

**ORACLE®**

PEOPLESOFT

---

# PeopleSoft EPM 9.1: Financial Management Solutions Warehouse

---

December 2013

**ORACLE®**

PeopleSoft EPM 9.1: Financial Management Solutions Warehouse  
CDSKU epm91pbr4  
Copyright © 1999, 2013, Oracle and/or its affiliates. All rights reserved.

## **Trademark Notice**

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

## **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 is software or related documentation that 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 END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

## **Hazardous Applications Notice**

This software or hardware 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 that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

## **Third Party Content, Products, and Services Disclaimer**

This software or hardware 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.

## **Alpha and Beta Draft Documentation Notice**

If this document is in preproduction status:

This documentation is in preproduction status and is intended for demonstration and preliminary use only. It may not be specific to the hardware on which you are using the software. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to this documentation and will not be responsible for any loss, costs, or damages incurred due to the use of this documentation.



# Contents

- Preface.....ix**
  - Understanding the PeopleSoft Online Help and PeopleBooks..... ix
  - PeopleSoft Hosted Documentation..... ix
  - Locally Installed Help..... ix
  - Downloadable PeopleBook PDF Files.....ix
  - Common Help Documentation.....ix
  - Field and Control Definitions..... x
  - Typographical Conventions..... x
  - ISO Country and Currency Codes..... xi
  - Region and Industry Identifiers..... xi
  - Access to Oracle Support.....xi
  - Documentation Accessibility.....xii
  - Using and Managing the PeopleSoft Online Help..... xii
  - PeopleSoft EPM Related Links..... xii
  - Contact Us.....xii
  - Follow Us.....xii
- Chapter 1: Getting Started with PeopleSoft Financial Management Solutions Warehouse..... 13**
  - PeopleSoft Financial Management Solutions Warehouse Overview..... 13
  - Oracle's PeopleSoft Products..... 13
  - Other Sources of Information..... 13
- Chapter 2: Understanding the EPM Warehouses..... 15**
  - Overview of PeopleSoft EPM Warehouses..... 15
    - Campus Solutions Warehouse..... 15
    - CRM Warehouse..... 16
    - FMS Warehouse..... 16
    - Financials Warehouse for Public Sector and Higher Education..... 17
    - HCM Warehouse..... 17
    - SCM Warehouse..... 18
  - Components of PeopleSoft EPM Warehouses.....18
    - Extract Transform and Load (ETL) Component..... 19
    - Infrastructure Tables and Tools..... 19
    - Security Tables..... 19
    - Staging Tables..... 19
    - Multidimensional Warehouse Fact Tables..... 20
    - Multidimensional Warehouse Dimension Tables..... 20
    - Data Models..... 24
    - Measures..... 24
  - EPM Architecture and Data Flow..... 25
    - Operational Warehouse - Staging (OWS)..... 26
    - Operational Warehouse - Enriched (OWE)..... 27
    - Multidimensional Warehouse (MDW)..... 27
  - Reporting on the EPM Warehouses..... 27
- Chapter 3: Understanding the FMS Warehouse..... 29**
  - FMS Warehouse Overview..... 29
  - Understanding the FMS Warehouse Structure..... 29
    - Enterprise Service Automation (ESA) Data Mart..... 30
    - Enterprise Service Automation Data Mart Delivered Fact and Dimension Tables..... 31

General Ledger and Profitability Data Mart.....	45
General Ledger and Profitability Data Mart Delivered Fact and Dimension Tables.....	46
Payables Data Mart.....	52
Payables Data Mart Delivered Fact and Dimension Tables.....	53
Receivables Data Mart.....	55
Receivables Data Mart Delivered Fact and Dimension Tables.....	56
Shared Dimensions.....	59
<b>Chapter 4: Running the FMS Warehouse Implementation Jobs.....</b>	<b>65</b>
Prerequisites.....	65
Running FMS Warehouse Implementation Jobs.....	66
Running FMS - OWS Jobs.....	66
Running Global Dimension Jobs for FMS.....	67
Running Local Dimension Jobs for FMS.....	68
Running Global - OWE Jobs for FMS.....	69
Running FMS - OWE Jobs.....	70
Running FMS SKU Jobs.....	71
<b>Chapter 5: Activating Inactive Dimension Keys in Fact Tables.....</b>	<b>73</b>
Activating Inactive Dimension Keys in the PS_F_LEDGER Table.....	73
Configuring PS_F_LEDGER in Application Designer.....	73
Configuring the J_Fact_PS_F_LEDGER ETL Job in DataStage.....	74
Configuring the Related ETL Job J_Fact_PS_F_LEDGER_LEDGER_BUDG in DataStage.....	78
Activating Inactive Dimension Keys in the PS_F_BEGIN_BAL Table.....	82
Configuring PS_F_BEGIN_BAL in Application Designer.....	83
Configuring the J_Fact_PS_F_BEGIN_BAL Job in DataStage.....	84
Activating Inactive Dimension Keys in the PS_F_KK_LEDGER Table.....	91
Configuring PS_F_KK_LEDGER in Application Designer.....	92
Configuring the J_Fact_PS_F_KK_LEDGER ETL Job in DataStage.....	93
Activating Inactive Dimension Keys in the PS_F_KK_BALANCES Table.....	96
Configuring PS_F_KK_BALANCES in Application Designer.....	97
Configuring the J_Fact_PS_F_KK_BALANCES ETL Job in DataStage.....	98
<b>Chapter 6: Implementing Materialized View Logs.....</b>	<b>105</b>
Understanding Materialized View Logs.....	105
Implementing Materialized View Logs in the Oracle Database.....	106
Delivered Materialized View Logs for PeopleSoft Financial Management Source Tables.....	106
Configuring Non-PeopleSoft Delivered Materialized View Logs.....	108
OLTP Table Security.....	108
Working with Materialized View Logs in Delivered ETL Jobs.....	109
Describing Materialized View Log ETL Jobs.....	110
<b>Chapter 7: Configuring Slowly Changing Dimensions.....</b>	<b>115</b>
Understanding Slowly Changing Dimensions.....	115
Type 1 Slowly Changing Dimensions.....	115
Type 2 Slowly Changing Dimensions.....	116
Type 3 Slowly Changing Dimensions.....	117
Understanding Slowly Changing Dimensions in EPM.....	118
Valid Date Range Subrecord.....	118
Design Differences Between Type 1 and Type 2 Slowly Changing Dimension Jobs.....	119
Fact Table Jobs and Slowly Changing Dimensions.....	121
Converting Type 1 Slowly Changing Dimension Jobs to Type 2.....	122
Overview.....	122
Method 1: Converting a Type 1 Slowly Changing Dimension Job Using the Effective Date and Effective Sequence.....	122

Method 2: Converting a Type 1 Slowly Changing Dimension Job Without Using the Effective Date..... 134

Converting a Hash File Lookup to a Dynamic DRS Lookup in the Related Fact Table Job..... 135

**Chapter 8: Managing Source System Deletes..... 137**

    Understanding Source System Deletes and the Source-Delete Diagnostic Staging Jobs..... 137

    Identifying Source Record Deletes with CRC Staging Jobs..... 138

    Identifying Source Record Deletes with Date-Time Staging Jobs..... 148

    Enabling the Source-Delete Diagnostic Feature..... 153

    Adjusting the Source-Delete Diagnostic Option after Implementation..... 155

        Adjusting the Source-Delete Diagnostic Option for CRC Staging Jobs..... 155

        Adjusting the Source-Delete Diagnostic Option for Standard Staging Jobs..... 158

**Chapter 9: Implementing Audit Jobs..... 159**

    Understanding Audit Jobs..... 159

    Understanding Audit Job Implementation..... 160

    Creating Audit Triggers and Running Trigger Scripts..... 161

        Pages Used to Create Audit Triggers and Run Trigger Scripts..... 162

        Audit Triggers Page..... 162

        Run Audit Triggers..... 163

**Chapter 10: Implementing Currency Conversion for Multiple Currencies..... 165**

    Understanding Currency Conversion..... 165

    Understanding Currency Conversion Methodology..... 169

    Understanding Currency Conversion Rules..... 174

    Setting Up Currency Conversion..... 180

        Pages Used to Set Up the Schema Definition and Currency Conversion Rules..... 181

        Schema Definition Page..... 181

        MDW Currency Conversion Rule page..... 182

        MDW Conversion Schema Rule Page..... 184

    Running the ETL Currency Conversion Process..... 185

**Chapter 11: Setting Up Multilanguage Processing and Running the Language Swap Utility..... 187**

    Understanding Multilanguage Processing..... 187

    Understanding the Language Swap Utility..... 189

    Running the Language Swap Jobs..... 193

**Chapter 12: Processing Trees and Recursive Hierarchies..... 195**

    Understanding Tree and Recursive Hierarchy Processing..... 195

        Trees and Recursive Hierarchies..... 195

        OWE Tree Flattener Versus MDW Tree Denormalizer..... 196

        Hierarchies Supported by the Tree and Recursive Hierarchy Process..... 197

        Denormalized Tree Result Balancing..... 198

        Skip Levels..... 199

        Tree and Recursive Hierarchy Source Tables..... 201

        Multilanguage Support for Relationship and Hierarchy Tables..... 202

    Understanding Tree and Recursive Hierarchy Process Results..... 203

        Tree Flattener and Tree Denormalizer Output Tables..... 203

        Tree Flattener and Denormalizer Results..... 213

    Setting Up Parameters for Tree and Recursive Hierarchy Processing..... 219

        Pages Used to Run the Tree and Recursive Hierarchy Process..... 219

        Defining Parameters for the Tree and Recursive Hierarchy Process..... 220

        Relationship Record Definition..... 220

        Hierarchy Record Definition Page..... 221

        Hierarchy Group Definition Page..... 222

    Running the Tree and Recursive Hierarchy ETL Process..... 225

Running Hash File Hierarchy Jobs.....	226
Running OWS Hierarchy Jobs.....	226
Running Hierarchy Utility Jobs.....	226
<b>Chapter 13: Extending the Multidimensional Warehouse Data Model.....</b>	<b>229</b>
Considerations for Modifying an EPM Warehouse.....	229
Adding a Fact or Dimension Table to the Multidimensional Warehouse Data Model.....	230
Extending a Fact Table in the Multidimensional Warehouse Data Model.....	235
Adding a New Measure to a Fact Table.....	235
Adding a New Surrogate Key to a Fact Table.....	239
Extending a Dimension Table in the Multidimensional Warehouse Data Model.....	242
<b>Appendix A: Using the PeopleSoft EPM Lineage Spreadsheets.....</b>	<b>253</b>
Understanding the EPM Lineage Spreadsheets.....	253
Viewing Lineage Information.....	256
Finding Lineage Information for a Server Job.....	256
Identifying the List of Jobs to be Run for a Data Mart.....	260
Generating Lineage Information for a Job.....	261



# Preface

---

## Understanding the PeopleSoft Online Help and PeopleBooks

The PeopleSoft Online Help is a website that enables you to view all help content for PeopleSoft Applications and PeopleTools. The help provides standard navigation and full-text searching, as well as context-sensitive online help for PeopleSoft users.

### PeopleSoft Hosted Documentation

You access the PeopleSoft Online Help on Oracle's PeopleSoft Hosted Documentation website, which enables you to access the full help website and context-sensitive help directly from an Oracle hosted server. The hosted documentation is updated on a regular schedule, ensuring that you have access to the most current documentation. This reduces the need to view separate documentation posts for application maintenance on My Oracle Support, because that documentation is now incorporated into the hosted website content. The Hosted Documentation website is available in English only.

### Locally Installed Help

If your organization has firewall restrictions that prevent you from using the Hosted Documentation website, you can install the PeopleSoft Online Help locally. If you install the help locally, you have more control over which documents users can access and you can include links to your organization's custom documentation on help pages.

In addition, if you locally install the PeopleSoft Online Help, you can use any search engine for full-text searching. Your installation documentation includes instructions about how to set up Oracle Secure Enterprise Search for full-text searching.

See *PeopleTools 8.53 Installation* for your database platform, "Installing PeopleSoft Online Help." If you do not use Secure Enterprise Search, see the documentation for your chosen search engine.

---

**Note:** Before users can access the search engine on a locally installed help website, you must enable the Search portlet and link. Click the Help link on any page in the PeopleSoft Online Help for instructions.

---

### Downloadable PeopleBook PDF Files

You can access downloadable PDF versions of the help content in the traditional PeopleBook format. The content in the PeopleBook PDFs is the same as the content in the PeopleSoft Online Help, but it has a different structure and it does not include the interactive navigation features that are available in the online help.

### Common Help Documentation

Common help documentation contains information that applies to multiple applications. The two main types of common help are:

- Application Fundamentals

- Using PeopleSoft Applications

Most product lines provide a set of application fundamentals help topics that discuss essential information about the setup and design of your system. This information applies to many or all applications in the PeopleSoft product line. Whether you are implementing a single application, some combination of applications within the product line, or the entire product line, you should be familiar with the contents of the appropriate application fundamentals help. They provide the starting points for fundamental implementation tasks.

In addition, the *PeopleTools: PeopleSoft Applications User's Guide* introduces you to the various elements of the PeopleSoft Pure Internet Architecture. It also explains how to use the navigational hierarchy, components, and pages to perform basic functions as you navigate through the system. While your application or implementation may differ, the topics in this user's guide provide general information about using PeopleSoft Applications.

## Field and Control Definitions

PeopleSoft documentation includes definitions for most fields and controls that appear on application pages. These definitions describe how to use a field or control, where populated values come from, the effects of selecting certain values, and so on. If a field or control is not defined, then it either requires no additional explanation or is documented in a common elements section earlier in the documentation. For example, the Date field rarely requires additional explanation and may not be defined in the documentation for some pages.

## Typographical Conventions

The following table describes the typographical conventions that are used in the online help.

<b><i>Typographical Convention</i></b>	<b><i>Description</i></b>
Key+Key	Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For Alt+W, hold down the Alt key while you press the W key.
... (ellipses)	Indicate that the preceding item or series can be repeated any number of times in PeopleCode syntax.
{ } (curly braces)	Indicate a choice between two options in PeopleCode syntax. Options are separated by a pipe ( ).
[ ] (square brackets)	Indicate optional items in PeopleCode syntax.
& (ampersand)	When placed before a parameter in PeopleCode syntax, an ampersand indicates that the parameter is an already instantiated object.  Ampersands also precede all PeopleCode variables.
⇒	This continuation character has been inserted at the end of a line of code that has been wrapped at the page margin. The code should be viewed or entered as a single, continuous line of code without the continuation character.

## ISO Country and Currency Codes

PeopleSoft Online Help topics use International Organization for Standardization (ISO) country and currency codes to identify country-specific information and monetary amounts.

ISO country codes may appear as country identifiers, and ISO currency codes may appear as currency identifiers in your PeopleSoft documentation. Reference to an ISO country code in your documentation does not imply that your application includes every ISO country code. The following example is a country-specific heading: "(FRA) Hiring an Employee."

The PeopleSoft Currency Code table (CURRENCY\_CD\_TBL) contains sample currency code data. The Currency Code table is based on ISO Standard 4217, "Codes for the representation of currencies," and also relies on ISO country codes in the Country table (COUNTRY\_TBL). The navigation to the pages where you maintain currency code and country information depends on which PeopleSoft applications you are using. To access the pages for maintaining the Currency Code and Country tables, consult the online help for your applications for more information.

## Region and Industry Identifiers

Information that applies only to a specific region or industry is preceded by a standard identifier in parentheses. This identifier typically appears at the beginning of a section heading, but it may also appear at the beginning of a note or other text.

Example of a region-specific heading: "(Latin America) Setting Up Depreciation"

### Region Identifiers

Regions are identified by the region name. The following region identifiers may appear in the PeopleSoft Online Help:

- Asia Pacific
- Europe
- Latin America
- North America

### Industry Identifiers

Industries are identified by the industry name or by an abbreviation for that industry. The following industry identifiers may appear in the PeopleSoft Online Help:

- USF (U.S. Federal)
- E&G (Education and Government)

## Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

---

## Using and Managing the PeopleSoft Online Help

Click the Help link in the universal navigation header of any page in the PeopleSoft Online Help to see information on the following topics:

- What's new in the PeopleSoft Online Help.
  - PeopleSoft Online Help accessibility.
  - Accessing, navigating, and searching the PeopleSoft Online Help.
  - Managing a locally installed PeopleSoft Online Help website.
- 

## PeopleSoft EPM Related Links

[My Oracle Support](#)

[PeopleSoft Information Portal on Oracle.com](#)

[PeopleSoft Training from Oracle University](#)

[PeopleSoft Video Feature Overviews on YouTube](#)

---

## Contact Us

[Send us your suggestions](#) Please include release numbers for the PeopleTools and applications that you are using.

---

## Follow Us



Get the latest PeopleSoft updates on [Facebook](#).



Follow PeopleSoft on [Twitter@PeopleSoft\\_Info](#).

## Chapter 1

# Getting Started with PeopleSoft Financial Management Solutions Warehouse

---

## PeopleSoft Financial Management Solutions Warehouse Overview

The Financial Management Solutions (FMS) Warehouse serves both as a repository of information and as the foundation for business intelligence reporting. The Financial Management Solutions Warehouse draws data from PeopleSoft source transaction systems to stage, store, and enrich information for reporting. serves both as a repository of information and as the foundation for business intelligence reporting. PeopleSoft FMS Warehouse draws data from PeopleSoft applications to stage, store, and enrich information for reporting.

---

## Oracle's PeopleSoft Products

This PeopleBook refers to these products:

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

## 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* which also provides overview information about EPM, the Multidimensional Warehouse, and required setup tasks for the EPM warehouses.

### Related Links

PeopleSoft PeopleTools PeopleBook: PeopleSoft Setup Manager  
About These PeopleBooks



## Chapter 2

# Understanding 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 Customer Relations Management (CRM) Warehouse.
- PeopleSoft Campus Solutions (CS) Warehouse
- PeopleSoft Financial Management Solutions (FMS) Warehouse.
- PeopleSoft Financials Warehouse for Public Sector and Higher Education.
- PeopleSoft Human Capital Management (HCM) Warehouse.
- PeopleSoft 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*

## 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
- CRM for Higher Education
- Institutional Research
- Student Records
- Student Financials

## **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

## **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)

## 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 "Financials Warehouse for Public Sector and Higher Education Overview (*PeopleSoft EPM 9.1: Financials Warehouse for Public Sector and Higher Education*)"

## 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 ."HCM Warehouse Overview (*PeopleSoft EPM 9.1: Human Capital Management Warehouse*)"

## 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 ."Overview of PeopleSoft EPM Warehouses (*PeopleSoft EPM 9.1: Supply Chain Management Warehouse*)"

---

## 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 *"Understanding ETL in EPM (PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals)"*

## 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 *"EPM Core Infrastructure and ETL Setup Tasks (PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals)"*.

## 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 *"Understanding EPM Security and Setups (PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals)"*.

## 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 "Operational Warehouse - Staging (OWS) (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)".

## 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 "MDW Fact Tables (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)".

## 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 "MDW Dimension Tables (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)".

## 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:

<b>Common Dimension</b>	<b>Description</b>
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> <hr/> <p><b>Note:</b> 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> <hr/>

<b>Common Dimension</b>	<b>Description</b>
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 organization's 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>

<b>Common Dimension</b>	<b>Description</b>
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 supplier, business unit, a particular item, and so on. The Unit of Measure table facilitates this conversion process.</p> <hr/> <p><b>Note:</b> 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 EPM 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.

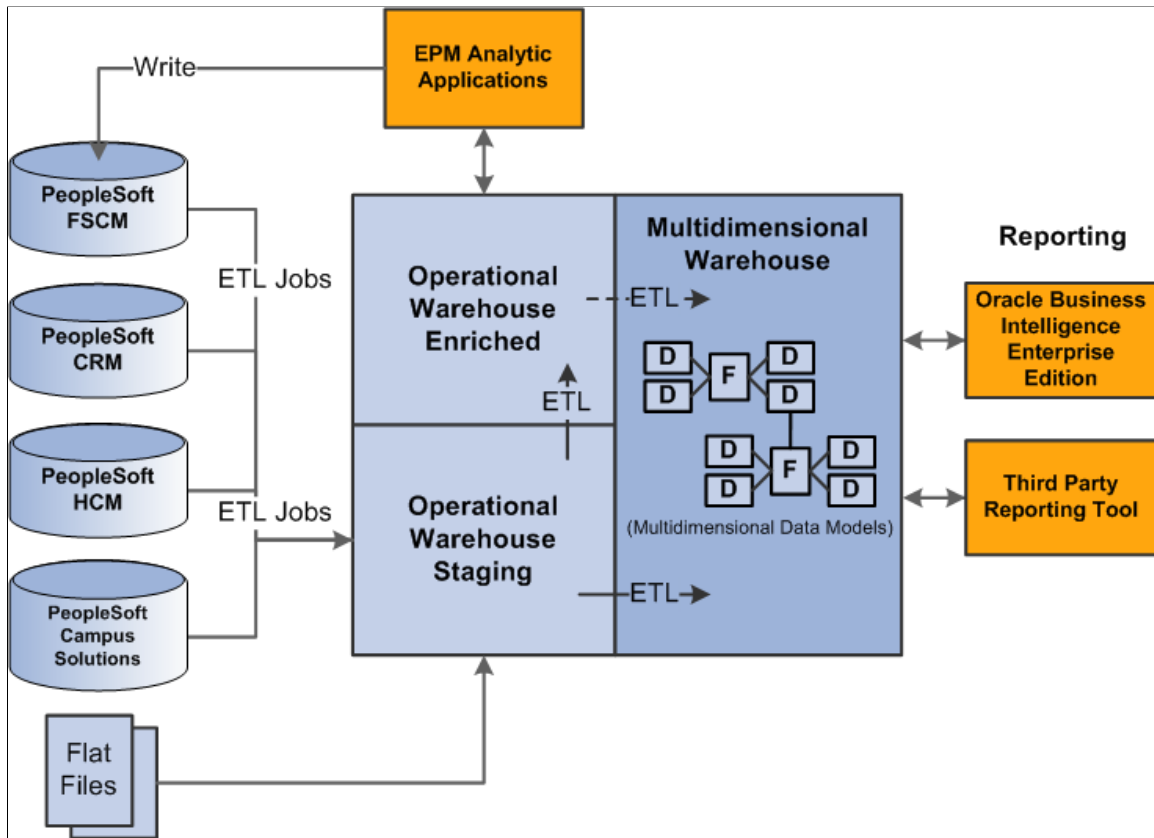
To bring source data into an EPM warehouse and prepare your data for reporting, you must run prepackaged ETL jobs that extract information contained in PeopleSoft source systems and load it into multidimensional warehouse data models:

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 sections 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 data mart topic in this documentation.

See "Understanding EPM Implementation (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)".

**Image: EPM Data Flow**

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:



**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. 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 "Operational Warehouse - Staging (OWS) (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)".

## 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 "Operational Warehouse - Enriched (OWE) (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)".

## 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 "Multidimensional Warehouse (MDW) (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)".

---

## 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.



# Understanding the FMS Warehouse

---

## FMS Warehouse Overview

The PeopleSoft FMS Warehouse is a comprehensive business intelligence platform for financials analytics. At the core of the FMS Warehouse are prepackaged dimensional data models, which optimize the arrangement, accessibility, and reportability of your payables, receivables, general ledger, and service automation data. With these data models you can review data against organizational metrics and perform strategic analysis using prepackaged measures such as debt-to-equity ratio, quick asset turnover, working capital-to-assets ratio, and receivables and payables turnover. The FMS Warehouse organizes your financials data into several easy-to-manipulate dimensions, including: Business Unit, Account, Time, Department, and Transaction Party. With the FMS Warehouse, you can identify and track multiple dimensions that affect profitability, identify key performance drivers, and determine profitability trends and opportunities for your organization.

---

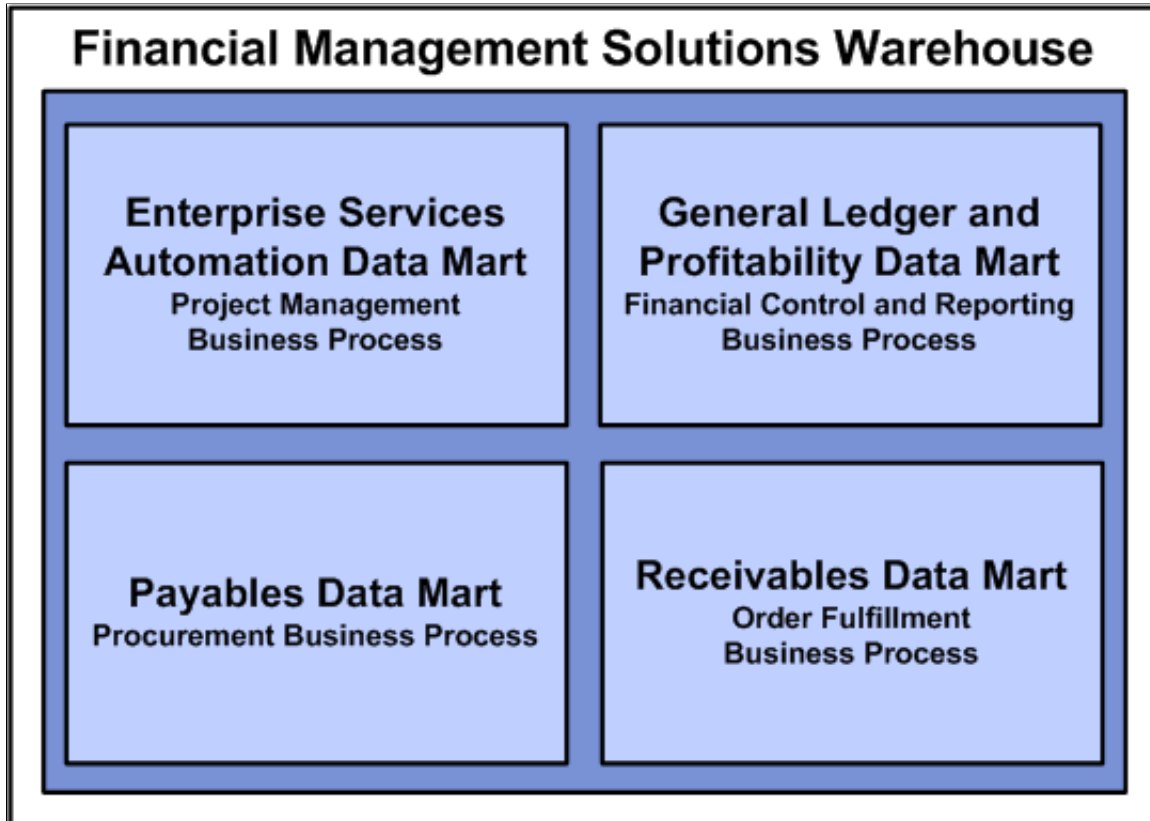
## Understanding the FMS Warehouse Structure

Data marts are logical divisions within the FMS Warehouse and are comprised of subject-specific dimensional data models designed around a specific institutional process. The FMS Warehouse includes

the Enterprise Service Automation (ESA) data mart, General Ledger and Profitability data mart, Payables data mart, and Receivables data mart.

**Image: FMS Warehouse Data Marts and Business Processes**

This example illustrates the fields and controls on the FMS Warehouse Data Marts and Business Processes. You can find definitions for the fields and controls later on this page.



Each data mart is associated with a business process that helps you answer the questions you need to keep your organization robust and ahead of its competition. With each data mart, PeopleSoft presents the associated business process and the fact tables that will help you answer your critical business questions.

This section discusses:

- Enterprise Service Automation (ESA) data mart
- General Ledger and Profitability data mart
- Payables data mart
- Receivables data mart

## Enterprise Service Automation (ESA) Data Mart

Analyzing projects and contracts is an increasingly important, yet difficult task in the growing services economy. Whether you are part of a professional services organization, an internal IT department, a research organization, or a capital project accounting department, the Enterprise Service Automation data mart helps you analyze project performance; contract profitability; resource utilization; and time and expense compliance. The ESA data mart includes data that you need to calculate standard measures such

as Proposal Pipeline, Project Profitability, Contract Revenue Recognized, Estimate to Complete, Resource Utilization, and Earned Value. With this information, you can measure your organization against external benchmarks. You can create a baseline for your own results, and then work towards improving project performance over time, as specified by your overall corporate strategy.

### **Project Management Business Process**

The ESA data mart is tied to PeopleSoft's Project Management business process. The Project Management business process fulfills an organization's requirements for the delivery of professional or internal services, such as planning, prioritizing, and managing resources and projects. If customer focused, this business process includes managing contracts, as well as contract changes, billing, and revenue information. Both the customer and internally focused process includes capturing project costs, adjusting projects, and settling expenses, and so on.

The information captured within the Project Management business process helps you determine answers to questions such as "How is my project performing?", "Is this contract profitable?", "What is the organization's resource utilization?" and "How much of the corporate expenses are compliant with policy?"

Create Project, Execute Project, and Assign Resources to Project are business sub-processes related to the ESA data mart. These business sub-processes aid you in areas such as creating projects and tasks, defining budget and resource requirements, tracking and adjusting projects.

### **Grant Analytics Business Process**

The ESA data mart is tied to PeopleSoft's Grants Management business process. The ESA data mart supports the data necessary for analyzing Grants activities, including *pre-award analytics*, to answer questions about outstanding proposals, funding, and budgets defined for proposals, and *post-award analytics*, to answer questions about post-award activities such as receipt of awards, project budgets, billing, receivables and expenditures activity related to the grant.

The information captured within the Grant Analytics business process helps you determine answers to questions such as:

- What proposals are outstanding and how much funding do they represent?
- What are the budgets defined for my research proposals?
- What F&A amounts are associated with my proposal budgets?
- What is the breakdown of funding requests by department and/or principal investigator?
- What is the total amount of proposals that were rejected versus funded?
- What amounts were budgeted, awarded, and proposed?

## **Enterprise Service Automation Data Mart Delivered Fact and Dimension Tables**

This section discusses the delivered fact and dimension tables for the ESA data mart.

### **Enterprise Service Automation Data Mart Fact Tables**

The following table describes the delivered ESA 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.

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
Assignment	F_ASGN	Represents information about assignments. (Assignments pair a resource with a unit of work.)	<p>Average length of an assignment.</p> <p>Average billing rate for an assignment.</p> <p>Number of hours scheduled against a particular assignment.</p> <p>Bill rate for an employee associated with an assignment.</p> <p>Bill rate for the project role associated with an assignment.</p> <p>Cost rate for the project role associated with an assignment.</p>
Contract Amendment	F_CA_AMD	Captures the change in the obligations and entitlements of a contract. The negotiated and discount amount changes are at the contract level.	<p>Total amount attributed to contract amendments in a period.</p> <p>Average percent increase due to contract amendments.</p> <p>Average gross amendment change for a given contract or customer.</p> <p>Total gross/negotiated change for a given contract or customer.</p> <p>Total discount and surcharge for a given contract or customer.</p>
Contract Distribution Transaction Detail	F_CA_DTL_DIST	Stores information relating to accounting distributions of a contract line.	<p>Average amount allocated to a given combination of General Ledger ChartFields.</p> <p>Total amount allocated to a given combination of General Ledger ChartFields.</p>



<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
Contract Forecast Current	F_CA_CUR_FRCST	Captures the current revenue backlog for both fixed-fee and rate-based contract lines and the probability percentage of the revenue forecast.	<p>Percentage change over time for a contract revenue forecast.</p> <p>Current total contract revenue forecast for a period.</p> <p>Total forecast revenue for a given contract, customer, product, or period.</p> <p>Probability associated with our forecasted contract revenue.</p>
Contract Transaction Detail	F_CA_DTL_TRANS	Stores information relating to a contract line by capturing the contract terms and revenue measures. The measures are at the contract line level of detail.	<p>Average discount given on a contract line.</p> <p>Largest or smallest discount given on a contract.</p> <p>Total discount given on all contract lines.</p> <p>Minimum contract line gross amount for a particular contract, customer, or product.</p> <p>Maximum contract line net amount for a particular contract, customer, or product.</p> <p>Average contract line list amount for a particular contract, customer, or product.</p>
Contract Forecast Periodic	F_CA_PRDC_FRCST	Captures the trend of revenue backlog for both fixed-fee and rate-based contract lines and the probability percentage of the revenue forecast.	See notes for Contract Forecast Current (F_CA_CUR_FRCST) table.

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
Contract Renewal	F_CA_RNW	Captures the contract renew amount and percentage increase over the different contracts and different renewal cycles.	<p>Average percentage increase in contract renewals.</p> <p>Total amount attributed to contract renewals for a period.</p> <p>Total contract renewal amount for a given contract, customer, product.</p> <p>Renewal percentage increase or decrease for a given contract, customer, product.</p>
Contract Revenue Recognition	F_CA_REV_RECOGN	Contains the revenue recognized and reversed accounting entries for a contract line.	<p>Total revenue amount recognized during a period.</p> <p>Revenue earned from a given customer.</p> <p>Average deferred revenue age for a given customer.</p>
Expense Distribution Transaction Detail	F_EXP_DTL_DIST	Stores measures related to expense transactions and captures the related accounting information and associated amounts.	<p>Average cost of an expense type (such as a hotel stay).</p> <p>Maximum amount spent on an expense type.</p> <p>Average number of nights spent in a hotel.</p> <p>Number of nights associated with an expense distribution level.</p> <p>Number of passengers associated with an expense distribution level.</p>

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
Expense Report Approval	F_EXP_RPT_APPRV	Stores measures related to an expense report's approval and captures information relating to approvers, types, and approval dates.	<p>Average length of time it takes for an expense report to be approved.</p> <p>Approvers taking the longest to approve expense reports.</p> <p>Number of expense approvals in a given period.</p> <p>Number of expense reports associated with a particular employee, by status.</p>
Employee Forecast	F_FCST	Stores information about employee forecasts.	<p>Revenue forecast for my organization for a given period.</p> <p>Forecast utilization for my organization for a given period.</p> <p>Total forecast amount.</p> <p>Number of hours an employee is forecasted to work on a given project.</p>
Grants Award	F_GM_AWARD	Contains award activity information, including the detail funding or budget information associated with an award.	<p>Awards this year verses last year.</p> <p>Sponsor awards funded.</p>
Grants Project Transactions	F_GM_PRJ_TRAN	Contains grant-specific project transaction activity information to enable you to analyze grants project budgets and actual transactions, such as expenses and revenues.	<p>Average number of research versus non research professionals per proposal/ activity.</p> <p>Total cost of a project.</p> <p>Total revenue of a project.</p> <p>Total cost of an activity.</p> <p>Total revenue for an activity for a specific accounting period.</p>

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
Proposal Details	F_GM_PROPOSAL	Contains details of the pre-award (proposal) business process, which enables analysis of proposed budget amounts at the proposal, version and proposed project level. The granularity is at business unit, proposal, version and proposed project. Proposal version activity is stored.	<p>Budgets defined for a research proposal.</p> <p>Direct costs of a proposed budget verses indirect cost.</p> <p>F&amp;A amounts associated with a proposal budget.</p> <p>Proposals with cost-sharing budgets with other institutions.</p> <p>Breakdown of proposal requests by department and/or PI (principal investigator).</p>
Proposal Award Summary	F_GM_PRP_AWD	Contains summary information of the proposal and award business processes at a very high level, for the purpose of success rate analysis. The granularity is at business unit and proposal. Proposal version activity is not stored. The Fact is of the type accumulating snap shot; existing rows in the fact table may need to be updated as proposal activity takes place in the source system.	<p>Total amount of proposals rejected verses funded.</p> <p>Proposal processing time from submission to funding.</p> <p>Breakdown of funding requests by department and/or PI (principal investigator).</p> <p>Success rate percentages calculated for both count and amount.</p>

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
Project Activity Current	F_PRJ_AC_CUR	Stores current project activities (tasks) measures.	<p>Actual cost of work performed (ACWP) for an activity.</p> <p>Budgeted cost of worked scheduled (BCWS) for an activity.</p> <p>Budgeted cost of work performed (BCWP) for an activity.</p> <p>Percent complete for an activity.</p> <p>Comparison of baseline to actual hours for an activity.</p> <p>Number of work hours required to complete an activity.</p>
Project Activity Periodic	F_PRJ_AC_PRDC	Stores trend project activities (tasks) measures.	See notes for Project Activity Current (F_PRJ_AC_CUR) table.
Project Change	F_PRJ_CHNG	Stores project change control information.	<p>Average number of changes per project.</p> <p>Change count by project, project activity, change type, change reason, user, or employee.</p>

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
Project Current	F_PRJ_CUR	Stores current project measures.	<p>Budgeted cost of worked scheduled for a project.</p> <p>Budgeted cost of work performed for a project.</p> <p>Actual cost of work performed for a project.</p> <p>Percent complete for a particular project.</p> <p>Comparison of budget to actual work hours.</p> <p>Comparison of actual to budget amounts for an activity.</p> <p>Number of billed hours associated with a project.</p> <p>Costs which have not been billed in the billing or accounts receivable system.</p>
Project Deliverable	F_PRJ_DELV	Stores project deliverable information.	<p>Average number of deliverables per project.</p> <p>Deliverables count by project, project activity, deliverable status, or employee.</p>
Project Issue	F_PRJ_ISS	Stores information about project issues. Project issues may be optionally related to a project activity.	<p>Average number of issues per project.</p> <p>Issue count by project, project activity, issue type, issue status, issue priority, or employee.</p>
Project Periodic	F_PRJ_PRDC	Stores trend project measure information.	See notes for Project Current ( F_PRJ_CUR) table.

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
Project Transaction	F_PRJ_TRAN	Stores project financial transactions data. With the exception of programs (summary projects), each project at the most basic level is composed of transactions. These transactions contain the quantity and amounts associated with each transaction.	<p>Number of hours billed against a project.</p> <p>Budget versus actual for a project.</p> <p>Total cost of a project.</p> <p>Total revenue of a project.</p> <p>Total cost of an activity.</p> <p>Total revenue for an activity.</p>
Proposal Grant Transaction Detail	F_PRP_VER_TRANS	Captures the information relating to proposals and grants. For proposals, it captures the different versions of project proposals created for business opportunities. For grants, it captures information relating to proposals, budgets, and so on.	<p>Average estimated margin percentage for all proposals.</p> <p>Percentage of proposals won.</p> <p>Probability that a proposal will be approved.</p> <p>Number of work days required for a given project role.</p> <p>Anticipated labor costs for a given project role.</p>

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
Resource Rate	F_RSRC_RT	Stores resource actuals, employee, and job code rates. It allows for analysis across different rates.	<p>Cost for a resource.</p> <p>Actual resource rate versus bill rate.</p> <p>Number of hours to complete an activity.</p> <p>Employee standard bill rate compared to the actual rate, job code rate, and project role rate for a given project activity.</p> <p>Actual bill rate compared to the job code bill rate for a particular project.</p> <p>Actual bill rate compare to the project role bill rate for a particular project.</p> <p>Project role cost rate compare to the job code cost rate for a particular project activity.</p>
Service Order	F_SO	Stores information about service orders (SO). SOs submit, track, and fulfill requests for services from external or internal customers. A SO defines the required services for a project and specifies the criteria that the system uses.	<p>Average number of days requested on a service order.</p> <p>Total number of resource requests in days.</p> <p>Number of estimated days for all service orders associated with a contract.</p> <p>Average project role bill rate for all service orders related to a project.</p> <p>Average project role cost for service orders associated with a particular contract.</p>



<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
Time Report Daily Detail	F_TIME_RPT	Stores information relating to a person's time report entry. It captures the related accounting information and associated hours. The details are captured at a day-level.	<p>Average number of billable hours entered in a period.</p> <p>Resource utilization of my organization in a period.</p> <p>Ratio of policy to billable hours for an employee during a given period.</p>

## Enterprise Service Automation Data Mart Dimension Tables

The following table describes the delivered ESA data mart dimension tables.

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
Project Analysis Group	D_ANLYS_GRP	Stores analysis group information. You can use analysis groups for analyzing projects and mapping analysis types.
Project Analysis Type	D_ANLYS_TYPE	Stores analysis type information. Analysis types are assigned to individual transactions to identify different types of transactions, such as estimated costs, budgeted amounts, actual costs, and billed costs.
Assignment	D_ASGN	Stores information about assignments. Assignments pair a resource with a unit of work.
Assignment Status	D_ASGN_STAT	The assignment status dimension represents information about the status of an assignment.
Billing Action	D_BLN_ACTN	The billing action dimension represents information (for example, Billable, Internal, or Nonbillable) about billing actions.
Contract Amend Reason	D_CA_AMD_RSN	Provides an explanation why an amendment was initiated. The amendment reason augments the amendment type and provides additional insight for the reason behind the amendment.

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
Contract Amendment Type	D_CA_AMD_TYPE	Stores information about the high-level classification of an amendment. Amendment types are dictated by internal amendment policies.
Contract Detail	D_CA_DTL	Stores contract detail information.
Contract Junk	D_CA_JNK	Stores a collection of low cardinality contract information.
Dist Account Type	D_DIST_ACCT	Stores information relating to the expense report's distribution account type.
Expense Location	D_EX_LOCATION	Stores the details of the expense location, such as location group, state, or country.
Expense Line Detail	D_EXP_LN	Stores information relating to an expense report line detail.
Expense Report	D_EXP_RPT	Stores information relating to an expense report.
Exp Approval Detail	D_EXP_RPT_APPRV	Stores information regarding the expense report approval process.
Employee Forecast	D_FCST	Stores information about employee forecasts.
Employee Forecast Approved	D_FCST_APPRV	Stores information, such as Approved, Denied, Sent Back for Revision, On Hold, Initial, Resubmitted, In Process, Rerouted, Submitted, about forecast approvals.
Award Attribute	D_GM_ATTR_CODE	Defines attributes that are associated with proposals and awards, which are used to classify them.
Grants Management Award	D_GM_AWARD	Defines a grant award. The award dimension is an extension of the contract dimension with grants award specific attributes.
Award Funding by Period	D_GM_BUD_PER	Stores the Budget Periods details for an award.
Grants Management Institution	D_GM_INST	Defines the Grants Management entity, Institution.

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
Award Keyword	D_GM_KEYWORD	Defines keywords that can be associated with awards so they can be categorized. The keyword provides the ability to view grant award amounts across multiple grants based on a select keyword value.
Grants Management Principal Investigator	D_GM_PI	Defines attributes that are associated with principal investigators, which are used to classify grants. It is an extension of the Employee dimension. Includes ethnicity, disabilities, fringe rates, conflict of interest, and so on.
Grants Management Project	D_GM_PRJ	Defines grant projects.
Grants Management Project Attributes	D_GM_PROP_PROJ	Defines attributes for proposed grants projects.
Grants Management Proposals	D_GM_PRP	Defines grant proposals.
Sponsor	D_GM_SPONSOR	Defines sponsors. This is an extension of the Customer dimension with grants-specific attributes.
Sponsor Type	D_SPNSR_TYPE	Defines attributes that classify sponsors.
Merchant	D_MRCHNT	Stores merchants available for selection in an expense report.
Project Activity	D_PRJ_AC	Stores information about a project's activities (tasks). This dimension rolls up to the project dimension.
Project AC Status	D_PRJ_AC_STAT	Stores information about user-defined activity statuses.
Project Change	D_PRJ_CHNG	Stores project change information.
Project Deliverable Status	D_PRJ_DELV_STAT	Stores the project deliverable status dimension.
Project Health	D_PRJ_HLTH	Stores information about project health. This information is maintained in a separate dimension to avoid forcing type-2 changes on the project dimension table or placing degenerate dimensions in the fact table.

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
Project Issue	D_PRJ_ISS	Contains unique combinations of issue priorities, statuses, and types.
Project Role	D_PRJ_ROLE	Stores information about project roles.
Project Resource	D_PRJ_RSRC	Stores information that classifies project financial transactions.
Project Status	D_PRJ_STAT	Stores information about user-defined project statuses.
Project Transaction	D_PRJ_TRAN_STAT	Stores combinations of resource transaction, business intelligence and GL distribution statuses in the data. Storing this data in one small dimension table reduces the number of surrogate keys in the fact table by two.
Proposal	D_PRP	Stores proposal information.
Proposal Detail	D_PRP_DTL	Stores proposal detail information.
Service Order	D_SO	Stores information about service orders (SOs). SOs are used to submit, track, and fulfill requests for services from external or internal customers.
Service Order Stat	D_SO_STAT	Stores information about service order statuses.
Task Type	D_TASK_TYPE	Stores information used to categorize tasks, such as Billable External, Billable Internal, Company Holiday, Corporate Event, Corporate Training, Education-College/Univ, Meeting, Nonbillable External, and Nonbillable Internal.
Time Reporting Code	D_TRC	Stores information about time reporting codes, such as Contract Holiday, Floating Holiday, Illness - Paid, Jury Duty, Personal - Paid, and Vacation.
User	D_USER	Stores information about the application's users.
Award Attributes <i>(Relationship Table)</i>	R_GM_AWD_ATTR	Links the one to many relationships between awards and custom attributes

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
Proposal Attributes <i>(Relationship Table)</i>	R_GM_PRP_ATTR	Links the one to many relationships between proposals and custom attributes
Award Keyword <i>(Relationship Table)</i>	R_GM_AWD_KEYWRD	Links the one to many relationships between awards and keywords.
Proposal Keyword <i>(Relationship Table)</i>	R_GM_PRP_KEYWRD	Links the one to many relationships between proposals and keywords.

## General Ledger and Profitability Data Mart

The General Ledger and Profitability data mart helps you analyze things such as the condition of assets, liabilities, and equity of an organization, as well as the profit and loss position of an organization. This data mart also enables your organization to compare results for a period across fiscal years. Additionally, it helps you determine which customers, products, and channels are most profitable and why. With it, you are able to rank customers based on revenue, gross margin, net income, or total expenses. You can identify changes in product mix, analyze profitability trends over time, and learn how each factor impacts costs, profit, and ultimately, the value of your organization. If you have purchased the Global Consolidations application, you can access this information through the Profitability data mart.

Global Consolidations introduces consolidation by using flows that enable you to track and reconcile the gross variation of an account. Gross variation is the difference between the opening and closing balances of an account. For example, the gross variation of a fixed asset account can be distinguished by additions, disposals, depreciation, asset impairment, current translation, and reclasses. Data flow of specific accounts is required as part of regulatory reporting, such as Security and Exchange Commission, International Accounting Standards, and so on. This feature enables you to identify and document the change in account activity, meet certain regulatory requirements, and provide financial statement footnotes.

---

**Note:** The Global Consolidation subject area is primarily sourced from the Operational Warehouse Enriched (OWE) tables GC\_CLED\_MGT\_F00 and GC\_FLOW\_MGT\_F00, which are the final output tables of the Global Consolidation application. The Profitability subject area is sourced from the OWE table PF\_LEDGER\_F00.

---

## Financial Control and Reporting Business Process

The Financial Control and Reporting business process fulfills an organization's requirements for recording, settling, consolidating, and reporting enterprise financial transactions. Using the Financial Control and Reporting business process, you capture and record financial transactions, transform financial information, allocate costs and adjust currencies, and close your books. You can use this information to report your results and to analyze your financial information, as well as to adjust your budgets, business events, and strategic plans. The Financial Control and Reporting business process helps you answer questions such as "How is my company performing against budget?", "How is my company performing against previous years?", "What is my company's cash flow position?" and "How strong is my company's liquidity and solvency?"

Analyze Financial Information is the business sub-process tied to the General Ledger and Profitability data mart, which aids you in analysis of areas such as creating allocations, revaluing balances, performing eliminations, recording period end adjustments, and consolidating data.

**Commitment Control Business Process**

The GL and the Profitability data mart enables you to view the current status of budgets, identify transactions associated with budgets, generate internal budget reports and reports to external parties. In addition to internal reporting and analytics, you can also perform regulatory or statutory reporting.

The Commitment Control business process helps you answer questions such as "What is my budget balance with and without encumbrances?" "What portion of my budgets were still encumbered at year-end and why?" "What remaining balances were rolled from the prior year?" "Which of my budgets typically ends the year with balances that are rolled?" "What is the budget to actuals variance average across a five year period?" and "Which budgets are being supplemented by revenues and by how much?"

**General Ledger and Profitability Data Mart Delivered Fact and Dimension Tables**

This section discusses the delivered fact and dimension tables for the General Ledger and Profitability data mart.

**General Ledger and Profitability Data Mart Fact Tables**

The following table describes the delivered General Ledger and Profitability 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.

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
GL Aggregated Balance	F_BEGIN_BAL	Contains aggregated period balances found in the F_LEDGER table and provides details such as requested budget amount, approved budget amount, and so on, which provides financial facts for the budget preparation process. You can use this information to provide financial facts for the budget preparation process.	Balance of an account at the end of last year.  Amount an account grew over the past year.  Requested budget amount.  Final budget amount after all adjustments have been applied.

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
Global Consolidations - Ledger	F_CLEDGER	This fact is an MDW version of GC_CLED_MGT_F00, which is the output of Global Consolidations process. By storing it in MDW, users can leverage the functionality available in MDW hierarchies.	Aggregate assets for all source ledgers.  Actual transactional amount posted to Global Consolidations from a source.  Actual amount in base currency posted to Global Consolidations from a source.
Global Consolidations - Flows	F_FLOWS	Based on output of Global Consolidations process, tracks and reconciles the gross variation of an account which can be due to additions, disposals, depreciation, asset impairment, and so on.	Breakdown of a fixed asset account by additions.  The impact fluctuations in currency conversion rates has on cash flow.
General Ledger Journal	F_JOURNAL	Stores all journal entries that are posted in GL and it provides an audit trail for activity affecting the ledger. Use this fact to determine journal entries that drove the change in ledger balance, for example.	Journal entries that drove the change for a particular ledger balance.
Commitment Control Activity Log	F_KK_ACTIVITY	Provides detailed transaction activity recorded for each budget ledger.	View transaction details for one or more budgets.  View the original transaction keys in the source systems.  Reconcile Audit Log activity to budgets.
Commitment Control Activity Fund Source Log	F_KK_ACT_FS	Captures detailed transaction activity recorded for a specific fund source.	Number of allocated funding sources spent or unspent.  Expenditures from a funding source across the organization and across multiple years with a breakdown by expenditure or revenue class.  Sum of allocated, unallocated, and available funds, and a breakdown of how the funds were spent.

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
Commitment Control Ledger Balances	F_KK_BALANCES	Contains details of Commitment Control ledger balances.	Budget that typically ends the year with rolled balances.  Budget balance with and without encumbrances.  Budget still encumbered at year-end.
Commitment Control Budget Association	F_KK_BUD_ASSOC	Captures information on revenue allocations, such as the method of allocation and revenue percent.	Budgets supplemented by revenues and by amount.  Number of budget accounts supplemented by a specific revenue budget.
Commitment Control Encumbrance and Liquidation	F_KK_ENCUMBRAN	Contains details on open encumbrances and relieving documents.	Identify vouchers that relieved encumbrances on a purchase order or budget account.  Identify any discrepancies or remaining reserved amounts that must be fully relieved.
Commitment Control Exception Log	F_KK_EXCEPTION	Contains budget checking errors activity. Errors are entered into this table each time a transaction line that is budget checked results in an error or warning	Number of budget checking exceptions for a particular budget account.  Number of budget check failures in the last week.  Top reasons for budget check failure.
Commitment Control Fund Source Allocation	F_KK_FS_ALLOC	Contains the monetary amounts of allocation as well as expenditure/revenue amounts recorded against the fund source.	Allocated fund sources for capital improvement projects.  Allocation methods, revenue percent, and caps.  Balance remaining for a fund source.  Number of unallocated funding sources.
Commitment Control Fund Source Received	F_KK_FS_RCVD	Captures the receipt of funds for a given fund source.	Funds spent for a specific period at the request of a foundation or grant sponsor.



<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
Budget Journal Fact Table	F_KK_JOURNAL	<p>Contains detailed information on budget journals. It enables you to:</p> <ul style="list-style-type: none"> <li>• Review budget adjustments for a business event, such as allocation of budgets for a special program or project.</li> <li>• Analyze the rate of adjustments made to original budgets by fiscal year and/or budget period.</li> <li>• Analyze budget transfer activity for a group of programs or projects over a multi-fiscal year period.</li> </ul>	Journal entries that drove the change for a particular ledger balance.
Commitment Control Ledger Detail	F_KK_LEDGER	Contains detailed information about Commitment Control ledgers.	<p>Budgets that are consistently overspent.</p> <p>Areas where revenues are not keeping up with spending.</p> <p>Spending pattern for my budgets.</p> <p>Revenue trend from utility fees over the last five years.</p> <p>Number of budgets that are strictly controlled.</p>
Commitment Control Override Log	F_KK_OVERRIDES	Contains information on budget overrides.	<p>Individual responsible for authorizing the most number of exception overrides.</p> <p>Number of overrides for particular budget account.</p>

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
Commitment Control Audit Log	F_KK_TRANS_LOG	Contains all events related to a transaction including changes made to distributions. If you are doing summary level budgeting you can use this data as a link between Commitment Control budget activity and General Ledger.	Transactions expensed against a budget account.  Percent of expenses incurred through procurement card transactions.  Purchase order types remaining open at year-end.
General Ledger Period Balances	F_LEDGER	Contains posted journals aggregated to the period level with a three-month rolling average value and helps in reporting on beginning and ending balances of accounts. It also provides the ability to compare results for a period across fiscal years.	Revenue recognized for a specific period.  Balance at the beginning of an accounting period.  Inventory value at the end of a period.  Amount the balance grew during a given accounting period.  Average amount for an account over the past six months.
Profitability Analysis	F_PROFITABILITY	Contains ledger level measures sourced from the PF_LEDGER_F00 table, which is populated by PeopleSoft Allocation Manager and Activity Based Management processes. This table helps you analyze profitability using dimensions such as Customer, Product, Channel, and so on.	Activity level for specific business activities, business units or entities.  Relative activity for specific business activities, business units or entities across time, within or across fiscal years.  Spending trend for specific expense line items across specific organizational units.  Revenue trend for specific products or services within a region or across organizational units.  Balance growth during the accounting period.

### General Ledger and Profitability Data Mart Dimension Tables

The following table describes the delivered General Ledger and Profitability data mart dimension tables.

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
Activity Based Manager Data	D_ABM_OBJECT	Stores activity based manager data used in profitability analysis.
Alternate Account	D_ALTACCT	Stores alternates to original accounts. Corporations use these mainly for accounting purposes.
Billing Status	D_BI_STATUS	Stores billing status. Possible values are Cancelled, New Bill, Hold Bill, and so on.
Billing Type	D_BI_TYPE	Stores billing types. Possible values are Asset Bills, Service Bills, and so on.
Commitment Control Ledger Group	D_BU_LED_GRP	Identifies the ledger groups associated with a business unit.
Expansion 1	D_DIMENSION1	This is an user defined expansion dimension.
Expansion 2	D_DIMENSION2	This is an user defined expansion dimension.
Expansion 3	D_DIMENSION3	This is an user defined expansion dimension.
Consol Ledger Source	D_GC_CLED_SRC	Identifies the source of Global Consolidations entry as Equalization, Elimination, and so on.
Flow Codes	D_GC_FLOW_CD	Describes the system and manual flow codes used to indicate the Global Consolidations flow process.
Flow Rates	D_GC_FLOW_RATE	Stores the rate at which the flow transaction was made, either Cash, Closing, or Both.
Flow Sources	D_GC_FLOW_SRC	Stores the source that generated the flow transaction, either Batch, Journal, or Source Input.
Global Consolidations Company Location	D_GC_LOCATION	Stores details about the location of the business unit associated with Global Consolidations.
Commitment Control Budget Attributes	D_KK_BUDG_JNK	Identifies all of the relevant translate values for Commitment Control attributes.

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
Commitment Control Budget Ledger Definition	D_KK_BUDG_TYPE	Identifies the attributes of a Commitment Control budget ledger group.
Commitment Control Fund Source	D_KK_FUND_SRCE	Identifies details about the fund source and includes information such as fund source type, reimbursement agreement, agency, and total fund source amount.
Commitment Control Source Transaction Definition	D_KK_SRC_TRAN	Identifies the type of transaction that was processed by Commitment Control, such as Purchase Order, Payables Voucher, and GL Journal.
Commitment Control Transaction Type	D_KK_TRANS_TYPE	Identifies the type of budgetary amount as original or adjusted or transfer of either type.
Line of Business	D_LINE_OF_BUS	Stores details about line of business of the business unit associated with Global Consolidations.
Scenario and Calendar	D_PFBUSCEN_DFN	Sourced from PF_BU_SCEN_DFN and is created in the MDW for metadata purposes only.
Commitment Control Project Resource Type	D_PRJ_RSRC_TYPE	Identifies resource types owned by a project.
Subsidiary	D_SUBSIDIARY	Stores information on subsidiary.

## Payables Data Mart

You use the Payables data mart to report and analyze the performance of your suppliers and your payables department. The Payables data mart includes information about suppliers, vouchers, match exceptions, and payments. This data mart helps you to assess your supplier relationships by answering questions such as "Which suppliers have billing errors?" and "How long does it take us to pay suppliers?" With this information, you can ask suppliers for better service or negotiate with them for better terms. To measure performance of your payables department, this mart helps answer questions such as "What are my accounts payable aging balances?", "Does my organization take advantage of supplier discounts?" and "How many vouchers are entered per day?" You can analyze results across many dimensions, enabling you to compare performance of multiple payables departments or even individual employees. After you understand current performance levels, you can set targets and monitor results.

The Payables data mart includes data needed to calculate standard measures such as Days Payable Outstanding, Accounts Payable Turnover, and Sales to Accounts Payable Ratio. With this information, you can measure your organization against external benchmarks. You can baseline your own results and then work towards improving payables performance over time, as specified by your overall corporate strategy.

## Procurement Business Process

The Payables data mart is tied to PeopleSoft's Procurement business process, which is also known as the Source to Settle business process. The Procurement business process fulfills an organization's requirements for sourcing, engaging, procuring, and settling payment for goods, services, or both. The Procurement business process enables you to determine profitability sourcing strategies, collaborate with suppliers, and drive efficient procurement and settlement for all goods and services.

Pay Supplier is the business sub-process related to Payables data mart, which aids you in areas such as creating and matching vouchers, resolving disputes with suppliers, and creating payment.

## Payables Data Mart Delivered Fact and Dimension Tables

This section discusses the delivered fact and dimension tables for the Payables data mart.

### Payables Data Mart Fact Tables

The following table describes the delivered Payables 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.

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
AP Account Line Entries	F_AP_ACCOUNT_LN	Stores all of the AP accounting entries posted by the AP module.	Posting details of the AP transactions to GL for a specified time period.  Journal amount posted to the GL for a selected AP account by AP transactions.
AP Aging Process	F_AP_AGING	Contains the aging information by aging category and supplier and can be used in analysis.	Money owed for a specific period.  Highest balance for a supplier.  Highest past due for a supplier.  Aging trends of accounts payable.  Total discount amount available in AP Vouchers.  Total prepaid amount to a specified supplier.  Total AP outstanding amount to be paid to a specified supplier.

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
AP Transactions	F_AP_TRAN	Contains the summary of AP transactional information at a header level and can be used to analyze vouchers and payments made by a specific supplier. Various amounts such as Paid, Open, Discount, Late Charge, and so on are available.	<p>Total number of AP vouchers and AP payments.</p> <p>Total AP voucher amount and total AP payment amount by supplier.</p> <p>Current status of the AP transactions.</p> <p>Voucher, adjustment and payment history for a supplier.</p> <p>Discount eligible to be deducted if AP vouchers are to be paid as per due date.</p> <p>Total late charge amount to a particular supplier over time.</p> <p>Late charge denied amount to a particular supplier.</p>
AP Voucher Match Exceptions	F_VCHR_MTCH_EXP	Contains information about all failed transactions of the AP matching process and can be used to report the number of exceptions generated by the process, by supplier and AP operator ID.	Number of exceptions generated by the AP voucher matching process, by supplier and AP operator ID.

### Payables Data Mart Dimension Tables

The following table describes the delivered Payables data mart dimension tables.

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
Aging Category	D_AGING_CTGRY	Stores aging categories used in AP and AR aging analysis. The AP_AR_IND field classifies the aging category as AP or AR.
AP Payment Transaction Status	D_AP_PTR_STAT	Stores accounts payable (AP) transaction status from a payment perspective. Possible values are Paid, Stopped, Void, and so on.

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
AP Payment Transaction Type	D_AP_PTR_TYPE	Stores AP transaction types from a payment perspective. Possible values are Express Payment, Manual Payment, Regular Payment, and so on.
AP Voucher Transaction Status	D_AP_VTR_STAT	Stores the AP voucher transaction status. Possible values are Approved, Denied, Pending, and so on.
AP Voucher Transaction Type	D_AP_VTR_TYPE	Stores AP voucher transaction types. Possible values are Adjustment Voucher, Prepaid Voucher, and do on.
AP Post Status	D_APPOST_STATUS	Stores AP posting status. Possible values are Posted, Unposted, Payment Not Applied, and so on.
AP Operator	D_PERSON_APOPID	Stores details such as name and contact details about the AP operator who handles the payments in the AP module.

## Receivables Data Mart

Accounts receivable are often the largest asset on a company's balance sheet and are critical to its financial health. While extending credit to the customers helps boost sales, it also increases the cost of working capital and financial risk, especially in challenging economic times.

You use the Receivables data mart to analyze your company's receivables. With it you can perform multidimensional analysis against your receivables activities and receivables information to see how much is due from customers, view accounts receivable transactional activities, analyze customer credit worthiness and risk, and conduct account receivable account level analysis.

The Receivables data mart helps you identify areas within your organization contributing to higher days sales outstanding, increased credit risk, poor collection performance, or more bad debt write-offs. In essence, it helps you better understand your receivables asset and identify areas for improvement within your organization that will enable you to improve the quality of your accounts receivable and reduce the cost of working capital.

### Order Fulfillment Business Process

The Receivables data mart is related to PeopleSoft's Order Fulfillment business process, which is also known as Order to Cash. The Order Fulfillment business process fulfills an organization's requirements for capturing, fulfilling, and settling goods sold. With the Order Fulfillment business process, you capture, confirm, and manage sales orders and contracts, deliver goods or services, and then invoice, collect, and resolve payment. The Order Fulfillment business process also helps you to manage returns and inventory, process customer payments, and maintain profitable customer relationships. The information captured within the Order Fulfillment business process helps you determine answers to questions such as "What are my cost per sales?", "What is my ending receivables balance?" and "What is my bad debt compared to sales?"

Process Customer Payments is the business sub-process related to the Receivables data mart, where you receive and apply customer payments, maintain customer balances, manage past due accounts, and resolve payment discrepancies.

## Receivables Data Mart Delivered Fact and Dimension Tables

This section discusses the delivered fact and dimension tables for the Receivables data mart.

### Receivables Data Mart Fact Tables

The following table describes the delivered Receivables 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.

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
AP-AR Netting	F_APAR_NETTING	Contains information about the AR amount outstanding from a customer and the AP amount outstanding from the company to that customer (if that customer is also a supplier to the company). This provides a net total for the customer or supplier.	<p>Net balance for a customer or supplier.</p> <p>Highest balance for a customer or supplier.</p> <p>Highest past due for a customer or supplier.</p> <p>Outstanding balance for a customer, with respect to the customer's credit limit.</p> <p>AR-AP Netting Balance Amount for customers who are also suppliers.</p> <p>Aging trends of accounts payable.</p>
AR Account Line Entries	F_AR_ACCOUNT_LN	Contains all AR accounting entries and can be used to answer questions related to the journal amount posted to GL for the selected AR accounts by AR transactions.	Journal amount posted to the general ledger for the selected AR account by AR Transactions.



<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
AR Aging process	F_AR_AGING	Contains the aging information by aging category and customer.	<p>Money owed for a specific period, and percentage past due.</p> <p>Highest past due amount for a customer.</p> <p>Aging trends of my accounts receivable.</p>
AR Credit Limit	F_CREDIT_LIMIT	Contains information about credit allocated to a customer based on the customer's credit rating.	<p>Available credit and credit limit by customer.</p> <p>Customer credit limit percentage over limit and how much has the customer actually exceeded this credit limit range.</p> <p>Outstanding AR amount and by what percentage the AR outstanding above or below the credit limit.</p>
AR Days Sales Outstanding	F_AR_DSO	Contains Days Sales Outstanding (DSO) information for accounting periods and helps to analyze the average time taken to turn the receivables into cash.	Days sales outstanding, by customer.

<b>Fact Name</b>	<b>Fact Record Name</b>	<b>Description</b>	<b>Helps Answer</b>
AR Transactions	F_AR_TRAN	<p>Stores summary of AR transactions and helps in analysis of total AR invoices and payments, AR invoice amount, AR receivables amount by a customer, and so on.</p> <p>This table helps you determine factors such as the AR Transactions Performance metrics, for example, transaction amounts, discount amounts, and so on.</p>	<p>Total AR invoice amount and total AR payment amount, by customer.</p> <p>Current status of AR transactions.</p> <p>Invoice, adjustment and payment history for a customer.</p> <p>Earned discount amount for a specified customer in a given time period.</p> <p>Dispute amount for a specified customer in a given time period.</p> <p>Transaction activity for a customer for a specific period of time.</p>

### Receivables Data Mart Dimension Tables

The following table describes the delivered Receivables data mart dimension tables.

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
AR Document Type	D_AR_DOC_TYPE	Stores details about various AR document types such as Receivables Adjustments, Receivables Revaluation, and so on.
AR Item Transaction Status	D_AR_ITR_STAT	Indicates the transaction status of an item as Closed or Open.
AR Item Transaction Type	D_AR_ITR_TYPE	Stores AR Item transaction types. There is no source for this table, a single row is created with hardcoded values.
AR Payment Transaction Status	D_AR_PTR_STAT	Stores AR payment status. Possible values are Applied, Complete, Unidentified, and so on.
AR Payment Transaction Type	D_AR_PTR_TYPE	Stores AR Payment transaction types. There is no source for this table, a single row is created with hardcoded values.

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
Collection Status	D_COLLECT_STAT	Stores collection status codes and descriptions.
Deduction Status	D_DEDUCT_STAT	Stores AR deduction status information.
Dispute Status	D_DISPUTE_STAT	Stores details about disputes raised in AR collections.
Entry Reason Type	D_ENTRY_RSTYP	Stores reasons used when an entry in GL is created from AR. Possible values are Adjustment Overpayment, Credit Memo, and so on.

## 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.

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
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.

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
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.
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.

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
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.
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.

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
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.
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.

<b>Dimension Name</b>	<b>Dimension Record Name</b>	<b>Description</b>
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.
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.

<b><i>Dimension Name</i></b>	<b><i>Dimension Record Name</i></b>	<b><i>Description</i></b>
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 the FMS Warehouse Implementation Jobs

---

## Prerequisites

After you have configured IBM WebSphere DataStage for EPM, and before you run FMS 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 FMS Warehouse.
- Run the Dimension Mapper setup jobs.
- Run shared lookup jobs.
- Run the Setup - OWE jobs.
- Run the Common Dimensions jobs.
- Configure the DATA\_VALIDATION\_SETUP and HANDLEDELETES\_SETUP parameter files per your requirements (for data validation and source delete ETL jobs).

For more information about these prerequisites, see:

- *PeopleSoft Enterprise Performance Management Fundamentals*.
- [Understanding Source System Deletes and the Source-Delete Diagnostic Staging Jobs](#)

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 "Understanding ETL in EPM (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)", "Verifying ETL Components Have Imported Properly (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)", "Understanding Warehouse Business Units, TableSet Sharing, and SetID Mapping (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)", "Using the Master Sequencer Utility to Create Master Sequencer Jobs (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)".

---

## Running FMS Warehouse Implementation Jobs

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

1. FMS - OWS jobs.
2. Global dimension jobs for FMS.
3. Local dimension jobs for FMS.
4. Global - OWE jobs for FMS.
5. FMS - OWE jobs.
6. FMS - SKU jobs.

### Running FMS - OWS Jobs

The first step in implementing the FMS Warehouse is to run the FMS - OWS jobs. These jobs consist of FMS-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.

See "Verifying ETL Components Have Imported Properly (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)".

#### FMS - OWS Hash File Jobs

Perform the following steps to run the FMS - 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: *FMS\_E, OWS, Base, Load\_Hash\_Files, Server*.
2. Select each FMS - 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*.

## FMS - OWS Jobs

Perform the following steps to run the FMS - OWS jobs:

1. In IBM WebSphere DataStage Director, navigate to the FMS - OWS jobs by expanding the nodes in the left navigation panel using the following path: *FMS\_E, OWS, Base, Load\_Tables, Sequence*.
2. Select each FMS - 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 FMS

The second step in implementing the FMS Warehouse is to run the global dimension jobs for FMS. 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 FMS

The third step in implementing the FMS Warehouse is to run the local dimension jobs for FMS. These jobs consist of local dimension hash file jobs and local 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 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: *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: *FMS\_E, Local\_Dimensions, Master\_Sequence*.

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

The Job Run Options box appears.

2. 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: *FMS\_E, Local\_Dimensions, OWS\_To\_MDW, 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: *FMS\_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 Global - OWE Jobs for FMS

The fourth step in implementing the FMS Warehouse is to run the Global - OWE jobs for FMS. These jobs consist of FMS Global - OWE hash file jobs and standard Global - OWE jobs. Run the hash file jobs first, as the tables that they load are required to run your standard FMS Global - OWE jobs.

### Global - OWE Hash File Jobs

Perform the following steps to run the Global - OWE hash file jobs individually:

1. In IBM WebSphere DataStage Director, navigate to the Global - OWE hash file jobs by expanding the nodes in the left navigation panel using the following path: *OWE\_E, Global\_D00, Base, Load\_Hash\_Files, Server*.

2. Select each Global - OWE 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*.

### **Global - OWE Dimension Jobs**

Perform the following steps to run the Global - OWE dimension jobs individually:

1. In IBM WebSphere DataStage Director, navigate to the Global - OWE dimension jobs by expanding the nodes in the left navigation panel using the following path: *OWE\_E, Global\_D00, Base, Load\_Tables, Sequence*.

2. Select each Global - OWE 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*.

## **Running FMS - OWE Jobs**

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

### **FMS - OWE Hash File Jobs**

Perform the following steps to run the FMS - OWE hash file jobs individually:

1. In IBM WebSphere DataStage Director, navigate to the FMS - OWE hash file jobs by expanding the nodes in the left navigation panel using the following path: *OWE\_E, FMS, Base, Load\_Hash\_Files, Server*.

2. Select the each FMS - OWE 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*.

## FMS - OWE Jobs

Perform the following steps to run the FMS - OWE jobs individually:

1. In IBM WebSphere DataStage Director, navigate to the FMS - OWE jobs by expanding the nodes in the left navigation panel using the following path: *OWE\_E, FMS, Base, Load\_Tables, Sequence*.
2. Select each FMS - OWE 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 FMS SKU Jobs

The sixth and final step in implementing the FMS Warehouse is to run the FMS 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 to run FMS SKU jobs individually or together using the master sequence job.

---

### FMS SKU Hash File Jobs

Perform the following steps to run the FMS SKU hash file jobs individually:

1. In IBM WebSphere DataStage Director, navigate to the FMS SKU hash file jobs by expanding the nodes in the left navigation panel using the following path: *FMS\_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 FMS 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: *FMS\_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*.

### FMS Dimension Jobs

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

1. In IBM WebSphere DataStage Director, navigate to the FMS dimension jobs by expanding the nodes in the left navigation panel using the following path: *FMS\_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 FMS 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: *FMS\_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*.

## **FMS Fact Jobs**

Perform the following steps to run the FMS fact jobs individually:

1. In IBM WebSphere DataStage Director, navigate to the FMS fact jobs by expanding the nodes in the left navigation panel using the following path: *FMS\_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: *FMS\_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

# Activating Inactive Dimension Keys in Fact Tables

## Activating Inactive Dimension Keys in the PS\_F\_LEDGER Table

The PS\_F\_LEDGER fact table is delivered with the following active dimension keys:

<i>Active Dimension Keys</i>	<i>Active Dimension Keys</i>
BU_SID	OPER_UNIT_SID
PPERIOD_SID	FUND_CODE_SID
LEDGER_SID	PRODUCT_SID
ACCOUNT_SID	PROJECT_SID
ALT_ACCT_SID	BUDGET_REF_SID
DEPT_SID	STAT_CODE_SID
SUBLEDGER_SID	SCENARIO_SID
AFFILIATE_BU_SID	CURRENCY_CD

The following inactive columns can be activated as additional dimension keys:

<i>Inactive Dimension Key Columns</i>	<i>Inactive Dimension Key Columns</i>
CBU_SID	ADJUST_TYPE_SID
PROGRAM_FDM_SID	CLASS_FIELD_SID
AFF_OU_SID	CHARTFIELD1_SID
AFF_FUND_SID	CHARTFIELD2_SID
BOOK_CODE_SID	CHARTFIELD3_SID

To activate inactive dimension keys, you will use the PeopleSoft Application Designer and IBM WebSphere DataStage tools.

## Configuring PS\_F\_LEDGER in Application Designer

To activate additional dimension keys in the PS\_F\_LEDGER table:

1. Open PeopleSoft Application Designer.
2. Select File, Open from the menu.
3. In the Open Definition dialog box, select *Record* for the Definition field, then enter *F\_LEDGER*.  
The Record Fields tab for the PS\_F\_LEDGER table appears. Note the inactive dimension keys.
4. In the Record Fields tab, right-click an inactive dimension key and select *Record Field Properties*.  
The Record Field Properties window appears for the column.
5. In the Use tab of the Record Field Properties window select the Key and Search Key check boxes.
6. Click OK to continue.
7. Save your changes.
8. Select Build, Project from the menu.  
The Build window appears.
9. In the Build window select the Create Indexes and Execute and Build Script check boxes.
10. Select Settings.  
The Build Settings window appears.
11. In the Create tab of the Build Settings window, select the Recreate Index only if Modified check box.
12. Click OK to continue.
13. Click Build to complete your configuration of the PS\_F\_LEDGER table.
14. Repeat steps 4 through 13 to activate other inactive dimension keys in the PS\_F\_LEDGER table.

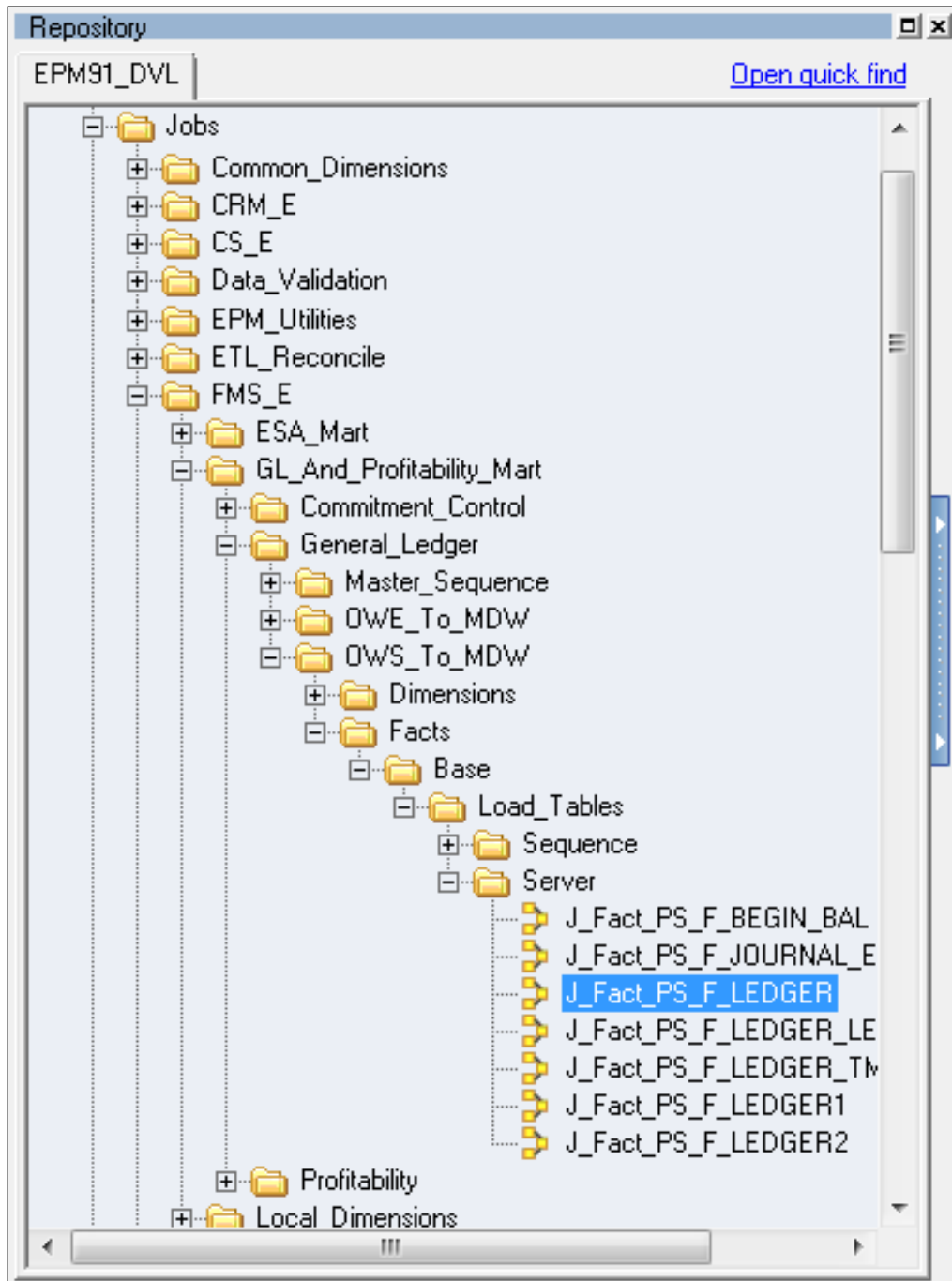
## Configuring the J\_Fact\_PS\_F\_LEDGER ETL Job in DataStage

To configure the new keys in the J\_Fact\_PS\_F\_LEDGER job:

1. In DataStage Designer client, navigate to the J\_Fact\_PS\_F\_LEDGER job in the project tree and open it for editing.

**Image: J\_Fact\_PS\_F\_LEDGER job in the project tree**

This example illustrates the fields and controls on the J\_Fact\_PS\_F\_LEDGER job in the project tree. You can find definitions for the fields and controls later on this page.

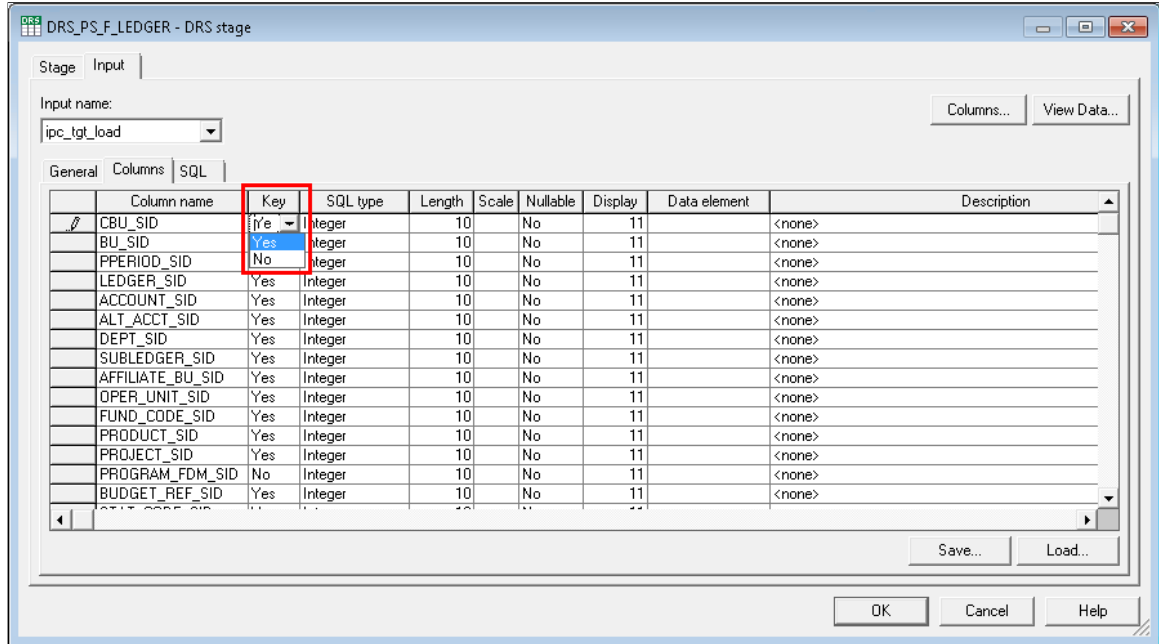


2. Open the DRS target stage (DRS\_PS\_F\_LEDGER) and select the Inputs tab.
3. Select the Columns sub-tab and locate each column you have activated as a key in Application Designer.

- For each column you have activated as a key in Application Designer, select *Yes* for the Key field.

**Image: Specifying keys in the DRS\_PS\_F\_LEDGER target stage**

This example illustrates the fields and controls on the Specifying keys in the DRS\_PS\_F\_LEDGER target stage. You can find definitions for the fields and controls later on this page.



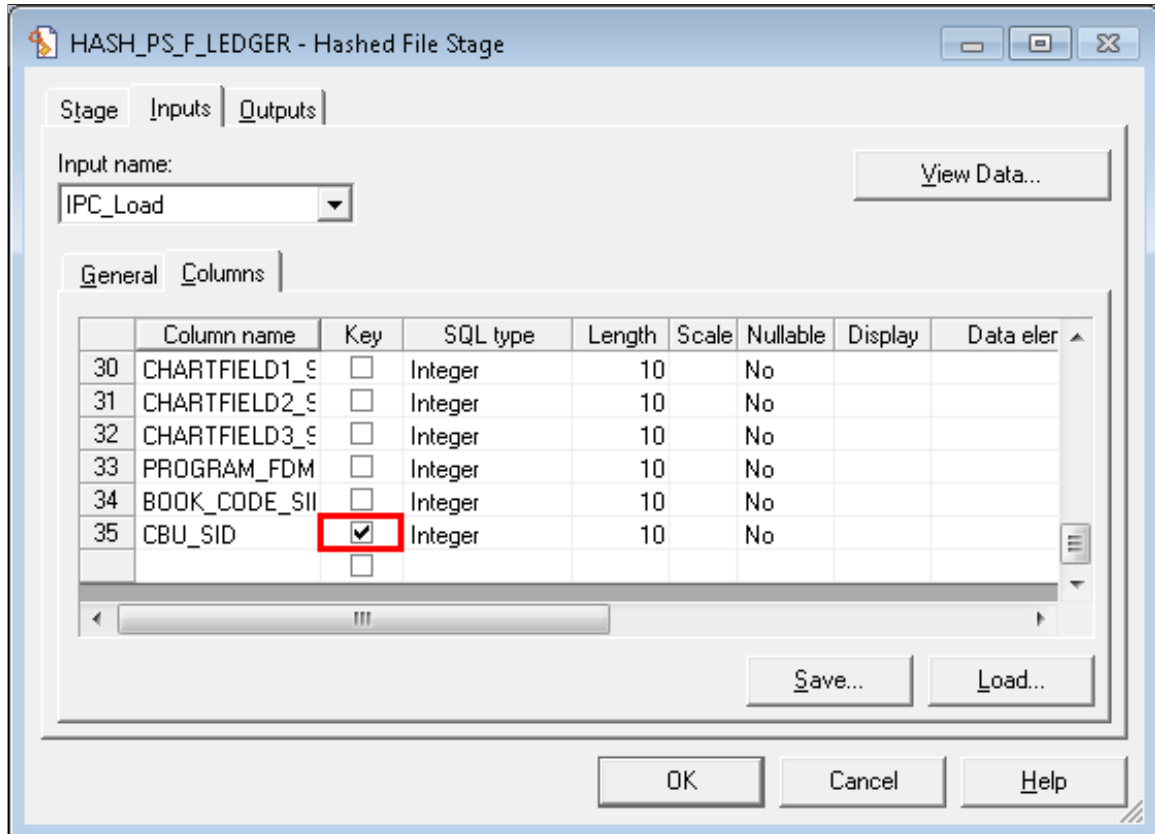
**Note:** Ensure that you add all the same keys that you added in Application Designer.

- Click OK to continue, then save your changes.
- Repeat steps 2 through 5 for the DRS\_PS\_F\_LEDGER\_UPD target stage.
- Open the HASH\_PS\_F\_LEDGER hashed file to configure the new keys in the lookup.
- In the HASH\_PS\_F\_LEDGER hashed file, select the Inputs tab, then the Columns sub-tab.

9. Select the Key field check box for each column you have activated as a key in the target DRS stage.

**Image: Specifying keys in the HASH\_PS\_F\_LEDGER hashed file**

This example illustrates the fields and controls on the Specifying keys in the HASH\_PS\_F\_LEDGER hashed file. You can find definitions for the fields and controls later on this page.

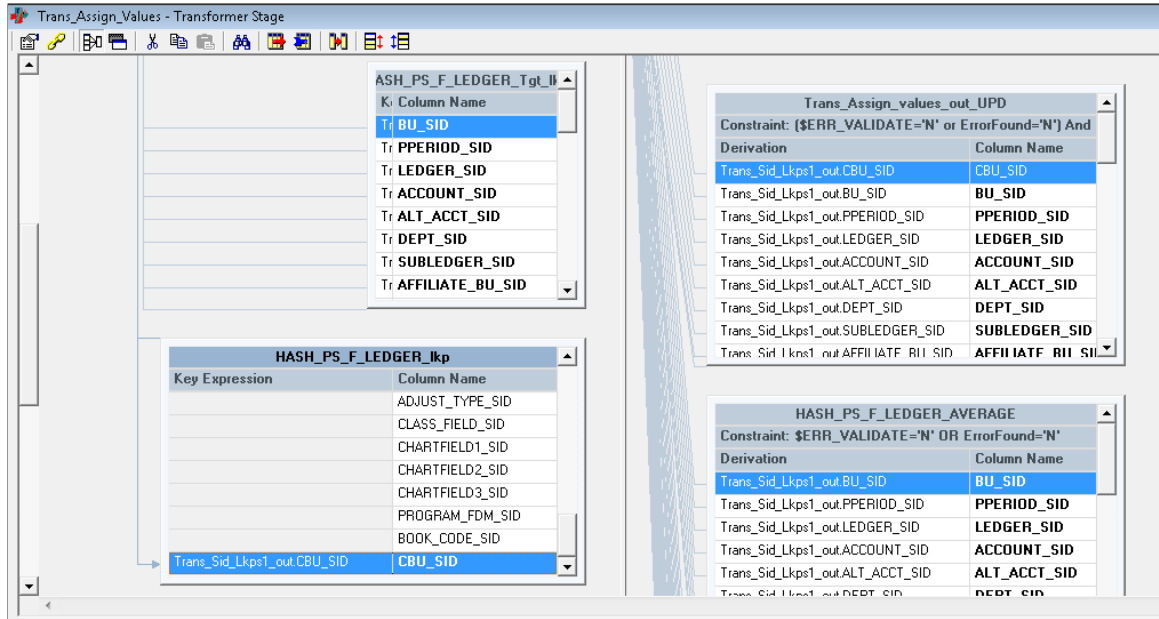


10. Click OK to continue, then save your changes.
11. Repeat steps 7 through 10 for the HASH\_F\_LEDGER\_INC\_UPDATE and HASH\_PS\_F\_LEDGER\_AVERAGE hashed files.

- Open the Trans\_Assign\_Values stage to map the newly activated keys in the hashed files to the source columns.

**Image: Mapping keys in the Trans\_Assign\_Values stage**

This example illustrates the fields and controls on the Mapping keys in the Trans\_Assign\_Values stage. You can find definitions for the fields and controls later on this page.



- Click OK to continue, then save your changes.

## Configuring the Related ETL Job J\_Fact\_PS\_F\_LEDGER\_LEDGER\_BUDG in DataStage

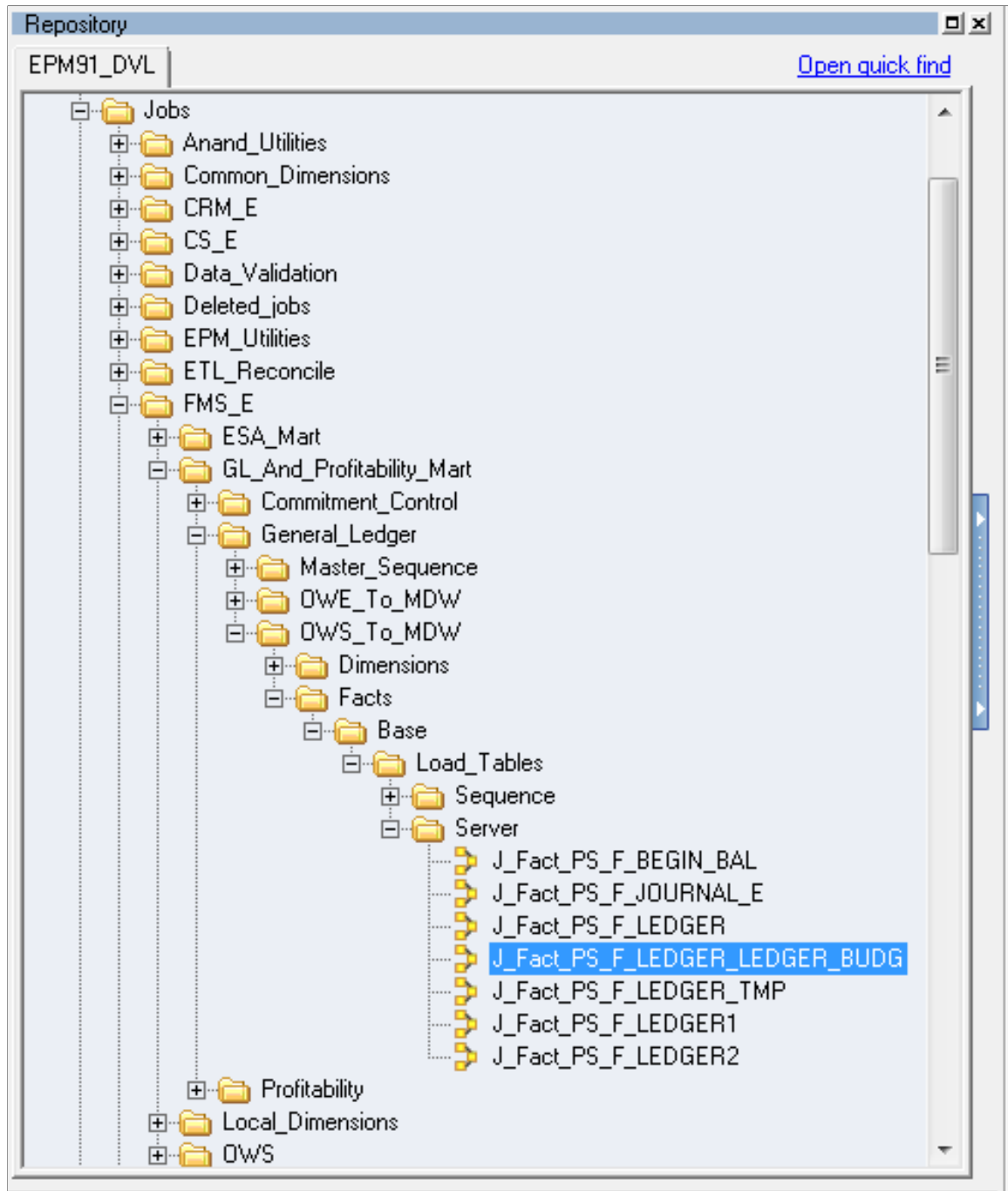
The job J\_Fact\_PS\_F\_LEDGER\_LEDGER\_BUDG also loads the PS\_F\_LEDGER table and must be configured to include new keys.

To configure the new keys in the J\_Fact\_PS\_F\_LEDGER\_LEDGER\_BUDG job:

1. In DataStage Designer client, navigate to the J\_Fact\_PS\_F\_LEDGER\_LEDGER\_BUDG job in the project tree and open it for editing.

**Image: J\_Fact\_PS\_F\_LEDGER\_LEDGER\_BUDG job in the project tree**

This example illustrates the fields and controls on the J\_Fact\_PS\_F\_LEDGER\_LEDGER\_BUDG job in the project tree. You can find definitions for the fields and controls later on this page.

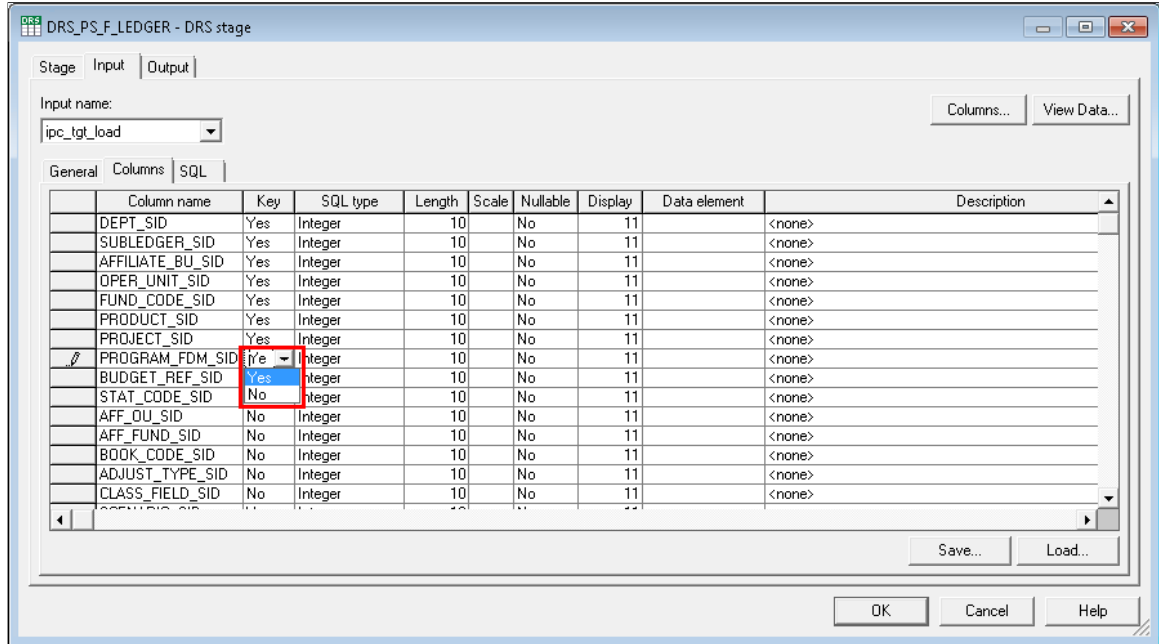


2. Open the DRS target stage (DRS\_PS\_F\_LEDGER) and select the Inputs tab.
3. Select the Columns sub-tab and locate each column you have activated as a key in Application Designer.

- For each column you have activated as a key in Application Designer, select *Yes* for the Key field.

**Image: Specifying keys in the DRS\_PS\_F\_LEDGER target stage**

This example illustrates the fields and controls on the Specifying keys in the DRS\_PS\_F\_LEDGER target stage. You can find definitions for the fields and controls later on this page.



**Note:** Ensure that you add all the same keys that you added in Application Designer.

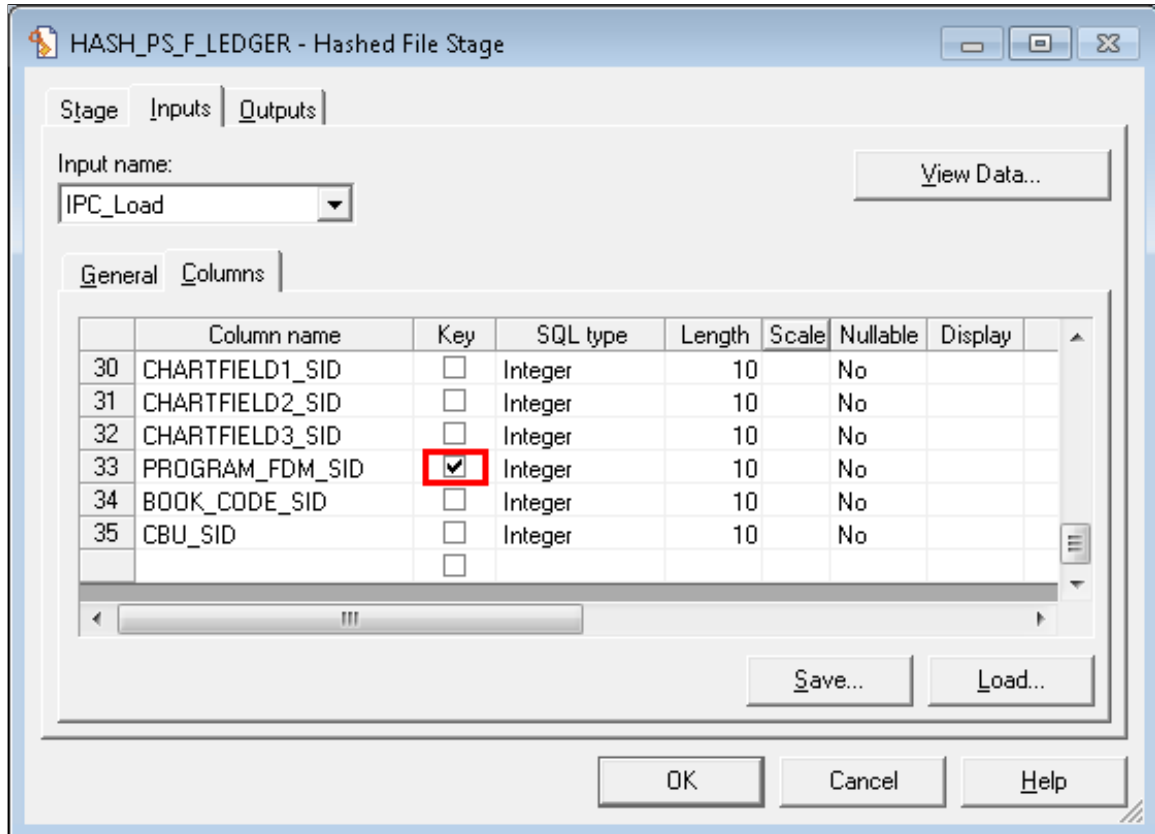
- Click OK to continue, then save your changes.
- Open the HASH\_PS\_F\_LEDGER hashed file to configure the new keys in the lookup.
- In the HASH\_PS\_F\_LEDGER hashed file, select the Inputs tab, then the Columns sub-tab.



8. Select the Key field check box for each column you have activated as a key in the target DRS stage.

**Image: Specifying keys in the HASH\_PS\_F\_LEDGER hashed file stage**

This example illustrates the fields and controls on the Specifying keys in the HASH\_PS\_F\_LEDGER hashed file stage. You can find definitions for the fields and controls later on this page.

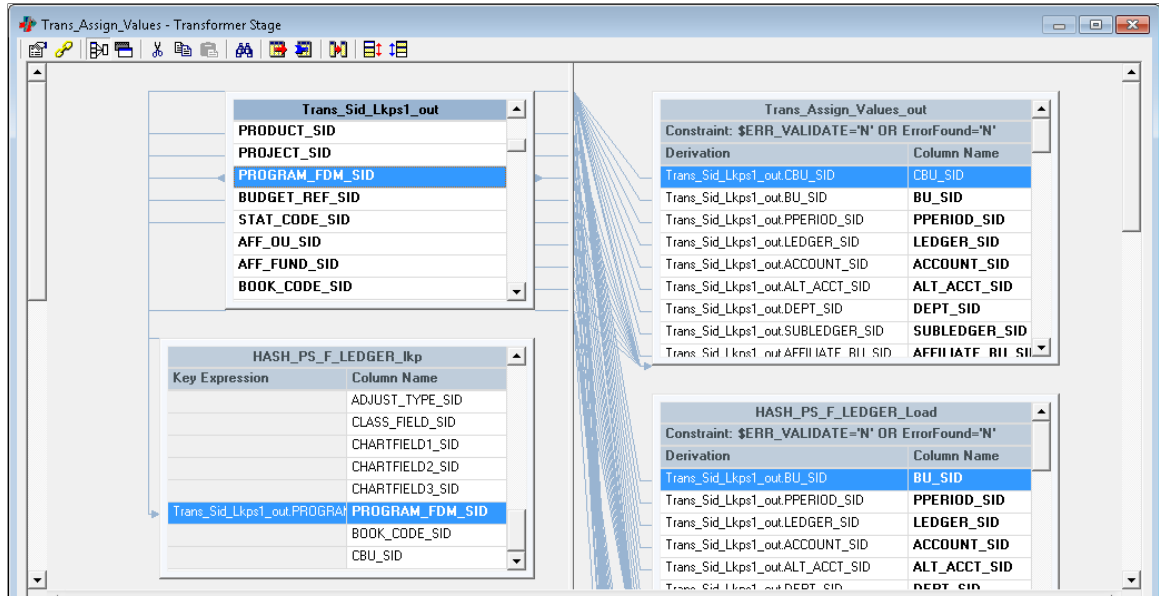


9. Click OK to continue, then save your changes.
10. Repeat steps 8 through 10 for the HASH\_PS\_F\_LEDGER1 hashed file.

- Open the Trans\_Assign\_Values stage to map the newly activated keys in the hashed files to the source columns.

**Image: Mapping keys in the Trans\_Assign\_Values stage**

This example illustrates the fields and controls on the Mapping keys in the Trans\_Assign\_Values stage. You can find definitions for the fields and controls later on this page.



- Click OK to continue, then save your changes.

## Activating Inactive Dimension Keys in the PS\_F\_BEGIN\_BAL Table

The PS\_F\_BEGIN\_BAL fact table is delivered with the following active dimension keys:

<i>Active Dimension Keys</i>	<i>Active Dimension Keys</i>
BU_SID	OPER_UNIT_SID
PYEAR_SID	FUND_CODE_SID
LEDGER_SID	PRODUCT_SID
ACCOUNT_SID	PROJECT_SID
ALT_ACCT_SID	BUDGET_REF_SID
DEPT_SID	STAT_CODE_SID
SUBLEDGER_SID	SCENARIO_SID
AFFILIATE_BU_SID	CURRENCY_CD

The following inactive columns can be activated as additional dimension keys:

<i>Inactive Dimension Key Columns</i>	<i>Inactive Dimension Key Columns</i>
PROGRAM_FDM_SID	ADJUST_TYPE_SID
CBU_SID	CLASS_FIELD_SID
AFF_OU_SID	CHARTFIELD1_SID
AFF_FUND_SID	CHARTFIELD2_SID
BOOK_CODE_SID	CHARTFIELD3_SID

To activate inactive dimension keys, you will use the PeopleSoft Application Designer and IBM WebSphere DataStage tools.

## Configuring PS\_F\_BEGIN\_BAL in Application Designer

To activate additional dimension keys in the PS\_F\_BEGIN\_BAL table:

1. Open PeopleSoft Application Designer.
2. Select File, Open from the menu.
3. In the Open Definition dialog box, select *Record* for the Definition field, then enter *F\_BEGIN\_BAL*.  
The Record Fields tab for the PS\_F\_BEGIN\_BAL table appears. Note the inactive dimension keys.
4. In the Record Fields tab, right-click an inactive dimension key and select Record Field Properties.  
The Record Field Properties window appears for the column.
5. In the Use tab of the Record Field Properties window select the Key and Search Key check boxes.
6. Click OK to continue.
7. Save your changes.
8. Select Build, Project from the menu.  
The Build window appears.
9. In the Build window select the Create Indexes and Execute and Build Script check boxes.
10. Select Settings.  
The Build Settings window appears.
11. In the Create tab of the Build Settings window, select the Recreate Index only if Modified check box.
12. Click OK to continue.
13. Click Build to complete your configuration of the PS\_F\_BEGIN\_BAL table.

14. Repeat steps 4 through 13 to activate other inactive dimension keys in the PS\_F\_BEGIN\_BAL table.

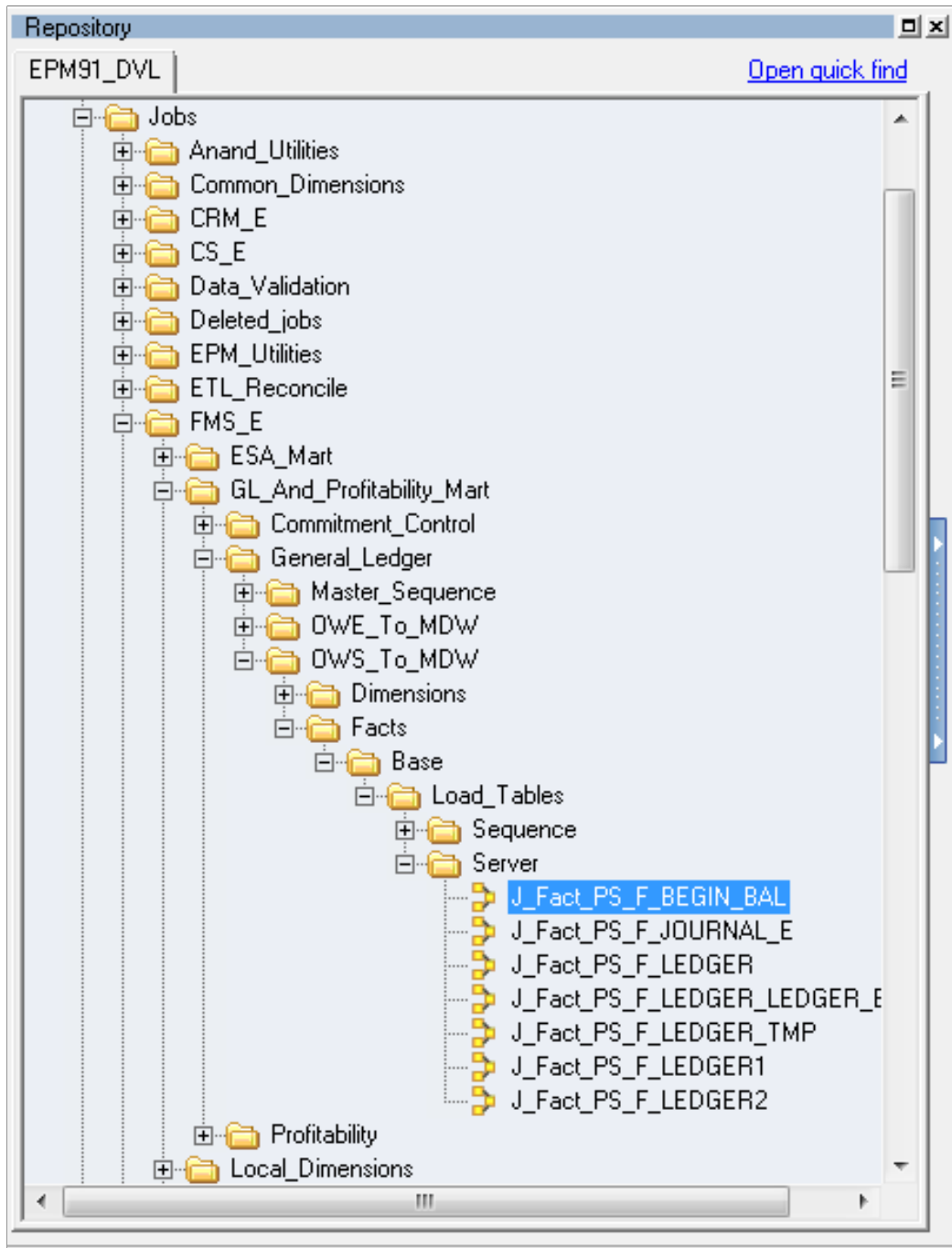
## **Configuring the J\_Fact\_PS\_F\_BEGIN\_BAL Job in DataStage**

To configure the new keys in the J\_Fact\_PS\_F\_BEGIN\_BAL job:

1. In DataStage Designer client, navigate to the J\_Fact\_PS\_F\_BEGIN\_BAL job in the project tree and open it for editing.

**Image: J\_Fact\_PS\_F\_BEGIN\_BAL job in the project tree**

This example illustrates the fields and controls on the J\_Fact\_PS\_F\_BEGIN\_BAL job in the project tree. You can find definitions for the fields and controls later on this page.

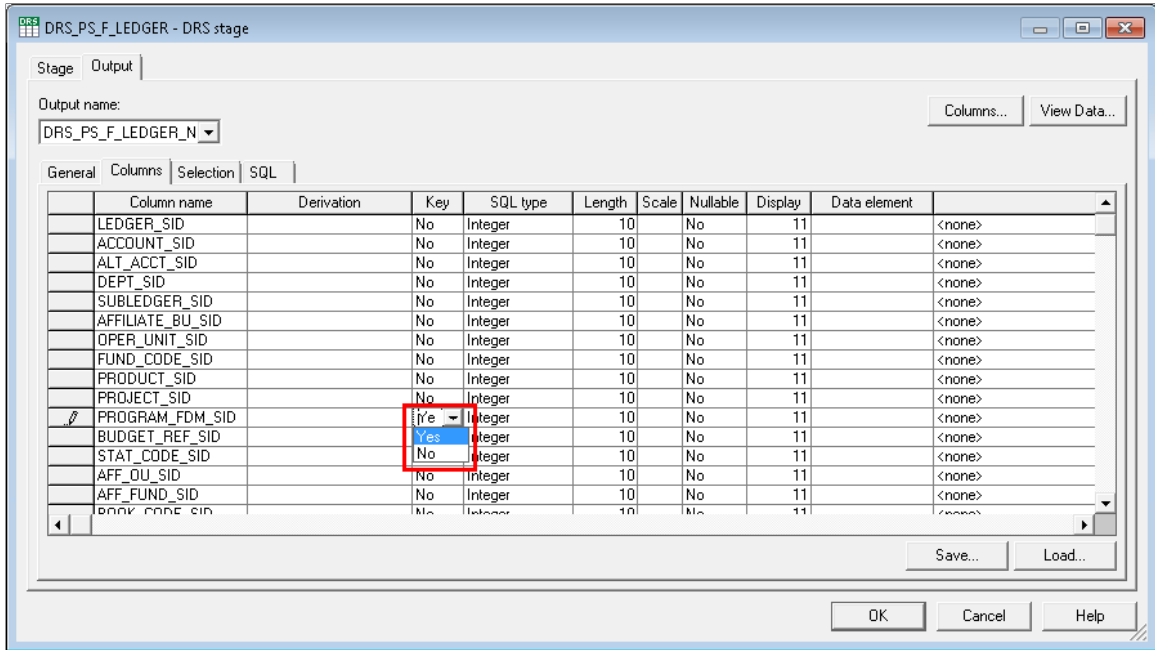


2. Open the source DRS stage (DRS\_PS\_F\_LEDGER) and select the Outputs tab.

3. Select the Columns sub-tab and select *Yes* for the Key field for each column you have activated as a key.

**Image: Specifying keys in the DRS\_PS\_F\_LEDGER target stage**

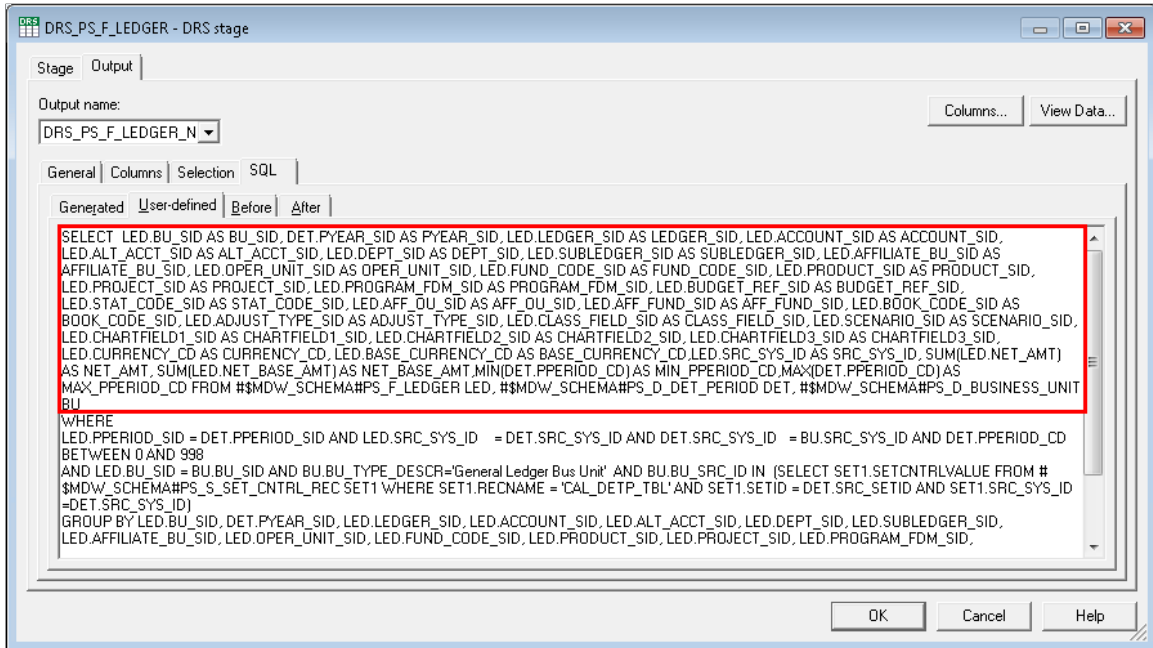
This example illustrates the fields and controls on the Specifying keys in the DRS\_PS\_F\_LEDGER target stage. You can find definitions for the fields and controls later on this page.



4. Select the SQL and User-Defined sub-tabs and modify the user defined query to add any newly activated dimension keys.

**Image: Modifying the SQL query in the DRS\_PS\_F\_LEDGER source stage**

This example illustrates the fields and controls on the Modifying the SQL query in the DRS\_PS\_F\_LEDGER source stage. You can find definitions for the fields and controls later on this page.

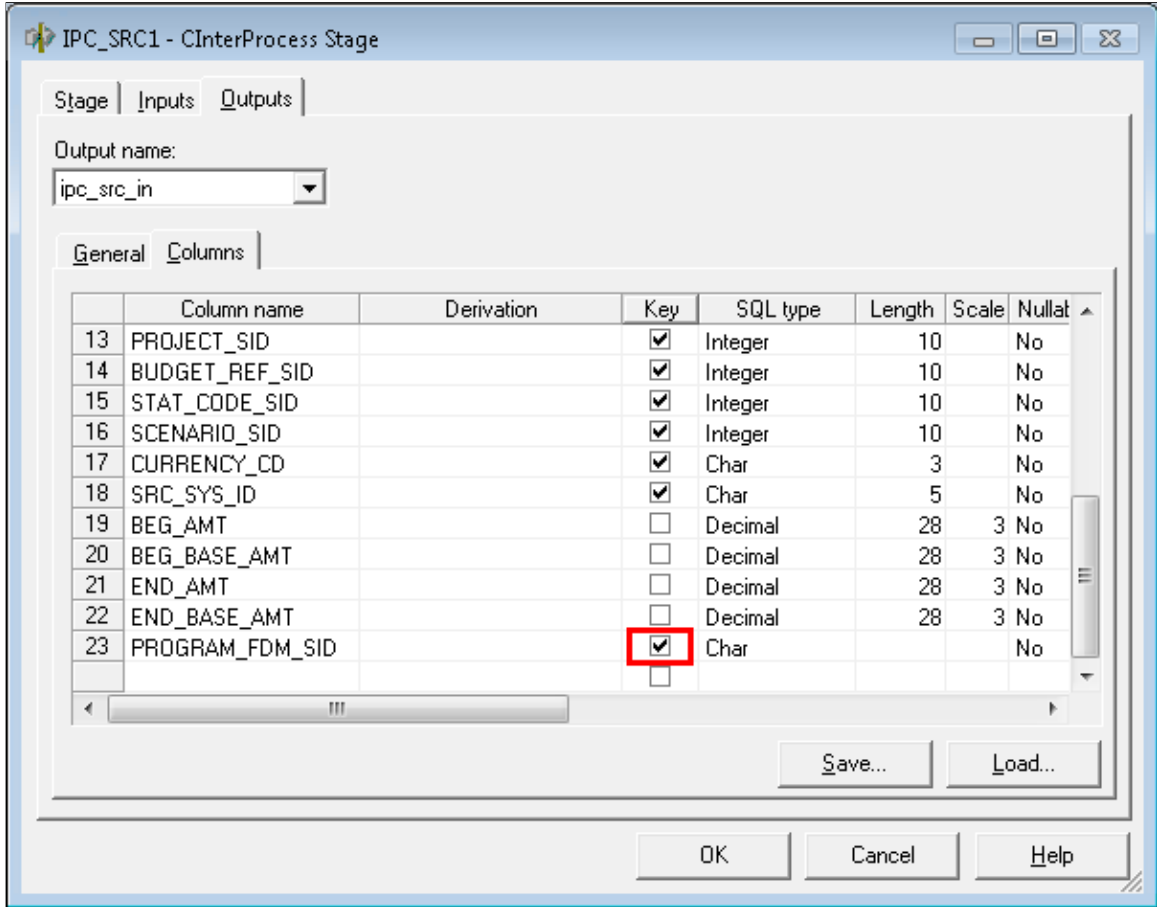


5. Open the IPC stage (IPC\_SRC1) and select the Outputs tab.
6. Select the Columns sub-tab and locate each column you have activated as a key in Application Designer.

7. Select the Key field check box for each column you have activated as a key.

**Image: Specifying keys in the IPC\_SRC1 stage**

This example illustrates the fields and controls on the Specifying keys in the IPC\_SRC1 stage. You can find definitions for the fields and controls later on this page.



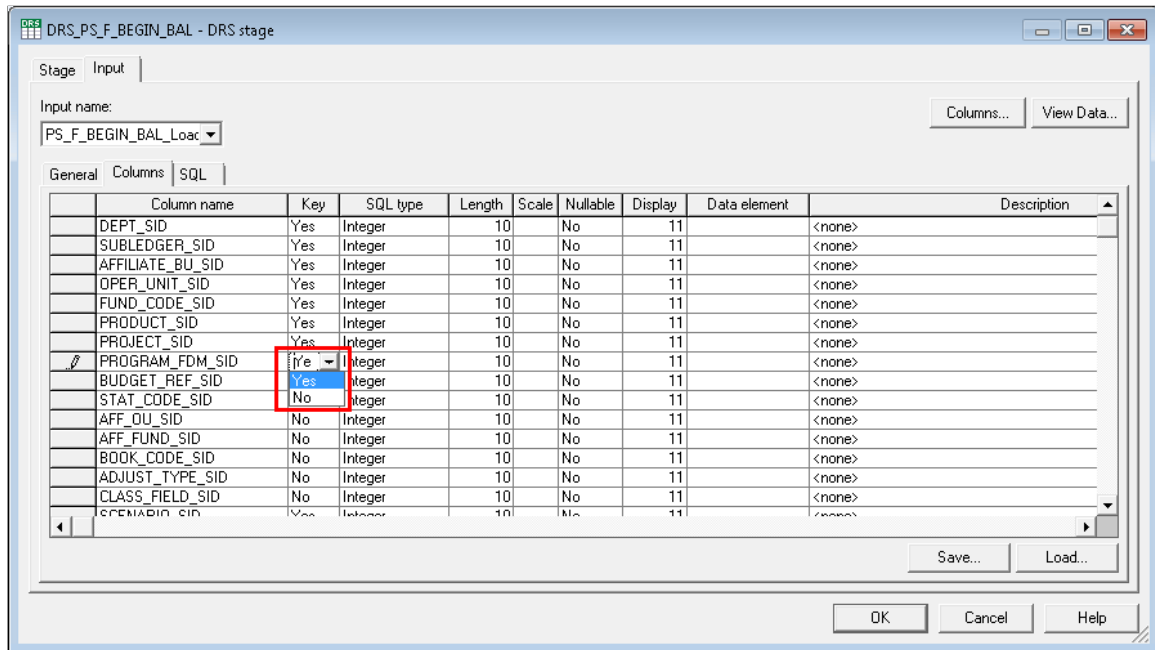
8. Open the DRS target stage (DRS\_PS\_F\_BEGIN\_BAL) and select the Inputs tab.



9. Select the Columns sub-tab and select *Yes* for the Key field for each column you have activated as a key.

**Image: Specifying keys in the DRS\_PS\_F\_BEGIN\_BAL target stage**

This example illustrates the fields and controls on the Specifying keys in the DRS\_PS\_F\_BEGIN\_BAL target stage. You can find definitions for the fields and controls later on this page.



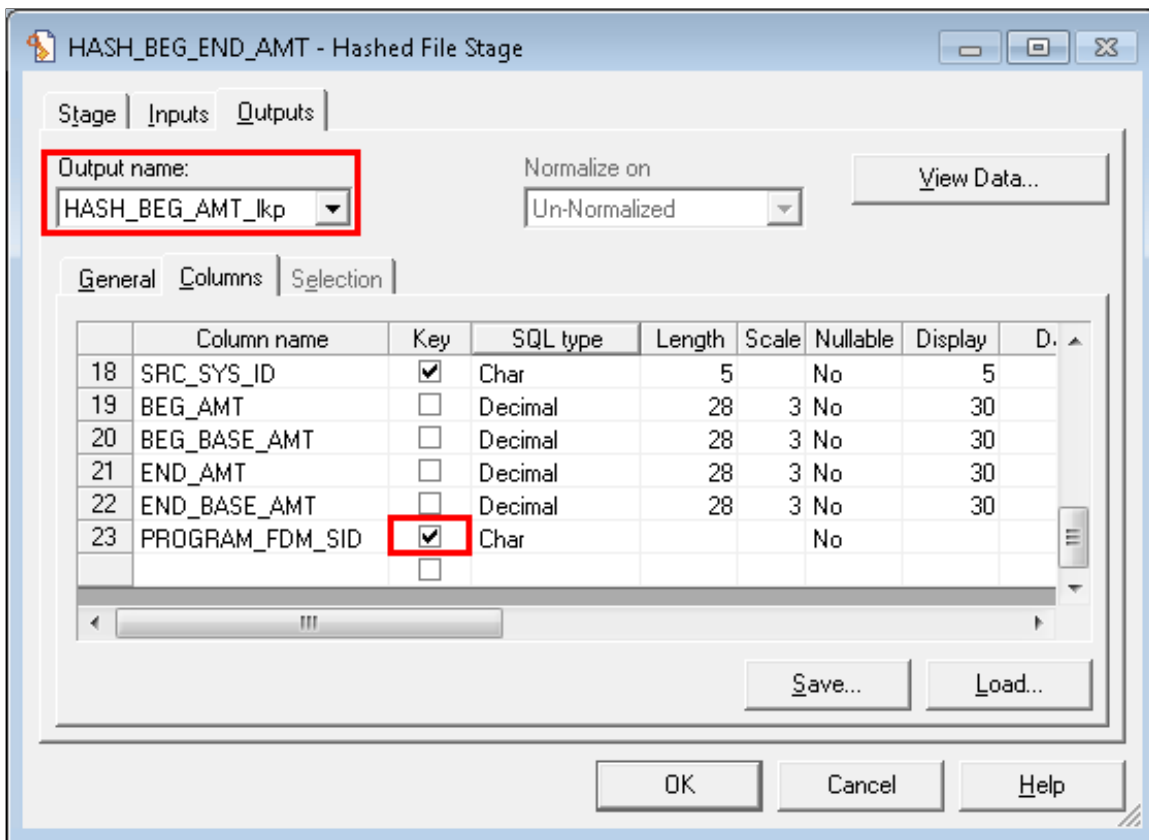
**Note:** Ensure that you add all the same keys that you added in Application Designer.

10. Select the SQL sub-tab and add the columns to the existing SQL statement.
11. Click OK to continue, then save your changes.
12. Open the HASH\_BEG\_END\_AMT hashed file to configure the new keys in the lookup.
13. In the HASH\_BEG\_END\_AMT hashed file, select the Outputs then the Columns sub-tab.
14. Select *HASH\_BEG\_AMT\_lkp* for the Output Name field.

15. Select the Key field check box for each column you have activated as a key.

**Image: Specifying keys in the HASH\_BEG\_END\_AMT hashed file**

This example illustrates the fields and controls on the Specifying keys in the HASH\_BEG\_END\_AMT hashed file. You can find definitions for the fields and controls later on this page.

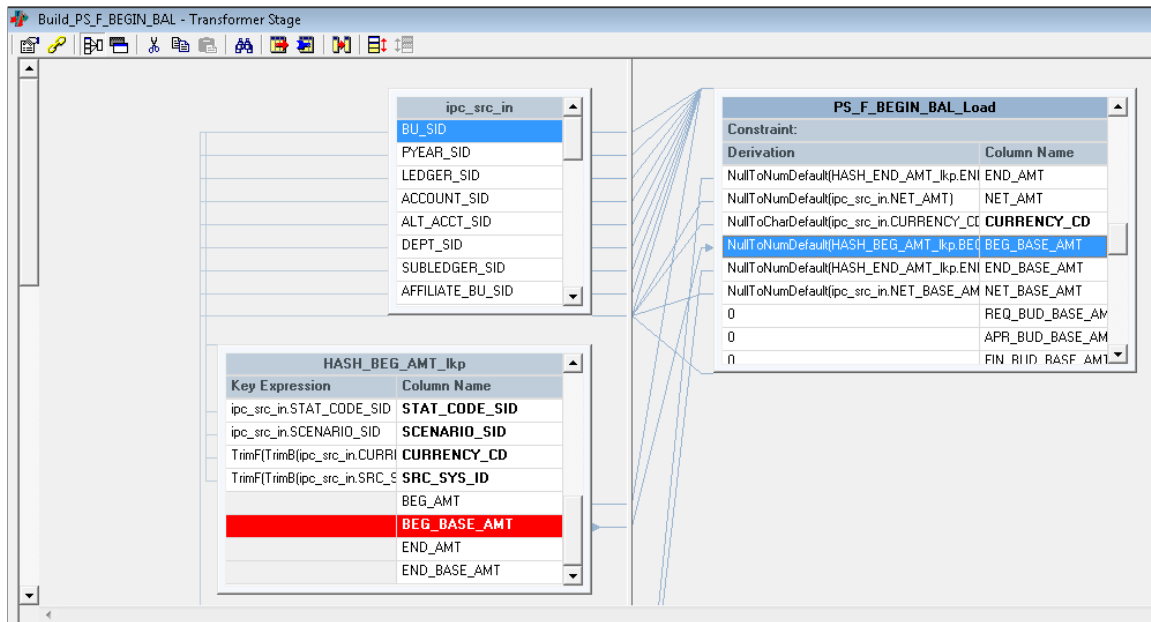


16. Select *HASH\_END\_AMT\_lkp* for the Output Name field and repeat step 12.
17. Click OK to continue, then save your changes.

- Open the Build\_PS\_F\_BEGIN\_BAL stage to map the newly activated keys in the hashed files to the source columns.

**Image: Mapping keys in the Build\_PS\_F\_BEGIN\_BAL stage**

This example illustrates the fields and controls on the Mapping keys in the Build\_PS\_F\_BEGIN\_BAL stage. You can find definitions for the fields and controls later on this page.



- Click OK to continue, then save your changes.

## Activating Inactive Dimension Keys in the PS\_F\_KK\_LEDGER Table

The PS\_F\_KK\_LEDGER fact table is delivered with the following active dimension keys:

<i>Active Dimension Keys</i>	<i>Active Dimension Keys</i>
BU_SID	ACCOUNT_SID
KK_BUDGET_SID	OPER_UNIT_SID
LEDGER_SID	FUND_CODE_SID
BPERIOD_SID	DEPT_SID
CURRENCY_CD	PROGRAM_FDM_SID
STAT_CODE_SID	CLASS_FIELD_SID
PPERIOD_SID	BUDGET_REF_SID
KK_TRANS_TYPE_SID	PRODUCT_SID

The following inactive columns can be activated as additional dimension keys:

<i>Inactive Dimension Key Columns</i>	<i>Inactive Dimension Key Columns</i>
CHARTFIELD1_SID	AFF_OU_SID
CHARTFIELD2_SID	PRJ_BU_SID,PRJ_SID
CHARTFIELD3_SID	AC_SID
AFFILIATE_BU_SID	PROJ_RSRC_TYPE_SID
AFF_FUND_SID	BU_LED_SID

To activate inactive dimension keys, you will use the PeopleSoft Application Designer and IBM WebSphere DataStage tools.

### Configuring PS\_F\_KK\_LEDGER in Application Designer

To activate additional dimension keys in the PS\_F\_KK\_LEDGER table:

1. Open PeopleSoft Application Designer.
2. Select File, Open from the menu.
3. In the Open Definition dialog box, select *Record* for the Definition field, then enter *F\_KK\_LEDGER*.

The Record Fields tab for the PS\_F\_KK\_LEDGER table appears. Note the inactive dimension keys.

4. In the Record Fields tab, right-click an inactive dimension key and select *Record Field Properties*.

The Record Field Properties window appears for the column.

5. In the Use tab of the Record Field Properties window select the Key and Search Key check boxes.

---

**Note:** For DB2 or OS390 databases, deactivate any unused dimension keys to maintain the 255 character index limit.

---

6. Click OK to continue.
7. Save your changes.
8. Select Build, Project from the menu.  
The Build window appears.
9. In the Build window select the Create Indexes and Execute and Build Script check boxes.
10. Select Settings.  
The Build Settings window appears.
11. In the Create tab of the Build Settings window, select the Recreate Index only if Modified check box.
12. Click OK to continue.

13. Click Build to complete your configuration of the PS\_F\_KK\_LEDGER table.
14. Repeat steps 4 through 13 to activate other inactive dimension keys in the PS\_F\_KK\_LEDGER table.

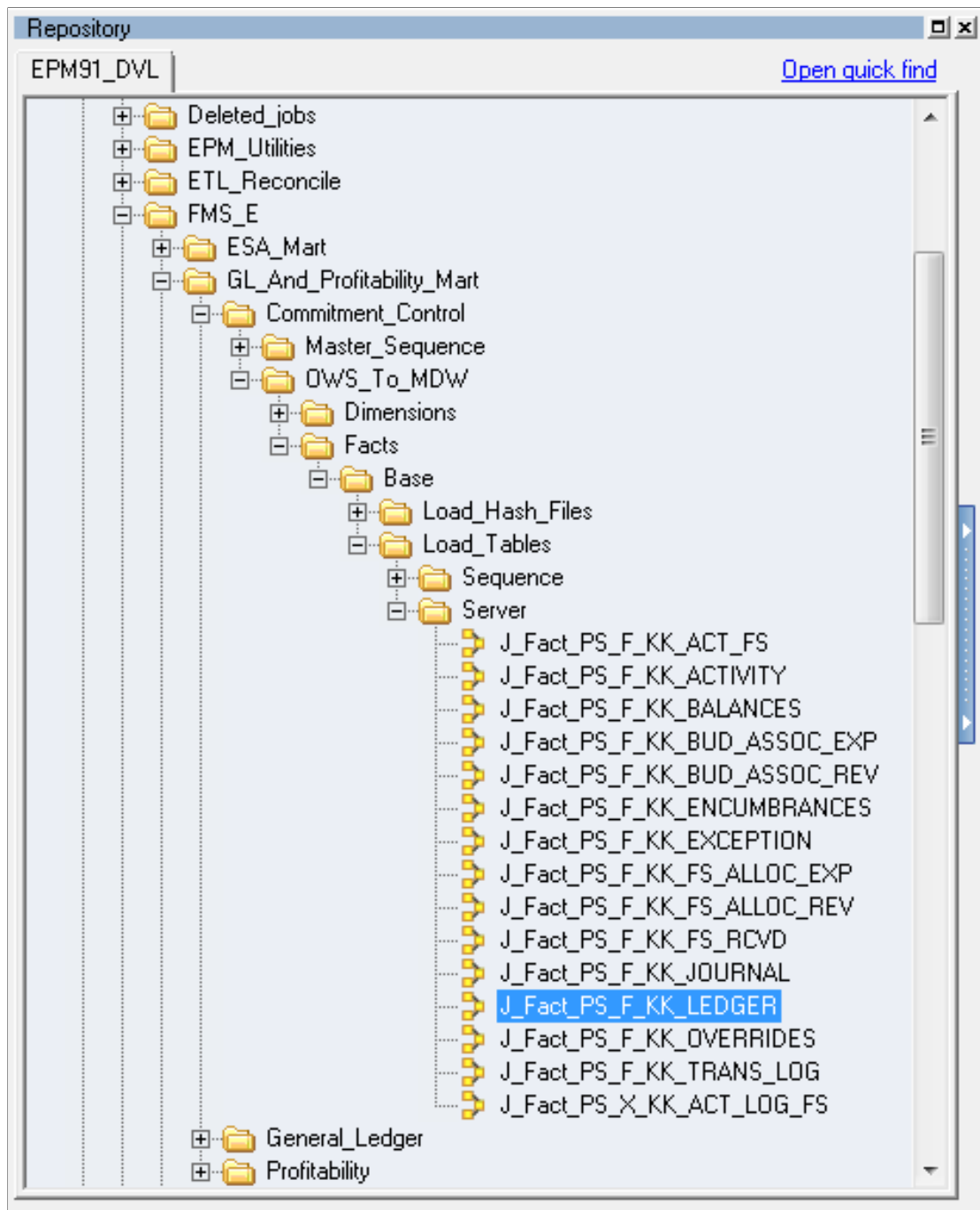
## **Configuring the J\_Fact\_PS\_F\_KK\_LEDGER ETL Job in DataStage**

To configure the new keys in the J\_Fact\_PS\_F\_KK\_LEDGER job:

1. In DataStage Designer client, navigate to the J\_Fact\_PS\_F\_KK\_LEDGER job in the project tree and open it for editing.

**Image: J\_Fact\_PS\_F\_KK\_LEDGER job in the project tree**

This example illustrates the fields and controls on the J\_Fact\_PS\_F\_KK\_LEDGER job in the project tree. You can find definitions for the fields and controls later on this page.

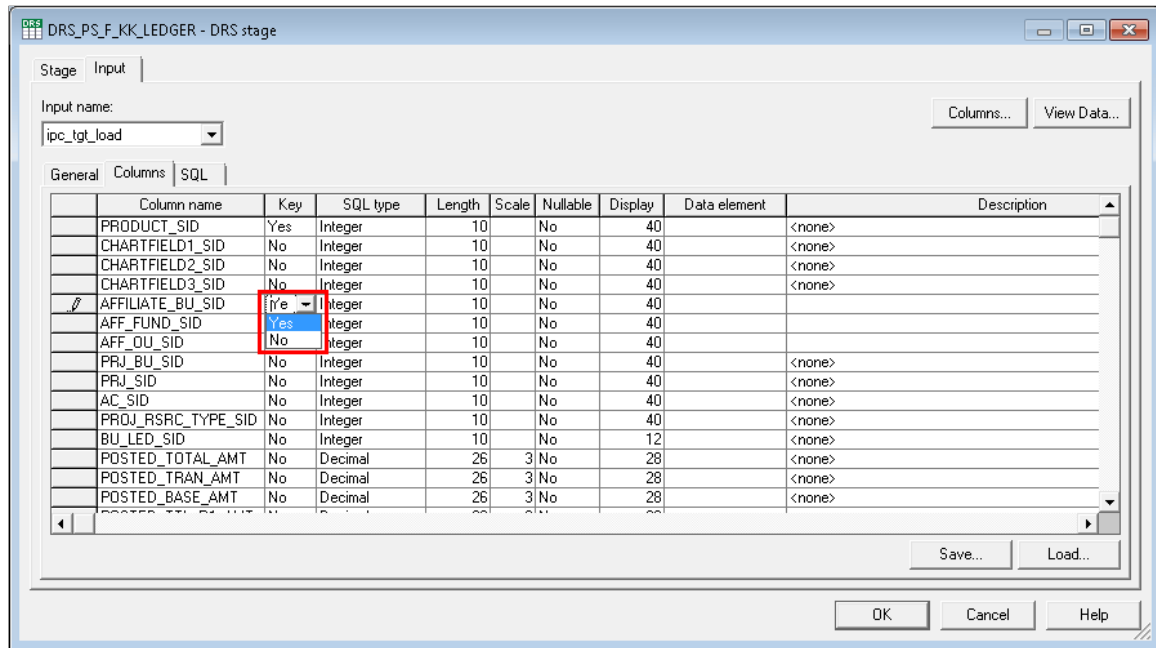


2. Open the DRS target stages (DRS\_PS\_F\_KK\_LEDGER and DRS\_PS\_F\_KK\_LEDGER\_UPD) and select the Inputs tab.

3. Select the Columns sub-tab and select *Yes* for the Key field for each column you have activated as a key.

**Image: Specifying keys in the DRS\_PS\_F\_KK\_LEDGER target stage**

This example illustrates the fields and controls on the Specifying keys in the DRS\_PS\_F\_KK\_LEDGER target stage. You can find definitions for the fields and controls later on this page.



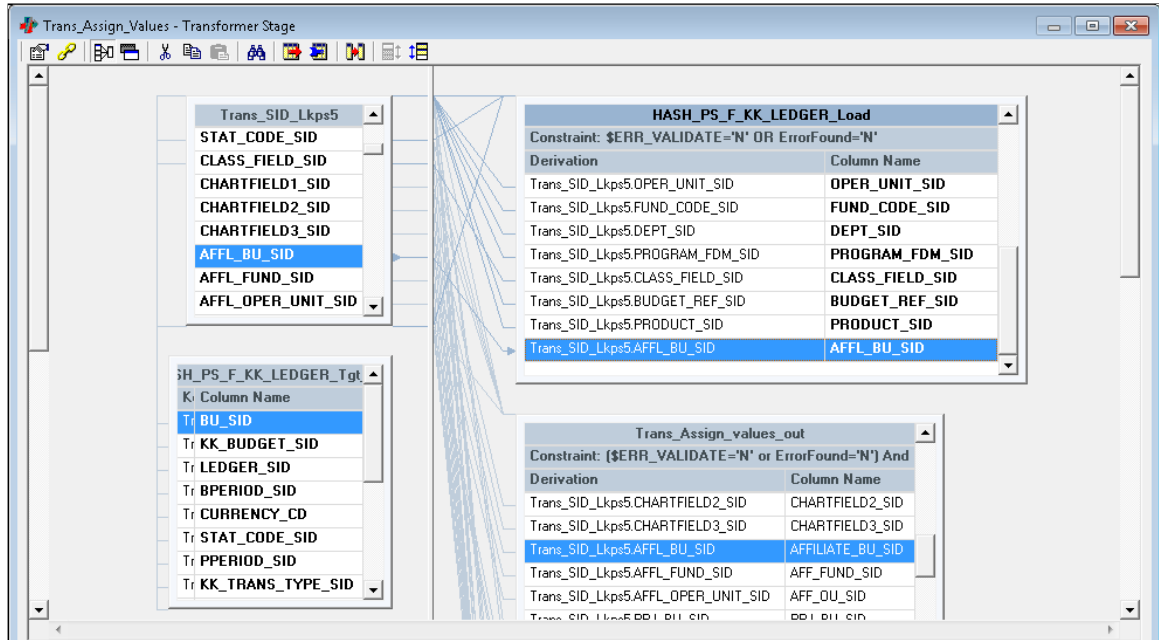
**Note:** Ensure that you add all the same keys that you added in Application Designer.

4. Click OK to continue, then save your changes.
5. Open the HASH\_F\_KK\_LEDGER hashed file to configure the new keys in the lookup.
6. In the HASH\_F\_KK\_LEDGER hashed file, select the Outputs tab, then the Columns sub-tab.
7. Select the Key field check box for each column you have activated as a key.
8. Click OK to continue.
9. Open the HASH\_F\_KK\_LEDGER1 hashed file, select the Inputs tab, then Columns sub-tab.
10. Select the Key field check box for each column you have activated as a key.
11. Click OK to continue, then save your changes.

- Open the Trans\_Assign\_Values stage to map the newly activated keys in the hashed files to the source columns.

**Image: Mapping keys in the Trans\_Assign\_Value stage**

This example illustrates the fields and controls on the Mapping keys in the Trans\_Assign\_Value stage. You can find definitions for the fields and controls later on this page.



- Click OK to continue, then save your changes.

## Activating Inactive Dimension Keys in the PS\_F\_KK\_BALANCES Table

The PS\_F\_KK\_BALANCES fact table is delivered with the following active dimension keys:

Active Dimension Keys	Active Dimension Keys
BU_SID	FUND_CODE_SID
KK_BUDGET_SID	DEPT_SID
BPERIOD_SID	PROGRAM_FDM_SID
CURRENCY_CD	CLASS_FIELD_SID
STAT_CODE_SID	BUDGET_REF_SID
PPERIOD_SID	PRODUCT_SID
ACCOUNT_SID	



<b>Active Dimension Keys</b>	<b>Active Dimension Keys</b>
OPER_UNIT_SID	

The following inactive columns can be activated as additional dimension keys:

<b>Inactive Dimension Key Columns</b>	<b>Inactive Dimension Key Columns</b>
CHARTFIELD1_SID	PRJ_BU_SID
CHARTFIELD2_SID	PRJ_SID
CHARTFIELD3_SID	AC_SID
AFFILIATE_BU_SID	PROJ_RSRC_TYPE_SID
AFF_FUND_SID	
AFF_OU_SID	

To activate inactive dimension keys, you will use the PeopleSoft Application Designer and IBM WebSphere DataStage tools.

## Configuring PS\_F\_KK\_BALANCES in Application Designer

To activate additional dimension keys in the PS\_F\_KK\_BALANCES table:

1. Open PeopleSoft Application Designer.
2. Select File, Open from the menu.
3. In the Open Definition dialog box, select *Record* for the Definition field, then enter *F\_KK\_BALANCES*.

The Record Fields tab for the PS\_F\_KK\_BALANCES table appears. Note the inactive dimension keys.

4. In the Record Fields tab, right-click an inactive dimension key and select *Record Field Properties*.

The Record Field Properties window appears for the column.

5. In the Use tab of the Record Field Properties window select the Key and Search Key check boxes.

---

**Note:** For DB2 or OS390 databases, deactivate any unused dimension keys to maintain the 255 character index limit.

---

6. Click OK to continue.
7. Save your changes.
8. Select Build, Project from the menu.

The Build window appears.

9. In the Build window select the Create Indexes and Execute and Build Script check boxes.
10. Select Settings.  
The Build Settings window appears.
11. In the Create tab of the Build Settings window, select the Recreate Index only if Modified check box.
12. Click OK to continue.
13. Click Build to complete your configuration of the PS\_F\_KK\_BALANCES table.
14. Repeat steps 4 through 13 to activate other inactive dimension keys in the PS\_F\_KK\_BALANCES table.

## **Configuring the J\_Fact\_PS\_F\_KK\_BALANCES ETL Job in DataStage**

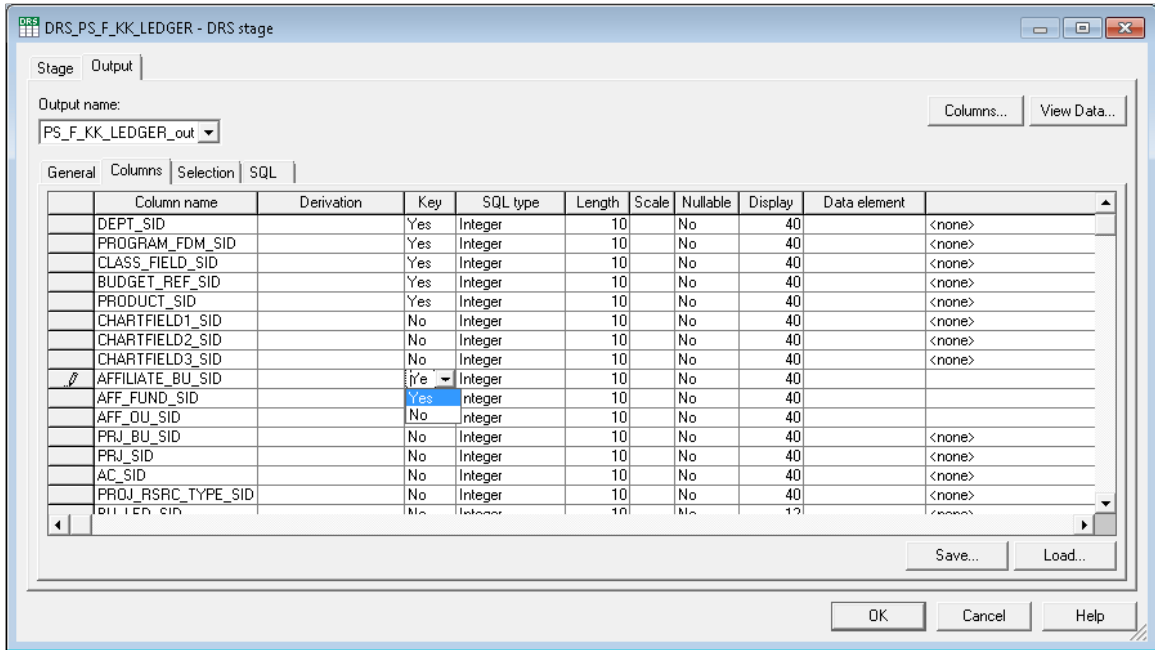
To configure the new keys in the J\_Fact\_PS\_F\_KK\_BALANCES job:



3. Select the Columns sub-tab and select *Yes* for the Key field for each column you have activated as a key.

**Image: Specifying keys in the DRS\_PS\_F\_KK\_LEDGER source stage**

This example illustrates the fields and controls on the Specifying keys in the DRS\_PS\_F\_KK\_LEDGER source stage. You can find definitions for the fields and controls later on this page.

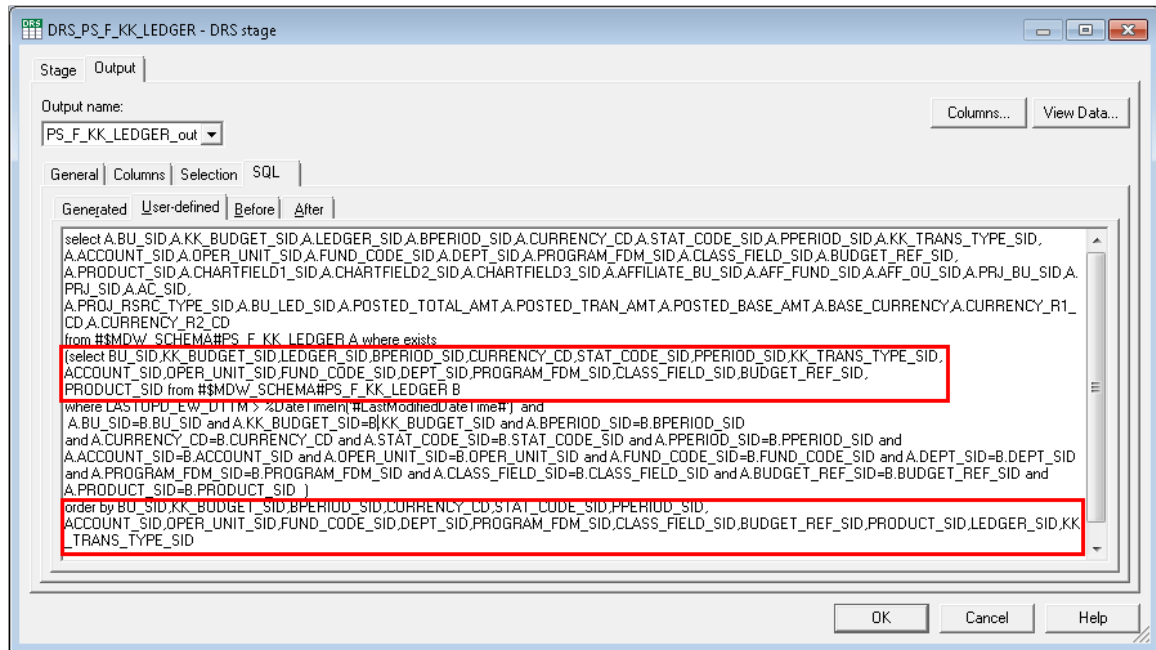


**Note:** Ensure that you add all the same keys that you added in Application Designer.

4. Select the SQL and User-Defined sub-tabs and modify the user defined query to use any newly activated dimension keys in the PS\_F\_KK\_LEDGER table.

**Image: Modifying the SQL query in the DRS\_PS\_F\_KK\_LEDGER source stage**

This example illustrates the fields and controls on the Modifying the SQL query in the DRS\_PS\_F\_KK\_LEDGER source stage. You can find definitions for the fields and controls later on this page.



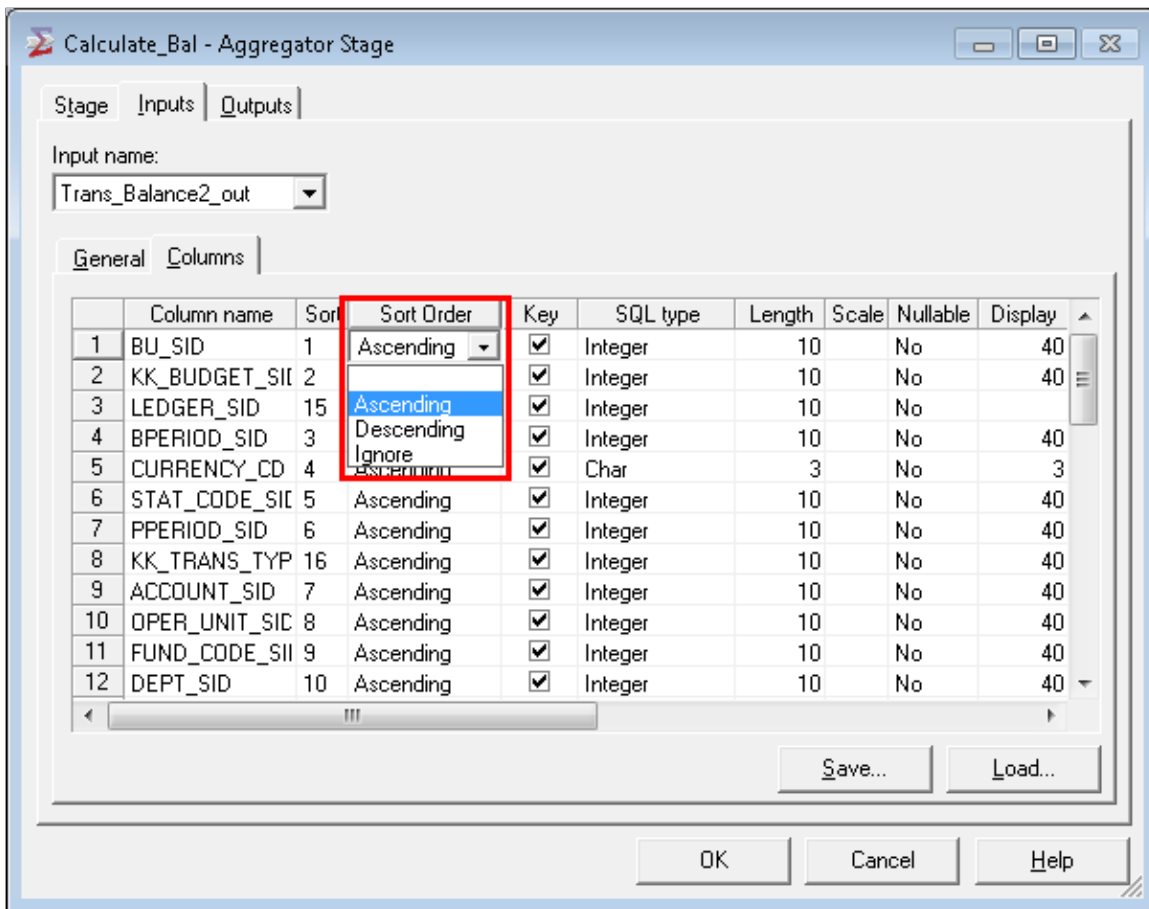
**Note:** Do not use LEDGER\_SID and KK\_TRANS\_TYPE\_SID in the join criteria.

5. Click OK to continue, then save your changes.
6. Open the DRS target stage DRS\_PS\_F\_KK\_BALANCES and select the Inputs tab.
7. Select the Columns sub-tab and select *Yes* for the Key field for each column you have activated as a key.
8. Click OK to continue, then save your changes.
9. Open the Aggregator stage Calculate\_Bal to define sort order and grouping parameters.
10. Select the Inputs tab, then Columns sub-tab.

- Use the Sort Order field to select *Ascending*, *Descending*, or *Ignore* for each key column; the value you choose should match the order defined in the SQL ORDER BY clause in the source DRS stage (DRS\_PS\_F\_KK\_LEDGER).

**Image: Specifying sort order in the Calculate\_Bal Aggregator stage**

This example illustrates the fields and controls on the Specifying sort order in the Calculate\_Bal Aggregator stage. You can find definitions for the fields and controls later on this page.

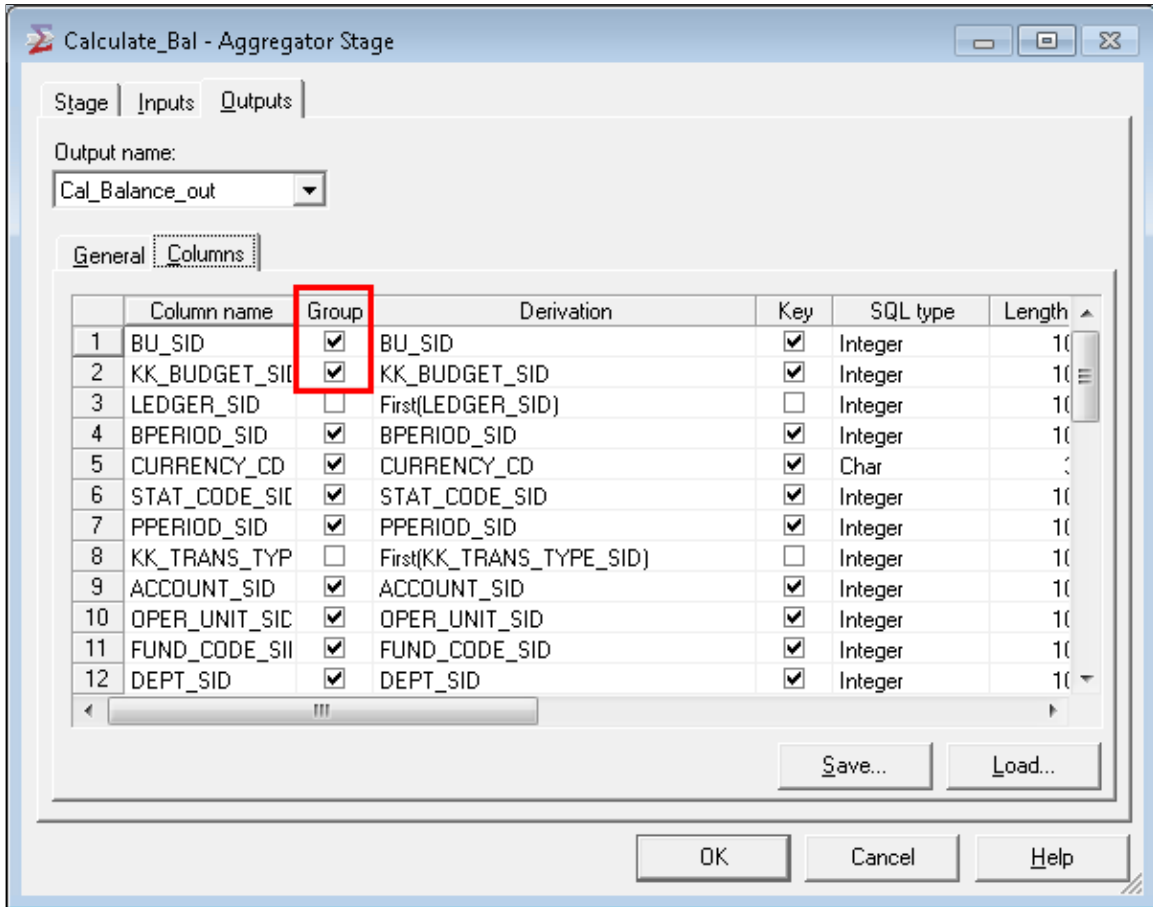


- Select the Outputs tab, then Columns sub-tab.

13. Select the Group check box for each key column you want grouped.

**Image: Specifying key column grouping in the Calculate\_Bal Aggregator stage**

This example illustrates the fields and controls on the Specifying key column grouping in the Calculate\_Bal Aggregator stage. You can find definitions for the fields and controls later on this page.



14. Click OK to continue, then save your changes.





# Implementing Materialized View Logs

---

## Understanding Materialized View Logs

During the source to OWS ETL load process, some jobs perform a cyclical redundancy check (CRC) to determine new or changed records. Unlike a traditional lookup process which targets the DTTM column for each record, the CRC process reads the entire record for every 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.

To solve this problem, PeopleSoft provides Materialized View Log functionality for Campus Solutions, FMS, and SCM Warehouse customers using an Oracle database. The new feature will dramatically reduce the processing time for OWS jobs that use the CRC process.

Additionally, the new Materialized View Log functionality includes logic that identifies source record deletes so that those records can later be deleted from EPM fact tables.

In an Oracle database, a materialized view log is a table associated with the master table of a materialized view. When master table data undergoes DML changes (such as INSERT, UPDATE, or DELETE), the Oracle database stores rows describing those changes in the materialized view log. A materialized view log is similar to an AUDIT table and is the mechanism used to capture changes made to its related master table. Rows are automatically added to the Materialized View Log table when the master table changes. The Oracle database uses the materialized view log to refresh materialized views based on the master table. This process is called fast refresh and improves performance in the source database.

A materialized view log can capture the primary keys, row IDs, or object identifiers of rows that have been updated in the master table. The standard naming convention for a materialized view log table is: MLOG\$\_<master\_name>.

### Exceptions

Materialized View Logs do not capture rows that are inserted in bulk to the master table through the Append Hint function. For example:

```
INSERT /*+ append */ INTO PS_STDNT_BUDGET_IT SELECT * FROM  
PS_STDNT_BUDGET_IT_TEMP WHERE EMPLID='FA0002';
```

Also, Materialized View Logs do not capture rows that are truncated in the master table. For example:

```
Truncate table PS_STDNT_BUDGET_IT;
```

---

**Note:** Only customers using an Oracle database can use Materialized View Logs.

---

## Implementing Materialized View Logs in the Oracle Database

To implement Materialized View Logs in the Oracle database, first determine if you have created customized (non PeopleSoft delivered) Materialized View Logs for a base table by entering the following SQL statement against the Oracle database:

```
SELECT * FROM USER_SNAPSHOT_LOGS WHERE MASTER='<BASE TABLE>';
```

If a customized view log exists, you must configure it to support PeopleSoft requirements. You must also create a DUMMY materialized view to prevent the Materialized View Log data from being purged.

**Note:** If the existing view log is one of the PeopleSoft delivered view logs, you do not need to configure it. See the following section for a list of delivered Materialized View Logs for PeopleSoft Financial Management source tables.

### Delivered Materialized View Logs for PeopleSoft Financial Management Source Tables

The following table provides a list of delivered Materialized View Logs and their corresponding source tables for the FMS Warehouse. The table also provides the EPM fact tables related to a Materialized View Log; these fact tables track deleted source records with the help of the Materialized View Log.

<i>Financial Management Source Table</i>	<i>Delivered Materialized View Log</i>	<i>Corresponding EPM Fact Table</i>
PS_AP_MTCH_EXCPTN	MLOG\$_PS_AP_MTCH_EXCPTN	PS_F_VCHR_MTCH_EXP
PS_CA_ACCTG_LINE	MLOG\$_PS_CA_ACCTG_LINE	PS_F_CA_REV_RECOGN
PS_CA_ACCTG_LN_PC	MLOG\$_PS_CA_ACCTG_LN_PC	PS_F_CA_REV_RECOGN
PS_CA_CONTR_HDR	MLOG\$_PS_CA_CONTR_HDR	PS_F_CA_AMD PS_F_CA_DTL_TRANS PS_F_CA_REV_RECOGN PS_F_CA_DTL_DIST PS_F_CA_RNW
PS_CA_DETAIL	MLOG\$_PS_CA_DETAIL	PS_F_CA_DTL_TRANS PS_F_CA_REV_RECOGN PS_F_CA_DTL_DIST PS_F_CA_CUR_FRCST PS_F_CA_PRDC_FRCST
PS_CA_DETAIL_PROJ	MLOG\$_PS_CA_DETAIL_PROJ	PS_F_GM_AWARD PS_F_GM_PRJ_TRAN

<b>Financial Management Source Table</b>	<b>Delivered Materialized View Log</b>	<b>Corresponding EPM Fact Table</b>
PS_CUST_HIST_E_TBL	MLOG\$_PS_CUST_HIST_E_TBL	PS_F_AR_DSO_E
PS_DEPOSIT_CONTROL	MLOG\$_PS_DEPOSIT_CONTROL	PS_F_AR_TRAN
PS_GM_BU_AWD_SETUP	MLOG\$_PS_GM_BU_AWD_SETUP	PS_F_GM_AWARD
PS_ITEM	MLOG\$_PS_ITEM	PS_F_AR_TRAN PS_F_AR_ACCOUNT_LN
PS_ITEM_ACTIVITY	MLOG\$_PS_ITEM_ACTIVITY	PS_F_AR_TRAN
PS_ITEM_DST	MLOG\$_PS_ITEM_DST	PS_F_AR_ACCOUNT_LN
PS_KK_LIQUIDATION	MLOG\$_PS_KK_LIQUIDATION	PS_F_KK_ENCUMBRAN
PS_KK_REFERENCED	MLOG\$_PS_KK_REFERENCED	PS_F_KK_ENCUMBRAN
PS_KK_SOURCE_LN	MLOG\$_PS_KK_SOURCE_LN	PS_F_KK_TRANS_LOG
PS_KK_TRANS_LOG	MLOG\$_PS_KK_TRANS_LOG	PS_F_KK_TRANS_LOG
PS_PAY_MISC_DST	MLOG\$_PS_PAY_MISC_DST	PS_F_AR_ACCOUNT_LN
PS_PAYMENT	MLOG\$_PS_PAYMENT	PS_F_AR_TRAN
PS_PAYMENT_TBL	MLOG\$_PS_PAYMENT_TBL	PS_F_AR_TRAN
PS_PC_BUD_DETAIL	MLOG\$_PS_PC_BUD_DETAIL	PS_F_GM_AWARD
PS_PROJ_ACTIVITY	MLOG\$_PS_PROJ_ACTIVITY	PS_F_PRJ_AC_CUR PS_F_PRJ_AC_PRDC PS_F_PRJ_AC_CUR PS_F_PRJ_AC_PRDC
PS_PROJ_RESOURCE	MLOG\$_PS_PROJ_RESOURCE	PS_F_GM_PRJ_TRAN PS_F_PRJ_TRAN PS_F_RSRC_RT
PS_PROJECT	MLOG\$_PS_PROJECT	PS_F_PRJ_PRDC PS_F_PRJ_CUR PS_F_PRJ_PRDC PS_F_PRJ_CUR
PS_PYMNT_VCHR_XREF	MLOG\$_PS_PYMNT_VCHR_XREF	PS_F_AP_TRAN

<i>Financial Management Source Table</i>	<i>Delivered Materialized View Log</i>	<i>Corresponding EPM Fact Table</i>
PS_VOUCHER	MLOG\$_PS_VOUCHER	PS_F_AP_TRAN PS_F_VCHR_MTCH_EXP
PS_VOUCHER_LINE	MLOG\$_PS_VOUCHER_LINE	PS_F_MTCH_ANLYS PS_F_VCHR_DIST_LN PS_F_VCHR_LN

## Configuring Non-PeopleSoft Delivered Materialized View Logs

If a non-PeopleSoft delivered Materialized View Log exists for the master table of a materialized view, you must configure the view log to support the following requirements:

- All base table key columns
- ROWID option
- SEQUENCE option
- INCLUDING NEW VALUES option

For example:

```
CREATE MATERIALIZED VIEW LOG ON <BASE TABLE> WITH ROWID, SEQUENCE (<BASE TABLE KEY ⇒
COLUMNS>) INCLUDING NEW VALUES;
```

Additionally, you must create a DUMMY materialized view table for each materialized view you created. The fast refresh process automatically purges Materialized View Log data when the materialized view is refreshed, so creating a related dummy materialized view will prevent the Materialized View Log data from being purged.

To create a DUMMY materialized view, you can enter a SQL statement against the Oracle database similar to the following:

```
CREATE MATERIALIZED VIEW <BASE TABLE>_DMV
REFRESH FAST WITH ROWID AS
SELECT <BASE TABLE COLUMN NAME> FROM <BASE TABLE> WHERE 1=2;
```

## OLTP Table Security

Materialized View Log ETL jobs delete Materialized View Log data once the job completes. This normally requires a user to have DELETE permissions, which may jeopardize source data. As such, PeopleSoft provides an additional user (SCHEMA), which has READ and DELETE rights to Materialized View Log data only. This user cannot select or delete records from master tables.

PeopleSoft also delivers two new environment variables, MLOG\_USERNAME, MLOG\_PASSWORD, and MLOG\_SCHEMA for the new user.

---

**Note:** Since Materialized View Log ETL jobs delete Materialized View Log data once the job completes, you should refresh non-PeopleSoft materialized views before running the ETL jobs.

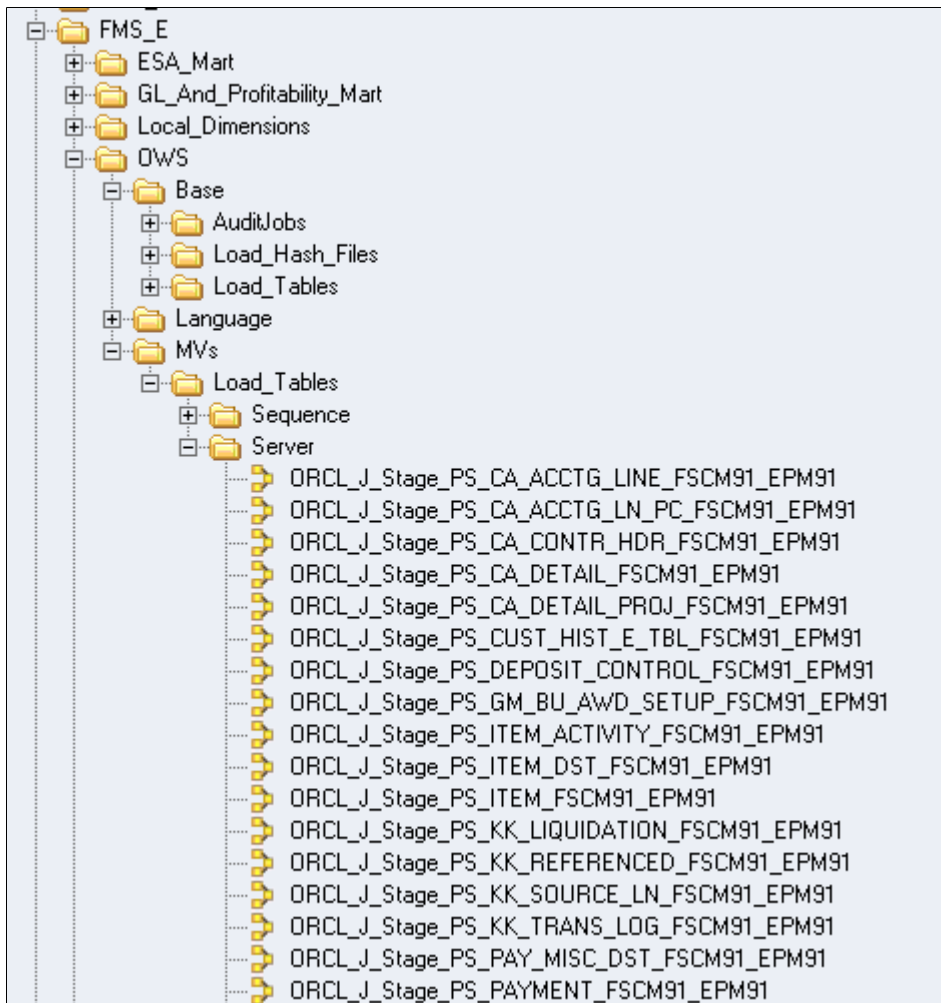
---

## Working with Materialized View Logs in Delivered ETL Jobs

Materialized View Log ETL jobs can be found in the following DataStage folder: *Jobs*, *FMS\_E*, *OWS*, *MVs*, *Load\_Tables*, *Sequence* (and) *Server*.

### Image: Materialized View Logs in the DataStage project tree

This example illustrates the Materialized View Logs in the DataStage project tree.



Materialized View Log ETL jobs are designed to source from Materialized View Logs to dramatically reduce the processing time for OWS jobs that use the CRC process. To facilitate this, the jobs are designed to run the existing CRC job for the first time after you implement the Materialized View Logs in the Oracle database, then run the Materialized-View-specific jobs every time thereafter. This method ensures that there is no data loss on the first job run. For example, if before you implement the Materialized View you insert or modify some records in the source table, they will not be captured in the MLOG table or the CRC hashed file. Thus, in order not to lose any data the very first run after the MLOG implementation will be the CRC job run.

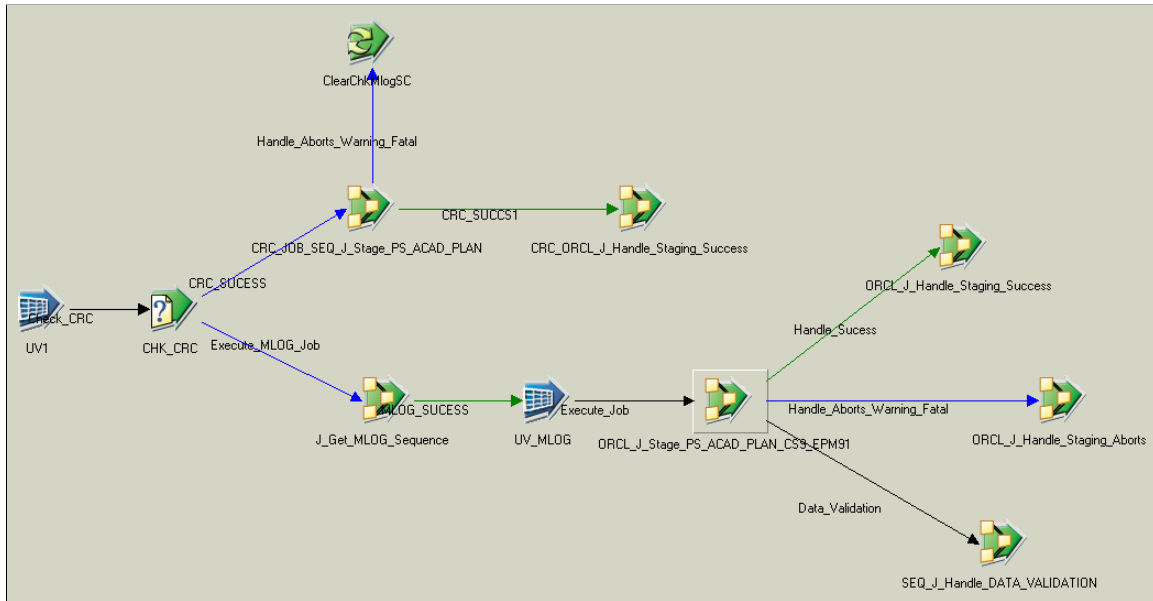
Additionally, Materialized View Log ETL jobs are designed to identify deleted source records so those records can later be deleted in EPM fact tables.

## Describing Materialized View Log ETL Jobs

This section provides a detailed overview of Materialized View Log ETL design, including the processing logic for identifying source record deletes. This section uses the sequencer job SEQ\_J\_Stage\_PS\_ACAD\_PLAN\_CS9\_EPM91 to describe the Materialized View Log ETL design:

### Image: SEQ\_J\_Stage\_PS\_ACAD\_PLAN\_CS9\_EPM91 materialized view log job

This example illustrates the Materialized View Log job SEQ\_J\_Stage\_PS\_ACAD\_PLAN\_CS9\_EPM91.



### UV1 (User Variable 1)

User Variable 1 executes the following routines:

- **RTNCHKMLOGSC:** This routine checks the job name value in the HASH\_MLOG\_END\_SEQUENCE hashed file and returns an appropriate value for the CHK\_CRC nested condition.

If a job name value exists, the routine returns the value 'MLOG,' and the CHK\_CRC nested condition calls the *J\_Get\_MLOG\_Sequence* job.

If a job name value does not exist, the routine returns the value 'CRC,' and the CHK\_CRC nested condition calls the existing CRC job (*CRC\_JOB\_SEQ\_J\_Stage\_PS\_ACAD\_PLAN* in this example).

- **RTNGETMAXMLOGSEQUENCE:** This routine fetches the concatenated value of the start sequence and the end sequence number from the MLOG hashed files HASH\_MLOG\_START\_SEQUENCE and HASH\_MLOG\_END\_SEQUENCE.

---

**Note:** For the first time run the return value will be -1|-1 as the job name will have no entries in either of the two hash files.

---

### CHK\_CRC (Nested Condition)

The CHK\_CRC nested condition checks the return value from the RTNCHKMLOGSC routine and executes one of two ETL jobs, based on the return value.

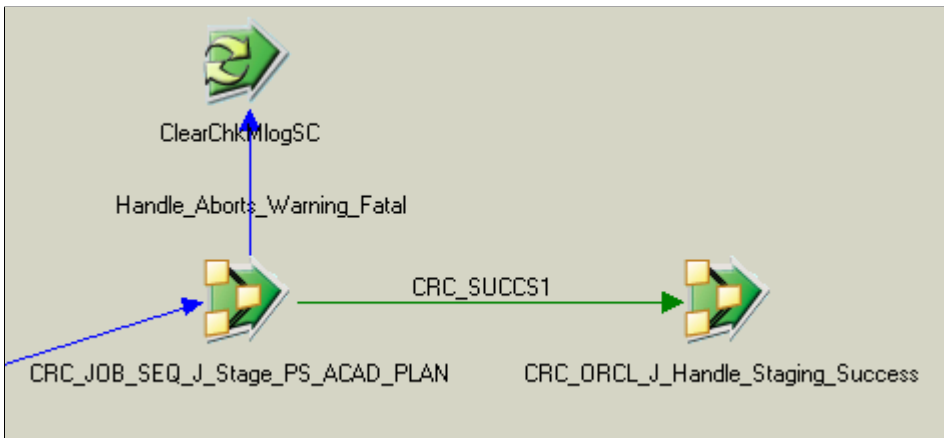
If the routine returns the value 'MLOG,' (job name value exists), the CHK\_CRC nested condition calls the *J\_Get\_MLOG\_Sequence* job. This job fetches the Maximum Sequence Number from the MLOG table and writes it to the hashed file HASH\_MLOG\_END\_SEQUENCE.

If the routine returns the value 'CRC,' (job name value does not exist), the CHK\_CRC nested condition calls the existing CRC job (*CRC\_JOB\_SEQ\_J\_Stage\_PS\_ACAD\_PLAN* in this example).

**Path 1: CRC\_JOB\_SEQ\_J\_Stage\_PS\_ACAD\_PLAN Job Activity**

**Image: CRC\_JOB\_SEQ\_J\_Stage\_PS\_ACAD\_PLAN Job Activity**

This example illustrates the CRC\_JOB\_SEQ\_J\_Stage\_PS\_ACAD\_PLAN Job Activity.



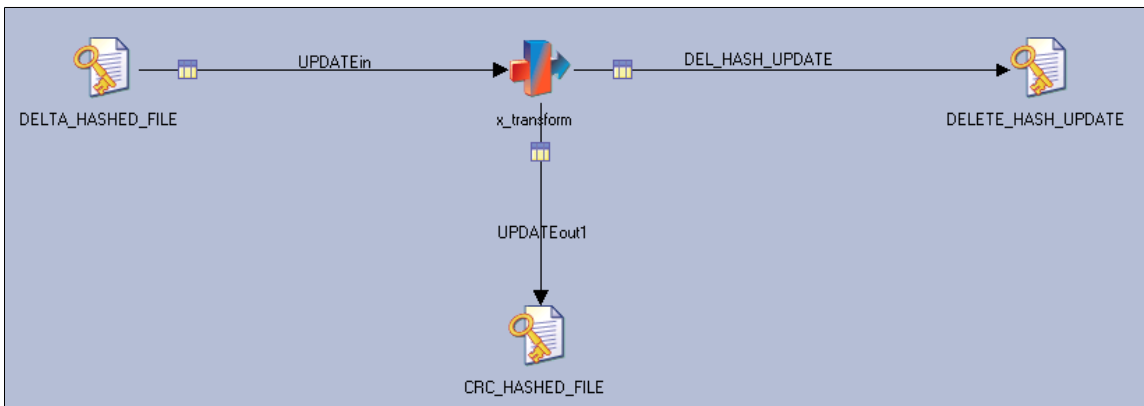
When the CHK\_CRC nested condition calls the existing CRC job (*CRC\_JOB\_SEQ\_J\_Stage\_PS\_ACAD\_PLAN* in this example), it can either run successfully or fail.

If the CRC job fails or aborts, the *ClearChkMlogSC* routine deletes the job name from the HASH\_MLOG\_END\_SEQUENCE hashed file. Thus, when the job runs a second time, it executes the existing CRC job.

If the CRC job runs successfully, the job *CRC\_ORCL\_J\_Handle\_Staging\_Success* is called. This job moves data from the delta hashed file to the CRC\_HASHED\_FILE and DELETE\_HASHED\_FILE:

**Image: CRC\_ORCL\_J\_Handle\_Staging\_Success job**

This example illustrates the CRC\_ORCL\_J\_Handle\_Staging\_Success job.

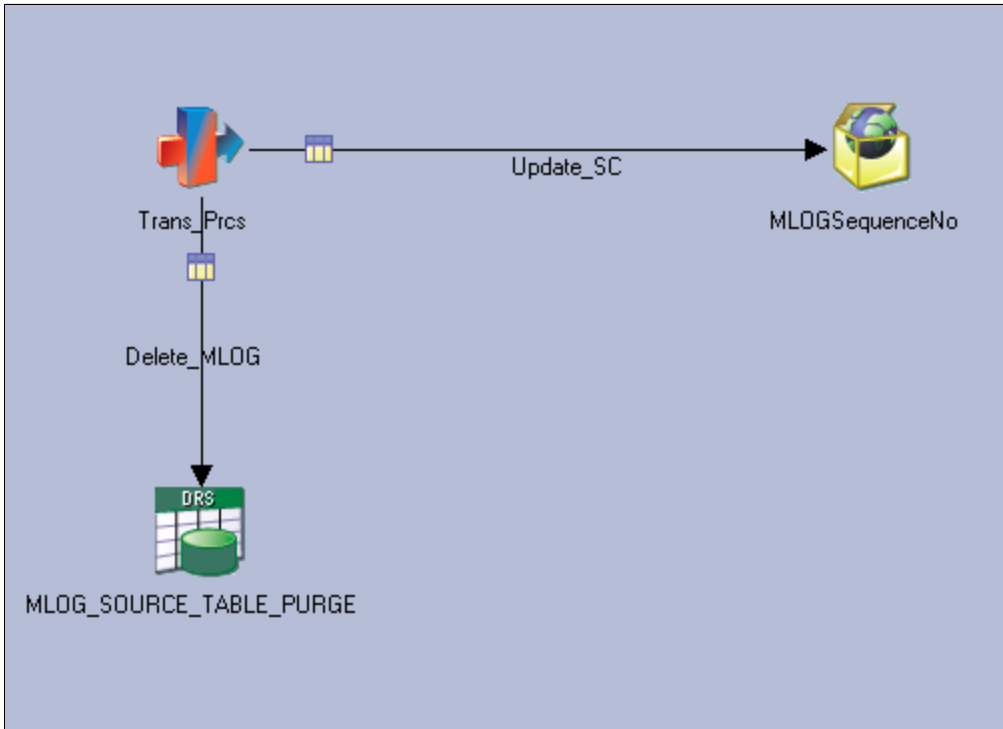


This process keeps the existing CRC hashed files and delete hashed files synchronized.

Also, the Delete\_MLOG output link deletes data from the MLOG table up to the Maximum Sequence Number using the after-SQL in the DRS stage:

**Image: MLOG source table purge**

MLOG source table purge.

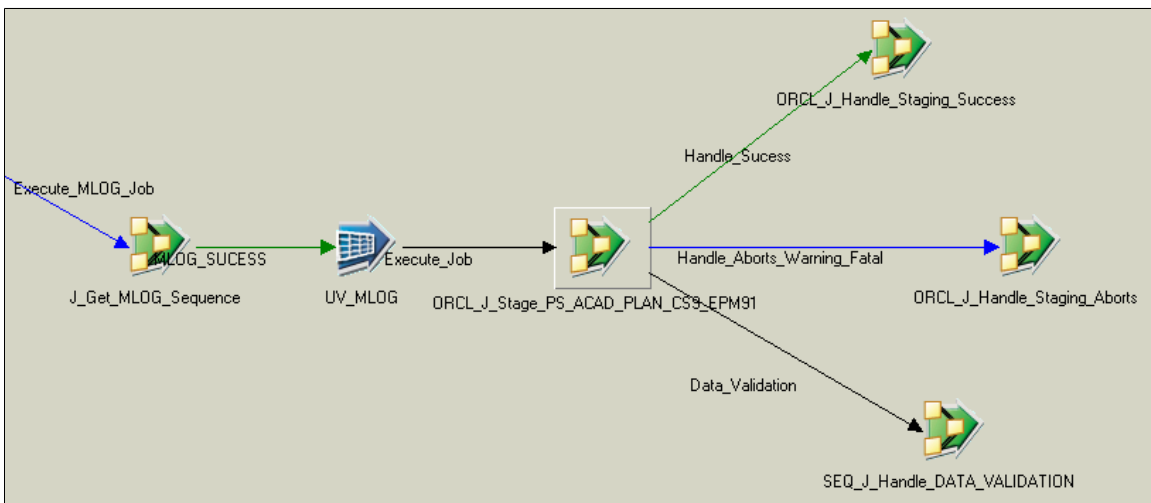


The Update\_SC output link stores the job name and one value higher than the maximum sequence number ( maximum\_seq\_number+1) in the HASH\_MLOG\_START\_SEQUENCE hash file present in the shared container. This value will be used as the START\_SEQUENCE value for the next run.

**Path 2: J\_Get\_MLOG\_Sequence Job Activity**

**Image: Path 2: J\_Get\_MLOG\_Sequence Job Activity**

This example illustrates the Path 2: J\_Get\_MLOG\_Sequence Job Activity.

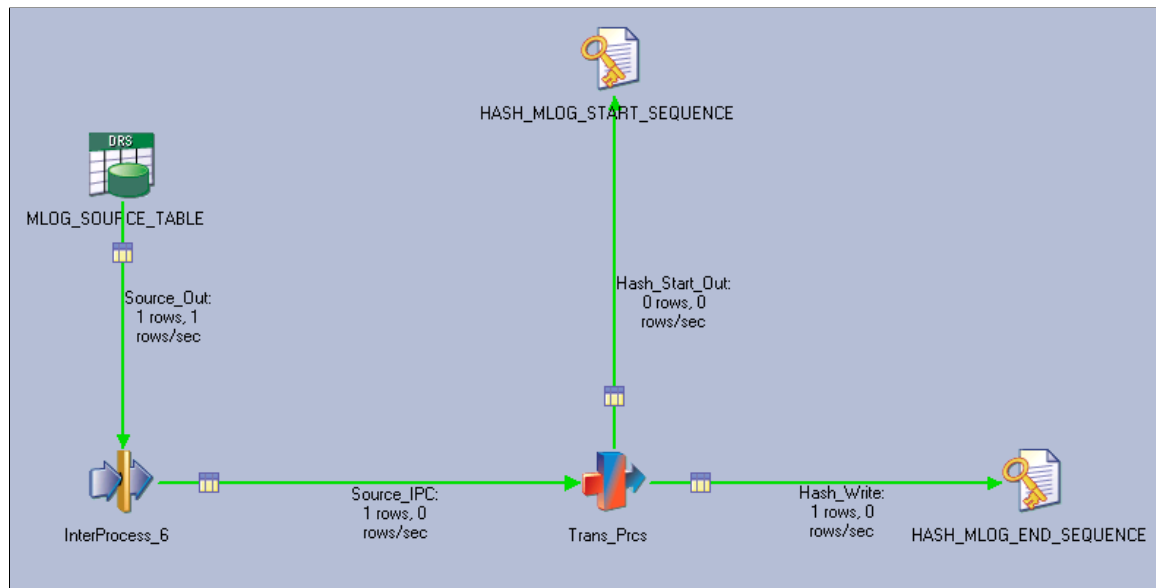




When the CHK\_CRC nested condition calls the J\_Get\_MLOG\_Sequence job, the job fetches the Maximum Sequence Number from the MLOG table and writes it to the HASH\_MLOG\_END\_SEQUENCE hashed file:

**Image: J\_Get\_MLOG\_Sequence job**

This example illustrates the J\_Get\_MLOG\_Sequence job.



The UV\_MLOG user variable calls the following routines:

- *MAX\_MLOG\_SEQ (RTNGETMAXMLOGSEQUENCE)*: Fetches the concatenated value of the start sequence and the end sequence numbers from the HASH\_MLOG\_START\_SEQUENCE and HASH\_MLOG\_END\_SEQUENCE hashed files.
- *BATCH\_SID (GETNEXTBATCHNUMBERPARALLEL)*: Generates BATCH\_SIDs.
- *HANDLE\_DELETES (READPARAMETERFILE)*: Fetches the ParameterName value from the CS\_HANDLEDELETES\_SETUP parameter file.

---

**Note:** For FMS Warehouse jobs, it fetches the ParameterName value from the FMS\_HANDLEDELETES\_SETUP parameter file.

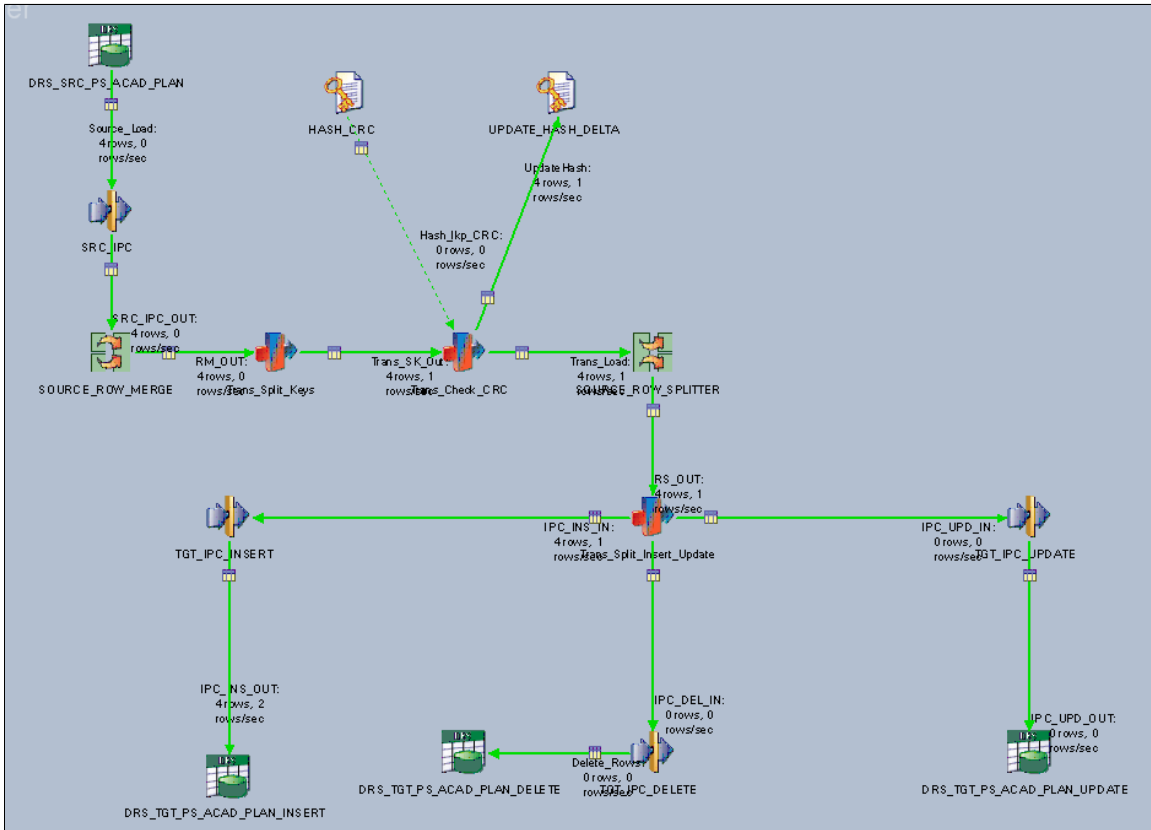
---

- *DELETE\_FLAG (RTNGETDELFLAG)*: Fetches the Delete Flag for the corresponding server job (J\_Stage\_PS\_ACAD\_PLAN\_CS9\_EPM91 in this example).

Next, the ORCL\_J\_Stage\_PS\_ACAD\_PLAN\_CS91\_EPM91 job activity calls the job *ORCL\_J\_Stage\_PS\_ACAD\_PLAN\_CS91\_EPM91*:

**Image: ORCL\_J\_Stage\_PS\_ACAD\_PLAN\_CS91\_EPM91 job**

ORCL\_J\_Stage\_PS\_ACAD\_PLAN\_CS91\_EPM91 job



This job fetches data from the base source table and the MLOG table and loads it to the target OWS table. A right outer join on the MLOG table ensures all newly inserted, modified and deleted rows are extracted. The existing CRC hashed file identifies inserted and updated records.

The source DRS identifies deleted records from the master table using the `BASE_KEY`, which can be any of the key columns of the source table. When a record is deleted from the master table, the MLOG table captures that record but the record will no longer exist in master. As such, a RIGHT OUTER JOIN on the MLOG table fetches the deleted rows. For deleted records the `BASE_KEY` column will be NULL, indicating that the record no longer exists in the source table.

The `UPDATE_HASH_DELTA` hashed file is also maintained in this job.

# Configuring Slowly Changing Dimensions

---

## 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.

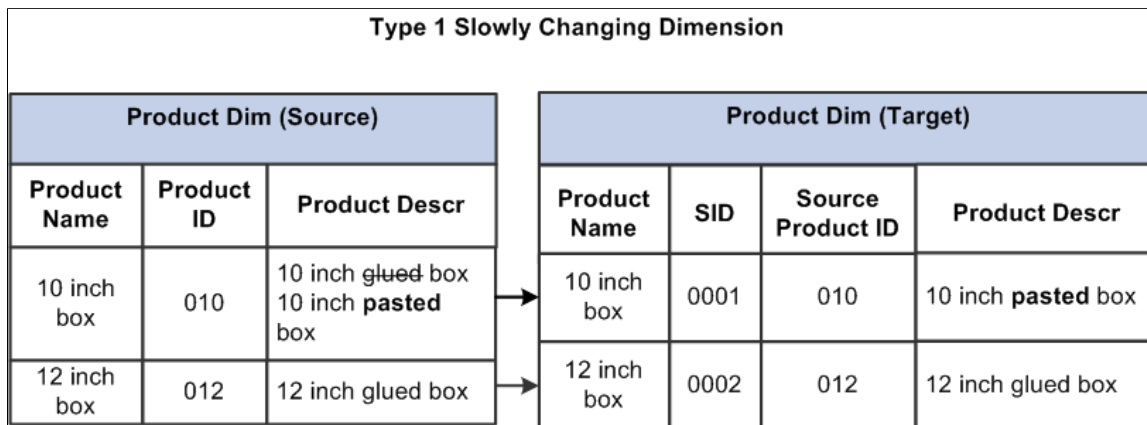
For example, a company that manufactures cardboard boxes might have a Product dimension table that tracks the product ID, product name, and product description. Similar columns would be present in the *warehouse* Product dimension, with the addition of a surrogate ID (primary key) to track each unique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

**Image: 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:

**Image: Type 2 slowly changing dimension**

Slowly changing dimension

Type 2 Slowly Changing Dimension								
Product Dim (Source)			Product Dim (Target)					
Product Name	Product ID	Product Descr	SID	Source Product ID	Product Name	Product Descr	EFF_START_DT	EFF_END_DT
12 inch box	012	12 inch glued box	0001	012	12 inch box	12 inch glued box	Jan-01-1753	Dec-31-9999
10 inch box	010	10 inch glued box	0002	010	10 inch box	10 inch glued box	Jan-01-1753	May-12-06
		10 inch pasted box	0003	010	10 inch box	10 inch pasted box	May-12-06	Dec-31-9999

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 documentation.

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:

<b>Column</b>	<b>Data Type</b>
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:

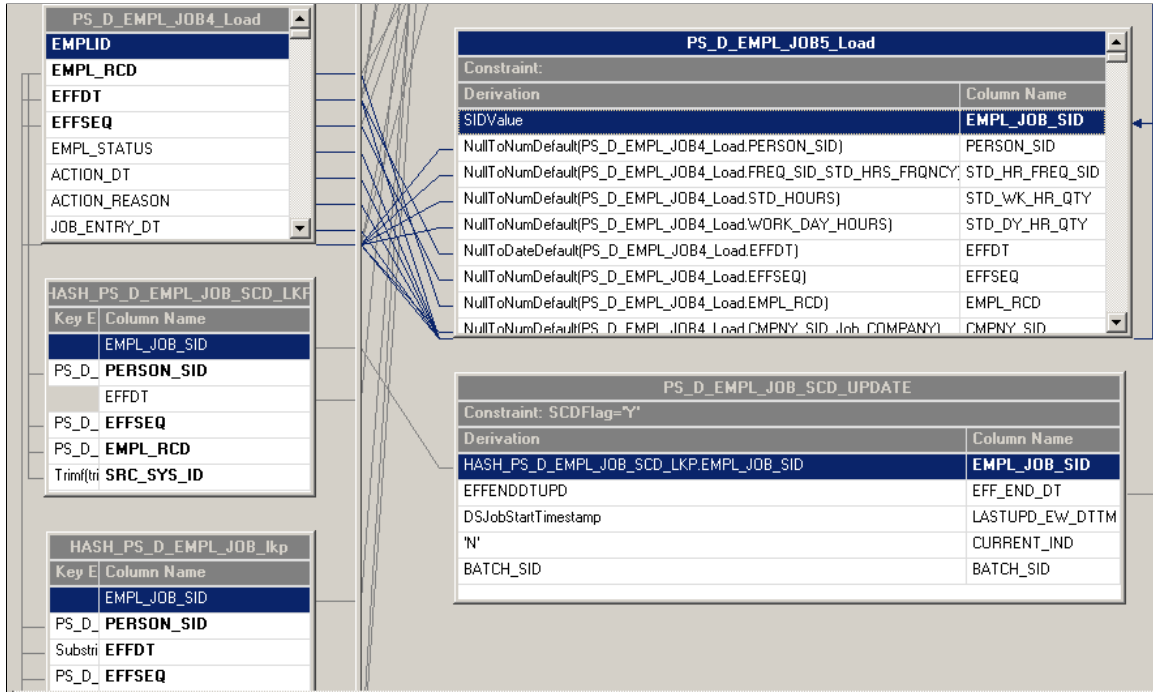
- The first target DRS stage uses an *update existing rows only* logic for its loading strategy.

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.

**Image: Slowly changing dimension and target DRS stage**

This example illustrates the fields and controls on the Slowly changing dimension and target DRS stage. You can find definitions for the fields and controls later on this page.



**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.

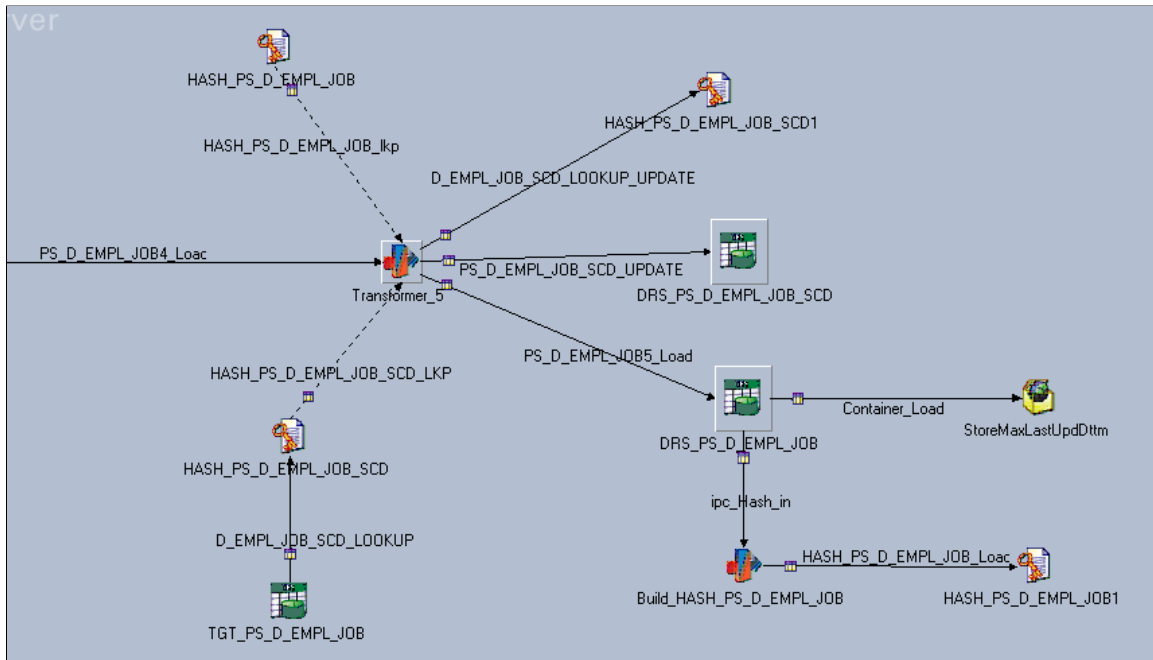
Please refer the figure below to get the information about the lookups (HASH\_PS\_D\_EMPL\_JOB\_SCD1 and HASH\_PS\_D\_EMPL\_JOB\_SCD) that are used to determine



the SCD logic. The following Stage Variables are added to determine the SCD logic: SCDFlag, EFFENDDTUPD and EFFENDDT.

### Image: Slowly changing dimension and target lookup stage

This example illustrates the fields and controls on the Slowly changing dimension and target lookup stage. You can find definitions for the fields and controls later on this page.



## 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 [Converting a Hash File Lookup to a Dynamic DRS Lookup in the Related Fact Table Job](#).

---

## 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 [Converting a Hash File Lookup to a Dynamic DRS Lookup in the Related Fact Table Job](#).

### 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.

---

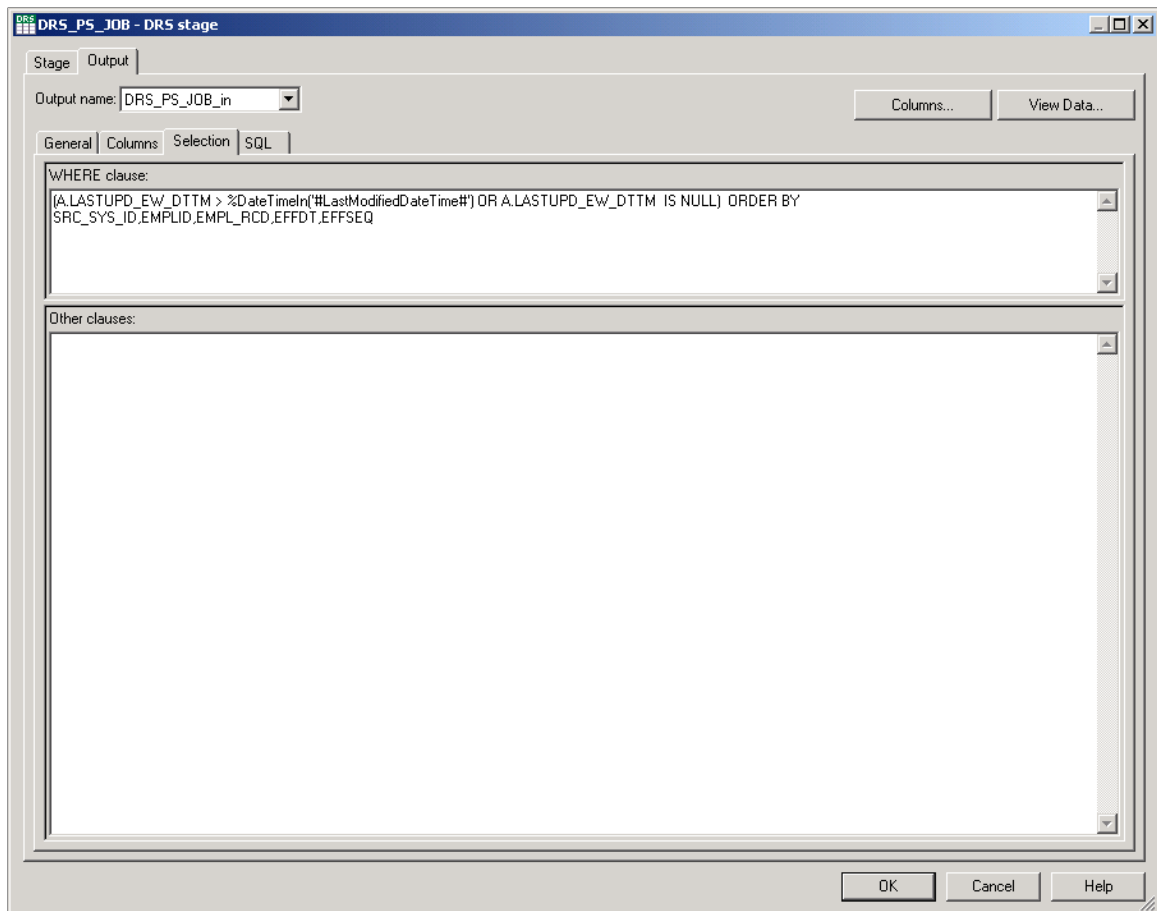
4. Remove the correlating sub-query.

You should be left with the following SQL statement:

(LASTUPD\_EW\_DTTM > %DateTimeIn('#LastModifiedDateTime#') OR LASTUPD\_EW\_DTTM IS NULL) ORDER BY EFFDT, EFFSEQ

### Image: DRS stage - Output tab

This example illustrates the fields and controls on the DRS stage - Output tab. You can find definitions for the fields and controls later on this page.



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

5. Click OK.

## Step 2: Modifying the Target Hash Lookup Stage

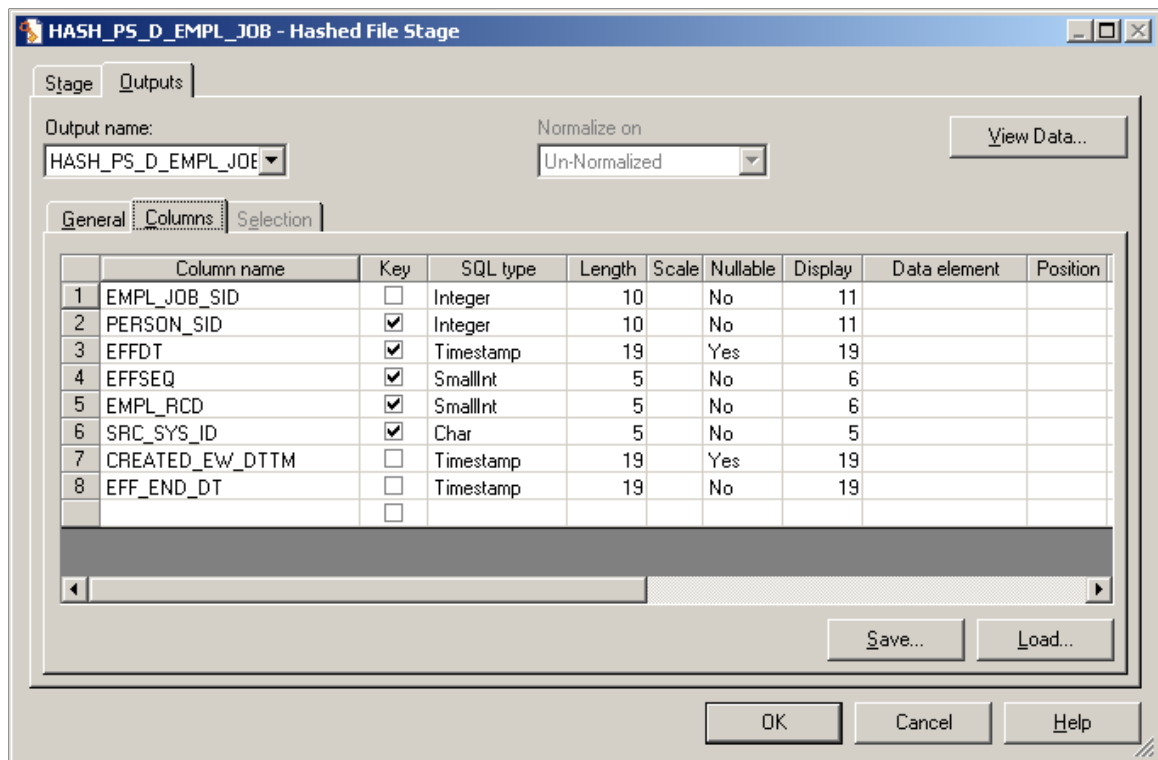
Perform the following steps to modify the target hash lookup stage:

1. Locate the target hash lookup stage within the job and open it.
2. 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.

### Image: Transformer stage - Output tab

This example illustrates the fields and controls on the Transformer stage - Output tab. You can find definitions for the fields and controls later on this page.



6. Click OK.

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

Perform the following steps to add lookup stages:

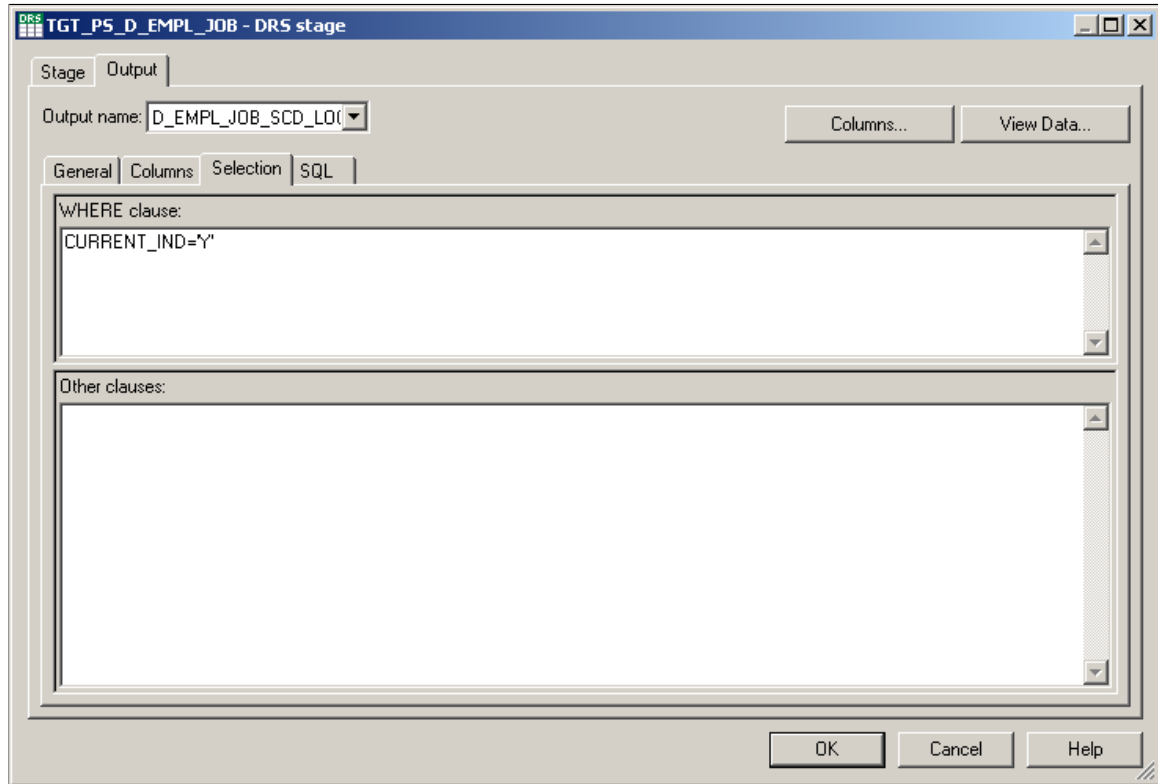
1. Add new *DRS* and *Hash File* stages to the job, placing them next to the transformer that loads the target table.
2. Link the *DRS* stage to the newly added Hash File stage.



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

### Image: DRS stage - Output tab

This example illustrates the fields and controls on the DRS stage - Output tab. You can find definitions for the fields and controls later on this page.



- 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

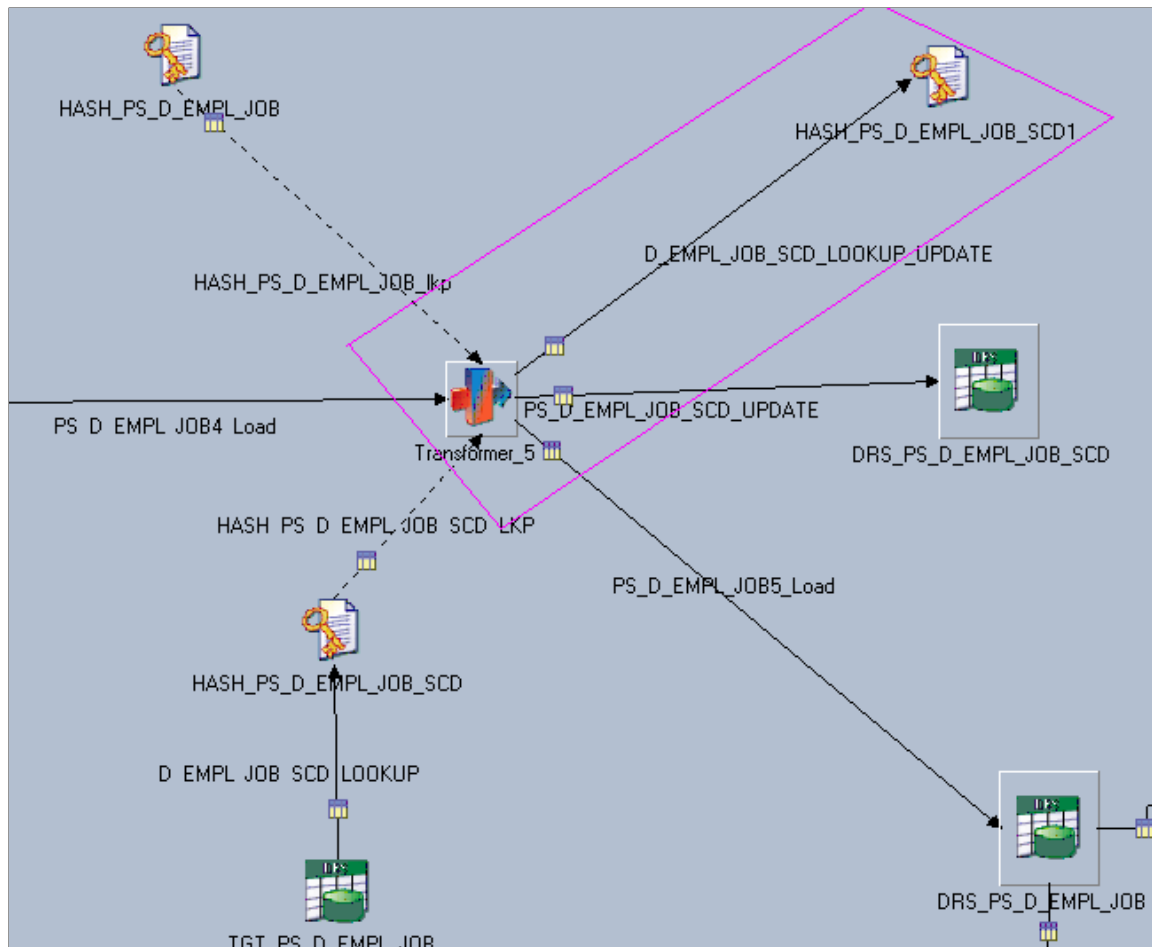
Perform the following steps to add a new hash file stage:

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

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

**Image: Adding a mew hash file stage**

This example illustrates the fields and controls on the Adding a mew hash file stage. You can find definitions for the fields and controls later on this page.



**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.

- Change the hash file name by adding the suffix `_SCD` to it.
- 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:

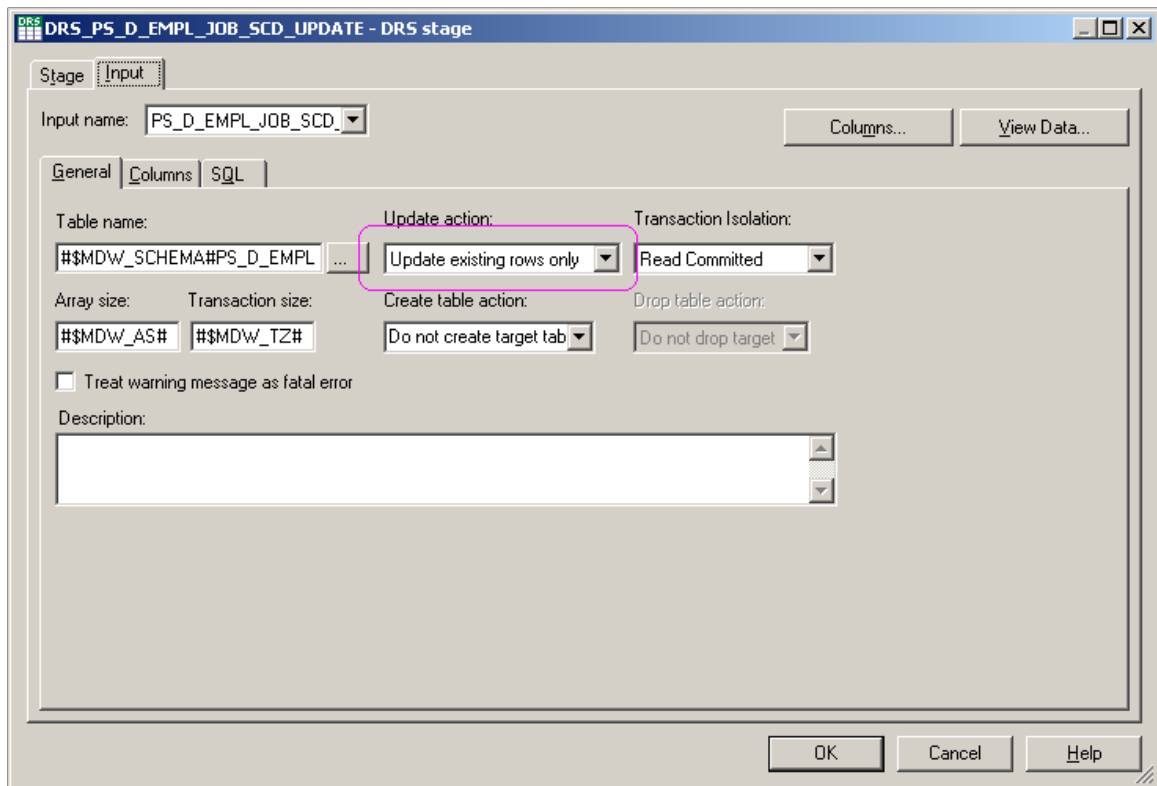
- Copy the DRS stage from where the data is loaded and paste it into the same job.
- Change the new *DRS stage* and *DRS link* names by adding the suffix `_SCD_UPDATE` to them.



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*.

**Image: DRS stage - Input tab**

This example illustrates the fields and controls on the DRS stage - Input tab. You can find definitions for the fields and controls later on this page.



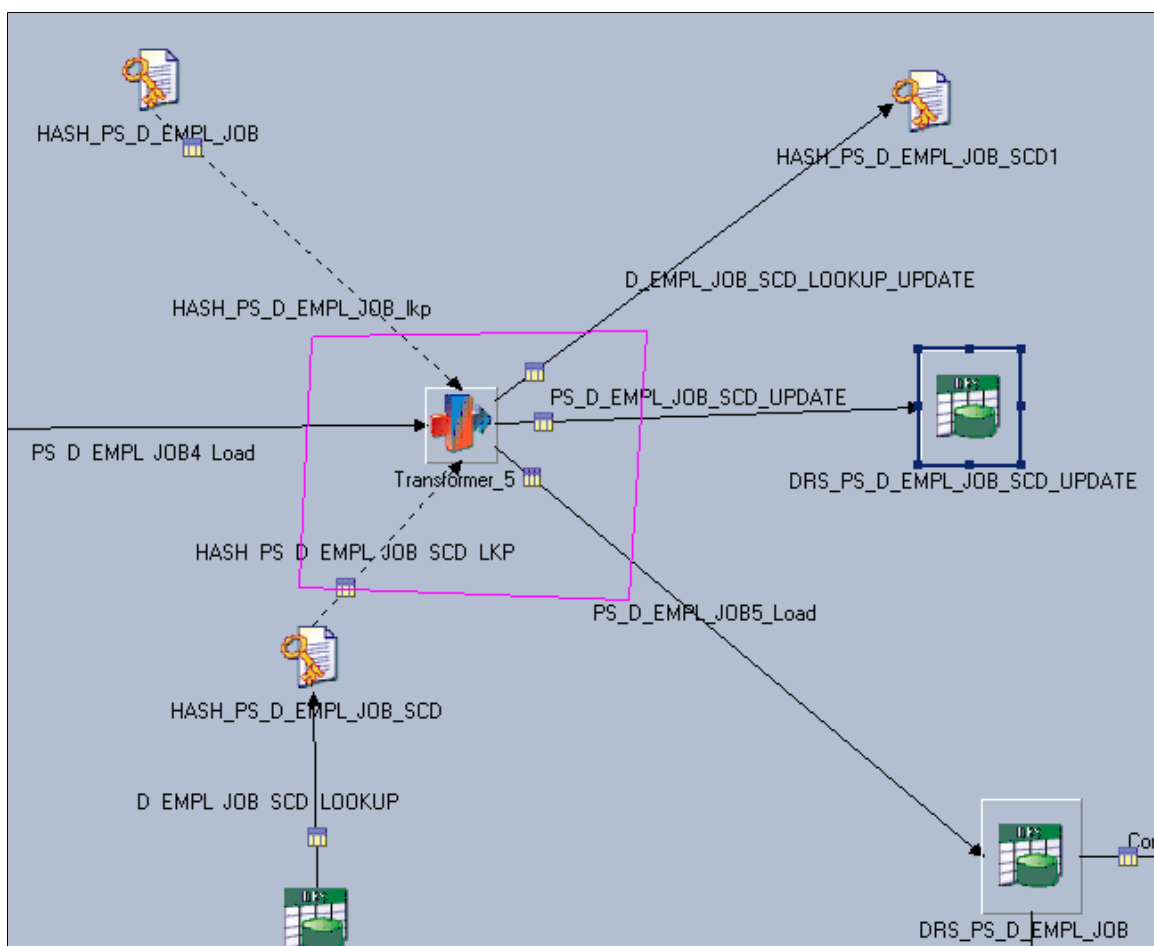
6. Click OK.

## 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:

### Image: Verifying links

This example illustrates the fields and controls on the Verifying links. You can find definitions for the fields and controls later on this page.



## 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>.EFFDT, '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.

### Image: Adding stage variables

This example illustrates the fields and controls on the Adding stage variables. You can find definitions for the fields and controls later on this page.

The screenshot shows the 'Transformer Stage' configuration window. On the left, there are two source link lists: 'PS\_D\_EMPL\_JOB4\_Load' and 'HASH\_PS\_D\_EMPL\_JOB\_SCD\_LKP'. The 'PS\_D\_EMPL\_JOB4\_Load' list includes columns like EMPLID, EMPL\_RCD, EFFDT, EFFSEQ, EMPL\_STATUS, ACTION\_DT, ACTION\_REASON, and JOB\_ENTRY\_DT. The 'HASH\_PS\_D\_EMPL\_JOB\_SCD\_LKP' list includes EMPL\_JOB\_SID, PERSON\_SID, EFFDT, EFFSEQ, EMPL\_RCD, and SRC\_SYS\_ID. On the right, the 'Stage Variables' table is visible, with a new variable 'EFFENDDTUPD' highlighted in pink. Its expression is: 'AddToDate(PS\_D\_EMPL\_JOB4\_Load.EFFDT, 'DD', -1)'. Below this, another expression is shown: 'If len(HASH\_PS\_D\_EMPL\_JOB\_TGT.EFF\_END\_DT)= 0 Then MaxDate Else HASH\_PS\_D\_EMPL\_JOB\_TGT.EFFENDDT'. At the bottom, there are two data tables. The first table shows the source link columns and their properties (Key, SQL type, Length, Scale, Nullable, Display). The second table shows the target link columns and their properties.

Column name	Key	SQL type	Length	Scale	Nullable	Display	Data element
1 EMPLID	<input checked="" type="checkbox"/>	Char	11	No			
2 EMPL_RCD	<input checked="" type="checkbox"/>	SmallInt	5	No			
3 EFFDT	<input checked="" type="checkbox"/>	Timestamp	19	No		19	
4 EFFSEQ	<input checked="" type="checkbox"/>	SmallInt	5	No		6	
5 EMPL_STATUS	<input type="checkbox"/>	Char	1	No		6	
6 ACTION_DT	<input type="checkbox"/>	Timestamp	19	Yes			
7 ACTION_REASON	<input type="checkbox"/>	Char	3	No			

Column name	Key	SQL type	Length	Scale	Nullable	Display	Data element
1 EMPL_JOB_SID	<input type="checkbox"/>	Integer	10	No		11	
2 PERSON_SID	<input checked="" type="checkbox"/>	Integer	10	No		11	
3 EFFDT	<input type="checkbox"/>	Timestamp	19	Yes		19	
4 EFFSEQ	<input checked="" type="checkbox"/>	SmallInt	5	No		6	
5 EMPL_RCD	<input checked="" type="checkbox"/>	SmallInt	5	No		6	
6 SRC_SYS_ID	<input type="checkbox"/>	Char	5	No		5	

## 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>.EFFDT
```

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*]<sub>SCD</sub>).

14. Map the alternate key columns from the input link to alternate keys in the lookup table.

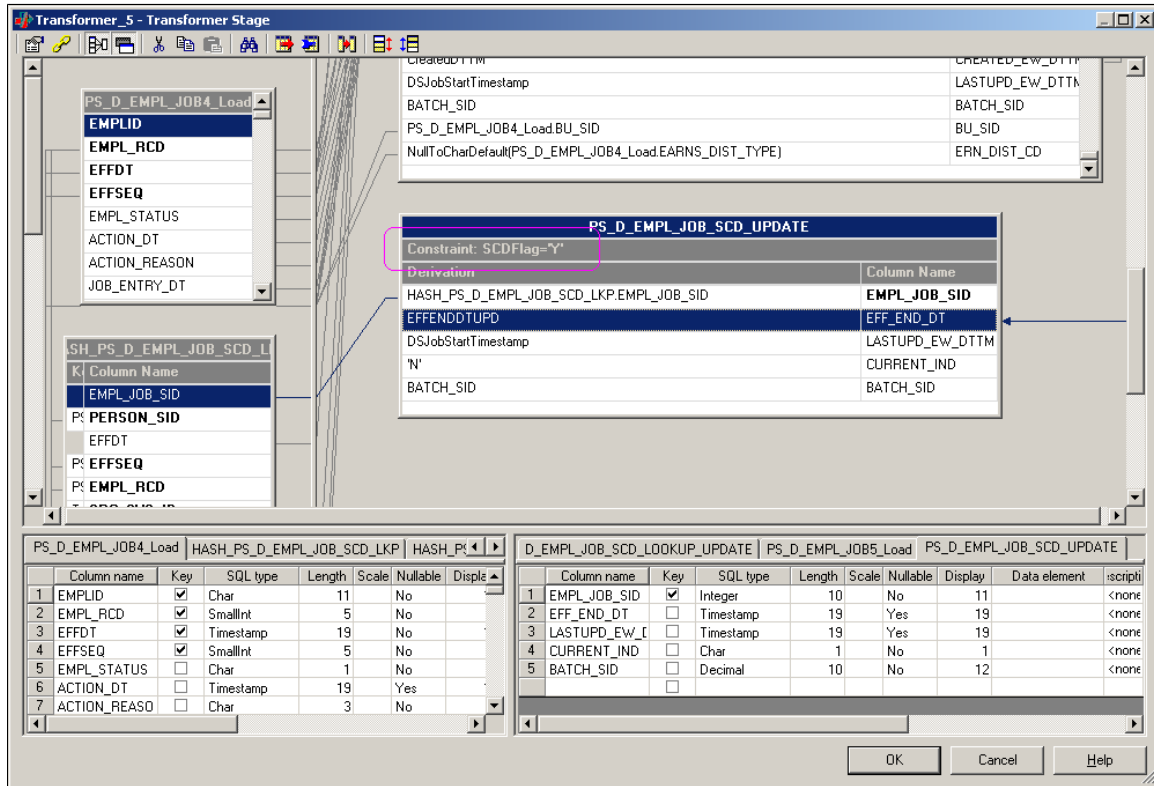
Although the column mapping is one-to-one, there should be proper NULL handling based on the column data type.

15. Modify the expression for the primary key (SID) column so that the expression uses the stage variable *SIDValue*.

16. Click OK.

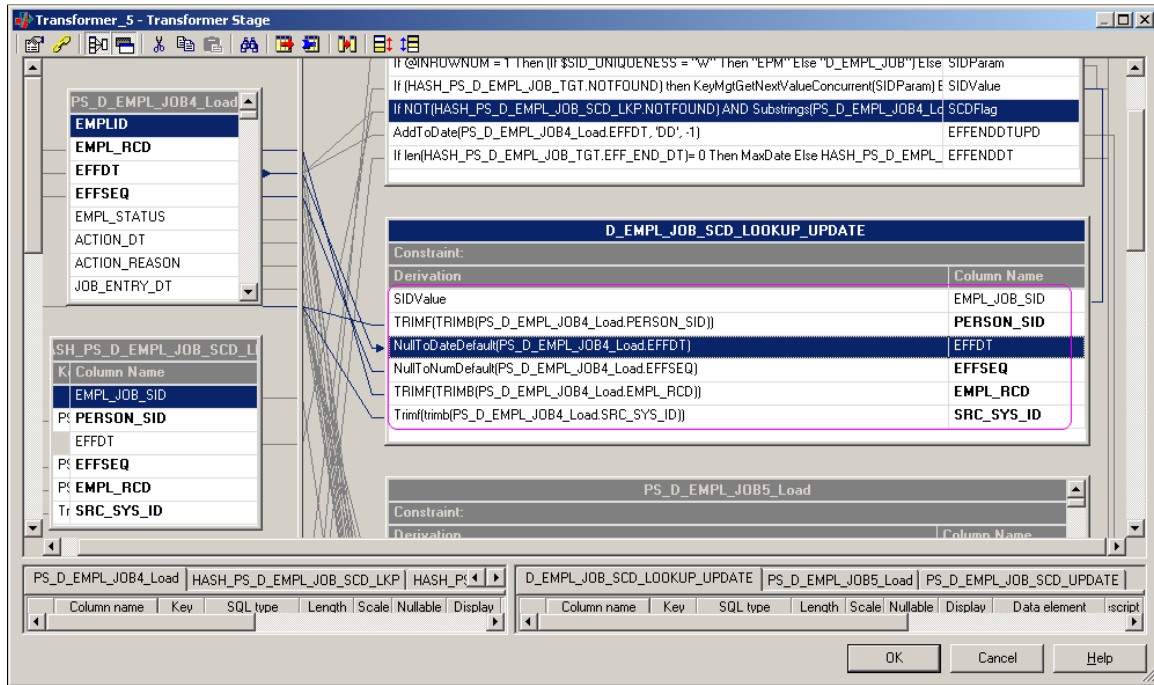
**Image: Modifying column expressions, 1 of 2**

This example illustrates the fields and controls on the Modifying column expressions, 1 of 2. You can find definitions for the fields and controls later on this page.



### Image: Modifying column expressions, 2 of 2

This example illustrates the fields and controls on the Modifying column expressions, 2 of 2. You can find definitions for the fields and controls later on this page.



### 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 your 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.

---



# Managing Source System Deletes

---

## Understanding Source System Deletes and the Source-Delete Diagnostic Staging Jobs

Depending on your business practices, it is sometimes necessary to delete records from your PeopleSoft source transaction system. Previously, incremental load jobs used in the ETL process might not reflect a deleted source record in an EPM Warehouse, which could cause the two systems to become unsynchronized. When the two systems are unsynchronized, an EPM Warehouse may produce reports with incorrect results. Therefore, to ensure synchronicity between the EPM Warehouses and your PeopleSoft source system, PeopleSoft provides new *source-delete diagnostic staging jobs* that identify physically deleted source records and reflect those changes in EPM fact tables.

To ensure synchronicity between the EPM Warehouses and your PeopleSoft source system, source-delete diagnostic staging jobs identify physically deleted source records and reflect those changes in EPM fact tables. Physically deleted source records are flagged in OWS tables and passed to MDW tables, where they are physically deleted from the table. This process varies slightly depending on whether it is executed in a staging job that uses the DTTM (date time stamp) or a staging job that uses Cyclic Redundancy Check (CRC). Both processes are discussed in more detail below.

If you wish to retain deleted source records in the FMS Warehouse, you can disable the source-delete diagnostic staging jobs. This process is discussed in more detail later in this documentation.

---

**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.

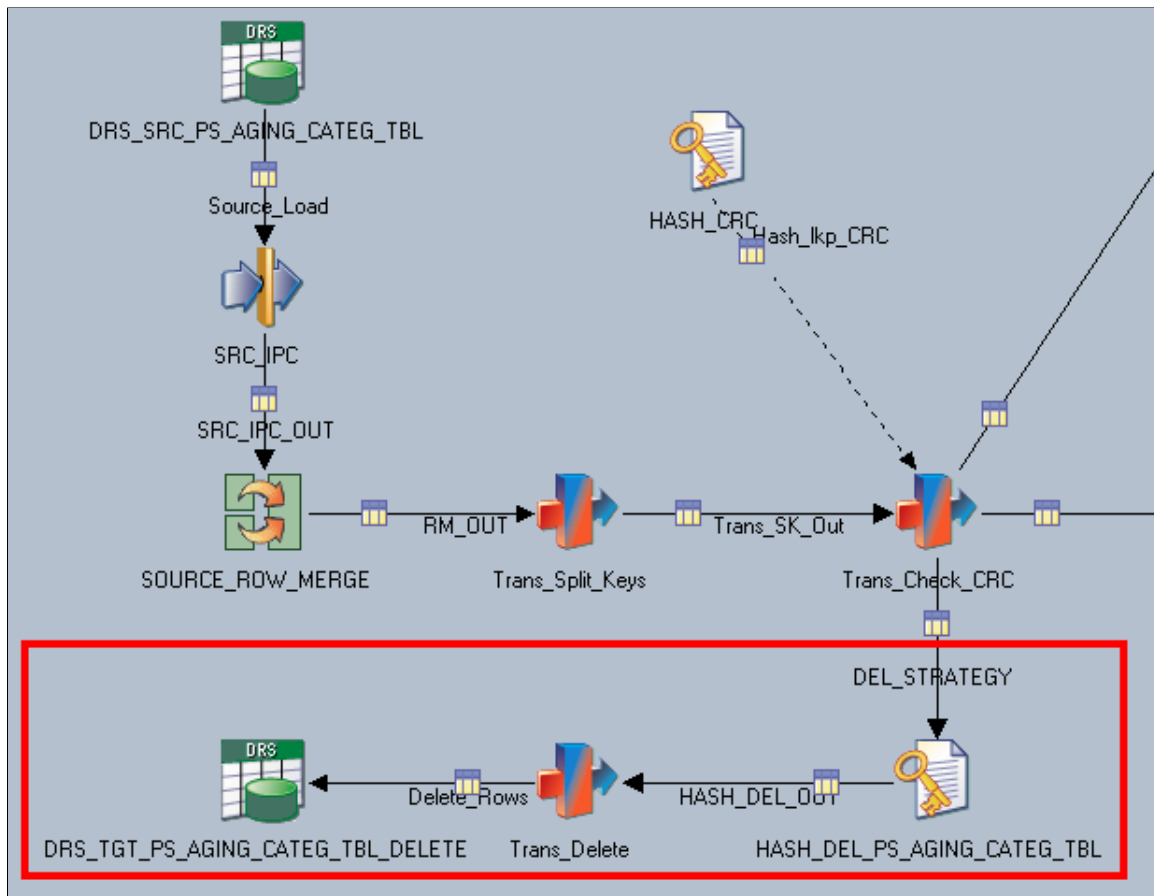
---

## Identifying Source Record Deletes with CRC Staging Jobs

In CRC staging jobs, the ETL logic for the source-delete diagnostic feature is contained in a separate delete strategy (*DEL\_STRATEGY*) path:

### Image: DEL\_STRATEGY path

DEL\_STRATEGY path



This section uses the CRC staging job, *J\_Stage\_PS\_AGING\_CATEG\_TBL\_FSCM91\_EPM91*, as an example.

### Populating the Delete Hashed File

When you first implement the source-delete diagnostic feature, you run a separate setup job (*J\_DELSTRATEGY\_InitialLoad*) that populates the delete hashed file with the complete set of records from the corresponding *HASH\_CRC* hashed file. The process also assigns each record a default value of 'E' for the Delete Flag (*DELFLAG*) column.

The following graphic depicts a hypothetical delete hashed file after it is populated with all the records from its corresponding HASH\_CRC hashed file:

**Image: Hypothetical delete hashed file**

Hypothetical delete hashed file after being populated with all the records from its corresponding hashed file

BUS_KEY				SRC_SYS_ID	DELFLAG
SHARE	10-20	1900-01-01 00:00:00	A1	FSCM	E
SHARE	EMPAG	2000-01-01 00:00:00	01	FSCM	E
SHARE	EMPAG	2000-01-01 00:00:00	04	FSCM	E
SHARE	STD	1900-01-01 00:00:00	02	FSCM	E
SHARE	STD	1900-01-01 00:00:00	03	FSCM	E
SHARE	STD	1900-01-01 00:00:00	05	FSCM	E

Note that every record in the delete hashed file is assigned a value of 'E' for the Delete Flag (DELFLAG) column.

For more information about required steps to implement the source-delete diagnostic feature, see [Enabling the Source-Delete Diagnostic Feature](#)

**Phase One: Trans\_Check\_CRC Stage in the CRC Staging Job**

When you run a CRC staging job, the Trans\_Check\_CRC transformer stage uses the constraint P\_HANDLE\_DELETES = Y to determine whether the job should utilize the delete strategy (DEL\_STRATEGY) path by allowing data loading to the delete hashed file:

**Image: Trans\_Check\_CRC transformer stage, handle deletes constraint**

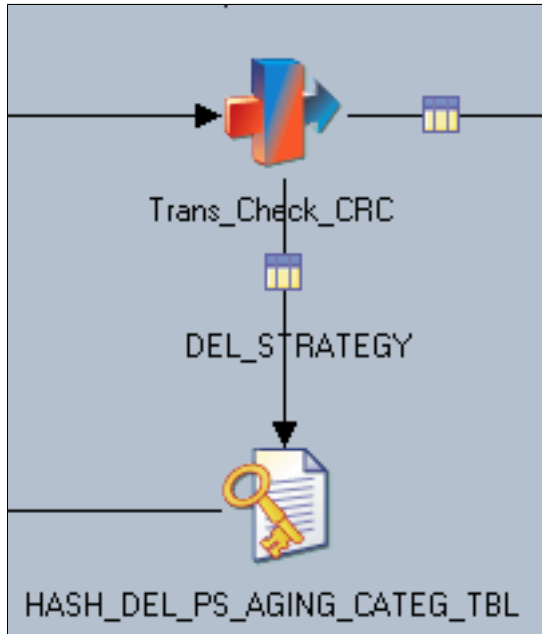
Trans\_Check\_CRC transformer stage, handle deletes constraint

DEL_STRATEGY	
Constraint: P_HANDLE_DELETES = 'Y'	
Derivation	Column Name
Trans_SK_Out.bKey	BUS_KEY
\$FSCM_SRC_SYS_ID	SRC_SYS_ID
P_CURR_PRCES_FLG	DELFLAG

If the parameter for the P\_HANDLE\_DELETES is set to 'Y,' the Trans\_Check\_CRC transformer stage loads the delete hashed file with source records:

**Image: Trans\_Check\_CRC transformer stage**

Trans\_Check\_CRC transformer stage



**Phase Two: HASH\_DEL\_PS\_AGING\_CATEG\_TBL in the CRC Staging Job**

The load process for the delete hashed file uses the business key (BUS\_KEY) and source system ID (SRC\_SYS\_ID) values to compare incoming source records against the records in the delete hashed file.

If an incoming source record matches an existing record in the delete hashed file, the Delete Flag (DELFLAG) value for that record is overwritten with 'EX.' When a record is deleted in the source, its corresponding record in the delete hashed file will remain unchanged, with its Delete Flag (DELFLAG) value remaining 'E.'

The following graphic depicts a hypothetical delete hashed file after being updated by the CRC staging job:

**Image: Hypothetical delete hashed file**

Hypothetical delete hashed file after being updated by the CRC staging job

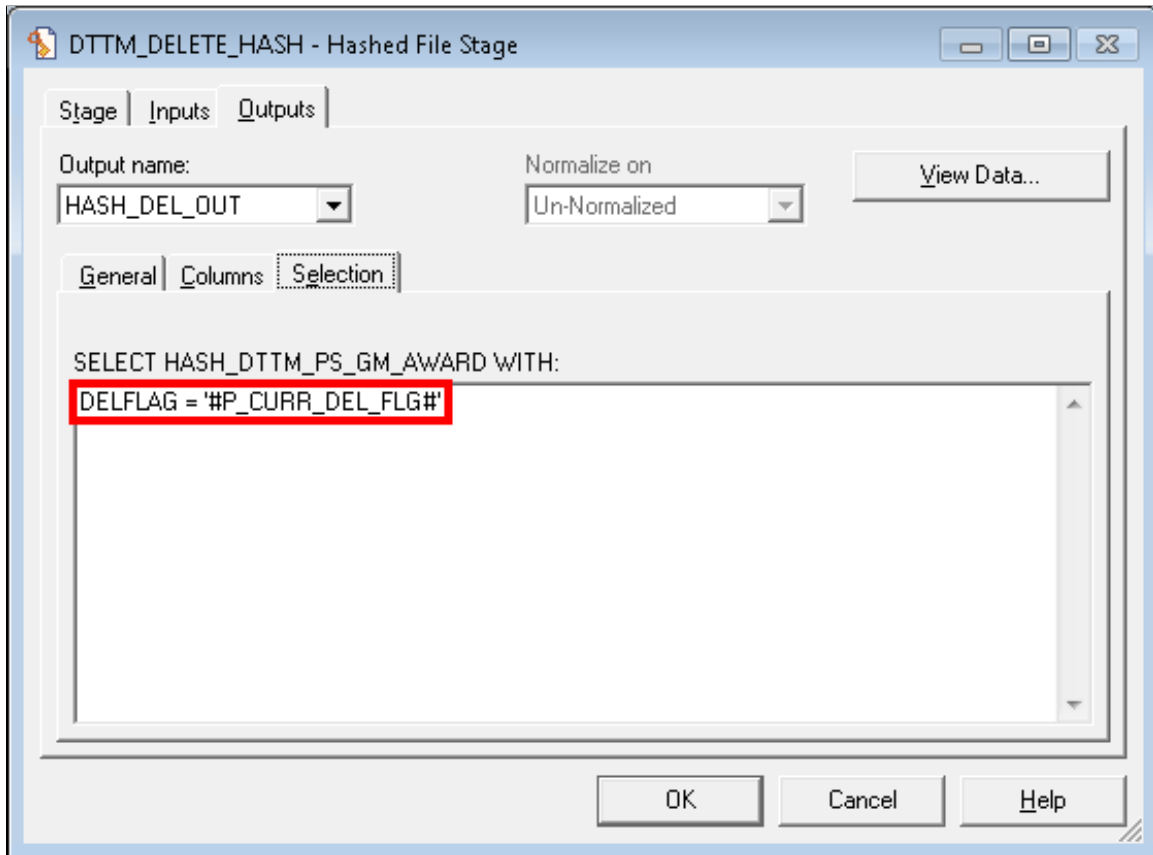
Hypothetical Source Table						Hypothetical Delete Hashed File					
BUS_KEY				SRC_SYS_ID	DELFLAG	BUS_KEY				SRC_SYS_ID	DELFLAG
SHARE	10-20	1900-01-01 00:00:00	A1	FSCM	E	SHARE	10-20	1900-01-01 00:00:00	A1	FSCM	<del>E</del> EX
*Deleted Source Record*						SHARE	EMPAG	2000-01-01 00:00:00	01	FSCM	E
*Deleted Source Record*						SHARE	EMPAG	2000-01-01 00:00:00	04	FSCM	E
SHARE	STD	1900-01-01 00:00:00	02	FSCM	E	SHARE	STD	1900-01-01 00:00:00	02	FSCM	<del>E</del> EX
SHARE	STD	1900-01-01 00:00:00	03	FSCM	E	SHARE	STD	1900-01-01 00:00:00	03	FSCM	<del>E</del> EX
SHARE	STD	1900-01-01 00:00:00	05	FSCM	E	SHARE	STD	1900-01-01 00:00:00	05	FSCM	<del>E</del> EX

Note that two records were deleted from the source and provide no match for the corresponding records in the delete hashed file. Hence, the DELFLAG values for the records remain 'E.'

The DELFLAG value is used to determine which records will be selected for deletion from the fact table. The following graphic shows the selection criteria in the delete hashed file output, which selects *only* records with the DELFLAG value equal to *P\_CURR\_DEL\_FLAG*:

**Image: Selection criteria in the delete hashed file output**

Selection criteria in the delete hashed file output

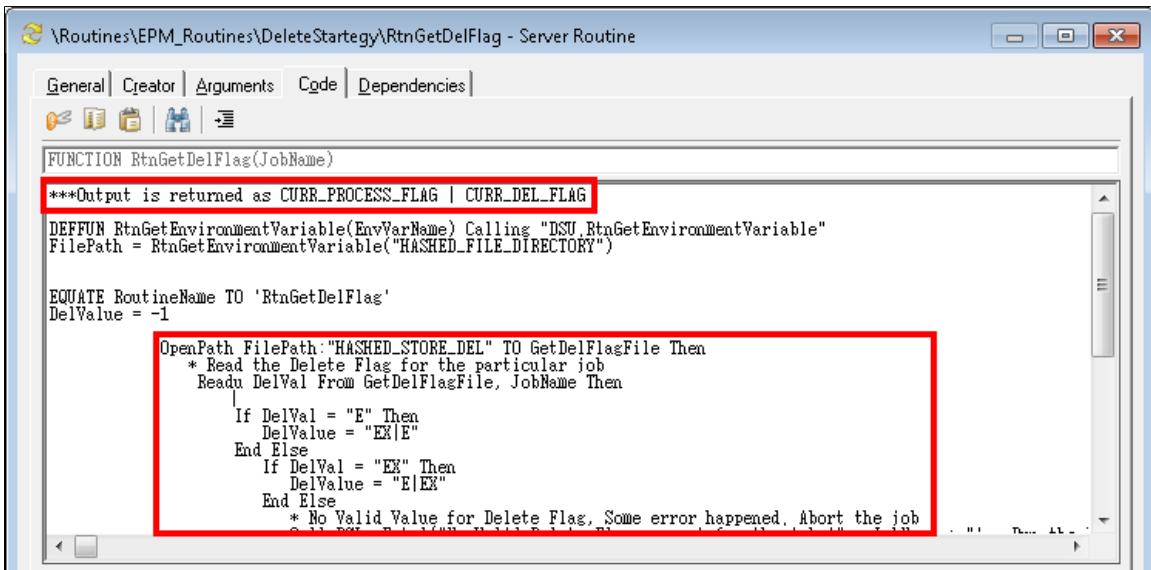


In our example, the P\_CURR\_DEL\_FLAG value is 'E.' Therefore, only records with 'E' for the DELFLAG are selected for deletion and passed to the Trans\_Delete transformer stage.

The current value for P\_CURR\_DEL\_FLAG is determined using the routine *RtnGetDelFlag*:

**Image: RtnGetDelFlag**

RtnGetDelFlag



See [Phase Six: J Handle Staging Success Job](#) for information about the methodology used to update delete hashed file records flagged for deletion.

**Phase Three: Trans\_Delete Transformer in the CRC Staging Delete Job**

At this point in the process, only records with 'E' for the DELFLAG are moved to the Trans\_Delete transformer stage for deletion:

**Image: Trans\_Delete Transformer**

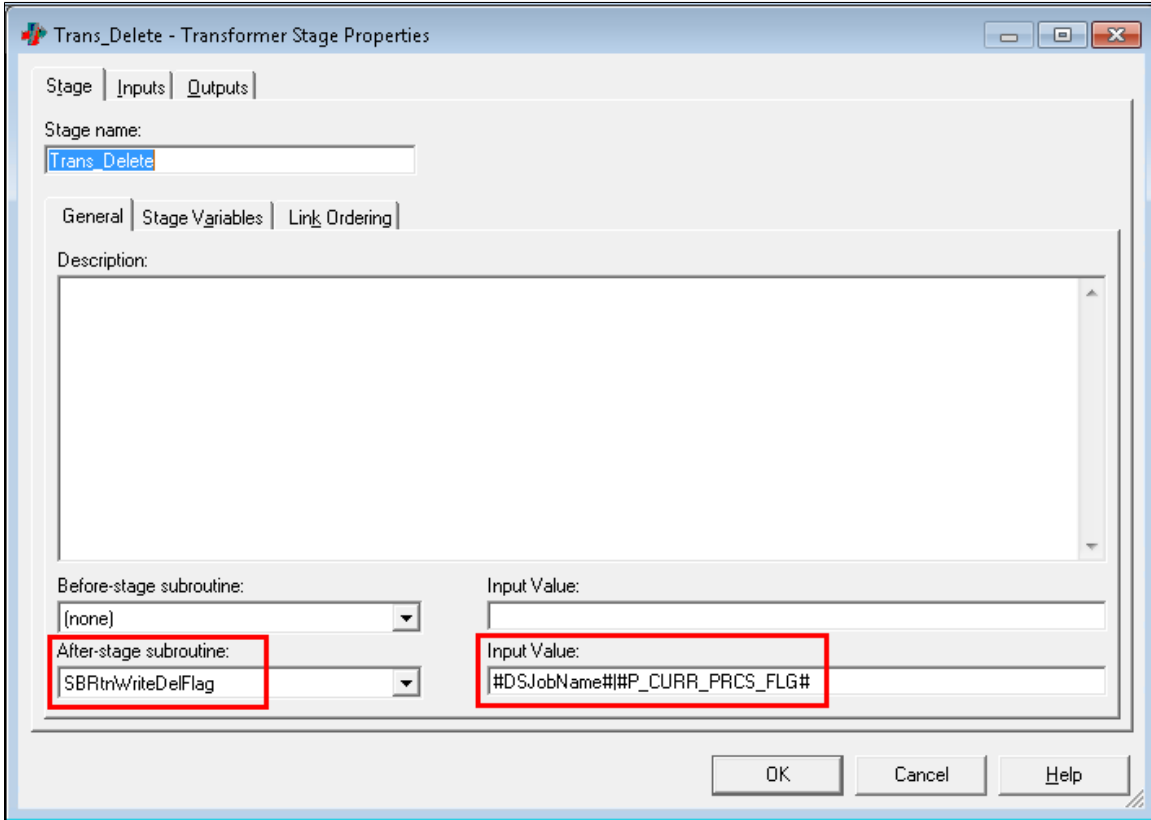
Trans\_Delete Transformer



The Trans\_Delete transformer stage calls the subroutine *SBRtnWriteDelFlag*, which updates the current DELFLAG value (stored in HASHED\_STORE\_DEL hashed file) from 'E' to 'EX.'

**Image: Trans\_Delete transformer stage logic to call the subroutine SBRtnWriteDelFlag**

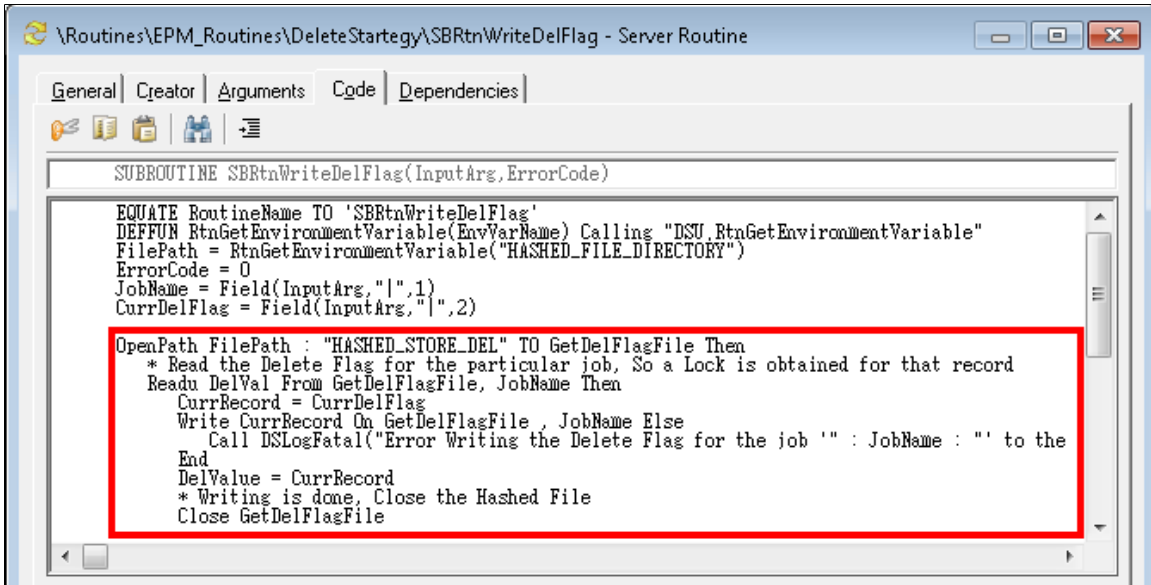
Trans\_Delete transformer stage logic to call the subroutine SBRtnWriteDelFlag



SBRtnWriteDelFlag logic:

**Image: SBRtnWriteDelFlag logic**

SBRtnWriteDelFlag logic



This way, when the job runs again in the future, the starting DELFLAG value will be 'EX.' For example:

**Image: Sample Delete hashed file**

Sample Delete hashed file

BUS_KEY				SRC_SYS_ID	DELFLAG
SHARE	10-20	1900-01-01 00:00:00	A1	FSCM	EX
SHARE	STD	1900-01-01 00:00:00	02	FSCM	EX
SHARE	STD	1900-01-01 00:00:00	03	FSCM	EX
SHARE	STD	1900-01-01 00:00:00	05	FSCM	EX

This swapping process occurs each time you run the CRC staging job.

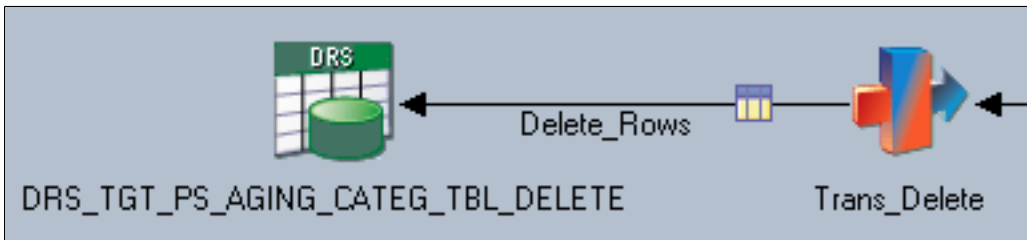


### Phase Four: Delete\_Rows Out Link in the CRC Staging Delete Job

Since the target fact table uses the value in the DATA\_ORIGIN column to identify records to delete, the *Delete\_Rows* out link populates the DATA\_ORIGIN column for outbound records with the value 'D.'

#### Image: Trans\_Delete Transformer to DRS\_TGT\_PS\_GM\_AWARD\_DELETE

Trans\_Delete Transformer to DRS\_TGT\_PS\_GM\_AWARD\_DELETE



#### Image: Trans\_DeleteStage

Trans\_DeleteStage

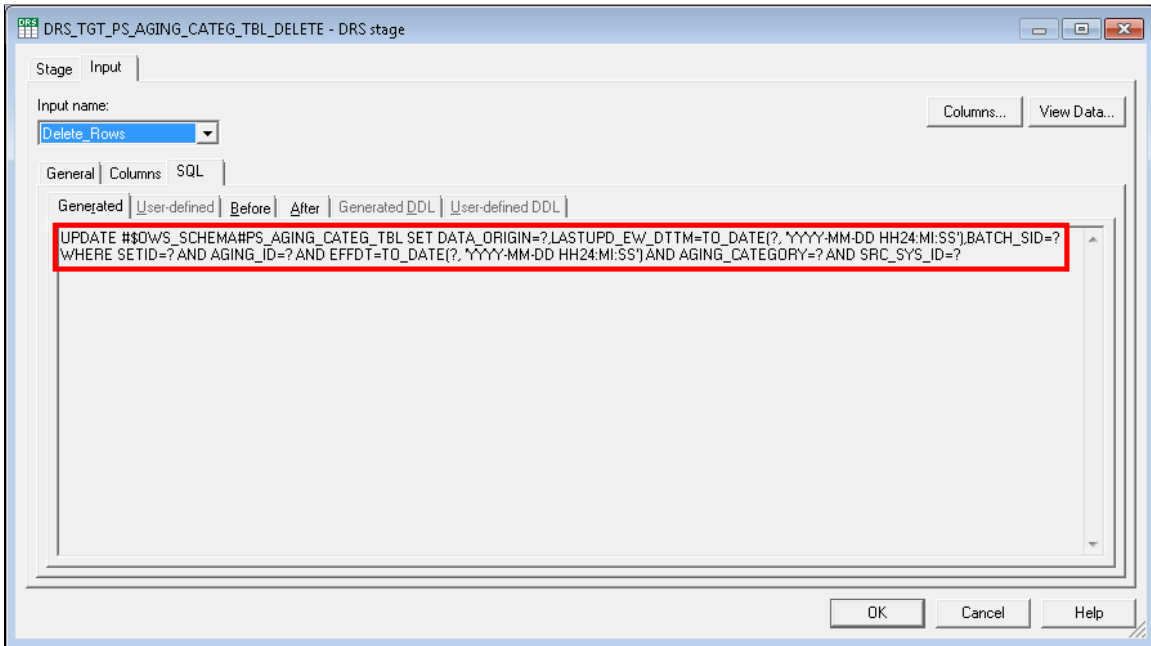
Delete_Rows	
Constraint: P_HANDLE_DELETES = 'Y'	
Derivation	Column Name
NullToCharDefault(Field(ClnsdFld,"I",1))	SETID
NullToCharDefault(Field(ClnsdFld,"I",2))	AGING_ID
NullToDateDefault(Field(ClnsdFld,"I",3))	EFFDT
NullToCharDefault(Field(ClnsdFld,"I",4))	AGING_CATEGORY
HASH_DEL_OUT.SRC_SYS_ID	SRC_SYS_ID
'D'	DATA_ORIGIN
JobStartTimeStamp	LASTUPD_EW_DTTM
BATCH_SID	BATCH_SID

### Phase Five: DRS\_TGT\_PS\_AGING\_CATEG\_TBL\_DELETE in the CRC Staging Delete Job

The DRS target delete stage contains a generated SQL update action that updates DATA\_ORIGIN value to 'D,' which identifies records to delete from the fact table.

#### Image: DRS target delete stage

DRS target delete stage

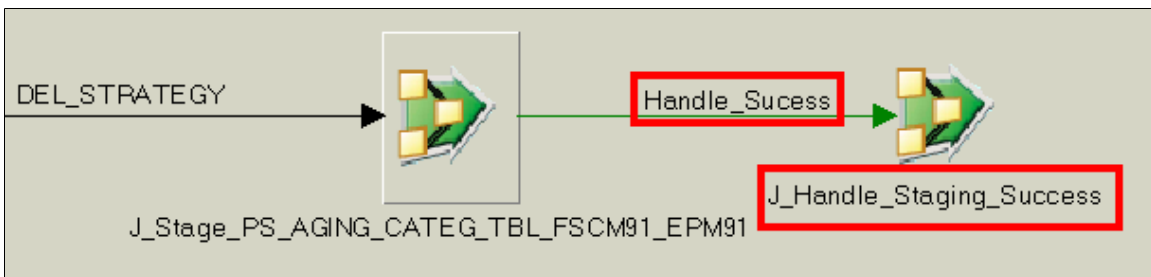


### Phase Six: J\_Handle\_Staging\_Success Job

After the delete strategy portion of the CRC staging job completes, the SEQ\_J\_Stage\_PS\_AGING\_CATEG\_TBL sequencer job calls the job *J\_Handle\_Staging\_Success*:

#### Image: SEQ\_J\_Stage\_PS\_AGING\_CATEG\_TBL sequencer job

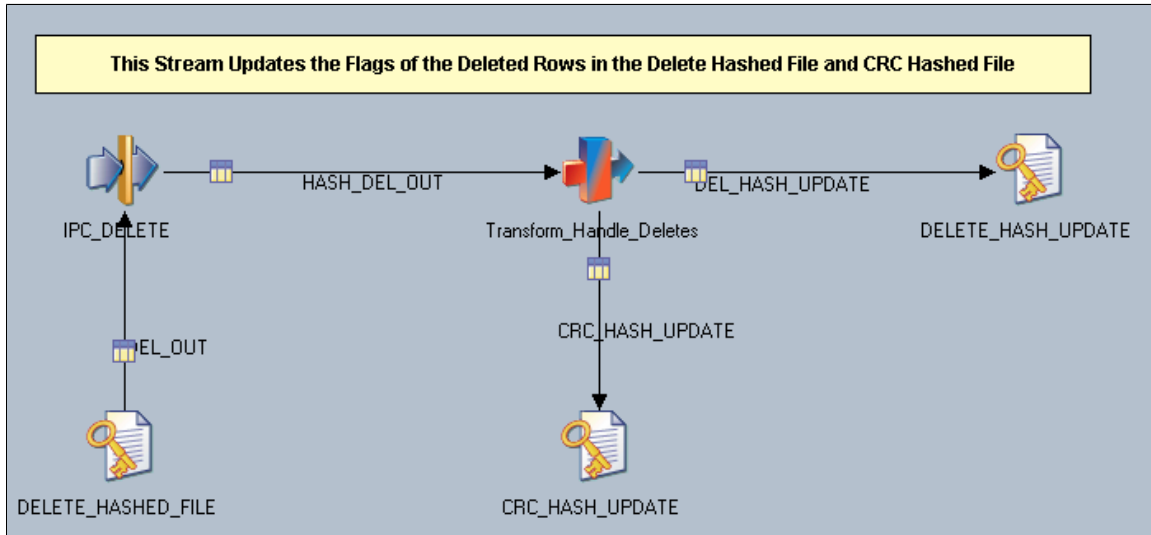
SEQ\_J\_Stage\_PS\_AGING\_CATEG\_TBL sequencer job



One of the tasks for the job `J_Handle_Staging_Success` is to update the delete flag for the Delete hashed file:

**Image: J\_Handle\_Staging\_Success job**

`J_Handle_Staging_Success` job



**Image: J\_Handle\_Staging\_Success job detail**

`J_Handle_Staging_Success` job detail

DEL_HASH_UPDATE	
<b>Constraint:</b>	
Derivation	Column Name
HASH_DEL_OUT.BUS_KEY	<b>BUS_KEY</b>
HASH_DEL_OUT.SRC_SYS_ID	<b>SRC_SYS_ID</b>
'DEL'	<b>DELFLAG</b>

Recall that the delete hashed file in our example CRC staging job output records with delete flag equal to 'E,' since 'E' indicated records for deletion. However, the delete hashed file also contained records flagged with 'EX,' which indicated records to keep.

See [Phase Two: HASH\\_DEL\\_PS\\_AGING\\_CATEG\\_TBL in the CRC Staging Job](#)

Also recall that the subroutine 'SBRtnWriteDelFlag' updated the starting delete flag value to 'EX,' so that the next time the CRC staging job runs, records with the delete flag value equal to 'EX' will be deleted (as opposed to records with 'E'). Therefore, the delete hashed file records flagged with 'E' must be modified in some way to avoid causing conflict during the next run of the CRC staging job. The job `J_Handle_Staging_Success` updates these delete hashed file records from 'E' to 'DEL.' Delete hashed file records flagged with 'DEL' are not included in the delete strategy process during subsequent runs of the CRC staging job.

## Identifying Source Record Deletes with Date-Time Staging Jobs

Date-Time staging jobs process source deletes in a two-phased approach:

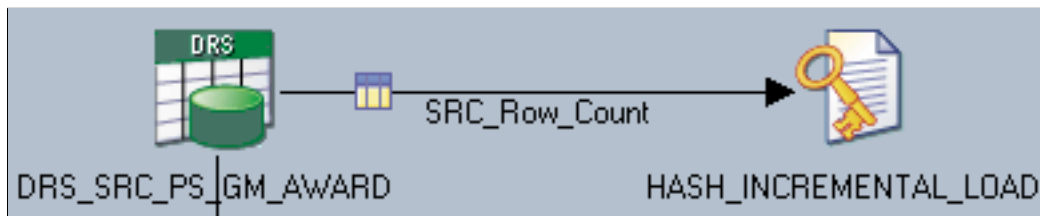
1. Incremental load staging jobs include source and target row count logic to determine whether a source record delete occurred.
2. If a source record delete occurred, a separate staging delete job processes the records for deletion.

### Phase One: Incremental Load Staging Jobs

Incremental load staging jobs first use a hashed file lookup (HASH\_INCREMENTAL\_LOAD) to compare row counts from the source with existing row counts from the hashed file:

#### Image: Incremental Load Staging Job

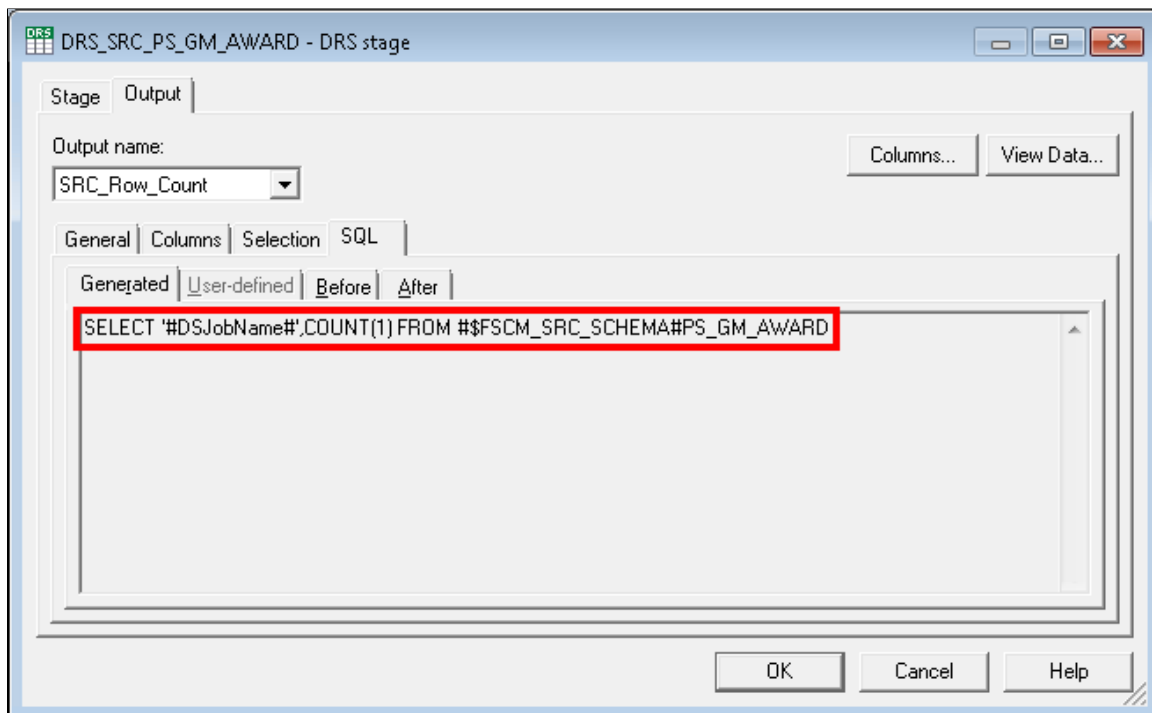
Incremental Load Staging Job



The DRS source stage contains a SQL select statement that obtains row count from the source table:

#### Image: DRS source stage

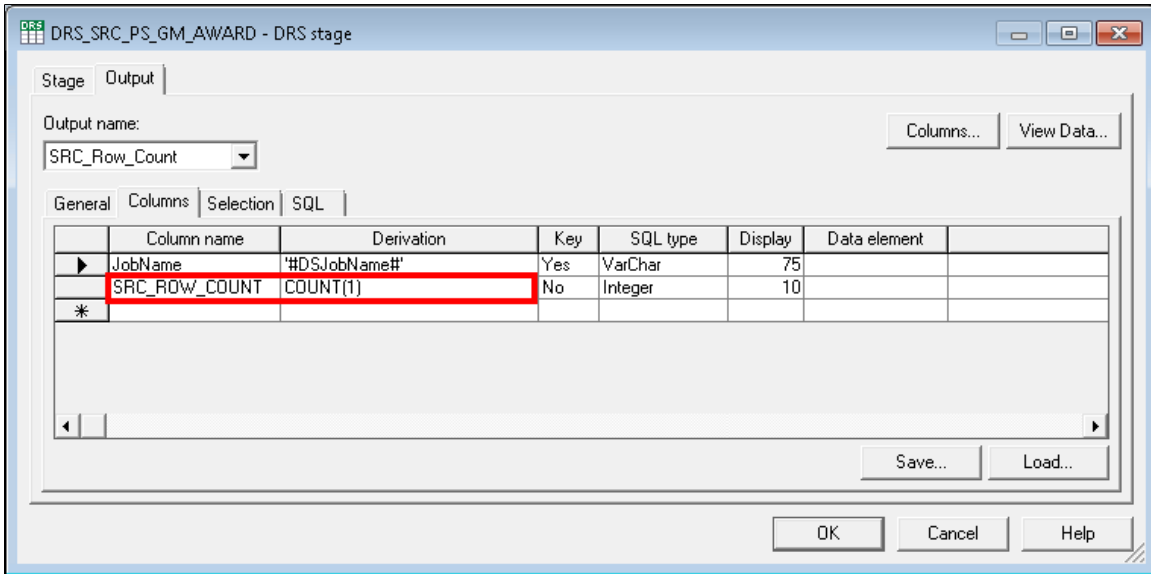
DRS source stage



The source row count value is output to the hashed file via the SRC\_Row\_Count out link:

**Image: SourceRowCount column - DRS source stage**

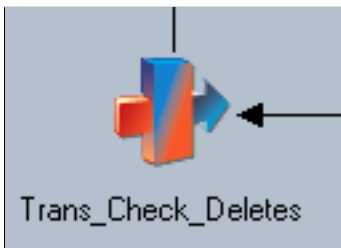
SourceRowCount column - DRS source stage



The Trans\_Check\_Deletes transformer stage contains the stage variables necessary to obtain source and target row counts:

**Image: Trans\_Check\_Deletes Transformer Stage**

Trans\_Check\_Deletes Transformer Stage



**Image: Stage Variables for Trans\_Check\_Deletes Transformer**

Stage Variables for Trans\_Check\_Deletes Transformer

Derivation	Stage Variable
Target_Count.TARGET_COUNT	TGTCOUNT
GetSourceRowCount(DSJobName)	SRCCOUNT
If Num(SRCCOUNT) And TGTCOUNT > SRCCOUNT AND P_HANDLE_DELETES = 'Y' Then 'Y' Else 'N'	svHandleDeletes
'BATCH_SID=: BATCH_SID : 'IP_CURR_DEL_FLG=: P_CURR_DEL_FLG : 'IP_CURR_PRCS_FLG=: P_CURR_PRCS_FLG :	AssignParams
If svHandleDeletes = 'Y' then UtilityRunJob(J_Stage_PS_GM_AWARD_DEL_FSCM91'EPM91'AssignParams,0,0) else -1	RunDelJob

Also note that the *svHandleDeletes* stage variable equals 'Y' when the target row count is greater than the source row count (due to source record deletes) and the constraint P\_HANDLE\_DELETES = Y.

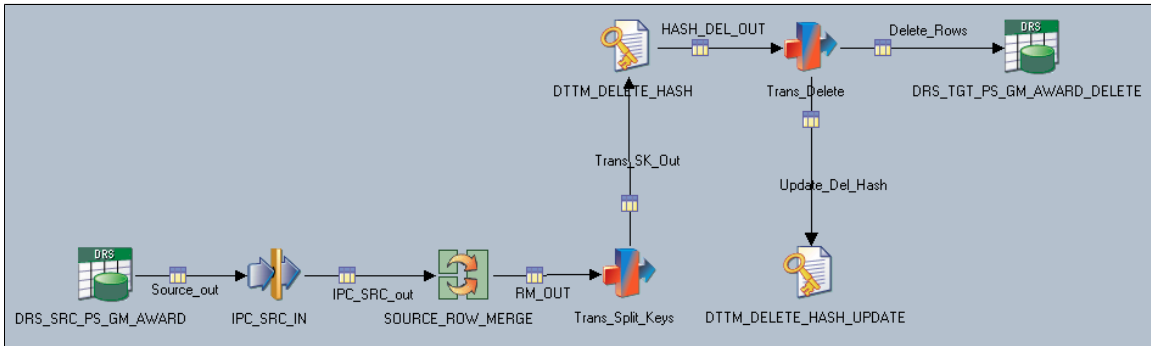
The *RunDelJob* stage variable will then call the corresponding staging delete job (using the routine *UtilityRunJob*) when the *svHandleDeletes* stage variable is set to 'Y.'

## Phase Two: Staging Delete Job

The staging delete job performs a key compare between source records and hashed table records to determine which records require deletion:

### Image: staging delete job

staging delete job



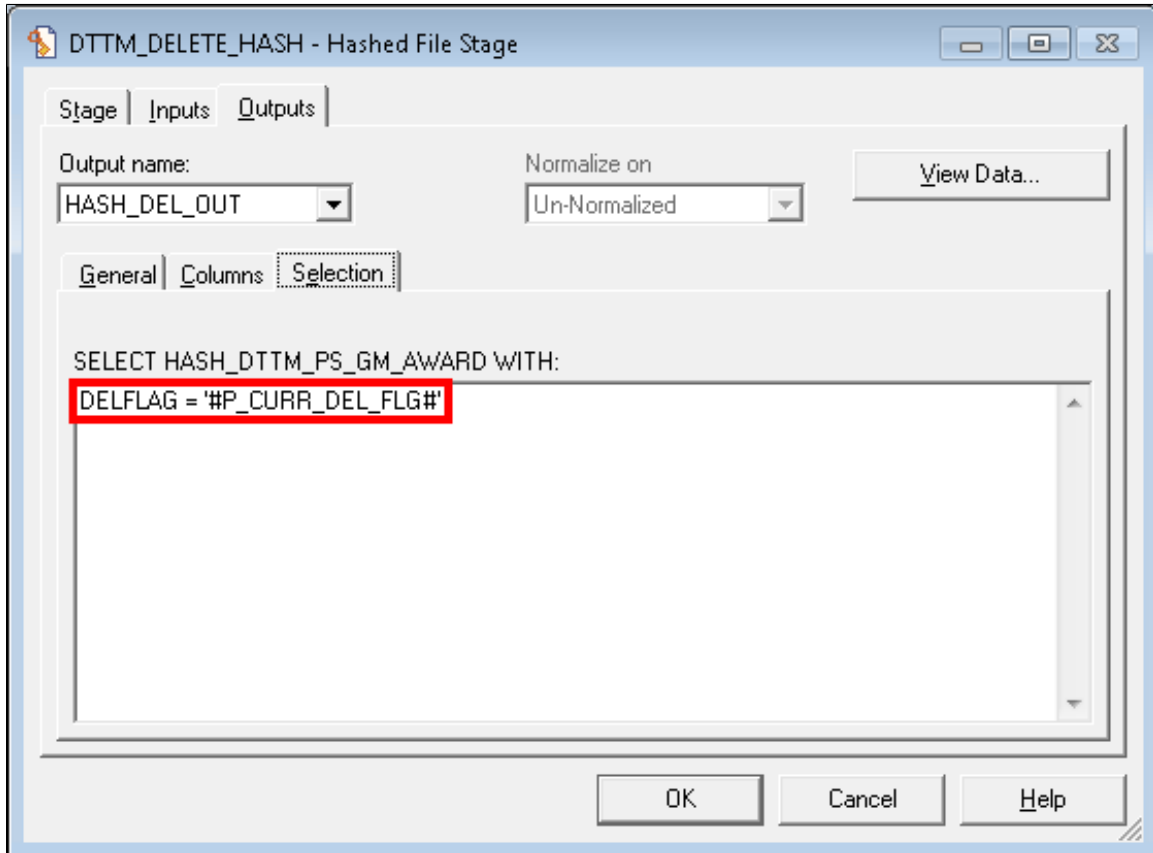
The `Trans_Split_Keys` transformer stage uses the date time key (`DTTM_KEY`), source system ID (`SRC_SYS_ID`), and delete flag (`DELFLAG`) values to update the incoming source records to the `DTTM_DELETE_HASH` hashed file.

If an incoming source record matches an existing record in the `DTTM_DELETE_HASH` hashed file, the Delete Flag (`DELFLAG`) value for that record is overwritten. When a record is deleted in the source, its corresponding record in the `DTTM_DELETE_HASH` hashed file will remain unchanged, thereby flagging it for deletion.

The following graphic shows the selection criteria in the DTTM\_DELETE\_HASH hashed file output, which selects *only* records with the DELFLAG value equal to *P\_CURR\_DEL\_FLAG*:

**Image: SelectionCriteria for DTTM\_DELETE\_HASH**

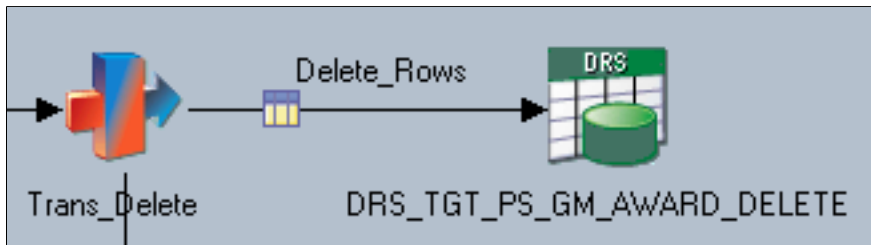
SelectionCriteria for DTTM\_DELETE\_HASH



The *Delete\_Rows* out link populates the DATA\_ORIGIN column for outbound records with the value 'D':

**Image: Trans\_Delete Transformer to DRS\_TGT\_PS\_GM\_AWARD\_DELETE**

Trans\_Delete Transformer to DRS\_TGT\_PS\_GM\_AWARD\_DELETE



**Image: Trans\_Delete Transformer Stage**

Trans\_Delete Transformer Stage

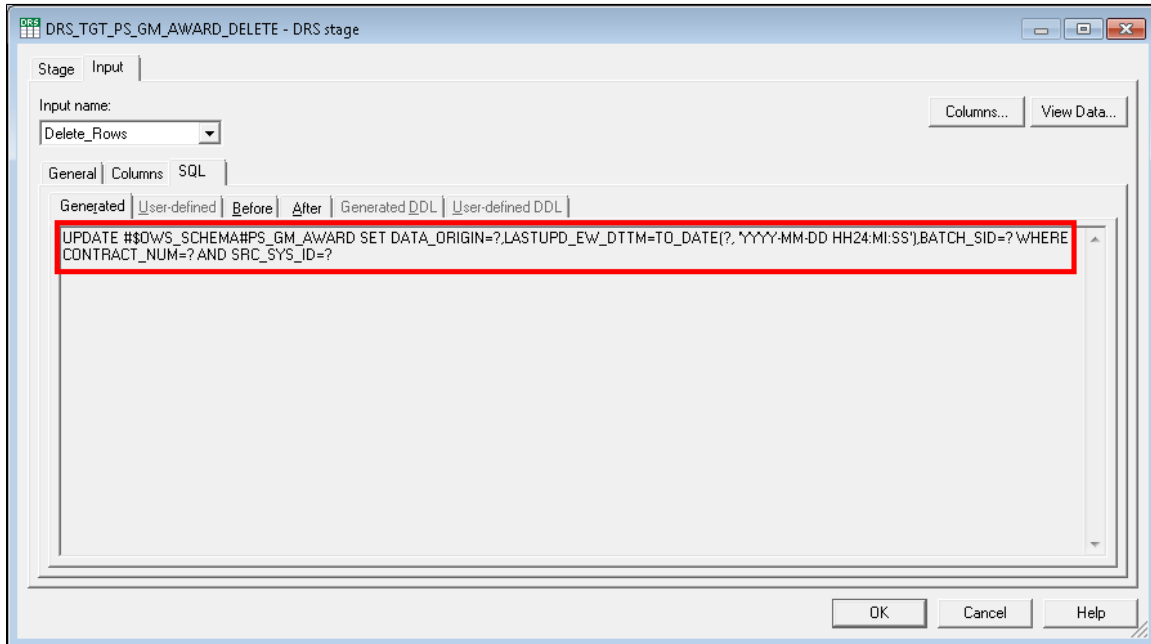
Delete_Rows	
Constraint: P_HANDLE_DELETES = 'Y'	
Derivation	Column Name
NullToCharDefault(Field(ClnsdFld,'"'),1))	CONTRACT_NUM
\$FSCM_SRC_SYS_ID	SRC_SYS_ID
'D'	DATA_ORIGIN
JobStartTimeStamp	LASTUPD_EW_DTTM
BATCH_SID	BATCH_SID



The DRS target delete stage contains a generated SQL update action that updates the DATA\_ORIGIN value to 'D.'

**Image: Generated SQL for DRS\_TGT\_PS\_GM\_AWARD\_DELETE**

Generated SQL for DRS\_TGT\_PS\_GM\_AWARD\_DELETE



## 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 following parameter file: *FMS\_HANDLEDETES\_SETUP.txt*.

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.

**Image: FMS\_HANDLEDELETES\_SETUP.txt parameter file**

FMS\_HANDLEDELETES\_SETUP.txt parameter file

```

FMS_HANDLEDELETES_SETUP - Notepad
File Edit Format View Help
-----FMS Staging Jobs
[J_Stage_PS_AGING_CATEG_TBL_FSCM91_EPM91]
HANDLEDELETES = N

[J_Stage_PS_AGING_TBL_FSCM91_EPM91]
HANDLEDELETES = N

[J_Stage_PS_AP_AGING_AGE_FSCM91_EPM91]
HANDLEDELETES = N

[J_Stage_PS_AP_AGING_CYCLE_FSCM91_EPM91]
HANDLEDELETES = N

[J_Stage_PS_AR_SPECIALIST_FSCM91_EPM91]
HANDLEDELETES = N

[J_Stage_PS_BANK_ACCT_CPTY_FSCM91_EPM91]

```

You can set the HANDLEDELETES value equal to 'Y' if you want your staging job to identify source deletes.

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

The following steps are required to implement and configure the source-delete diagnostic feature in the Campus Solutions warehouse staging jobs. These steps must be completed even if you do not plan to use the source-diagnostic feature.

1. Copy the delivered parameter file, FMS\_HANDLEDELETES\_SETUP.txt, to the same directory as the \$PARAM\_FILE\_DIR environment variable.

---

**Note:** The parameter files 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.

---

2. Open the parameter file and set the HANDLEDELETES value equal to 'Y' if you want the staging job to identify source deletes, or leave as 'N' if you do not want the staging job to identify source deletes.
3. In IBM WebSphere DataStage Designer, open the job *JC\_DeleteStrat\_SequentialRun*.
4. Click the Run button on the toolbar and in the Job Run Options window set the Source Category job parameter equal to *FMS\_SETUP* for FMS setup jobs or *FMS\_OWS* for FMS staging jobs.
5. 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 as instructed above.

If you have activated the source-delete diagnostic feature for staging jobs, this job populates delete hashed files with data from their corresponding 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 FMS\_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 FMS\_HANDLEDELETES\_SETUP.txt parameter file (as discussed in the preceding sections of this document).

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:

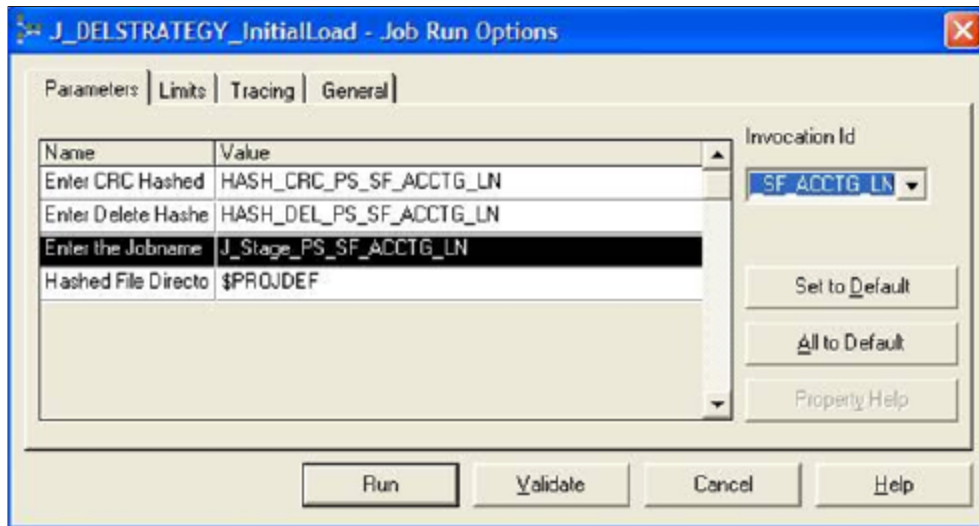
Name	Value
<CRC hashed file name>	<CRC hashed file name>  <b>Note:</b> Enter the name of the CRC hashed file associated with the CRC staging job being modified. For example, HASH_CRC_PS_SF_ACCTG_LN.

Name	Value
<Delete hashed file name>	<Delete hashed file name>  <b>Note:</b> 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>  <b>Note:</b> 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.

**Image: 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.

**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):

1. In IBM WebSphere DataStage Designer, navigate to the job *J\_DELSTRATEGY\_InitialLoad\_NonDeletes* under the ReusableJobs\_Parallel, DeleteStrategy nodes.

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

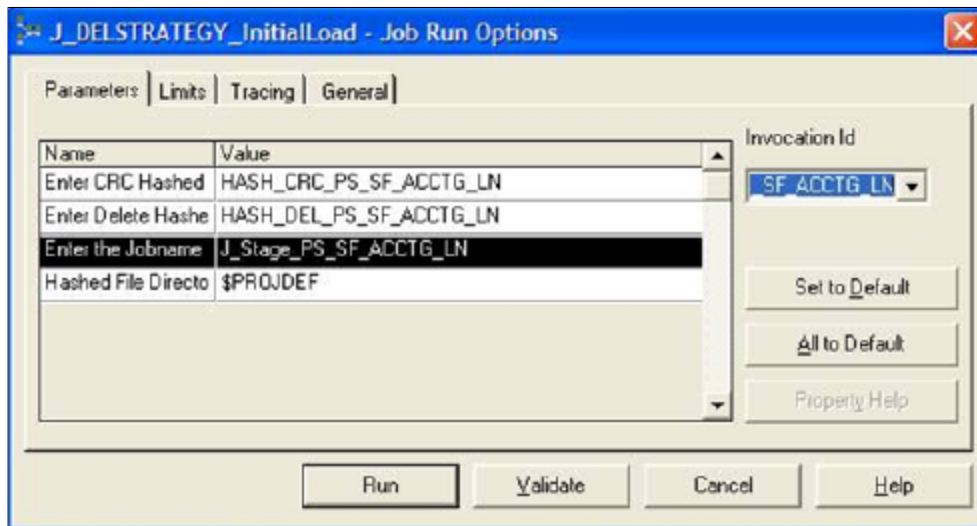
Name	Value
<CRC hashed file name>	<CRC hashed file name>  <b>Note:</b> You enter the name of the CRC hashed file associated with the CRC staging job being modified. For example, HASH_CRC_PS_SF_ACCTG_LN.
<Delete hashed file name>	<Delete hashed file name>  <b>Note:</b> You enter the name of the Delete hashed file associated with the CRC staging job being modified. For example, HASH_DEL_PS_SF_ACCTG_LN.
<Staging job name>	<Staging job name>  <b>Note:</b> 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.

**Image: Job run parameters 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.

---

## 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.

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, FSCM nodes.

For example, if you are modifying the J\_Stage\_PS\_AR\_SPECIALIST\_FSCM91\_EPM91 staging job, the job you select under the FSCM node is  
J\_Stage\_PS\_AR\_SPECIALIST\_PRV\_DEL\_FSCM91\_EPM91.

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 Audit Jobs

---

## Understanding Audit Jobs

Source table records might not have a date time stamp field. When source table records lack a date time stamp, a cyclical redundancy check (CRC) must be performed 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:

<b>Base Table (PS_CA_STATUS_TBL)</b>	<b>Audit Table (PS_AUDIT_CA_STATUS_TBL)</b>
	AUDIT_OPRID
	AUDIT_STAMP

<b>Base Table (PS_CA_STATUS_TBL)</b>	<b>Audit Table (PS_AUDIT_CA_STATUS_TBL)</b>
	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

Page Name	Definition Name	Navigation	Usage
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.

### Audit Triggers Page

Use the Audit Triggers page (TRIGAUDPNL) to create audit triggers and trigger statements.

#### Navigation

PeopleTools, Utilities, Audit, Update Database Level Auditing, Audit Triggers

#### Image: Audit Triggers page

This example illustrates the fields and controls on the Audit Triggers page. You can find definitions for the fields and controls later on this 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.

<b>Trigger Name</b>	<p>Displays the name of the audit trigger.</p> <p>The system automatically names the audit trigger using the following naming convention: [<i>base record name</i>]<sub>TR</sub>.</p>
<b>Create Trigger Statement</b>	<p>Displays the trigger statement (code).</p> <p>The code is automatically populated when you click Generate Code.</p> <p>You can edit the script as needed.</p>
<b>Generate Code</b>	<p>Click this button to generate the SQL that creates the trigger.</p> <p>Clicking this button also populates the Create Trigger Statement field with the trigger statement (code).</p>
<b>Audit Options</b>	
<b>Add</b>	Select this option if you want the audit table to track newly added records.
<b>Change</b>	Select this option if you want the audit table to track changed records.
<b>Delete</b>	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.

## Run Audit Triggers

Use the Run Audtrgs (Run Audit Triggers) page (RUN\_AUDTRGS) to run trigger scripts.

## Navigation

PeopleTools, Utilities, Audit, Perform Database Level Audit, Run Audtrgs (Run Audit Triggers)

### Image: Run Audtrgs (Run Audit Triggers) page

This example illustrates the fields and controls on the Run Audtrgs (Run Audit Triggers) page . You can find definitions for the fields and controls later on this 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.

---

# Implementing Currency Conversion for Multiple Currencies

---

## Understanding Currency Conversion

Companies spanning national boundaries often experience problems handling multiple currencies, as well as problems providing a unified view of their 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 data. For these reasons, the MDW currency conversion utility has been created.

---

**Note:** The MDW currency conversion process discussed in this documentation 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:

<i>AMOUNT</i>	<i>AMT_CD</i>	<i>RPT_AMT1</i>	<i>RPT_AMT1_CD</i>	<i>RPT_AMT2</i>	<i>RPT_AMT2_CD</i>
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:

<b>Field Type</b>	<b>Field Name</b>
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.

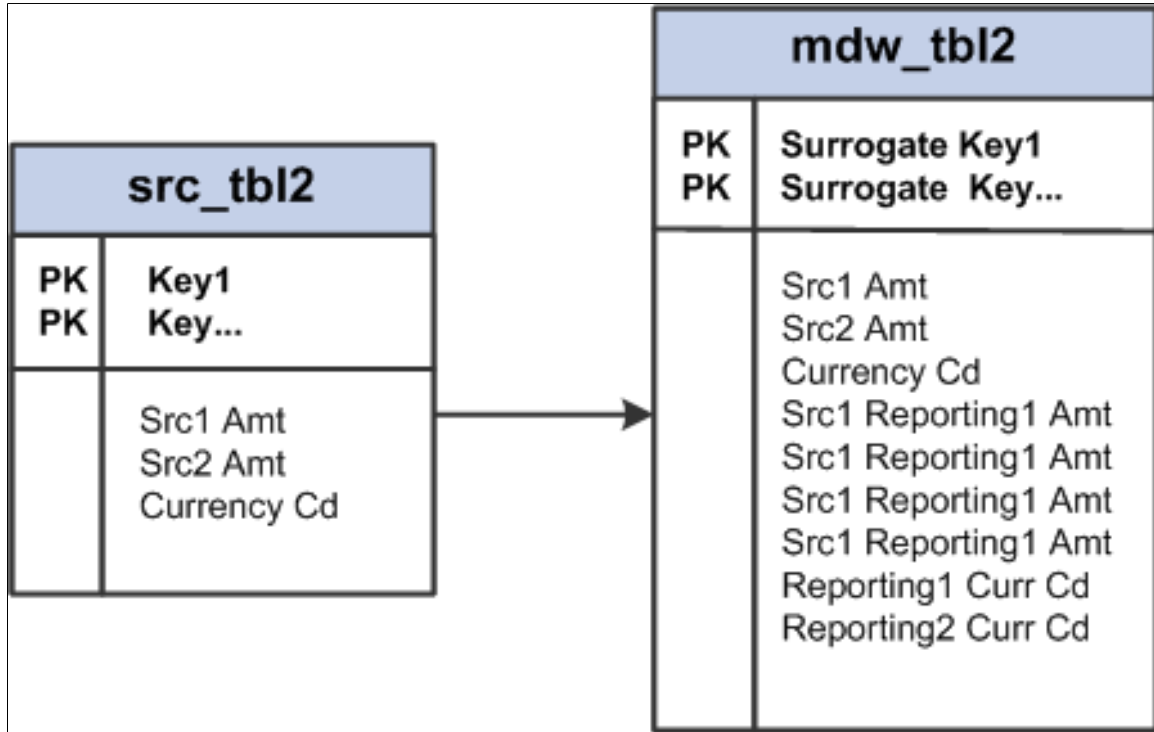
That is, base amount and base currency code fields are source database fields. If multiple base amounts from the same source tables are stored in the MDW tables, they may share the same base currency code field.

- 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.

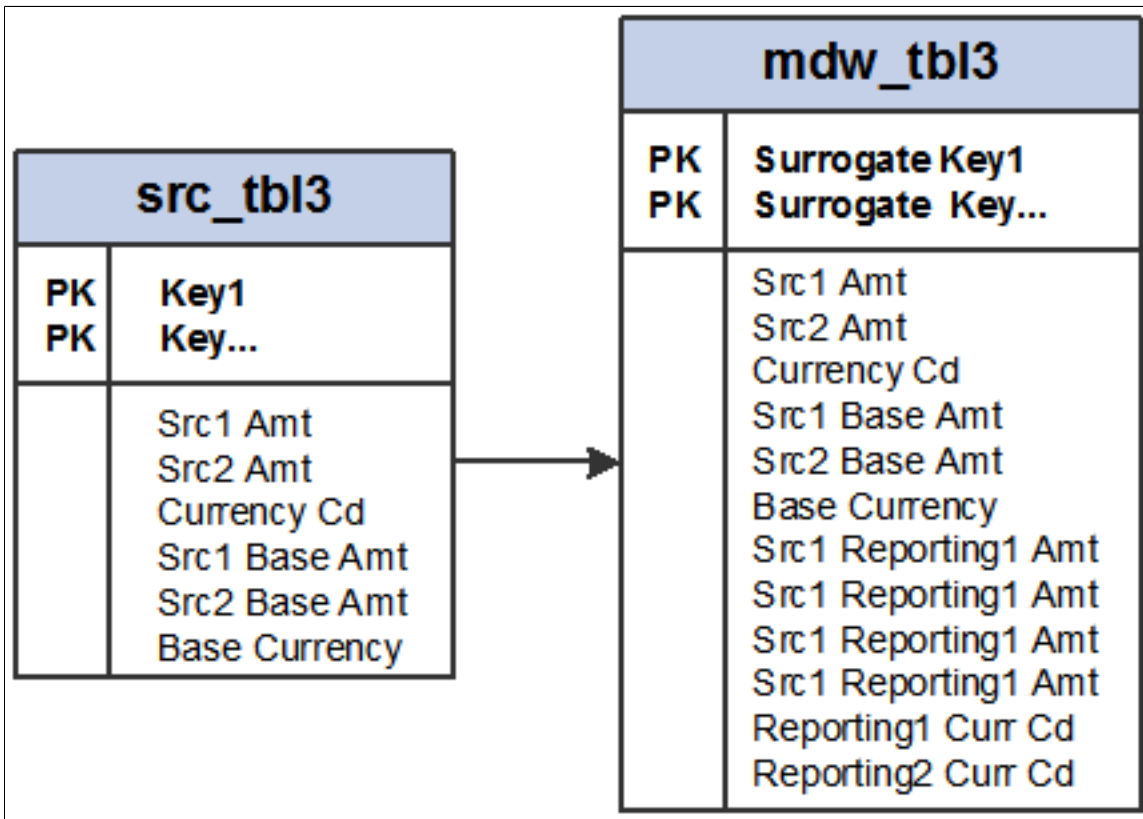
**Image: Carrying multiple source amounts into the MDW fact table**

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



**Image: Carrying multiple source and base amounts into the MDW fact table**

Carrying multiple source and base amounts into the MDW fact table

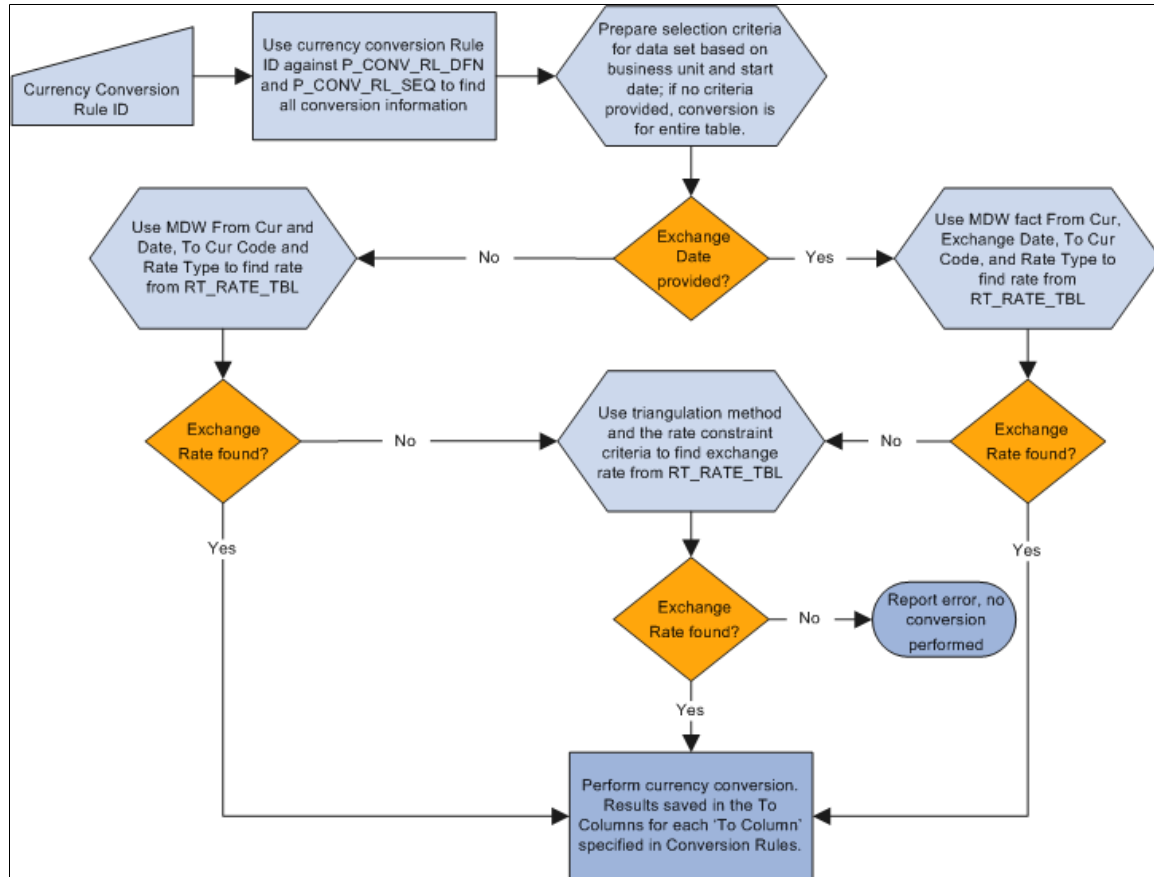




## Understanding Currency Conversion Methodology

### Image: Currency conversion methodology flow

The following diagram represents the currency conversion process:



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 PWEEK\_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.

**Image: RT\_RATE\_TBL structure**

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

**Image: CURR\_QUOTE\_TBL structure**

The CURR\_QUOTE\_TBL has the following structure:

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

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

<b>Condition</b>	<b>Result</b>
If Currency Quotation method is defined and RATE_TRIANGULATE = 'Y'	<p>then get REF_CUR field from CURR_QUOTE_TBL</p> <p>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.</p> <p>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.</p> <p>Converted Amount = Source Amount * (RATE_MULT of leg 1 * RATE_MULT of leg 2) / (RATE_DIV of leg 1 * RATE_DIV of leg 2).</p>
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.

**Image: Schema Definition page for F\_AP\_TRAN fact table**

The following example shows the schema definitions for the F\_AP\_TRAN fact table:

The screenshot shows the 'Schema Definition' page for the F\_AP\_TRAN fact table. It includes a navigation bar with 'Schema Definition', 'Schema Source Columns', and 'Schema Target Columns'. The main content area is titled 'Schema Definition' and shows the following configuration:

Record (Table) Name	F_AP_TRAN
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

**Image: Schema Source Columns page for F\_AP\_TRAN fact table**

Schema Source Columns page for F\_AP\_TRAN fact table

The screenshot shows the 'Schema Source Columns' page for the F\_AP\_TRAN fact table. It includes a navigation bar with 'Schema Definition', 'Schema Source Columns', and 'Schema Target Columns'. The main content area is titled 'Schema Source Columns' and shows the following configuration:

Record (Table) Name	F_AP_TRAN		
Schema Source Columns	Customize   Find   View All	First	1-2 of 11
			Last
	<b>*From Amount Column</b>	<b>*From Currency Column</b>	
1	ADJ_BASE_AMT	BASE_CRNCY_CD	+ -
2	DILS_BASE_AMT	BASE_CRNCY_CD	+ -



**Image: Schema Target Columns page for F\_AP\_TRAN fact table**

Schema Target Columns page for F\_AP\_TRAN fact table

Schema Target Columns					
Record (Table) Name			F_AP_TRAN		
Schema Target Columns			Customize   Find   View All   First 1-2 of 22 Last		
	*To Amount Column	*To Currency Column	*From Amount Column		
1	ADJ_R1_AMT	R1_CRNCY_CD	ADJ_BASE_AMT	+	-
2	ADJ_R2_AMT	R2_CRNCY_CD	ADJ_BASE_AMT	+	-

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 [Setting Up Parameters for Tree and Recursive Hierarchy Processing](#).

## 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.

**Image: MDW Currency Conversion Rule page for AP\_TRANS fact table**

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

The screenshot shows the 'MDW Currency Conversion Rule' page for the 'AP\_TRANS' fact table. The page is divided into several sections:

- MDW Currency Conversion Rule**: Shows the rule name 'AP\_TRANS' and a description 'AP Transaction'.
- Conversion Rule Definition**: Includes fields for '\*Rate Type' (AVG), '\*To Currency Code' (USD), and 'Specify conversion date' (Recent dt.).
- Chunking Rule Definition**: Includes fields for 'Warehouse BU', 'Start Date', and 'End Date'.
- Notes**: A text area for additional information.

**Image: MDW Conversion Schema Rule page for AP\_TRANS fact table**

MDW Conversion Schema Rule page for AP\_TRANS fact table

The screenshot shows the 'MDW Conversion Schema Rule' page for the 'AP\_TRANS' fact table. The page displays a table with the following columns:

*Record (Table) Name	*To Amount Column	From Amount Column
1 F_AP_TRAN	ADJ_R1_AMT	ADJ_BASE_AMT

The table also includes a 'Customize | Find | View All' menu and a 'First 1 of 11 Last' navigation bar.

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.
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.

### Schema Definition Page

Use the Schema Definition page (CCU\_SCHEMA\_DEFN) to set up the schema definition.

**Navigation**

EPM Foundation, EPM Setup, Common Definitions, MDW Currency Conversions, Schema Definition

**Image: Schema Definition page**

This example illustrates the fields and controls on the Schema Definition page. You can find definitions for the fields and controls later on this page.

- Business Unit Column** Enter the name of the PF (performance) business unit column.
- MDW Fact Exchange Dt Column (MDW fact exchange date column)** Enter the date column used to determine the exchange date or conversion date.
- Date/Period Dimension Record** Enter the level (year, month, day, and so on) for the Fact Exchange Date column.

**MDW Currency Conversion Rule page**

Use the MDW Currency Conversion Rule page (CCU\_CONV\_DEFN) to set up currency rule definition and chunking rule definition.

## Navigation

EPM Foundation, EPM Setup, Common Definitions, MDW Currency Conversions, MDW Currency Conversion Rules

### Image: MDW Currency Conversion Rule page

This example illustrates the fields and controls on the MDW Currency Conversion Rule page. You can find definitions for the fields and controls later on this 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.

*Exchng 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.

## MDW Conversion Schema Rule Page

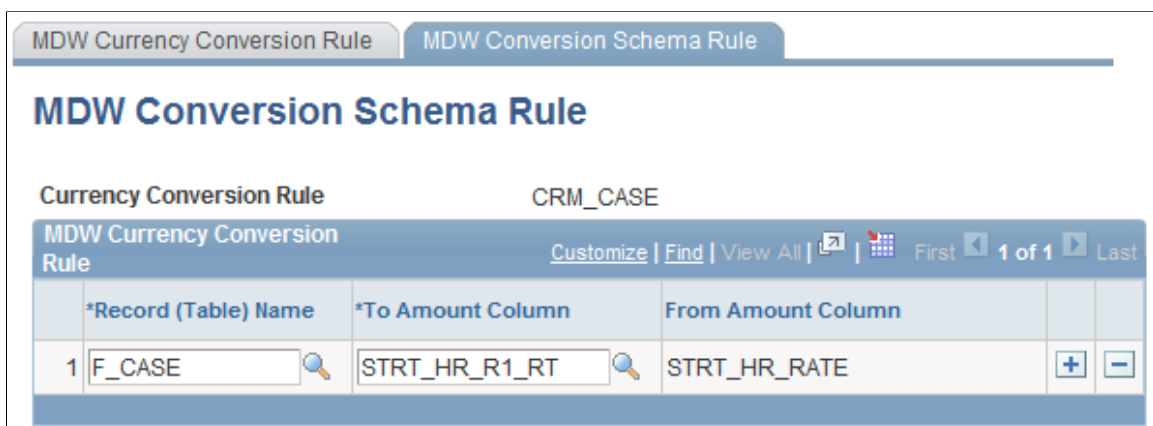
Use the MDW Conversion Schema Rule page (CCU\_CONV\_SEQ) to set up conversion schema rule.

**Navigation**

EPM Foundation, EPM Setup, Common Definitions, MDW Currency Conversions, MDW Conversion Schema Rules

**Image: MDW Conversion Schema Rule page**

This example illustrates the fields and controls on the MDW Conversion Schema Rule page. You can find definitions for the fields and controls later on this 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: *Jobs, EPM\_Utilities, Currency\_Conversion, Init\_Process, 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.

"Understanding ETL in EPM (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)"

# Setting Up Multilanguage Processing and Running 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 "Specifying Your EPM Sources (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)".

- 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.

---

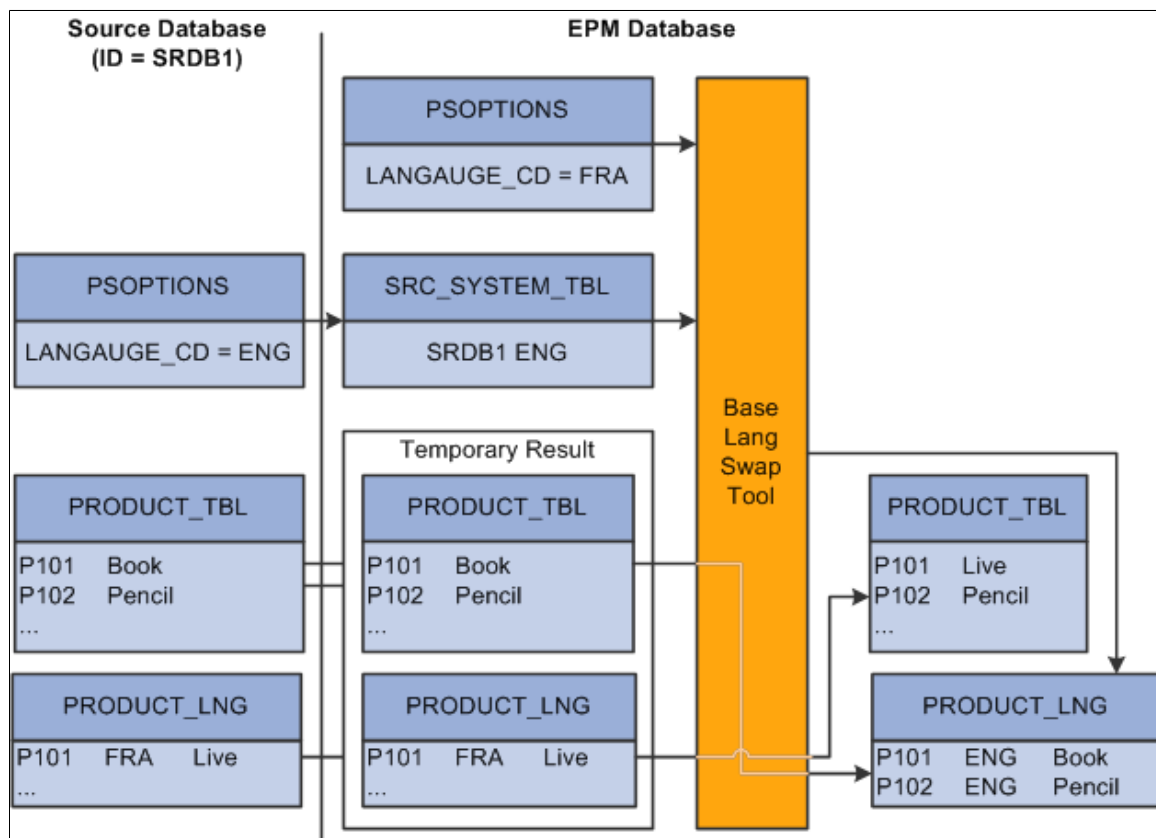
## Understanding the Language Swap Utility

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.

### Image: Language swap process flow

This example illustrates the fields and controls on the Language swap process flow. You can find definitions for the fields and controls later on this page.



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:

<b>ProductID</b>	<b>Description</b>
P101	Libro
P102	Lápiz
P103	Pluma

Here is the Product Language table from the source database:

<b>ProductID</b>	<b>Lang CD</b>	<b>Description</b>
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:

<b>ProductID</b>	<b>Description</b>
P101	Book
P102	Pencil
P103	Pluma

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

<b>ProductID</b>	<b>Lang CD</b>	<b>Description</b>
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:

<b>Time Key</b>	<b>Product Key</b>	<b>Store Key</b>	<b>Quantity</b>	<b>Amount</b>
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:

<b>Product Key</b>	<b>SKU</b>	<b>Description</b>
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):

<b>Product Key</b>	<b>Language Code</b>	<b>Description</b>
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: *Jobs, EPM\_Uilities, Language\_Swap, 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.



# Processing Trees and Recursive Hierarchies

---

## 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.

---

## Related Links

PeopleSoft 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.

<b>Subject</b>	<b>MDW Tree Denormalizer</b>	<b>OWE Tree Flattener</b>
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:

<b><i>OWS Table Name</i></b>	<b><i>Source Table in Source Database</i></b>	<b><i>Description</i></b>
PS_S_TREESTRUCT	PSTREESTRUCT	Tree Structure table
PS_S_TREEDFN	PSTREEDFN	Tree Definition table
PS_S_TREENODE	PSTREENODE	Tree Node table
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:

<b>Table Name</b>	<b>Description</b>
PSTREESTRUCT	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>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

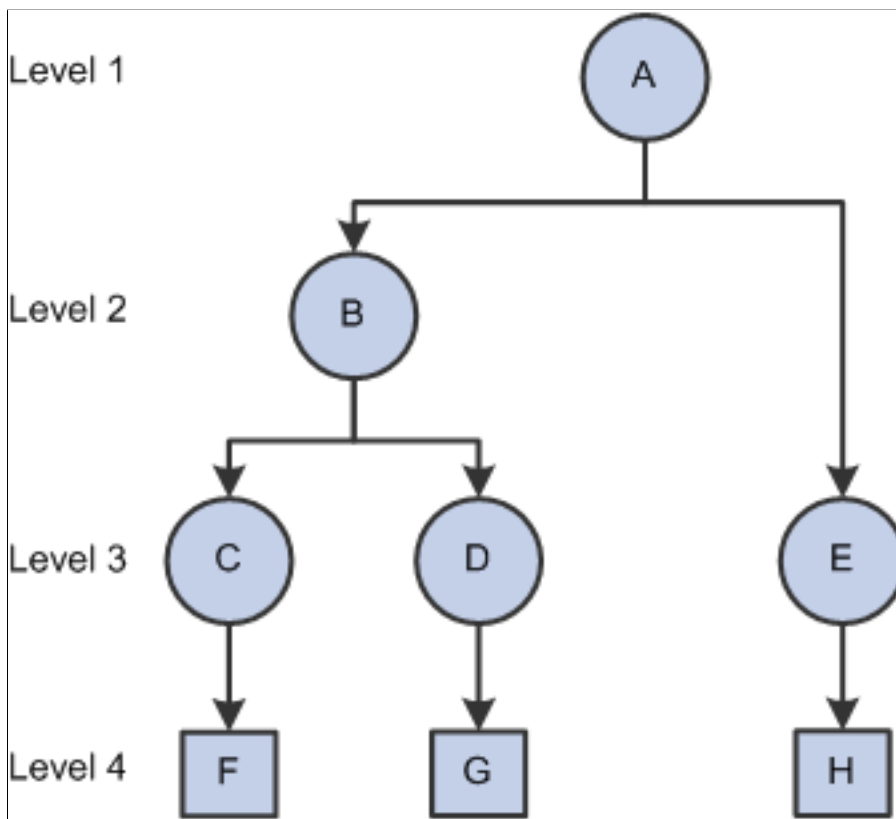
indicate processing of skip levels. The special mark for nodes that result from skip level processing is applied to the description field. The special mark is `<dds>`, where `<dd>` is the two-digit level number and `<s>` is the special character that you enter in the Skip Level Infix field on the Hierarchy 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

**Image: Skip level summer tree**

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



The following table represents the tree flattener result:

<i>Tree Node</i>	<i>Entity ID</i>	<i>Parent Level Number</i>	<i>Child Level Number</i>
A	F	1	0
A	G	1	0
A	H	1	0
B	F	2	0



<i>Tree Node</i>	<i>Entity ID</i>	<i>Parent Level Number</i>	<i>Child Level Number</i>
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:

-	<i>Row 1</i>	<i>Row 2</i>	<i>Row 3</i>
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.

---

## Tree and Recursive Hierarchy Source Tables

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 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.

## 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.

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
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_NOD_DTL	Char	30	K	Y	N	N	None
TREE_NODE_DESCR	Char	50	-	N	N	N	None
NODE_DET_FLAG	Char	1	-	N	XLAT	N	None
RANGE_FROM	Char	30	-	N	N	N	None
RANGE_TO	Char	50	-	N	N	N	None
TREE_LEVEL_NUM	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

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
BUSINESS_UNIT	Char	5	K	Y	N	N	None

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
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
TREE_NODE	Char	30	K	Y	N	N	None
CHILD_NOD _DTL	Char	30	K	Y	N	N	None
TREE_NODE _DESCR	Char	50	-	N	N	N	None
NODE_DET_ FLAG	Char	1	-	N	XLAT	N	None
RANGE_ FROM	Char	30	-	N	N	N	None
RANGE_TO	Char	30	-	N	N	N	None
TREE_ LEVEL_ NUM	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

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
SETCNTRLVA LUE	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

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
TREE_NODE	Char	30	K	Y	N	N	None
CHILD_NOD _DTL	Char	30	K	Y	N	N	None
TREE_NODE _DESCR	Char	50	-	N	N	N	None
NODE_DET_ FLAG	Char	1	-	N	XLAT	N	None
RANGE_ FROM	Char	30	-	N	N	N	None
RANGE_TO	Char	30	-	N	N	N	None
TREE_ LEVEL_ NUM	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 table represents an output relationship table for *SetID* based *recursive hierarchy* trees:

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
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_NOD _DTL	Char	30	K	Y	N	N	None
NODE_ DESCR	Char	50	-	N	N	N	None

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
NODE_DET_FLAG	Char	1	-	N	XLAT	N	None
LEVEL_NUM	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:

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
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
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_DESCR	Char	50	-	N	N	N	None
NODE_DET_FLAG	Char	1	-	N	XLAT	N	None
LEVEL_NUM	Number	3	-	N	N	N	None
CHILD_LEVEL_NUM	Number	3	-	N	N	N	None

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
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:

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
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_NOD_DTL	Char	30	K	Y	N	N	None
NODE_DESCR	Char	50	-	N	N	N	None
NODE_DET_FLAG	Char	1	-	N	XLAT	N	None
LEVEL_NUM	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

## 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:

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
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
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:

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
BUSINESS_UNIT	Char	5	K	Y	N	N	None

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
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:

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
SETCNTRLVAL	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
...	-	-	-	-	-	-	-

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
DATA_SRC_SYS_ID	Char	5	-	Y	N	N	None

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

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
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
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 *recursive hierarchy* trees:

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
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
EFFDT	Date	10	K	Y	N	N	None

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
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 *non business unit* and *non SetID* based recursive hierarchy trees:

<b>Field Name</b>	<b>Type</b>	<b>Length</b>	<b>Key</b>	<b>Required</b>	<b>Edit</b>	<b>Prompt</b>	<b>Default</b>
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
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.

## 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

#### Image: Example of a summer tree before processing

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



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

<i>Tree Node</i>	<i>Entity ID</i>	<i>Parent Level Number</i>	<i>Child Level Number</i>
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
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.

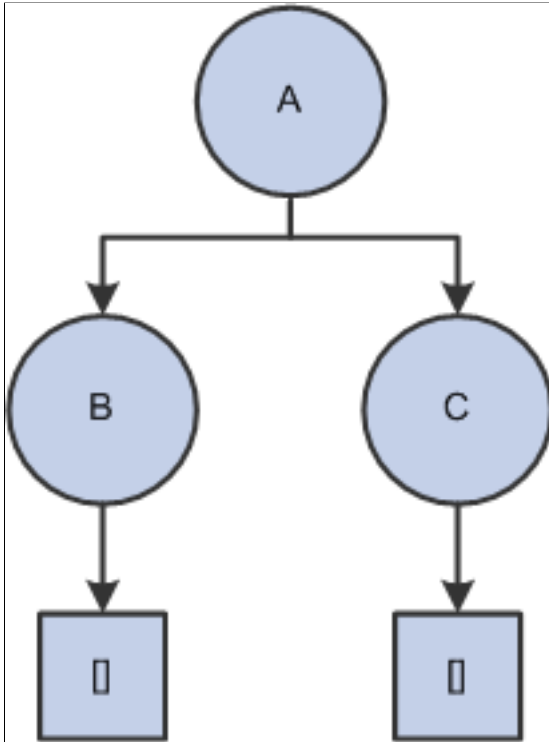
If your relationship or hierarchy tables require multilanguage support, then you must create the outrigger tables for the relationship and hierarchy tables.

---

### Dynamic Summer Tree

Image: Dynamic summer tree

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



The following table represents the Department database table:

<i>Department ID</i>	<i>Department Name</i>
A	HR
B	Benefit
C	Payroll

The following table represents the Employee table:

<i>Employee ID</i>	<i>Department ID</i>	<i>Employee Name</i>
D	B	Jane Doe
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

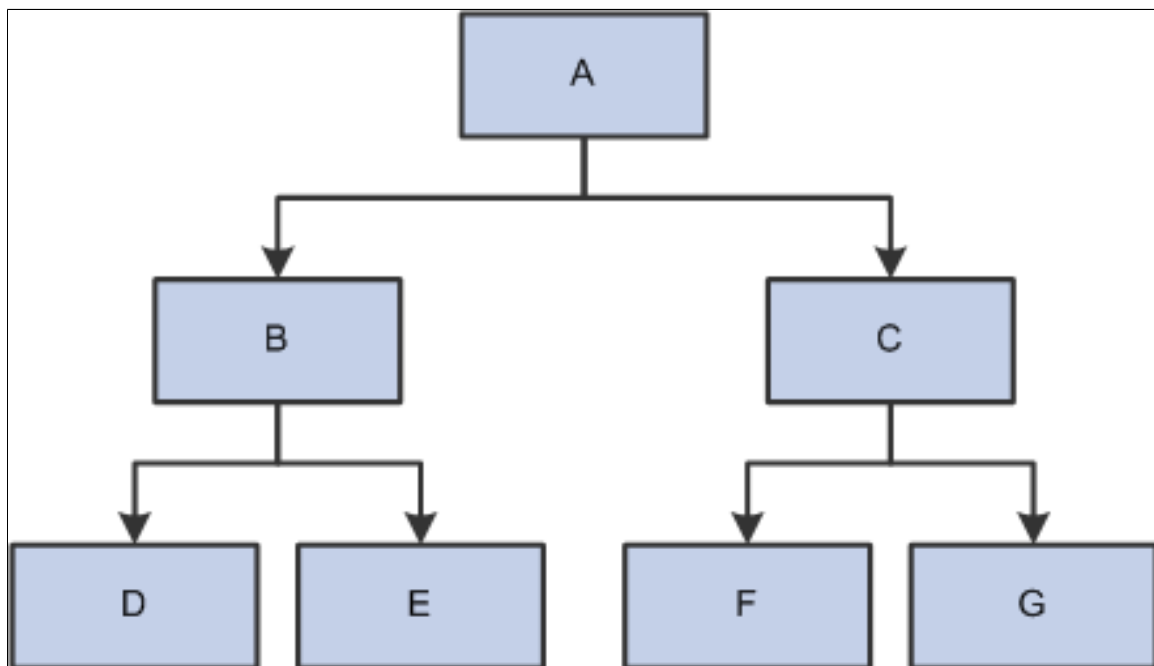
The following table shows the hierarchy table, without balancing:

<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

**Image: Example of a winter tree before processing**

This graphic shows an 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
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

<b>Entity</b>	<b>L31</b>	<b>L30</b>	<b>...</b>	<b>L3</b>	<b>L2</b>	<b>L1</b>
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:

<b>Entity ID</b>	<b>Entity Parent ID</b>
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.

<b>Parent Node</b>	<b>Entity</b>	<b>Parent Level Number</b>	<b>Child Level Number</b>
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

<i>Parent Node</i>	<i>Entity</i>	<i>Parent Level Number</i>	<i>Child Level Number</i>
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.

## Pages Used to Run the Tree and Recursive Hierarchy Process

<i>Page Name</i>	<i>Definition Name</i>	<i>Navigation</i>	<i>Usage</i>
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.

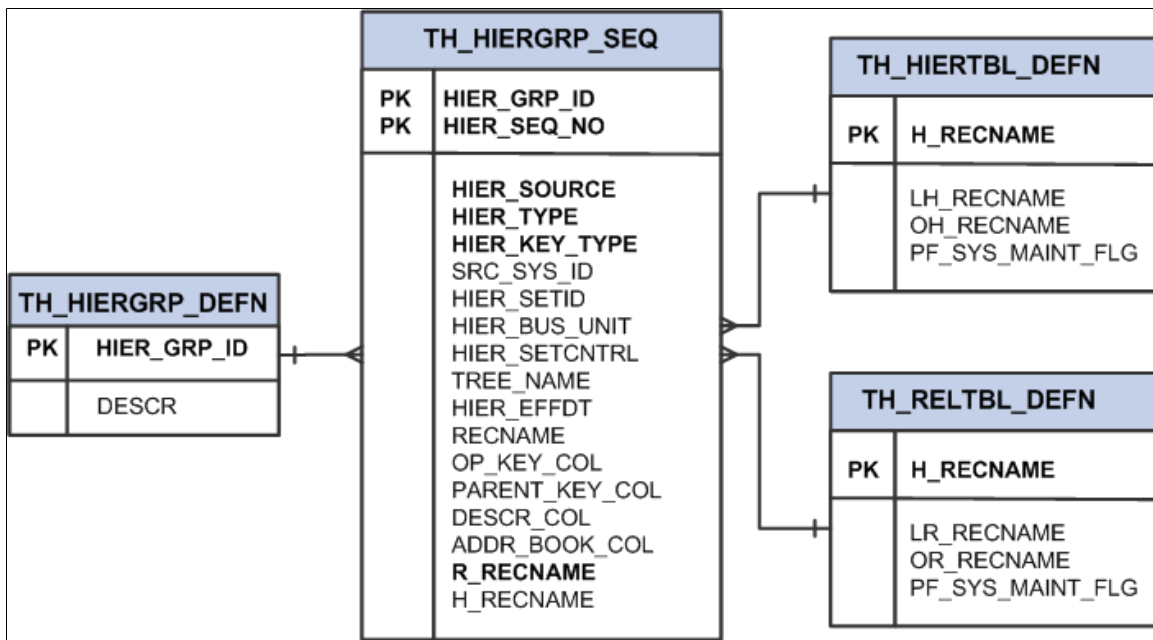
## Defining Parameters for the Tree and Recursive Hierarchy Process

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.

### Image: Tree and recursive hierarchy processing setup pages

The following diagram shows the Tree and Recursive Hierarchy processing setup pages:



## Relationship Record Definition

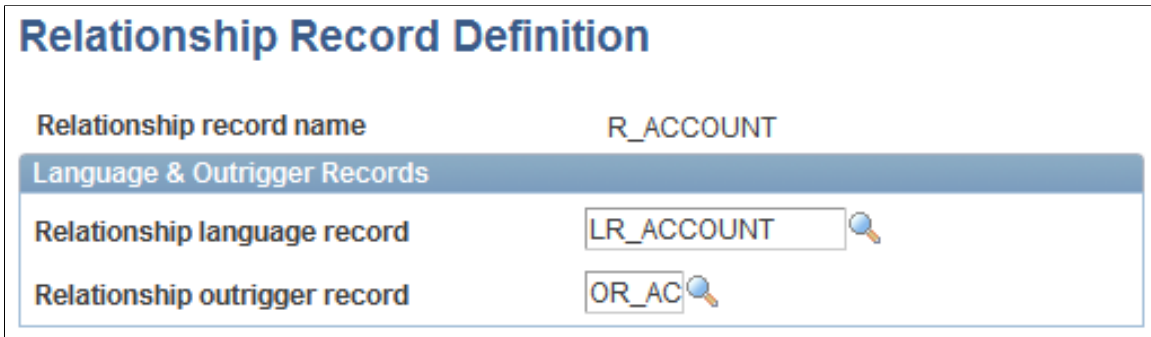
Use the Relationship Record Definition page (TH\_RELTBL\_DEFN) to define the target and language tables for tree flattening.

**Navigation**

EPM Foundation, EPM Setup, Common Definitions, Hierarchy Group Definition, Relationship Table Definition

**Image: Relationship Record Definition page**

This example illustrates the fields and controls on the Relationship Record Definition page . You can find definitions for the fields and controls later on this 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.

**Hierarchy Record Definition Page**



Use the Hierarchy Record Definition page (TH\_HIERTBL\_DEFN) to define the target and language tables for tree and hierarchy denormalizing.

## Navigation

EPM Foundation, EPM Setup, Common Definitions, Hierarchy Group Definition, Hierarchy Table Definition

### Image: Hierarchy Record Definition page

This example illustrates the fields and controls on the Hierarchy Record Definition page. You can find definitions for the fields and controls later on this page.

Hierarchy Record Definition	
Hierarchy record name	H_COMPCD
Language & Outtrigger Records	
Hierarchy language record	LH_COMPCD 
Hierarchy outtrigger record	OH_COMPCD 

<b>Hierarchy record name</b>	Displays the target record for the denormalizing portion of the Tree and Recursive Hierarchy processing.
<b>Hierarchy language record</b>	Enter the language record for this hierarchy table. You must enter this value only if this table is used for multilanguage processing.
<b>Hierarchy outtrigger record</b>	Enter the outtrigger record for this hierarchy table. You must enter this value only if this table is used for multilanguage processing.

## Hierarchy Group Definition Page

Use the Hierarchy Group Definition page (TH\_HIERGRP\_DEFN) to enter parameters for the Tree and Recursive Hierarchy process.

## Navigation

EPM Foundation, EPM Setup, Common Definitions, Hierarchy Group Definition, Hierarchy Group Definition

### Image: Hierarchy Group Definition page

This example illustrates the fields and controls on the Hierarchy Group Definition page . You can find definitions for the fields and controls later on this page.

**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

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:

- *Up Balancing*
- *Down Balancing*
- *No Balancing*

**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 *Up Balancing* or *Down Balancing* 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 *Up Balancing* or *Down Balancing* 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:

*Current EPM Database* (for OWE).

*Non Current EPM Database* (for OWS). This value refers to the PeopleSoft source database, as exists on the OWS.

**Hierarchy Type**

Select the type of hierarchy.

The options are:

*Tree hierarchy.* When this option is selected, the Hierarchy effective date field displays.

*Recursive hierarchy.*

**Hierarchy Key Type**

Select the additional key type for this hierarchy. Options are:

- *Business Unit based*
- *No Additional Key Defined*
- *SetID based*
- *User defined*

**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 *SetID*.



<b>Hierarchy Business Unit</b>	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> .
<b>User Defined Value</b>	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> .
<b>Record (Table) Name</b>	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> .
<b>Tree Name</b>	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> .
<b>Hierarchy effective date</b>	Enter the effective date for the tree in Tree Manager.
<b>Operational Key Column</b>	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> .
<b>Parent Key Column</b>	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> .
<b>Description Column</b>	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> .
<b>Source System Identification</b>	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 hash file hierarchy jobs.
2. Running OWS 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: *Jobs, EPM\_Utilities, 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: *Jobs, EPM\_Utilities, Tree\_Recursive\_Hierarchy, Trees, ESourceTrees, StagingTreeMetadata, FSCM, SEQ\_J\_Stage\_[table name]*.

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: *Jobs, EPM\_Utilities, Tree\_Recursive\_Hierarchy, Init\_Process, J\_Hierarchy\_Startup\_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*.



# Extending the Multidimensional Warehouse 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

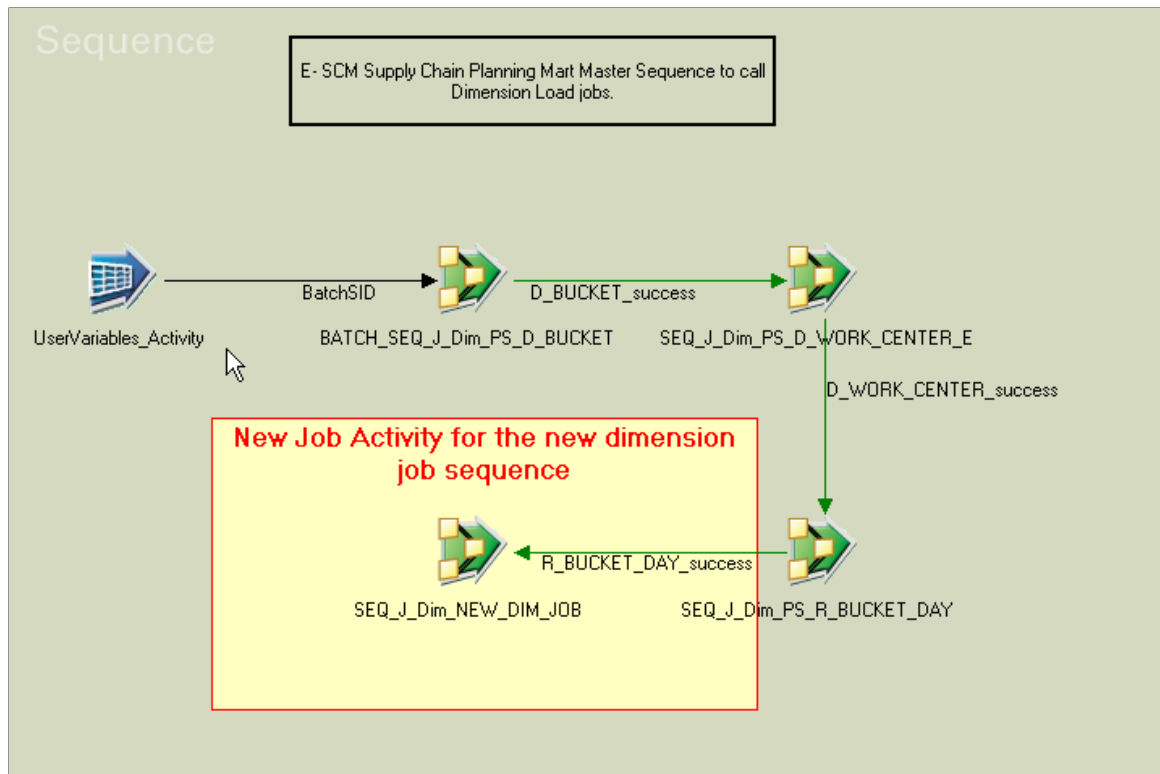
Perform the following steps to add a new fact or dimension job to a master sequence:

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.

**Image: Add new job to master sequence**

This example illustrates the fields and controls on the Add new job to master sequence. You can find definitions for the fields and controls later on this page.



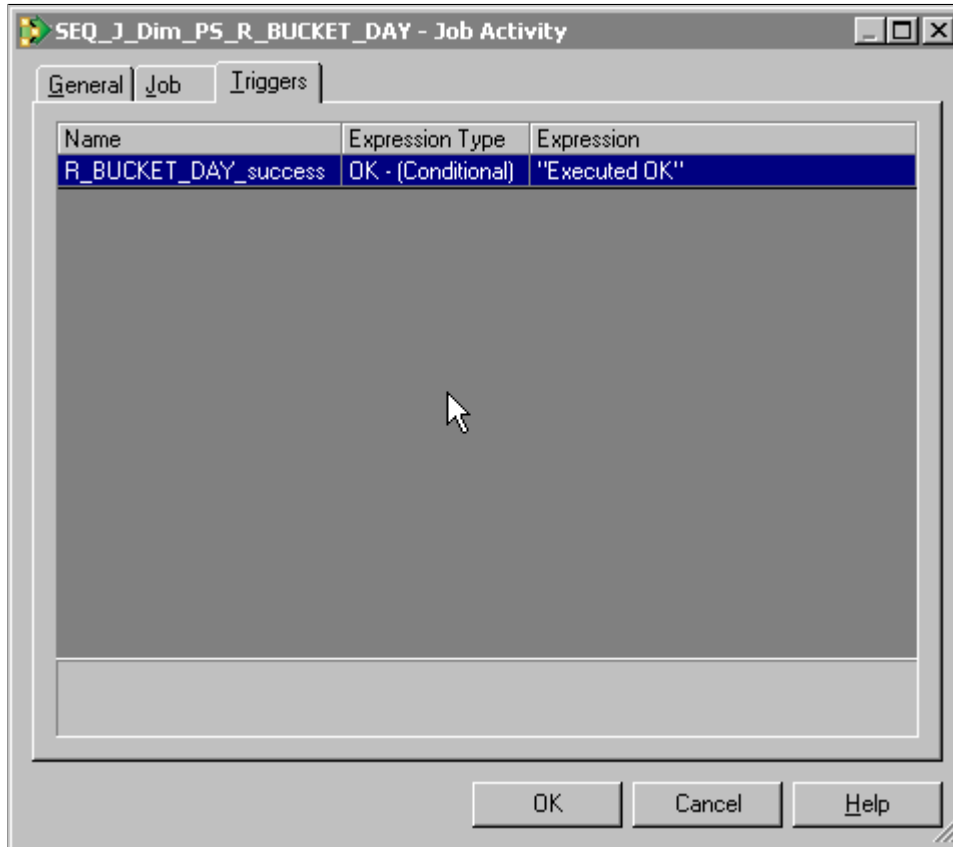
4. Connect the new job activity to the existing activity using a *trigger*.



- In the *Triggers* tab of the properties box, edit the expression of the output trigger as appropriate.

**Image: Job Activity box with Triggers tab selected**

This example illustrates the fields and controls on the Job Activity box with Triggers tab selected. You can find definitions for the fields and controls later on this page.



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.

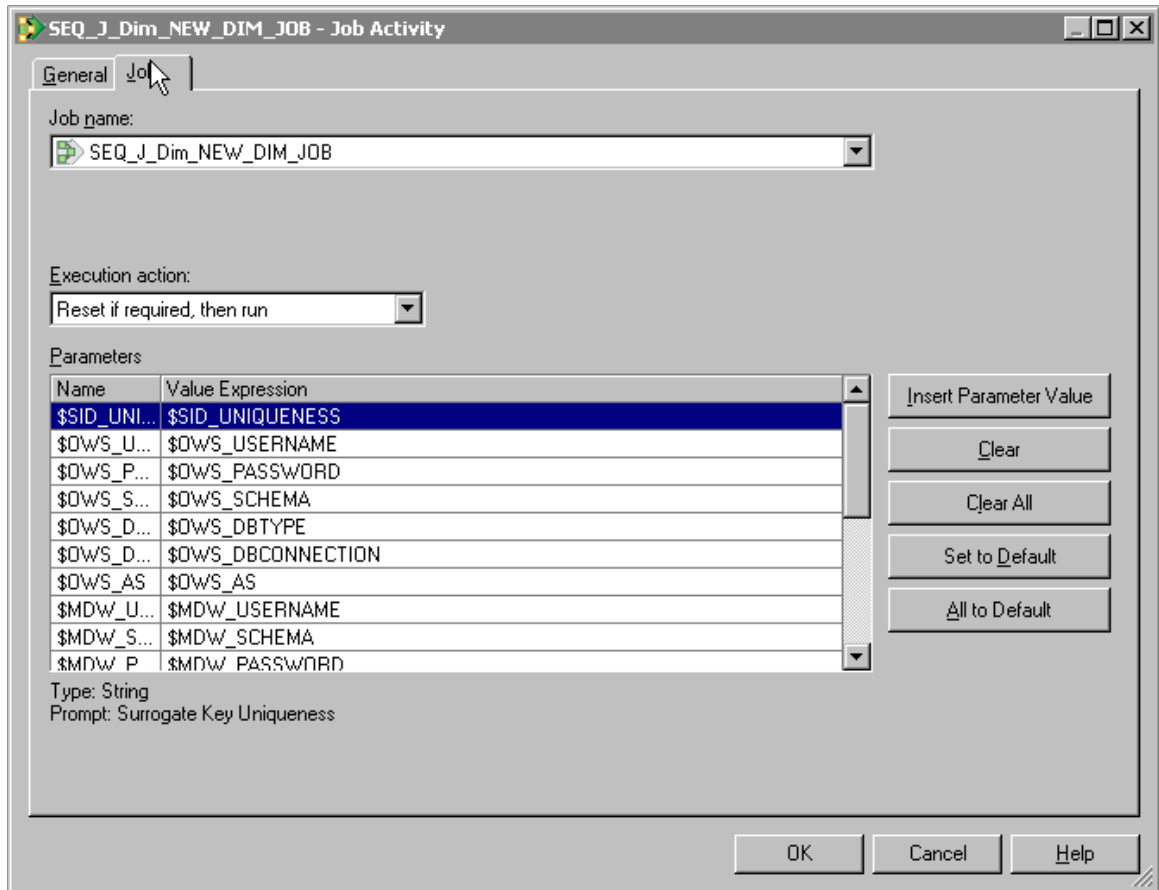
- Open the new job activity for editing.

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.

**Image: Job Activity box with Job tab selected**

This example illustrates the fields and controls on the Job Activity box with Job tab selected. You can find definitions for the fields and controls later on this page.



- 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 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

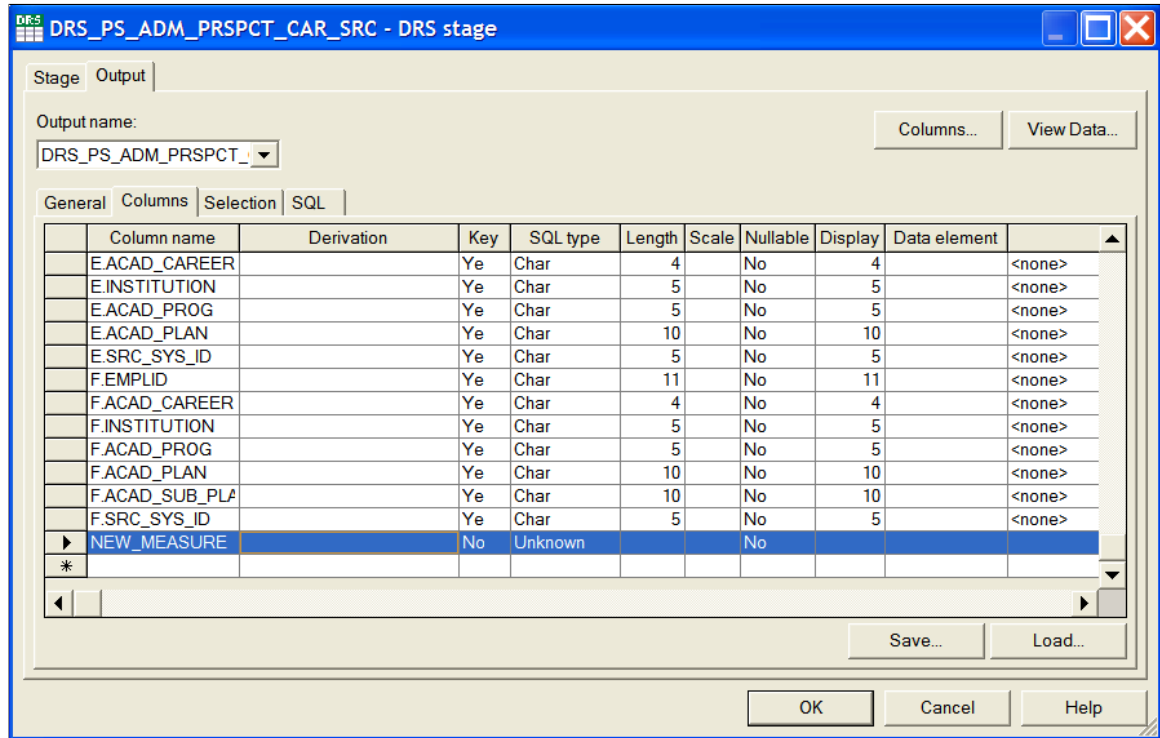
Perform the following steps to update a fact job with a new measure that originates from the same source table:

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.

**Image: Adding a new row for the new measure**

This example illustrates the fields and controls on the Adding a new row for the new measure. You can find definitions for the fields and controls later on this page.



- 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.

### Image: General sub-tab with new source table information

This example illustrates the fields and controls on the General sub-tab with new source table information. You can find definitions for the fields and controls later on this page.

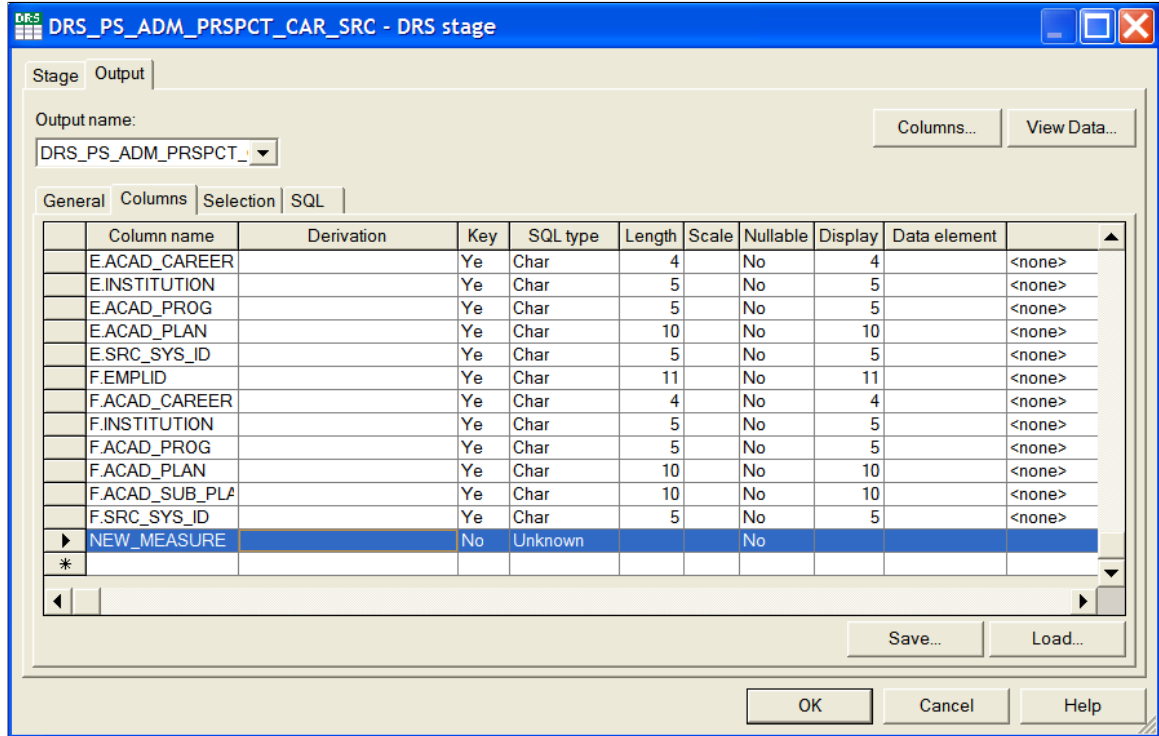
The screenshot shows the 'Output' tab in the IBM WebSphere DataStage Designer. The 'Output name' is set to 'DRS\_PS\_EN\_WORK\_Cf'. The 'General' sub-tab is selected, showing the following configuration:

- Table names:** NTER A, #DWS#SCHEMA#PS\_NEW\_SOURCE\_TBL B
- Transaction Isolation:** Read Committed
- Array size:** #DWS\_AS#
- Query type:** Generated SQL query
- Description:** (Empty text area)

4. Select the Columns sub-tab and add a new row for the new measure.

**Image: Adding a new row for the new measure**

This example illustrates the fields and controls on the Adding a new row for the new measure. You can find definitions for the fields and controls later on this page.



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

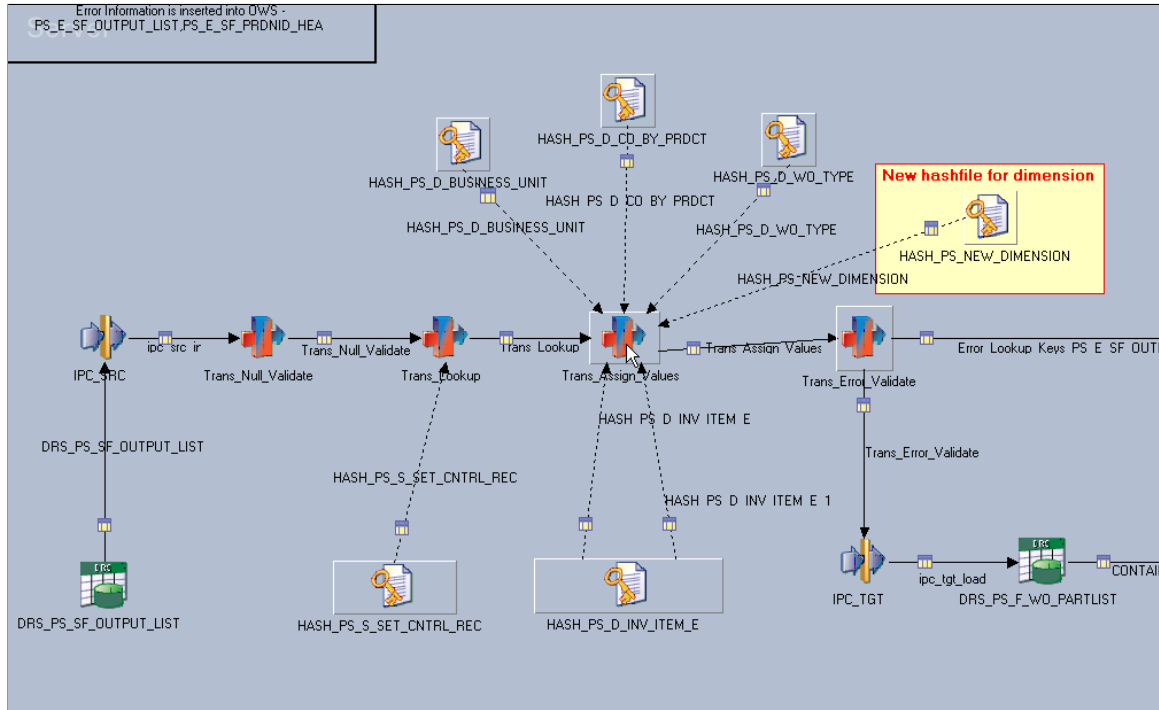
Perform the following steps to add a new surrogate key:

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.

**Image: Dimension table hash file added**

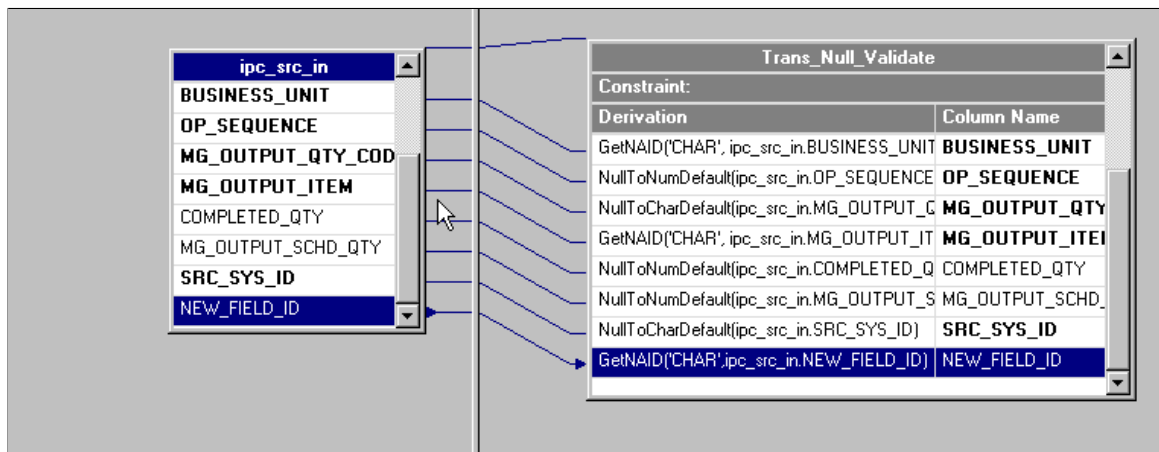
This example illustrates the fields and controls on the Dimension table hash file added. You can find definitions for the fields and controls later on this page.



3. Open the first transformer stage that follows the source (usually the *Trans\_Null\_Validate* stage).

**Image: Trans\_Null\_Validate Stage**

This example illustrates the fields and controls on the *Trans\_Null\_Validate* Stage. You can find definitions for the fields and controls later on this page.



4. The new field must be processed by the *GetNAID* routine.
5. Link the new field ID to all succeeding stages (up to the transformer stage) where the new hash file lookup is connected.



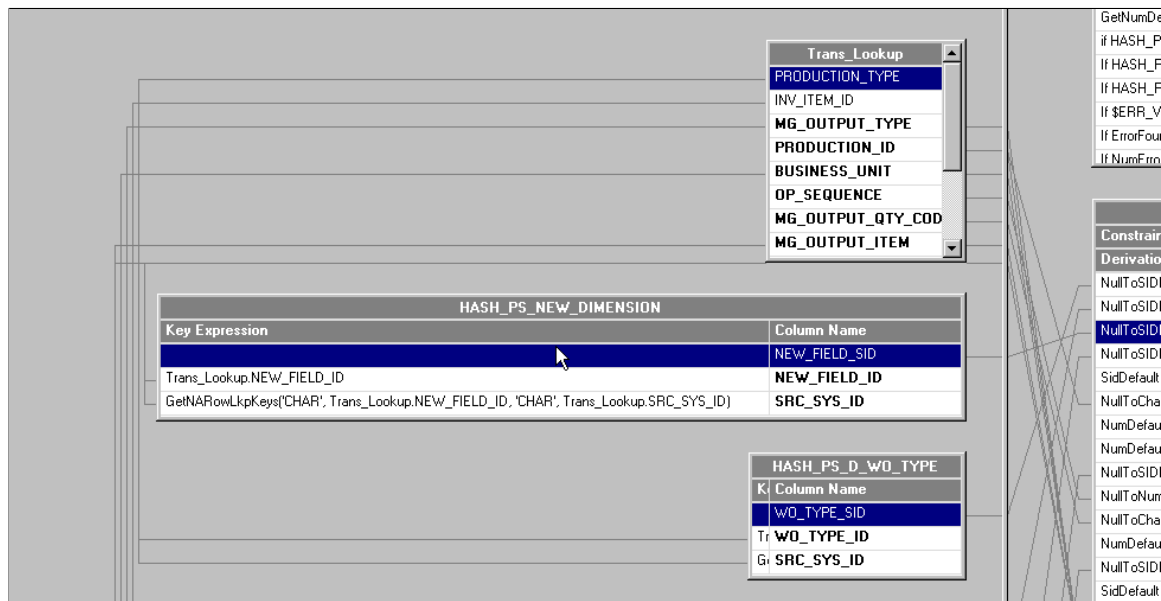
**Note:** The new dimension hash file usually is the same dimension hash file populated by the respective dimension server job. In cases where the hash file definition is different from the main dimension server job, a DRS stage must be defined in the fact job to create the dimension hash file.

- In the transformer stage, join the new field ID with the key ID field of the dimension hash file.

All other relevant fields, such as `SRC_SYS_ID`, must also be joined to the corresponding fields in the dimension hash files. Such fields must be validated by the *GetNARowLkpKeys* routine to support the *Not Available* row in the dimension hash file.

**Image: Transformer and Dimension Hash file join**

This example illustrates the fields and controls on the Transformer and Dimension Hash file join. You can find definitions for the fields and controls later on this page.



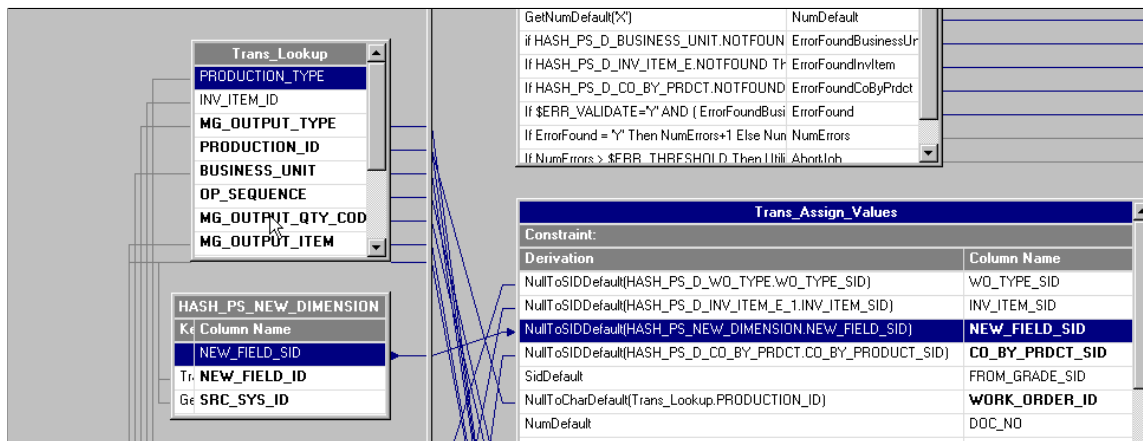
**Note:** If necessary, the new field must also be passed through any other stages between this transformer and the target DRS.

- 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.

**Image: SID extract**

This example illustrates the fields and controls on the SID extract. You can find definitions for the fields and controls later on this page.



8. Select *File, Save* from the menu to save the job.
9. 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.

10. If your job successfully compiles, select Close.

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

11. 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:

1. Define the new attribute desired and determine whether there are any corresponding related or outrigger language requirements for that attribute.
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 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.

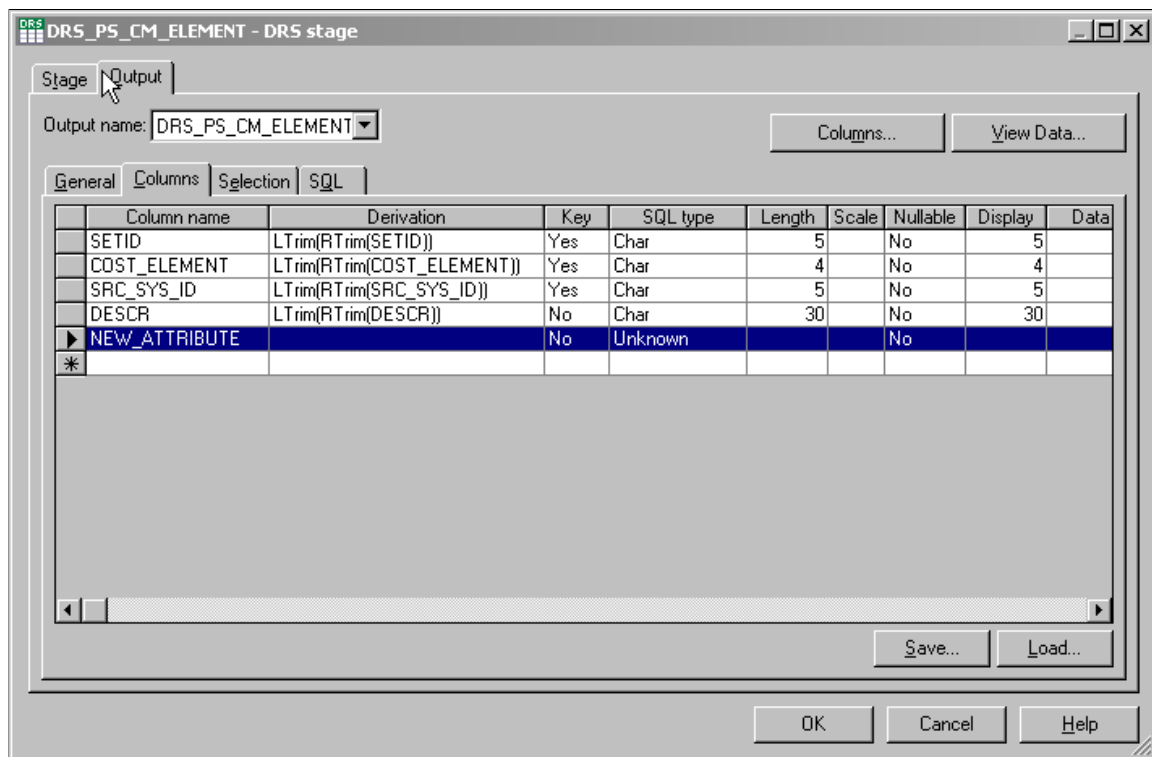
### Updating a Dimension Job with a New Attribute That Originates from a Source Table

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.

#### Image: DRS Stage with Columns sub-tab selected

This example illustrates the fields and controls on the DRS Stage with Columns sub-tab selected. You can find definitions for the fields and controls later on this page.



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.

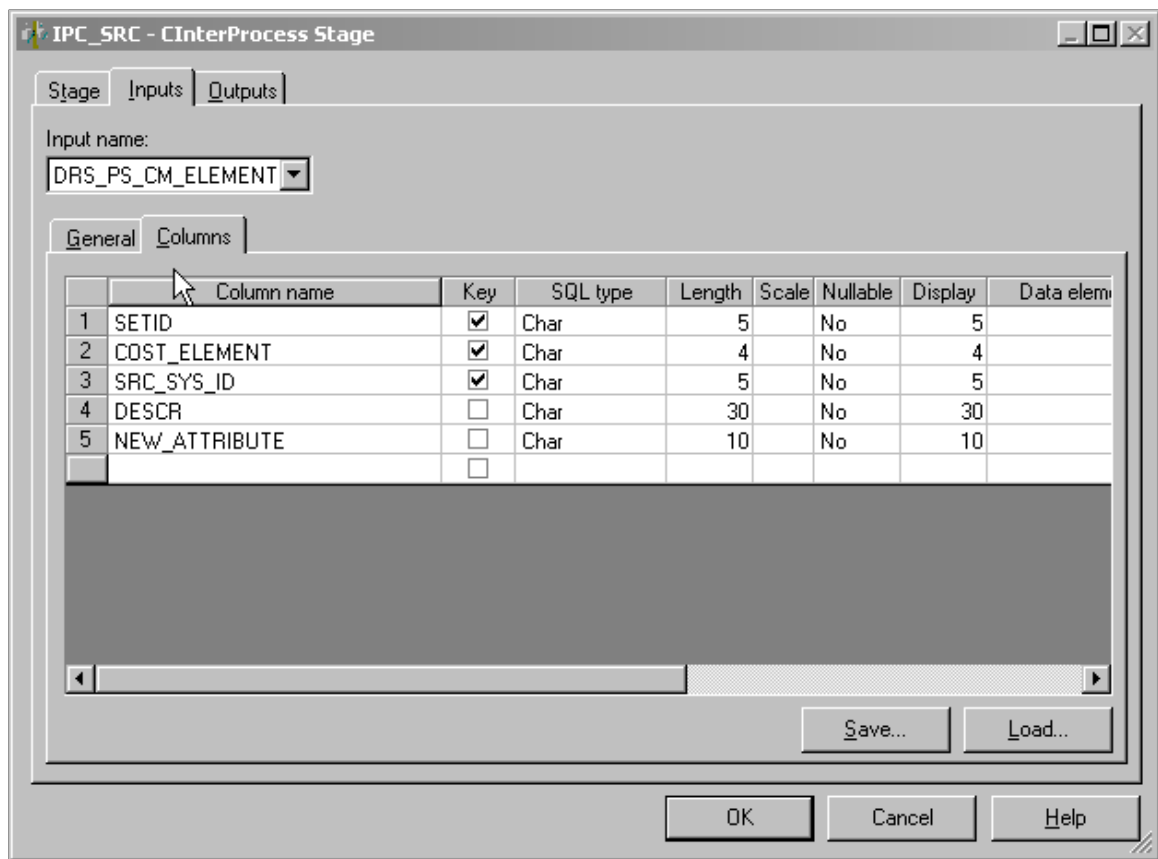
- In the Transformer Stage - Input Links window, apply any transformation logic, such as any string or number functions, as necessary.

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.

**Image: IPC\_SRC Stage**

This example illustrates the fields and controls on the IPC\_SRC Stage. You can find definitions for the fields and controls later on this page.

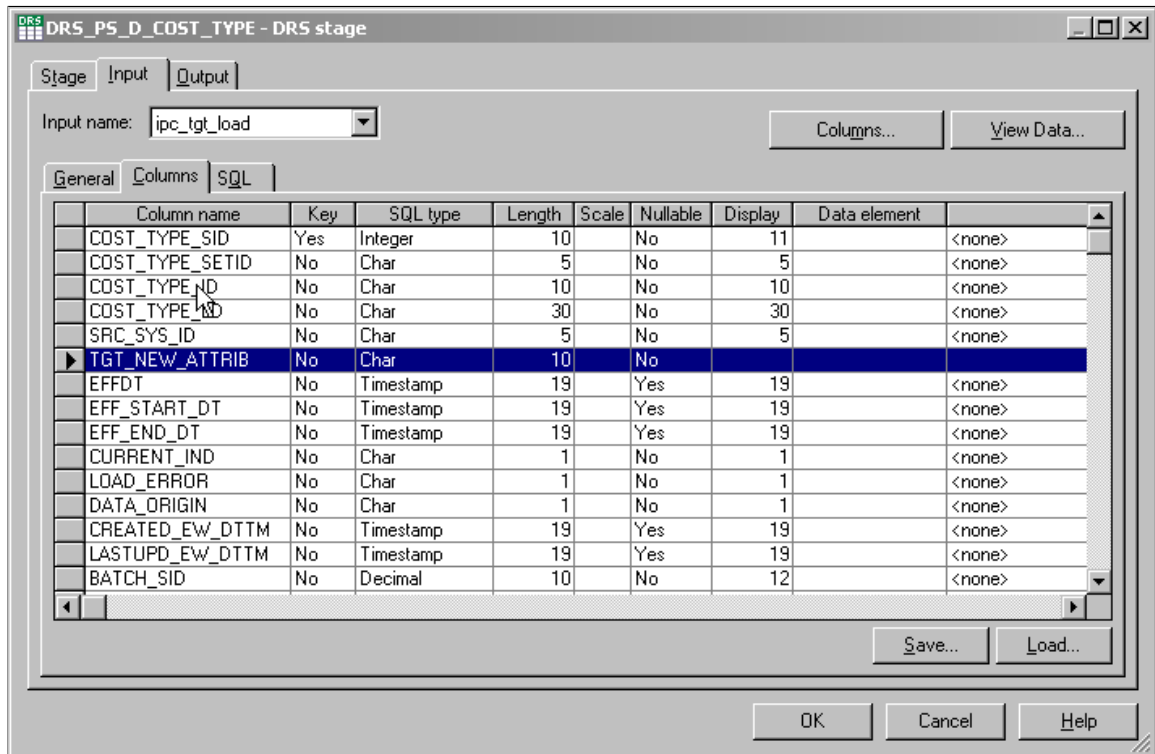


- In the Columns sub-tab ensure that the new attribute column is present and properly defined.
- Open the target DRS stage and select the Input tab.

- In the Columns sub-tab ensure that the new attribute column is present and properly defined.

**Image: Target DRS stage with new attribute row**

This example illustrates the fields and controls on the Target DRS stage with new attribute row. You can find definitions for the fields and controls later on this page.



- 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 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

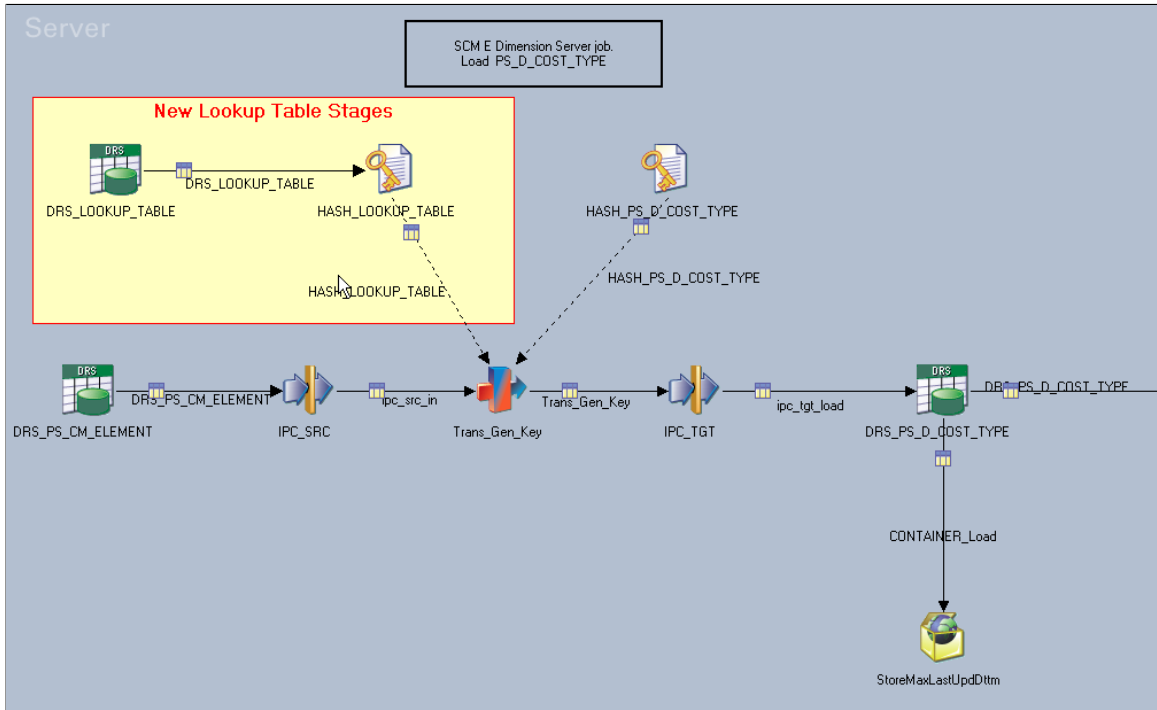
Perform the following steps to update a dimension job with an attribute that originates from a lookup table:

- 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.

**Image: New lookup table stages**

This example illustrates the fields and controls on the New lookup table stages. You can find definitions for the fields and controls later on this page.



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.

**Image: DRS Lookup with Stage tab and General sub-tab selected**

This example illustrates the fields and controls on the DRS Lookup with Stage tab and General sub-tab selected. You can find definitions for the fields and controls later on this page.

The screenshot shows a dialog box titled "DRS\_LOOKUP\_TABLE - DRS stage". It has two tabs: "Stage" and "Output", with "Stage" selected. Below the tabs, there is a "Stage name:" field containing "DRS\_LOOKUP\_TABLE". Below that, there are two sub-tabs: "General" and "NLS", with "General" selected. The "General" sub-tab contains the following fields and controls:

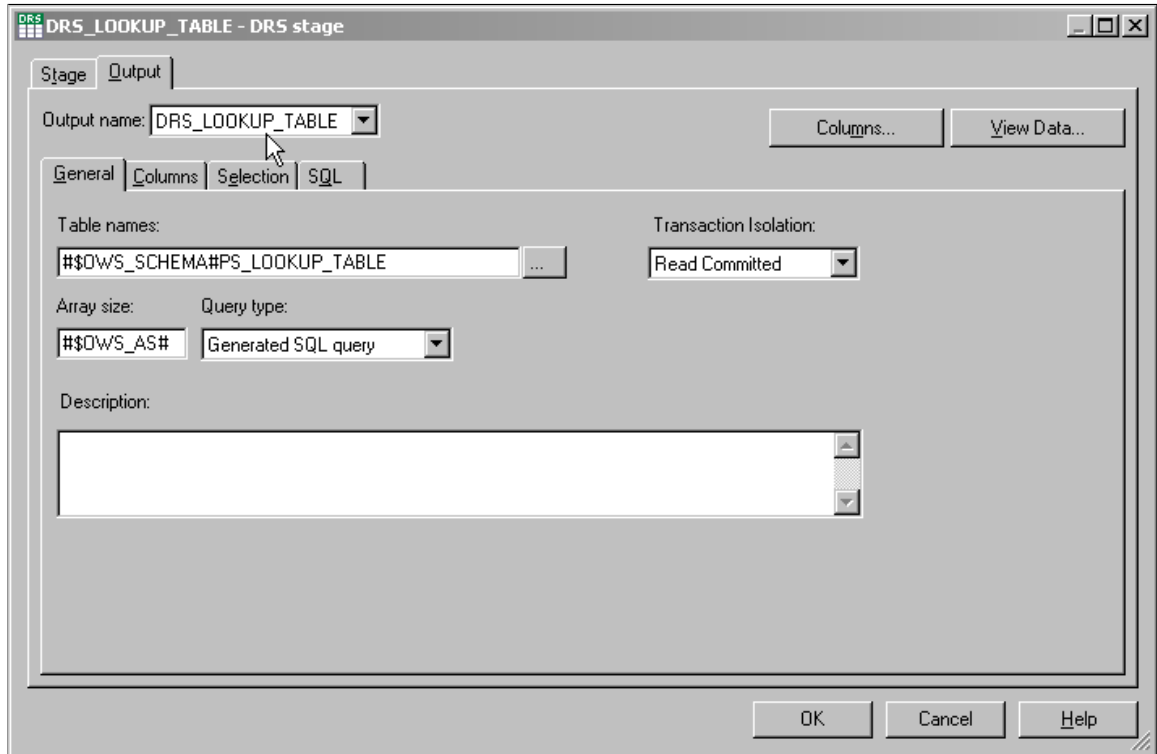
- Database type:** A text field containing "#OWS\_DBTYPE#" and a "DBMS Type" dropdown menu.
- Connection name:** A text field containing "#OWS\_DBCONNECTION#".
- User ID:** A text field containing "#OWS\_USERNAME#".
- Password:** A text field containing "XXXXXXXXXXXXXXXXXXXX".
- Description:** A text area containing "This stage is used to extract the incremental data from the source table PS\_LOOKUP\_TABLE".

At the bottom of the dialog box, there are three buttons: "OK", "Cancel", and "Help".

4. Select the Output tab and then the General sub-tab.

**Image: DRS Lookup with Output tab and General sub-tab selected**

This example illustrates the fields and controls on the DRS Lookup with Output tab and General sub-tab selected. You can find definitions for the fields and controls later on this page.



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.

**Image: DRS Lookup with Output tab and Columns sub-tab selected**

This example illustrates the fields and controls on the DRS Lookup with Output tab and Columns sub-tab selected. You can find definitions for the fields and controls later on this page.

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	
*								

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.

**Image: DRS Lookup with Output tab and Selection sub-tab selected**

This example illustrates the fields and controls on the DRS Lookup with Output tab and Selection sub-tab selected. You can find definitions for the fields and controls later on this page.

WHERE clause:

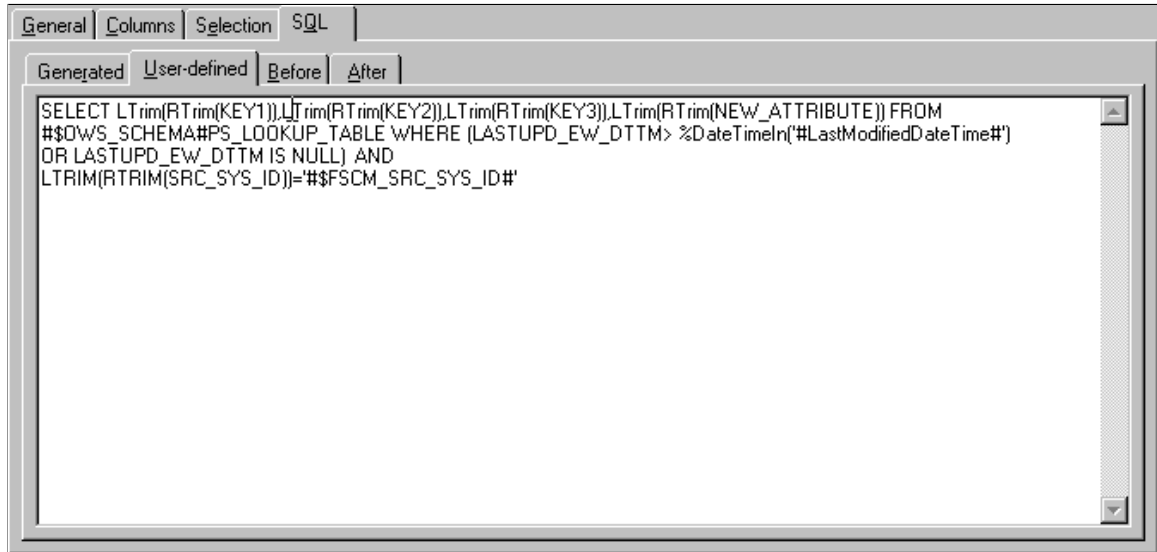
```
(LASTUPD_EW_DTTM > %DateTimeIn('#LastModifiedDate#')
OR LASTUPD_EW_DTTM IS NULL) AND
LTRIM(RTRIM(SRC_SYS_ID))=#FSCM_SRC_SYS_ID#
```

Other clauses:

9. Select the SQL sub-tab and input any user-defined query for the attribute in the User-Defined tab.

**Image: DRS Lookup with user defined SQL**

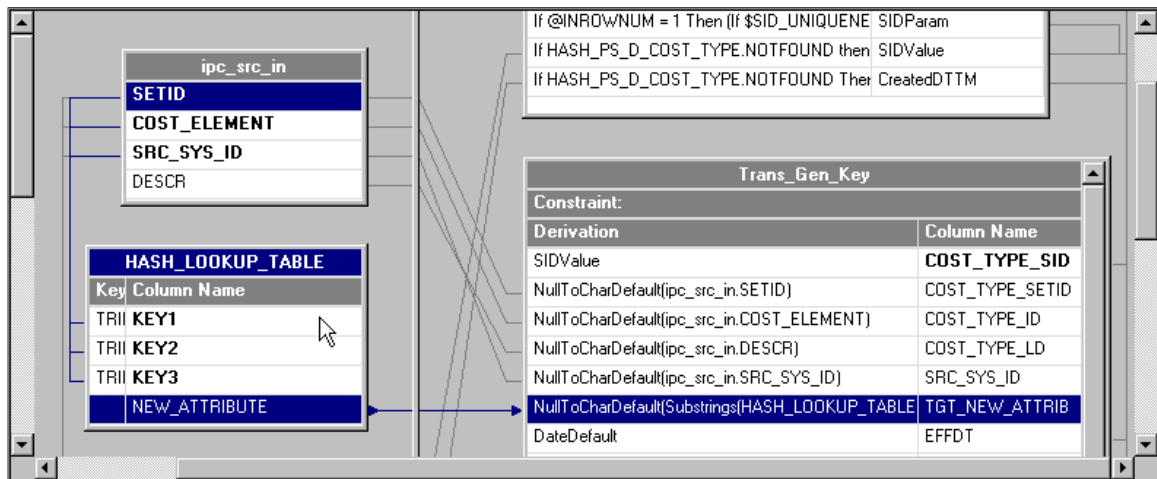
This example illustrates the fields and controls on the DRS Lookup with user defined SQL. You can find definitions for the fields and controls later on this page.



10. Connect an output link from the new DRS lookup stage to the applicable hash file stage.

**Image: Attribute to target mapping**

This example illustrates the fields and controls on the Attribute to target mapping. You can find definitions for the fields and controls later on this page.



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.

- 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.

---

- Connect the output link of the hash file stage to the transformer stage (Trans\_Gen\_Key).

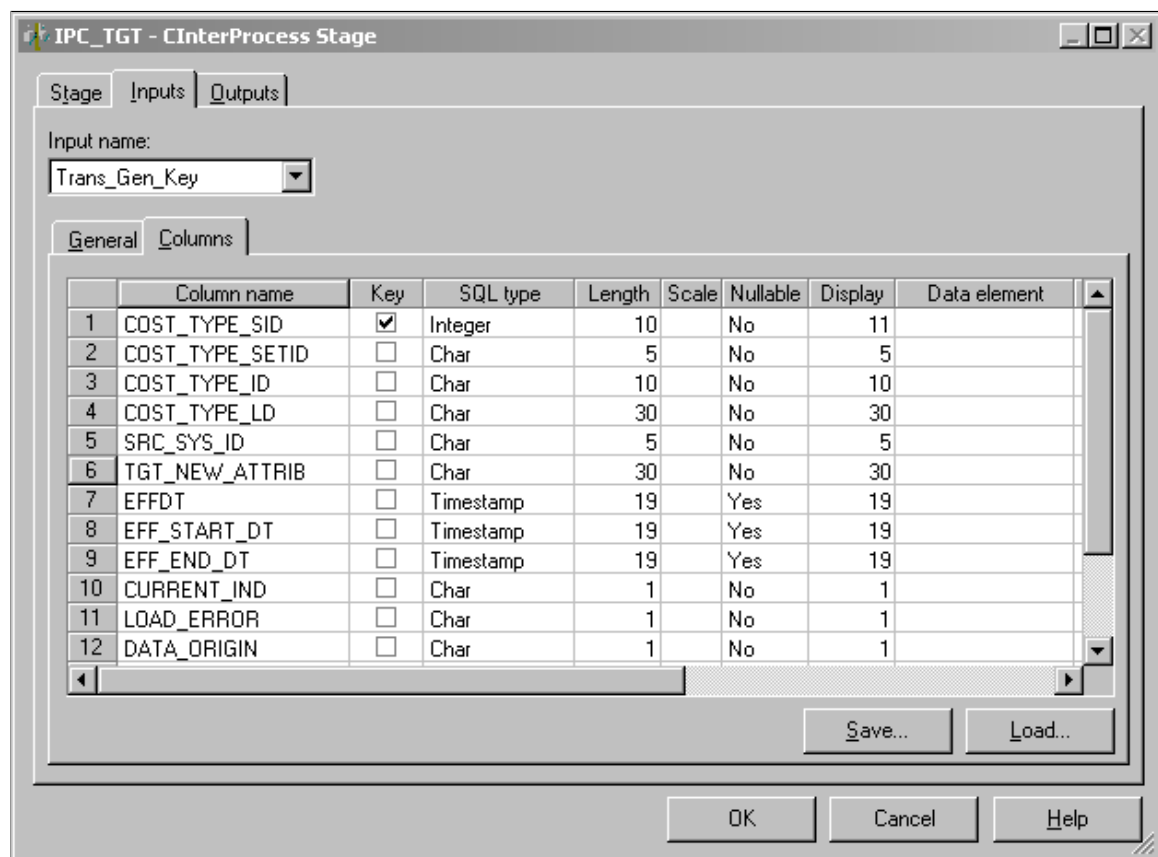
Once the link is connected to the transformer stage, the new lookup table becomes available in the inputs pane of the transformer stage.

- 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.

**Image: IPC Stage with column definitions**

This example illustrates the fields and controls on the IPC Stage with column definitions. You can find definitions for the fields and controls later on this page.



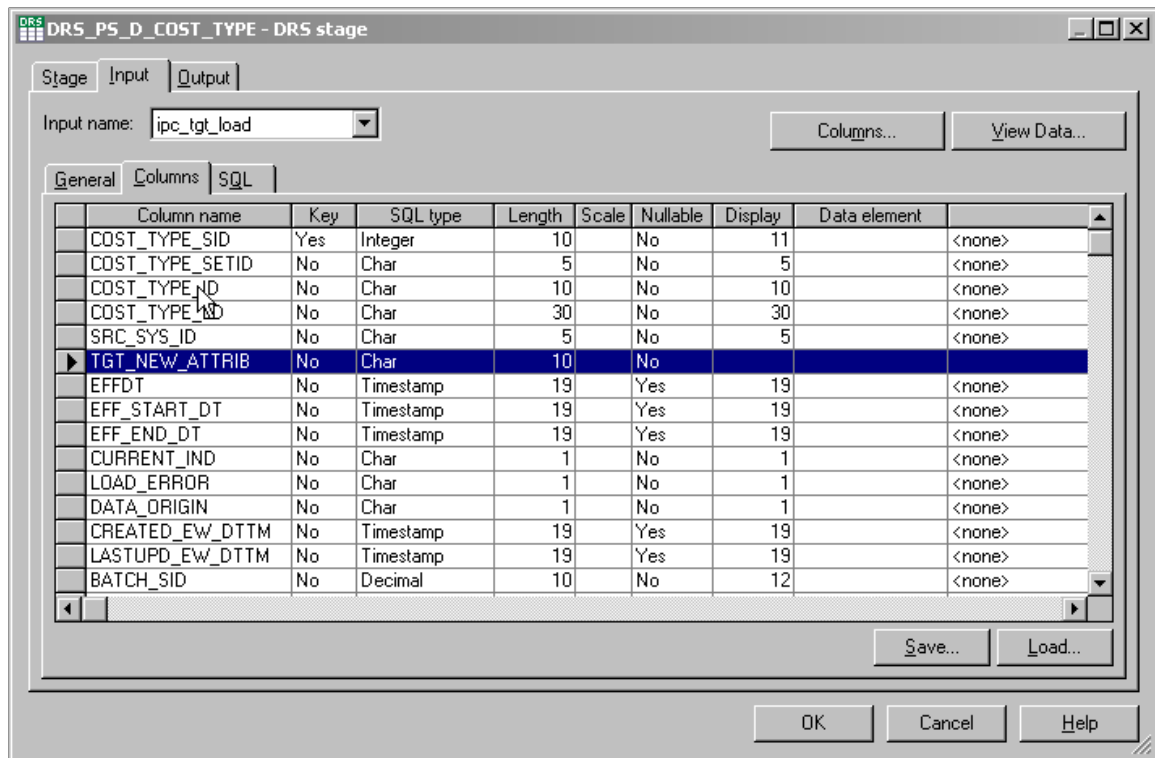
- Apply transformation logic, such as any string or number functions to the new attribute from the lookup table, as necessary.

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.

**Image: Target DRS stage with new attribute row**

This example illustrates the fields and controls on the Target DRS stage with new attribute row. You can find definitions for the fields and controls later on this page.



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 you 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.

# Using the PeopleSoft EPM Lineage Spreadsheets

---

## 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:

<b>Lineage Spreadsheet Filename</b>	<b>Warehouse</b>
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:

<b>Worksheet</b>	<b>Description</b>
Template	This worksheet contains overview information, a legend, and a definition of the columns used in the worksheets.

<b>Worksheet</b>	<b>Description</b>
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.
<Data Mart> For example: GL & Profitability, ESA, Campus Community, and so on.	This worksheet contains ETL lineage information for the jobs required for a specific data mart. <hr/> <b>Note:</b> Each spreadsheet includes several data mart worksheets. <hr/>
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.

<b>Column</b>	<b>Description</b>
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.
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.

<b>Column</b>	<b>Description</b>
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.
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.



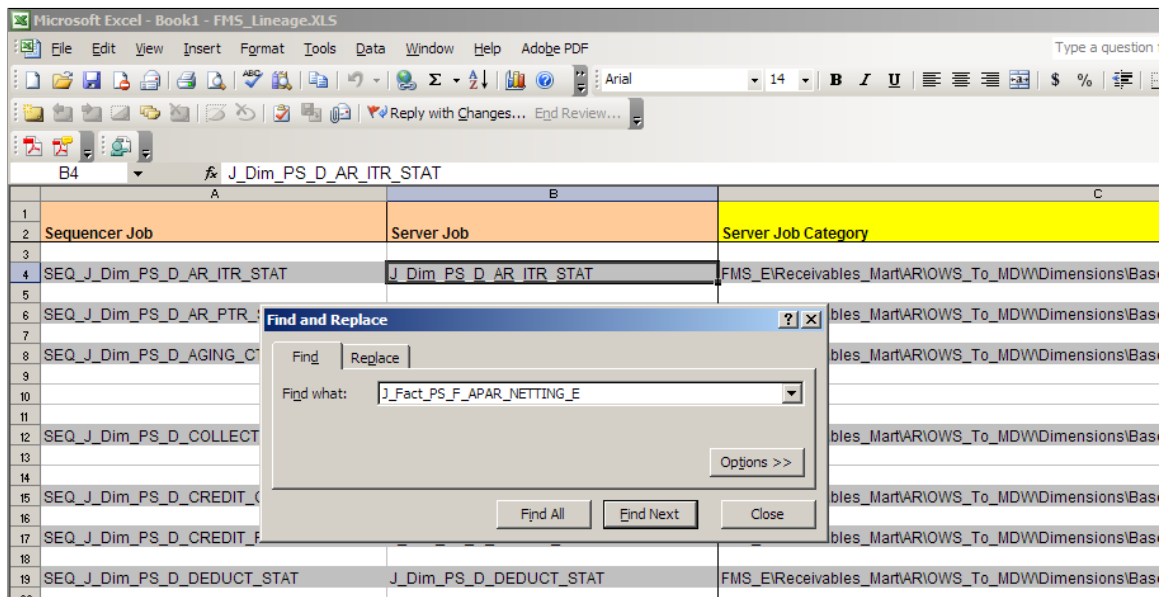
### Example

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.

#### Image: Find and Replace Dialog Box

This example illustrates the fields and controls on the Find and Replace Dialog Box. You can find definitions for the fields and controls later on this page.



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.

- Close the Find and Replace dialog box. You should see the following information:

**Image: J\_Fact\_PS\_F\_APAR\_NETTING\_E job displayed in spreadsheet**

This example illustrates the fields and controls on the J\_Fact\_PS\_F\_APAR\_NETTING\_E job displayed in spreadsheet. You can find definitions for the fields and controls later on this page.

	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_E\Receivables_Mart\AR\IOWS_T
92			
93			
94	SEQ_Dims_L_O_CREDIT_RISK	J_Dim_PS_O_CREDIT_RISK	FMS_E\Receivables_Mart\AR\IOWS_T
95			
96			
97	SEQ_Dims_L_O_DEDUCT_STAT	J_Dim_PS_O_DEDUCT_STAT	FMS_E\Receivables_Mart\AR\IOWS_T
98			
99			
100	SEQ_Dims_L_O_DISPUTE_STAT	J_Dim_PS_O_DISPUTE_STAT	FMS_E\Receivables_Mart\AR\IOWS_T
101			
102			
103	SEQ_Dims_L_O_ENTRY_RSTYP	J_Dim_PS_O_ENTRY_RSTYP	FMS_E\Receivables_Mart\AR\IOWS_T
104			
105			
106	SEQ_J_Fact_PS_F_APAR_NETTING_E	J_Fact_PS_F_APAR_NETTING_E	FMS_E\Receivables_Mart\AR\IOWS_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_E\Receivables_Mart\AR\IOWS_T

- Scroll to the right to review the columns shown here:

**Image: Reviewing data associated with the J\_Fact\_PS\_F\_APAR\_NETTING\_E job**

This example illustrates the fields and controls on the Reviewing data associated with the J\_Fact\_PS\_F\_APAR\_NETTING\_E job. You can find definitions for the fields and controls later on this page.

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	

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.

**Image: Lookup Tables Column**

This example illustrates the fields and controls on the Lookup Tables Column. You can find definitions for the fields and controls later on this page.

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		

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).

**Image: Cross-warehouse dependencies for PS\_D\_SUPPLIER**

This example illustrates the fields and controls on the Cross-warehouse dependencies for PS\_D\_SUPPLIER. You can find definitions for the fields and controls later on this page.

F	L	R
Source Table	DEPENDENCIES: ETL jobs which populate the source and the lookup t	
	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\OWS_To_MDW\Facts\Base\Load_Tables\Server
	SEQ_J_Fact_PS_F_AP_AGING_E	FMS_E\Payables_Mart\OWS_To_MDW\Facts\Base\Load_Tables\Server

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\AP\OWS\_To\_MDW\Facts\Base\Load\_Tables\Server. This fact job belongs to a different mart as indicated by the different color.

### Image: Cross-warehouse dependencies for PS\_F\_AP\_AGING

This example illustrates the fields and controls on the Cross-warehouse dependencies for PS\_F\_AP\_AGING. You can find definitions for the fields and controls later on this page.

H	J	K	L	M	N	O	P
DEPENDENCIES: ETL jobs which populate the source and the lookup							
Lookup Tables	MDW	MDW Sequencer	Category				
	J_Dim_PS_D_CUST_ORG_SCM	SEQ_J_Dim_PS_D_CUST_ORG_SCM	Global_Dimensions_EIOWS_To_MDW\Base\Load_Tables				
	J_Dim_PS_D_SUPPLIER	SEQ_J_Dim_PS_D_SUPPLIER	Global_Dimensions_EIOWS_To_MDW\Base\Load_Tables				
	J_Fact_PS_F_AR_AGING_E	SEQ_J_Fact_PS_F_AR_AGING_E	FMS_E\Receivables_Mart\AP\OWS_To_MDW\Facts\Base				
PS_F_AP_AGING	J_Fact_PS_F_AP_AGING	SEQ_J_Fact_PS_F_AP_AGING_E	FMS_E\Payables_Mart\AP\OWS_To_MDW\Facts\Base\Load				

## 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 CS 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 "Understanding Warehouse Business Unit Setup (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)"

- 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 "Understanding Warehouse Business Units, TableSet Sharing, and SetID Mapping (*PeopleSoft EPM 9.1: Enterprise Performance Management Fundamentals*)".

- 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.