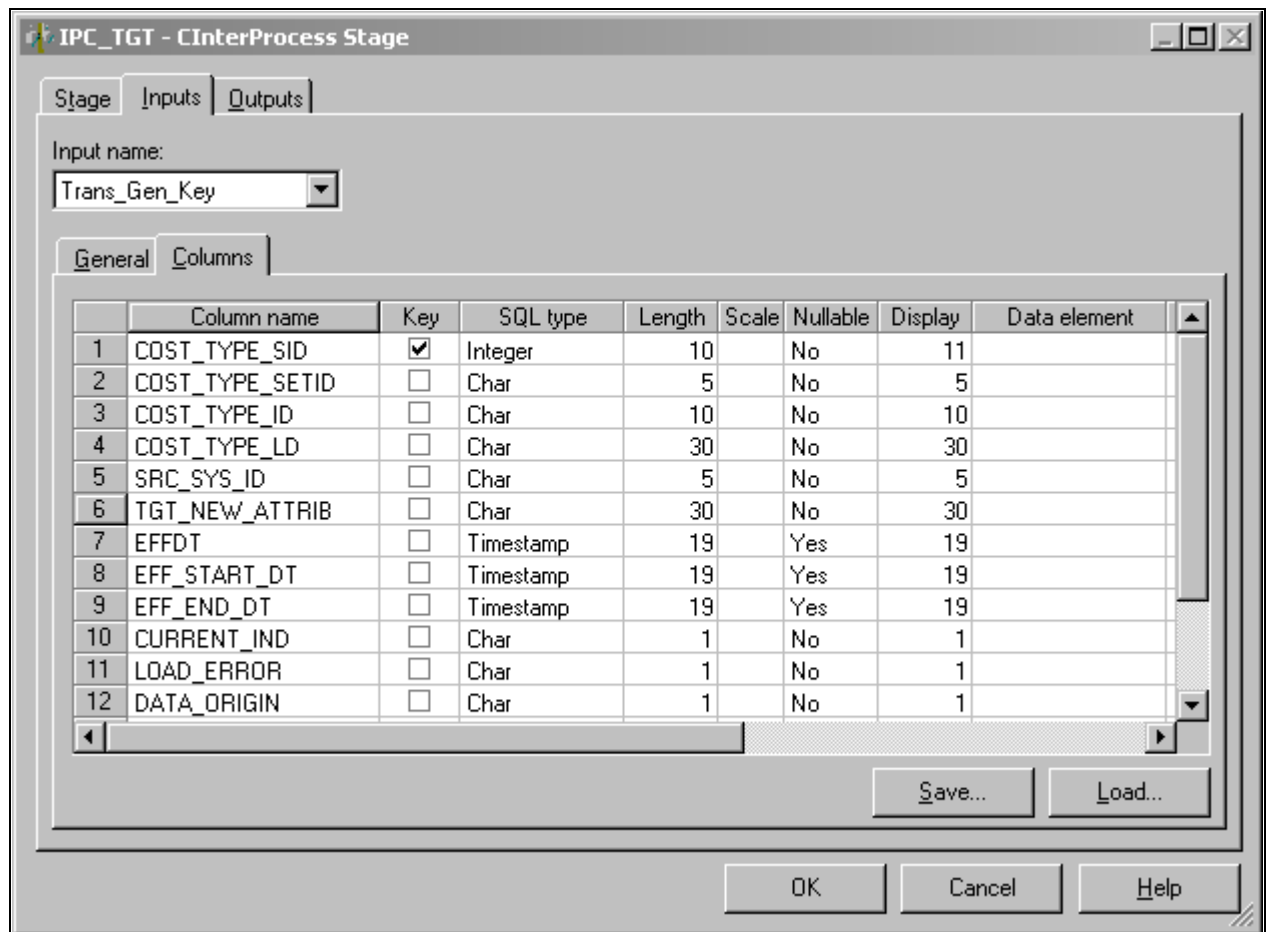
The hash file name must match the name specified in the Inputs and Outputs tabs. The hash file will provide erroneous values if the column definitions and hash file names are synchronized between the aforementioned tabs in the hash file stage.

15. Connect the output link of the hash file stage to the transformer stage (Trans_Gen_Key).

Once the link is connected to the transformer stage, the new lookup table becomes available in the inputs pane of the transformer stage.

16. In the inputs pane of the transformer stage, define the key expression for each key field.

The value of the key expressions is sourced from the main input link (ipc_src_in) of the transformer stage. Parameters and constant values can also be used to match with the key fields of the lookup table.

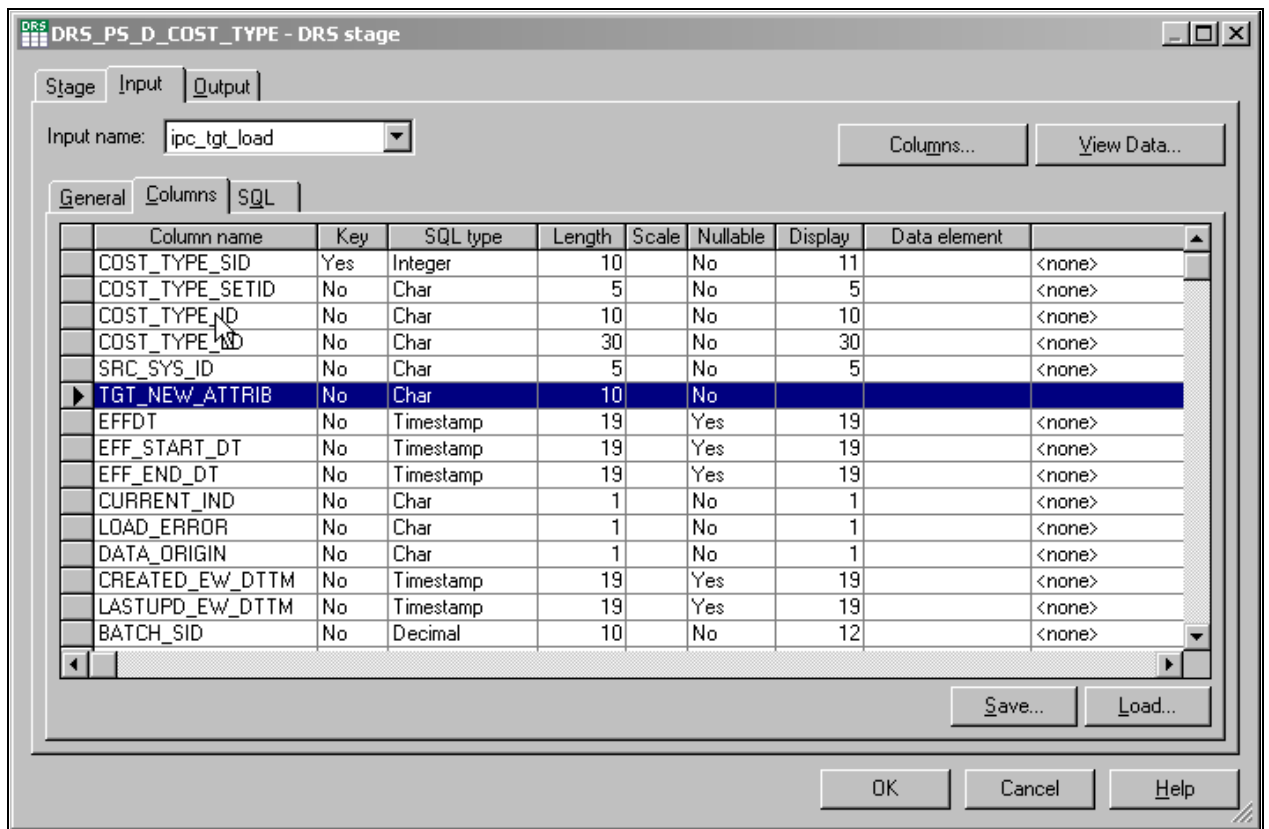


IPC Stage with column definitions

17. Apply transformation logic, such as any string or number functions to the new attribute from the lookup table, as necessary.

The logic is defined in the derivations field of the output link for the target table.

18. Connect the output link of the transformer stage to the target dimension table.
19. Open the IPC stage and select the Inputs tab.
20. In the Columns sub-tab ensure that the new attribute column is present and properly defined.



Target DRS stage with new attribute row

21. Open the target DRS stage and select the Input tab.
22. In the Columns sub-tab ensure that the new attribute column is present and properly defined.
23. Select *File, Save* from the menu to save the job.
24. Select *File, Compile* from the menu to compile the job.

If your mapping is correct, the Compilation Status window displays the *Job successfully compiled with no errors* message. If your mapping is incorrect, the Compilation Status window displays an error message.

25. If your job successfully compiles, select Close.

If your job does not compile successfully, you must return to the job and troubleshoot the errors.

26. You should perform technical unit testing and regression testing on the server job to ensure that the new attribute is populated properly.

Appendix A

Frequently Asked Questions for Campus Solutions Warehouse Implementation

This appendix provides answers to frequently asked implementation questions for the Campus Solutions Warehouse, and covers these topics:

- Implementation Scenarios
- Environmental Parameters Configuration
- ETL Loading Sequence

Implementation Scenarios

This table provides answers to questions about Campus Solutions Warehouse implementation scenarios.

Question	Answer
What source versions of Campus Solutions does EPM support?	EPM supports PeopleSoft Enterprise Campus Solutions 8.9 and 9.0.
What is the SRC_SYS_ID value that needs to be set for the Campus Solutions Warehouse?	<p><i>Scenario One:</i> Fresh installation of EPM 9.1 (Not an upgrade)</p> <p>HCM and Campus Solutions can share the same database instance or Campus Solutions can exist in a separate database instance. In both the cases, we recommend you provide a unique value for the SRC_SYS_ID, different from the SRC_SYS_ID value that is being used for other warehouses (when there are other warehouses being implemented). For example, the SRC_SYS_ID can be set to CS9 or CS.</p> <p><i>Scenario Two:</i> Upgrade to EPM 9.1</p> <p>Upgrading from EPM 9 to 9.1, the SRC_SYS_ID value that was used for Campus Solutions 9.0 should be carried over to EPM 9.1 as well.</p> <p>If upgrading from earlier EPM version, you can provide a unique value for the SRC_SYS_ID, different from the SRC_SYS_ID value that is being used for other warehouses (when there are other warehouses being implemented). For example, the SRC_SYS_ID can be set to CS9 or CS.</p>

Question	Answer
<p>What are the common jobs used by the Campus Solutions Warehouse?</p>	<ul style="list-style-type: none"> • PS_D_BUS_UNIT_COMP (Common Dimensions) • PS_D_BUS_UNIT_PF (Common Dimensions) • PS_D_LOCATION (Global Dimensions) • PS_D_PERSON (Global Dimensions) • PS_D_YEAR (Common Dimensions) • PS_D_DT_PATTERN (Common Dimensions)

ETL Load Sequence

This table provides answers to questions about ETL load sequence for the Campus Solutions Warehouse.

Question	Answer
<p>In what order must I run dimension and fact jobs to implement the different Campus Solution Warehouse data marts?</p>	<p>The appendix <i>Using the PeopleSoft EPM Lineage Spreadsheets</i> provides information about the ETL jobs that are delivered with the Campus Solutions Warehouse and links to the ETL lineage spreadsheet, which you can use like a reverse-engineering tool and identify the sequence of jobs to run for a specific data mart.</p> <p>See <i>PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook</i>, "Using the PeopleSoft EPM Lineage Spreadsheets."</p>
<p>What is the loading sequence for Campus Solutions Warehouse fact jobs?</p>	<p>There are some aggregate facts in the Campus Solutions Warehouse that have a dependency on the other facts as lookup or source tables. For example, in Student Records, the Academic Plan Summary Fact is the source table for the Institution Summary Fact.</p> <p>The mandatory loading sequence for each module is provided below.</p> <p><i>Admissions and Recruiting:</i></p> <ul style="list-style-type: none"> • SEQ_J_Fact_PS_F_ADM_APPL • SEQ_J_Fact_PS_F_ADM_APPL_STAT • SEQ_J_Fact_PS_F_EXT_ACAD_SUMM • SEQ_J_Fact_PS_F_EXT_TESTSCORE • SEQ_J_Fact_PS_F_STU_RECRT

Question	Answer
	<p><i>Student Financials - Financial Aid:</i></p> <ul style="list-style-type: none"> • SEQ_J_Fact_PS_F_AWD_DISB • SEQ_J_Fact_PS_F_AWD_SNPSHT <p><i>Student Financials - Student Financials:</i></p> <ul style="list-style-type: none"> • SEQ_J_Fact_PS_F_BIL_SNPSHT • SEQ_J_Fact_PS_F_PYMNT_PENDING • SEQ_J_Fact_PS_F_SF_TRAN <p><i>Student Records:</i></p> <ul style="list-style-type: none"> • SEQ_J_Fact_PS_F_ACAD_PROG_DTL • SEQ_J_Fact_PS_F_ACADPLAN_SUMM • SEQ_J_Fact_PS_F_CLASS_ENRLMT • SEQ_J_Fact_PS_F_CLASS_INSTRCT • SEQ_J_Fact_PS_F_TERM_ENRLMT • SEQ_J_Fact_PS_F_INST_SUMM
<p>What aggregate and/or snapshot fact tables are delivered for Campus Solutions Warehouse?</p>	<p>The Campus Solutions Warehouse provides these aggregate and snapshot tables:</p> <ul style="list-style-type: none"> • Award Summary (F_AWD_SNPSHT): Student Financials data mart • Bill Summary (F_BIL_SNPSHT): Student Financials data mart • Academic Plan Summary (F_ACADPLAN_SUMM): Student Records data mart • Institution Summary (F_INST_SUMM): Student Records data mart <p>For more information on these tables, please refer to the fact table documentation provided in the <i>Understanding the Campus Solutions Warehouse</i> chapter of this guide.</p>

Environmental Parameters Configuration

This table provides answers to questions about configuring environmental parameters.

Question	Answer
What are the list of environmental parameters that can be configured globally to tailor the ETL/Transformation logic for your institution without modifying the job?	See the table below for answer.

Parameter	Fact	Parameter Details	Sample Value
CS_FROM_DATE	Bill Summary	Specifies the start date for snapshot creation.	1985-01-01 00:00:00
CS_TO_DATE	Bill Summary	Specifies the end date for snapshot creation.	9999-12-31 00:00:00
CS_TRANZ_DT	Award Summary	Is used to. assign the date to your snapshot. This is converted to Date SID in the target fact.	1985-01-01 00:00:00
CS_ACAD_CAR_LST	Institution Summary	Lists the Academic Careers for which the Institution Summary Fact will hold information.	'GRAD','PGRD','RSCH','UGRD'
CS_ACN_RSN_CODES	Institution Summary	Lists the Action Code and Action Reason Code concatenated as a single word. For example, Action Code is 'DISC' and the Action reason Code is 'PDIS'. The value for CS_ACN_RSN_CODES will be 'DISCPDIS'.	'DATAPDIS','DISCARMF','LEAVOFCM','DISCDEAT'
CS_FT_IND	Institution Summary	Lists the different values which consider a particular Academic Load as a Full time Academic load.	'F'
CS_PT_IND	Institution Summary	Lists the different values which indicate if the Student has opted for a Part time Academic load.	'P','H','L'

Appendix B

Using the PeopleSoft EPM Lineage Spreadsheets

This document provides an overview of the EPM lineage spreadsheets and discusses how to use the spreadsheets to:

- View lineage information.
- Generate lineage information for a job.

Understanding the EPM Lineage Spreadsheets

The EPM lineage spreadsheets provide information about the ETL jobs that are delivered with the EPM warehouses. The spreadsheets act like a reverse-engineering tool or family tree; they enable you to view the ancestry of source, target, and lookup tables and their relevant ETL jobs. Each spreadsheet provides lineage information for a single warehouse. The following table lists the lineage spreadsheets that are currently available:

<i>Lineage Spreadsheet Filename</i>	<i>Warehouse</i>
ETL_CS_Lineage_Spreadsheet.xls	Campus Solutions Warehouse
ETL_CRM_Lineage_Spreadsheet.xls	CRM Warehouse
ETL_FMS_Lineage_Spreadsheet.xls	FMS Warehouse
ETL_HCM_Lineage_Spreadsheet.xls	HCM Warehouse
ETL_SCM_Lineage_Spreadsheet.xls	SCM Warehouse

By using the spreadsheets, you can:

- View lineage information for staging, dimension, and fact ETL jobs, or source, target, and lookup tables.
- Identify the sequence of jobs to run for a specific data mart.
- Identify inter-mart and cross-warehouse dependencies.
- Generate lineage information for a specific ETL job.

Spreadsheet Structure

Each EPM lineage spreadsheet includes several worksheets. The following table provides a description of each worksheet, by name, listed in the order in which it appears:

Worksheet	Description
Template	This worksheet contains overview information, a legend, and a definition of the columns used in the worksheets.
Setup	This worksheet contains ETL lineage information for all of the setup and staging jobs required for the warehouse.
Com Dims	This worksheet contains ETL lineage information for the common dimension jobs required for the warehouse.
Utils	This worksheet contains ETL lineage information for the currency conversion jobs required for the warehouse.
Global Dims	This worksheet contains ETL lineage information for the global dimension jobs required for the warehouse.
Local Dims	This worksheet contains ETL lineage information for the local dimension jobs required for the warehouse.
<SKU/Data Mart Name> For example: GL & Profitability, Campus Community, and so on.	This worksheet contains ETL lineage information for the jobs required for a specific data mart. Note. Each spreadsheet includes several data mart worksheets.
Dynamic_Lineage_Generator	This worksheet provides a macro that enables you to enter the name of an ETL job and automatically generate a list of the complete lineage for that job.
JobOrder	This worksheet is an extension of the Dynamic_Lineage_Generator worksheet. It displays the order in which jobs need to be run.

Column Descriptions

The following table provides descriptions of the columns in the worksheets.

Column	Description
Sequencer Job	The name of the job sequencer, which is responsible for invoking and running other ETL server jobs.
Server Job	The name of the server job that is called by the job sequencer.

Column	Description
Server Job Category	The location of the server job in the IBM WebSphere DataStage project.
Target Table	The name of the target table used in the server job.
Target Update Action	The target load strategy for the server job.
Source Table	The name of the source table used in the server job.
Source Extraction Type	The type of extraction from the source table in the server job (for example, incremental date time or cyclical redundancy check).
Lookup Tables	The name of the lookup tables that are used in the server job. Lookups can be hashed files or direct DRS lookups. The lineage information captures the table names from which the hash files are populated and the table names for the direct DRS lookup.
Setup Jobs	The name of the setup job that populates the source and/or the lookup table.
Setup Sequencer Job	The name of the job sequencer that calls the setup server job.
MDW	The name of the MDW server job. This column has an entry if the source table or lookup table is populated from an MDW server job.
MDW Sequencer	The name of the MDW sequence job.
OWS	The name of the OWS server job. This column has an entry if the source table or lookup tables are populated from an OWS server job.
OWS Sequencer	The name of the OWS sequence job.
OWE	The name of the OWE server job. This column has an entry if the source table or lookup tables are populated from an OWE server job.
OWE Sequencer	The name of the OWE sequence job.
EPM Foundation	The application or EPM foundation setup page that populates the source table or the lookup table, such as Global Consolidations, Dimension Mapper, or setup PIA pages.
Category	The categories in which the setup jobs, MDW jobs, OWS jobs or OWE jobs are placed.

Column	Description
Comments	Any additional comments, if applicable.

Note. The spreadsheet does not contain lineage details for OWE jobs and Tree jobs, except for the GL&Profitability Mart of the FMS warehouse, which does include lineage information for OWE jobs.

Viewing Lineage Information

This section discusses how to use the spreadsheet to:

- Find lineage information for a server job.
- Identify the list of Jobs to be run for a data mart.

Finding Lineage Information for a Server Job

To find lineage information for a server job:

1. Access the worksheet in which the job is categorized.
2. Use Excel's Find feature to find the server job name in column B.
 - a. Type Ctrl-F to access the Find and Replace Dialog box.
 - b. Enter the name of the server job in the Find what edit box.
 - c. Click Find Next until the job name is found in the Server Job column (column B).
 - d. Close the Find dialog box.
3. Review the lineage information in the adjacent columns.

The Sequencer Job column (column A) lists the sequencer which calls this job. The Server Job Category column (column C) lists the category this job is associated with. The Target Table, Target Update Action, Source Table, and Source Extraction Type for this server job are listed in columns D, E, F, and G respectively. The Lookup Tables column (Column H) lists all the lookups used by this job.

The source tables and the lookup tables are placed in separate rows. This enables you to find the lineage information for each of these tables by navigating through the other subsequent columns in the same row. Columns I through R list the dependent jobs that are required to populate the source and lookup tables, and entries in these columns indicate whether the table is populated by Setup jobs, (column I), MDW jobs (column K), OWS jobs (column M), OWE jobs (column O), or Foundation setup / Apps (column Q). The Category column (column R) lists the category that the dependent job is associated with.

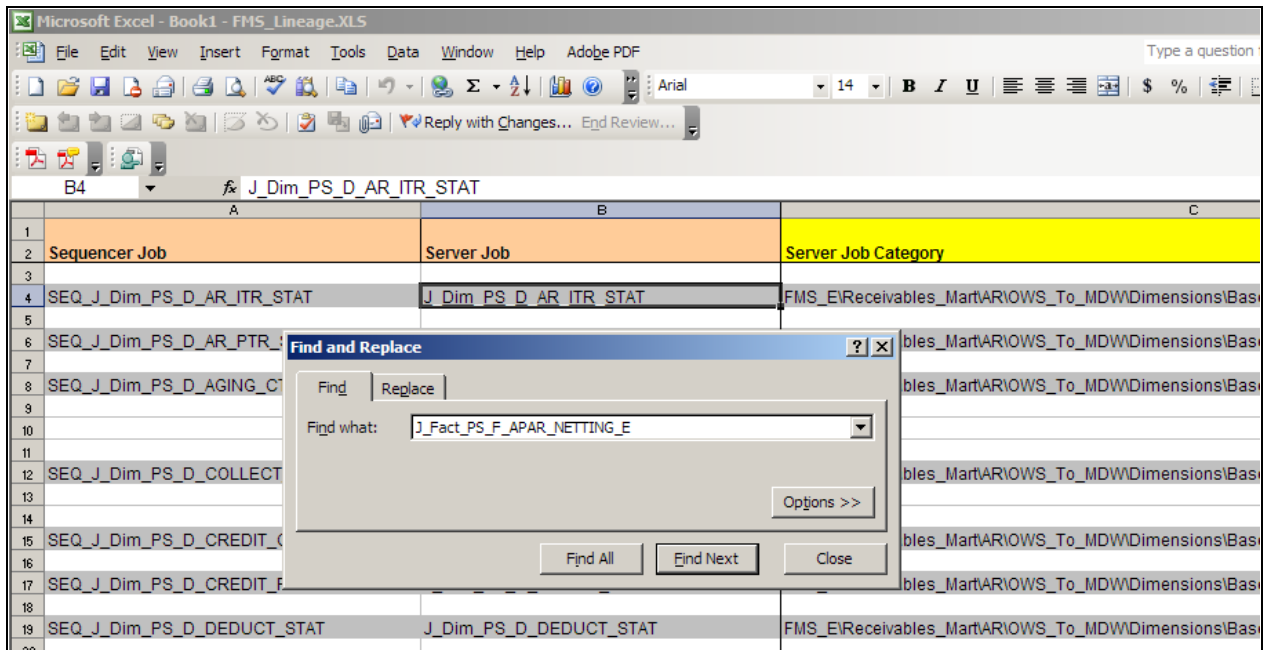
Source tables that are from a different data mart (inter-mart) or different warehouse (cross-warehouse) are indicated by the colors specified in the legend on the Template worksheet page.

The spreadsheet lists the lineage of a source or lookup table to the level of the job that directly populates it. The lineage information does not extend to level of the last staging job. To get the complete lineage for a fact or dimension job fully extended through the lowest staging level, you can use the dynamic lineage generator tool, which generates a list of all the required dependent jobs that need to be run in order to load a particular fact or dimension.

Example

This example, from the ETL FMS Lineage spreadsheet, takes you through the tasks you would complete to review the information for the fact job J_Fact_PS_F_APAR_NETTING_E, which is used for the AR Data Mart.

1. Navigate to the AR worksheet page.
2. Type Ctrl-F and type J_Fact_PS_F_APAR_NETTING_E into the Find and Replace dialog box.



Find and Replace Dialog Box

3. Type Ctrl-F and type J_Fact_PS_F_APAR_NETTING_E into the Find and Replace dialog box.
4. Click Find Next until you access the cell in the Server Job column that contains the J_Fact_PS_F_APAR_NETTING_E job.

5. Close the Find and Replace dialog box. You should see the following information:

	A	B	
1			
2	Sequencer Job	Server Job	Server Job Category
90			
91	SEQ_Dims_L_O_CREDIT_CLASS	J_Dim_PS_O_CREDIT_CLASS	FMS_EIReceivables_MartAR\OWS_T
92			
93			
94	SEQ_Dims_L_O_CREDIT_RISK	J_Dim_PS_O_CREDIT_RISK	FMS_EIReceivables_MartAR\OWS_T
95			
96			
97	SEQ_Dims_L_O_DEDUCT_STAT	J_Dim_PS_O_DEDUCT_STAT	FMS_EIReceivables_MartAR\OWS_T
98			
99			
100	SEQ_Dims_L_O_DISPUTE_STAT	J_Dim_PS_O_DISPUTE_STAT	FMS_EIReceivables_MartAR\OWS_T
101			
102			
103	SEQ_Dims_L_O_ENTRY_RSTYP	J_Dim_PS_O_ENTRY_RSTYP	FMS_EIReceivables_MartAR\OWS_T
104			
105			
106	SEQ_J_Fact_PS_F_APAR_NETTING_E	J_Fact_PS_F_APAR_NETTING_E	FMS_EIReceivables_MartAR\OWS_T
107			
108			
109			
110			
111	SEQ_J_Fact_PS_F_AR_ACCOUNT_LN_E	J_Fact_PS_F_AR_ACCOUNT_LN_E_ITEM	FMS_EIReceivables_MartAR\OWS_T

J_Fact_PS_F_APAR_NETTING_E job displayed in spreadsheet

6. Scroll to the right to review the columns shown here:

D	E	F	G
Target Table	Target Update Action	Source Table	Source Extraction Type
		PS_D_ENTRY_RSTYP	
PS_F_APAR_NETTING	Truncate table then insert rows	PS_D_CUST_ORG	
		PS_D_SUPPLIER	
		PS_F_AR_AGING	
PS_F_AR_ACCOUNT_LN	Insert new rows or update existing ones	PS_ITEM_DST	DateTime Incremental
		PS_ITEM	

Reviewing data associated with the J_Fact_PS_F_APAR_NETTING_E job

The Target Table, Target Update Action, Source Table, and Source Extraction Type for the J_Fact_PS_F_APAR_NETTING_E server job are listed in columns D, E, F, and G, respectively.

- Continue to scroll to the right to view the remaining columns.

The Lookup Tables column (Column H) lists all the lookups used in J_Fact_PS_F_APAR_NETTING_E.

F	G	H
Source Table	Source Extraction Type	Lookup Tables
PS_D_CUST_ORG		
PS_D_SUPPLIER		
PS_F_AR_AGING		PS_F_AP_AGING
PS_ITEM_DST	DateTime Incremental	
PS_ITEM		

Lookup Tables Column

In this example there are three source tables: PS_D_CUST_ORG, PS_D_SUPPLIER, PS_F_AR_AGING. The lookup table is PS_F_AP_AGING. The source tables and the lookup tables are each placed in a unique row one after the other. This enables you to view the lineage information for each of these tables by navigating through the succeeding columns within the same row.

Columns I through R list out the dependent jobs required to populate these source and lookup tables. In this example, the source table PS_D_CUST_ORG has an entry in the MDW column, which means that it is populated from the MDW dimension J_Dim_PS_D_CUST_ORG_SCM, which is placed in the category Global_Dimensions_E\OWS_To_MDW\Base\Load_Tables\Server.

As shown in the following screenshot, the source table PS_D_SUPPLIER is an SCM warehouse dimension. The cross-warehouse dependency is identified by the different color (the color legend is located on the first worksheet page).

F	L	R
Source Table	MDW Sequencer	Category
PS_D_CUST_ORG	SEQ_J_Dim_PS_D_CUST_ORG_SCM	Global_Dimensions_E\OWS_To_MDW\Base\Load_Tables\Server
PS_D_SUPPLIER	SEQ_J_Dim_PS_D_SUPPLIER	Global_Dimensions_E\OWS_To_MDW\Base\Load_Tables\Server
PS_F_AR_AGING	SEQ_J_Fact_PS_F_AR_AGING_E	FMS_E\Receivables_Mart\A\OWS_To_MDW\Facts\Base\Load_Tables\Server
	SEQ_J_Fact_PS_F_AP_AGING_E	FMS_E\Payables_Mart\A\OWS_To_MDW\Facts\Base\Load_Tables\Server

Cross-warehouse dependencies for PS_D_SUPPLIER

Similarly, the lookup table PS_F_AP_AGING is populated from the fact job J_Fact_PS_F_AP_AGING placed in the category FMS_E\Payables_Mart\A\OWS_To_MDW\Facts\Base\Load_Tables\Server. This fact job belongs to a different mart as indicated by the different color.

H	K	L	R
Lookup Tables	MDW	MDW Sequencer	Category
	J_Dim_PS_D_CUST_ORG_SCM	SEQ_J_Dim_PS_D_CUST_ORG_SCM	Global_Dimensions_E\OWS_To_MDW\Base\Load_Tables\Server
	J_Dim_PS_D_SUPPLIER	SEQ_J_Dim_PS_D_SUPPLIER	Global_Dimensions_E\OWS_To_MDW\Base\Load_Tables\Server
	J_Fact_PS_F_AR_AGING_E	SEQ_J_Fact_PS_F_AR_AGING_E	FMS_E\Receivables_Mart\A\OWS_To_MDW\Facts\Base\Load_Tables\Server
PS_F_AP_AGING	J_Fact_PS_F_AP_AGING	SEQ_J_Fact_PS_F_AP_AGING_E	FMS_E\Payables_Mart\A\OWS_To_MDW\Facts\Base\Load_Tables\Server

Cross-warehouse dependencies for PS_F_AP_AGING

Identifying the List of Jobs to be Run for a Data Mart

You can use the information in the spreadsheet to identify the list of jobs that need to be run for a specific data mart. These include common jobs that are required for every data mart, which we refer to as prerequisite jobs, as well as jobs specific to the particular data mart.

If you prefer, you can create your own master sequencers based on the information provided in this section.

Alternatively, you can generate the list of jobs by using the Dynamic Lineage Generator tool. For more information, see "Generating Lineage Information for a Job".

Note. All the server jobs relating to Hash files that are present within the Load_Hash_Files category need to be run first before running other Sequence jobs within the Load_Tables category since these hash files are being used in other server jobs.

Prerequisite Jobs

The prerequisite jobs include setup jobs, staging jobs, and dimension jobs.

The following sets of jobs need to be run for *every* mart, in the order that they are listed in the worksheets:

1. Run these setup jobs in the Setup worksheet:

- a. All jobs within the Setup_E\OWS*<Warehouse>* category.

(For example all jobs within the Setup_E\OWS\FSCM category for the FMS warehouse and all jobs within the Setup_E\OWS\CS category for the Campus Solutions warehouse).

- b. All jobs within the Setup_E\Dimension mapper category.

Note. Please ensure that you run the Business Unit Wizard before proceeding with the following steps.

See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Importing Source Business Units into EPM to Create Warehouse Business Units."

- c. All jobs within the Shared_Lookups\DimensionMapper_Lookups category.
- d. All jobs within the Shared_Lookups\Control_Tables category.
- e. All jobs within the Shared_Lookups\System_Lookups category.
- f. All jobs within the Shared_Lookups\Language_Lookups category.
- g. All jobs within the Setup_E\OWE category (this step does not apply to the Campus Solutions warehouse).
- h. If you are implementing currency conversion, then run the jobs listed in the Utils worksheet.

2. Run the staging jobs listed in the OWS Sequencer column (column N) in the following worksheets:
 - a. Com Dims.
 - b. Global Dims.
 - c. Local Dims.
 - d. *<Data Mart>*, where *<Data Mart>* is the name of the data mart, for example AP, AR, Campus Community, Student Financials .
3. Run the Common Dimension Jobs listed in the Com Dims worksheet.
4. Run the Global Dimensions jobs listed in the Global Dims worksheet. (These jobs are required for running the FMS warehouse jobs.)
5. Run the Local Dimension Jobs placed in the Local Dims worksheet.

Data Mart Specific Jobs

Run all the Server jobs listed in column B of the worksheet for the specific data mart, to populate the corresponding Dimension and Fact tables for that mart.

Note. Do not run the jobs that are listed within the Reusable Jobs category. These jobs are not used to load target tables. They are automatically triggered by various Sequence jobs.

Generating Lineage Information for a Job

The Dynamic_Lineage_Generator worksheet contains a macro that generates a list of all the dependent jobs that are required for any ETL job. This will easily help you identify all the list of jobs to be run for a specific fact or dimension job.

To use the Dynamic Lineage Generator:

1. Access the Dynamic_Lineage_Generator worksheet.
2. Enter the job name in cell B1.
3. Click the Get Job Lineage button.

The macro retrieves the lineage required for running this fact job from the setup, staging, and the dimension jobs and displays it in the cells below. The macro also copies the entire list of dependent jobs to the JobOrder worksheet, so you can identify the complete list to be run in sequence.

You must run the following prerequisite setup jobs before you run the jobs listed in the JobOrder worksheet:

- Setup_E\OWS*<Warehouse Name>* Job Sequencer.

For example Setup_E\OWS\FSCM Job Sequencer or Setup_E\OWS\CS Job Sequencer.

- Setup_E\Dimension mapper Job Sequencer.

- Run the Business Unit Wizard to populate the Dimension mapper tables.

See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Importing Source Business Units into EPM to Create Warehouse Business Units."

- Shared_Lookups\DimensionMapper_Lookups
- Shared_Lookups\Control_Tables
- Shared_Lookups\System_Lookups
- Shared_Lookups\Language_Lookups
- Setup_E\OWE Job Sequencer (this step does not apply to the Campus Solutions warehouse).

After you run the prerequisite setup jobs, then run the jobs listed in the JobOrder worksheet.

Index

A

- Academic Program Activation 34
- Academic Program Activation and Management
 - Business Process 34
- Academic Program Management 34
- Admissions and Recruiting Data Mart
 - delivered fact and dimension tables 19
 - overview 19
- Audit Jobs
 - creating audit triggers 94
 - implementation 92
 - overview 91
- Audit Triggers page 94
- Award Business Process 44

B

- balancing
 - denormalized tree 143
 - notation 143
 - parameters required 143
 - up-balancing, down-balancing 143

C

- Campus Community Data Mart
 - delivered fact and dimension tables 28
 - overview 28
- Campus Solutions Warehouse
 - Admissions and Recruiting Data Mart 19
 - business processes 4
 - Campus Community data mart 28
 - common dimensions 53
 - data marts 4
 - getting started 1
 - implementation frequently asked questions 197
 - overview 17
 - preface ix
 - running implementation jobs for 62
 - Structure 18
 - Student Financial Services data mart 43
 - Student Records data mart 34
- Campus Solutions Warehouse Frequently Asked Implementation Questions
 - environmental parameters configuration 199
 - ETL load sequence 198
 - implementation scenarios 197
- CCU_CONV_DEFN 111
- CCU_CONV_DEFN component 125
- CCU_SCHEMA_DEFN 111, 125
- chunking rules
 - for currency conversion 124
- Common Dimensions 53
- conversion rules
 - for currency conversion 122

- CRM Warehouse
 - business processes 3
 - data marts 3
- currency conversion 111
 - chunking rules 124
 - conversion rules 122
 - methodology 114
 - overview 111
 - running 129
 - schema rules 120
 - setting up 125
- Currency Conversion component (CCU_CONV_DEFN) 125
- Currency Conversion Definition (CCU_CONV_DEFN) 111
- currency conversion rules 119
 - setting up 127
- Currency Conversion Schema Definition (CCU_SCHEMA_DEFN) 111

D

- Data Marts
 - Campus Solutions Warehouse 4
 - CRM Warehouse 3
 - Financials Warehouse for Public Sector and Higher Education 5
 - FMS Warehouse 5
 - HCM Warehouse 6
 - SCM Warehouse 6
- Data Models 12
- Denormalization 139
- denormalized
 - tree result balancing 143
- dimension, evaluating requirements 173

E

- EPM Architecture
 - overview 13
- EPM Warehouse
 - components 7
 - reporting 15
 - understanding 3
- EPM Warehouse Business Process
 - Campus Solutions Warehouse 4
 - CRM Warehouse 3
 - Financials Warehouse for Public Sector and Higher Education 5
 - FMS Warehouse 4
 - HCM Warehouse 6
 - SCM Warehouse 6
- EPM Warehouse Component
 - data models 12
 - ETL 7
 - infrastructure tables and tools 7
 - MDW dimension tables 9
 - MDW fact tables 8

- measures 12
- security tables 8
- staging tables 8
- ETL**
 - audit jobs 91
 - managing source system deletes and archiving 97
 - multilanguage processing 131
 - running Campus Solutions Warehouse - OWS jobs 62
 - running Campus Solutions Warehouse SKU jobs 65
 - running global dimension jobs for Campus Solutions Warehouse 63
 - running implementation jobs 62
 - running language swap jobs 137
 - running local dimension jobs for Campus Solutions Warehouse 64
 - running the currency conversion process 129
 - running the language swap utility 131
 - running the tree and recursive hierarchy process 169
 - slowly changing dimensions 69
 - slowly changing dimensions in EPM 72
 - Source-Archiving Diagnostic feature 108
 - Source-Delete Diagnostic feature 103
 - source system archiving 101
 - source system deletes 97
 - Type 1 Slowly Changing Dimension 69
 - Type 2 Slowly Changing Dimension 70
 - Type 3 Slowly Changing Dimension 71
 - using EPM lineage spreadsheets 201
- ETL Lineage Spreadsheets 201**
 - generating lineage information for jobs 209
 - understanding 201
 - viewing lineage information 204

F

- Financials Warehouse for Public Sector and Higher Education**
 - business processes 5
 - data marts 5
- FMS Warehouse**
 - business processes 4
 - data marts 5

H

- H_HIERTBL_DEFN component 163**
- HCM Warehouse**
 - business processes 6
 - data marts 6
- Hierarchy Group Definition page 165, 166**
- Hierarchy Record Definition page 165**

L

- Language Swap Utility 131**
 - outrigger tables 134
 - overview 133
- language tables**

- related, in tree and recursive hierarchy processing 157**

M

- MDW *See* multidimensional warehouse**
- MDW Conversion Schema Rule page 126, 128**
- MDW Currency Conversion Rule page 127**
- MDW Currency Conversion Rules page 126**
- MDW Dimension Table 9**
 - commonly used 9
 - shared and global 9
- MDW Fact Table 8**
- measure, evaluating requirements 174**
- Measures 12**
- Multidimensional Warehouse (MDW)**
 - overview 15
- Multidimensional Warehouse Data Model**
 - adding fact and dimension tables to 174
 - extending 173
 - extending dimensions 186
 - extending fact tables 179
- Multilanguage Processing 131**
 - overview 131
 - setup overview 132

O

- Operational Warehouse - Enriched (OWE)**
 - overview 14
- Operational Warehouse - Staging (OWS)**
 - overview 14
- output tables**
 - tree denormalizer and tree flattener 147
- OWE *See* operational warehouse - enriched**
- OWS *See* operational warehouse - staging**

R

- recursive hierarchy processing 139**
 - OWE process vs. MDW process 140
 - supported hierarchies 141
- related language tables**
 - in tree and recursive hierarchy processing 157
- Relationship Record Definition page 164, 165**
- relationship tables 147**
- ROLAP**
 - closing skip levels 144
 - denormalized dimension balancing 143
- Run Audtrgs page 96**

S

- Schema Definition component (CCU_SCHEMA_DEFN) 125**
- Schema Definition page 125, 126**
- schema rules**
 - for currency conversion 120
- Schema Source Columns page 125**

- Schema Target Columns page 125
- SCM Warehouse
 - business processes 6
 - data marts 6
- Shared dimensions 55
- skip levels
 - in trees 144
 - notation 144
- Slowly Changing Dimensions 69
 - converting type 1 to type 2 76
 - converting type 1 to type 2 using the EFFDT and EFFSEQ fields 76
 - converting type 1 to type 2 without the EFFDT and EFFSEQ fields 88
 - EPM fact table support 75
 - in EPM 72
 - overview 69
 - Type 1 69
 - type 1 vs. type 2 in EPM 73
 - Type 2 70
 - Type 3 71
 - valid date range subrecords in EPM 72
- source tables
 - Enterprise tree and recursive hierarchy 146
 - tree and recursive hierarchy 146
 - tree and recursive hierarchy Enterprise 146
- Student Financial Services Data Mart
 - delivered fact and dimension tables 44
 - overview 43
- Student Records Data Mart
 - delivered fact and dimension tables 34
 - overview 34

T

- TH_HIERGRP_DEFN 163
- TH_RELTBL_DEFN component 163
- tree
 - balancing 143
 - denormalized balancing 143
- Tree and Recursive Hierarchy Processing
 - ETL process 169
 - output 153
 - overview 139
 - parameters, defining 163
 - process results 147
- tree denormalizer 140
 - output tables 147
 - results 157
- tree flattener 140
 - output tables 147
 - OWE process vs. MDW process 140
 - results 157
- Tree Hierarchy-Hierarchy Group Definition
 - component (TH_HIERGRP_DEFN) 163
- Tree Hierarchy-Hierarchy Table component
 - (TH_HIERTBL_DEFN) 163
- Tree Hierarchy Relational Table component
 - (TH_RELTBL_DEFN) 163
- tree processing 139
- trees
 - skip levels 144

