ORACLE
PEOPLESOFT

PeopleSoft EPM 9.1 Supply Chain Management Warehouse PeopleBook

April 2012

ORACLE

PeopleSoft EPM 9.1 Supply Chain Management Warehouse PeopleBook
SKU epm91pscw-b0412

Trademark Notice

License Restrictions Warranty/Consequential Damages Disclaimer

Warranty Disclaimer

Restricted Rights Notice

Hazardous Applications Notice

Third Party Content, Products, and Services Disclaimer

Contents

PeopleSoft Supply Chain Management Warehouse Preface

This preface discusses:

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

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.

PeopleSoft Application Fundamentals

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

PeopleBooks and the PeopleSoft Online Library

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

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

- ISO country codes and currency codes.
- PeopleBooks that are common across multiple applications.
- Common elements used in PeopleBooks.
- Navigating the PeopleBooks interface and searching the PeopleSoft online library.
- Displaying and printing screen shots and graphics in PeopleBooks.
- How to manage the locally installed PeopleSoft online library, including web site folders.
- Understanding documentation integration and how to integrate customized documentation into the library.
- Application abbreviations found in application fields.

You can find *PeopleBooks and the PeopleSoft Online Library* in the online PeopleBooks Library for your PeopleTools release.

Chapter 1

Getting Started with PeopleSoft Supply Chain Management Warehouse

This chapter provides an overview of PeopleSoft Supply Chain Management Warehouse and discusses other sources of information.

PeopleSoft Supply Chain Management Warehouse Overview

PeopleSoft Supply Chain Management Warehouse serves both as a repository of information and as the foundation for business intelligence reporting. PeopleSoft Supply Chain Management Warehouse draws data from PeopleSoft source transaction systems to stage, store, and enrich information for reporting.

Other Sources of Information

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

See Also

PeopleSoft PeopleTools PeopleBook: PeopleSoft Setup Manager

About These PeopleBooks

Chapter 2

Understanding the EPM Warehouses

This chapter provides an overview of the EPM Warehouses.

Overview of PeopleSoft EPM Warehouses

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

The following sections describe these PeopleSoft EPM Warehouses:

- PeopleSoft 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 PeopleBook.*

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

Campus Solutions Warehouse

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

- Student Recruiting
- Student Admission Application
- Application Evaluations
- Student Responses
- External Test Scores

- External Education
- Class Meeting Patterns
- Course Catalog, Class Scheduling, Instructor Workload
- Program Activation & Management
- Student Career Term Record Management
- Enrollment
- Award
- Student Payments

The Campus Solutions warehouse consists of the following data marts:

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

See *PeopleSoft Campus Solutions Warehouse 9.1 PeopleBook*, "PeopleSoft Campus Solutions Warehouse Preface."

CRM Warehouse

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

- Marketing
- Support
- Sales

The CRM Warehouse consists of these data marts:

- Sales
- Service
- Marketing
- Customer Segment

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

FMS Warehouse

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

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

The Financial Management Solutions Warehouse consists of these data marts:

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

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

Financials Warehouse for Public Sector and Higher Education

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

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

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

- Procurement (from the Supply Chain Management Warehouse)
- Spend (from the Supply Chain Management Warehouse)
- Receivables (from the Financial Solutions Management Warehouse)
- Payables (from the Financial Solutions Management Warehouse)
- General Ledger and Profitability (from the Financial Solutions Management Warehouse)
- Enterprise Services Automation (from the Financial Solutions Management Warehouse)

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

HCM Warehouse

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

- Deployment
- Reward
- Development
- Recruiting

The HCM Warehouse consists of these data marts:

- Workforce Profile
- Compensation
- Learning and Development
- Recruiting

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

SCM Warehouse

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

- Order Fulfillment
- Procurement
- Production

The Supply Chain Warehouse consists of these data marts:

- Fulfillment and Billing
- Procurement

- Spend
- Inventory
- Manufacturing
- Supply Chain Planning

See Chapter 3, "Understanding the Supply Chain Warehouse," page 17.

Components of PeopleSoft EPM Warehouses

PeopleSoft delivers the following content with an EPM warehouse:

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

Each bullet is discussed in more detail below.

Extract Transform and Load (ETL) Component

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

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

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

Infrastructure Tables and Tools

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

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

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

Security Tables

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

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

Staging Tables

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

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

Multidimensional Warehouse Fact Tables

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

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

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

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

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

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

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

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

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

Multidimensional Warehouse Dimension Tables

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

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

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

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

Shared and Global Dimensions

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

Commonly Used Dimensions

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

Common Dimension	*Description*
Business Unit Dimension	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. 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. 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 additional to being a regular business unit in the Business Unit dimension table. **Note.** Business units that come from different source systems are different business units, even if they have the same name and the same BUSINESS_UNIT value.

Common Dimension	***Description***
Calendar Dimension	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. In addition to having a granularity of *day*, the Gregorian calendar also provides hierarchies of *week*, *month*, *quarter*, and *year*. 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. For user-defined calendars, the lowest granularity is also a *day*, 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.
Currency Dimension	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.
Language	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.
Time Dimension	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. The Time dimension includes a textural Time Period attribute. This attribute refers to specific periods of time, such as AM or PM.

Common Dimension	*Description*
Unit of Measure Dimension	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. 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. Some conversions of UOM are standard and are independent from the subject of measurement, such as from meters to feet. However, some conversions depend on a set of attributes, such as shipping vendor, business unit, a particular item, and so on. The Unit of Measure table facilitates this conversion process. **Note.** You must populate this relationship table according to your particular requirements.
Time Zone Dimension	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.

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

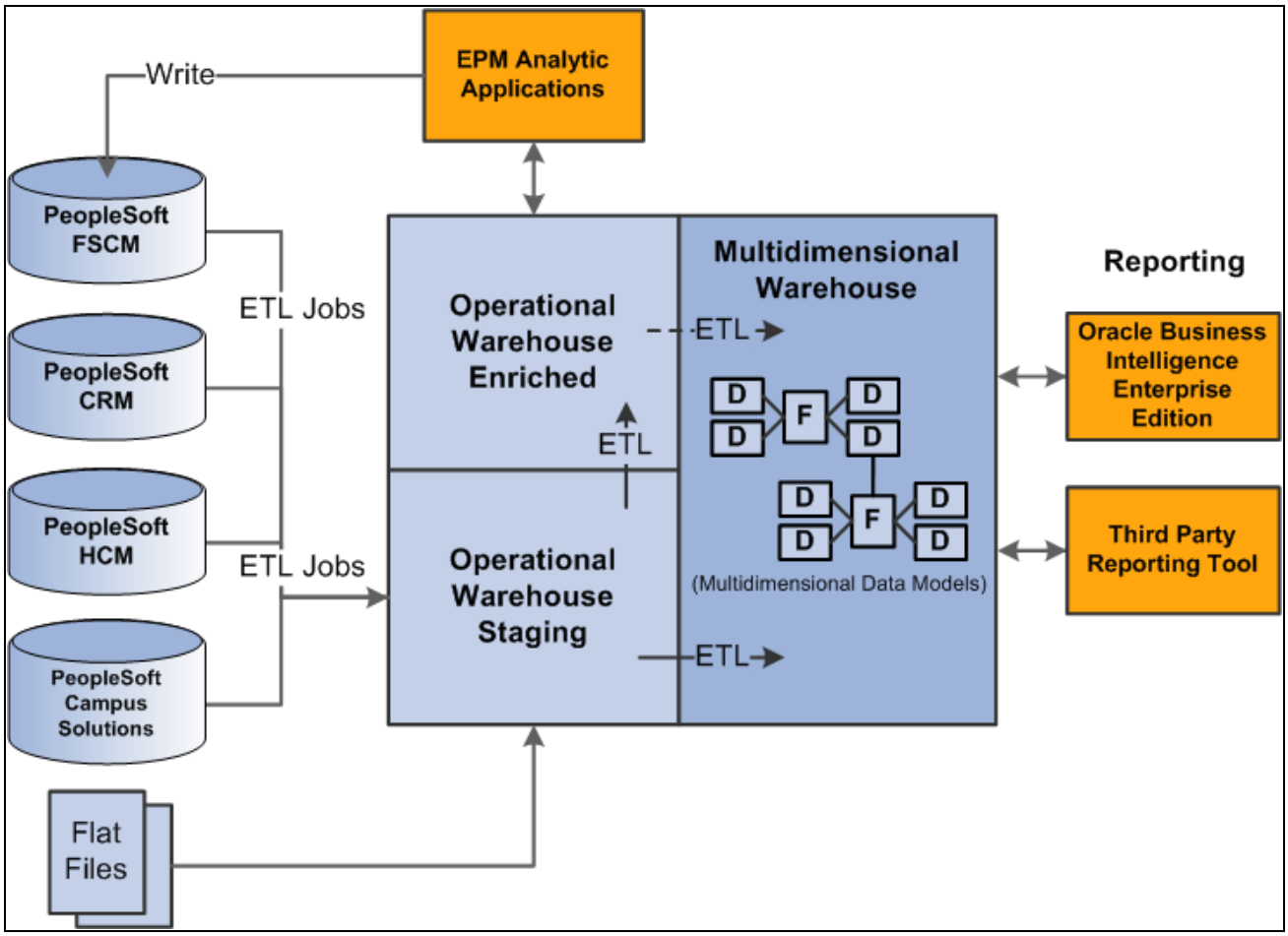
PeopleSoft EPM warehouses are built on a foundation of infrastructure tables and tools, ETL platform, and staging/multidimensional tables, all of which enable the warehouses to bring together data from different PeopleSoft source systems. Prepackaged data models enable complex analysis and reporting 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 chapters that describe the specific data marts that you are licensed to use and complete any additional setup that is necessary. Each data mart might have additional setup or processing steps that you must perform before creating the data mart. Review these steps in the chapter for that data mart in this PeopleBook.

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

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



EPM Data Flow

Operational Warehouse - Staging (OWS)

The first step in preparing your data for multidimensional reporting is to load source data from your PeopleSoft source transaction system into the OWS layer. You use PeopleSoft delivered ETL jobs to extract and load the data into the OWS. 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 ads additional fields to facilitate incremental loading (date stamps), and source and error tracking. These can typically be found in the LOAD_OWS_SBR subrecord.

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

Operational Warehouse - Enriched (OWE)

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

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

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

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

Multidimensional Warehouse (MDW)

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

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

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

Reporting on the EPM Warehouses

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

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

Chapter 3

Understanding the Supply Chain Warehouse

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

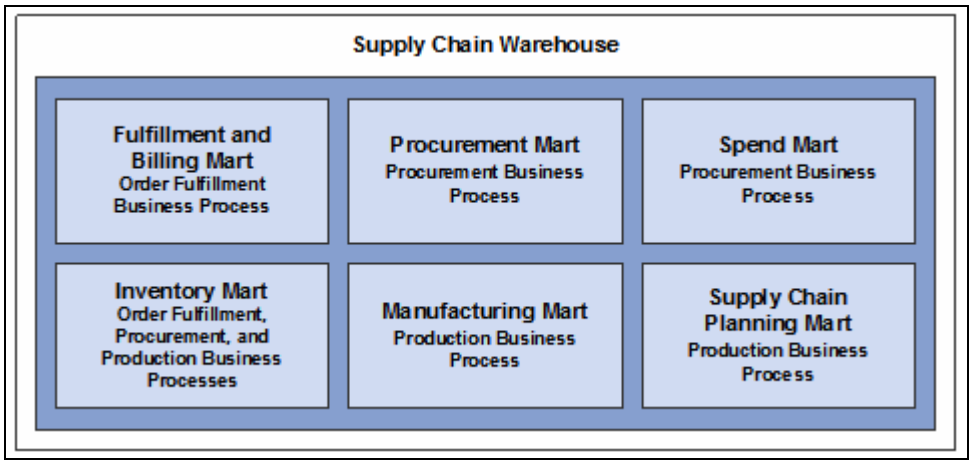
Supply Chain Warehouse Overview

The PeopleSoft SCM Warehouse is a comprehensive business intelligence platform for supply chain analytics. At the core of the SCM Warehouse are prepackaged dimensional data models, which optimize the arrangement, accessibility, and reportability of your manufacturing, inventory, spend, and supply planning data. With these data models you can review data against organizational metrics and perform strategic analyses using prepackaged measures such as order amount, units ordered, units cost, and quantities ordered and shipped.

The SCM Warehouse enables your organization to identify and track multiple dimensions that affect profitability, identify key performance drivers, trends, and opportunities. The SCM Warehouse also supports detailed production and supply chain planning compliance analysis, consolidating procurement spending, aggregating inventory availability, and combining customer satisfaction measures for corporate, operation, or department level analysis.

Understanding the Supply Chain Warehouse Structure

Data marts are logical divisions within the SCM Warehouse and are comprised of subject-specific dimensional data models designed around a specific institutional process. The SCM Warehouse includes the Fulfillment and Billing data mart, Procurement data mart, Spend data mart, Inventory data mart, Manufacturing data mart, and Supply Chain Planning data mart.



Supply Chain Warehouse marts and business processes

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

Note. Although we list only the actual fact name, all facts that represent dollar amounts have three facts in the fact table, one for each possible selected reporting currency. Populating the facts for alternate currencies is optional.

For additional details of facts, see the PDF file that is published on CD-ROM with your documentation.

This section discusses:

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

Fulfillment and Billing Data Mart

Analytics derived from Fulfillment and Billing data mart support the order-to-cash processes within the supply chain by providing analytics that track customers, orders, invoices, returns, and shipping transactions. You can use the analysis of individual product sales by channel, region, and other dimensions to greatly assist future marketing and product development initiatives, yielding a significant bottom line impact. Analytics cannot only help maintain customer satisfaction, but they can also help drive efficiencies into the supply chain, maintain cost controls, and ensure that fulfillment is aligned with manufacturing capabilities. The data mart also enables complete margin analysis, helping you to identify and track sales, returns, costs, discounts, and gross profit by customer.

With Fulfillment and Billing data mart you can answer questions such as:

- Which customers are returning the most products?
- Which products are being returned the most often?
- Why are products being returned?
- Which customers contribute the most to profit?
- Which products produce the most profit?
- Which products are discounted the most, and to which customers?
- What is the overall order fill rate?
- What is the fill rate for each product?
- What is the percentage of late shipments?
- What is the value of the late shipments?
- What are the total dollars shipped per period?
- What was the total value of shipped products for a given period?
- What was the total cost of the shipped products?
- What is the gross margin per product shipped?

Order Fulfillment Business Process

Fulfillment and Billing data mart is related to the 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.

Fulfillment and Billing Data Mart Delivered Fact and Dimension Tables

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

Fulfillment and Billing Data Mart Fact Tables

The following table describes the delivered Fulfillment and Billing data mart fact tables.

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

Fact Name	*Fact Record Name*	*Description*	*Helps Answer*
Billing	F_BILLING	The Billing fact table contains billing transaction information. This is an incremental fact table.	Revenue by customer. Quantity charged on an invoice. Discount charged on an invoice. Tax amount on an invoice. Invoice gross amount. Invoice net amount.
Booking	F_BOOKINGS	Stores information related to sales booking and enables you to evaluate an employee based on the sales they generate.	Number of sales orders, booked amount, and discount by product, sales person and by requested ship date. Sales order net booked quantity. Sales order net booked amount. Set booked cost for a sales order. Net booked shipped quantity for a sales order. Backordered quantity for a sales order.
Sales Kit Item	F_KIT_ITEM	Stores information related to sales kit items.	Unit cost of each sales order line item. Number of kit items, the item unit cost, and shipped quantities associated with a product, a sales order, and a business unit. Unit price of each sales order line item. Ordered kit item quantity. Shipped kit item quantity. Cancelled kit item quantity.

Fact Name	*Fact Record Name*	*Description*	*Helps Answer*
Sales Order Cycle Time	F_O2C_LTCT	Stores information related to sales order cycle time and lead time and provides a view of the complete order to cash process so you can evaluate the time it takes to complete the process − or life cycle − of Order to Cash. The order to cash cycle time is a key measure of cash flow efficiency. The facts enable you identify each segment of the cycle to understand which segments need improvement.	Cycle time and lead time for sales order booking to shipping. Shipping to delivery. Delivery to return Order to invoice Order to cash. Lead time in days from sales order to order pick. Lead time in days from order schedule to order shipment. Lead time in days from order date to actual ship date. Lead time in days from order date to delivery date. Lead time in days from delivery date to return date. Lead time in days from sales order to invoice. Lead time in days from sales order to cash. Difference between quoted lead time and actual shipment lead time.
Return Material Authorization (RMA)	F_RMA_RECEIVED	Stores information about returned material from a sales order and aids in evaluating return sales order price, quantity, and so on. This fact group helps you analyze customer returns by product, channel, carrier, and shipment date.	Number of products returned. Returned amount by customer, by product, by channel, by carrier. Number of sales order line items returned. Return price for the sales order line item. Total sales order quantity returned. Total sales order return amount.

Fact Name	*Fact Record Name*	*Description*	*Helps Answer*
Sales Order	F_SALES_ORDER	Stores information related to the sales order header level and helps evaluate metrics such as Net Price, Quantity, List Amount, and so on.	Product sales by channel, by customers, by shipment dates, or by business unit. Price ordered by a customer. Billback value associated with a sales order. List price of the product that was ordered. Total dollars ordered by product or customer. Sales order extended price.
Sales Order Line	F_SO_LINE	Stores information related to sales order line items, such as order quantities and amounts, at a sales order line level of detail.	Unit cost and unit price for each item Number of units ordered for each item. Lead time for an ordered item and number shipped. Number of backordered units. Number of units that are cancelled.
Sales Order Shipment	F_SO_SHIP	Stores information related to sales order line and shipping line.	Shipment dates and delivery performance by customer, by channel, by product, or by carrier. Number of items shipped for a sales order. Total shipping cost for an invoice. Number of line items for a sales order. Shipping price for a specific line item. Unit price of each sales order line item. Quantity shipped for each sales order line item. Quantity backordered for each sales order line item. Extended cost for the sales order line item.

Fulfillment and Billing Data Mart Dimension Tables

The following table describes the delivered Fulfillment and Billing data mart dimension tables.

Dimension Name	*Dimension Record Name*	*Description*
Back Order Status	D_BCKORD_STATUS	Stores back order statuses.
Fulfilment Status	D_FULFLMNT_STAT	Stores fulfillment status.
Last Status	D_LAST_STAT	Stores last type.
Order Type	D_ORD_TYP	Stores supply chain order types.
Reason Code	D_REASON_CODE	Stores reason codes, such as inventory reason code, rejection reason codes, and so on.
RMA Status	D_RMA_STATUS	Stores return material authorization statuses.
RMA Cancel	D_RMACNL_REASON	Stores return material cancel reason statuses.
RMA Line	D_RMALN_STATUS	Stores return material line statuses.
Sales Organization	D_SALES_ORG	Stores sales organization and its hierarchy.
Sales Order Type	D_SLSORD_TYPE	Stores sales order types.
Sales Quote Type	D_SLSQUOTE_TYPE	Stores sales quote type.
Sales Order Status	D_SO_STATUS	Stores sales order statuses.
SO Line Status	D_SOLN_STATUS	Stores sales order line statuses.

Procurement Data Mart

With the Procurement data mart, you evaluate the reliability, timeliness, and efficiency of your procurement process and supply base. Analyzing buyer workload, requisition fulfillment and purchase order processing enables you to identify and eliminate key procurement process inefficiencies. Procurement data mart also enables you to evaluate detailed supplier receipt, such as on time deliveries, over or under shipments, and product quality for complete supplier performance analysis. This crucial view of your procurement process can help your company expose problems early, reducing cost and increasing customer satisfaction.

Procurement data mart enables you to answer questions such as:

- What is the total quantity ordered under contract per supplier or buyer?
- What is the total dollar amount committed under contract per supplier or buyer?

- What is the total dollar amount committed (ordered) per supplier?
- Which suppliers contribute the most in terms of purchase order activity?
- Which buyers are the most productive?
- Which suppliers have the most returns?
- How much improvement has a supplier made in returns?
- How much more purchasing volume has transpired over the year?
- What is the total amount ordered from a specific supplier over a specified time period?
- What is the total purchase order dollars controlled by one buyer?
- Has the number of rejected units from a supplier improved over time? Is a specific item causing the greatest number of rejects?
- What is the supplier service quality?
- How often does a supplier under or over ship?
- Which suppliers have the best acceptance rates?
- What percentage of time does a supplier ship early or late?
- Do the early or late shipments occur at the beginning or end of the month?
- What is the average difference between the shipment due date and receipt date? Has the supplier improved over time?
- How often does a supplier over or under ship orders?
- What is the value of those under or over shipments?
- Is a supplier improving shipment accuracy from period to period?
- What is the acceptance rate for orders received?
- Has the percent inspected increased due to supplier past acceptance rates?
- What is the value of the rejected orders?
- How long does a requisition take to process?
- What is the amount requisitioned by each department?

Procurement Business Process

Procurement 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 and sourcing strategies, collaborate with suppliers, and drive efficient procurement and settlement for all goods and services.

Procurement Data Mart Delivered Fact and Dimension Tables

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

Procurement Data Mart Fact Tables

The following table describes the delivered Procurement data mart fact tables.

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

Fact Name	*Fact Record Name*	*Description*	*Helps Answer*
Match Analysis	F_MTCH_ANLYS	Stores the voucher dollar amount and age of vouchers related to invoice-entered date, invoice last updated date, and invoice age. This information is stored at the Voucher Line level.	Age of an invoice. Lead-time for each matching process by supplier, item, date, or business unit. Amount for a voucher. Length of time between invoice entered date and invoice matching date.
PO Disposition	F_PO_DISP	Stores information on purchase order disposition, such as units disposed. This helps in maintaining an accurate unit count for a particular item.	Number of disposed units in the primary unit of measure.
PO Distribution	F_PO_DIST	Stores purchase order distribution information such as Merchandise Amount and Ordered Quantity. This information is stored at the Purchase Order Distribution Line level.	PO freight amount for a supplier. PO merchant amount for a carrier. PO sales tax amount for a business unit or item. Number of lines on the purchase order that have shipped.

Fact Name	*Fact Record Name*	*Description*	*Helps Answer*
PO Line	F_PO_LINE	Stores information on purchase order line, such as Unit Cost of a Commodity, Units On Hold, Units Ordered, on Hold Amount, Open Amount, Relieved Amount. This information is stored at purchase order line level.	PO unit price by supplier. Total number of purchase order units ordered. Total received amount. Number of units on hold. Total number of units received. Total amount deducted from inventory to fill an order.
PO Line Matching	F_PO_LN_MACT	Stores information on purchase order line match.	-
PO Receipt	F_PO_RCPT	Stores information on purchase order receipts and vouchers. It captures the information of receipt record and voucher record with details such as open quantity and open amount. All measures are at the inventory item level of detail.	PO open quantity and open amount for a business unit and for a corresponding account. PO receipt adjusted quantity. PO receipt closed amount.
PO Receipt Matching	F_PO_RCPT_MACT	Stores information on purchase order receipt match.	-
PO Shipment Receipt	F_PO_SHIP_RCPT	Stores information on purchase order receipts for the received line of Shipments. This information is stored at the Receipt Line Shipment level.	Receipt Amount for a line of shipment from a particular location or a particular receipt date or for a given due date. PO accepted amount for a particular shipment. PO accepted quantity for a particular shipment. PO rejected quantity for a particular shipment.

Fact Name	*Fact Record Name*	*Description*	*Helps Answer*
Procurement Cycle Time	F_PROC_CTLT	Stores procurement cycle time lead time analysis measures. This information is stored at the Purchase Order Line level.	Lead time for a procurement process by supplier, business unit, date, or by reject reason. Lead time quote by a supplier. Number of days from the purchase order dispatch to the material receipt. Number of days from material receipt to the voucher.
Requisition	F_REQN	Stores requisition line level measures.	Quantity for a material requisition. Price for a material requisition. Buyer and buyer's department for a material requisition.
Requisition Analysis	F_REQN_ANLYSIS	Stores requisition analysis measures. This information is stored at the Requisition Line level.	Lead time for a requisition process by business unit, buyer, buyer department, or location. Time taken from purchase order requisition to approval. Time taken from purchase order dispatch to material receipt. Time taken from the voucher to the last payment.
Return to Vendor	F_RTV	Stores return to vendor line level facts.	Price of the units returned to vendor for a particular supplier. Number of units were returned to vendor for a specific item. Number of returned items on a purchase order.

Fact Name	Fact Record Name	Description	Helps Answer
Return to Vendor Distribution	F_RTV_DIST	Stores return to vendor shipping line level information, such as PO Amount Returned and PO Quantity Returned. This information is stored at the RTV Shipping Line level.	Number of returned items for a business unit. Return reason. Number of items returned for a particular date. Price of the returned item. Total amount for returned items on a purchase order.

Procurement Data Mart Dimension Tables

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

Dimension Name	Dimension Record Name	Description
Cancel Status	D_CNCL_STATUS	Stores supply chain cancel statuses.
Completion Status	D_COMP_STATUS	Stores completion statuses regarding supply chain.
Delivery Status	D_DLVRY_STATUS	Stores delivery statuses regarding supply chain.
Item Cost	D_ITEM_COST	Specifies the basis for calculating item costs.
Match Status	D_MTCH_STATUS	Stores purchase order match statuses.
Origin	D_ORIGIN	Stores origin codes, such as purchase order origin codes.
PO Status	D_PO_STATUS	Stores purchase order status.
PO Distribution Status	D_PODIST_STATUS	Stores purchase order distribution status.
Receive Status	D_RECV_STATUS	Stores receive statuses regarding supply chain.
Requisition Status	D_REQ_STATUS	Stores purchase order requisition statuses.
Requisition Line Status	D_REQLN_STATUS	Stores requisition line statuses.
Return Reason Status	D_RTRN_REASON	Stores return reason statuses regarding supply chain.
Return to Vendor Status	D_RTV_STATUS	Stores return to vendor statuses.

Dimension Name	*Dimension Record Name*	*Description*
Return to Vendor line Status	D_RTVLN_STATUS	Stores return to vendor line statuses.
Shipment Status	D_SHIP_STATUS	Stores ship statuses.
Ship To	D_SHIPTO	Stores ship to locations.

Spend Data Mart

Every supply chain manager has an implicit goal to drive down unit cost without sacrificing product and service quality. But most managers do not have enough visibility into their supply chain to effectively manage the cost. The Spend data mart enables you to examine the source-to-settle process to isolate maverick spending, excessive purchase price variances, and contract compliance issues. With the Spend data mart, you are able to evaluate your supplier diversity programs to increase your government subsidies, look for opportunities to consolidate your supply base, and maximize your contract negotiations by leveraging supplier performance statistics. You will also gain visibility into spending across divisions and access, analyze and uncover information about spending patterns to help enhance relationships with existing suppliers, and initiate more profitable relationships with new suppliers. With Spend Mart you will be able to answer questions such as:

- With which suppliers do we spend the most
- On which supplies do we spend the most
- What departments have the most maverick spend
- With which minority owned businesses are we spending the most
- Why are invoices not being paid on time
- Which supplier and products have the largest purchase price variance

Spend Business Process

The Spend data mart is related to PeopleSoft's Procurement business process, which is also known as the *Source to Settle* business process. This business process is described with the Procurement data mart. The Spend data mart helps measure the dollars flowing through the Source to Settle process. This data mart enables you to analyze potential spending, for example, requisition amounts or open purchase order amounts, or invoiced amounts from suppliers, actual payments to suppliers, net discounts and adjustments.

Spend Data Mart Delivered Fact and Dimension Tables

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

Spend Data Mart Fact Tables

The following table describes the delivered Spend data mart fact tables.

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

Fact Name	*Fact Record Name*	*Description*	*Helps Answer*
Voucher	F_VCHR	Stores information relating to a voucher at the header level, such as payment and discount amounts. All measures are at the voucher level of detail.	Amount paid out against a voucher and associated discounts by business unit or by date. Number of payments made against a voucher. Amount of discount associated with payment for a voucher.
Voucher Line	F_VCHR_LN	Stores information relating to a voucher at the line level, such as voucher amounts, price variances, and amounts paid to date. All measures are at the voucher line level of detail.	PO price compared to the voucher price by business unit, buyer, or by date. PO price associated with a voucher line item. Voucher price associated with the voucher line item.

Spend Data Mart Dimension Tables

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

Dimension Name	*Dimension Record Name*	*Description*
Commodity Code	D_CMMDTY_CD	Standard code used throughout business industries that assists in categorizing or grouping procurement items.
Payment Status	D_PYMT_STATUS	Stores voucher payment statuses.
United Nations Standard Product and Service Codes	D_UNSPSC_CODE	Stores the standard product and service codes developed by Dun & Bradstreet for use in electronic commerce for data sourced from PeopleSoft.

Inventory Data Mart

Inventory management is an act of balancing cost and service. Maintaining large inventories is expensive and risky. But, small inventories may compromise your ability to satisfy a sudden increase in demand. The Inventory data mart provides supply chain performance analytics centered around these key inventory issues, such as demand, ability to meet demand, inventory turns, inbound supplies, quantities on hand, and other key metrics. These analytics can enable you to limit the direct costs of maintaining excess inventory, as well as the direct and indirect costs of not meeting the services levels required by your customers.

The Inventory data mart enables you to answer questions such as:

- How accurate is my inventory application count versus physical count
- Which inventory items have the best or worst accuracy
- cost of the inaccuracy
- value of inventory on hand
- Number of weeks of supply are on hand
- demand by item and by location
- inventory value of the demand
- Has the weeks of supply improved from last year
- How much lower or higher are the inventory turns from last year
- Has the inventory accuracy improved from last year
- Is there enough supply to meet demand
- status of the on hand stock
- Which items, and Number of, are on hold
- What are the fast and slow movers (items)
- What are the fast and slow mover (item) trends
- value of the fast and slow movers
- How often does inventory turn by item or by inventory location
- inventory value by item or by location
- Number of units of an item are currently in inventory
- Number of weeks of supply are available to meet current demand
- Number of units are on order to help meet demand

Inventory Business Process

The Inventory data mart supports multiple business process, including Order Fulfillment, Procurement, and Production, which is also known as *Plan to Produce* business process. The Order Fulfillment and Procurement business are described previously. With the Production business process, you can assess demand, determine supply requirements and allocate production capacity, as well as continually monitor and manage the efficiency of your production process.

Inventory Data Mart Delivered Fact and Dimension Tables

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

Inventory Data Mart Fact Tables

The following table describes the delivered Inventory data mart fact tables.

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

Fact Name	*Fact Record Name*	*Description*	*Helps Answer*
Inventory History Summary Ledger	F_HIST_SUM_LEDG	Stores information on Inventory History Summary ledger. Specifically, this fact table evaluates primary unit of measure quantity, unit of measure unit cost, extended cost, and transaction explanation.	Sum of the quantities on all of the transactions associated with each ledger. Sum of the amounts related to the quantities on all of the transactions associated with each ledger.
Inventory Cycle Count	F_INV_CYCLE_CNT	Stores information on inventory cycle count. Specifically, it stores inventory information at a point in time in which inventory was counted, as well as how often those events occurred. This fact table helps evaluate metrics such as Unit Cost and Adjusted Cost that are calculated during the various cycle counts for a specific invoice.	Total adjusted cost and adjusted quantities per cycle count by store, inventory item, or business unit. Adjusted quantity for a cycle count. Absolute value of a quantity that is adjusted. Unit cost for a cycle count. Adjusted cost associated with the adjusted quantity for a cycle count. Absolute value of the adjusted cost associated with the adjusted quantity for a cycle count.

Fact Name	*Fact Record Name*	*Description*	*Helps Answer*
Inventory Ledger	F_INV_LDGR	Stores information on Inventory ledger. With this fact you can evaluate primary unit of measure quantity, unit of measure unit cost, extended cost, and transaction explanation.	Total quantity of items involved in the transaction in the primary unit of measure. Sum of all of the quantities associated with a ledger transaction. Unit cost of each item measured in the primary unit of measure. Sum of the amounts for all transactions associated with a ledger. Total cost associated with each transaction on a ledger.
Inventory Transaction	F_INV_TRANS	Stores inventory transaction information, such as Demand, Received, Issued, Quantity on Hand, Backorder, Purchase Order, Work Order Received, Hard Committed, Soft Committed, Future Commit, Units in Transit, Units in Inspection, and Inter Unit Transfer Quantity and Amount. This fact table also helps evaluate item cost and Quantity on Hand Cost.	Count of the quantity demanded for a transaction. Number of units involved in a transaction. Quantity demanded for a transaction. Count of the quantity issued for a transaction. Count of the quantity acquired through interunit transfers for a transaction. Count of the quantity on the purchase order for a transaction. Count of quantity as a soft commit on a work order. Quantity on loan to manufacturing.
Physical Inventory	F_PHYSICAL_INV	Stores information on physical inventory measures, such as Open, Received, Hold, and Rejected Quantities. The measures are by inventory location, item and date dimensional criteria. That is, you can evaluate physical inventory by its location, by a specific item, or by a specific date or date range, or by all of these criteria.	Number of available items for a transaction. Number of restricted items not available for a transaction. Number of items on hold for a transaction. Number of rejected items on a transaction.

Inventory Data Mart Dimension Tables

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

Dimension Name	*Dimension Record Name*	*Description*
Country of Origin	D_CNTRY_OF_ORIG	Stores country of origin.
Inventory Status Reason	D_INVST_REASON	Stores inventory status reason.
Item Branch	D_ITEM_BR	Stores item branch.
Pack Size	D_PACK_SIZE	Stores pack size.
Stock Type	D_STOCK_TYP	Stores stock types.

Manufacturing Data Mart

In today's manufacturing environments, one of the challenges facing product managers, manufacturing managers, and operations analysts is understanding how production and machine schedules will meet orders and forecasted demand across multiple plants or production lines. The Manufacturing data mart enables you to analyze your production schedule effectively to make informed decisions and ultimately optimize plant, equipment, and production investments. The capability provides the visibility into cost, reliability, timeliness, and efficiency of your manufacturing operations. Armed with this level of insight, you will be able to determine planned production downtime and product availability, calculate scrap costs for plant-level items, and capture production lead times by work order. With the Manufacturing data mart, you can answer questions such as:

- How effectively is production capacity being utilized
- How much time does it take to complete work orders
- scrap cost per product Per business unit
- How much downtime can be scheduled for maintenance
- What are the manufacturing lead times by work order

Production Business Process

The Manufacturing data mart supports the Production business process, which is also known as the *Plan to Produce* business process. This business process is described previously. The Production business process includes a production plan or schedule that identifies the type and amount of products to be manufactured over a given time period. Depending on the type of production, work orders or schedules are issued to the shop floor to commence production. Production is tracked through the production line until the order is complete. The completed production is either placed into inventory (make-to-stock) or may be shipped directly to the customer (make-to-order).

Manufacturing Data Mart Delivered Fact and Dimension Tables

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

Manufacturing Data Mart Fact Tables

The following table describes the delivered Manufacturing data mart fact tables.

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

Fact Name	*Fact Record Name*	*Description*	*Helps Answer*
Production Costs	F_PRDN_COST	Stores information on work order production costs.	Cost of producing a product, and which components are the most costly. Standard number of units that can be produced, measured in the primary unit of measure. Actual cost associated with the actual number of units that can be produced, measured in the primary unit of measure. Planned number of units that can be produced, measured in the primary unit of measure. Cost associated with the planned number of units that can be produced, measured in the primary unit of measure. Number of scrapped units that can be produced, measured in the primary unit of measure.

Fact Name	*Fact Record Name*	*Description*	*Helps Answer*
Work Order Master	F_WO_MASTER	Stores information on work orders such as the date and quantity of work order.	Number of work orders due for a specific date, percent complete of those orders, and amount of scrap being produced Number of units requested on the work order, measured in the primary unit of measure. Date a work order was ordered. Date a work order started. Date a work order was completed.
Work Order Part List	F_WO_PARTLIST	Stores part list information for work orders.	Do I have enough parts to complete the scheduled production at a work center. Available part list quantity available. Quantity ordered for the part list measured in the primary unit of measure.
Work Order Routing	F_WO_ROUTING	Stores information on work order routing such as queue time, runtime, and labor time.	Standard number of hours required to move units per work order. Standard number of hours that the item on a work order is in the queue. Standard number of hours that a machine is required to run for a work order. Standard number of hours required to set up the labor for a work order. Actual number of machine work hours reported for a work order. Actual number of labor work hours reported for a work order.
Work Order Time Transaction	F_WO_TIME_TR	Stores information on the time transaction of work orders.	Do I have enough work hours to complete the scheduled production at a work center. Capacity of the work of a work center on a given day.

Fact Name	Fact Record Name	Description	Helps Answer
Work Center Master	F_WRKCNTR_MST	Stores information related to work center (work center is analogous to a production area).	Work center effectiveness and total capacity of a work center. Utilization capacity at which a work center can be used. Rate of efficiency at which a work center can be used. Standard capacity at which a work center can be used.
Work Center Resource Unit	F_WRKCNTRESUT	Stores resource information related to a work center.	Capacity of a work center on a given day.

Manufacturing Data Mart Dimension Tables

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

Dimension Name	Dimension Record Name	Description
Category 2	D_CATEGORY_02	Stores Category 02 information; indicates the type or category of a work order.
Category 3	D_CATEGORY_03	Stores Category 03 information; indicates the type or category of the work order.
Component by Product	D_CO_BY_PRDCT	Stores component by product. A code that distinguishes standard components or ingredients from co-products, byproducts, and intermediates. Co-products are (concurrent) end items as the result of a process. Byproducts are items that can be produced at any step of a process, but were not planned. Intermediate products are items defined as a result of a step, but are automatically consumed in the following step.
Cost Type	D_COST_TYPE	Stores cost types. It includes a code that designates each element of cost for an item.
Dispatch Group	D_DISPTCH_GRP	Stores dispatch groups. It includes a category code used to group work centers within an over business unit.

Dimension Name	*Dimension Record Name*	*Description*
From Grade	D_FROM_GRADE	Stores from grade. It indicates the minimum grade that is acceptable for an item.
Hour Type	D_HOUR_TYPE	Stores hour types. It includes a code that indicates the type of time entered. Valid values are: • 1, Run Labor Hours • 2, Setup Labor Hours • 3, Machine Hours • 4, Quantities Completed • 5, Quantities Scrapped • 9, Miscellaneous (such as piece rate bonus)
Operation Type	D_OPERTN_TYPE	Stores operation types. It indicates the type of operation. Valid values include: • A - Alternate routing • TT- Travel time • IT - Idle time • T- Text
Phase	D_PHASE	Stores work order phase. It indicates the current stage or phase of development for a work order. You can assign a work order to only one phase code at a time.
Planner	D_PLANNER	Stores manufacturing planner - the address book number of a manager or planner.
Prime Load Code	D_PRIME_LOAD_CD	Stores prime load. It determines if a work center is machine or labor intensive. The system also uses prime load codes in resource requirements planning and capacity requirements planning calculations to develop load profiles.
Parent Child Relationship	D_PRNT_CHLD_REL	Stores the parent-child relationship between work order items. It specifies whether the system displays parent records or child records.

Dimension Name	*Dimension Record Name*	*Description*
Shift Code	D_SHIFT_CD	Stores manufacturing shift codes. It identifies daily work shifts.
Supervisor	D_SUPERVISOR	Stores supervisor, including the address book number of the supervisor.
Time Basis	D_TMEBASIS_CD	Stores manufacturing time basis codes. It indicates how machine or labor hours are expressed for a product. Time basis codes identify the time basis or rate to be used for machine or labor hours entered for every routing step.
Variance Flag	D_VARIANCE_FLAG	Stores the manufacturing variance flag. 1 indicates that this work order has had work order activity run against it in update mode. 2 indicates that this work order has had the appropriate variances calculated and reported to the general ledger.
Work Order Status	D_WO_STATUS	Stores work order status. It describes the status of a work order, rate schedule, or engineering change order. Any status change from 90 through 99 triggers the system to automatically update the completion date.
Work Order Type	D_WO_TYPE	Stores work order types. It indicates the classification of a work order or engineering change order.
Work Date	D_WORKDATE	Stores manufacturing work dates.

Supply Chain Planning Data Mart

Supply Chain Planning applications identify the right supply of materials to the right place, at the right time, at the lowest cost. The supposed optimization algorithms match product demand with supply to determine the most cost effective method to fulfill the requirements. The plan may recommend transfers of supply from other inventory locations, new purchases from suppliers, and the production of new products to meet the demand. If products must be produced, the planning application may also recommend the appropriate manufacturing sequence. Often, these recommended plans are altered to fit the conditions "on the floor." These deviations cause the performance-against-plan to deteriorate and in extreme cases, make the plan unusable. To mitigate this problem, PeopleSoft has developed the Supply Chain Planning data mart to provide visibility into the *compliance* of the plan to the actual transfer, purchase, or production. Supply Chain Planning Mart enables you to analyze the deviation from the recommended plan to actual order fulfillment, purchasing, production, inter unit transfers, inventory levels, and capacity utilization.

The Supply Chain Planning data mart helps you answer questions such as:

- How much has the plan changed from actual execution
- Which products are most out of compliance
- Which business units change the plan the most often

Production Business Process

The Supply Chain Planning data mart supports the Production business process, which is also known as the *Plan to Produce* business process.

Supply Chain Planning Data Mart Delivered Fact and Dimension Tables

This section discusses the delivered fact and dimension tables for the Supply Chain Planning data mart.

Supply Chain Planning Data Mart Fact Tables

The following table describes the delivered Supply Chain Planning data mart fact tables.

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

Fact Name	*Fact Record Name*	*Description*	*Helps Answer*
Capacity	F_SCP_CAPACITY	Captures the daily capacity utilization associated with production at a specific work center. All the measures are 'bucketed' by time to match planning information. The grain of the fact table is captured at the work center daily production level.	Planned production capacity for a work center and actual production capacity for a business unit. Actual utilized capacity in time units. Actual production capacity in time units. Planned utilized capacity in time units.
Inventory	F_SCP_INVENTORY	Captures daily inventory quantities. All the measures are 'bucketed' by time to match planning information. The grain of the fact table is captured at the daily inventory balance level.	Planned inventory quantity. Actual ending inventory quantity.
Production	F_SCP_PRODN	Captures the quantities and costs associated with production against a specific work order. All the measures are 'bucketed' by time to match planning information. The grain of the Production fact table is captured at the completed work orders level.	Planned production quantity for a work center and actual production quantity for a business unit. Actual production quantity at each work center. Actual production unit cost. Planned production quantity.
Purchasing	F_SCP_PURCHASNG	Captures the quantities and costs associated with purchase orders. All the measures are 'bucketed' by time to match planning information. The grain of the fact table is captured at the received and accepted purchase order line level.	Planned purchasing quantity for a business unit on a particular date and actual purchasing quantity for a supplier for a particular item. Actual purchase order quantity for each line item. Actual voucher cost. Planned purchase order quantity. Quantity accepted on the purchase order receipt. Quantity received on the purchase order receipt. Quantity rejected on the purchase order receipt.

Fact Name	Fact Record Name	Description	Helps Answer
Sales Forecast	F_SCP_SLS_FCST	Captures the product quantities associated with shipments against sales orders and how they compare with predictions based on historical analysis. All the measures are 'bucketed' by time to match planning information. The grain of the Order Shipment fact table is captured at the per item sales for a particular forecasting bucket level.	Planned sales quantity for a specific forecast time bucket. Predicted sales quantity for a specific time period.
Transfers	F_SCP_TRANSFERS	Stores information on the quantities associated with a transfer order between inventory business units. All the measures are 'bucketed' by time to match planning information. The grain of the fact table is captured at the received inventory transfer order level.	Planned inter unit transfer quantities for the business unit and actual inter unit transfer quantities for an item on a specific date. Planned inter unit transfer quantity. Actual inter unit transfer quantity.
Sales Actual	F_SCPSLS_ACTUAL	Stores information on actual sales facts. All the measures are 'bucketed' by time to match planning information.	Actual sales quantity for a particular customer. Actual shipped quantity on a specific sales order.

Supply Chain Planning Data Mart Dimension Tables

The following table describes the delivered Supply Chain Planning data mart dimension tables.

Dimension Name	Dimension Record Name	Description
Bucket	D_BUCKET	Stores supply chain time bucket information.
Work Center	D_WORK_CENTER	Stores Supply Chain Planning related manufacturing work center.

Shared Dimensions

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

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

Dimension Name	*Dimension Record Name*	*Description*
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.
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.

Dimension Name	*Dimension Record Name*	*Description*
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.
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.

Dimension Name	*Dimension Record Name*	*Description*
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.

Dimension Name	*Dimension Record Name*	*Description*
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.

Dimension Name	*Dimension Record Name*	*Description*
Subledger	D_SUBLEDGER	Stores information on subledger, which groups the accounting information.
Supplier	D_SUPPLIER	Stores information on suppliers, such as remit to supplier and corporate supplier.
Sales Territory	D_TERRITORY	Stores sales territory information. Sales territories are user defined sales regions independent of geography or proximity.
Unit	D_UNIT	Stores detail information on real estate properties.
Unit of Measure	D_UOM	Indicates the quantity in which an inventory item is expressed, such as case (CS) or box (BX).

Chapter 4

Running SCM Warehouse Implementation Jobs

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

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

Prerequisites

After you have configured IBM WebSphere DataStage for EPM, and before you run SCM 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 SCM Warehouse.
- Run the Dimension Mapper setup jobs.
- Run shared lookup jobs.
- Run the Setup - OWE jobs.
- Run the Common Dimensions jobs.

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

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

- OWS
- GLOBAL_DIMENSION
- LOCAL_DIMENSION
- Global_D00

- OWE_E

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

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

See Appendix A, "Using the PeopleSoft EPM Lineage Spreadsheets," page 167.

Running SCM Warehouse Implementation Jobs

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

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

Running SCM - OWS Jobs

The first step in implementing the SCM Warehouse is to run the SCM - OWS jobs. These jobs consist of SCM-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 *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Running Initial Setup Jobs."

SCM - OWS Hash File Jobs

Perform the following steps to run the SCM - 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: *SCM_E, OWS, Base, Load_Hash_Files, Server*.
2. Select each SCM - 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.*

SCM - OWS Jobs

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

1. In IBM WebSphere DataStage Director, navigate to the SCM - OWS jobs by expanding the nodes in the left navigation panel using the following path: *SCM_E, OWS, Base, Load_Tables, Sequence.*
2. Select each SCM - 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 SCM

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

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

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

Local Dimension Hash File Jobs

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

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

   The Job Run Options box appears.
3. Update the job parameters if necessary and click Run.

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

Local Dimension Jobs

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

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

The fourth step in implementing the SCM Warehouse is to run the Global - OWE jobs for SCM. These jobs consist of SCM 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 SCM 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 SCM - OWE Jobs

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

SCM - OWE Hash File Jobs

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

1. In IBM WebSphere DataStage Director, navigate to the SCM - OWE hash file jobs by expanding the nodes in the left navigation panel using the following path: *OWE_E, SCM, Base, Load_Hash_Files, Server.*

2. Select the each SCM - 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.*

SCM - OWE Jobs

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

1. In IBM WebSphere DataStage Director, navigate to the SCM - OWE jobs by expanding the nodes in the left navigation panel using the following path: *OWE_E, SCM, Base, Load_Tables, Sequence.*

2. Select each SCM - 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 SCM SKU Jobs

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

Note. You can run SCM SKU jobs individually or together using the master sequence job.

SCM SKU Hash File Jobs

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

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

SCM Dimension Jobs

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

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

SCM Fact Jobs

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

1. In IBM WebSphere DataStage Director, navigate to the SCM fact jobs by expanding the nodes in the left navigation panel using the following path: *SCM_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: *SCM_E, [SKU/Data Mart Name], [Business Process], Master_Sequence.*

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

   The Job Run Options box appears.

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

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

Chapter 5

Configuring Slowly Changing Dimensions

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

Understanding Slowly Changing Dimensions

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

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

There are three types of slowly changing dimensions:

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

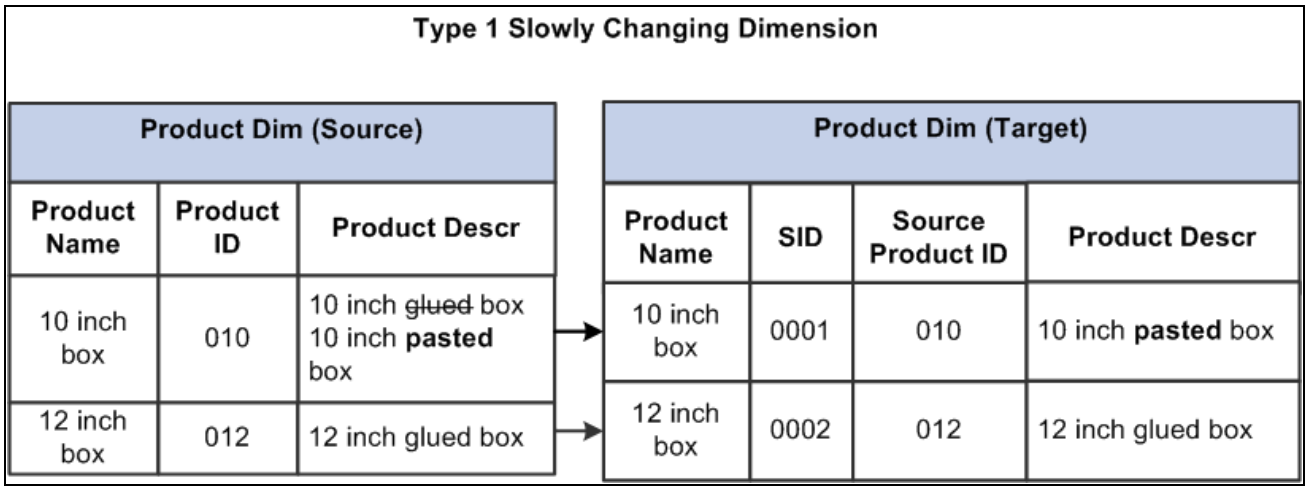
Note. PeopleSoft does not support type 3 slowly changing dimensions.

Type 1 Slowly Changing Dimensions

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

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

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



Type 1 slowly changing dimension

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

Type 2 Slowly Changing Dimensions

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

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

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

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

Type 2 Slowly Changing Dimension

Product Dim (Source)		
Product Name	Product ID	Product Descr
12 inch box	012	12 inch glued box
10 inch box	010	10 inch ~~glued~~ box
		10 inch **pasted** box

Product Dim (Target)					
SID	Source Product ID	Product Name	Product Descr	EFF_ START_DT	EFF_ END_DT
0001	012	12 inch box	12 inch glued box	Jan-01-1753	Dec-31-9999
0002	010	10 inch box	10 inch glued box	Jan-01-1753	May-12-06
0003	**010**	**10 inch box**	**10 inch pasted box**	**May-12-06**	**Dec-31-9999**

Type 2 slowly changing dimension

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

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

Type 3 Slowly Changing Dimensions

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

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

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

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

Understanding Slowly Changing Dimensions in EPM

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

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

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

Valid Date Range Subrecord

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

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

EFF_START_DT and EFF_END_DT

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

For type 2 slowly changing dimensions, the EFF_START_DT and EFF_END_DT columns serve to partition the related dimension records and indicate which version is active. When a changed record is extracted into an MDW dimension table, the new record is assigned an EFF_START_DT value, which is derived from the EFFDT column. The old record is assigned an EFF_END_DT value equal to the new EFFDT minus one day (EFFDT - 1 day = 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 have default value of 'Y'.

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

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

- Source Query
- Target DRS Stage
- Target Lookup Stage

Source Query

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

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

Target DRS Stage

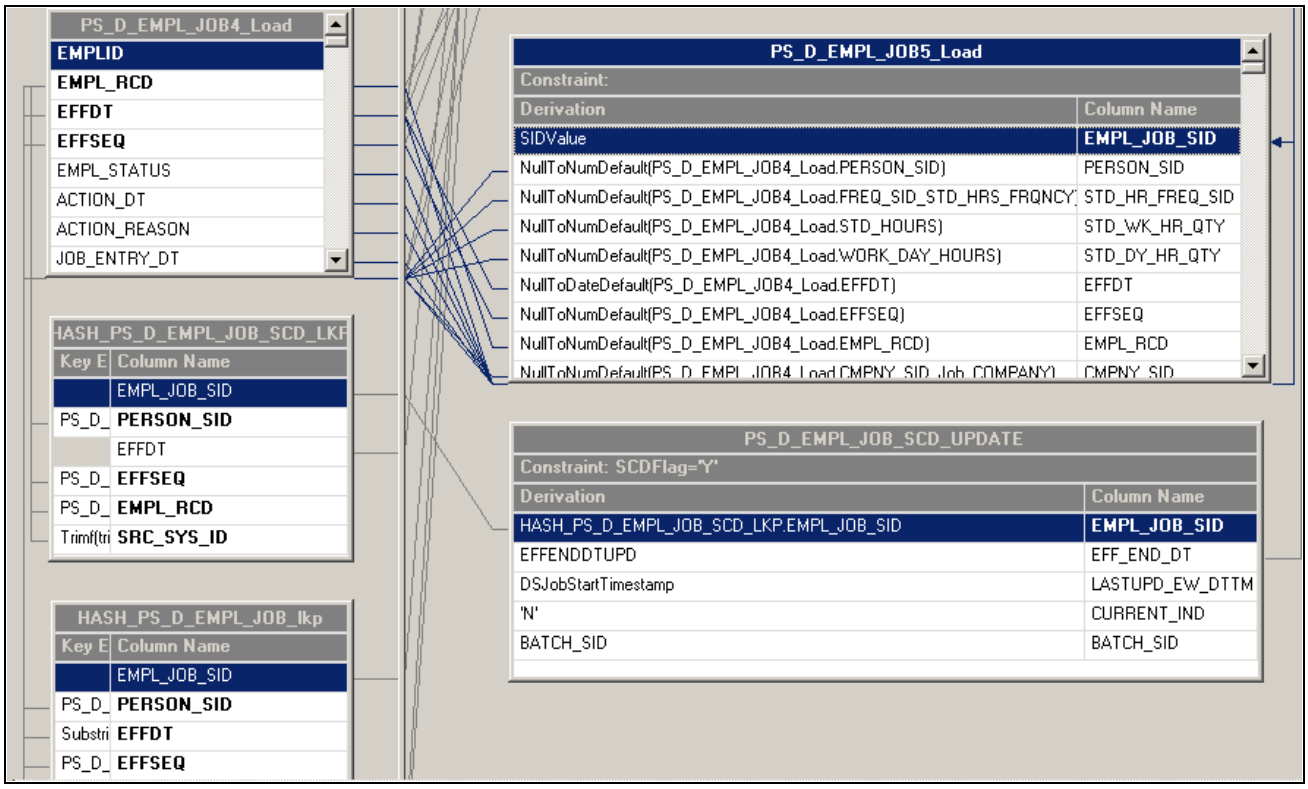
There is only one target DRS stage for a type 1 slowly changing dimension and it uses an *update existing rows* or *insert new rows* logic for its loading strategy.

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

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

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

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



Slowly changing dimension and target DRS stage

Target Lookup Stage

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

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

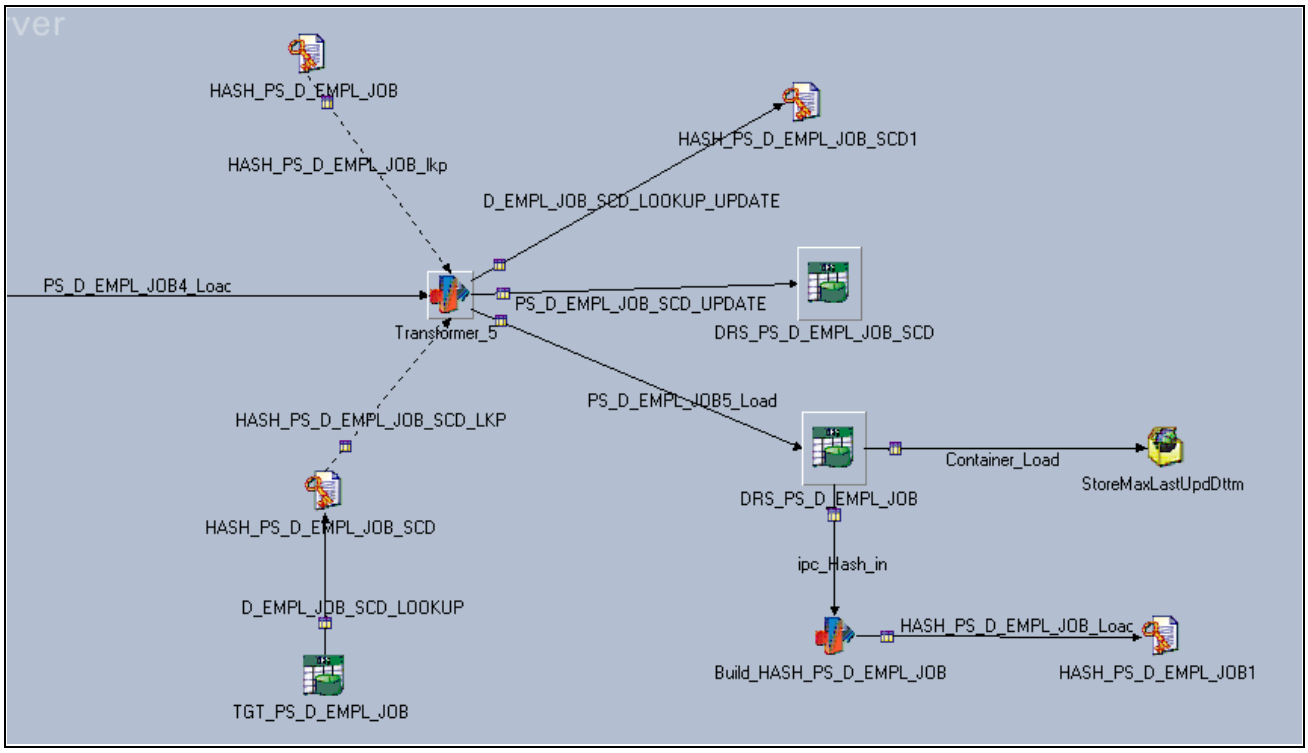
- The first target lookup stage retrieves the latest SID in case the incoming rows are already there in the target dimension table.

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

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

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

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



Slowly changing dimension and target lookup stage

Fact Table Jobs and Slowly Changing Dimensions

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

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

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

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

See Chapter 5, "Configuring Slowly Changing Dimensions," Converting a Hash File Lookup to a Dynamic DRS Lookup in the Related Fact Table Job, page 79.

Converting Type 1 Slowly Changing Dimension Jobs to Type 2

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

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

Overview

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

Converting a Hash File Lookup in a Related Fact Table Job

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

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

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

See Chapter 5, "Configuring Slowly Changing Dimensions," Converting a Hash File Lookup to a Dynamic DRS Lookup in the Related Fact Table Job, page 79.

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.

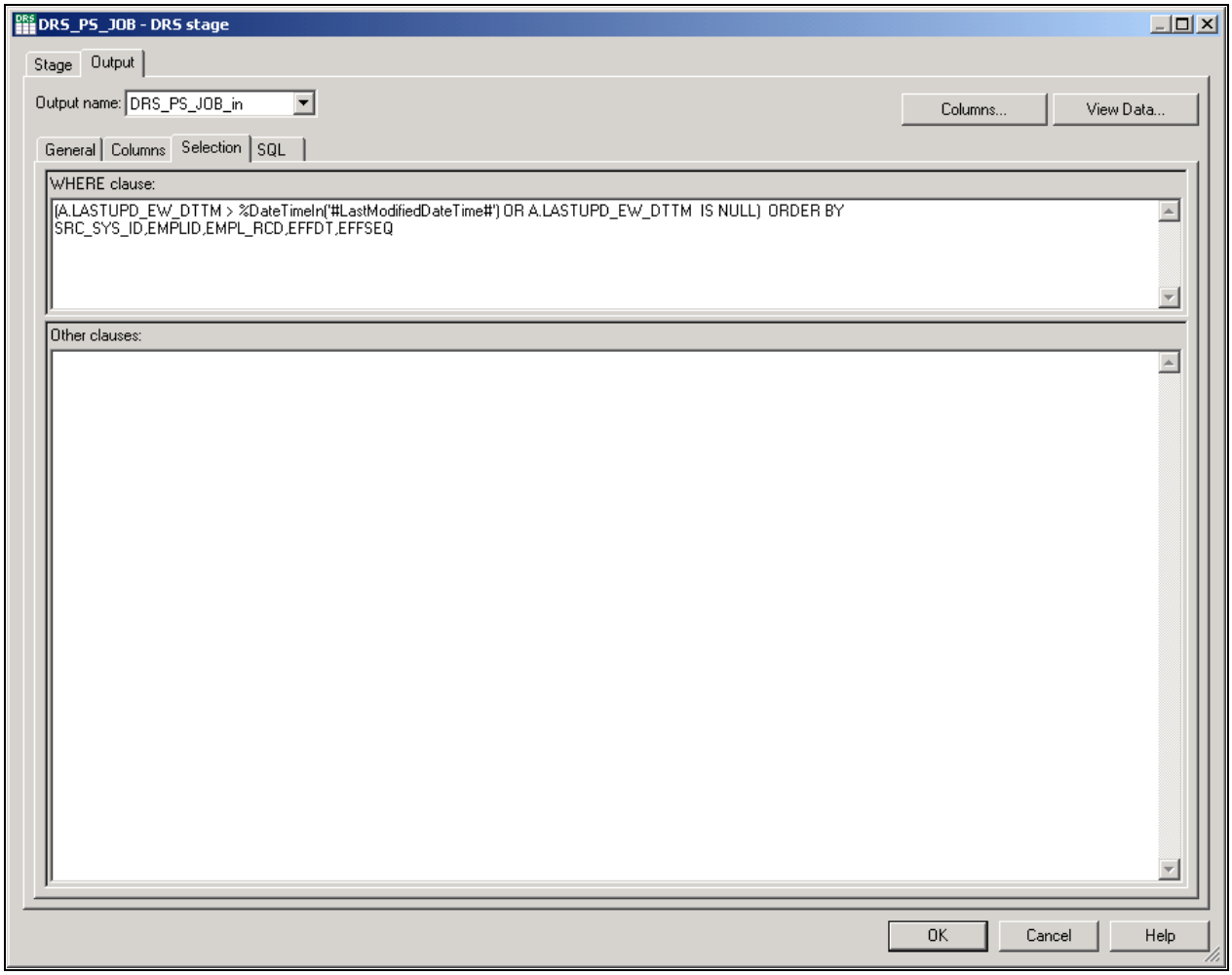
Note. Most of the Type 1 dimension jobs have the correlating sub-query to get the latest effective dated dimensional record.

---

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



DRS stage - Output tab

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

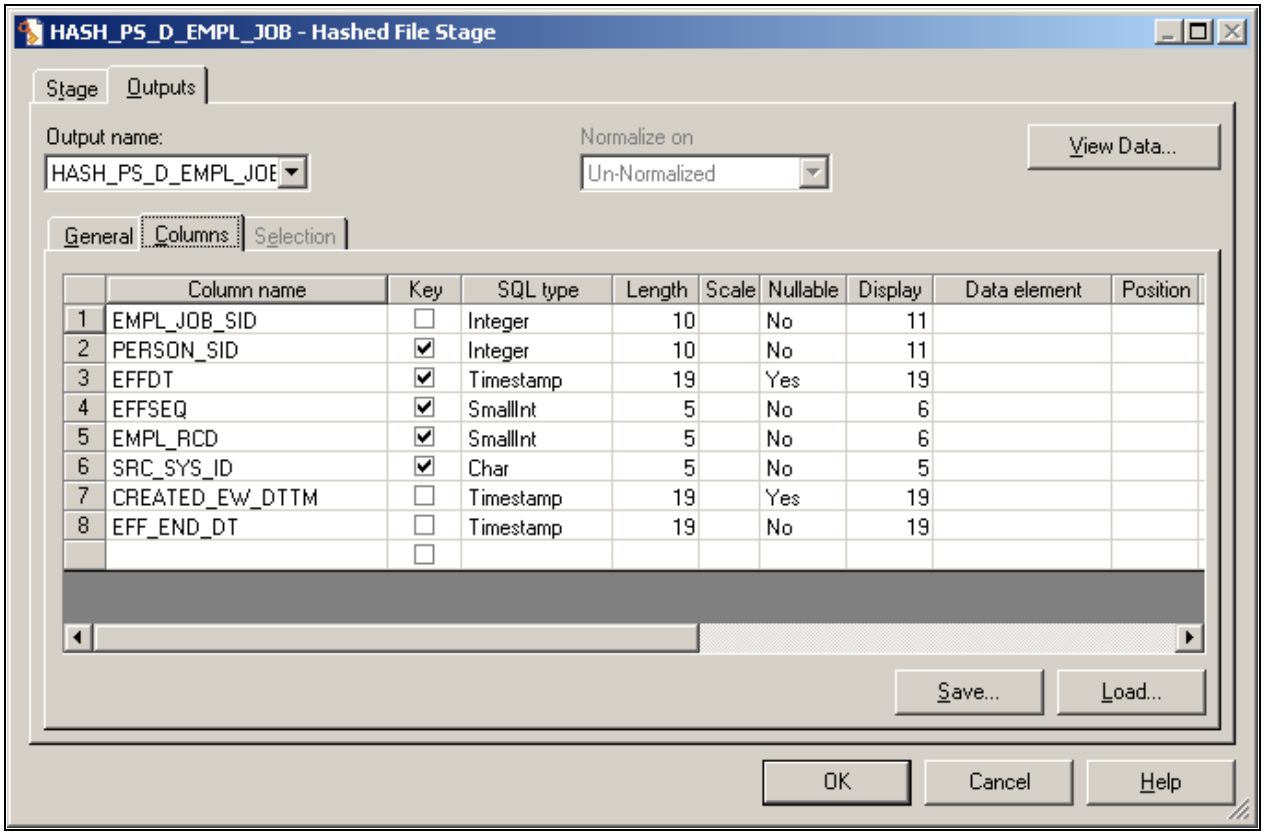
Perform the following steps to modify the target hash lookup stage:

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.



Transformer stage - Output tab

6. Click OK.

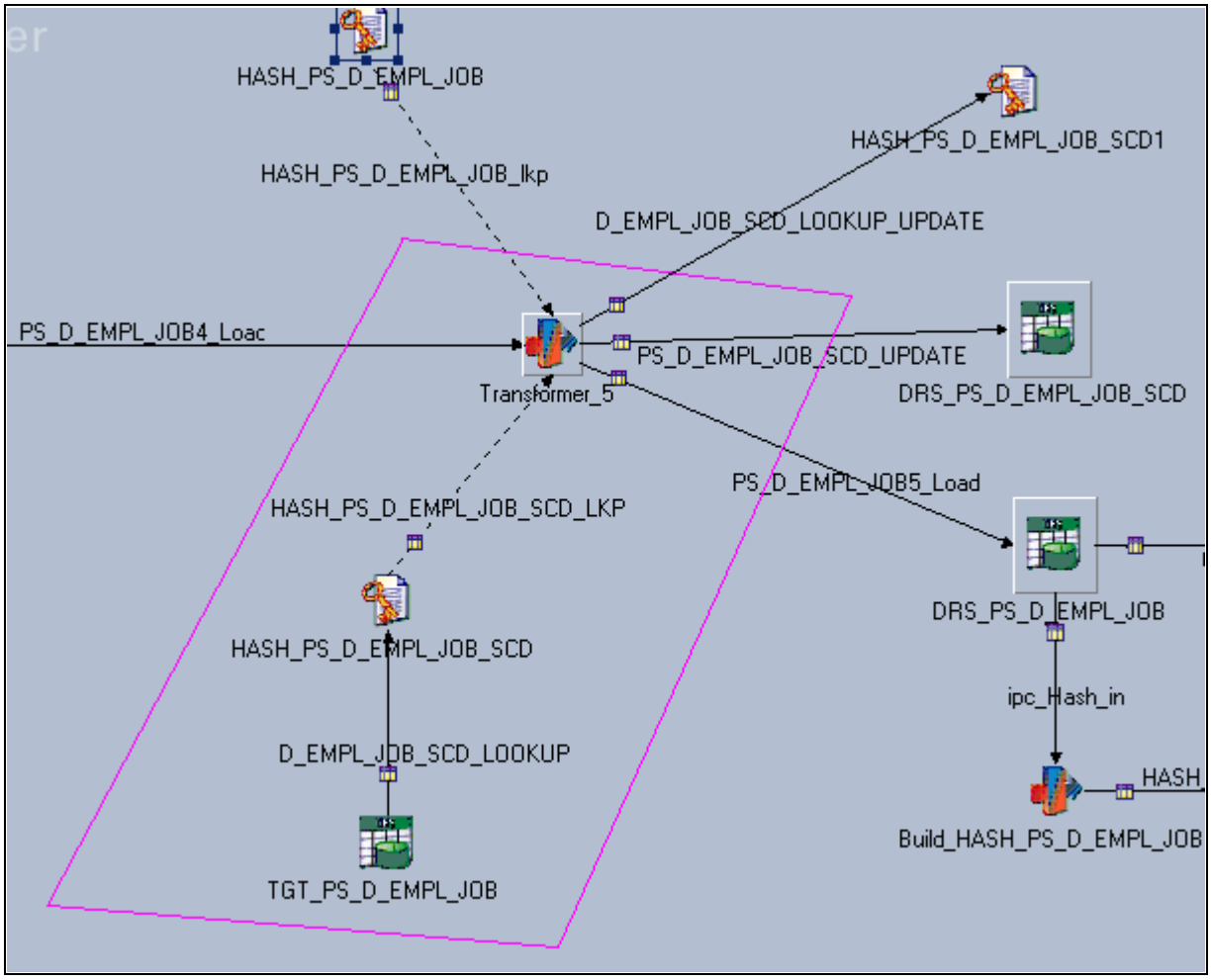
Step 3: Adding Lookup Stages to Identify Slowly Changing Dimension Logic

Perform the following steps to add lookup stages:

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

2. Link the DRS stage to the newly added Hash File stage.

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



Adding lookup stages

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

---

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

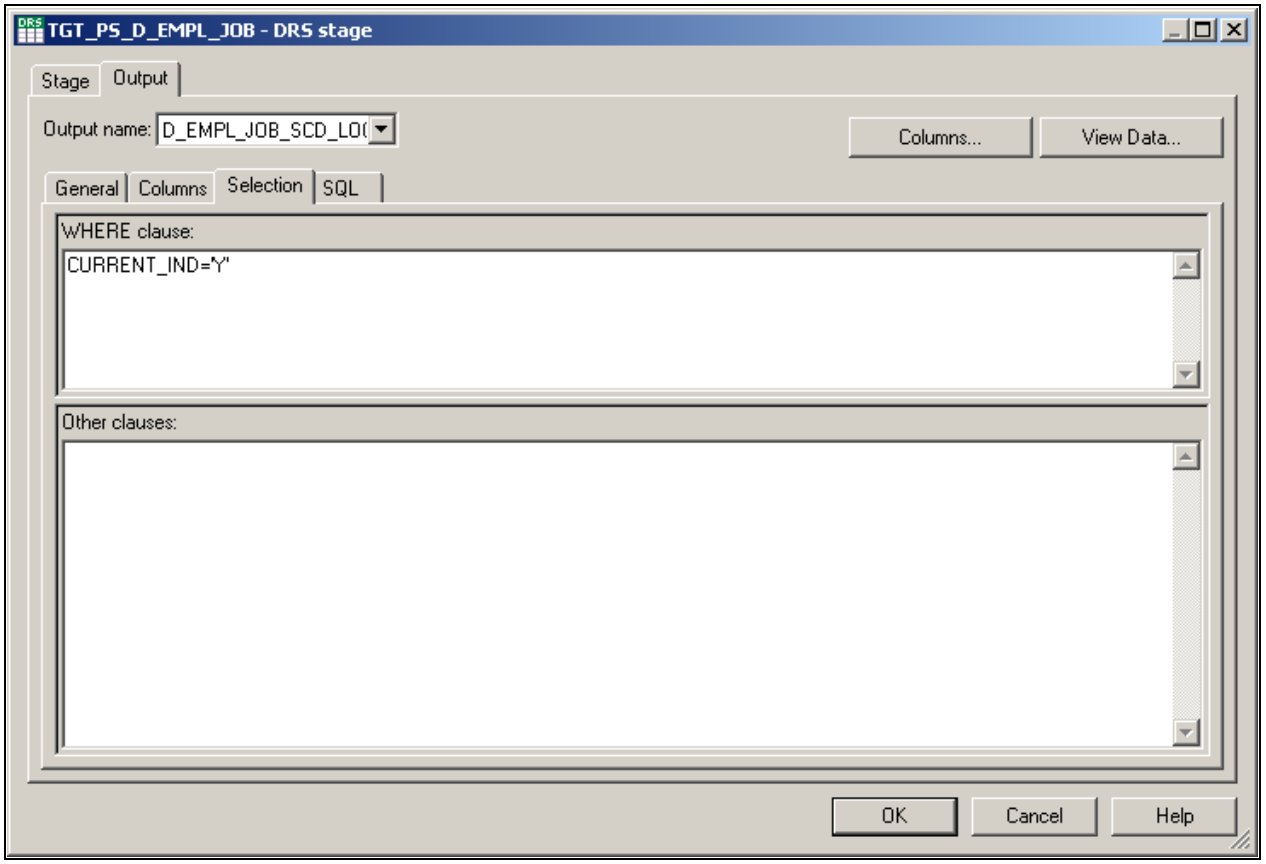
5. Click OK.

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

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

1. Locate the new DRS stage and open it.

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



DRS stage - Output tab

3. Add the following WHERE condition to get the most recent SID value:

   CURRENT_IND = 'Y'

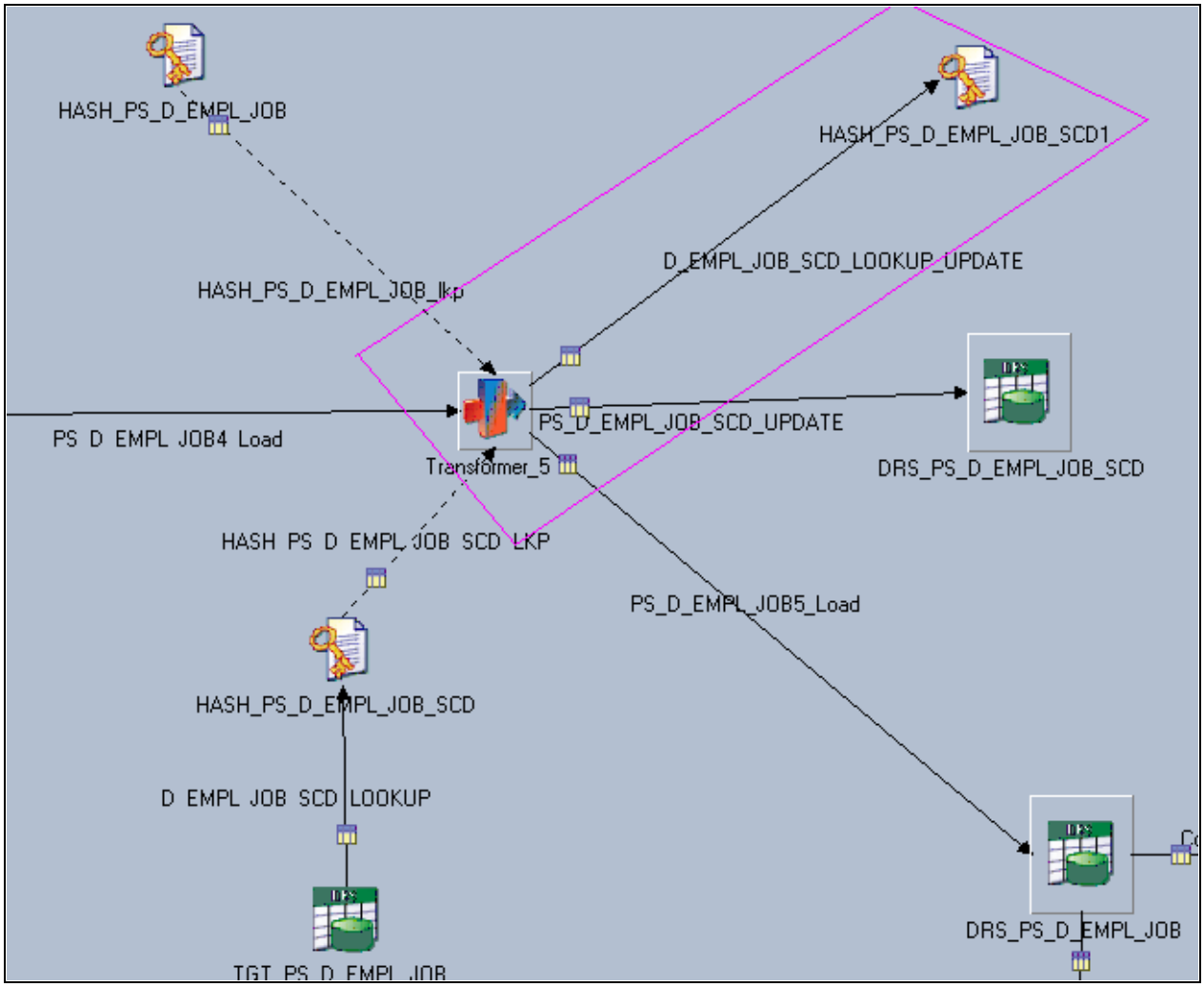
4. Specify the database connection parameters in the General tab of the DRS stage.
5. Click OK.

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

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

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

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



Adding a mew hash file stage

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

---

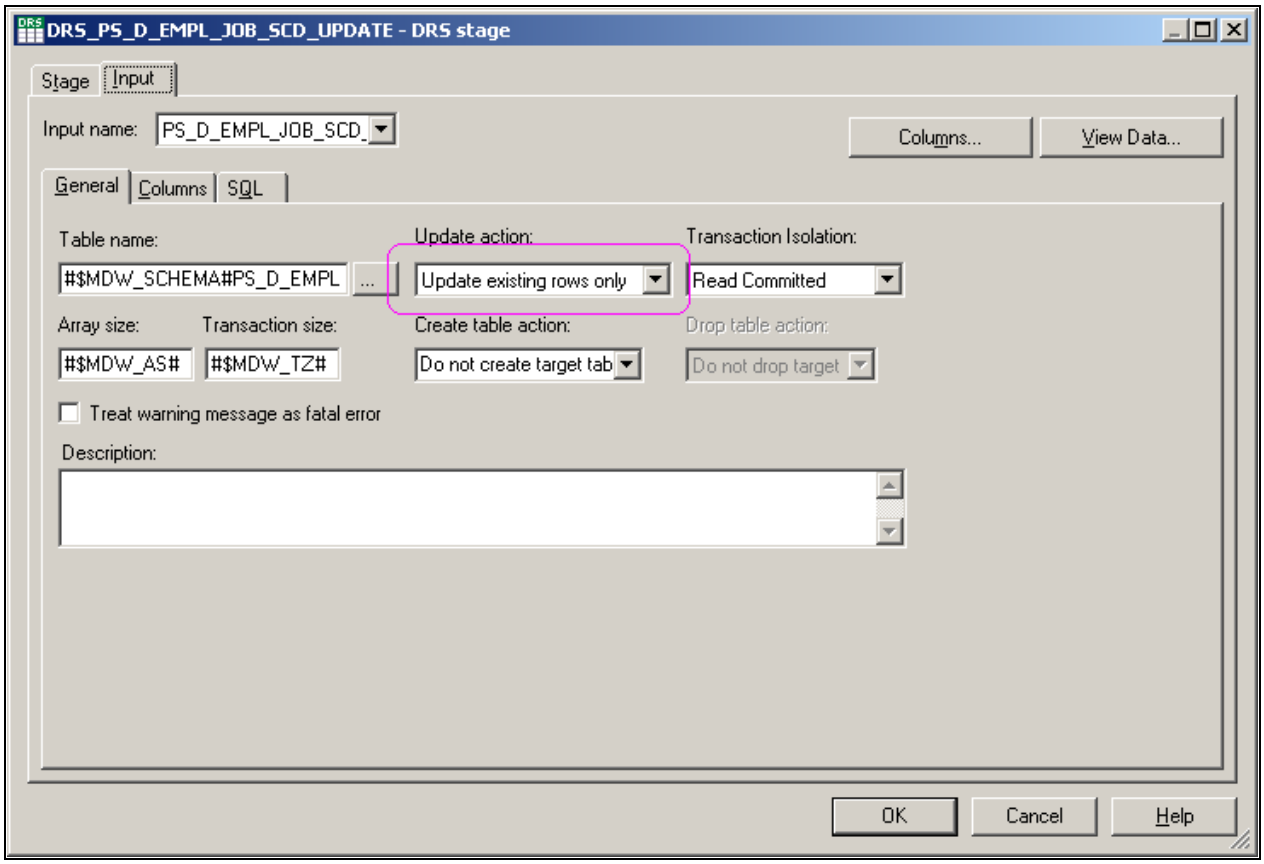
3. Change the hash file name by adding the suffix *_SCD* to it.
4. Click OK.

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

Perform the following steps to add a target DRS stage:

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

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

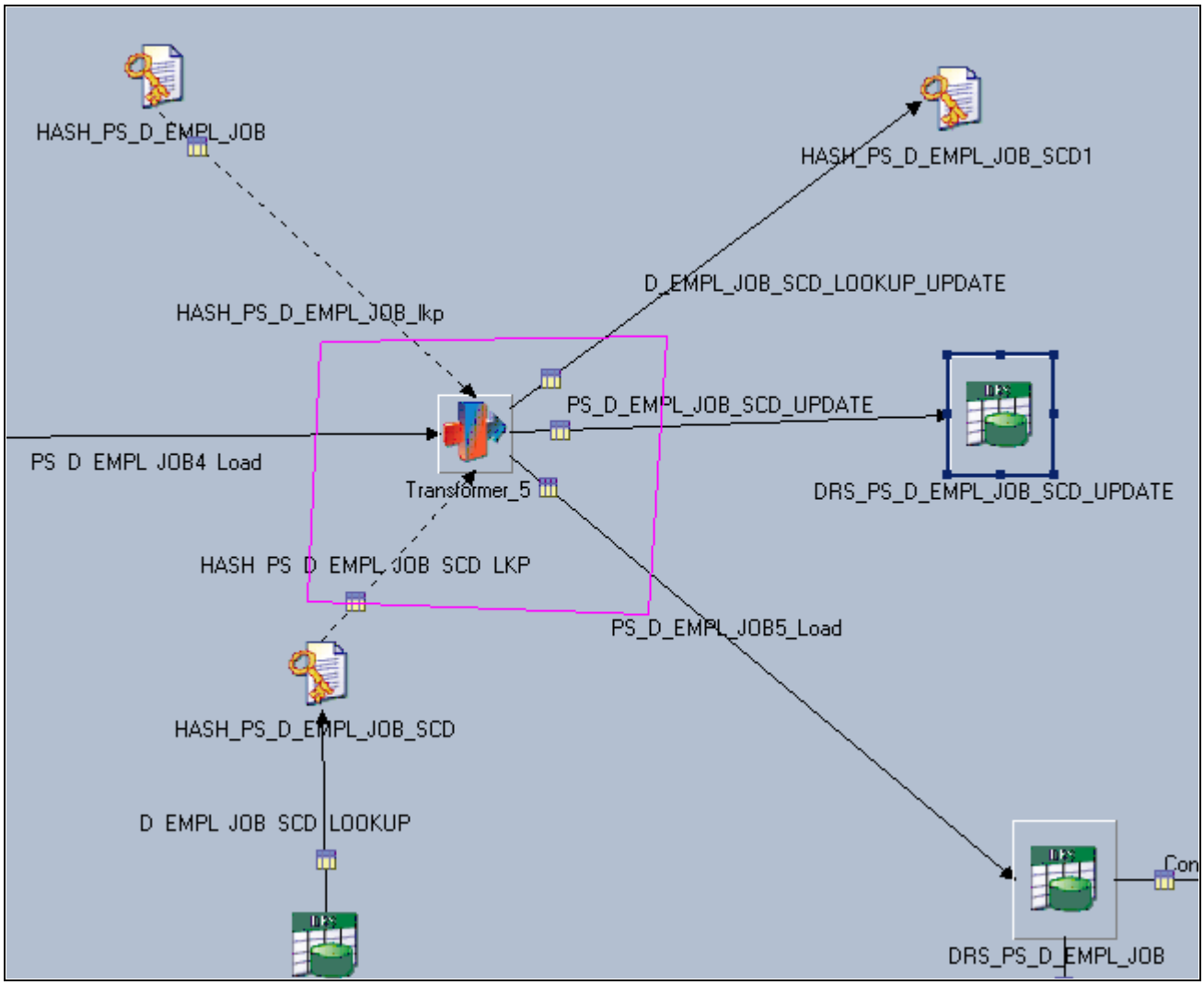


DRS stage - Input tab

6. Click OK.

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

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



Verifying links

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

Perform the following steps to add stage variables:

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

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

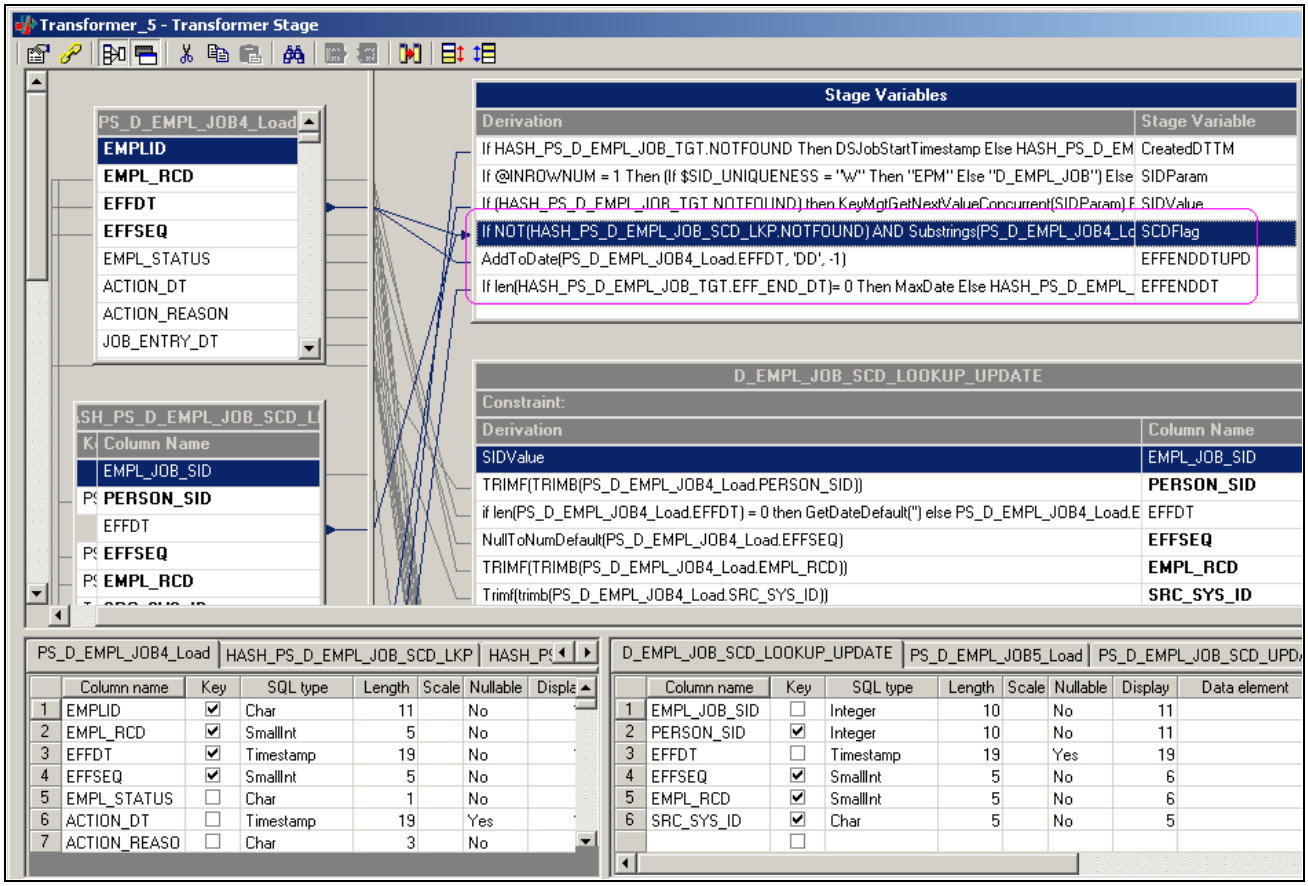
5. Add a new EFFENDDTUPD stage variable to the transformer, using the following expression:

   AddToDate(<INPUT_LINK_NAME>.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.



Adding stage variables

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

Perform the following steps to modify column expressions:

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

   EFF_START_DT = <INPUT_LINK_NAME>.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:

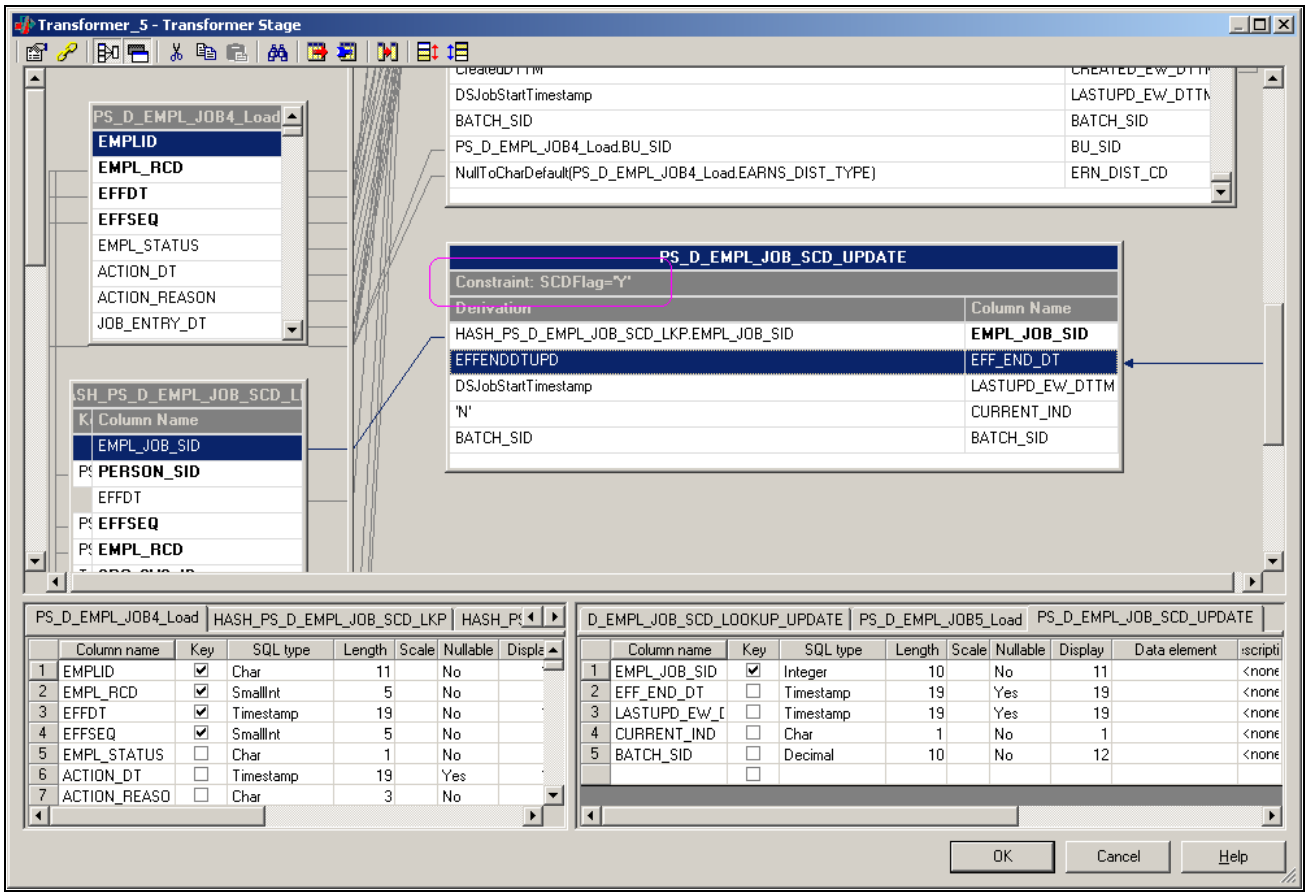
   LASTUPD_EW_DTTM = DSJobStartTimestamp
10. Modify the expression for the CURRENT_IND column as follows:

    CURRENT_IND = 'N'
11. Modify the expression for the BATCH_SID column as follows:

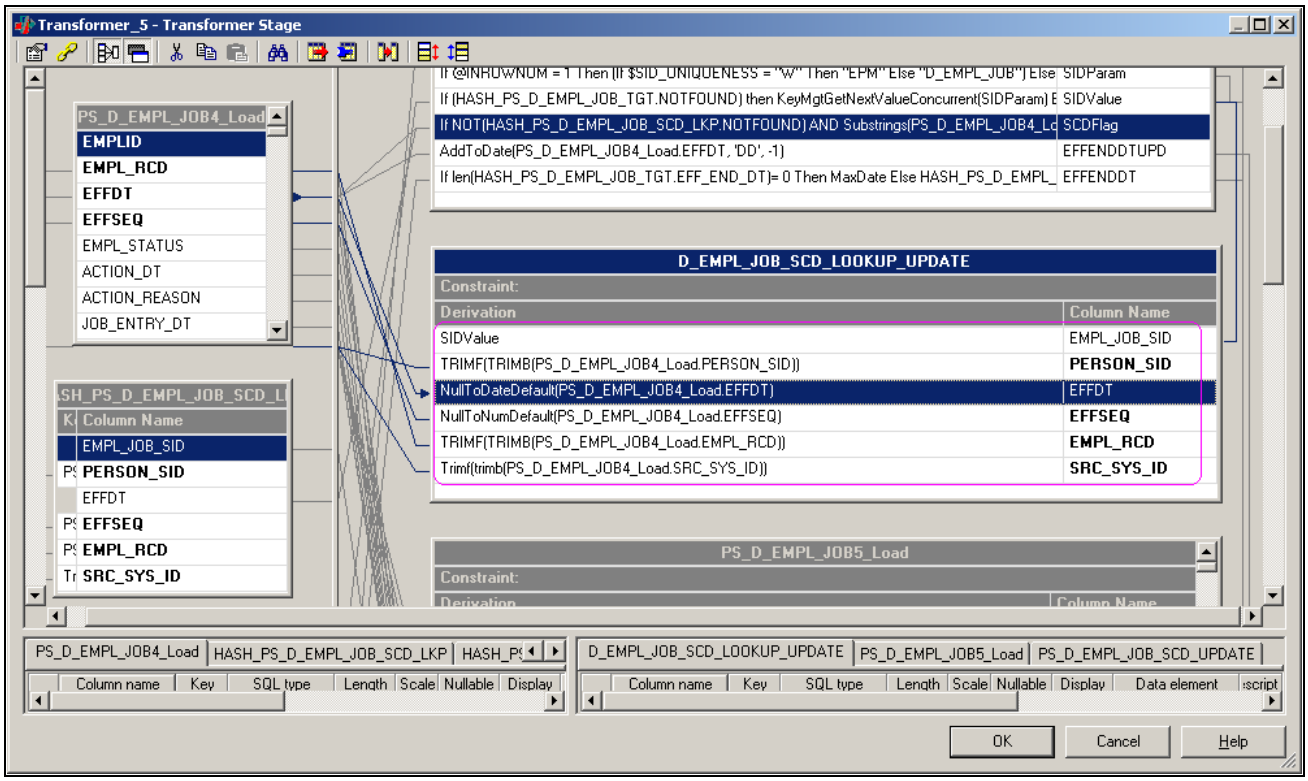
    BATCH_SID = BATCH_SID
12. Add the following constraint to the link so that the EFF_END_DT of the old dimension record is updated:

    SCDFlag = 'Y'
13. Locate the output link that updates the new Hash file (for example, [*hash file name*]_SCD).
14. Map the alternate key columns from the input link to alternate keys in the lookup table.

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



Modifying column expressions, 1 of 2



Modifying column expressions, 2 of 2

Step 10: Compiling the job

Perform the following steps to compile the job:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

   If your hash file is populated by the DRS stage in the same job, delete the DRS stage (including the link) and replace the hash file stage with a DRS stage.
4. Open the DRS stage for editing.
5. In the Stage tab, select the General sub-tab and specify the database connection parameters.
6. In the Output tab, select the General sub-tab and specify the corresponding table name (the table name should always include the schema name as its prefix).

   Specify the appropriate job parameter for array size and change query type to *User-defined SQL query*.
7. Select the Columns sub-tab and specify parameters for EFF_START_DT and EFF_END_DT.

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

   For example,

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

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

Chapter 6

Implementing Currency Conversion for Multiple Currencies

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

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

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

Understanding Currency Conversion

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

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

MDW Table Structure Used to Support Currency Conversion

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

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

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

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

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

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

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

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

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

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

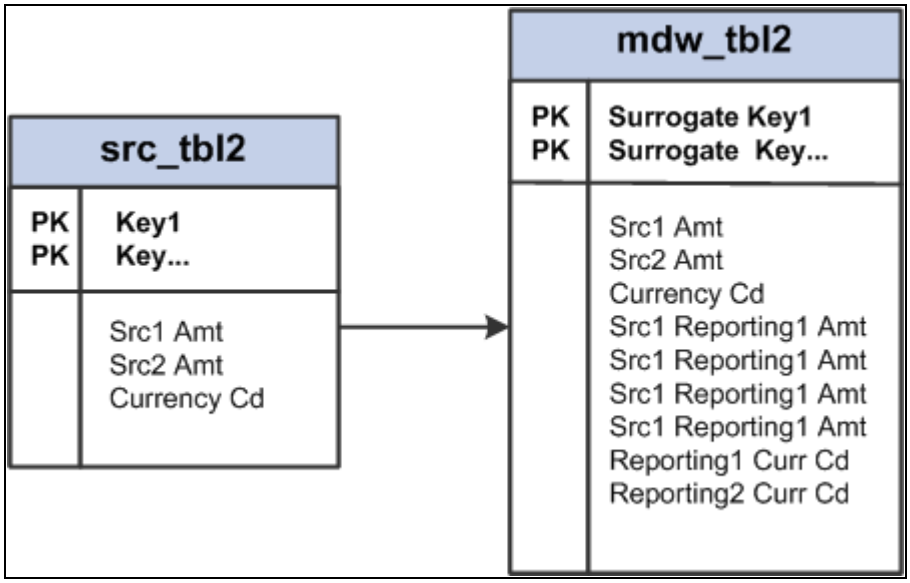
  That is, base amount and base currency code fields are source database fields. If multiple base amounts from the same source tables are stored in the MDW tables, they may share the same base currency code field.

- Source amounts in MDW tables must have corresponding reporting1 amount and reporting2 amount fields, if that amount requires currency conversion.

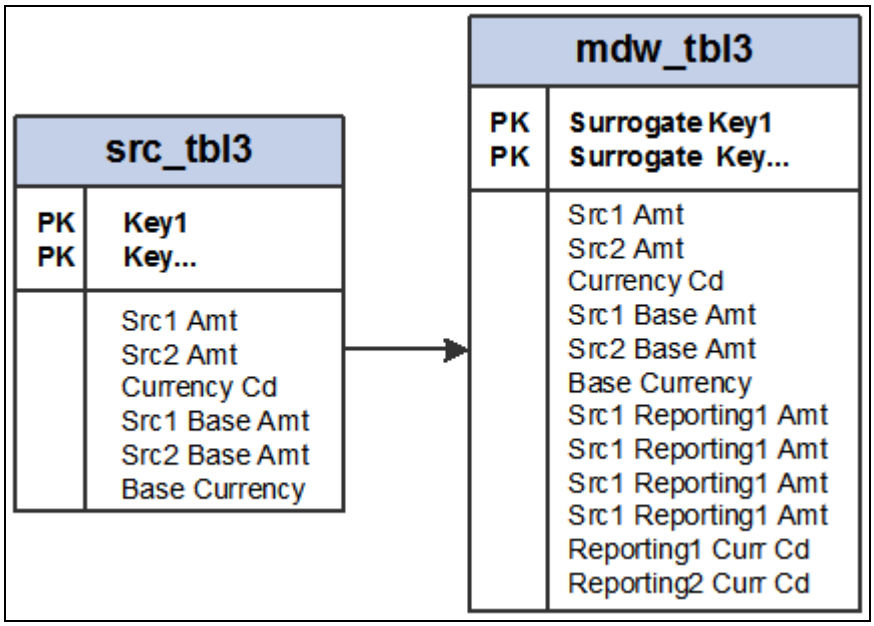
- MDW tables must have only one reporting1 currency code and reporting2 currency code fields that serve as the currency codes for all reporting1 amounts and reporting2 amounts in that MDW table.

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

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



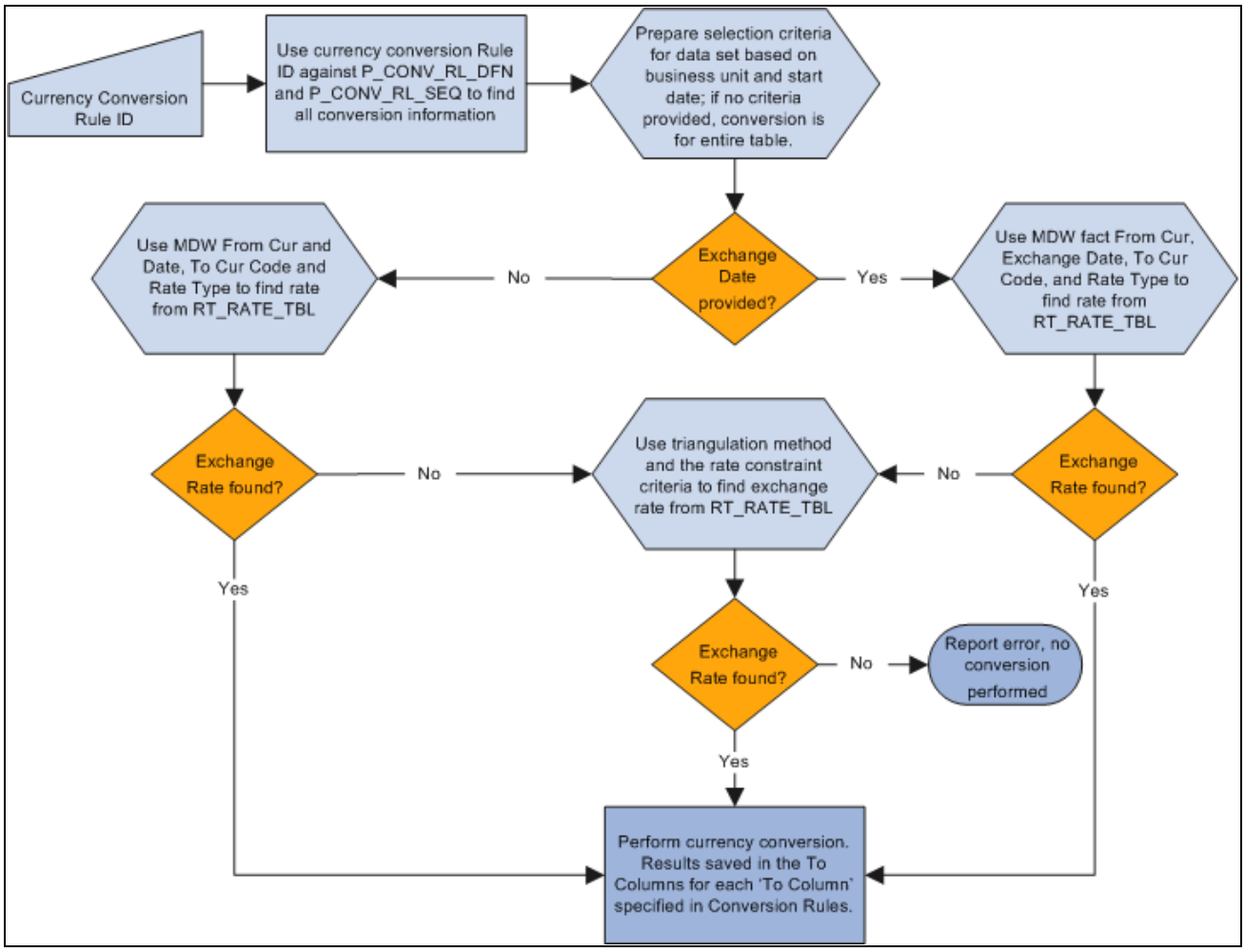
Carrying multiple source amounts into the MDW fact table



Carrying multiple source and base amounts into the MDW fact table

Understanding Currency Conversion Methodology

The following diagram represents the currency conversion process:



Currency conversion methodology flow

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

Identifying the Data Set from a Conversion Rule Business Unit

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

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

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

Identifying the Exchange Date from Date Columns

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

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

- D_DAY

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

- D_MONTH

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

- D_WEEK

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

- D_QUARTER

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

- D_YEAR

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

- D_PATTERN_DAY

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

- D_DET_PERIOD

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

- D_SUM_PERIOD

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

- D_DET_BUDGET

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

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

- D_PATTERN_YEAR

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

- D_PATTERN_WEEK

  SELECT MAX(DAY_DT) FROM PS_D_PATTERN_DAY WHERE 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.

The RT_RATE_TBL has the following structure:

RT_RATE_TBL (Record)

Record Fields | Record Type

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

RT_RATE_TBL structure

The CURR_QUOTE_TBL has the following structure:

CURR_QUOTE_TBL (Record)

Record Fields | Record Type

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

CURR_QUOTE_TBL structure

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

Condition	***Result***
If exchange rate is found in RT_RATE_TBL	then Converted Amount = Source Amount * (RATE_MULT/RATE_DIV)
If exchange rate is not found in RT_RATE_TBL	then verify using the From and To Currencies and Exchange Date when the currency quotation method is defined in CURR_QUOTE_TBL
If Currency Quotation method is defined and RATE_TRIANGULATE = 'Y'	then get REF_CUR field from CURR_QUOTE_TBL Leg 1: Find in the RT_RATE_TBL the exchange rates (RATE_MULT and RATE_DIV) between the From Currency and REF_CUR for the given Rate Type and Date. Leg 2: Find in the RT_RATE_TBL the exchange rates (RATE_MULT and RATE_DIV) between REF_CUR and the To Currency for the given Rate Type and Date. Converted Amount = Source Amount * (RATE_MULT of leg 1 * RATE_MULT of leg 2) / (RATE_DIV of leg1 * RATE_DIV of leg 2).
If Currency Quotation method is not defined, or if RATE_TRIANGULATE = 'N'	then = error (because no conversion rate is found)

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

Error Logging

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

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

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

The following information is made available in this table:

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

Understanding Currency Conversion Rules

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

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

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

Schema Rules

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

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

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

Schema rules consist of:

- Table name.
- Business unit column.

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

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

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

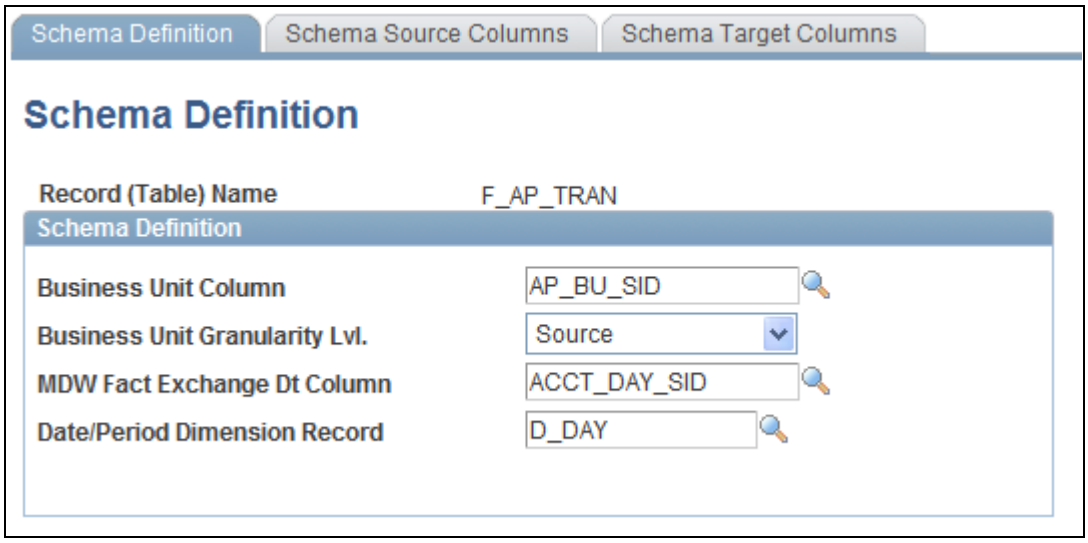
  This is required only if the exchange date/period column is specified.
- *From* amount column name.
- *To* amount column name.
- *From* currency code column name.

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

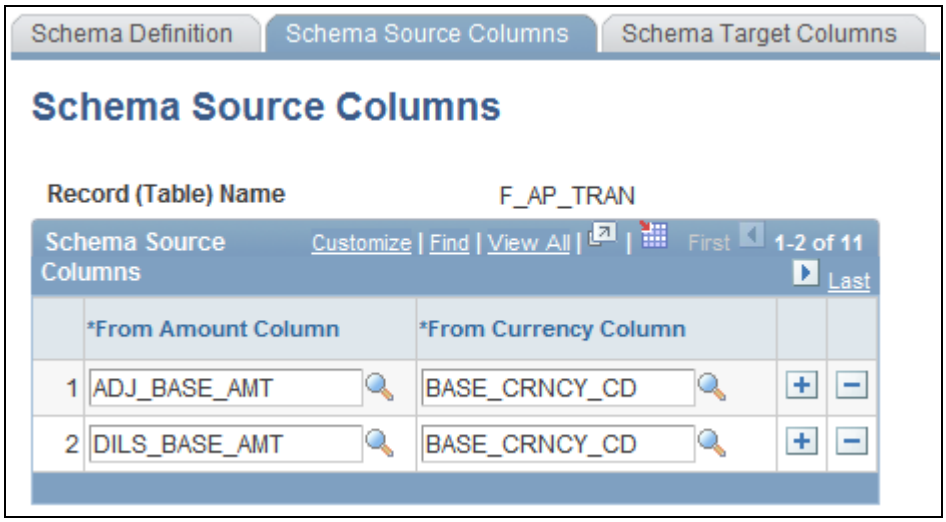
- *To* currency code column name.

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

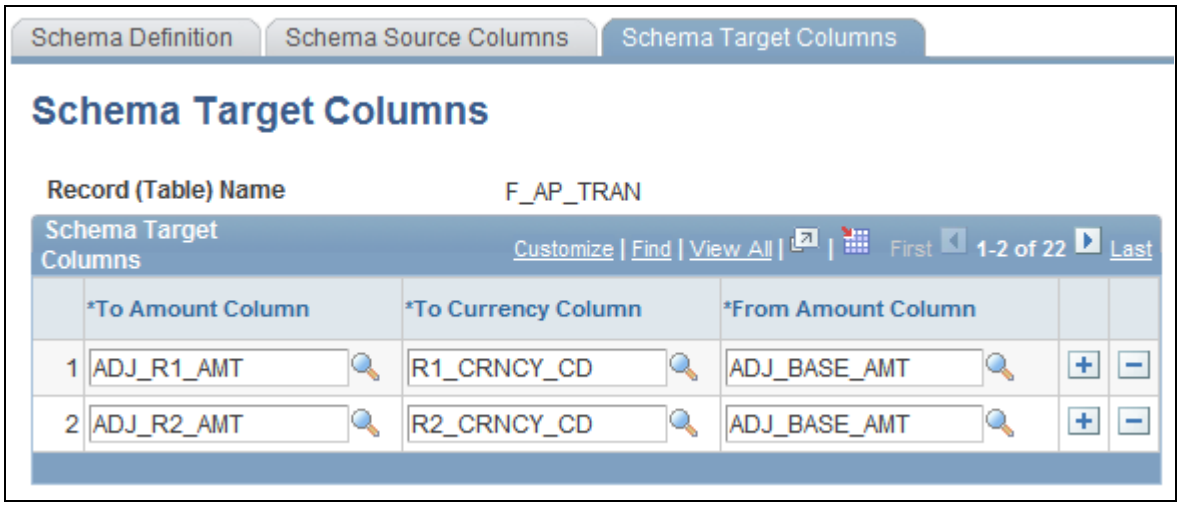
The following example shows the schema definitions for the F_AP_TRAN fact table:



Schema Definition page for F_AP_TRAN fact table



Schema Source Columns page for F_AP_TRAN fact table



Schema Target Columns page for F_AP_TRAN fact table

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

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

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

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

See Chapter 8, "Processing Trees and Recursive Hierarchies," Setting Up Parameters for Tree and Recursive Hierarchy Processing, page 133.

Conversion Rules

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

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

Exchange rate rules consist of:

- Rate Type.

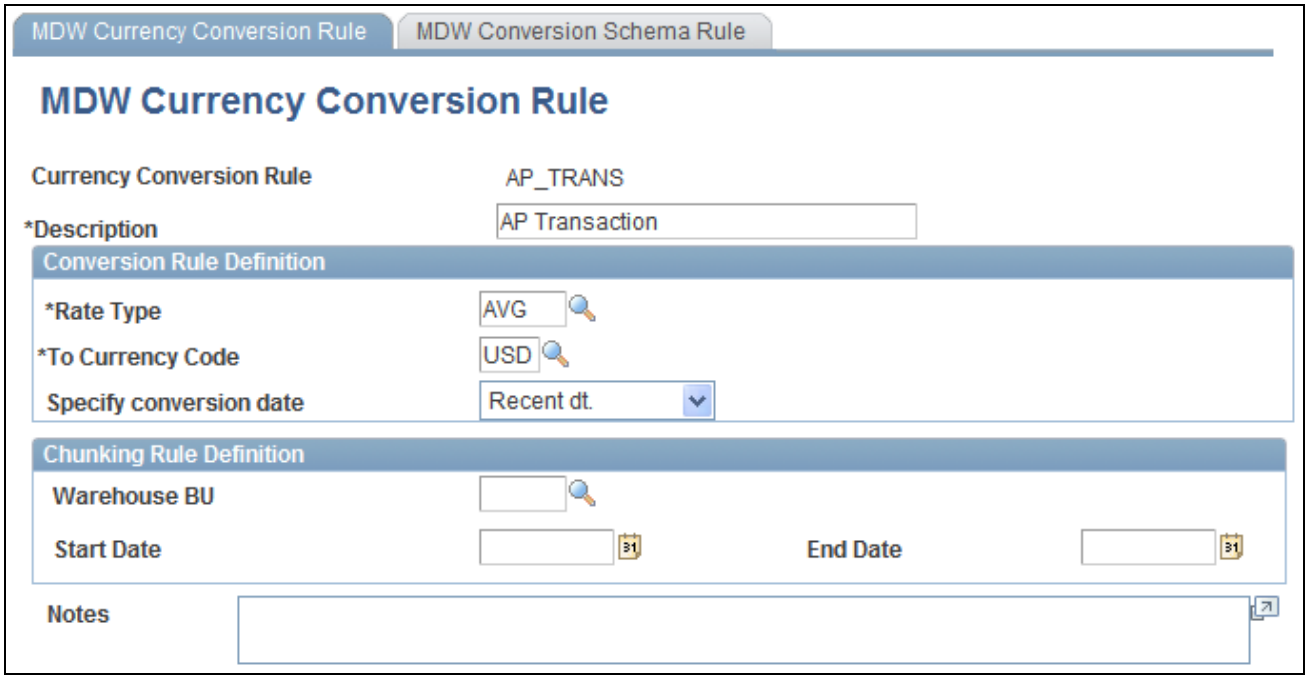
- *To* currency code.
- Conversion date.

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

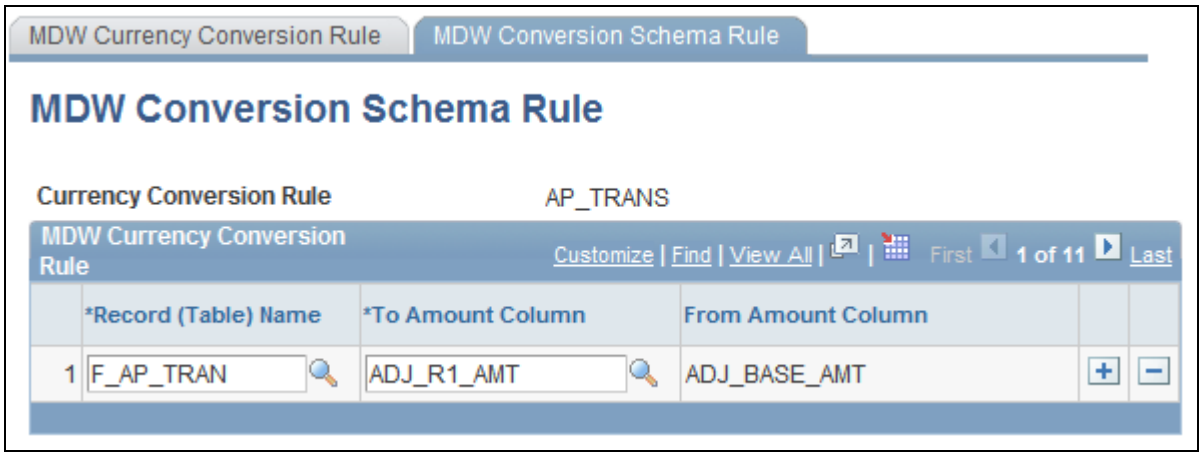
  You must choose either:

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

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



MDW Currency Conversion Rule page for AP_TRANS fact table



MDW Conversion Schema Rule page for AP_TRANS fact table

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

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

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

Chunking Rules

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

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

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

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

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

Setting Up Currency Conversion

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

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

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

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

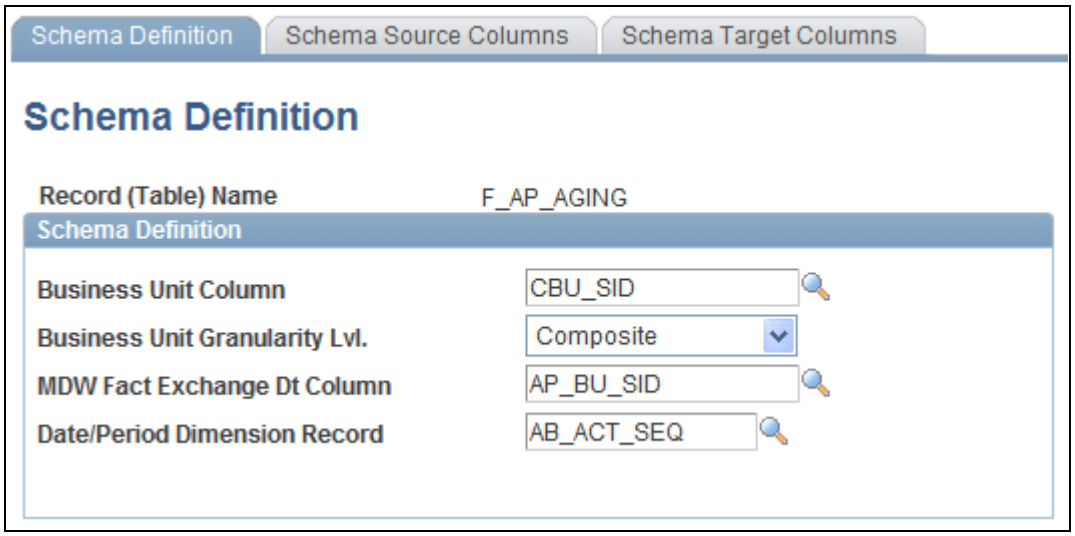
Pages Used to Set Up the Schema Definition and Currency Conversion Rules

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

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

Setting Up the Schema Definition

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



Schema Definition page

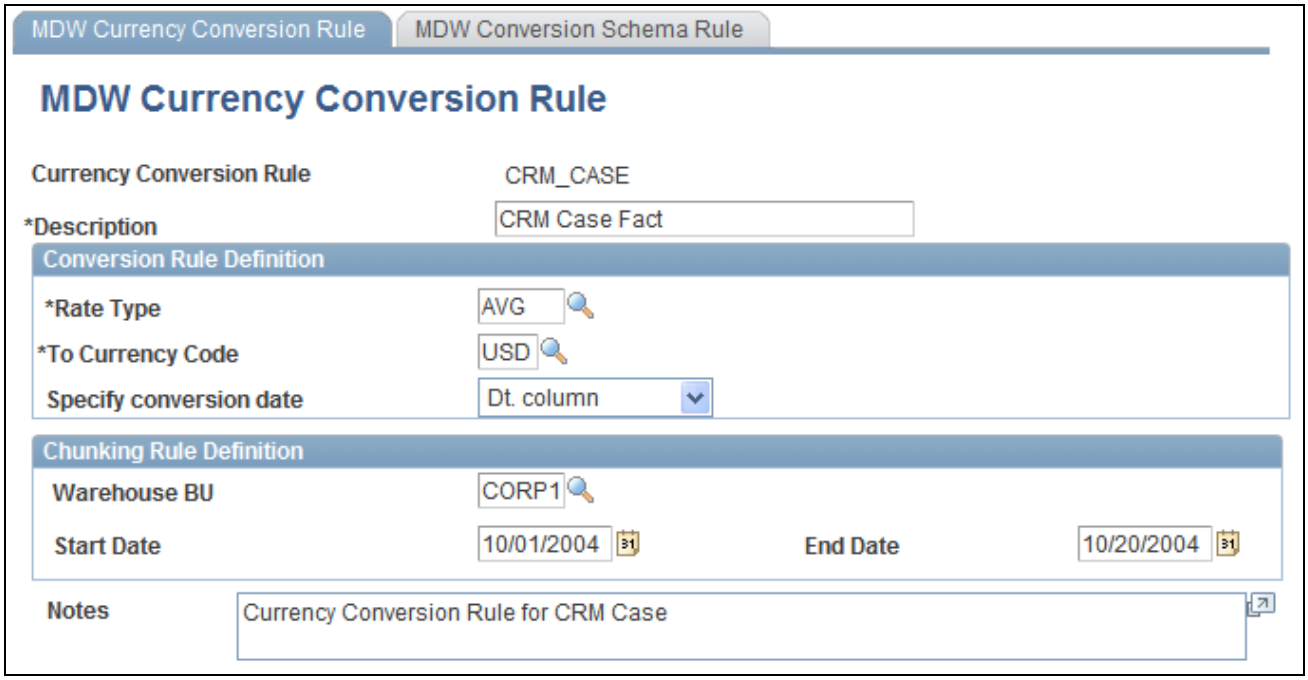
Business Unit Column Enter the name of the PF (performance) business unit column.

MDW Fact Exchange Dt Column (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.

Setting Up Currency Conversion Rules

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



MDW Currency Conversion Rule page

Currency Conversion Rule The name of the currency conversion rule.

Conversion Rule Definition

Rate Type Select the rate type for this rule.

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

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

Dt. column (date column): From the transaction record.

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.

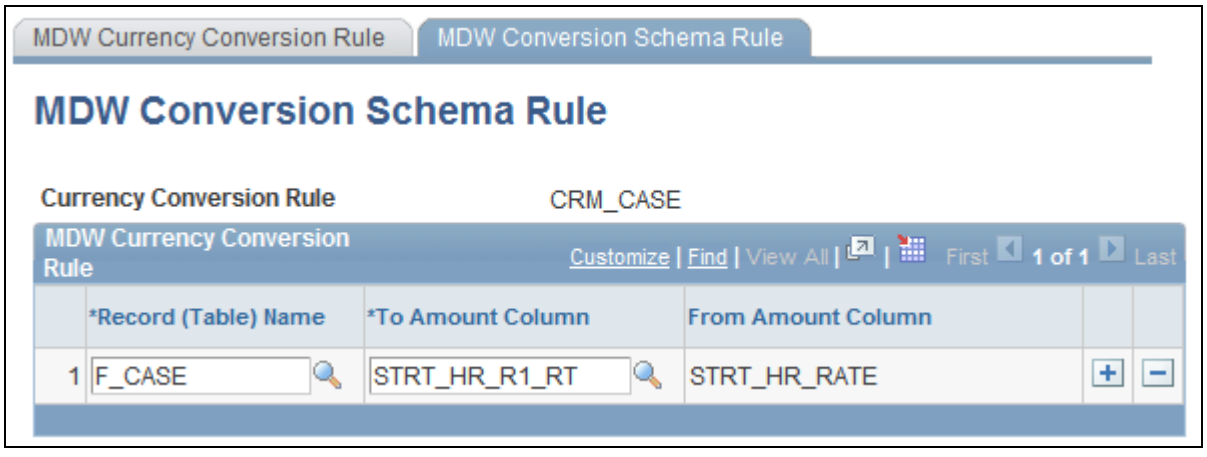
Start Date (Optional) Enter the date for which the currency conversion should begin. If you supply a start date parameter, any date greater than or equal to this parameter value will experience currency conversion. If you do not supply a start date parameter, then the currency conversion process uses *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.

Setting up the Conversion Schema Rule

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



MDW Conversion Schema Rule page

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

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

Running the ETL Currency Conversion Process

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

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

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

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

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

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

Steps Required to Run the MDW ETL Currency Conversion Utility

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

1. In IBM WebSphere DataStage Director, navigate to the MDW currency conversion job by expanding the nodes in the left navigation panel using the following path: *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.

See Also

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

Chapter 7

Setting Up Multilanguage Processing and Running the Language Swap Utility

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

Understanding Multilanguage Processing

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

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

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

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

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

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

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

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

The Language Swap Utility and multilanguage processing enables you to:

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

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

Understanding Multilanguage Setup

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

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

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

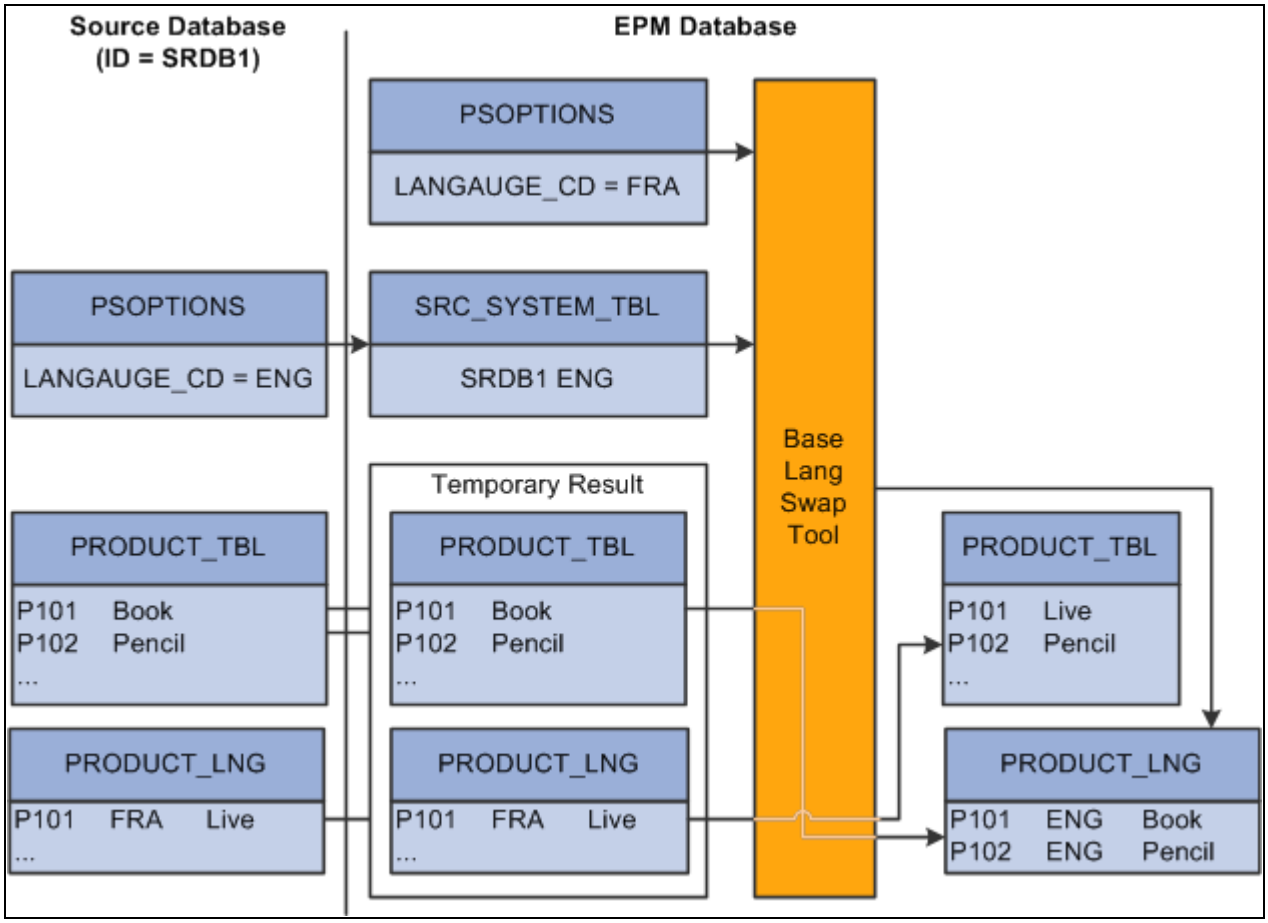
- Run the language swap utility.

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

Understanding the Language Swap Utility

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

The following graphic depicts the language swap process.



Language swap process flow

The swapping process works as follows:

1. Check if the base language of the source database is the same as the base language of the EPM database.

   If the languages are the same, then continue to the last step in this process. Otherwise, proceed to the next step.

2. Check if the base language of the source database is a supported foreign language in the EPM database.

   If the language is the same, then create a corresponding entry in the related-language table for every new (not yet swapped) row in the base table. The LANGUAGE_CD used for tagging the new entry in the related-language table is the LANGUAGE_CD for the source database as it is found in the SRC_SYSTEM_TBL table.

3. Locate the description in the related language table where LANGUAGE_CD for the row is the same as the base language code of the EPM database. Once identified, perform the swap between the description in the base table and the related-language table.

4. Delete from the related-language table any rows where the LANGUAGE_CD is the base language code of the EPM database or is not any of the supported foreign languages in the EPM

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

Outrigger Tables

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

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

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

Sample Outcome of the Language Swap Process

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

Here is the Product table from the source database:

ProductID	Description
P101	Libro
P102	Lápiz
P103	Pluma

Here is the Product Language table from the source database:

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

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

Here is the OWS Product table from the EPM database:

ProductID	Description
P101	Book
P102	Pencil
P103	Pluma

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

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

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

This table shows an example of a Sales fact table:

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

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

Product Key	*SKU*	*Description*
1	A123	*Bread*
2	B234	*Marmalade*
3	C345	*Milk*

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

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

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

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

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

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

Running the Language Swap Jobs

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

Perform the following steps to run the language swap jobs:

1. In IBM WebSphere DataStage Director navigate to the language swap jobs by expanding the nodes in the left navigation panel using the following path: *Jobs, EPM_Utilities, 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.

Chapter 8

Processing Trees and Recursive Hierarchies

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

Understanding Tree and Recursive Hierarchy Processing

This section discusses:

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

Trees and Recursive Hierarchies

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

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

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

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

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

See Also

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.

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

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

PeopleSoft trees and recursive hierarchies relate each node in a hierarchy only to its direct parent or child. Data stored in this way makes it difficult to access non-subsequent child notes (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 descendents, direct or indirect.

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

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

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

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

Hierarchies Supported by the Tree and Recursive Hierarchy Process

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

Source Database Tree

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

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

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

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

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

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

Source Database Recursive Hierarchy

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

EPM Tree

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

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

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

EPM Recursive Hierarchy

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

Denormalized Tree Result Balancing

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

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

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

As a result of balancing an unbalanced tree, the description field for the newly created nodes contains a specific notation. This notation is <*dd*>~, where <*dd*> is the two-digit level number, for example *03* for level three, and ~, 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 ~, then these are the balancing results.

Balancing up:

E_ID	*E_Desc*	*L31_ID*	*L31_Desc*	*...*	*L2_ID*	*L2_Desc*	*L1_ID*	*L1_Desc*
C	Car	C	31~Car	-	C	02~Car	A	Auto

Balancing down:

E_ID	*E_Desc*	*L31_ID*	*L31_Desc*	*...*	*L2_ID*	*L2_Desc*	*L1_ID*	*L1_Desc*
C	Car	A	31~Auto	-	A	02~Auto	A	Auto

No balancing:

E_ID	E_Desc	L31_ID	L31_Desc	...	L2_ID	L2_Desc	L1_ID	L1_Desc
C	Car	-	-	-	-	-	A	Auto

Skip Levels

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

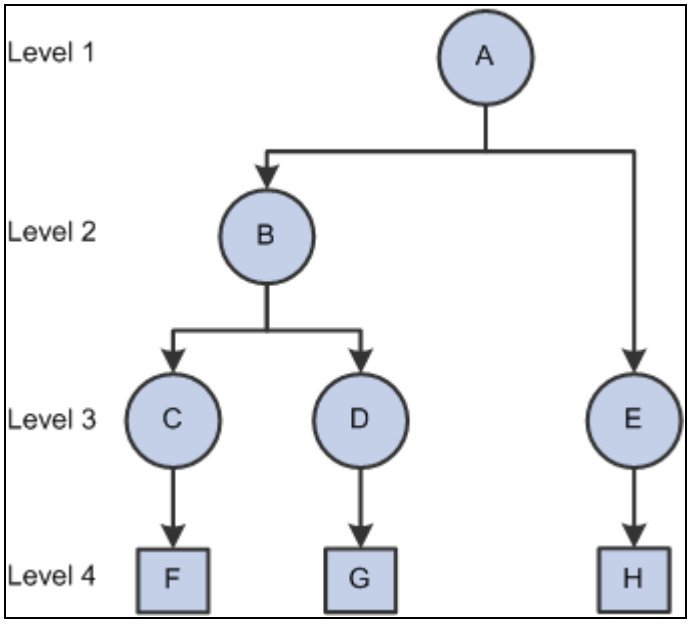
Similar to tree balancing, skip level handling produces synthetic, or artificial, node entries to fill the gap in a denormalized table. For some business intelligence tools, especially ROLAP tools, you must close the gap produced by a skipped level to use the denormalized table effectively.

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

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

Skipping Levels Example

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



Skip level summer tree

The following table represents the tree flattener result:

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

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

-	*Row 1*	*Row 2*	*Row 3*
Ent_ID	F	G	H
Ent_Dec	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:

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

Multilanguage Support for Relationship and Hierarchy Tables

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

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

See Chapter 7, "Setting Up Multilanguage Processing and Running the Language Swap Utility," Understanding Multilanguage Processing, page 101.

Understanding Tree and Recursive Hierarchy Process Results

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

Tree Flattener and Tree Denormalizer Output Tables

Running the Tree and Recursive Hierarchy process creates:

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

  In the hierarchy table:

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

Output Relationship Tables

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

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

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

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

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

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

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

Field Name	*Type*	*Length*	*Key*	*Required*	*Edit*	*Prompt*	*Default*
SETID	Char	5	K	Y	N	N	None
SRC_SETID	Char	5	K	N	N	N	None
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

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
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

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

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
BUSINESS_UNIT	Char	5	K	Y	N	N	None
SRC_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

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
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 *user-defined* (no key) trees:

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
SETCNTRLVALUE	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

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

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
SETID	Char	5	K	Y	N	N	None
SRC_SETID	Char	5	K	N	N	N	None
RECORD	Char	30	K	Y	N	N	None
SRC_SYS_ID	Char	5	K	Y	N	N	None
EFFDT	Date	10	K	Y	N	N	None
NODE	Char	30	K	Y	N	N	None
CHILD_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

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

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

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
EFFDT	Date	10	K	Y	N	N	None
NODE	Char	30	K	Y	N	N	None
CHILD_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

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

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
RECORD	Char	30	K	Y	N	N	None
SRC_SYS_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

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
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:

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
SETID	Char	5	K	Y	N	N	None
SRC_SETID	Char	5	K	N	N	N	None
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

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

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

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

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

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
SETCNTRLVALUE	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 *setID* based *recursive hierarchy* trees:

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

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
…	-	-	-	-	-	-	-
DATA_SRC_SYS_ID	Char	5	-	Y	N	N	None

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

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
BUSINESS_UNIT	Char	5	K	Y	N	N	None
SRC_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
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:

Field Name	Type	Length	Key	Required	Edit	Prompt	Default
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

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

Related Language Tables

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

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

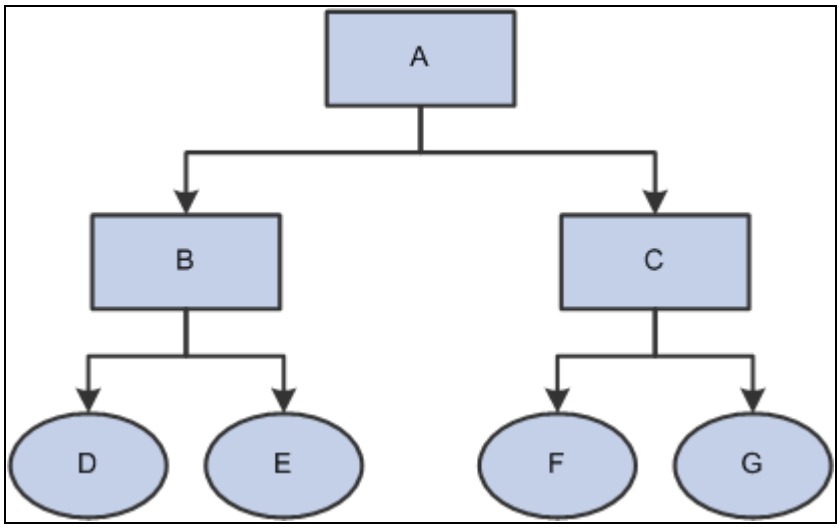
See Chapter 7, "Setting Up Multilanguage Processing and Running the Language Swap Utility," Understanding Multilanguage Processing, page 101.

Tree Flattener and Denormalizer Results

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

Summer Tree

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



Example of a summer tree before processing

Processing of this tree results in the following relationship table:

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

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

The following table shows the hierarchy table, without balancing:

Entity	*L31*	*L30*	*...*	*L2*	*L1*
D	-	-	-	B	A
E	-	-	-	B	A
F	-	-	-	C	A

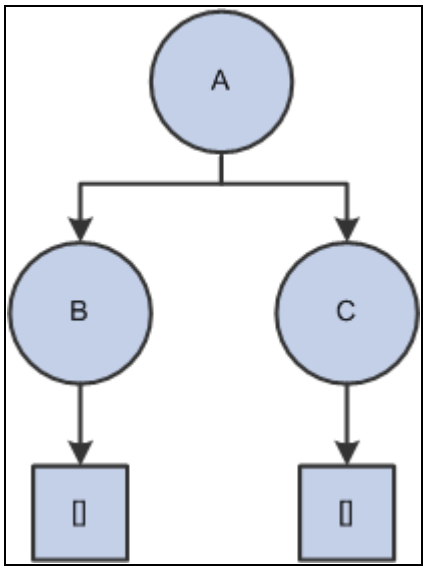
Entity	*L31*	*L30*	*...*	*L2*	*L1*
G	-	-	-	C	A

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

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

Dynamic Summer Tree

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



Dynamic summer tree

The following table represents the Department database table:

Department ID	*Department Name*
A	HR
B	Benefit
C	Payroll

The following table represents the Employee table:

Employee ID	*Department ID*	*Employee Name*
D	B	Jane Doe

Employee ID	*Department ID*	*Employee Name*
E	B	Joe Bloe
F	C	John Who

Processing of this tree results in the following relationship table:

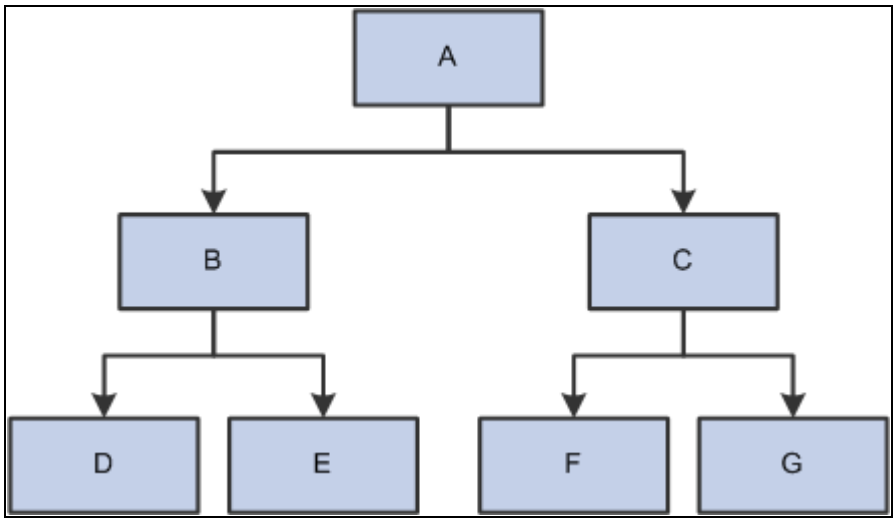
Tree Node	*Entity ID*	*Parent Level Number*	*Child Level Number*
A	D	1	0
A	E	1	0
A	F	1	0
B	D	2	0
B	E	2	0
C	F	2	0

The following table shows the hierarchy table, without balancing:

Entity	*L31*	*L30*	*...*	*L2*	*L1*
D	-	-	-	B	A
E	-	-	-	B	A
F	-	-	-	C	A

Winter Tree

This graphic shows an example of a winter tree before processing:



Example of a winter tree before processing

This table shows the winter tree relationship table after tree flattening:

Tree Node	*Entity ID*	*Parent Level Number*	*Child Level Number*
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

Tree Node	Entity ID	Parent Level Number	Child Level Number
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:

Entity	L31	L30	...	L3	L2	L1
A						A
B					B	A
C					C	A
D				D	B	A
E				E	B	A
F				F	C	A
G				G	C	A

Recursive Hierarchy

The following table provides an example of a recursive hierarchy table:

Entity ID	Entity Parent ID
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.

Parent Node	*Entity*	*Parent Level Number*	*Child Level Number*
1	1	1	1
1	2	1	2
1	3	1	2
1	4	1	3
1	5	1	3
2	2	2	2
2	4	2	3
2	5	2	3
3	3	2	2
4	4	3	3
5	5	3	3

The following table shows the hierarchy table without balancing:

Entity	*L31*	*L30*	*...*	*L3*	*L2*	*L1*
1	-	-	-	-	-	1
2	-	-	-	-	2	1
3	-	-	-	-	3	1
4	-	-	-	4	2	1
5	-	-	-	5	3	1

Setting Up Parameters for Tree and Recursive Hierarchy Processing

Before you can run the actual Tree and Recursive Hierarchy ETL process, you must first define the parameters for process.

This section provides an overview of parameters for the Tree and Recursive Hierarchy process and discusses how to:

- Define the target and language tables for tree flattening.
- Define the target and language tables for tree denormalizing.

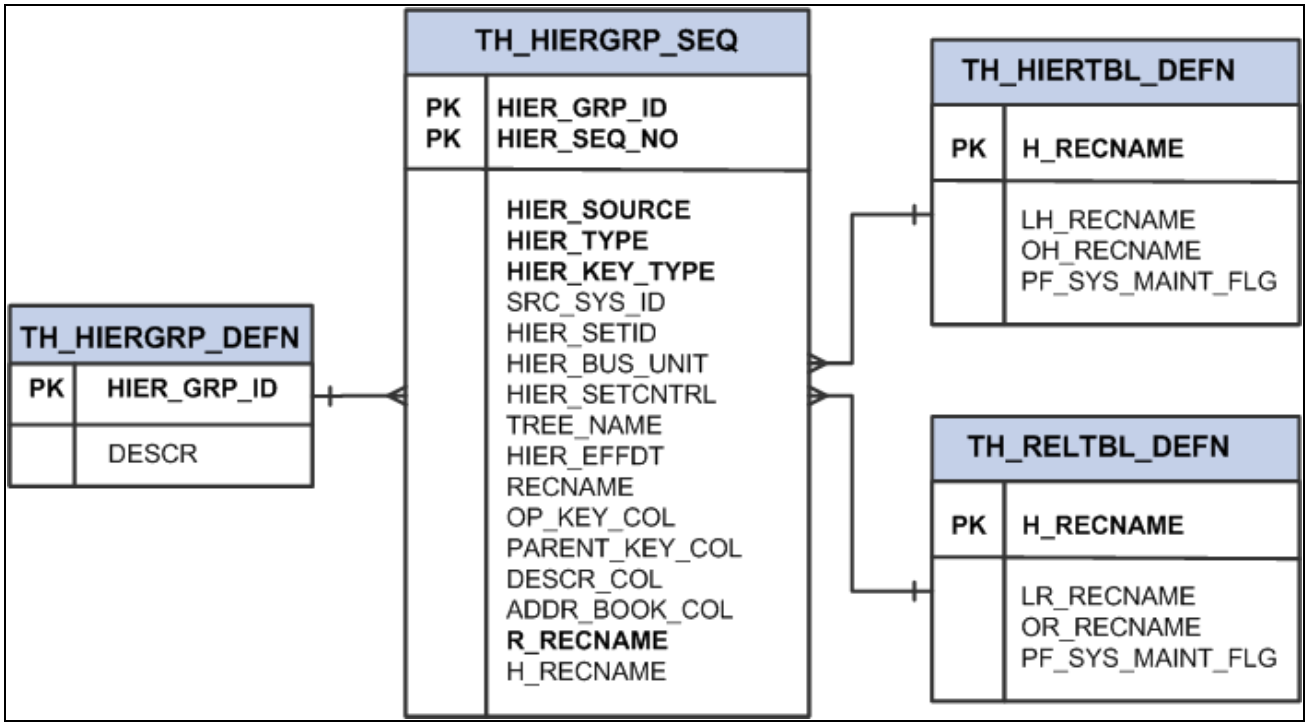
- Create the hierarchy group definition.

Defining Parameters for the Tree and Recursive Hierarchy Process

To run the Tree and Recursive Hierarchy process, use the Tree Hierarchy-Relational Table (TH_RELTBL_DEFN) component, Tree Hierarchy-Hierarchy Table (TH_HIERTBL_DEFN) component, and Tree Hierarchy-Hierarchy Group Definition (TH_HIERGRP_DEFN) component.

Because you must first flatten all hierarchies that are processed, you must define the *relationship table* that is the target for the flattening process. Because the denormalization process is optional, you must define the *hierarchy table* only if you intend to denormalize the flattened table.

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



Tree and recursive hierarchy processing setup pages

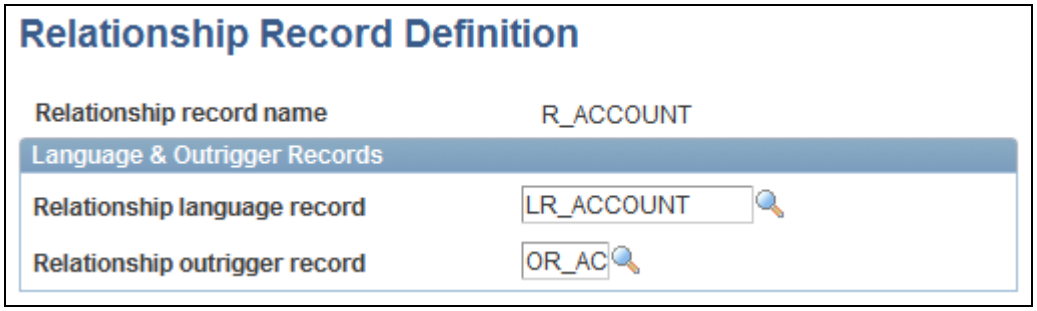
Pages Used to Run the Tree and Recursive Hierarchy Process

Page Name	*Definition Name*	*Navigation*	*Usage*
Relationship Record Definition	TH_RELTBL_DEFN	EPM Foundation, EPM Setup, Common Definitions, Hierarchy Group Definition, Relationship Table Definition	Define the target and language tables for tree flattening.

Page Name	Definition Name	Navigation	Usage
Hierarchy Record Definition	TH_HIERTBL_DEFN	EPM Foundation, EPM Setup, Common Definitions, Hierarchy Group Definition, Hierarchy Table Definition	Define the target and language tables for tree and hierarchy denormalizing.
Hierarchy Group Definition	TH_HIERGRP_DEFN	EPM Foundation, EPM Setup, Common Definitions, Hierarchy Group Definition, Hierarchy Group Definition	Enter parameters for the Tree and Recursive Hierarchy process.

Defining the Target and Language Tables for Tree Flattening

Access the Relationship Record Definition page (EPM Foundation, EPM Setup, Common Definitions, Relationship Group Definition, Relationship Record Definition).



Relationship Record Definition page

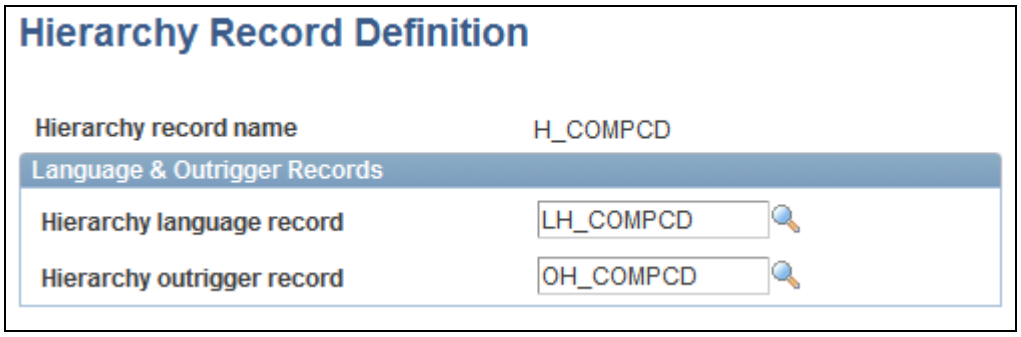
Relationship record name Displays the target record for the flattening portion of the Tree and Recursive Hierarchy process.

Relationship language record Enter the language record for this relationship table. This value is required only if this table is used for multilanguage processing.

Relationship outrigger record Enter the outrigger record for this relationship table. This value is required only if this table is used for multilanguage processing.

Defining the Target and Language Tables for Tree Denormalizing

Access the Hierarchy Record Definition page (EPM Foundation, EPM Setup, Common Definitions, Hierarchy Group Definition, Hierarchy Table Definition).



Hierarchy Record Definition page

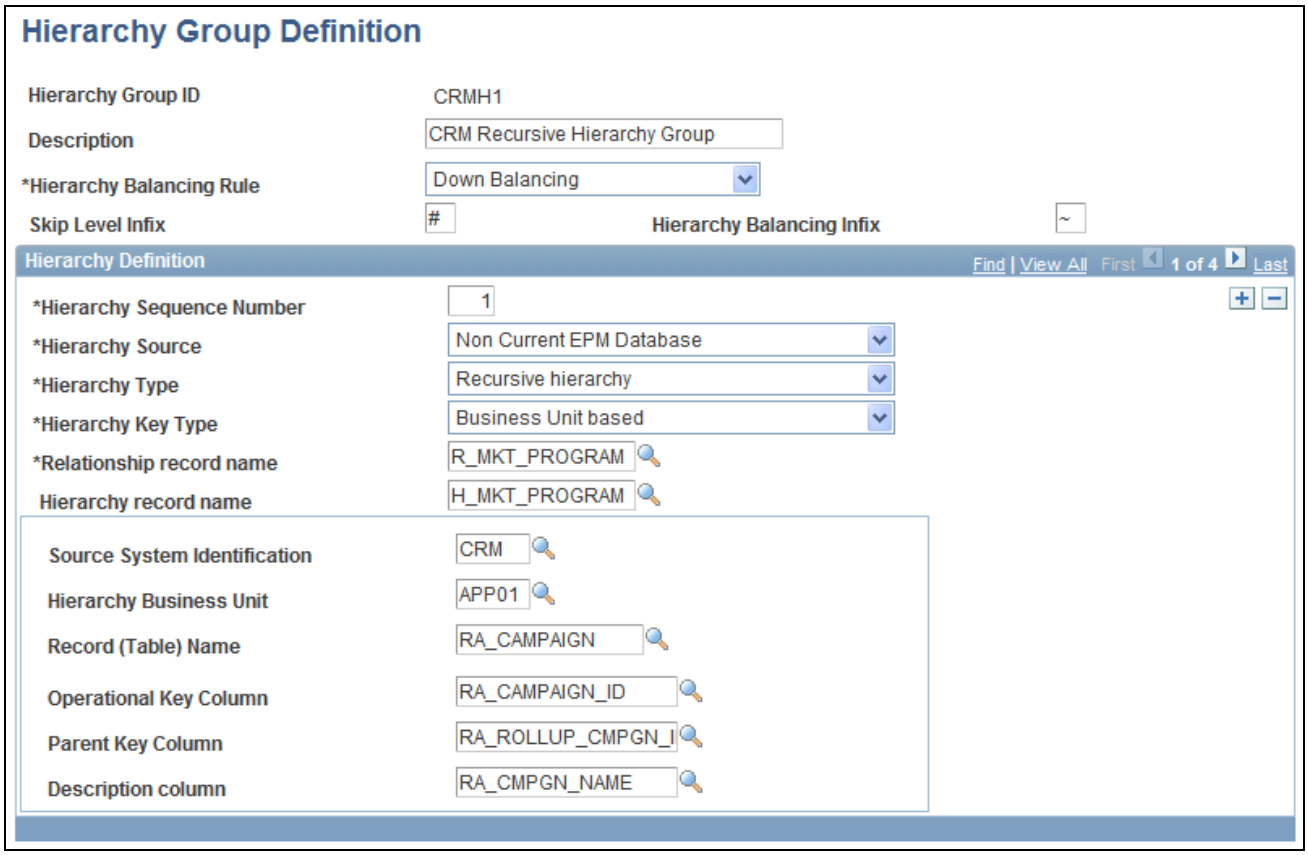
Hierarchy record name Displays the target record for the denormalizing portion of the Tree and Recursive Hierarchy processing.

Hierarchy language record Enter the language record for this hierarchy table. You must enter this value only if this table is used for multilanguage processing.

Hierarchy outrigger record Enter the outrigger record for this hierarchy table. You must enter this value only if this table is used for multilanguage processing.

Creating the Hierarchy Group Definition

Access the Hierarchy Group Definition page (EPM Foundation, EPM Setup, Common Definitions, Hierarchy Group Definition, Hierarchy Group Definition).



Hierarchy Group Definition page

The Hierarchy Group Definition page contains a list of trees, recursive hierarchies, or both, that are related to a particular business process. For example, when you perform a workforce composition analysis, you must analyze data along organization, jobcode, and compensation code hierarchies. In this case, you can define the organization tree, jobcode tree, and compensation tree in one hierarchy group on the Hierarchy Group Definition page. Then, to perform workforce composition analysis, you need only to run tree processing using that hierarchy group ID as the parameter. When you run the Tree and Recursive Hierarchy process for that hierarchy group ID, the trees and recursive hierarchies that are associated with that ID are processed into either relational or hierarchical tables.

Note. The Hierarchy Group Definition page shown above is an example of this page using certain field values. If your field values differ, the fields that are available may be different. The following table of terms includes a list of all possible fields and the situations under which they display on this page.

Hierarchy Group ID Displays the identifier for a group of trees, recursive hierarchies, or both, that relate to a specific business process that you intend to process into relationship or hierarchical tables. You can add a new hierarchy group or modify an existing hierarchy froup 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.*

Hierarchy Business Unit	Enter the business unit for this hierarchy. This field is available only if the value in the Hierarchy Key Type field is *Business Unit based.*
User Defined Value	Enter the user-defined key value for this hierarchy. This field is available only if the value in the Hierarchy Key Type field is *User defined.*
Record (Table) Name	Enter the recursive hierarchy source table name for this hierarchy. This field is available only if the value in the Hierarchy Type field is *Recursive hierarchy.*
Tree Name	Enter the name of the tree for this process. This field is available only if the value in the Hierarchy Type field is *Tree hierarchy.*
Hierarchy effective date	Enter the effective date for the tree in Tree Manager.
Operational Key Column	Enter the column for the operational key for this hierarchy. This field is available only if the value in the Hierarchy Type field is *Recursive Hierarchy.*
Parent Key Column	Enter the column for the parent key for this hierarchy. This field is available only if the value in the Hierarchy Type field is *Recursive Hierarchy.*
Description Column	Enter the column for the description for this hierarchy. This field is available only if the value in the Hierarchy Type field is *Recursive Hierarchy.*
Source System Identification	Enter the name of the source for this hierarchy. This field is available only if the value in the Hierarchy Source field is *Non Concurrent EPM Database.*

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, SCM, 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.*

Chapter 9

Extending the Multidimensional Warehouse Data Model

This chapter provides considerations for modifying an EPM warehouse and discusses:

- Add a fact or dimension table to the MDW data model.
- Extend a fact table in the MDW data model.
- Extend a dimension table in the MDW data model.

Note. The procedures discussed in this chapter are not supported by PeopleSoft, they are merely a guide for extending the MDW data model.

Considerations for Modifying an EPM Warehouse

The prepackaged MDW data model is designed to facilitate the introduction of a new dimension, a new attribute on an existing dimension table, a new dimension and/or a new fact based on an existing fact table. Any modifications that you make to the overall solution, including data mart content, will affect the reporting results. Consider the following questions when modifying the data model and developing reports.

Before modifying the data model, determine:

- What business intelligence reporting tool you will use— relational online analytic processing (ROLAP) or multidimensional online analytic processing (MOLAP).
- Whether you must build aggregate fact tables (if using ROLAP).

  If so, determine what aggregate levels of each dimension to use for this fact table.
- The typical profile of your end-users.

Evaluate Dimension Requirements

To ensure that the dimensions meet the needs of your business, determine:

- The dimensions in the delivered data mart that you want to keep and the ones that you can eliminate.
- The dimensions that you must create that are not part of the delivered data mart.
- The changes that you must make to delivered dimensions.

If you alter an existing dimension or add a new dimension, determine:

- Whether this is a dimension or merely an attribute of an existing dimension.
- The hierarchies in this dimension.
- The hierarchy levels in each hierarchy of this dimension.
- The attributes of this dimension.
- Whether you are altering the lowest level of an existing dimension.
- Whether you have facts at this level.
- Whether your multidimensional analysis will still work.

Evaluate Measure Requirements

To ensure that the measures you use meet the needs of your business, determine:

- The measures in the delivered data mart to keep and those to eliminate.
- The measures that you must create that are not part of the delivered data mart.
- What changes you must make to delivered measures, such as changing calculations or populating required fields.

If you alter an existing measure or add a new measure, determine:

- Whether they are base measures that you can store on a row by row basis, or are runtime calculations that you must define in the reporting tool.
- What dimensions qualify this measure.
- Whether this measure is an addition to an existing fact table, or must be part of a new fact table.
- Whether you will use this measure along with other measures on the same report.

Adding a Fact or Dimension Table to the Multidimensional Warehouse Data Model

You can extend the analytic capabilities of the prepackaged Multidimensional Warehouse (MDW) data model by adding new fact and dimension tables. New fact tables can become necessary when you introduce new sources of data to the data warehouse and/or new business processes are desired for analytic analysis.

Steps Required to Add a New Fact Table

The following steps are required to add a new fact table to the MDW data model:

1. Select the business process to be modeled.
2. Decide the grain for the new fact table.
3. Choose the dimensionality of the new fact table.

4. Identify the facts to be represented on the new fact table.
5. Identify the source of the new content and its corresponding error table.
6. Design the table structure required and apply it to the database.
7. Create new fact ETL job.
8. Declare the category or mart to which the fact applies.
9. Modify the applicable master sequence to include the new ETL fact job.

Note. This step is discussed in more detail below.

Use the following path in the left navigation panel of IBM WebSphere DataStage Director to locate the fact master sequence referenced in step nine above: *E, [business process], [data mart], Master_Sequence.*

Steps Required to Add a New Dimension Table

The following steps are required to add a new dimension table to the MDW data model:

1. Choose the new dimensionality desired and determine whether there are any corresponding related or outrigger language requirements.
2. Identify the source of the new content and its corresponding error table.
3. Design the table structure required and apply to the database.
4. Create new dimension ETL jobs (including any corresponding related or outrigger language jobs).
5. Declare the category or mart to which the dimension applies.
6. Modify the master sequence for this mart to include the new ETL dimension job as described in this section.
7. Modify the applicable master sequence to include the new ETL dimension job.

   If required, the Language dimension sequence must be modified, as well.

Note. This step is discussed in more detail below.

The are three types of dimension master sequences that must be modified, as referenced in step seven above:

- Global Dimension Master Sequence.
- Local Dimension Master Sequence
- Data Mart Business Process Dimension Master Sequence

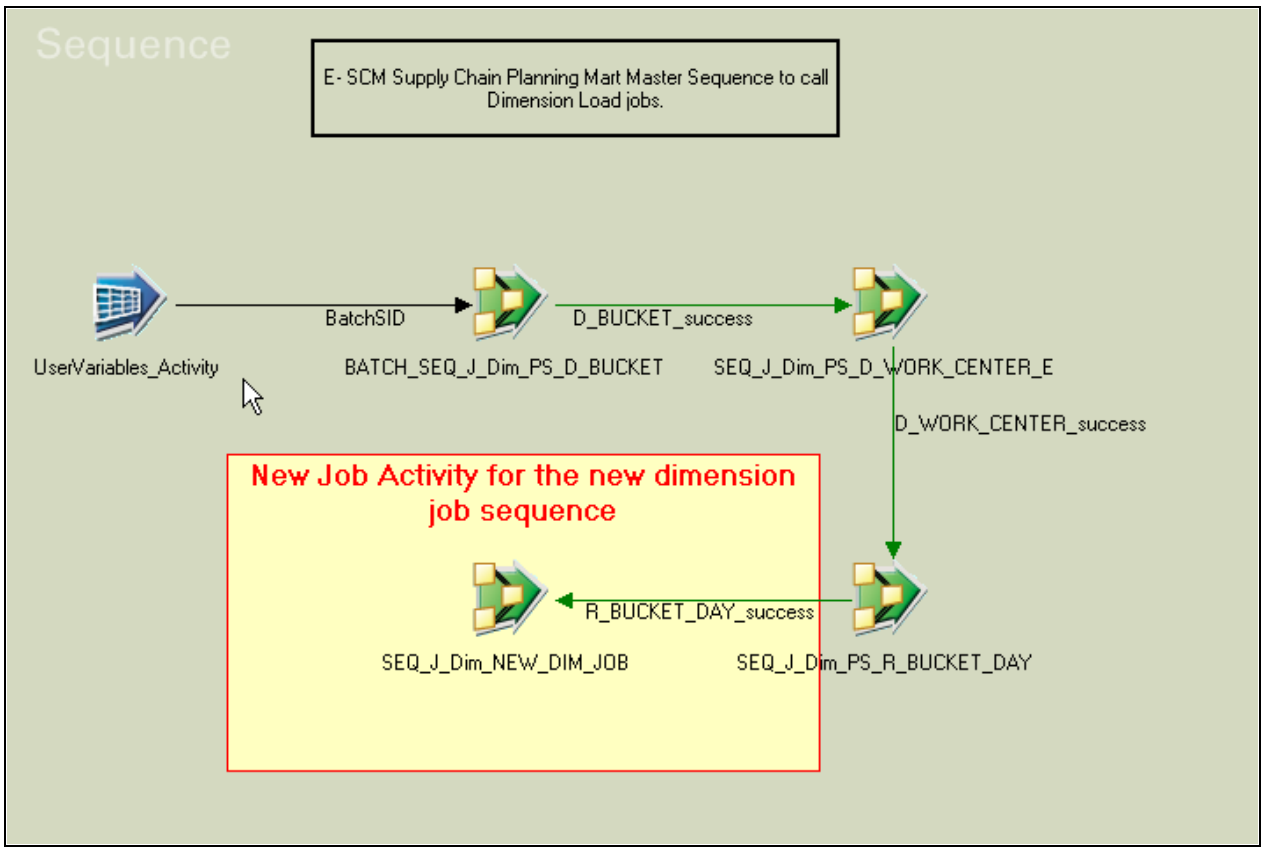
  Use the following path in the left navigation panel of IBM WebSphere DataStage Director to locate the applicable dimension master sequence: *E, [business process], [data mart], Master_Sequence.*

Adding a New Fact or Dimension Job to a Master Sequence

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

1. In IBM WebSphere DataStage Designer, locate the appropriate fact or dimension master sequence using the navigation provided in the preceding sections.
2. Open the master sequence for editing.
3. Add your new job (as a *job activity*) to the master sequence.

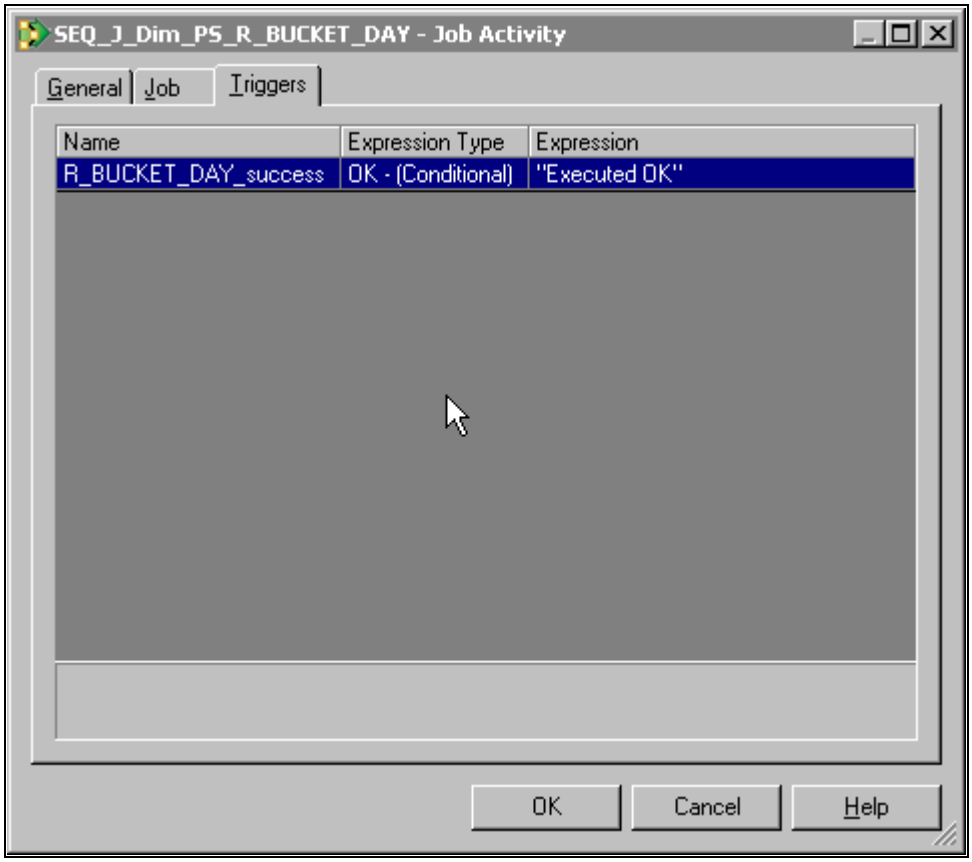
   Drag the new job from the IBM WebSphere DataStage Designer Repository window and drop it in the Diagram window. The job appears as an activity in the Diagram window.



Add new job to master sequence

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

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



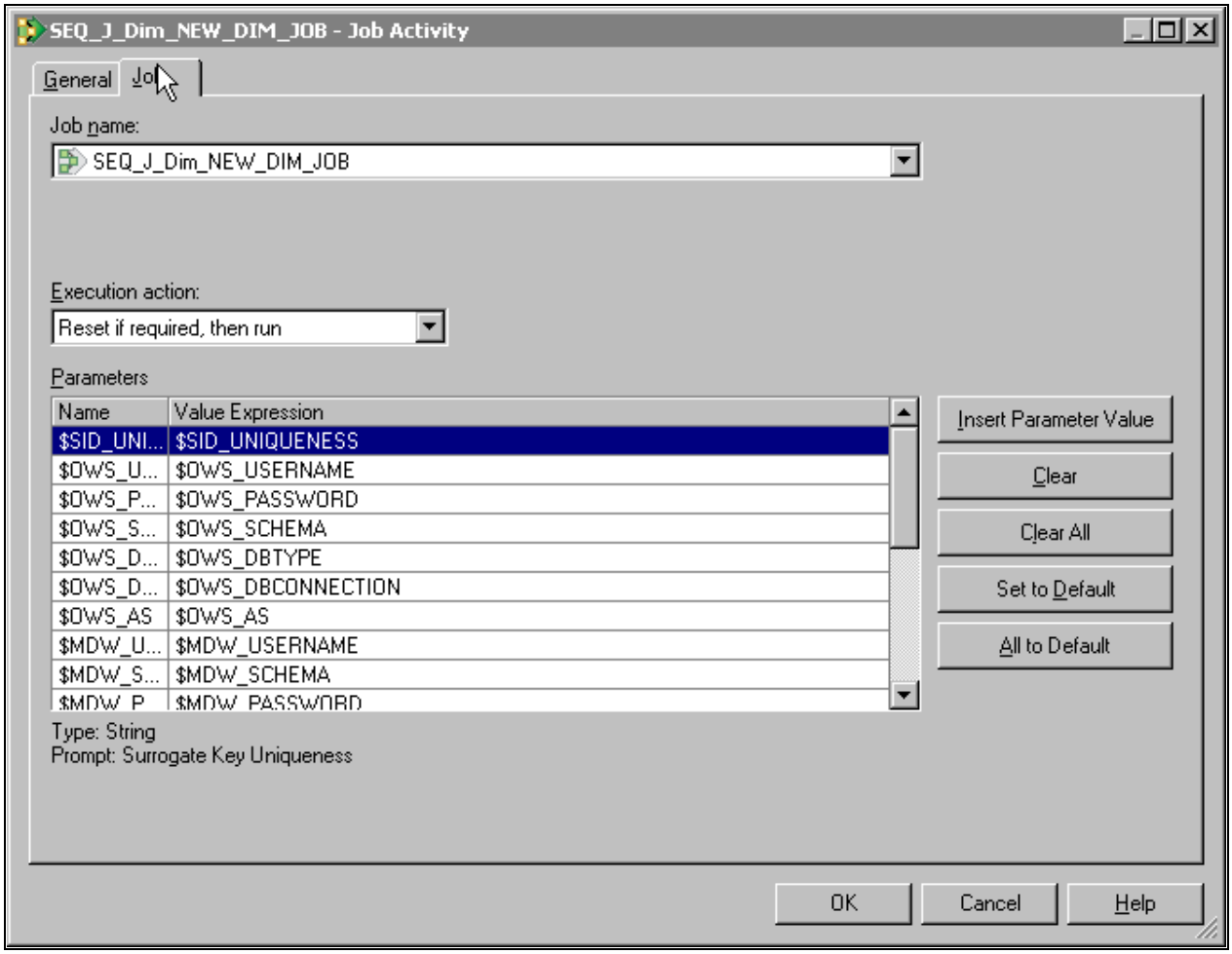
Job Activity box with Triggers tab selected

By default, the expression type on the triggers is set to *OK – (Conditional)*. This condition can be modified if you have a different business requirement.

6. Open the new job activity for editing.

   The job activity property box appears.

7. In the Jobs tab, change the job name to reflect the name of the new dimension sequence job and modify the value expression in the parameters as appropriate.



Job Activity box with Job tab selected

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, theCompilation 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 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.

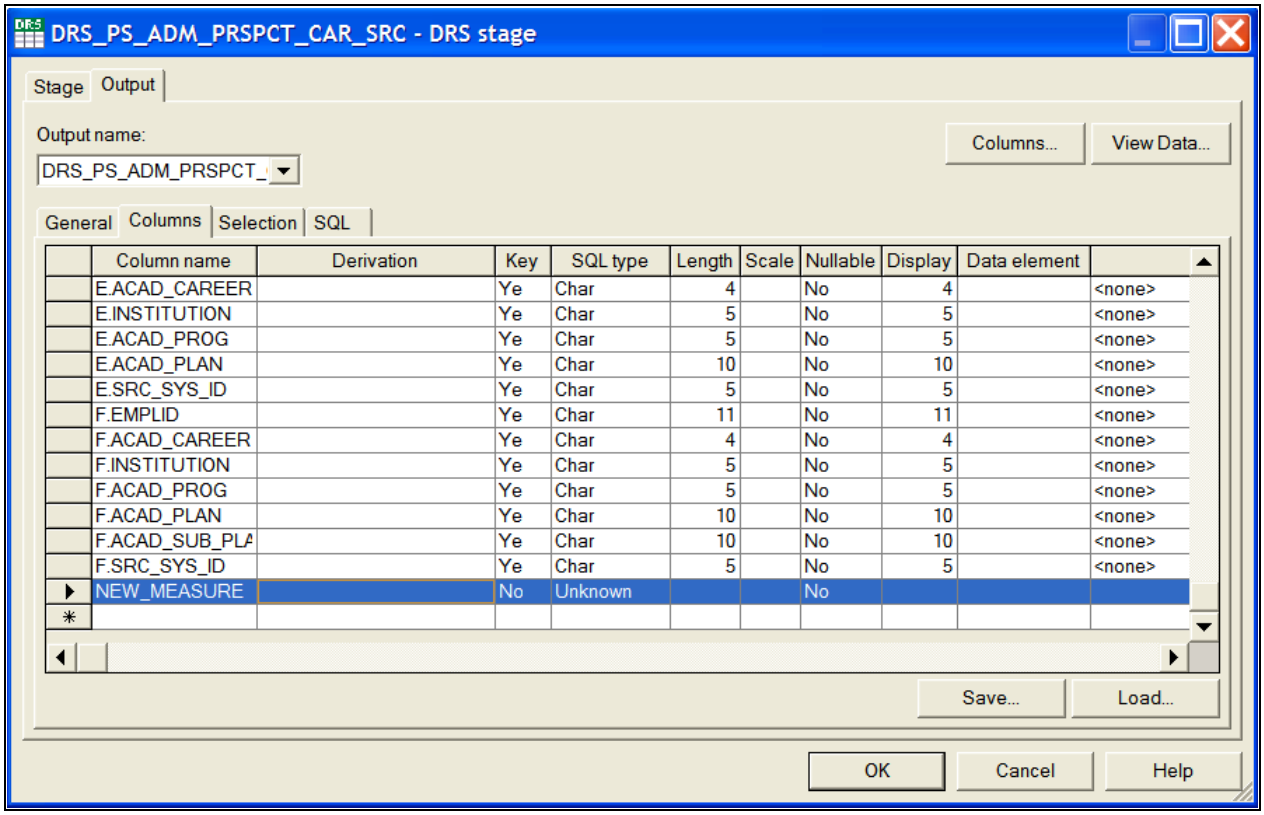
Note. This step is discussed in more detail below.

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.

3. In the Columns sub-tab, add a new row for the new measure.



Adding a new row for the new measure

4. Input the appropriate values for the derivation, key, SQL type, and other applicable properties of the new measure.
5. 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.
6. 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.
7. Link all ports as necessary.
8. Open the target DRS stage and select the Input tab.
9. In the Columns sub-tab ensure that the new measure column is present and properly defined.
10. Select *File, Save* from the menu to save the job.
11. 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, theCompilation Status window displays an error message.

12. If your job successfully compiles, select Close.

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

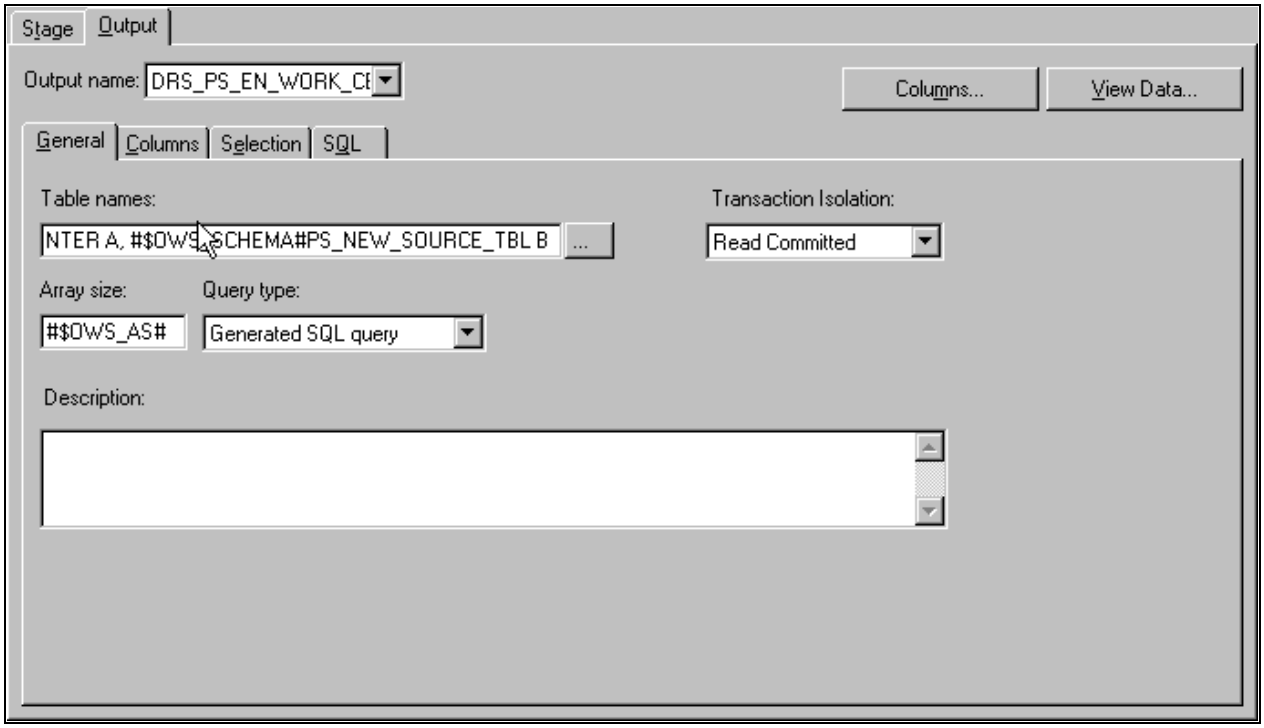
13. You should perform technical unit testing and regression testing on the server job to ensure that the new measure is populated properly.

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:

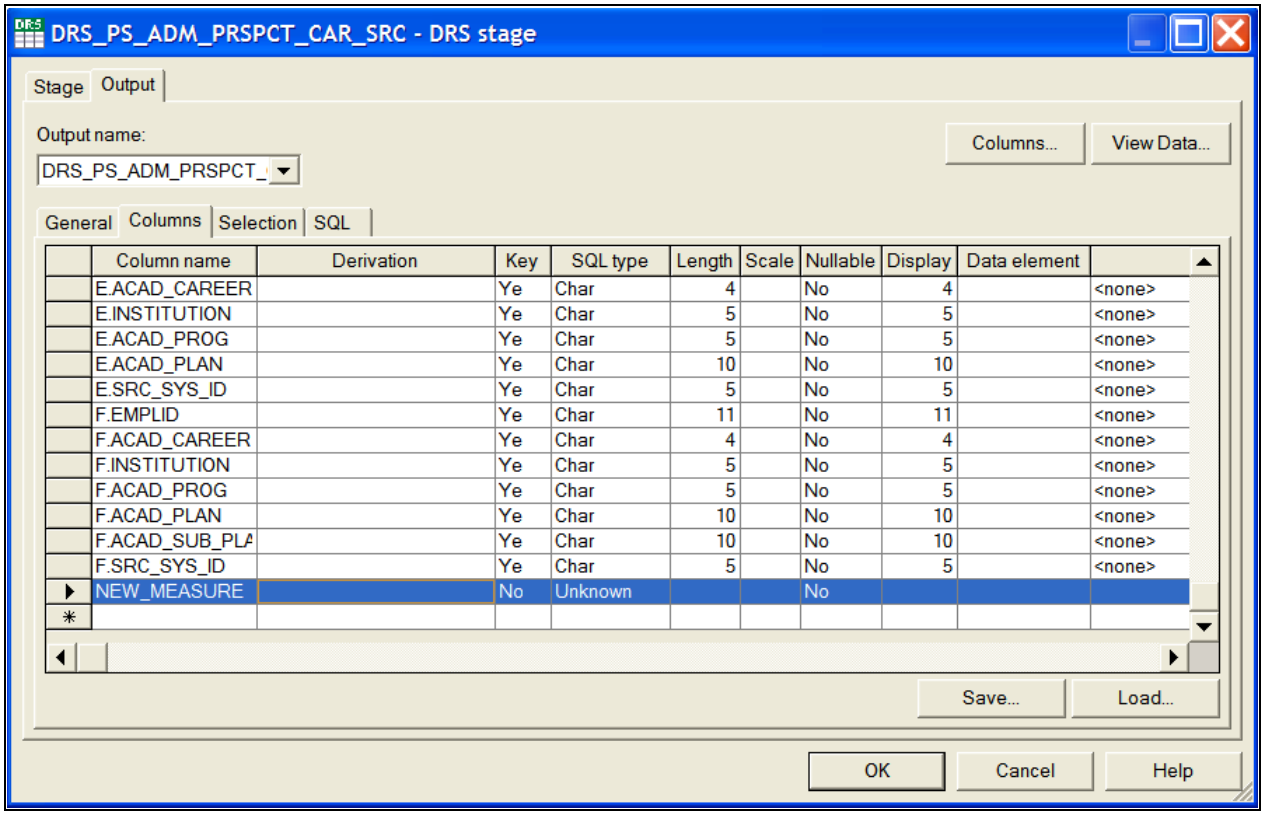
1. In IBM WebSphere DataStage Designer, locate the appropriate fact job and open it for editing.
2. Open the *source DRS stage* and select the Output tab.
3. In the General sub-tab, define the new source table.

   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.



General sub-tab with new source table information

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



Adding a new row for the new measure

5. Input the appropriate values for the derivation, key, SQL type, length, scale, and other applicable properties of the new measure.

   The derivations of the columns must indicate the source table alias for each field.

6. Repeat steps two through four for all the stages between the source DRS and the target DRS, but provide information for the Input tab as well as the Output tab.

   Once the new attribute is defined in the IPC stage, it becomes available on the Transformer Stage - Input Links window.

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

   The logic is defined in the derivations field of the output link for the target table.

8. Link all ports as necessary.

9. Open the target DRS stage and select the Input tab.

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

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

12. Select *File, Compile* from the menu to compile the job.

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

13. If your job successfully compiles, select Close.

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

14. You should perform technical unit testing and regression testing on the server job to ensure that the new measure is populated properly.

Adding a New Surrogate Key to a Fact Table

The MDW data model enables you to add new dimension tables to it. If you add a new dimension table, you must update the corresponding fact table with the primary/foreign key relationship.

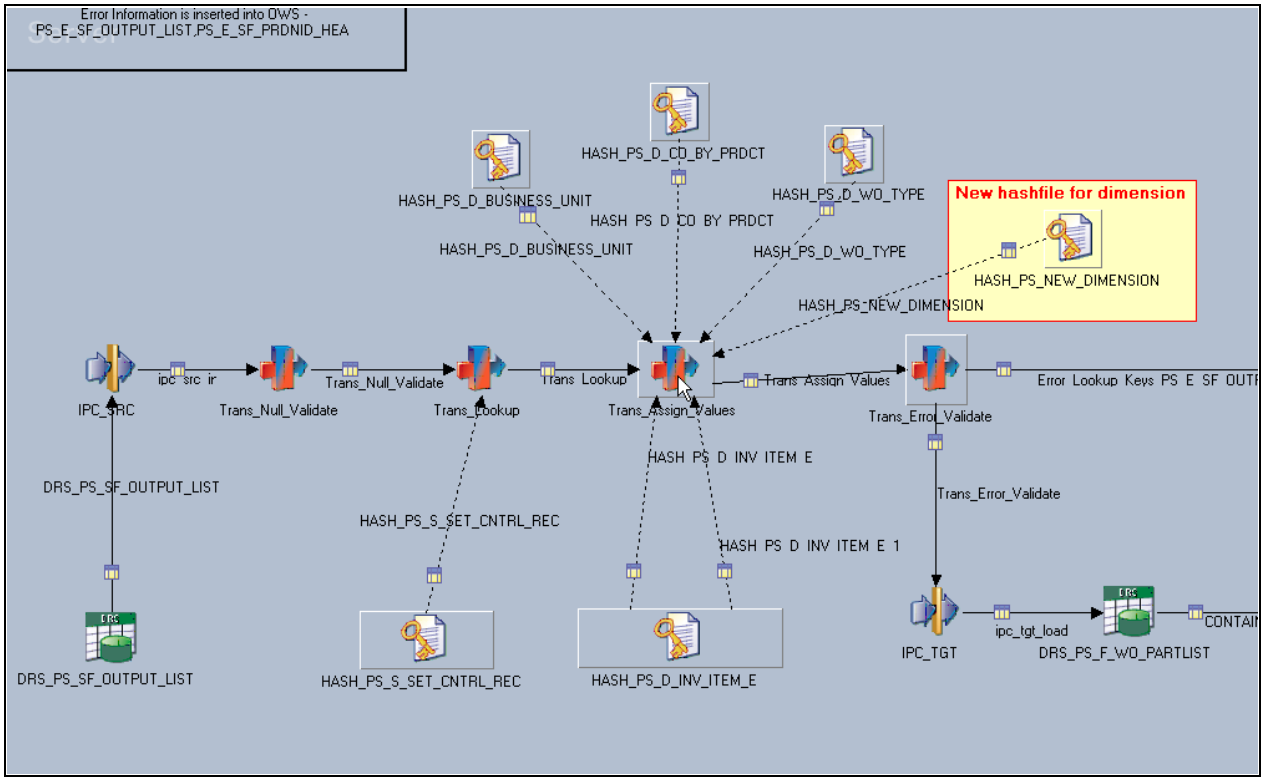
The dimension can be associated with the fact table by adding a new surrogate ID (SID) field (the foreign key field) and populating it appropriately with values of the primary key from the associated dimension. To populate a new SID field in a fact table, a new lookup must be performed on a dimension table hash file. Performing a lookup on a dimension table requires a field from the source to be joined with the key fields of the dimension hash file. Existing fields from the source can be used for the join or a new field must be extracted from the source for the join.

Steps to Add a New Surrogate Key to a Fact Table

Perform the following steps to add a new surrogate key:

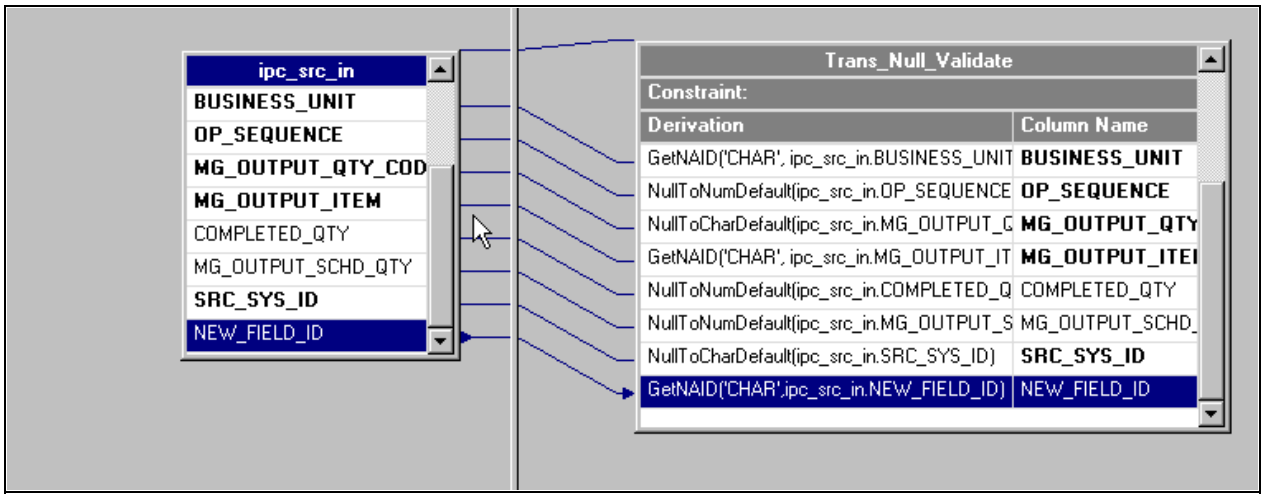
1. In IBM WebSphere DataStage Designer, locate the appropriate fact job and open it for editing.

2. Add a new *dimension table hash file* for the new dimension you have added to the data model.



Dimension table hash file added

3. Open the first transformer stage that follows the source (usually the *Trans_Null_Validate* stage).



Trans_Null_Validate Stage

4. The new field must be processed by the *GetNAID* routine.

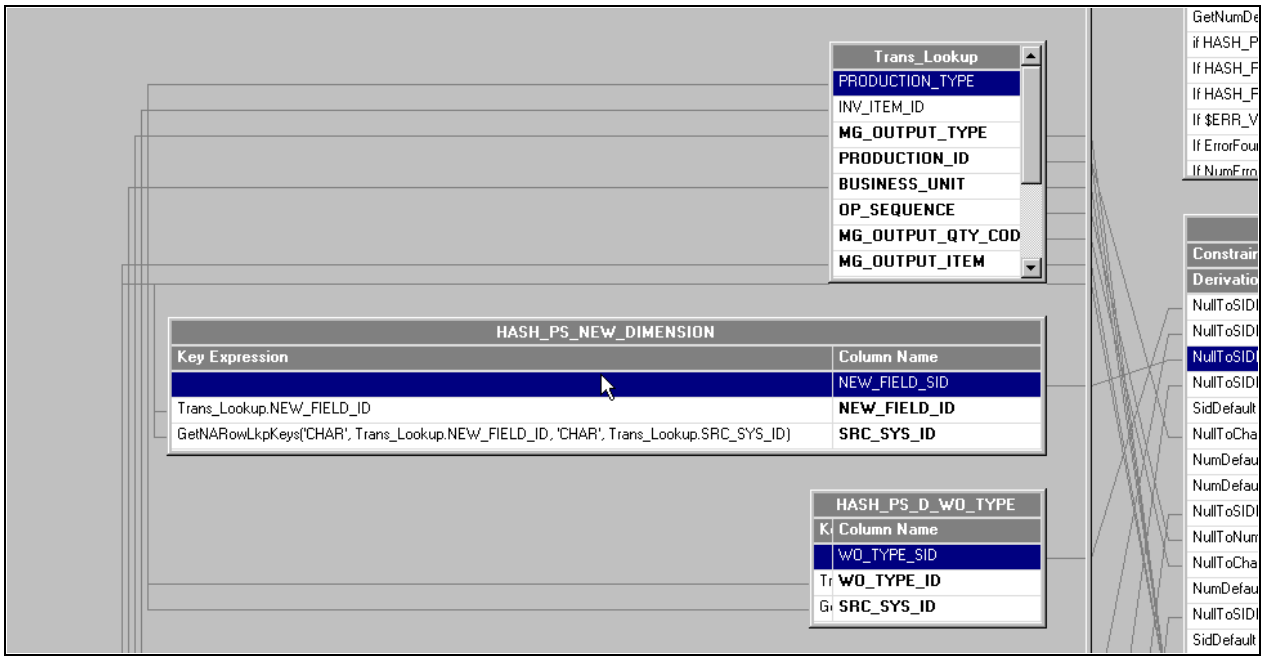
5. Link the new field ID to all succeeding stages (up to the transformer stage) where the new hash file lookup is connected.

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.

---

6. 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.



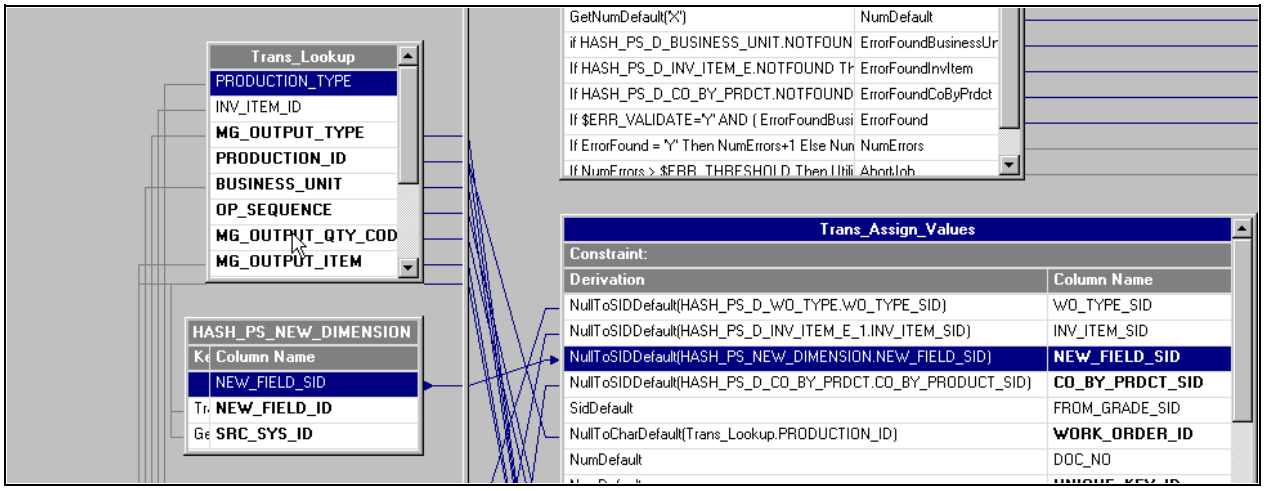
Transformer and Dimension Hash file join

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

---

7. Once the key fields of the dimension table are matched with the input fields, the SID is extracted to the target SID field in the fact table.

   This field must be validated by the *NullToSIDDefault* routine in case the lookup results in a null value.



   SID extract

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, theCompilation 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.

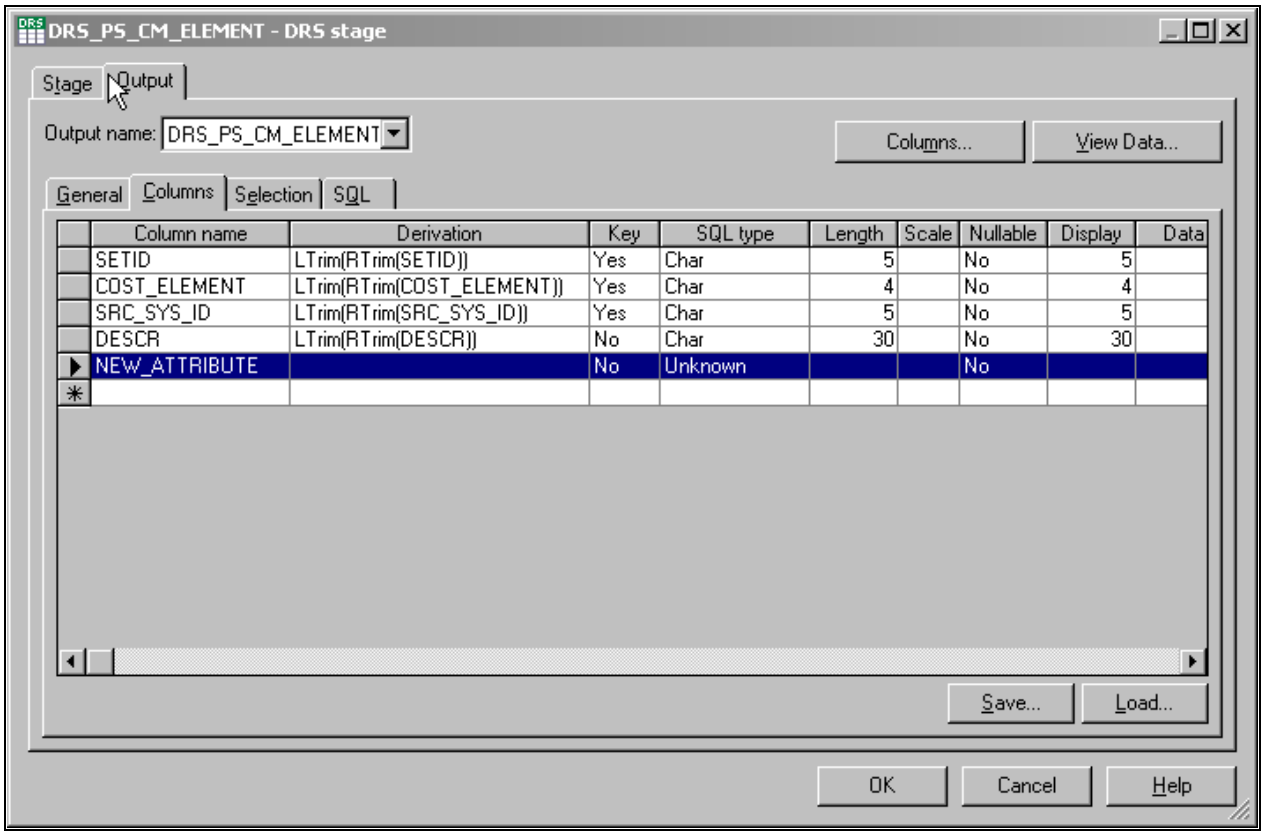
---

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.



DRS Stage with Columns sub-tab selected

3. In the Columns sub-tab, add a new row for the new attribute.
4. Input the appropriate values for the derivation, data type, data size, and other applicable properties of the new attribute.

5. Repeat steps two through four for the *IPC stage* but provide information for the Input tab as well as the Output tab.

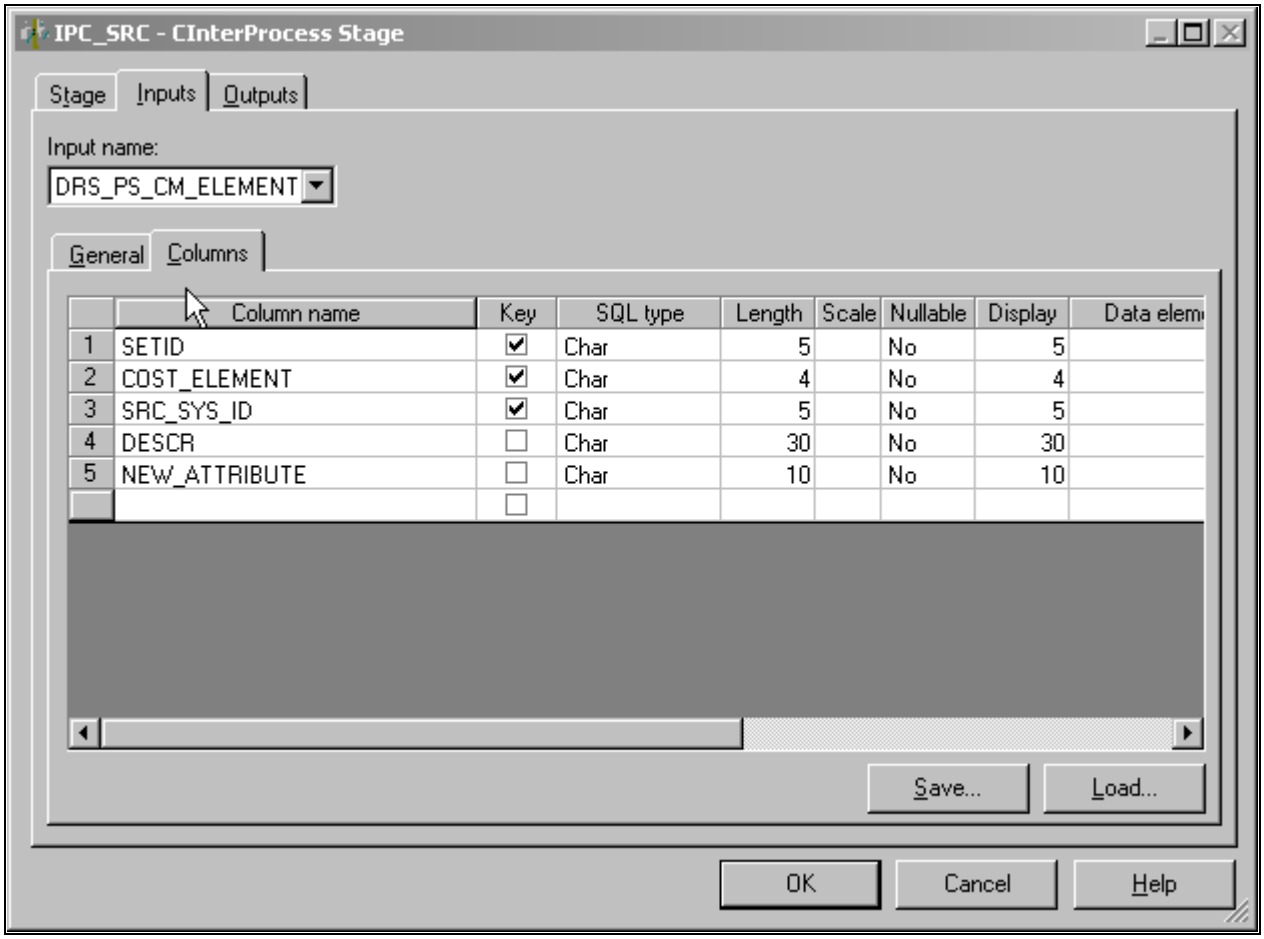
   Once the new attribute is defined in the IPC stage, it becomes available on the Transformer Stage - Input Links window.

6. 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.

7. Connect the output link of the transformer stage to the target dimension table.

8. Open the IPC stage and select the Inputs tab.

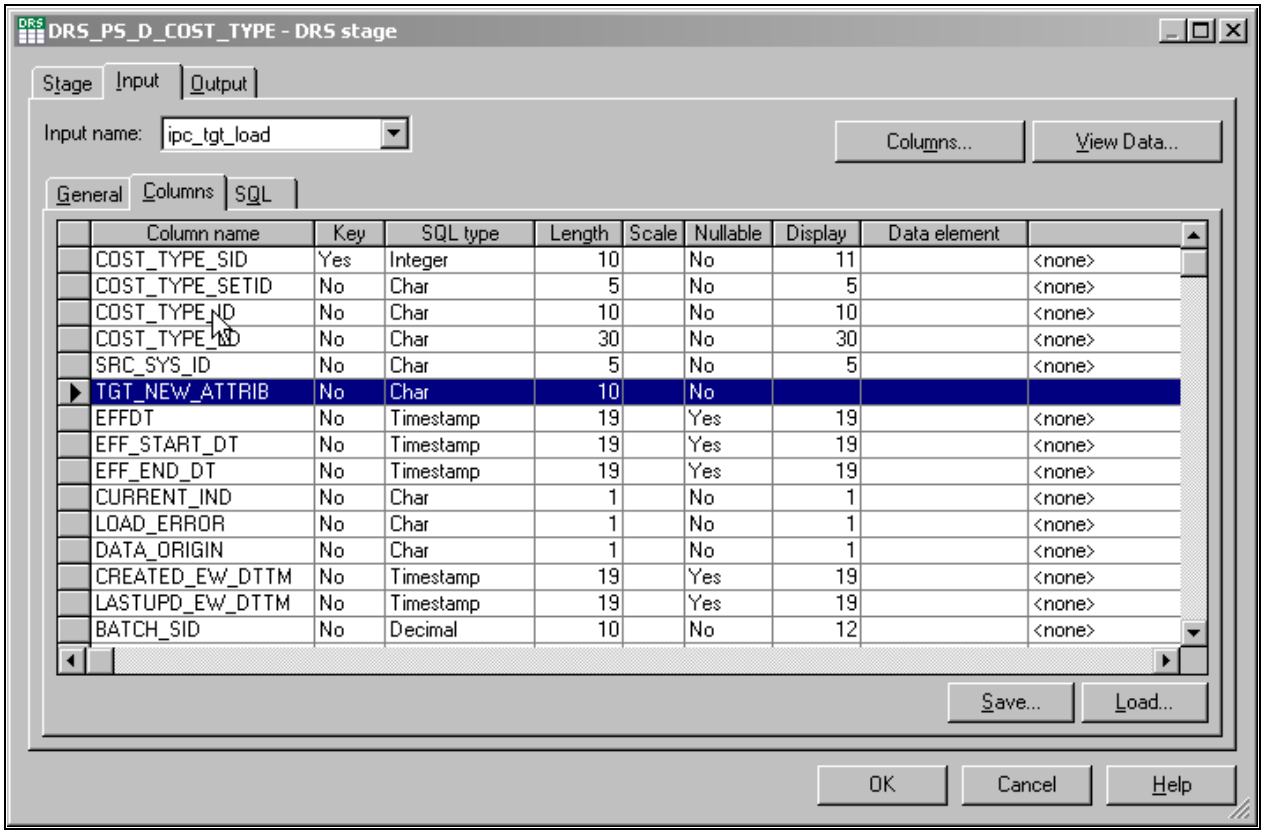


IPC_SRC Stage

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

10. Open the target DRS stage and select the Input tab.

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



Target DRS stage with new attribute row

12. Select *File, Save* from the menu to save the job.
13. Select *File, Compile* from the menu to compile the job.

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

14. If your job successfully compiles, select Close.

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

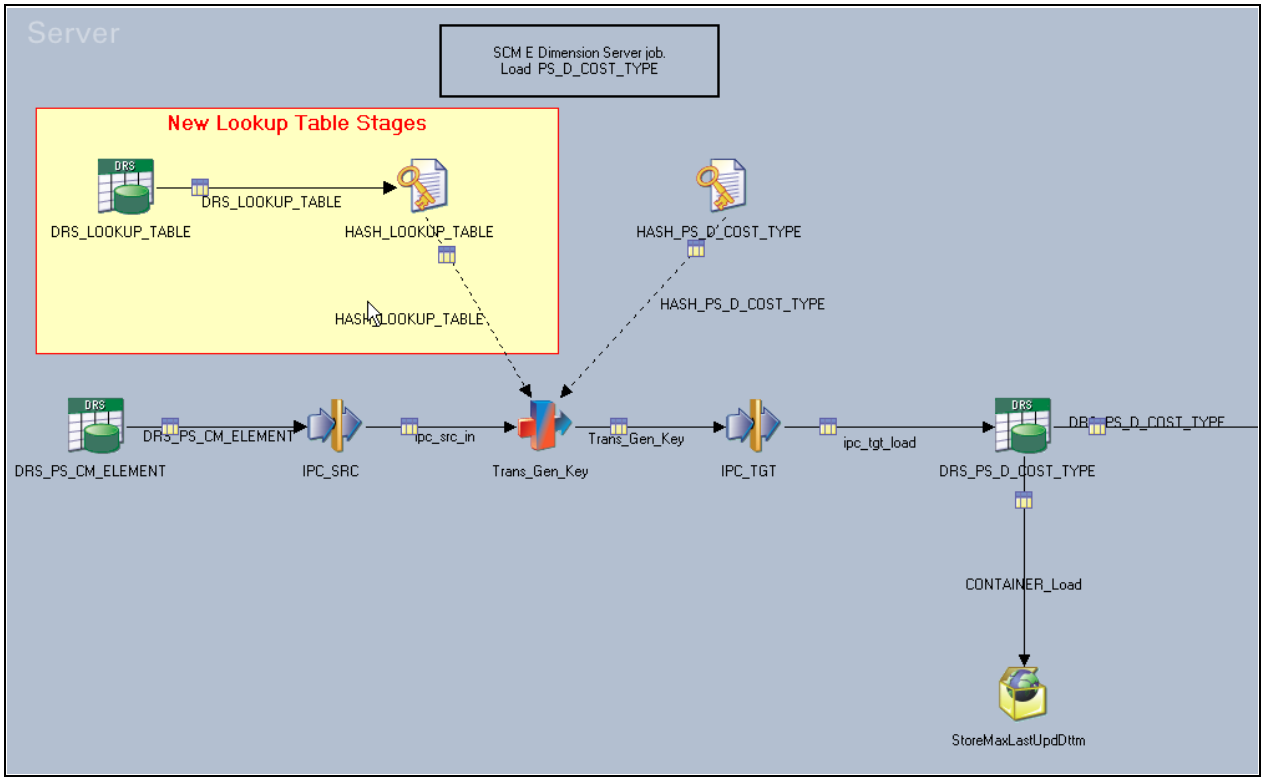
15. You should perform technical unit testing and regression testing on the server job to ensure that the new attribute is populated properly.

Updating a Dimension Job with a New Attribute That Originates from a Lookup Table

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

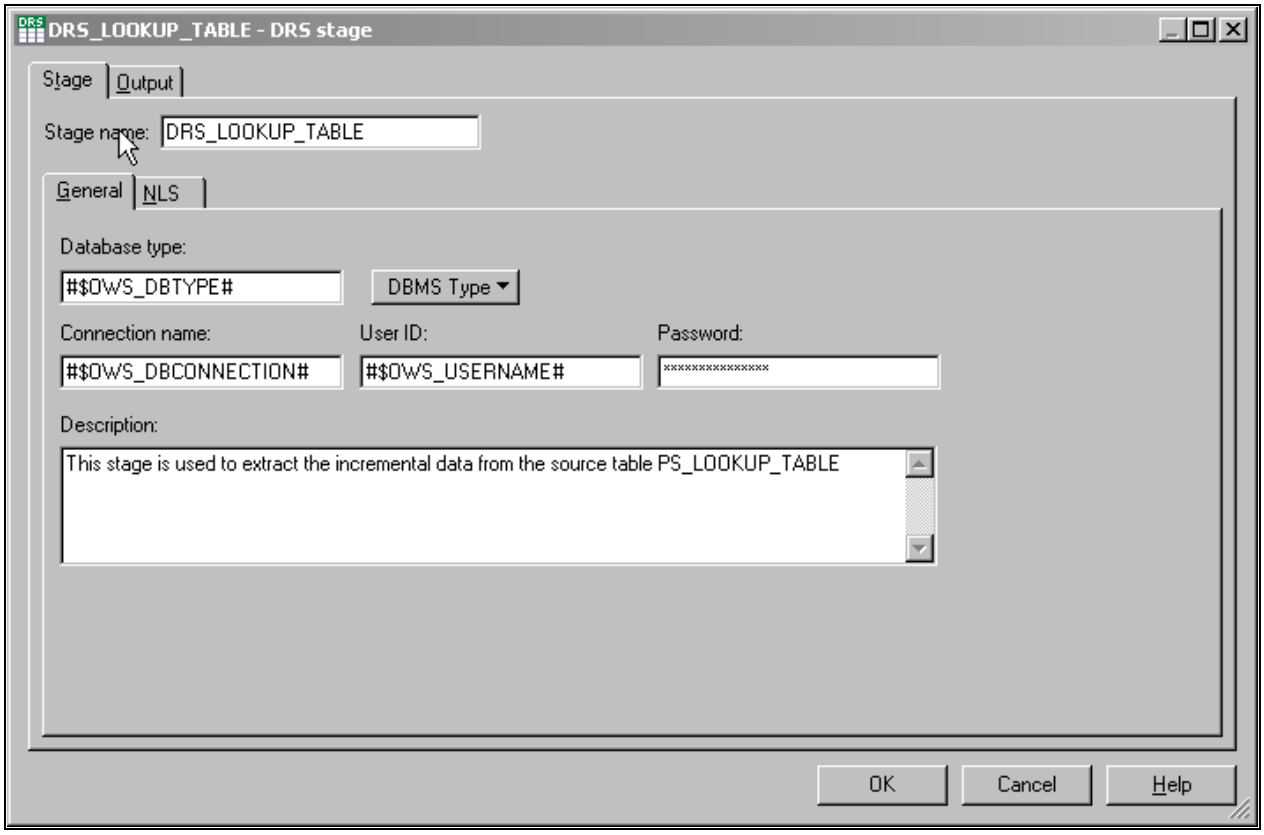
1. In IBM WebSphere DataStage Designer, locate the appropriate dimension job and open it for editing.

2. Create a new DRS lookup stage and open it for editing.



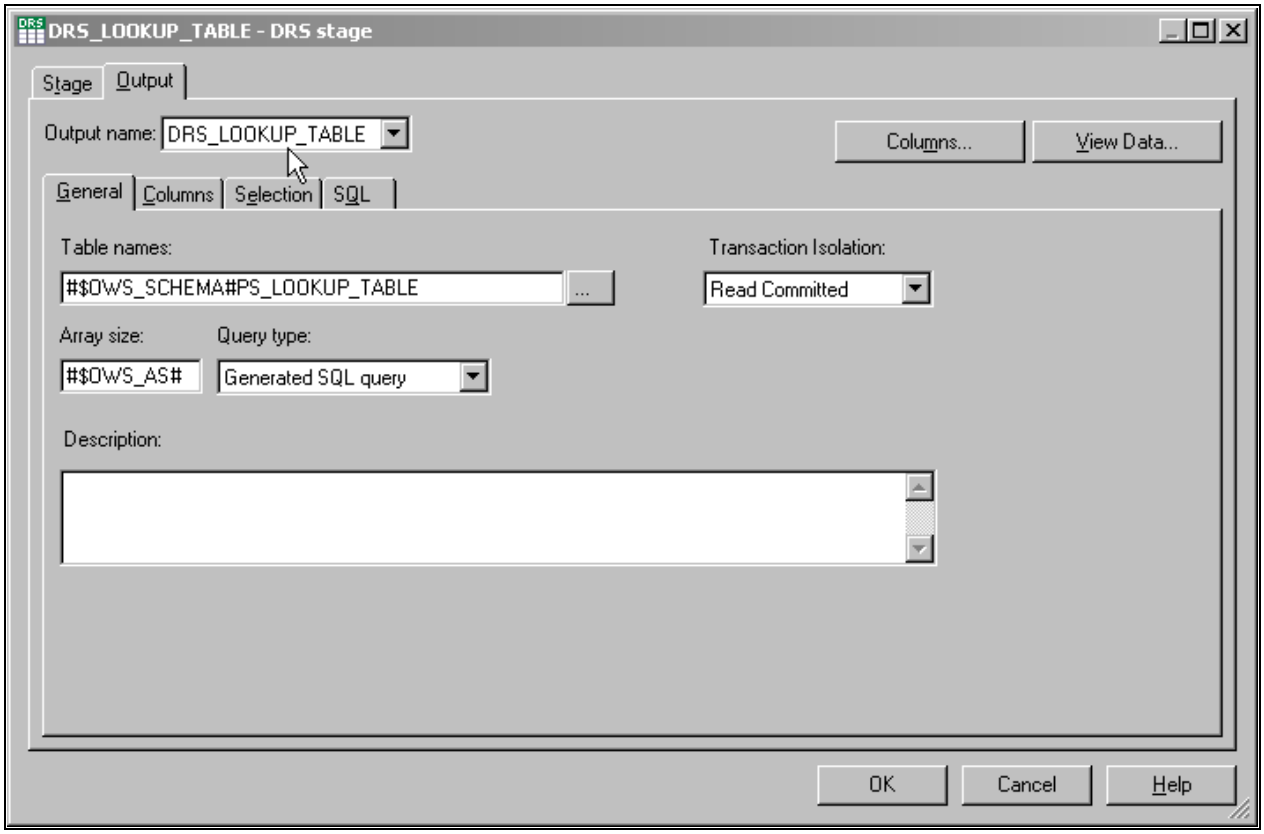
New lookup table stages

3. Select the main Stage tab and input the appropriate values for database type, connection name, user ID and user password in the General sub-tab.



DRS Lookup with Stage tab and General sub-tab selected

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

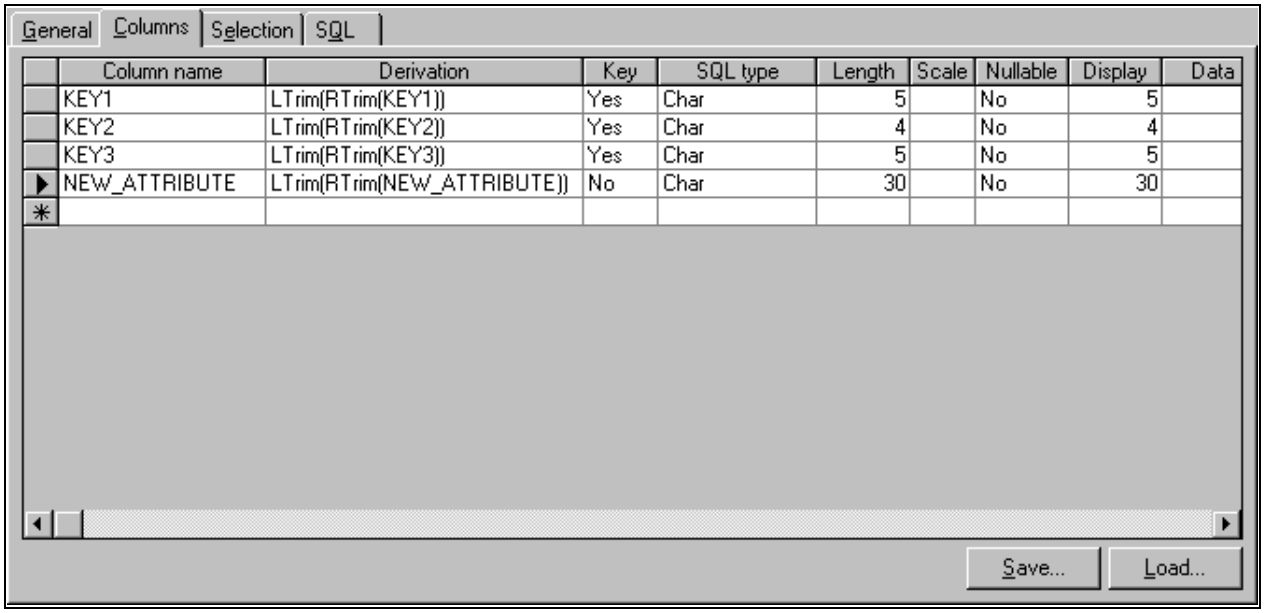


DRS Lookup with Output tab and General sub-tab selected

5. Input the appropriate values for the table names, transaction isolation, array size, and query type.

   The query type can be user defined or generated by SQL.

6. Select the Columns sub-tab and add a new row for the new attribute.

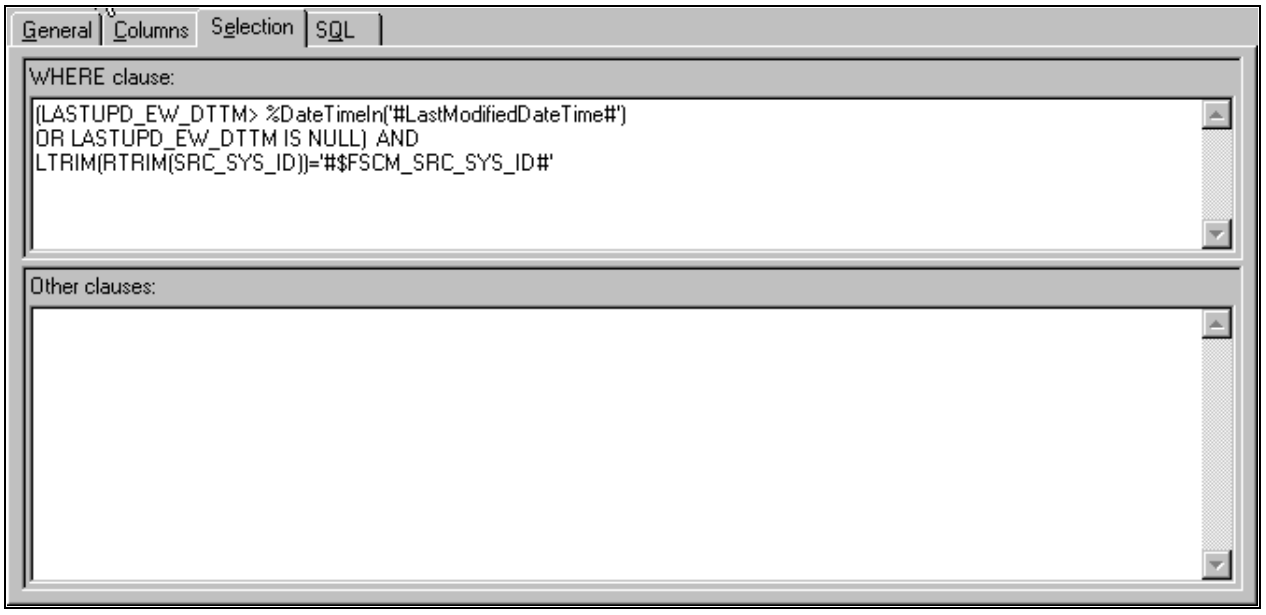


DRS Lookup with Output tab and Columns sub-tab selected

7. Input the appropriate values for the derivation, key, SQL type, length, scale, and other applicable properties of the new attribute.

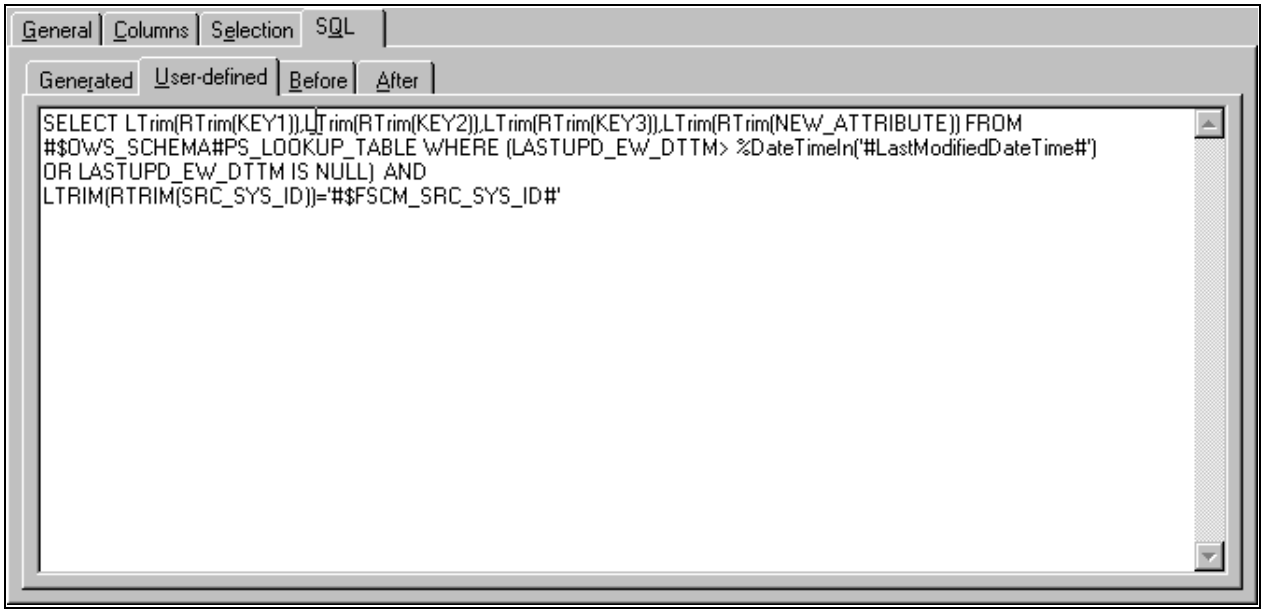
   The key fields must be marked appropriately as they are used to extract the value for the new attribute.

8. Select the Selection sub-tab and input any selection criteria for the attribute.



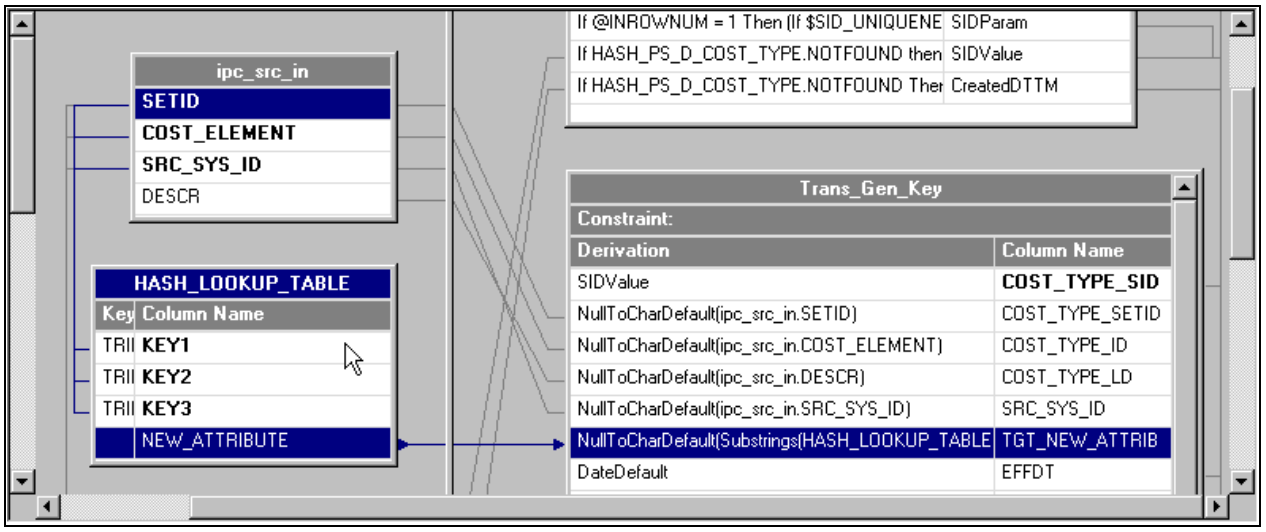
DRS Lookup with Output tab and Selection sub-tab selected

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



DRS Lookup with user defined SQL

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



Attribute to target mapping

11. Open the aforementioned hash file stage for editing and select the Inputs tab.

12. On the Inputs tab, input the appropriate file name and description, and select the options that are applicable to the attribute.

13. Select the Columns sub-tab and add a new row for the new attribute.

14. Input the appropriate values for the key, SQL type, length, scale, and other applicable properties of the new attribute.

    Note. The column definitions must match those defined on the Output tab in the DRS lookup stage.

    The hash file name must match the name specified in the Inputs and Outputs tabs. The hash file will provide erroneous values if the column definitions and hash file names are synchronized between the aforementioned tabs in the hash file stage.

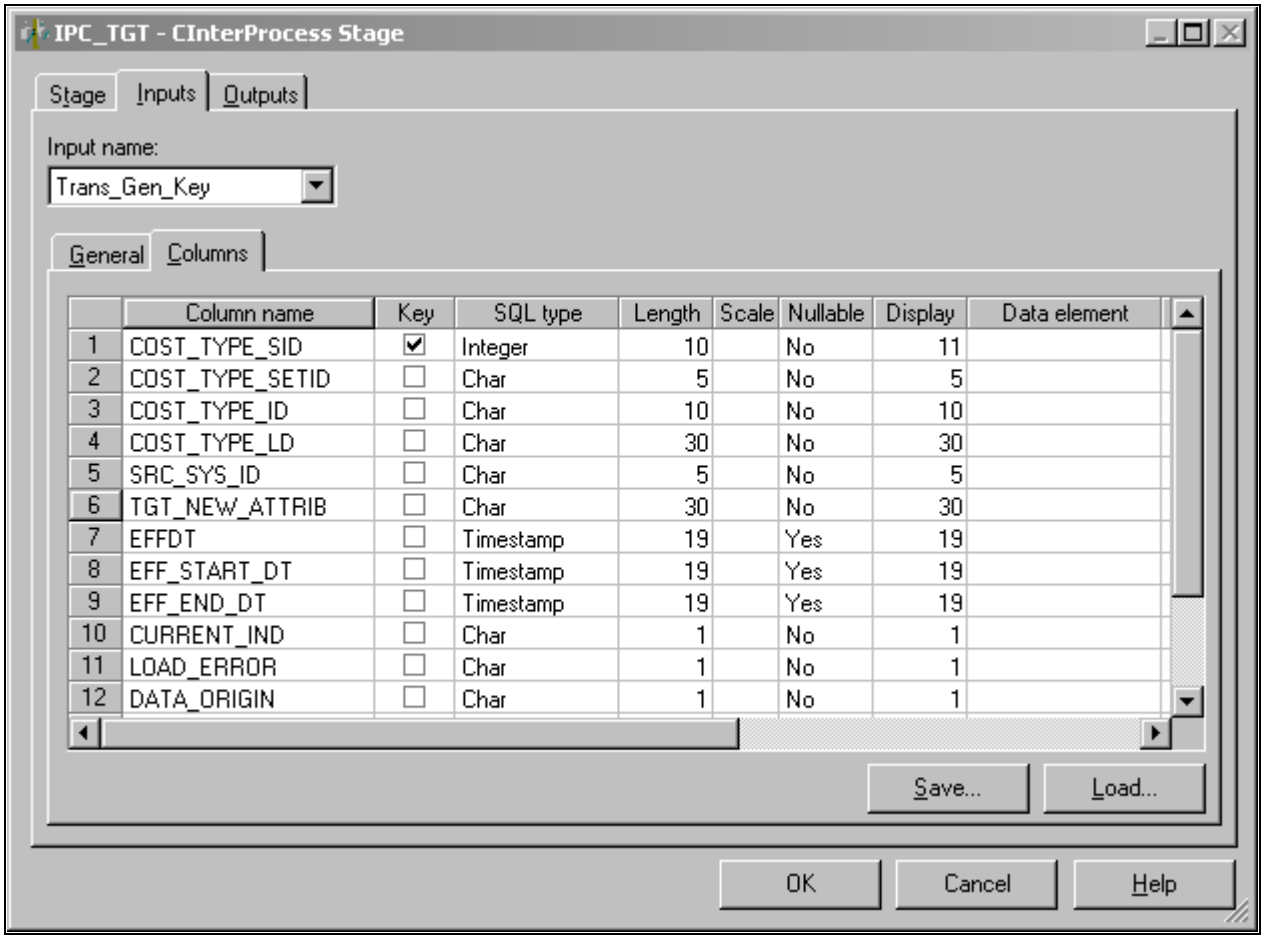
    ---

15. Connect the output link of the hash file stage to the transformer stage (Trans_Gen_Key).

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

16. In the inputs pane of the transformer stage, define the key expression for each key field.

    The value of the key expressions is sourced from the main input link (ipc_src_in) of the transformer stage. Parameters and constant values can also be used to match with the key fields of the lookup table.



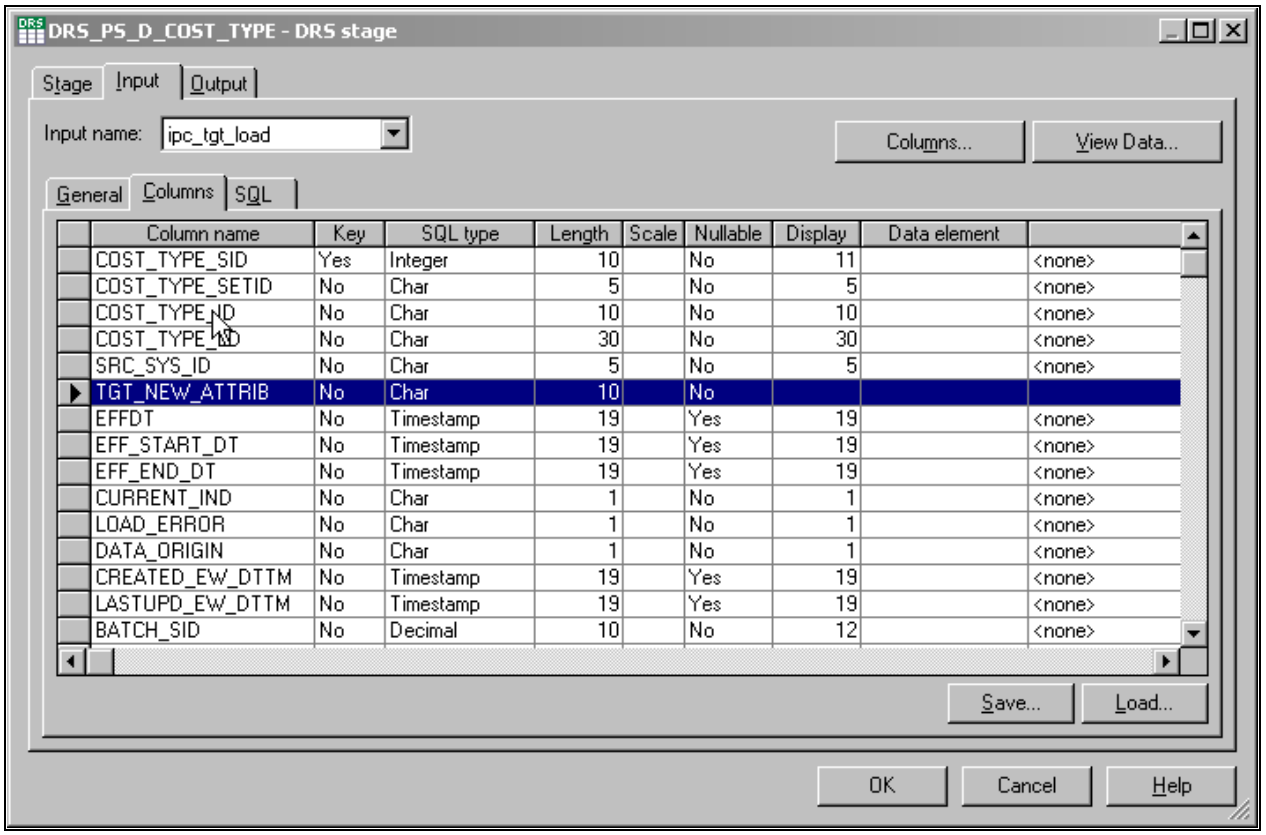
	Column name	Key	SQL type	Length	Scale	Nullable	Display	Data element
1	COST_TYPE_SID	☑	Integer	10		No	11	
2	COST_TYPE_SETID	☐	Char	5		No	5	
3	COST_TYPE_ID	☐	Char	10		No	10	
4	COST_TYPE_LD	☐	Char	30		No	30	
5	SRC_SYS_ID	☐	Char	5		No	5	
6	TGT_NEW_ATTRIB	☐	Char	30		No	30	
7	EFFDT	☐	Timestamp	19		Yes	19	
8	EFF_START_DT	☐	Timestamp	19		Yes	19	
9	EFF_END_DT	☐	Timestamp	19		Yes	19	
10	CURRENT_IND	☐	Char	1		No	1	
11	LOAD_ERROR	☐	Char	1		No	1	
12	DATA_ORIGIN	☐	Char	1		No	1	

IPC Stage with column definitions

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

    The logic is defined in the derivations field of the output link for the target table.

18. Connect the output link of the transformer stage to the target dimension table.
19. Open the IPC stage and select the Inputs tab.
20. In the Columns sub-tab ensure that the new attribute column is present and properly defined.



Target DRS stage with new attribute row

21. Open the target DRS stage and select the Input tab.
22. In the Columns sub-tab ensure that the new attribute column is present and properly defined.
23. Select *File, Save* from the menu to save the job.
24. Select *File, Compile* from the menu to compile the job.

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

25. If your job successfully compiles, select Close.

    If 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.

Appendix A

Using the PeopleSoft EPM Lineage Spreadsheets

This document provides an overview of the EPM lineage spreadsheets and discusses how to use the spreadsheets to:

- View lineage information.
- Generate lineage information for a job.

Understanding the EPM Lineage Spreadsheets

The EPM lineage spreadsheets provide information about the ETL jobs that are delivered with the EPM warehouses. The spreadsheets act like a reverse-engineering tool or family tree; they enable you to view the ancestry of source, target, and lookup tables and their relevant ETL jobs. Each spreadsheet provides lineage information for a single warehouse. The following table lists the lineage spreadsheets that are currently available:

Lineage Spreadsheet Filename	***Warehouse***
ETL_CS_Lineage_Spreadsheet.xls	Campus Solutions Warehouse
ETL_CRM_Lineage_Spreadsheet.xls	CRM Warehouse
ETL_FMS_Lineage_Spreadsheet.xls	FMS Warehouse
ETL_HCM_Lineage_Spreadsheet.xls	HCM Warehouse
ETL_SCM_Lineage_Spreadsheet.xls	SCM Warehouse

By using the spreadsheets, you can:

- View lineage information for staging, dimension, and fact ETL jobs, or source, target, and lookup tables.
- Identify the sequence of jobs to run for a specific data mart.
- Identify inter-mart and cross-warehouse dependencies.
- Generate lineage information for a specific ETL job.

Spreadsheet Structure

Each EPM lineage spreadsheet includes several worksheets. The following table provides a description of each worksheet, by name, listed in the order in which it appears:

Worksheet	Description
Template	This worksheet contains overview information, a legend, and a definition of the columns used in the worksheets.
Setup	This worksheet contains ETL lineage information for all of the setup and staging jobs required for the warehouse.
Com Dims	This worksheet contains ETL lineage information for the common dimension jobs required for the warehouse.
Utils	This worksheet contains ETL lineage information for the currency conversion jobs required for the warehouse.
Global Dims	This worksheet contains ETL lineage information for the global dimension jobs required for the warehouse.
Local Dims	This worksheet contains ETL lineage information for the local dimension jobs required for the warehouse.
<*Data Mart*> For example: GL & Profitablity, ESA, Campus Community, and so on.	This worksheet contains ETL lineage information for the jobs required for a specific data mart. **Note.** Each spreadsheet includes several data mart worksheets.
Dynamic_Lineage_Generator	This worksheet provides a macro that enables you to enter the name of an ETL job and automatically generate a list of the complete lineage for that job.
JobOrder	This worksheet is an extension of the Dynamic_Lineage_Generator worksheet. It displays the order in which jobs need to be run.

Column Descriptions

The following table provides descriptions of the columns in the worksheets.

Column	Description
Sequencer Job	The name of the job sequencer, which is responsible for invoking and running other ETL server jobs.
Server Job	The name of the server job that is called by the job sequencer.

Column	*Description*
Server Job Category	The location of the server job in the IBM WebSphere DataStage project.
Target Table	The name of the target table used in the server job.
Target Update Action	The target load strategy for the server job.
Source Table	The name of the source table used in the server job.
Source Extraction Type	The type of extraction from the source table in the server job (for example, incremental date time or cyclical redundancy check).
Lookup Tables	The name of the lookup tables that are used in the server job. Lookups can be hashed files or direct DRS lookups. The lineage information captures the table names from which the hash files are populated and the table names for the direct DRS lookup.
Setup Jobs	The name of the setup job that populates the source and/or the lookup table.
Setup Sequencer Job	The name of the job sequencer that calls the setup server job.
MDW	The name of the MDW server job. This column has an entry if the source table or lookup table is populated from an MDW server job.
MDW Sequencer	The name of the MDW sequence job.
OWS	The name of the OWS server job. This column has an entry if the source table or lookup tables are populated from an OWS server job.
OWS Sequencer	The name of the OWS sequence job.
OWE	The name of the OWE server job. This column has an entry if the source table or lookup tables are populated from an OWE server job.
OWE Sequencer	The name of the OWE sequence job.
EPM Foundation	The application or EPM foundation setup page that populates the source table or the lookup table, such as Global Consolidations, Dimension Mapper, or setup PIA pages.
Category	The categories in which the setup jobs, MDW jobs, OWS jobs or OWE jobs are placed.

Column	*Description*
Comments	Any additional comments, if applicable.

Note. The spreadsheet does not contain lineage details for OWE jobs and Tree jobs, except for the GL&Profitability Mart of the FMS warehouse, which does include lineage information for OWE jobs.

Viewing Lineage Information

This section discusses how to use the spreadsheet to:

- Find lineage information for a server job.
- Identify the list of Jobs to be run for a data mart.

Finding Lineage Information for a Server Job

To find lineage information for a server job:

1. Access the worksheet in which the job is categorized.
2. Use Excel's Find feature to find the server job name in column B.
   a. Type Ctrl-F to access the Find and Replace Dialog box.
   b. Enter the name of the server job in the Find what edit box.
   c. Click Find Next until the job name is found in the Server Job column (column B).
   d. Close the Find dialog box.
3. Review the lineage information in the adjacent columns.

   The Sequencer Job column (column A) lists the sequencer which calls this job. The Server Job Category column (column C) lists the category this job is associated with. The Target Table, Target Update Action, Source Table, and Source Extraction Type for this server job are listed in columns D, E, F, and G respectively. The Lookup Tables column (Column H) lists all the lookups used by this job.

   The source tables and the lookup tables are placed in separate rows. This enables you to find the lineage information for each of these tables by navigating through the other subsequent columns in the same row. Columns I through R list the dependent jobs that are required to populate the source and lookup tables, and entries in these columns indicate whether the table is populated by Setup jobs, (column I), MDW jobs (column K), OWS jobs (column M), OWE jobs (column O), or Foundation setup / Apps (column Q). The Category column (column R) lists the category that the dependent job is associated with.

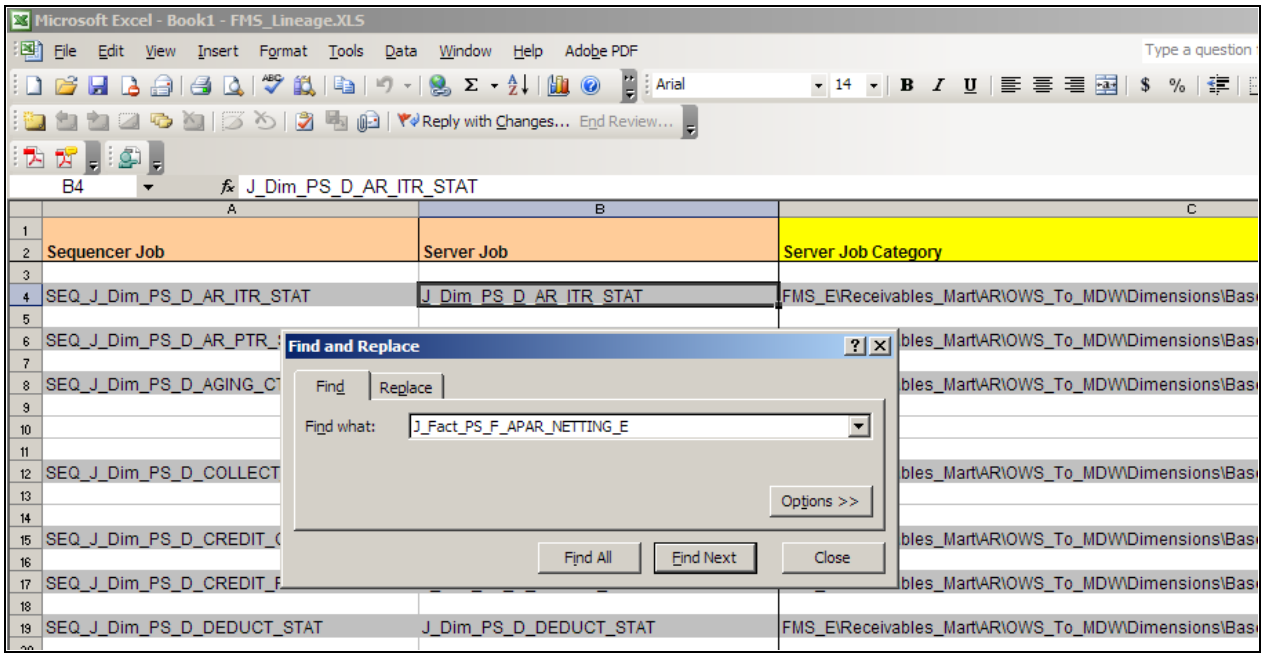
   Source tables that are from a different data mart (inter-mart) or different warehouse (cross-warehouse) are indicated by the colors specified in the legend on the Template worksheet page.

The spreadsheet lists the lineage of a source or lookup table to the level of the job that directly populates it. The lineage information does not extend to level of the last staging job. To get the complete lineage for a fact or dimension job fully extended through the lowest staging level, you can use the dynamic lineage generator tool, which generates a list of all the required dependent jobs that need to be run in order to load a particular fact or dimension.

Example

This example, from the ETL FMS Lineage spreadsheet, takes you through the tasks you would complete to review the information for the fact job J_Fact_PS_F_APAR_NETTING_E, which is used for the AR Data Mart.

1. Navigate to the AR worksheet page.
2. Type Ctrl-F and type J_Fact_PS_F_APAR_NETTING_E into the Find and Replace dialog box.



Find and Replace Dialog Box

3. Type Ctrl-F and type J_Fact_PS_F_APAR_NETTING_E into the Find and Replace dialog box.
4. Click Find Next until you access the cell in the Server Job column that contains the J_Fact_PS_F_APAR_NETTING_E job.

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

	A	B	
1			
2	Sequencer Job	Server Job	Server Job Category
90			
91	SEQ_Dims_L_O_CREDIT_CLASS	J_Dim_PS_O_CREDIT_CLASS	FMS_E\Receivables_Mart\AR\OWS_T
92			
93			
94	SEQ_Dims_L_O_CREDIT_RISK	J_Dim_PS_O_CREDIT_RISK	FMS_E\Receivables_Mart\AR\OWS_T
95			
96			
97	SEQ_Dims_L_O_DEDUCT_STAT	J_Dim_PS_O_DEDUCT_STAT	FMS_E\Receivables_Mart\AR\OWS_T
98			
99			
100	SEQ_Dims_L_O_DISPUTE_STAT	J_Dim_PS_O_DISPUTE_STAT	FMS_E\Receivables_Mart\AR\OWS_T
101			
102			
103	SEQ_Dims_L_O_ENTRY_RSTYP	J_Dim_PS_O_ENTRY_RSTYP	FMS_E\Receivables_Mart\AR\OWS_T
104			
105			
106	SEQ_J_Fact_PS_F_APAR_NETTING_E	J_Fact_PS_F_APAR_NETTING_E	FMS_E\Receivables_Mart\AR\OWS_T
107			
108			
109			
110			
111	SEQ_J_Fact_PS_F_AR_ACCOUNT_LN_E	J_Fact_PS_F_AR_ACCOUNT_LN_E_ITEM	FMS_E\Receivables_Mart\AR\OWS_T

J_Fact_PS_F_APAR_NETTING_E job displayed in spreadsheet

6. Scroll to the right to review the columns shown here:

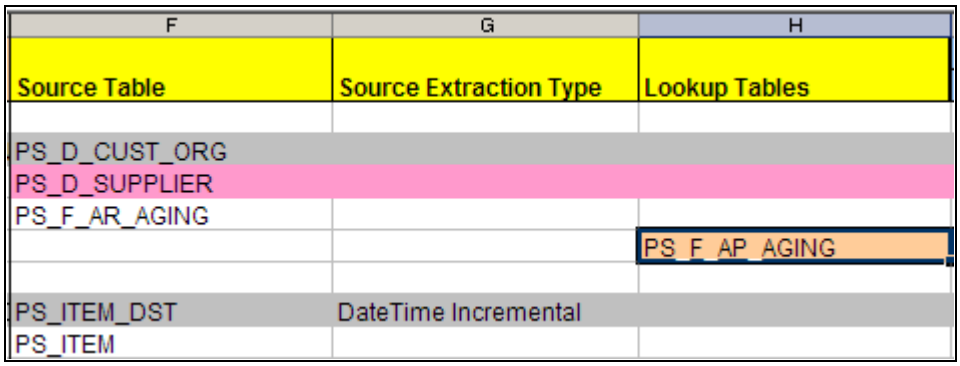
D	E	F	G
Target Table	Target Update Action	Source Table	Source Extraction Type
		PS_D_ENTRY_RSTYP	
PS_F_APAR_NETTING	Truncate table then insert rows	PS_D_CUST_ORG	
		PS_D_SUPPLIER	
		PS_F_AR_AGING	
PS_F_AR_ACCOUNT_LN	Insert new rows or update existing ones	PS_ITEM_DST	DateTime Incremental
		PS_ITEM	

Reviewing data associated with the J_Fact_PS_F_APAR_NETTING_E job

The Target Table, Target Update Action, Source Table, and Source Extraction Type for the J_Fact_PS_F_APAR_NETTING_E server job are listed in columns D, E, F, and G, respectively.

7. Continue to scroll to the right to view the remaining columns.

   The Lookup Tables column (Column H) lists all the lookups used in J_Fact_PS_F_APAR_NETTING_E.



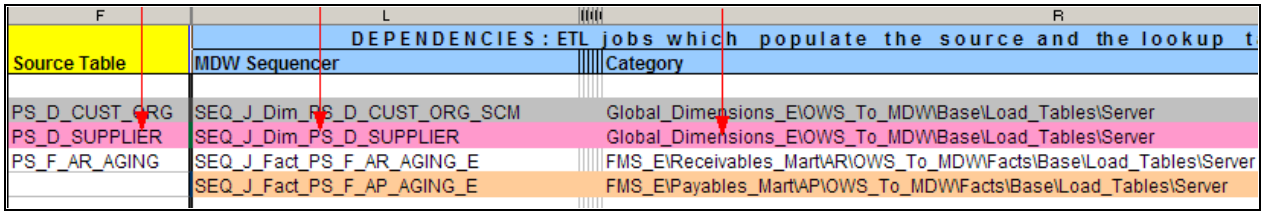
F	G	H
Source Table	Source Extraction Type	Lookup Tables
PS_D_CUST_ORG		
PS_D_SUPPLIER		
PS_F_AR_AGING		
		PS_F_AP_AGING
PS_ITEM_DST	DateTime Incremental	
PS_ITEM		

Lookup Tables Column

   In this example there are three source tables: PS_D_CUST_ORG, PS_D_SUPPLIER, PS_F_AR_AGING. The lookup table is PS_F_AP_AGING. The source tables and the lookup tables are each placed in a unique row one after the other. This enables you to view the lineage information for each of these tables by navigating through the succeeding columns within the same row.

   Columns I through R list out the dependent jobs required to populate these source and lookup tables. In this example, the source table PS_D_CUST_ORG has an entry in the MDW column, which means that is is populated from the MDW dimension J_Dim_PS_D_CUST_ORG_SCM, which is placed in the category Global_Dimensions_E\OWS_To_MDW\Base\Load_Tables\Server.

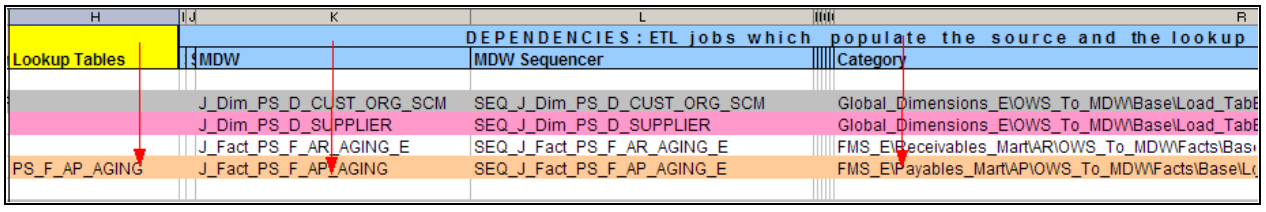
   As shown in the following screenshot, the source table PS_D_SUPPLIER is an SCM warehouse dimension. The cross-warehouse dependency is identified by the different color (the color legend is located on the first worksheet page).



F	L	R
	DEPENDENCIES: ETL jobs which populate the source and the lookup t	
Source Table	MDW Sequencer	Category
PS_D_CUST_ORG	SEQ_J_Dim_PS_D_CUST_ORG_SCM	Global_Dimensions_E\OWS_To_MDW\Base\Load_Tables\Server
PS_D_SUPPLIER	SEQ_J_Dim_PS_D_SUPPLIER	Global_Dimensions_E\OWS_To_MDW\Base\Load_Tables\Server
PS_F_AR_AGING	SEQ_J_Fact_PS_F_AR_AGING_E	FMS_E\Receivables_Mart\AR\OWS_To_MDW\Facts\Base\Load_Tables\Server
	SEQ_J_Fact_PS_F_AP_AGING_E	FMS_E\Payables_Mart\AP\OWS_To_MDW\Facts\Base\Load_Tables\Server

Cross-warehouse dependencies for PS_D_SUPPLIER

   Similarly, the lookup table PS_F_AP_AGING is populated from the fact job J_Fact_PS_F_AP_AGING placed in the category FMS_E\Payables_Mart\AP\OWS_To_MDW\Facts\Base\Load_Tables\Server. This fact job belongs to a different mart as indicated by the different color.



H	J	K	L	R
			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_E\OWS_To_MDW\Base\Load_Tabl
	J_Dim_PS_D_SUPPLIER		SEQ_J_Dim_PS_D_SUPPLIER	Global_Dimensions_E\OWS_To_MDW\Base\Load_Tabl
	J_Fact_PS_F_AR_AGING_E		SEQ_J_Fact_PS_F_AR_AGING_E	FMS_E\Receivables_Mart\AR\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\L

Cross-warehouse dependencies for PS_F_AP_AGING

Identifying the List of Jobs to be Run for a Data Mart

You can use the information in the spreadsheet to identify the list of jobs that need to be run for a specific data mart. These include common jobs that are required for every data mart, which we refer to as prerequisite jobs, as well as jobs specific to the particular data mart.

If you prefer, you can create your own master sequencers based on the information provided in this section.

Alternatively, you can generate the list of jobs by using the Dynamic Lineage Generator tool. For more information, see "Generating Lineage Information for a Job".

Note. All the server jobs relating to Hash files that are present within the Load_Hash_Files category need to be run first before running other Sequence jobs within the Load_Tables category since these hash files are being used in other server jobs.

Prerequisite Jobs

The prerequisite jobs include setup jobs, staging jobs, and dimension jobs.

The following sets of jobs need to be run for *every* mart, in the order that they are listed in the worksheets:

1. Run these setup jobs in the Setup worksheet:

    a. All jobs within the Setup_E\OWS\<*Warehouse*> category.

    (For example all jobs within the Setup_E\OWS\FSCM category for the FMS warehouse and all jobs within the Setup_E\OWS\CS category for the 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 *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Importing Source Business Units into EPM to Create Warehouse Business Units."

    c. All jobs within the Shared_Lookups\DimensionMapper_Lookups category.

    d. All jobs within the Shared_Lookups\Control_Tables category.

    e. All jobs within the Shared_Lookups\System_Lookups category.

    f. All jobs within the Shared_Lookups\Language_Lookups category.

    g. All jobs within the Setup_E\OWE category (this step does not apply to the Campus Solutions warehouse).

    h. If you are implementing currency conversion, then run the jobs listed in the Utils worksheet.

2. Run the staging jobs listed in the OWS Sequencer column (column N) in the following worksheets:

   a. Com Dims.

   b. Global Dims.

   c. Local Dims.

   d. *<Data Mart>*, where *<Data Mart>* is the name of the data mart, for example AP, AR, Campus Community, Student Financials .

3. Run the Common Dimension Jobs listed in the Com Dims worksheet.

4. Run the Global Dimensions jobs listed in the Global Dims worksheet. (These jobs are required for running the FMS warehouse jobs.)

5. Run the Local Dimension Jobs placed in the Local Dims worksheet.

Data Mart Specific Jobs

Run all the Server jobs listed in column B of the worksheet for the specific data mart, to populate the corresponding Dimension and Fact tables for that mart.

Note. Do not run the jobs that are listed within the Reusable Jobs category. These jobs are not used to load target tables. They are automatically triggered by various Sequence jobs.

Generating Lineage Information for a Job

The Dynamic_Lineage_Generator worksheet contains a macro that generates a list of all the dependent jobs that are required for any ETL job. This will easily help you identify all the list of jobs to be run for a specific fact or dimension job.

To use the Dynamic Lineage Generator:

1. Access the Dynamic_Lineage_Generator worksheet.

2. Enter the job name in cell B1.

3. Click the Get Job Lineage button.

   The macro retrieves the lineage required for running this fact job from the setup, staging, and the dimension jobs and displays it in the cells below. The macro also copies the entire list of dependent jobs to the JobOrder worksheet, so you can identify the complete list to be run in sequence.

You must run the following prerequisite setup jobs before you run the jobs listed in the JobOrder worksheet:

- Setup_E\OWS\<*Warehouse Name*> Job Sequencer.

  For example Setup_E\OWS\FSCM Job Sequencer or Setup_E\OWS\CS Job Sequencer.

- Setup_E\Dimension mapper Job Sequencer.

- Run the Business Unit Wizard to populate the Dimension mapper tables.

  See *PeopleSoft Enterprise Performance Management Fundamentals 9.1 PeopleBook*, "Importing Source Business Units into EPM to Create Warehouse Business Units."

- Shared_Lookups\DimensionMapper_Lookups
- Shared_Lookups\Control_Tables
- Shared_Lookups\System_Lookups
- Shared_Lookups\Language_Lookups
- Setup_E\OWE Job Sequencer (this step does not apply to the Campus Solutions warehouse).

After you run the prerequisite setup jobs, then run the jobs listed in the JobOrder worksheet.

Index

B

C

D

E

F

H

I

L

M

O

P

R

S

T