IBM InfoSphere DataStage Data Flow and Job Design

IBM InfoSphere DataStage overview

Retail industry scenario

IBM Information Server setups

Nagraj Alur
Celso Takahashi
Sachiko Toratani
Denis Vasconcelos

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM InfoSphere DataStage Data Flow and Job Design

July 2008

SG24-7576-00

Note: Before using this information and the product it supports, read the information in “Notices” on page xxxi.

First Edition (July 2008)

This edition applies to Version 8, Release 1, Modification 0 of IBM Information Server (5724-Q36).

Contents

Figures

Tables

Examples

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®
DataStage®
DB2®
Ernie®
IBM®
IMS™
Informix®
InfoSphere™
LoadLeveler®
MVS™
Orchestrate®
Rational®
Redbooks®
Redbooks (logo) ®
System p™
System z™
Tivoli®
WebSphere®
z/OS®

The following terms are trademarks of other companies:

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

AMD, the AMD Arrow logo, and combinations thereof, are trademarks of Advanced Micro Devices, Inc.

EJB, J2EE, Java, JDBC, Solaris, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Expression, Microsoft, SQL Server, Windows Server, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Itanium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

This IBM® Redbooks® publication documents the procedures for implementing IBM InfoSphere™ DataStage® and related technologies using a typical retail industry scenario.

It is aimed at IT architects, Information Management specialists, and Information Integration specialists responsible for developing IBM InfoSphere DataStage on a Red Hat Enterprise Linux® 4.0 platform.

The book offers a step-by-step approach to implementing IBM InfoSphere DataStage on Red Hat Enterprise Linux 4.0 platforms accessing information stored on IBM z/OS® and IBM AIX® platforms.

This book is organized as follows:

- Chapter 1, “IBM InfoSphere DataStage overview” on page 1 provides an overview of IBM Information Server architecture and main components, IBM InfoSphere DataStage within the IBM Information Server architecture, and IBM InfoSphere DataStage’s main functions.
- Chapter 2, “IBM InfoSphere DataStage stages” on page 35 provides an overview of some of the commonly used stages in IBM InfoSphere DataStage.
- Chapter 3, “Retail industry scenario” on page 139 describes a step-by-step approach to implementing a “real world” retail industry scenario involving a typical star-schema data warehousing flow using IBM InfoSphere DataStage. Included in the flow are the Complex Flat File, Distributed Transaction Stage, and Slowly Changing Dimension stage.
- Appendix A, “IBM Information Server setups” on page 563 describes the setups of various components required to implement the retail industry scenario.
- Appendix B, “Code and scripts used in the retail industry scenario” on page 597 documents some of the code and scripts used in the retail industry scenario.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Nagraj Alur is a Project Leader with the IBM ITSO, San Jose Center. He holds a Master's degree in Computer Science from the Indian Institute of Technology (IIT), Mumbai, India. He has more than 33 years of experience in database management systems (DBMSs), and has been a programmer, systems analyst, project leader, independent consultant, and researcher. His areas of expertise include DBMSs, data warehousing, distributed systems management, database performance, information integration, and client/server and Internet computing. He has written extensively on these subjects and has taught classes and presented at conferences all around the world. Before joining the ITSO in November 2001, he was on a two-year assignment from the Software Group to the IBM Almaden Research Center, where he worked on Data Links solutions and an eSourcing prototype.

Celso Takahashi is a Technical Sales Specialist in IBM Brazil. He has 12 years experience in database management systems such as Informix® and Oracle®. He has done Proof of Concept (POC) projects involving IBM Information Server for customers in Brazil. His areas of expertise include DataStage, QualityStage, Information Analyzer, DataMirror, Business Glossary and Metadata Workbench. Celso has a Bachelor's degree in Computer Science and an MBA degree in Project Management.

Sachiko Toratani is an IT Specialist providing technical support on IBM Information Platform products to customers in Japan. She has more than eight years experience in database management systems (DBMSs), and application development in government related systems. Her areas of expertise include Information Integration and DBMSs, with extensive skills in IBM Information Server, IBM InfoSphere DataStage, and DB2® for Linux®, UNIX®, and Windows®. She is IBM Certified in Database Administrator DB2 UDB for Linux, UNIX, and Windows.

Denis Vasconcelos is a Data Specialist with IBM Brazil. He had over five years experience with several non-IBM data management systems before joining IBM in 2006. His areas of expertise include database administration, data modeling, heterogeneous database migration, and project management. Denis has a Bachelor's degree in Computer Science and a post-graduate degree in Project Management.

Thanks to the following people for their contributions to this project:

Aarti Borkar
Brian Caufield
Atul Chadha
Gary Faircloth
Jennifer Fell
Sreejith Kurup
Tamara Khaleel
Gaurav Rawal
Paul Stanley
Gregg Upton
IBM Silicon Valley Laboratory, San Jose

Paul Christensen
Tony Curcio
Ernie® Ostic
Barry Scott Rosen
Emily White
IBM USA

Carmen Ruppach
IBM Germany

Deepak Naik
IBM India

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

IBM InfoSphere DataStage overview

In this chapter we provide an overview of IBM Information Server architecture and main components, IBM InfoSphere DataStage within the IBM Information Server architecture, IBM InfoSphere DataStage's main functions, and best practices.

The topics covered are:

- IBM Information Server architecture
- IBM InfoSphere DataStage within the IBM Information Server architecture
- IBM InfoSphere DataStage main functions
- Best practices overview

1.1 Introduction

Over the years, most organizations have made significant investments in enterprise resource planning, customer relationship management, and supply chain management packages in addition to their home grown applications. This has resulted in larger amounts of data being captured about their businesses. To turn all this data into consistent, timely, and accurate information for decision-making requires an effective means of integrating information. Statutory compliance requirements such as Basel II and Sarbanes-Oxley place additional demands for consistent, complete, and trustworthy information.

IBM Information Server addresses these critical information integration requirements of consistent, complete, and trustworthy information with a comprehensive, unified foundation for enterprise information architectures. IBM Information Server is capable of scaling to meet any information volume requirement so that companies can deliver business results faster and with higher quality results for all their critical initiatives such as business intelligence, master data management, infrastructure rationalization, business transformation, and risk and compliance.

IBM Information Server combines the technologies of key information integration functions within the IBM Information Platform & Solutions portfolio into a single unified platform that enables companies to understand, cleanse, transform, and deliver trustworthy and context-rich information as shown in Figure 1-1.

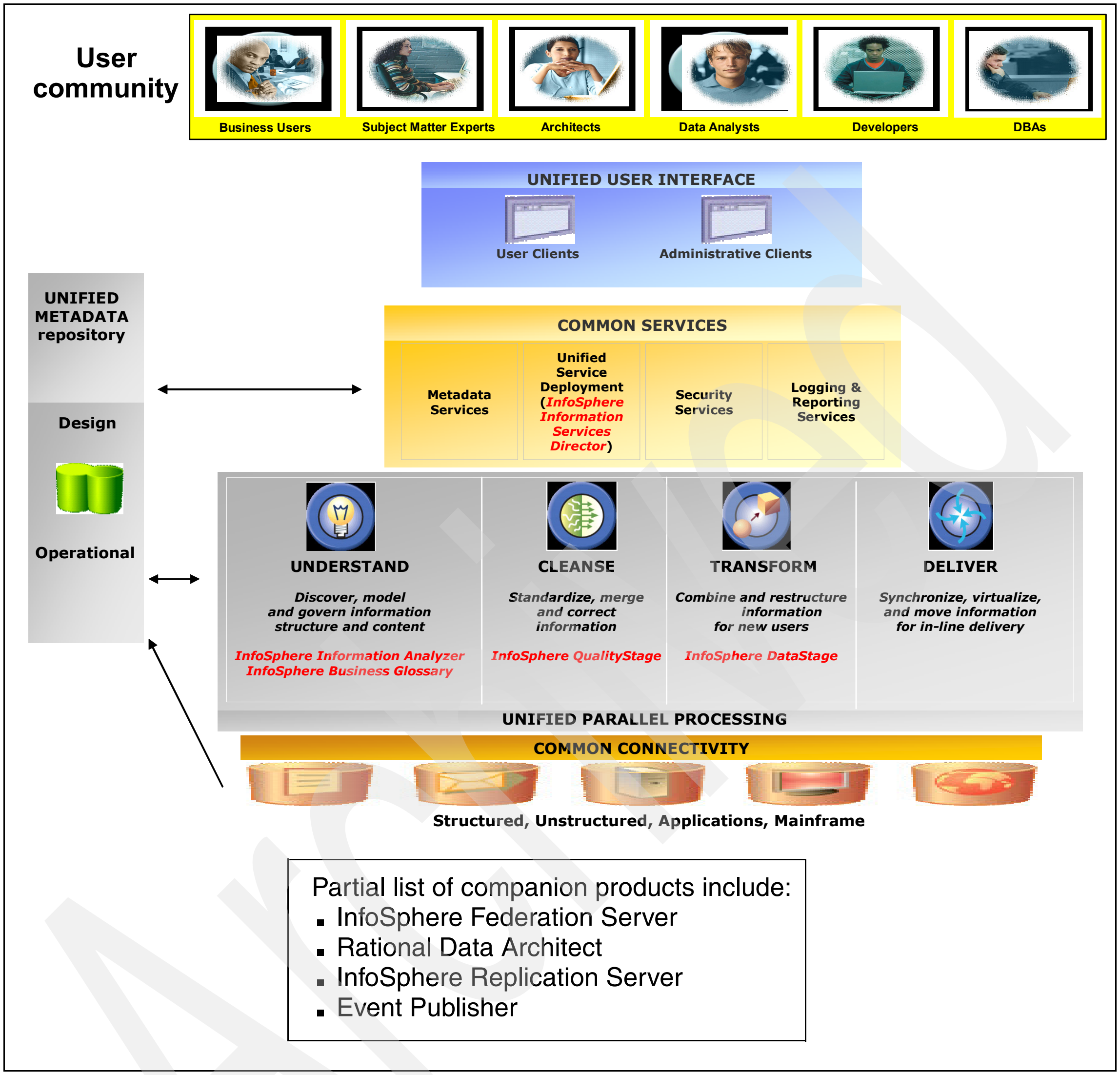


Figure 1-1 IBM Information Server architecture

IBM Information Server includes the following product modules:

- IBM InfoSphere DataStage

  Enables organizations to design data flows that extract information from multiple source systems, transform it in ways that make it more valuable, and then deliver it to one or more target databases or applications.

  This is the focus of this Redbooks publication.

- IBM InfoSphere QualityStage

  Designed to help organizations understand and improve the overall quality of their data assets, IBM InfoSphere QualityStage provides advanced features to help investigate, repair, consolidate, and validate heterogeneous data within an integration workflow.

  This was the focus of a previous Redbooks publication, *IBM WebSphere QualityStage Methodologies, Standardization, and Matching,* SG24-7546.

- IBM InfoSphere Information Services Director

  IBM Information Server provides a unified mechanism for publishing and managing shared Service Oriented Architecture (SOA) services across data quality, data transformation, and federation functions, allowing information specialists to easily deploy services for any information integration task and consistently manage them. This enables developers to take data integration logic built using IBM Information Server and publish it as an "always on" service — in minutes. The common services also include the metadata services, which provide standard service-oriented access and analysis of metadata across the platform.

  This was the focus of a previous Redbooks publication, *SOA Solutions Using IBM Information Server,* SG24-7402.

- IBM InfoSphere Information Analyzer

  IBM InfoSphere Information Analyzer profiles and analyzes data so that you can deliver trusted information to your users. It can automatically scan samples of your data to determine their quality and structure. This analysis aids you in understanding the inputs to your integration process, ranging from individual fields to high-level data entities. Information analysis also enables you to correct problems with structure or validity before they affect your project. While analysis of source data is a critical first step in any integration project, you must continually monitor the quality of the data. IBM InfoSphere Information Analyzer enables you to treat profiling and analysis as an ongoing process and create business metrics that you can run and track over time.

  This was the focus of a previous Redbooks publication, *IBM WebSphere Information Analyzer & Data Quality Assessment,* SG24-7508.

- IBM Information Server FastTrack

  Simplifies and streamlines communication between the business analyst and developer by capturing business requirements and automatically translating into IBM InfoSphere DataStage ETL jobs.

- IBM InfoSphere Business Glossary

  IBM Information Server provides a Web-based tool that enables business analysts and subject-matter experts to create, manage, and share a common enterprise vocabulary and classification system. IBM InfoSphere Business Glossary enables users to link business terms to more technical artifacts managed by the metadata repository. The metadata repository also enables sharing of the business terms by IBM Rational® Data Architect and IBM InfoSphere Information Analyzer, creating a common set of semantic tags for reuse by data modelers, data analysts, business analysts, and end users.

A number of companion products support IBM Information Server, such as InfoSphere Federation Server, Rational Data Architect, InfoSphere Replication Server, and Event Publisher.

> **Note:** For an overview of IBM Information Server, refer to the Web site
> http://www.ibm.com/software/data/integration/info_server/

In the following sections, we describe IBM Information Server's architecture, IBM InfoSphere DataStage within the IBM Information Server architecture, and IBM InfoSphere DataStage's main functions. We also provided an overview of best practices.

> **Attention:** This Redbooks publication does not cover all the functions and features of IBM InfoSphere DataStage. Refer to the resources described in "Related publications" on page 607 for complete details on IBM InfoSphere DataStage.

1.2 IBM Information Server architecture

IBM Information Server is a client-server architecture made up of client-based design, administration, and operation tools that access a set of server-based data integration capabilities through a common services layer as shown here in Figure 1-2. This is a slightly different and more detailed view of the same information shown previously in Figure 1-1 on page 3.

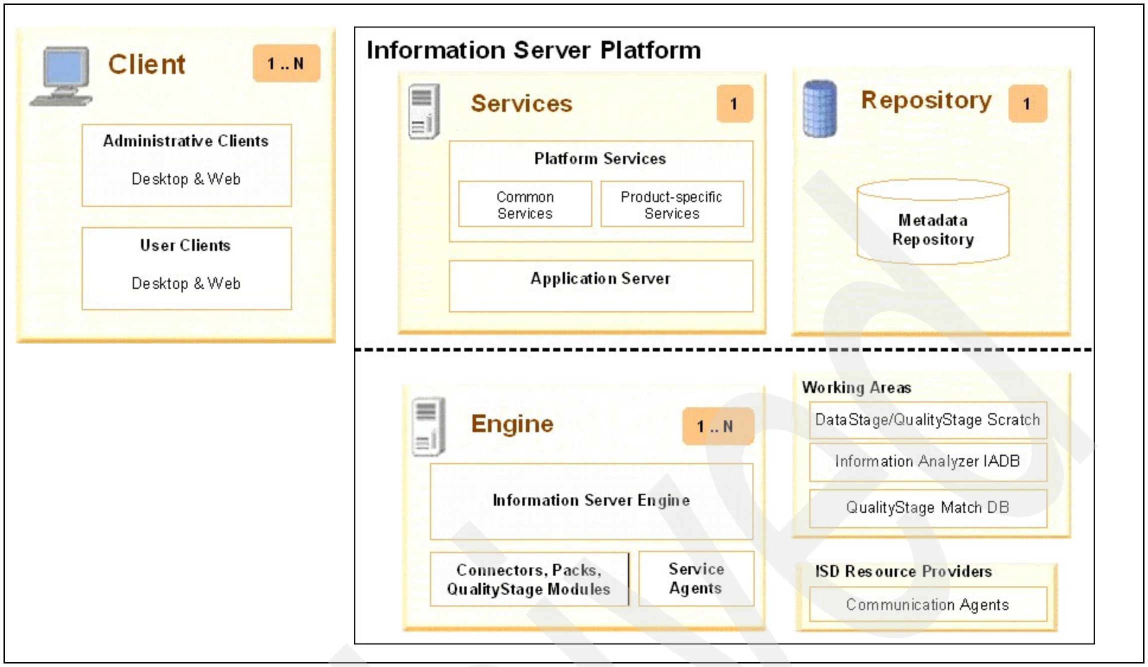


Figure 1-2 IBM Information Server client/server architecture perspective

In this section, we briefly discuss the following topics:

- Component overview
- Topologies supported

1.2.1 Component overview

The main components shown in Figure 1-2 on page 6 are briefly described here.

Client tier

IBM Information Server provides a number of client interfaces, optimized to different user roles within an organization. The clients tier includes IBM InfoSphere DataStage and QualityStage clients (Administrator, Designer, and Director), IBM Information Server console, and IBM Information Server Web console.

There are two broad categories of clients — Administrative clients and User clients. Both these types of clients have desktop and Web based interfaces.

- Administrative clients

  These clients allow you to manage the areas of security, licensing, logging, and scheduling.

  – Administration tasks are performed in the IBM Information Server Web console. The IBM Information Server Web console is a browser-based interface for administrative activities such as managing security and creating views of scheduled tasks.

  – For IBM InfoSphere DataStage and IBM InfoSphere QualityStage project administration, you use the IBM InfoSphere DataStage Administrator client. It administers IBM InfoSphere DataStage projects and conducts housekeeping on the server. It is used to specify general server defaults, add and delete projects, and to set project properties. User and group privileges are also set using the Administrator client.

- User clients

  These clients help perform client tasks such as creating, managing, and designing jobs, as well as validating, running, scheduling. and monitoring jobs. The IBM Information Server console is a rich client-based interface for activities such as profiling data and developing service-oriented applications.

  – The IBM InfoSphere DataStage and QualityStage Designer helps you create, manage, and design jobs. You can also use the Designer client to define tables and access metadata services.

    The Designer client allows you to move DataStage and QualityStage objects between projects on the same Information Server engine, or on different Information Server engines. You can also use the Information Server Manager client to move objects from one domain to another. The Information Server Manager supports the model of having separate systems for the developing, testing and running of DataStage and QualityStage jobs. It facilitates the model by providing secure and managed methods of moving objects between the different systems.

  – The IBM InfoSphere DataStage and QualityStage Director client is the client component that validates, runs, schedules, and monitors jobs on the IBM InfoSphere DataStage Server.

> **Note:** Clients are supported on 32-bit Microsoft® Windows XP Pro, Vista, and Server 2003.

Server tiers

The server tiers of the Information Server Platform that includes the Services, Engine, Repository, Working Areas, and Information Services Director Resource Providers as follows:

- Services tier

  IBM Information Server is built entirely on a set of shared services that centralize core tasks across the platform. Shared services allow these tasks to be managed and controlled in one place, regardless of which suite component is being used.

  The Services Tier includes both common and product-specific services:

  – Common services are used across the Information Server suite for tasks such as security, user administration, logging, reporting, metadata, and execution.

  – Product-specific services provide tasks for specific products within the Information Server suite. For example, IBM InfoSphere Information Analyzer calls a column analyzer service (a product-specific service) that was created for enterprise data analysis. The shared service environment allows integration across IBM Information Server because they are deployed using common SOA standards.

  IBM Information Server products can access three general categories of service:

  – Design

    Design services help developers create function-specific services that can also be shared.

  – Execution

    Execution services include logging, scheduling, monitoring, reporting, security, and Web framework.

  – Metadata

    Using metadata services, metadata is shared “live” across tools so that changes made in one IBM Information Server component are instantly visible across all of the suite components. Metadata services are tightly integrated with the common repository. You can also exchange metadata with external tools by using metadata services.

  The common services layer is deployed on the J2EE™-compliant application server IBM WebSphere® Application Server, which is included with IBM Information Server.

  Note: An Application Server is a high performance transaction engine that helps you build, run, integrate, and manage dynamic Web based applications typically involving HTTP protocol.

- Repository tier

  The shared repository is used to store all IBM Information Server product module objects[1] (including IBM InfoSphere DataStage objects), and is shared with other applications in the suite. Clients can access metadata and results of data analysis from the respective service layers.

  > **Note:** The repository supports DB2 for LUW 9, Oracle10g R2, or SQLServer 2005 as the underlying database.

- Engine tier

  This is the parallel runtime engine that executes the IBM Information Server tasks. It comprises the Information Server engine, Service Agents, and Connectors and Packaged Application Connectivity Kits (PACKS[2]).

  – The IBM Information Server engine consists of the products that you install, such as IBM InfoSphere DataStage and IBM InfoSphere QualityStage. It runs jobs to extract, transform, load, and standardize data. The engine runs DataStage and QualityStage jobs. It also executes the parallel jobs for Information Analyzer tasks.

  – Service Agents are Java™ processes that run in the background on each computer that hosts IBM InfoSphere DataStage.They provide the communication between the Services and Engine tiers of Information Server.

  – Connectors and PACKS

    IBM Information Server connects to a variety of information sources whether they are structured, unstructured, on the mainframe, or applications. Metadata-driven connectivity is shared across the suite components, and connection objects are reusable across functions.

    Connectors provide design-time importing of metadata, data browsing and sampling, run-time dynamic metadata access, error handling, and high functionality and high performance run-time data access.

    Prebuilt interfaces for packaged applications called PACKS provide adapters to SAP®, Siebel®, Oracle, and others, enabling integration with enterprise applications and associated reporting and analytical systems.

[1] This includes jobs and table definitions, as well as operational metadata such as job start and stop times. The repository is also used to store Information Server configuration settings, such as user group assignments and roles.

[2] PACKs provide an application-specific view of data, using the packaged application vendor's APIs for connectivity and business metadata.

- Working areas

  These are temporary storage areas used by the suite components.

- Information Services Director (ISD) Resource Providers

  Information service providers are the (data) sources of operations for your services. Using IBM InfoSphere Information Services Director, you can create services from five sources — IBM InfoSphere DataStage and QualityStage, IBM DB2 for LUW, IBM InfoSphere Federation Server, IBM InfoSphere Classic Federation Server for z/OS, and Oracle Database Server.

> **Note:** The Information Server Platform 8.0.1 release supports the following operating systems:
>
> - IBM AIX 5.2, 5.3
> - HP-UX Itanium™ 11i v2
> - HP-UX PA-RISC 11i v2
> - Sun™ Solaris™ 9, 10
> - Red Hat Enterprise Server Linux 4 (Intel®, AMD™)
> - SUSE Linux Enterprise Server Linux 10 (Intel, AMD, System p™, System z™)
> - Microsoft Windows Server® 2003 (32-bit)

1.2.2 Topologies supported

IBM Information Server is built on a highly scalable parallel software architecture that delivers high levels of throughput and performance. For maximum scalability, integration software must do more than run on Symmetric Multiprocessing (SMP) and Massively Parallel Processing (MPP) computer systems. If the data integration platform does not saturate all of the nodes of the MPP box or system in the cluster or Grid, scalability cannot be maximized. The IBM Information Server components fully exploit SMP, clustered, Grid, and MPP environments to optimize the use of all available hardware resources.

IBM Information Server supports multiple topologies to satisfy a variety of your data integration and hardware business requirements, as follows:

- Two-tier
- Three-tier
- Cluster
- Grid

For all topologies, you can add clients and engines (for scalability) on additional computers.

To select a topology, you must consider your performance needs by reviewing the capacity requirements for the topology elements — the server, disk, network, data sources, targets, data volumes, processing requirements, and any service-level agreements.

Each of these topologies is briefly described here.

Tip: We recommend that you use the same topology for your test and production environments to minimize issues when a job is deployed into production.

Note: On a Microsoft Windows platform, the clients, engine, application server, and metadata repository can be collocated on the same machine. This topology (not shown here) is only suitable for demonstrations, as an educational or proof-of-concept platform.

Two-tier

The engine, application server, and metadata repository are all on the same computer system, while the clients are on a different machine as shown in Figure 1-3.

High availability and failover are simpler to manage with two computers because all the servers fail over at the same time.

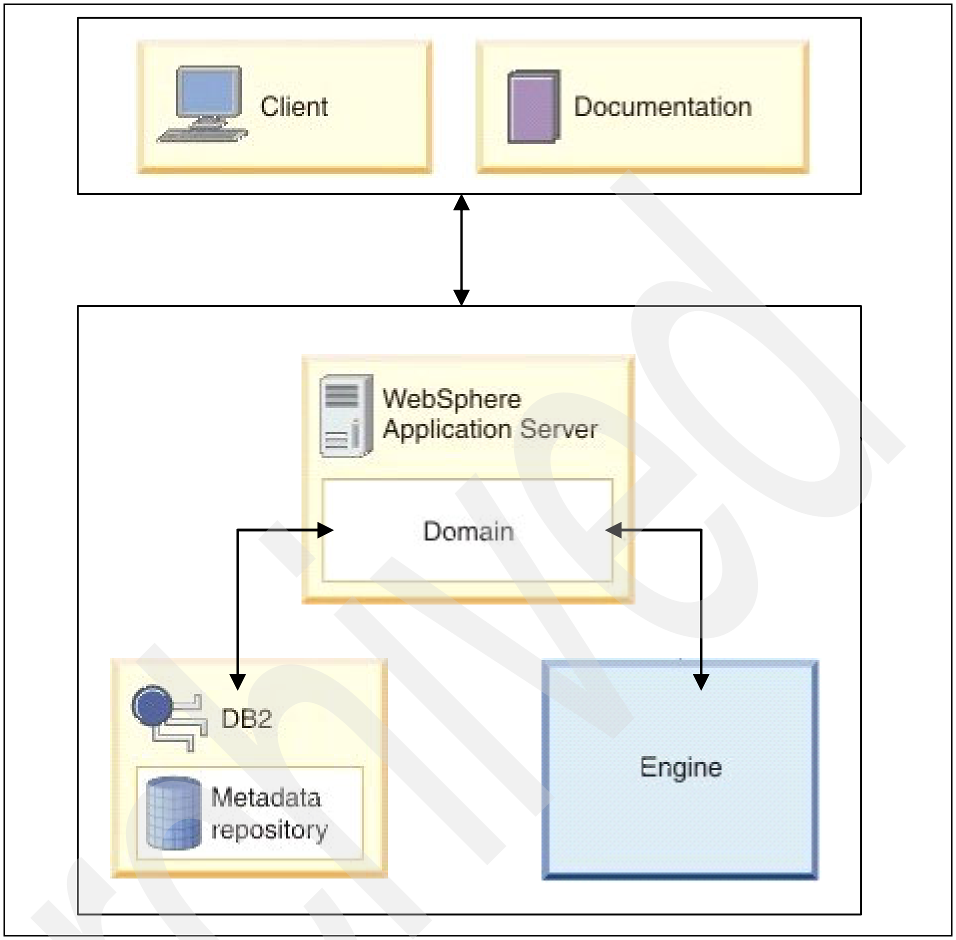


Figure 1-3 Two-tier

Three tier

The engine is on one machine, the application server and metadata repository are co-located on another machine, while the clients are on a third machine as shown in Figure 1-4.

Failover configuration is more complex because of the increased number of failover scenarios that are required by three or more computers.

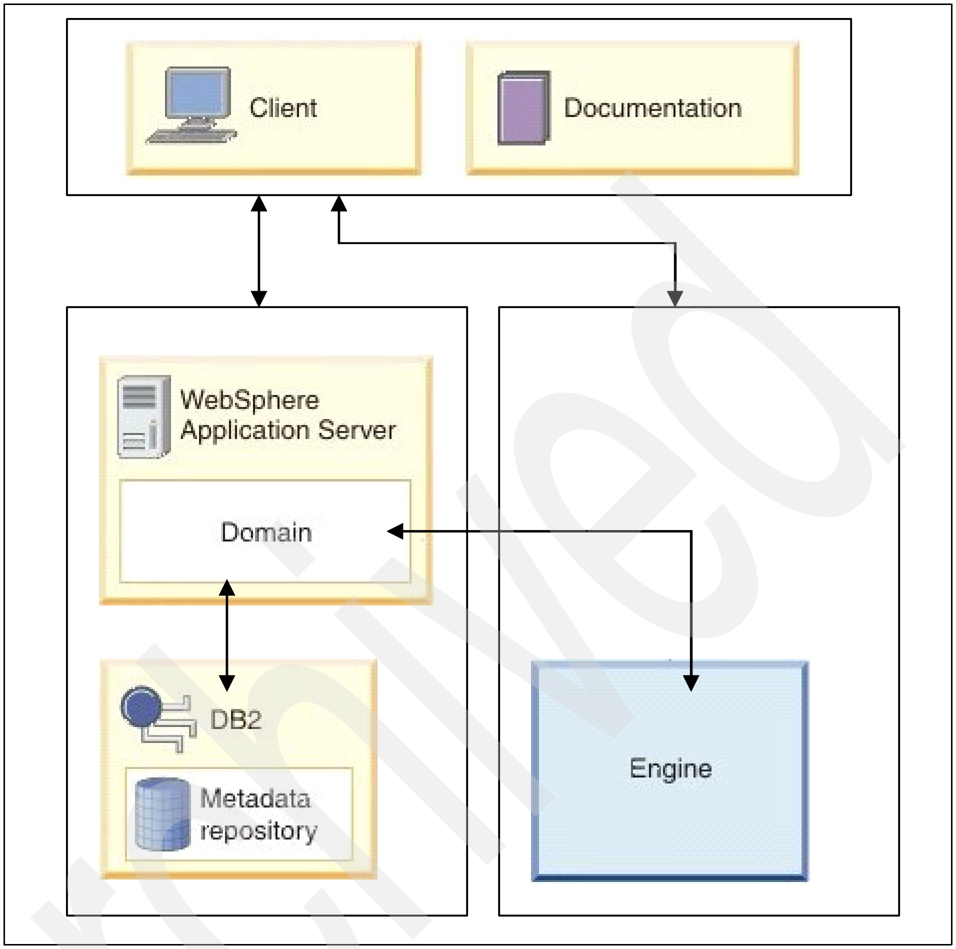


Figure 1-4 Three tier topology

Cluster

This is a slight variation of the three-tier topology with the engine duplicated over multiple computers as shown in Figure 1-5.

In a cluster environment, a single parallel job execution can span multiple computers, each with its own engine.

The processing of a job on the multiple machines is driven by a configuration file associated with the job. The configuration file specifies the machines to be used by the job.

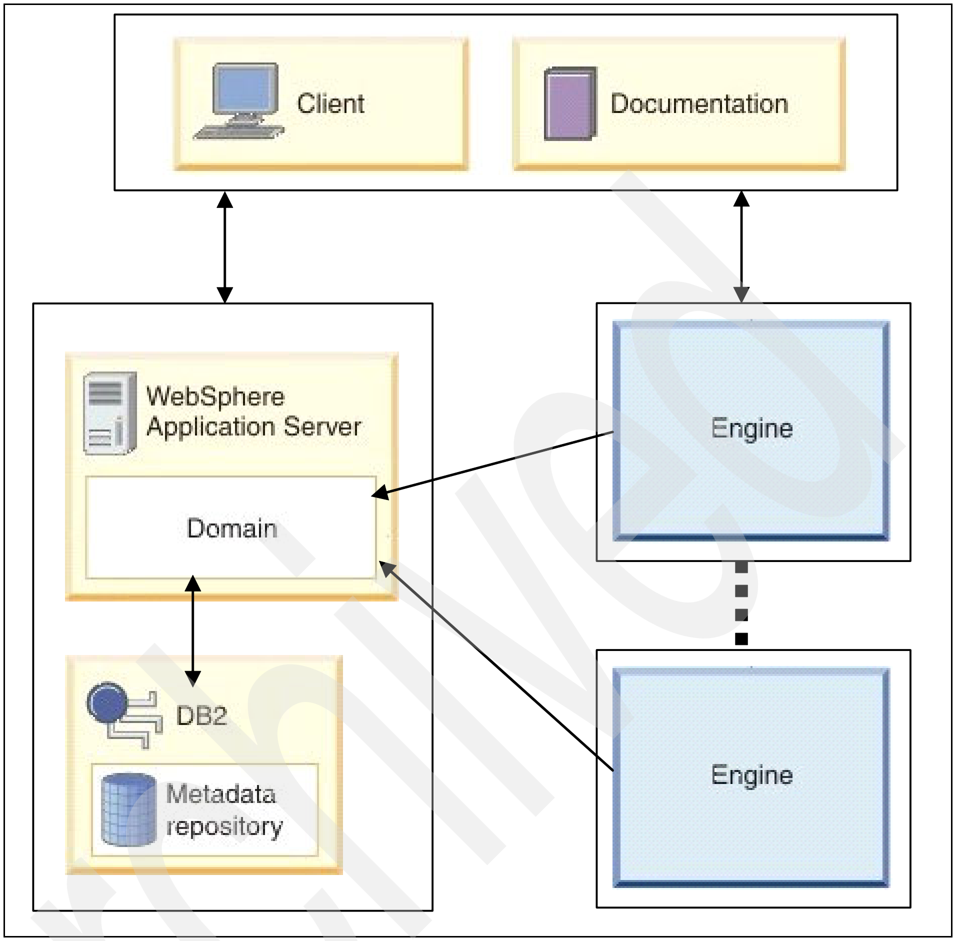


Figure 1-5 Cluster and Grid

Grid

With hardware computing power a commodity, Grid computing allows you to apply more processing power to a task than was previously possible. Grid computing uses all of the low-cost computing resources, processors, and memory that are available on the network to create a single system image.

Grid topology is very similar to that of a cluster (Figure 1-5 on page 14) with engines distributed over multiple machines. As in the case of a cluster environment, a single parallel job execution can span multiple computers, each with its own engine.

The key difference with cluster computing is that in a Grid environment, the machines over which a job executes are dynamically determined (through the generation of a dynamic configuration file) using an integrated resource manager such as IBM Tivoli® Workload Scheduler LoadLeveler®.

The parallel processing architecture of IBM Information Server leverages the computing power of Grid environments and greatly simplifies the development of scalable integration systems that run in parallel for Grid environments.

1.3 IBM InfoSphere DataStage within the IBM Information Server architecture

IBM InfoSphere DataStage facilitates data integration in both high-volume batch and services-oriented deployment scenarios required by enterprise system architectures. As part of the integrated IBM Information Server platform, it is supported by a broad range of shared services and benefits from the reuse of several suite components.

IBM InfoSphere DataStage and IBM InfoSphere QualityStage share the same infrastructure for importing and exporting data, designing, deploying, and running jobs, and reporting. The developer uses the same design canvas to specify the flow of data from preparation to transformation and delivery.

Multiple discrete services give IBM InfoSphere DataStage the flexibility to match increasingly varied customer environments and tiered architectures. Figure 1-1 on page 3 shows how IBM InfoSphere DataStage Designer (labeled “User Clients”) interacts with other elements of the IBM Information Server platform to deliver enterprise data analysis services.

In this section, we briefly describe the following topics:

- Shared components
- Runtime architecture

1.3.1 Shared components

With reference to Figure 1-1 on page 3, the following suite components are shared between IBM InfoSphere DataStage and IBM Information Server:

- **Unified user interface**

  The following client applications comprise the IBM InfoSphere DataStage user interface:

  – IBM InfoSphere DataStage and QualityStage Designer

    A graphical design interface is used to create InfoSphere DataStage applications (known as jobs). Because transformation is an integral part of data quality, the InfoSphere DataStage and QualityStage Designer is the design interface for both InfoSphere DataStage and InfoSphere QualityStage.

    Each job specifies the data sources, the required transformations, and the destination of the data. Jobs are compiled to create parallel job flows and reusable components that are scheduled by the InfoSphere DataStage and QualityStage Director and run in parallel by the Information Server engine. The Designer client manages design metadata in the repository, while compiled execution data is deployed on the Information Server Engine tier.

  – IBM InfoSphere DataStage and QualityStage Director

    A graphical user interface that is used to validate, schedule, run, and monitor InfoSphere DataStage job sequences. The Director client displays job run-time information including job status and detailed job logs. This client can also be used to establish schedules for job execution.

  – InfoSphere DataStage and InfoSphere QualityStage Administrator

    A graphical user interface that is used for administration tasks such as:

    - Setting up DataStage and Information Server Engine users
    - Creating, deleting, and customizing projects
    - Setting up criteria for purging runtime log records.

- **Common services**

  As part of the IBM Information Server Suite, DataStage leverages the common services as well as DataStage-specific services.

  The common services provides flexible, configurable interconnections among the many parts of the architecture as follows:

  – Metadata services such as impact analysis and search

  – Execution services that support all InfoSphere DataStage functions

  – Design services that support development and maintenance of InfoSphere DataStage tasks

- **Common repository**

  The common repository holds three types of metadata that are required to support IBM InfoSphere DataStage, as follows:

  – Project metadata

    All the project-level metadata components including IBM InfoSphere DataStage jobs, table definitions, built-in stages, reusable subcomponents, and routines are organized into folders.

  – Operational metadata

    The repository holds metadata that describes the operational history of integration process runs, success or failure of jobs, parameters that were used, and the time and date of these events.

  – Design metadata

    The repository holds design time metadata that is created by the IBM InfoSphere DataStage and QualityStage Designer and IBM InfoSphere Information Analyzer.

- **Common parallel processing engine**

  The engine runs executable jobs that extract, transform, and load data in a wide variety of settings. The engine uses parallelism and pipelining to handle high volumes of work more quickly and to scale a single job across the boundaries of a single server in cluster or Grid topologies.

- **Common connectors**

  The connectors provide connectivity to a large number of external resources and access to the common repository from the processing engine. Any data source that is supported by IBM Information Server can be used as input to or output from an IBM InfoSphere DataStage job.

1.3.2 Runtime architecture

This section briefly describes the generation of the OSH (**O**rchestrate® **SH**ell Script) script, and the execution flow of IBM InfoSphere DataStage using the Information Server engine.

OSH script

The IBM InfoSphere DataStage and QualityStage Designer client creates IBM InfoSphere DataStage jobs that are compiled into parallel job flows, and reusable components that execute on the parallel Information Server engine. It allows you to use familiar graphical point-and-click techniques to develop job flows for extracting, cleansing, transforming, integrating, and loading data into target files, target systems, or packaged applications.

The Designer generates all the code. It generates the OSH (**O**rchestrate **SH**ell Script) and C++ code for any Transformer stages used.

Briefly, the Designer performs the following tasks:

- Validates link requirements, mandatory stage options, transformer logic, etc.
- Generates OSH representation of data flows and stages (representations of framework “operators”).
- Generates transform code for each Transformer stage which is then compiled into C++ and then to corresponding native operators.
- Reusable BuildOp stages can be compiled using the Designer GUI or from the command line.

Here is a brief primer on the OSH:

- Comment blocks introduce each operator, the order of which is determined by the order stages were added to the canvas.
- OSH uses the familiar syntax of the UNIX shell. such as Operator name, schema, operator options (“-name value” format), input (indicated by n< where n is the input#), and output (indicated by the n> where n is the output #).
- For every operator, input and/or output data sets are numbered sequentially starting from zero.
- Virtual data sets (in memory native representation of data links) are generated to connect operators.

> **Note:** The actual execution order of operators is dictated by input/output designators, and not by their placement on the diagram. The data sets connect the OSH operators. These are “virtual data sets”, that is, in memory data flows. Link names are used in data set names — it is therefore good practice to give the links meaningful names.

Framework (Information Server Engine) terms and DataStage terms have equivalency. The GUI frequently uses terms from both paradigms. Runtime messages use framework terminology because the framework engine is where execution occurs. The following list shows the equivalency between framework and DataStage terms:

- Schema corresponds to table definition
- Property corresponds to format
- Type corresponds to SQL type and length
- Virtual data set corresponds to link
- Record/field corresponds to row/column
- Operator corresponds to stage

- Step, flow, OSH command correspond to a job
- Framework corresponds to Information Server Engine

Execution flow

When you execute a job, the generated OSH and contents of the configuration file ($APT_CONFIG_FILE) is used to compose a “score”. This is similar to a SQL query optimization plan.

At runtime, IBM InfoSphere DataStage identifies the degree of parallelism and node assignments for each operator, and inserts sorts and partitioners as needed to ensure correct results. It also defines the connection topology (virtual data sets/links) between adjacent operators/stages, and inserts buffer operators to prevent deadlocks (for example, in fork-joins). It also defines the number of actual OS processes. Multiple operators/stages are combined within a single OS process as appropriate, to improve performance and optimize resource requirements.

The job score is used to fork processes with communication interconnects for data, message and control[3]. Processing begins after the job score and processes are created. Job processing ends when either the last row of data is processed by the final operator, a fatal error is encountered by any operator, or the job is halted by DataStage Job Control or human intervention such as DataStage Director STOP.

> **Note:** You can direct the score to a job log by setting $APT_DUMP_SCORE. To identify the Score dump, look for “main program: This step....”.

Job scores are divided into two sections — data sets (partitioning and collecting) and operators (node/operator mapping). Both sections identify sequential or parallel processing.

The execution (orchestra) manages control and message flow across processes and consists of the conductor node and one or more processing nodes as shown in Figure 1-6. Actual data flows from player to player — the conductor and section leader are only used to control process execution through control and message channels.

- Conductor is the initial framework process. It creates the Section Leader (SL) processes (one per node), consolidates messages to the DataStage log, and manages orderly shutdown. The Conductor node has the start-up process. The Conductor also communicates with the players.

[3] Set $APT_STARTUP_STATUS to show each step of the job startup, and $APT_PM_SHOW_PIDS to show process IDs in the DataStage log.

- Section Leader is a process that forks player processes (one per stage) and manages up/down communications. SLs communicate between the conductor and player processes only. For a given parallel configuration file, one section leader will be started for each logical node.
- Players are the actual processes associated with the stages. It sends stderr and stdout to the SL, establishes connections to other players for data flow, and cleans up on completion. Each player has to be able to communicate with every other player. There are separate communication channels (pathways) for control, errors, messages and data. The data channel does not go through the section leader/conductor as this would limit scalability. Data flows directly from upstream operator to downstream operator.

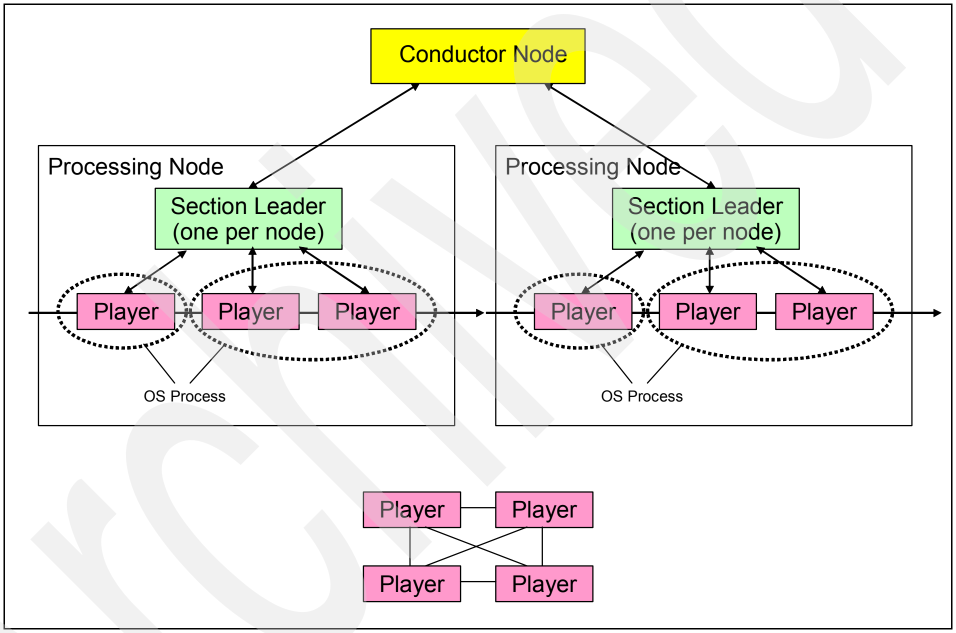


Figure 1-6 Parallel execution flow

1.4 IBM InfoSphere DataStage main functions

In its simplest form, IBM InfoSphere DataStage performs data transformation and movement from source systems to target systems in batch and in real time. The data sources might include indexed files, sequential files, relational databases, archives, external data sources, enterprise applications, and message queues.

DataStage manages data that arrives and data that is received on a periodic or scheduled basis. It enables companies to solve large-scale business problems with high-performance processing of massive data volumes. By leveraging the parallel processing capabilities of multiprocessor hardware platforms, DataStage can scale to satisfy the demands of ever-growing data volumes, stringent real-time requirements, and ever-shrinking batch windows.

Leveraging the combined suite of IBM Information Server, DataStage can simplify the development of authoritative master data by showing where and how information is stored across source systems. DataStage can also consolidate disparate data into a single, reliable record, cleanses and standardizes information, removes duplicates, and links records together across systems. This master record can be loaded into operational data stores, data warehouses, or master data applications such as IBM MDM using IBM InfoSphere DataStage.

IBM InfoSphere DataStage delivers four core capabilities:

- Connectivity to a wide range of mainframe, legacy, and enterprise applications, databases, file formats, and external information sources.
- Prebuilt library of more than 300 functions including data validation rules and very complex transformations.
- Maximum throughput using a parallel, high-performance processing architecture.
- Enterprise-class capabilities for development, deployment, maintenance, and high-availability. It leverages metadata for analysis and maintenance. It also operates in batch, real time, or as a Web service.

IBM InfoSphere DataStage enables an integral part of the information integration process — data transformation as shown in Figure 1-1 on page 3.

In the following sections, we briefly describe the following aspects of IBM InfoSphere DataStage:

- Data transformation
- Jobs
- Parallel processing

1.4.1 Data transformation

Data transformation and movement is the process by which source data is selected, converted, and mapped to the format required by targeted systems. The process manipulates data to bring it into compliance with business, domain, and integrity rules and with other data in the target environment. Transformation can take some of the following forms:

- Aggregation

  Consolidating or summarizing data values into a single value. Collecting daily sales data to be aggregated to the weekly level is a common example of aggregation.

- Basic conversion

  Ensuring that data types are correctly converted and mapped from source to target columns.

- Cleansing

  Resolving inconsistencies and fixing the anomalies in source data.

- Derivation

  Transforming data from multiple sources by using a complex business rule or algorithm.

- Enrichment

  Combining data from internal or external sources to provide additional meaning to the data.

- Normalizing

  Reducing the amount of redundant and potentially duplicated data.

- Combining

  The process of combining data from multiple sources via parallel Lookup, Join, or Merge operations.

- Pivoting

  Converting records in an input stream to many records in the appropriate table in the data warehouse or data mart.

- Sorting

  Grouping related records and sequencing data based on data or string values.

1.4.2 Jobs

An IBM InfoSphere DataStage job consists of individual stages linked together which describe the flow of data from a data source to a data target.

A stage usually has at least one data input and/or one data output. However, some stages can accept more than one data input, and output to more than one stage. Each stage has a set of predefined and editable properties that tell it how to perform or process data. Properties might include the file name for the Sequential File stage, the columns to sort, the transformations to perform, and the database table name for the DB2 stage. These properties are viewed or edited using stage editors. Stages are added to a job and linked together using the Designer. Figure 1-7 shows some of the stages and their iconic representations.

Icon	Stage	Description
	Transformer stage	Performs any required conversions on an input data set, and then passes the data to another processing stage or to a stage that writes data to a target database or file.
	Sort stage	Performs complex high-speed sort operations.
	Aggregator stage	Classifies data rows from a single input data set into groups and computes totals or aggregations for each group.
	Complex Flat File stage	Extracts data from a flat file containing complex data structures, such as arrays or groups.
	DB2 stage	Reads data from or writes data to IBM DB2.

Figure 1-7 Stage examples

Stages and links can be grouped in a shared container. Instances of the shared container can then be reused in different parallel jobs. You can also define a local container within a job — this groups stages and links into a single unit, but can only be used within the job in which it is defined.

The different types of jobs have different stage types. The stages that are available in the Designer depend on the type of job that is currently open in the Designer.

Parallel Job stages are organized into different groups on the Designer palette as follows:

- General includes stages such as Container and Link.
- Data Quality includes stages such as Investigate, Standardize, Reference Match, and Survive.

> **Note:** Applies when IBM InfoSphere QualityStage is installed.

- Database includes stages such as Classic Federation, DB2 UDB, DB2 UDB/Enterprise, Oracle, Sybase, SQL Server®, Teradata, Distributed Transaction, and ODBC.
- Development/Debug includes stages such as Peek, Sample, Head, Tail, and Row Generator.
- File includes stages such as Complex Flat File, Data Set, Lookup File Set, and Sequential File.
- Processing includes stages such as Aggregator, Copy, FTP, Funnel, Join, Lookup, Merge, Remove Duplicates, Slowly Changing Dimension, Surrogate Key Generator, Sort, and Transformer
- Real Time includes stages such as Web Services Transformer, WebSphere MQ, and Web Services Client.
- Restructure includes stages such as Column Export and Column Import.

> **Note:** For details on all the available stages, refer to *IBM WebSphere DataStage and QualityStage Parallel Job Developer Guide*, SC18-9891-00 and relevant connectivity guides for stages concerned with connecting to external data sources and data targets.

1.4.3 Parallel processing

Figure 1-8 represents one of the simplest jobs you could have — a data source, a Transformer (conversion) stage, and the data target. The links between the stages represent the flow of data into or out of a stage.

In a parallel job, each stage would normally (but not always) correspond to a process. You can have multiple instances of each process to run on the available processors in your system.

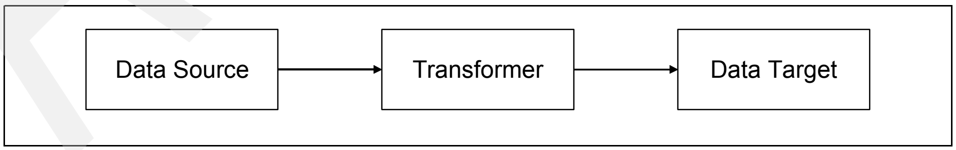


Figure 1-8 Simple IBM InfoSphere DataStage job

A parallel DataStage job incorporates two basic types of parallel processing — pipeline and partitioning. Both of these methods are used at runtime by the Information Server engine to execute the simple job shown in Figure 1-8.

To the DataStage developer, this job would appear the same on your Designer canvas, but you can optimize it through advanced properties.

- Pipeline parallelism

  In the Figure 1-8 example, all stages run concurrently, even in a single-node configuration. As data is read from the Oracle source, it is passed to the Transformer stage for transformation, where it is then passed to the DB2 target. Instead of waiting for all source data to be read, as soon as the source data stream starts to produce rows, these are passed to the subsequent stages. This method is called pipeline parallelism, and all three stages in our example operate simultaneously regardless of the degree of parallelism of the configuration file. The Information Server Engine always executes jobs with pipeline parallelism.

  If you ran the example job on a system with multiple processors, the stage reading would start on one processor and start filling a pipeline with the data it had read. The transformer stage would start running as soon as there was data in the pipeline, process it and start filling another pipeline. The stage writing the transformed data to the target database would similarly start writing as soon as there was data available. Thus all three stages are operating simultaneously.

  Attention: You do not need multiple processors to run in parallel. A single processor is capable of running multiple concurrent processes.

- Partition parallelism

  When large volumes of data are involved, you can use the power of parallel processing to your best advantage by partitioning the data into a number of separate sets, with each partition being handled by a separate instance of the job stages. Partition parallelism is accomplished at runtime, instead of a manual process that would be required by traditional systems.

  The DataStage developer only needs to specify the algorithm to partition the data, not the degree of parallelism or where the job will execute. Using partition parallelism the same job would effectively be run simultaneously by several processors, each handling a separate subset of the total data. At the end of the job the data partitions can be collected back together again and written to a single data source. This is shown in Figure 1-9.

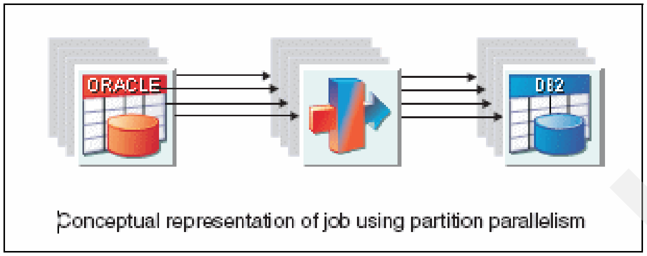


Figure 1-9 Partition parallelism

- Combining pipeline and partition parallelism

  The Information Server engine combines pipeline and partition parallel processing to achieve even greater performance gains. In this scenario you would have stages processing partitioned data and filling pipelines so the next one could start on that partition before the previous one had finished. This is shown in Figure 1-10.

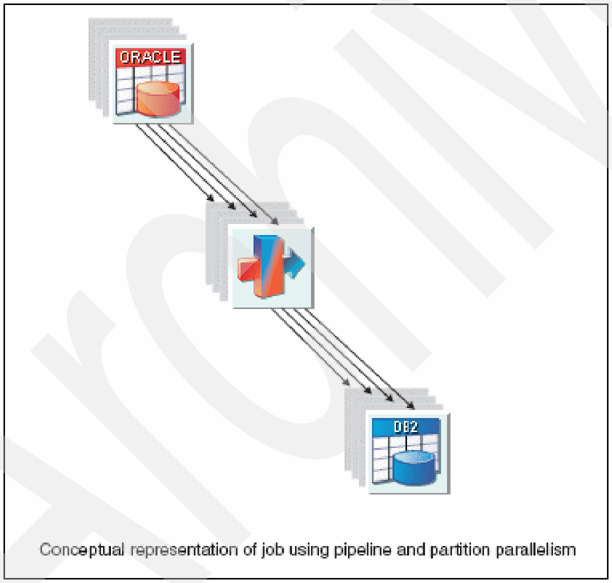


Figure 1-10 Pipeline and partition parallelism

In some circumstances you might want to actually re-partition your data between stages. This could happen, for example, where you want to group data differently. Suppose that you have initially processed data based on customer last name, but now you want to process on data grouped by zip code. You will have to re-partition to ensure that all customers sharing the same zip code are in the same group. DataStage allows you to re-partition between stages as and when necessary. With the Information Server engine, re-partitioning happens in memory between stages, instead of writing to disk.

For full details on parallelism, refer to *IBM WebSphere DataStage and QualityStage Parallel Job Developer Guide*, SC18-9891-00.

1.5 Best practices overview

This section provides an overview of recommendations for standard practices.

> **Note:** A detailed discussion of these practices is beyond the scope of this Redbooks publication, and you should speak to your Account Executive to engage IBM IPS Services.

The recommendations are categorized as follows:

- Standards
- Development guidelines
- Component usage
- DataStage Data Types
- Partitioning data
- Collecting data
- Sorting
- Stage specific guidelines

1.5.1 Standards

It is important to establish and follow consistent standards in:

- Directory structures for installation and application support directories.
- Naming conventions, especially for DataStage Project categories, stage names, and links.

All DataStage jobs should be documented with the Short Description field, as well as Annotation fields.

It is the DataStage developer's responsibility to make personal backups of their work on their local workstation, using DataStage's DSX export capability. This can also be used for integration with source code control systems.

1.5.2 Development guidelines

Modular development techniques should be used to maximize re-use of DataStage jobs and components:

- Job parameterization allows a single job design to process similar logic instead of creating multiple copies of the same job. The Multiple-Instance job property allows multiple invocations of the same job to run simultaneously.
- A set of standard job parameters should be used in DataStage jobs for source and target database parameters (DSN, user, password, etc.) and directories where files are stored. To ease re-use, these standard parameters and settings should be made part of a Designer Job Parameter Sets.
- Create a standard directory structure outside of the DataStage project directory for source and target files, intermediate work files, and so forth.
- Where possible, create re-usable components such as parallel shared containers to encapsulate frequently-used logic.
- DataStage Template jobs should be created with:
  - Standard parameters such as source and target file paths, and database login properties
  - Environment variables and their default settings
  - Annotation blocks
- Job Parameters should always be used for file paths, file names, database login settings.
- Standardized Error Handling routines should be followed to capture errors and rejects.

1.5.3 Component usage

The following guidelines should be followed when constructing parallel jobs in IBM InfoSphere DataStage Enterprise Edition:

- Never use Server Edition components (BASIC Transformer, Server Shared Containers) within a parallel job. BASIC Routines are appropriate only for job control sequences.
- Always use parallel Data Sets for intermediate storage between jobs unless that specific data also needs to be shared with other applications.
- Use the Copy stage as a placeholder for iterative design, and to facilitate default type conversions.
- Use the parallel Transformer stage (not the BASIC Transformer) instead of the Filter or Switch stages.

- Use BuildOp stages only when logic cannot be implemented in the parallel Transformer.

1.5.4 DataStage data types

The following guidelines should be followed with DataStage data types:

- Be aware of the mapping between DataStage (SQL) data types and the internal DS/EE data types. If possible, import table definitions for source databases using the Orchestrate Schema Importer (orchdbutil) utility.
- Leverage default type conversions using the Copy stage or across the Output mapping tab of other stages.

1.5.5 Partitioning data

In most cases, the default partitioning method (Auto) is appropriate. With Auto partitioning, the Information Server Engine will choose the type of partitioning at runtime based on stage requirements, degree of parallelism, and source and target systems. While Auto partitioning will generally give correct results, it might not give optimized performance. As the job developer, you have visibility into requirements, and can optimize within a job and across job flows.

Given the numerous options for keyless and keyed partitioning, the following objectives form a methodology for assigning partitioning:

- Objective 1

  Choose a partitioning method that gives close to an equal number of rows in each partition, while minimizing overhead. This ensures that the processing workload is evenly balanced, minimizing overall run time.

- Objective 2

  The partition method must match the business requirements and stage functional requirements, assigning related records to the same partition if required.

  Any stage that processes groups of related records (generally using one or more key columns) must be partitioned using a keyed partition method.

  This includes, but is not limited to: Aggregator, Change Capture, Change Apply, Join, Merge, Remove Duplicates, and Sort stages. It might also be necessary for Transformers and BuildOps that process groups of related records.

> **Note:** In satisfying the requirements of this second objective, it might not be possible to choose a partitioning method that gives an almost equal number of rows in each partition.

- Objective 3

  Unless partition distribution is highly skewed, minimize re-partitioning, especially in cluster or Grid configurations.

  Re-partitioning data in a cluster or Grid configuration incurs the overhead of network transport.

- Objective 4

  Partition method should not be overly complex. The simplest method that meets the above objectives will generally be the most efficient and yield the best performance.

  Using the above objectives as a guide, the following methodology can be applied:

  a. Start with Auto partitioning (the default).

  b. Specify Hash partitioning for stages that require groups of related records as follows:

     - Specify only the key column(s) that are necessary for correct grouping as long as the number of unique values is sufficient
     - Use Modulus partitioning if the grouping is on a single integer key column
     - Use Range partitioning if the data is highly skewed and the key column values and distribution do not change significantly over time (Range Map can be reused)

  c. If grouping is not required, use Round Robin partitioning to redistribute data equally across all partitions.

     - Especially useful if the input Data Set is highly skewed or sequential

  d. Use Same partitioning to optimize end-to-end partitioning and to minimize re-partitioning

     - Be mindful that Same partitioning retains the degree of parallelism of the upstream stage
     - Within a flow, examine up-stream partitioning and sort order and attempt to preserve for down-stream processing. This may require re-examining key column usage within stages and re-ordering stages within a flow (if business requirements permit).

Across jobs, persistent Data Sets can be used to retain the partitioning and sort order. This is particularly useful if downstream jobs are run with the same degree of parallelism (configuration file) and require the same partition and sort order.

1.5.6 Collecting data

Given the options for collecting data into a sequential stream, the following guidelines form a methodology for choosing the appropriate collector type:

1. When output order does not matter, use Auto partitioning (the default).
2. Consider how the input Data Set has been sorted:
   – When the input Data Set has been sorted in parallel, use Sort Merge collector to produce a single, globally sorted stream of rows.
   – When the input Data Set has been sorted in parallel and Range partitioned, the Ordered collector might be more efficient.
3. Use a Round Robin collector to reconstruct rows in input order for round-robin partitioned input Data Sets, as long as the Data Set has not been re-partitioned or reduced.

1.5.7 Sorting

Apply the following methodology when sorting in an IBM InfoSphere DataStage Enterprise Edition data flow:

1. Start with a link sort.
2. Specify only necessary key column(s).
3. Do not use Stable Sort unless needed.
4. Use a stand-alone Sort stage instead of a Link sort for options that are not available on a Link sort:
   – The “Restrict Memory Usage” option should be included here. If you want more memory available for the sort, you can only set that via the Sort Stage — not on a sort link. The environment variable $APT_TSORT_STRESS_BLOCKSIZE can also be used to set sort memory usage (in MB) per partition.
   – Sort Key Mode, Create Cluster Key Change Column, Create Key Change Column, Output Statistics.
   – Always specify “DataStage” Sort Utility for standalone Sort stages.
   – Use the “Sort Key Mode=Don’t Sort (Previously Sorted)” to resort a sub-grouping of a previously-sorted input Data Set.

5. Be aware of automatically-inserted sorts:
   – Set $APT_SORT_INSERTION_CHECK_ONLY to verify but not establish required sort order.
6. Minimize the use of sorts within a job flow.
7. To generate a single, sequential ordered result set, use a parallel Sort and a Sort Merge collector.

1.5.8 Stage specific guidelines

The guidelines by stage are as follows:

- Transformer

  Take precautions when using expressions or derivations on nullable columns within the parallel Transformer:

  – Always convert nullable columns to in-band values before using them in an expression or derivation.

  – Always place a reject link on a parallel Transformer to capture / audit possible rejects.

- Lookup

  It is most appropriate when reference data is small enough to fit into available shared memory. If the Data Sets are larger than available memory resources, use the Join or Merge stage.

  Limit the use of database Sparse Lookups to scenarios where the number of input rows is significantly smaller (for example 1:100 or more) than the number of reference rows, or when exception processing.

- Join

  Be particularly careful to observe the nullability properties for input links to any form of Outer Join. Even if the source data is not nullable, the non-key columns must be defined as nullable in the Join stage input in order to identify unmatched records.

- Aggregators

  Use Hash method Aggregators only when the number of distinct key column values is small. A Sort method Aggregator should be used when the number of distinct key values is large or unknown.

- Database Stages

  The following guidelines apply to database stages:

  – Where possible, use the Connector stages or native parallel database stages for maximum performance and scalability.

  – The ODBC Connector and ODBC Enterprise stages should only be used when a native parallel stage is not available for the given source or target database.

  – When using Oracle, DB2, or Informix databases, use Orchestrate Schema Importer (orchdbutil) to properly import design metadata.

  – Take care to observe the data type mappings.

  – If possible, use an SQL where clause to limit the number of rows sent to a DataStage job.

  – Avoid the use of database stored procedures on a per-row basis within a high-volume data flow. For maximum scalability and parallel performance, it is best to implement business rules natively using DataStage parallel components.

2

IBM InfoSphere DataStage stages

In this chapter we provide an overview of some of the commonly used stages in IBM InfoSphere DataStage, including the new stages available in Version 8.1.

2.1 Introduction

As mentioned in “Jobs” on page 22, an IBM InfoSphere DataStage job consists of individual stages linked together, which describe the flow of data from a data source to a data target.

A stage usually has at least one data input and/or one data output. However, some stages can accept more than one data input, and output to more than one stage. Each stage has a set of predefined and editable properties that tell it how to perform or process data. Properties might include the file name for the Sequential File stage, the columns to sort, the transformations to perform, and the database table name for the DB2 stage. These properties are viewed or edited using stage editors. Stages are added to a job and linked together using the Designer.

In this chapter we focus on the most commonly used stages and the new stages available in Version 8.1, as follows:

- Aggregator
- Complex Flat File
- Column Import
- Column Export
- Data Set
- Distributed Transaction (new in Version 8.1)
- FTP Enterprise
- Funnel
- Join
- Lookup
- Merge
- Sequential File
- Slowly Changing Dimension
- Sort
- Surrogate Key Generator
- Transformer

For details on all the available stages, refer to *IBM WebSphere DataStage and QualityStage Parallel Job Developer Guide*, SC18-9891-00, and relevant connectivity guides for stages concerned with connecting to external data sources and data targets.

Attention: In all the following sections, to avoid overburdening you with excessive screen captures, we have *not* included all the panels that you would typically navigate through in order to perform the desired function. Instead we have focused on including select screen captures (and in some cases, just portions of them) that highlight the key items of interest, thereby skipping both initial screen captures, as well as some intervening ones, in the process. Screen captures involving default values are not shown here either. And finally, also not covered is a discussion of each property of the stages since they are all well described in the *IBM WebSphere DataStage and QualityStage Parallel Job Developer Guide*, SC18-9891-00.

2.2 Aggregator

The Aggregator stage is a processing stage. It classifies data rows from a single input link into groups and computes totals or other aggregate functions for each group. The summed totals for each group are output from the stage via an output link. This is shown in Figure 2-1.

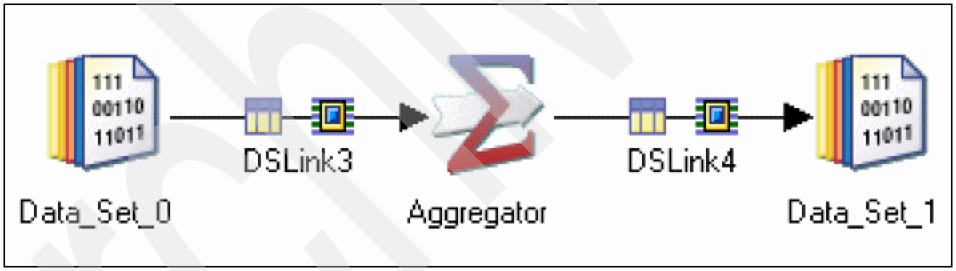


Figure 2-1 Aggregator stage

The Aggregator stage gives you access to grouping and summary operations. Records can be grouped by one or more characteristics, where record characteristics correspond to column values. In other words, a group is a set of records with the same value for one or more columns. For example, transaction records might be grouped by both day of the week and by month. These groupings might show that the busiest day of the week varies by season.

In addition to revealing patterns in your data, grouping can also reduce the volume of data by summarizing the records in each group, making it easier to manage. If you group a large volume of data on the basis of one or more characteristics of the data, the resulting data set is generally much smaller than the original and is therefore easier to analyze.

> **Note:** In a parallel environment, the way that you partition data before grouping and summarizing it can affect the results. For example, if you partitioned using the round robin method, records with identical values in the column you are grouping on would end up in different partitions. If you then performed a sum operation within these partitions, you would not be operating on all the relevant rows. In such circumstances you might want to key partition the data on one or more of the grouping keys to ensure that your groups are "entire" (which is another partitioning method).

Figure 2-2 on page 40 through Figure 2-7 on page 43 show an example of an Aggregator stage in a job ("J15_Daily_CreateSalesAggDS (Day 1)" on page 387 in the retail industry scenario described in "Retail industry scenario" on page 140), as follows:

1. Figure 2-2 on page 40 shows the job that enhances the sales transaction records for input to the Slowly Changing Dimension stage. This is described in "J15_Daily_CreateSalesAggDS (Day 1)" on page 387 and is not repeated here. Instead, we only focus on the configuration of the Aggregator stage in this job.
2. The Agg_Sales - Aggregator window (Figure 2-3 on page 41) shows the configured properties under the **Properties** tab in the Stage page. It allows you to specify properties that determine what the stage actually does. Some of the properties are mandatory, although many have default settings. We only described some of the more important properties here, as follows:
   - The Grouping Keys category identifies the input columns you are using as group keys. It shows seven columns (CUSTOMER_ID, PRODUCT_ID, STORE_ID, MEMBERSHIP_EXPIRE_DT, MEMBERSHIP_LEVEL, and MANAGER_NAME) forming the Grouping Keys.
   - The Aggregations category has multiple properties Aggregation type, Column for calculation, and Count output column.
     - Aggregation type property allows you to specify the type of aggregation operation your stage is performing, such as Calculate (the default), Recalculate, and Count Rows. Since this was a calculation, we selected Calculate.
     - For the Calculate aggregate type, you can identify the column or columns in the input whose contents you want to summarize, by applying one or more aggregate functions to it. We selected three columns (TOTAL_LOCAL_CURRENCY, QUANTITY, and TOTAL_USD) for calculation using the Sum function. The output column in this case is the same as the input column. The output type of a calculation or recalculation column is double, but setting this property

(Decimal Output = 10,2) causes it to default to decimal with the appropriate default precision and scale.

   – The Options category has the following properties set:

      • The Allow Null Output property is set to False to specify that the null value will have 0 substituted when all input values for the calculation column are null.

      • The aggregate stage has two modes of operation — hash and sort. Your choice of mode depends primarily on the number of groupings in the input data set, taking into account the amount of memory available. Hash mode is typically used for a relatively small number of groups. We chose sort.

3. We set all the properties to default under the **Advanced** tab in the Stage page.

   – The Execution Mode specifies whether the stage can execute in parallel mode or sequential mode.

   – The Combinability mode allows IBM InfoSphere DataStage to combine the operators that underlie parallel stages so that they run in the same process if it is sensible for this type of stage.

   – Preserve partitioning when Set, requests that the next stage in the job attempt to maintain the partitioning.

   – Node pool and resource constraints when selected constrains parallel execution to the node pool or pools and/or resource pool or pools specified in the grid.

4. Figure 2-4 on page 41 shows the properties under the **Partitioning** tab in the Input page, which allows you to specify details about how the incoming data is partitioned or collected before it is grouped and/or summarized. It also allows you to specify that the data should be sorted before being operated on.

   Since the Aggregator stage is set to execute in parallel, we selected the partitioning method of Same from the Partition type drop-down list, which preserves the partitioning already in place.

5. Figure 2-5 on page 42 shows the columns under the **Columns** tab in the Input page, which specifies the column definitions of incoming data.

6. Figure 2-6 on page 42 shows the properties under the **Mapping** tab in the Output page, which allows you to specify the relationship between the processed data being produced by the Aggregator stage and the Output columns. The left pane shows the input columns and/or the generated columns. The right pane shows the output columns for each link.

   The aggregated columns using the Sum function is mapped as shown in the Derivation field.

7. Figure 2-7 on page 43 shows the columns under the **Columns** tab in the Output page, which specifies the column definitions of outgoing data that you define through mapping.

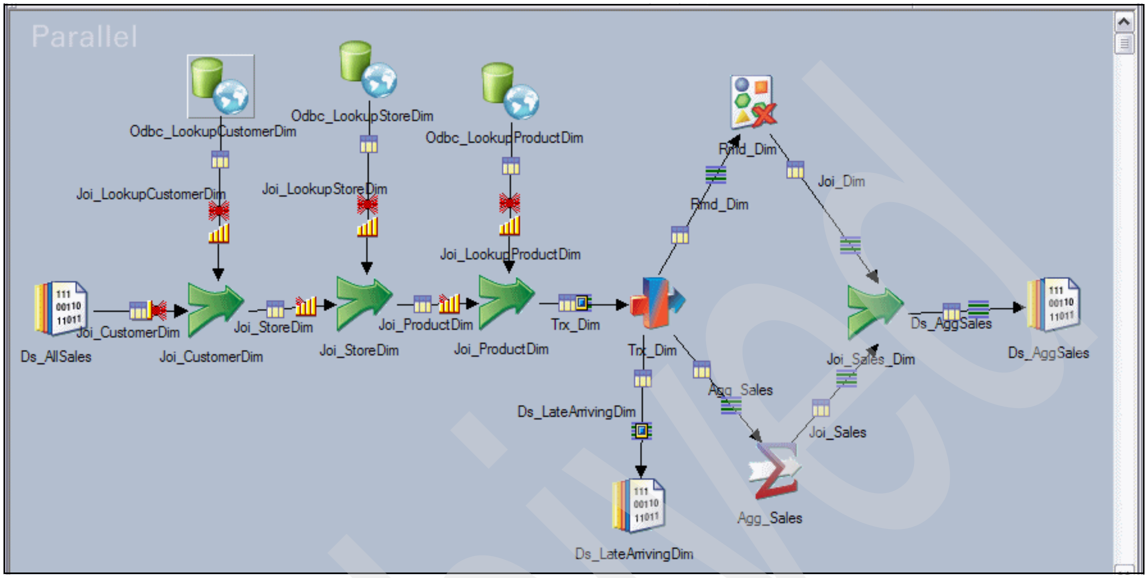


Figure 2-2 Aggregator stage example 1/6

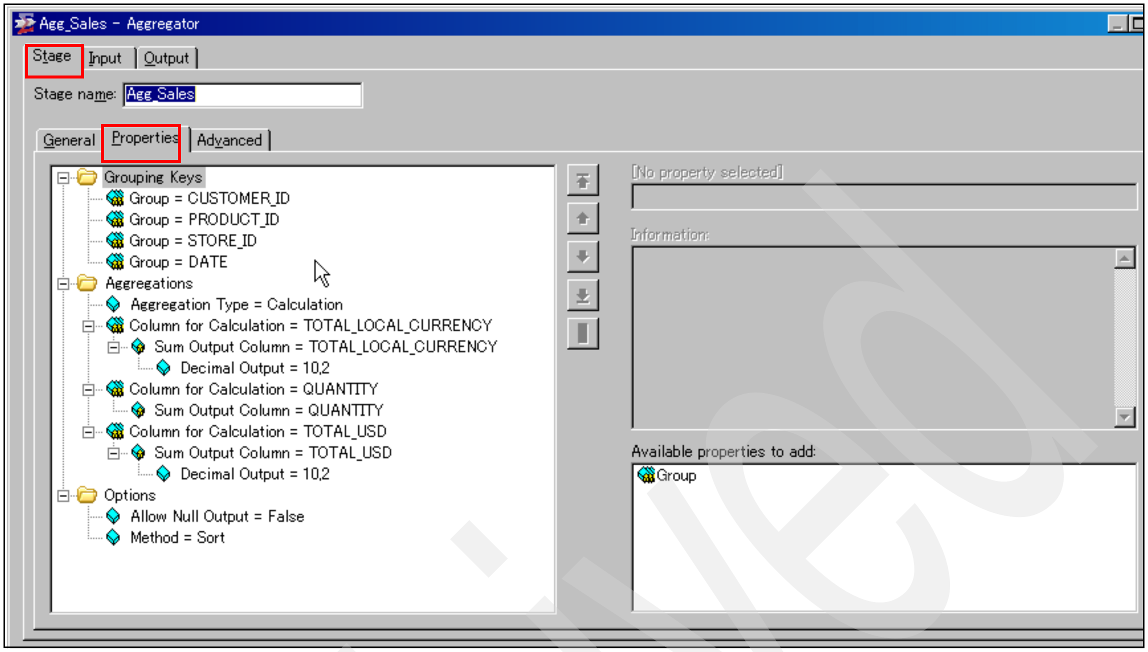


Figure 2-3 Aggregator stage example 2/6

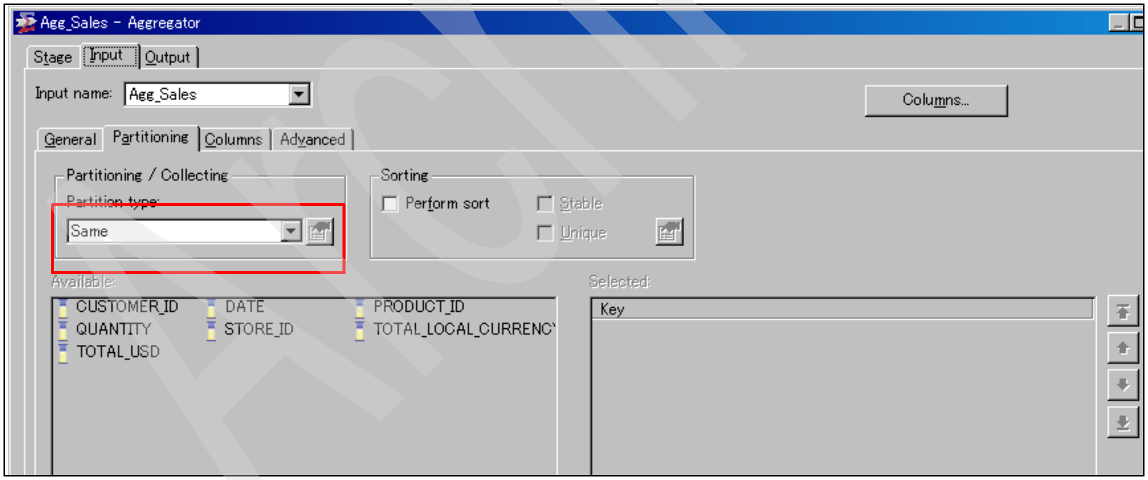


Figure 2-4 Aggregator stage example 3/6

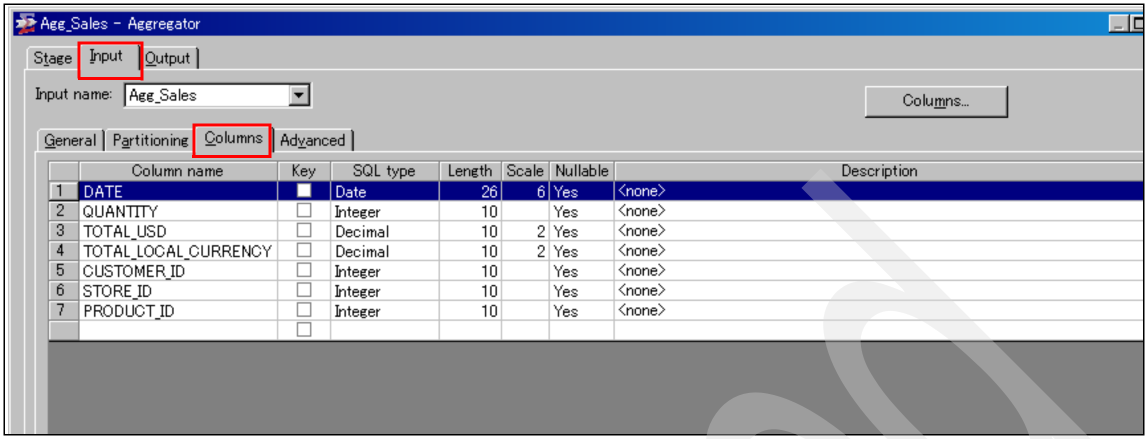


Figure 2-5 Aggregator stage example 4/6

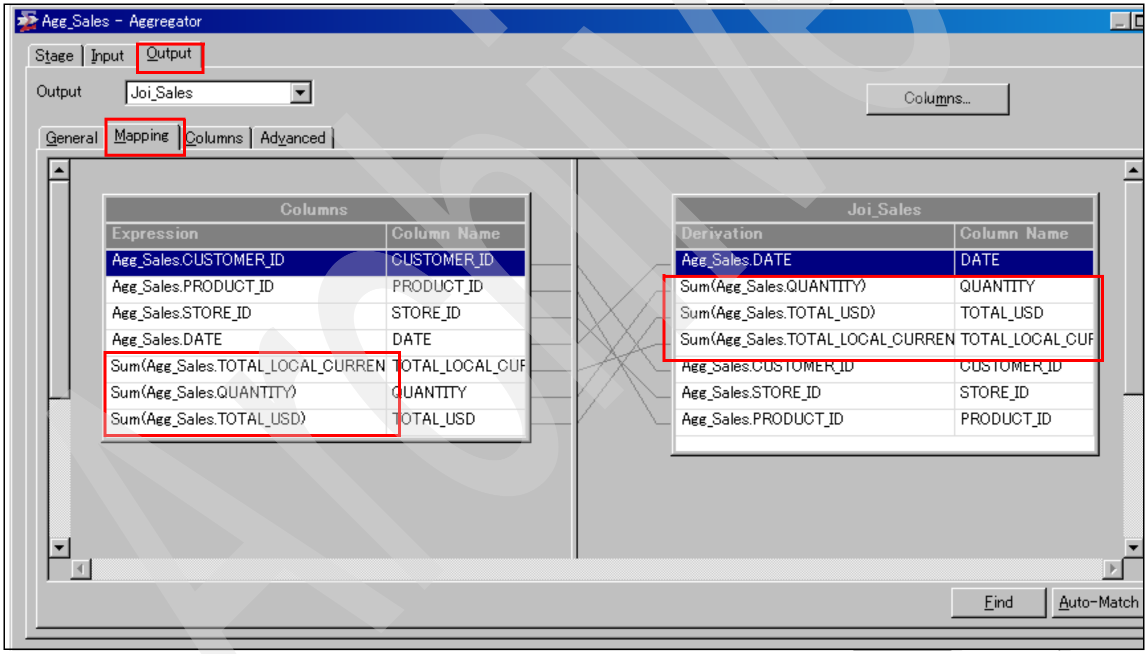


Figure 2-6 Aggregator stage example 5/6

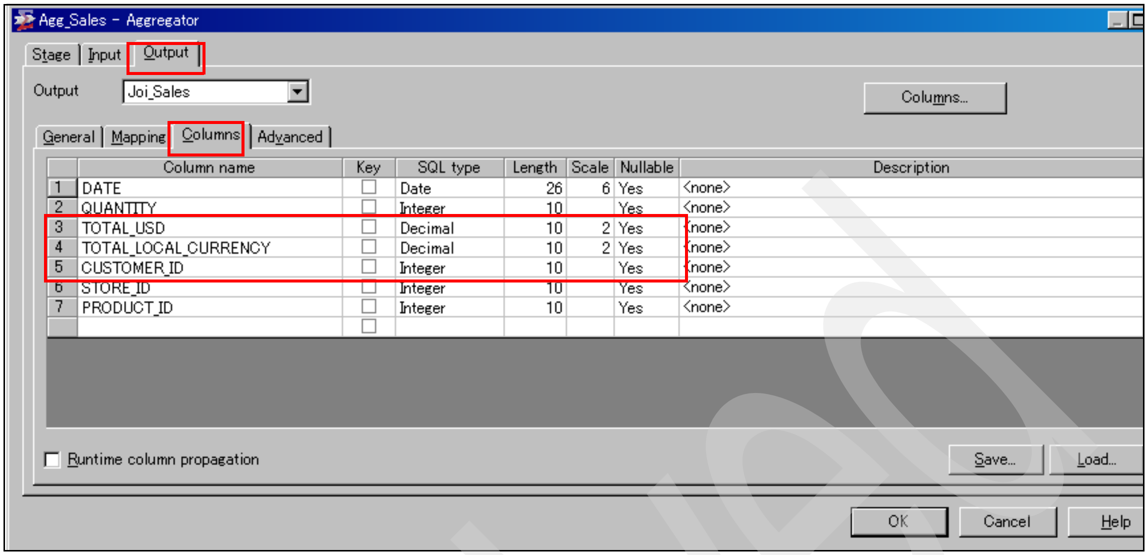


Figure 2-7 Aggregator stage example 6/6

2.3 Complex Flat File

The Complex Flat File (CFF) stage is a file stage. You can use the stage to read a file or write to a file, but you cannot use the same stage instance to do both.

- As a source, the CFF stage can have multiple output links and a single reject link. You can read data from one or more complex flat files, including MVS™ data sets with QSAM and VSAM files. You can also read data from files that contain multiple record types.

  The source data can contain one or more of the following clauses:

  – GROUP
  – REDEFINES
  – OCCURS
  – OCCURS DEPENDING ON

  CFF source stages run in parallel mode when they are used to read multiple files.

  When a CFF stage is defined as a source, you must provide details about the file that the stage will read, create record definitions for the data, define the column metadata, specify record ID constraints, and select output columns.

  – If you are reading data from a file that contains multiple record types, you must create a separate record definition for each type.

- You must define columns to specify what data the CFF stage will read (or write). The fastest way to define column metadata is to load columns from a table definition in the repository. But you can also define column metadata by typing column definitions in the columns grid.

  Mainframe table definitions frequently contain hundreds of columns. If you do not want to display all of these columns in the CFF stage, you can create fillers to save storage space and processing time.

- If you are using the CFF stage to read data from a file that contains multiple record types, you must specify a record ID constraint to identify the format of each record. Columns that are identified in the record ID clause *must be* in the same physical storage location (offset) across records. The constraint must be a simple equality expression, where a column equals a value such as COL1='Y'.

- You can specify which columns from the source file the CFF stage should pass to the output links. You can select columns from multiple record types to output from the stage. If you do not select columns to output on each link, the CFF stage automatically propagates all of the stage columns except group columns to each empty output link. You can also filter the data on each output link from the CFF stage by defining a constraint.

- As a target, the CFF stage can have a single input link and a single reject link. You can write data to one or more complex flat files. You cannot write to MVS data sets or to files that contain multiple record types.

  When a CFF stage is defined as a target, you must provide details about the file that the stage will write, define the record format of the data, and define the column metadata.

The CFF stage can have a single reject link, whether you use the stage as a source or a target.

- For CFF source stages, reject links are supported only if the source file contains a single record type without any OCCURS DEPENDING ON (ODO) columns.
- For CFF target stages, reject links are supported only if the target file does not contain ODO columns.

You cannot change the selection properties of a reject link. The Selection tab for a reject link is blank. You cannot edit the column definitions for a reject link. For writing files, the reject link uses the input link column definitions. For reading files, the reject link uses a single column named “rejected” that contains raw data for the columns that were rejected after reading because they did not match the schema.

Figure 2-8 shows a job that has a Complex Flat File source stage with a single reject link, and a Complex Flat File target stage with a single reject link.

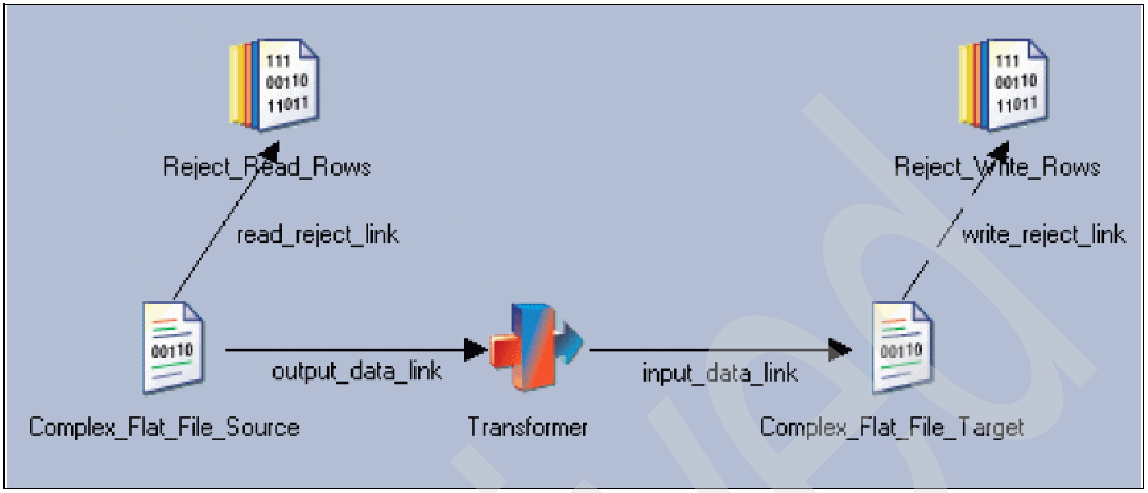


Figure 2-8 Complex Flat File stage

Figure 2-9 on page 46 through Figure 2-19 on page 52 show an example of an Complex Flat File stage in a job ("J02_IL_LoadCustomerDim" on page 184 in the retail industry scenario described in "Retail industry scenario" on page 140), as follows:

1. Figure 2-9 on page 46 shows the job that extracts and processes customer information from a file that contains multiple record types for loading into the dimension table. This is described in "J02_IL_LoadCustomerDim" on page 184 and is not repeated here. Instead, we only focus on the configuration of the CFF stage in this job.

   In the CFF stage, you must provide details about the file that the stage will read, create record definitions for the data, define the column metadata, specify record ID constraints, and select output columns.

2. Figure 2-10 on page 46 shows the **File options** tab in the Stage page, which provides details about the file that the stage will read.
3. Figure 2-11 on page 47 shows the **Record options** tab in the Stage page, which describes the format of the data in the file.
4. Since the stage will be reading a file containing multiple record types, we must create the record definitions of the data. Figure 2-12 on page 47 through Figure 2-14 on page 49 show the **Records** tab in the Stage page, which identify the three record definitions in the customer file by either typing or loading column definitions from the repository.

5. Figure 2-15 on page 49 through Figure 2-17 on page 50 define the record ID constraint for each record (CUSTOMER record type with a value ‘CD’, HOMEADDRESS record type with a value ‘HA’, and WORKADDRESS record type with a value ‘WA’) on the **Records ID** tab.
6. Figure 2-18 on page 51 shows the **Selection** tab in the Output page, which specifies how to read data from the source file. It shows the selection of multiple columns (excluding only the RECTYPE, RECTYPE_2, and RECTYPE_3 columns from the input) for the Trx_Customer output link.
7. Figure 2-19 on page 52 shows the **Constraint** tab in the Output page, which filters the rows (based on the values ‘CD’, ‘HA’, and ‘WA’ in the record type columns in this case) on the output.

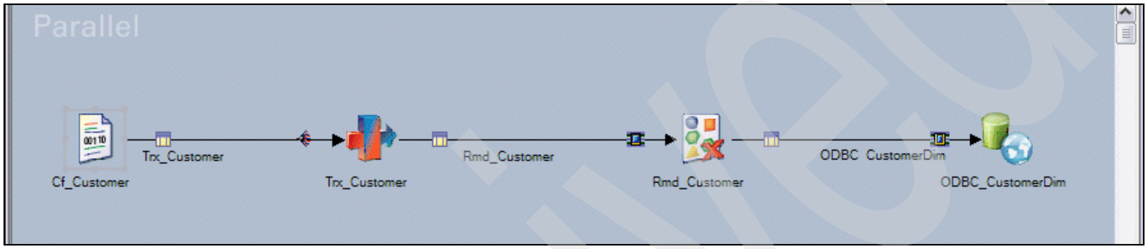


Figure 2-9 Complex Flat File stage example 1/11

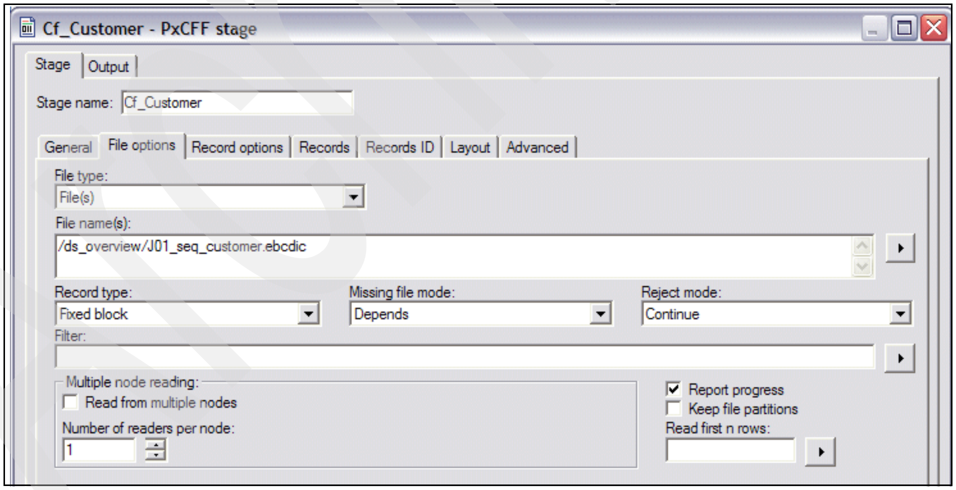


Figure 2-10 Complex Flat File stage example 2/11

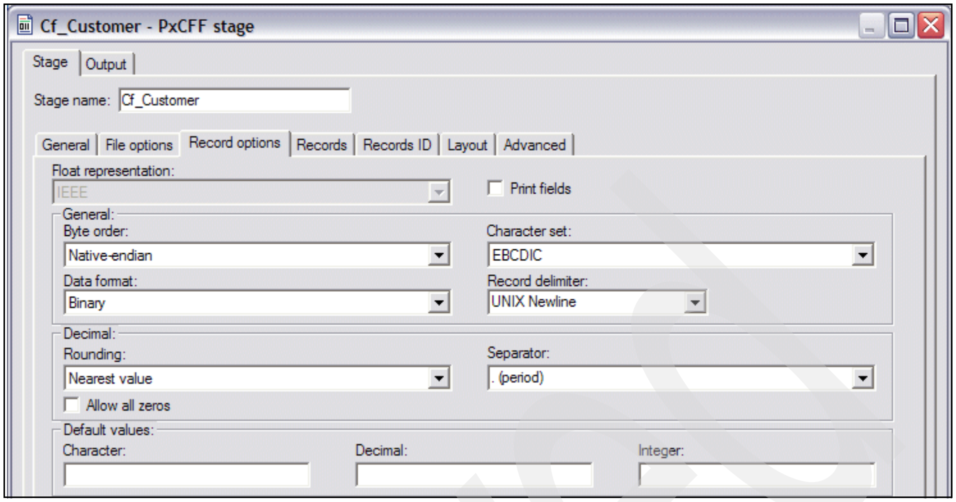


Figure 2-11 Complex Flat File stage example 3/11

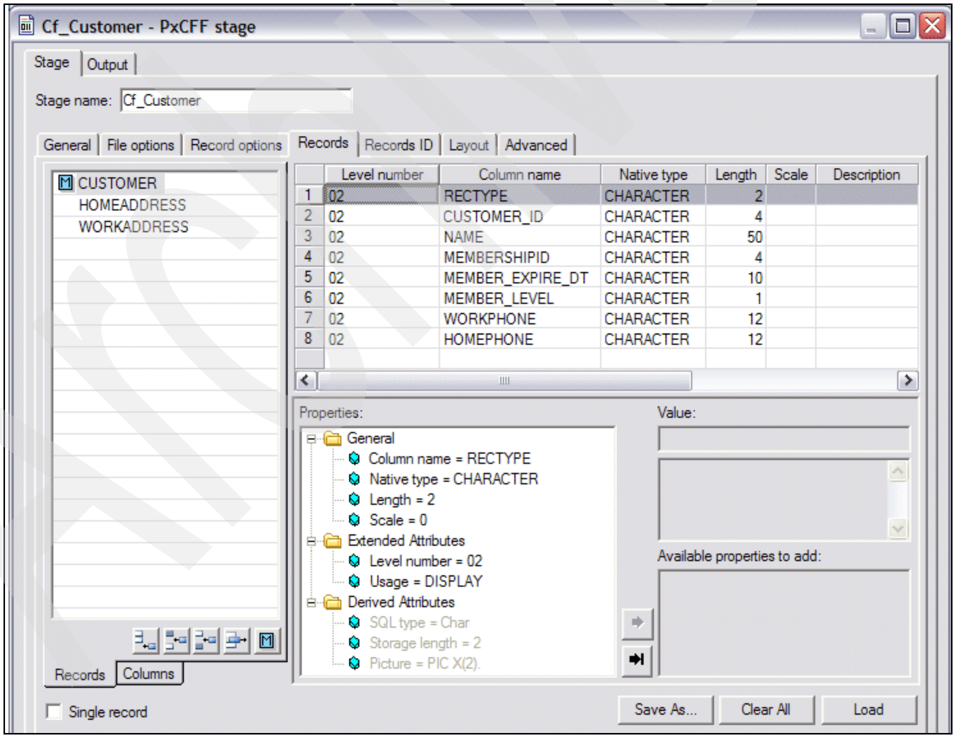


Figure 2-12 Complex Flat File stage example 4/11

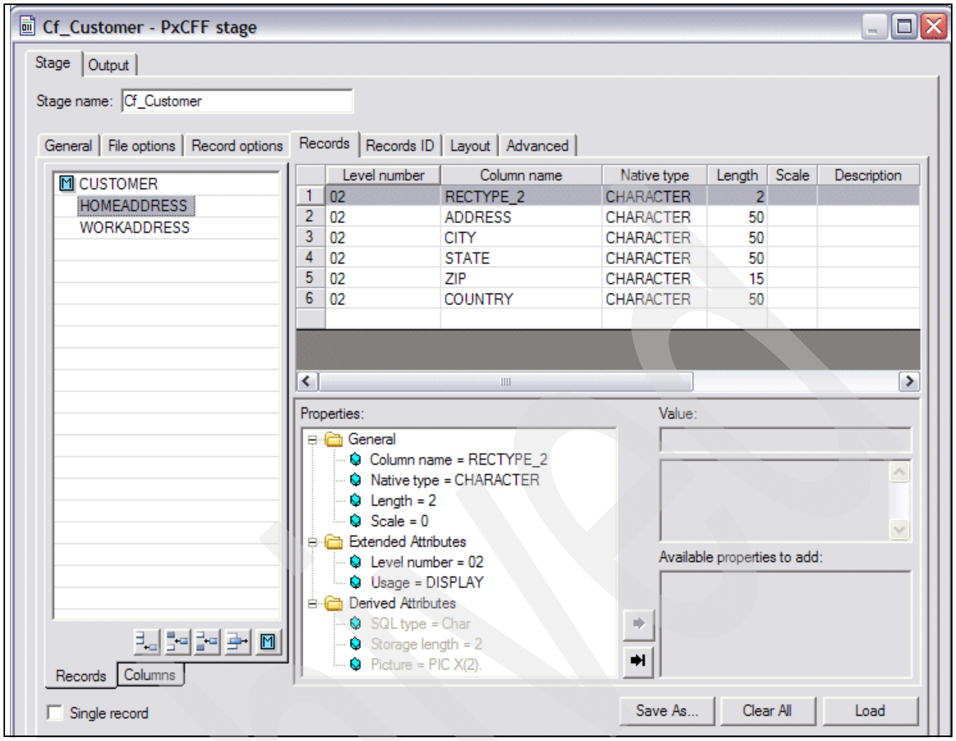


Figure 2-13 Complex Flat File stage example 5/11

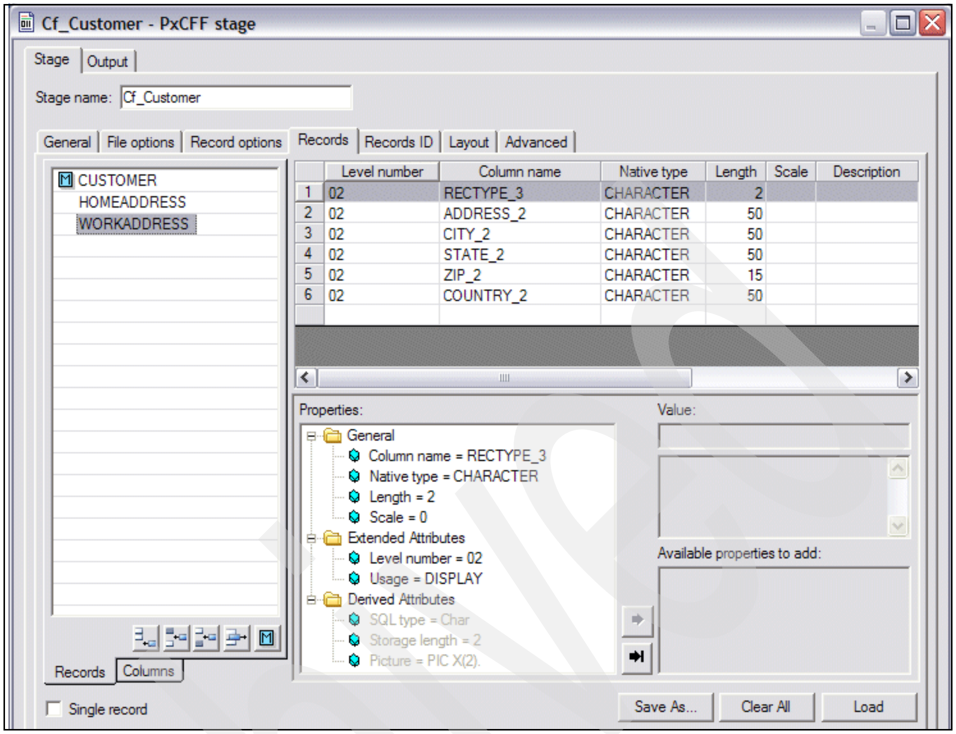


Figure 2-14 Complex Flat File stage example 6/11

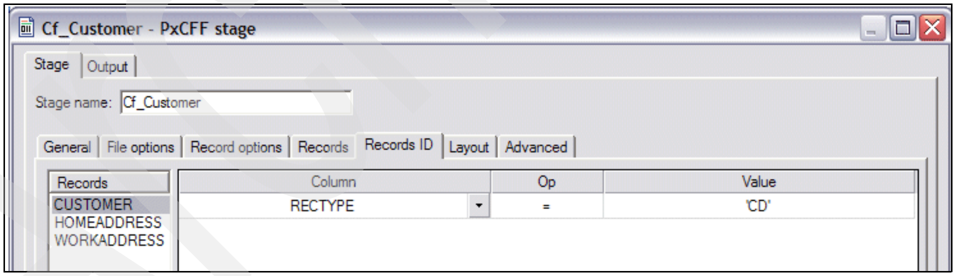


Figure 2-15 Complex Flat File stage example 7/11

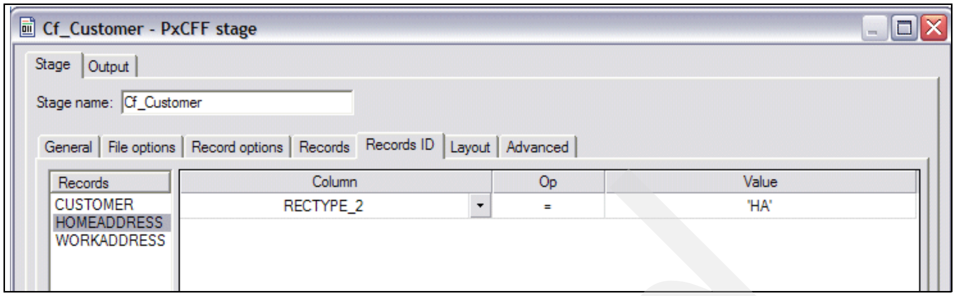


Figure 2-16 Complex Flat File stage example 8/11

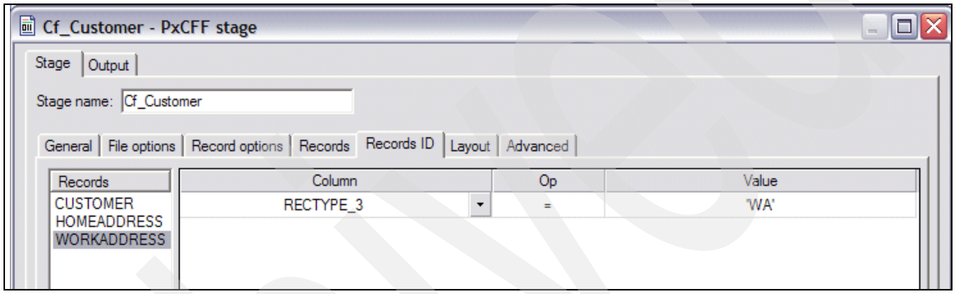


Figure 2-17 Complex Flat File stage example 9/11

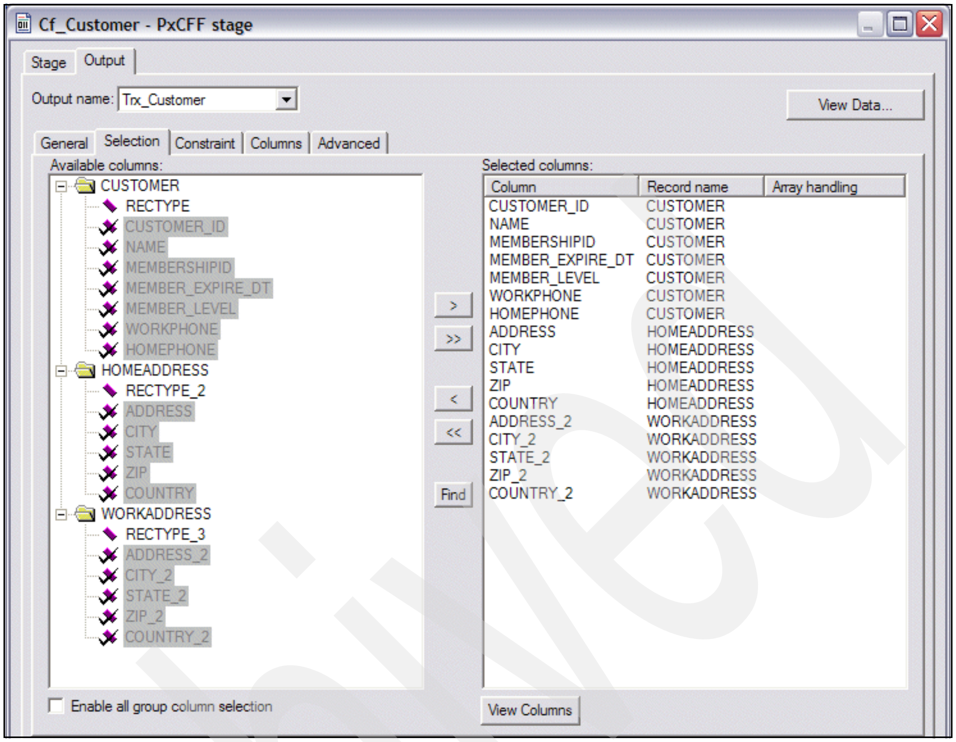


Figure 2-18 Complex Flat File stage example 10/11

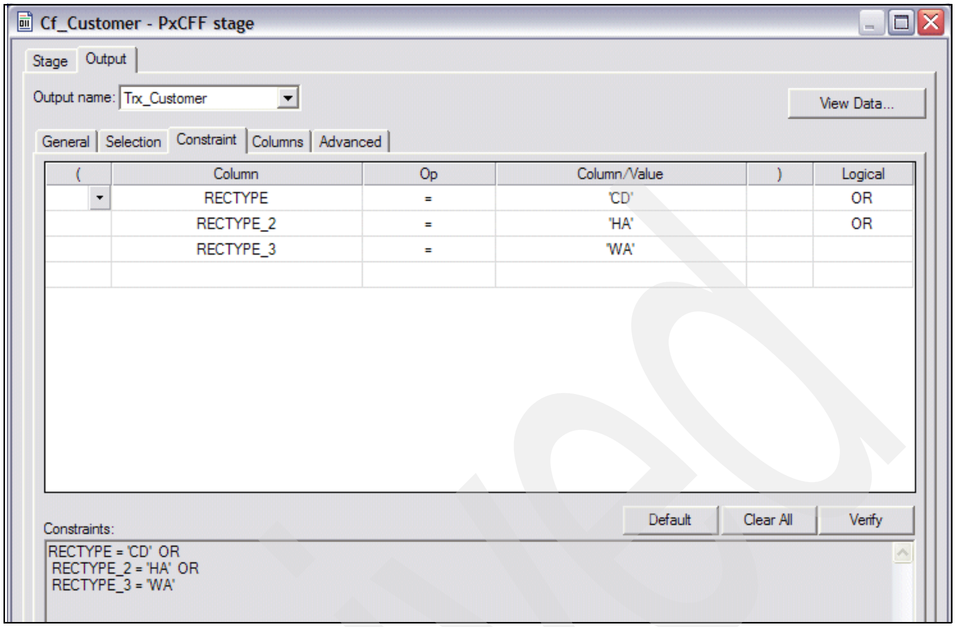


Figure 2-19 Complex Flat File stage example 11/11

2.4 Column Import

The Column Import stage is a restructure stage. It can have a single input link, a single output link and a single reject link as shown here in Figure 2-20. The complement to this stage is the Column Export stage, described in 2.5, “Column Export” on page 60.

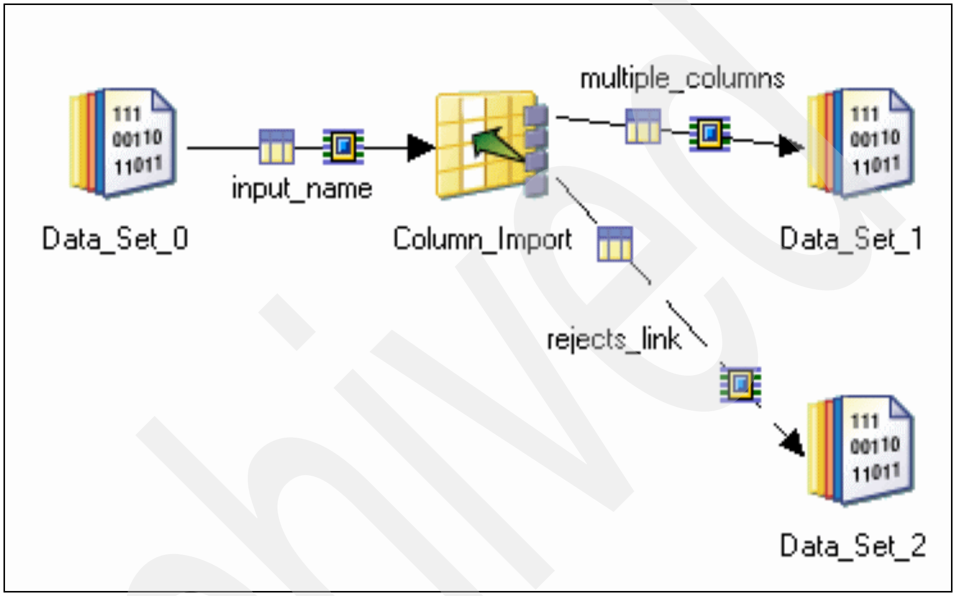


Figure 2-20 Column Import stage

The Column Import stage imports data from a single column and outputs it to one or more columns. You would typically use it to divide data arriving in a single column into multiple columns. The data would be structured in some way to tell the Column Import stage where to make the divisions.

The input column must be a string or binary data; the output columns can be any data type.

You supply an import table definition to specify the target columns and their types. This also determines the order in which data from the import column is written to output columns. Information about the format of the incoming column (for example, how it is delimited) must be provided. You can optionally save reject[1] records and write them to a reject link. In addition to importing a column you can also pass other columns straight through the stage. So, for example, you could pass a key column straight through.

[1] Records whose import was rejected

Figure 2-21 on page 55 through Figure 2-21 on page 55 show an example of a Column Import stage in a job ("J13_Daily_UpdateLookupDim (Day 1)" on page 356 in the retail industry scenario described in "Retail industry scenario" on page 140), as follows:

1. Figure 2-21 on page 55 shows the job that processes changes to attributes of dimension tables arriving via a IBM WebSphere MQ queue. This is described in "J13_Daily_UpdateLookupDim (Day 1)" on page 356 and is not repeated here. Instead, we only focus on the configuration of the Column Import stage in this job.
2. Figure 2-22 on page 56 shows the **Properties** tab in the Stage page, which allows you to specify properties that determine what the stage actually does.
   - The Input category Import input column specifies the name of the column (body_customer) containing the string or binary data to import.
   - The Output category Column method specifies whether the columns to import should be derived from column definitions on the Output page Columns tab (Explicit) or from a schema file (Schema File). We specified Explicit.
   - The Output category Column to Import specifies an output column. The metadata for this column determines the type that the import column will be converted to. The order of the Columns to Import that you specify should match the order on the Columns tab.
   - The Options category Keep Import Column specifies whether the original input column should be transferred to the output data set unchanged in addition to being imported and converted. Default is False.
   - The Options category Reject Mode specification of Continue directs the stage is to continue but report failures to the log file.
3. Figure 2-23 on page 56 shows the **Columns** tab in the Input page, which specifies the column definitions of incoming data.
4. The Output page allows you to specify details about data output from the Column Import stage. Figure 2-24 on page 57 shows the **Format** tab in the Output page, which allows you to specify details about how data in the column you are importing is formatted so the stage can divide it into separate columns.
5. Figure 2-25 on page 58 shows the **Mapping** tab in the Output page, which allows you to specify how the output columns are derived. We recommend that you maintain the automatic mappings of the generated columns when using this stage.
6. Figure 2-26 on page 59 shows the **Columns** tab in the Output page, which specifies the column definitions of the output data. We did not select Runtime column propagation since all columns were explicitly defined.

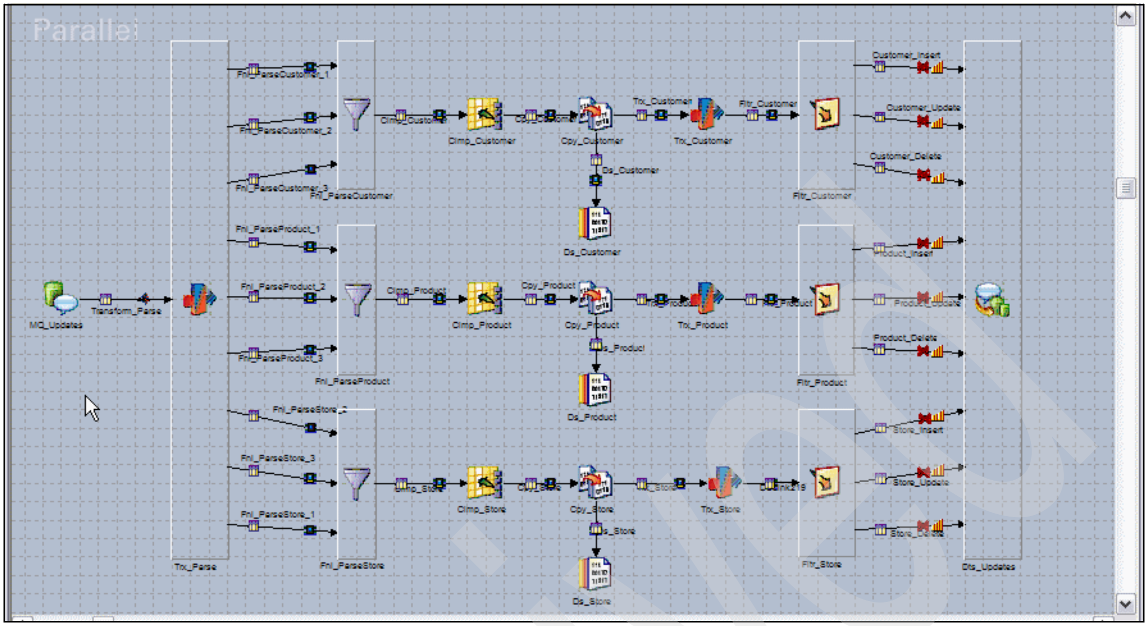


Figure 2-21   Column Import stage example 1/6

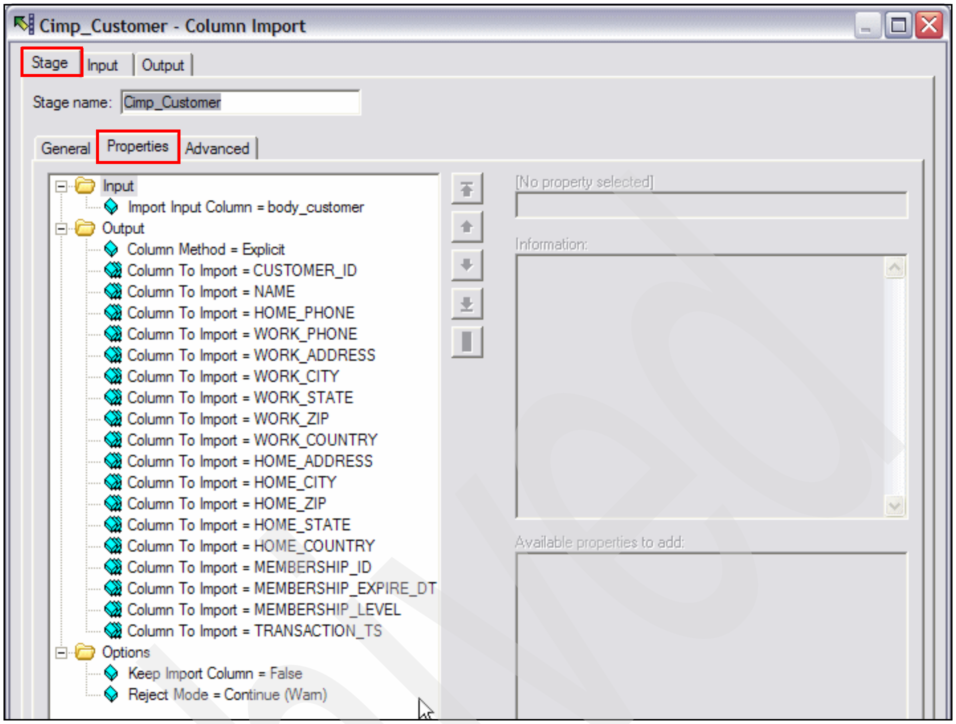


Figure 2-22   Column Import stage example 2/6

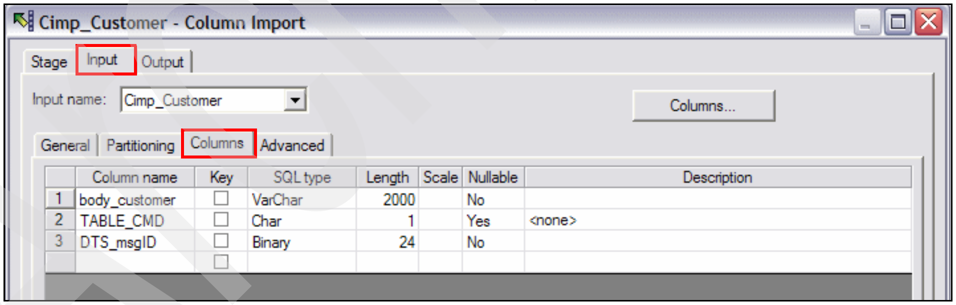


Figure 2-23   Column Import stage example 3/6

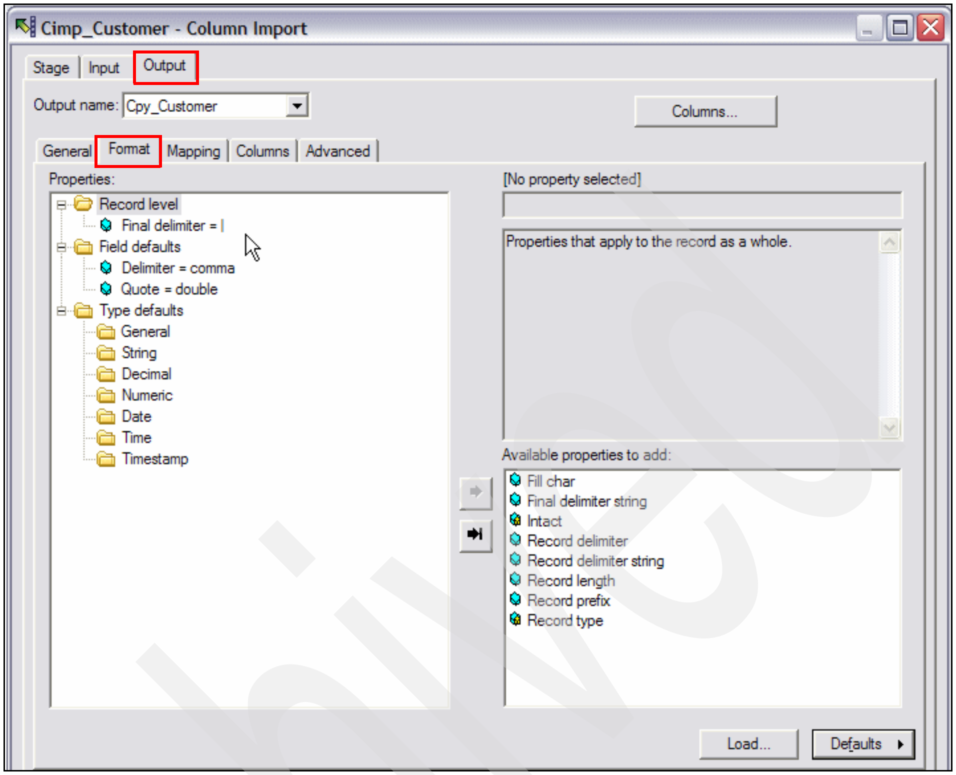


Figure 2-24 Column Import stage example 4/6

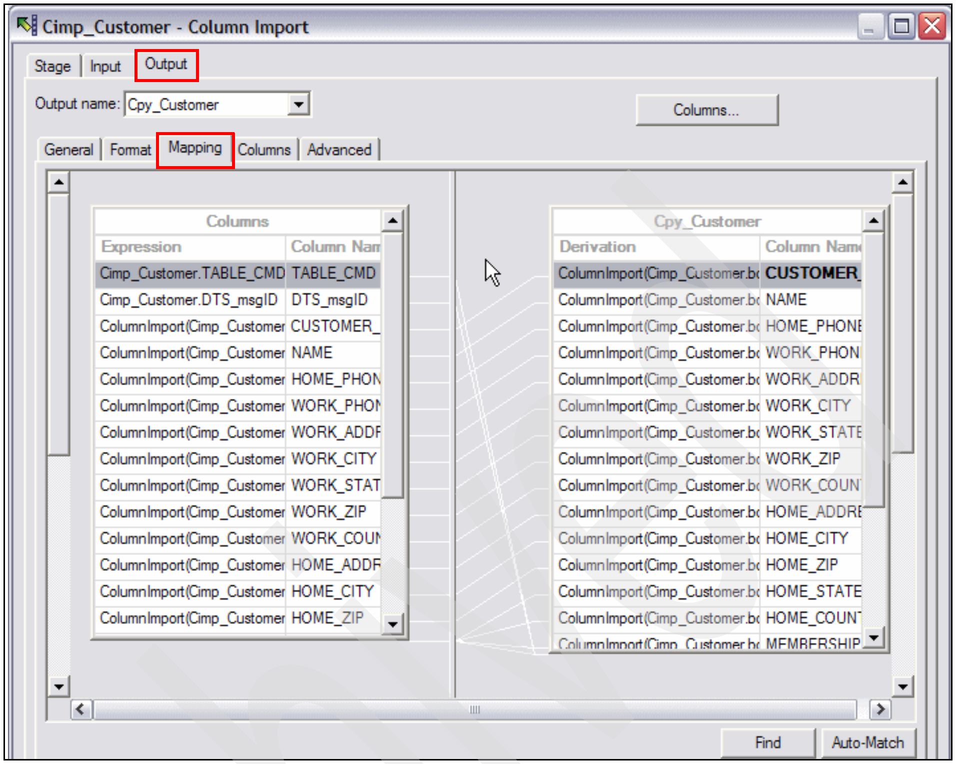


Figure 2-25 Column Import stage example 5/6

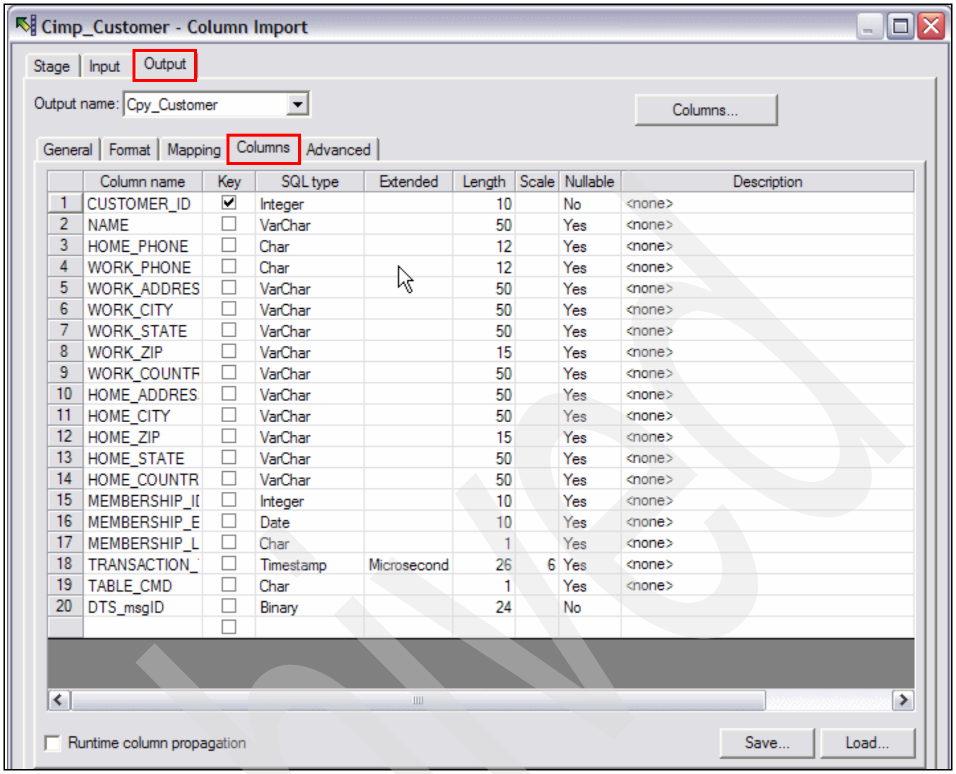


Figure 2-26 Column Import stage example 6/6

2.5 Column Export

The Column Export stage is a restructure stage. It can have a single input link, a single output link, and a single reject link as shown here in Figure 2-27.

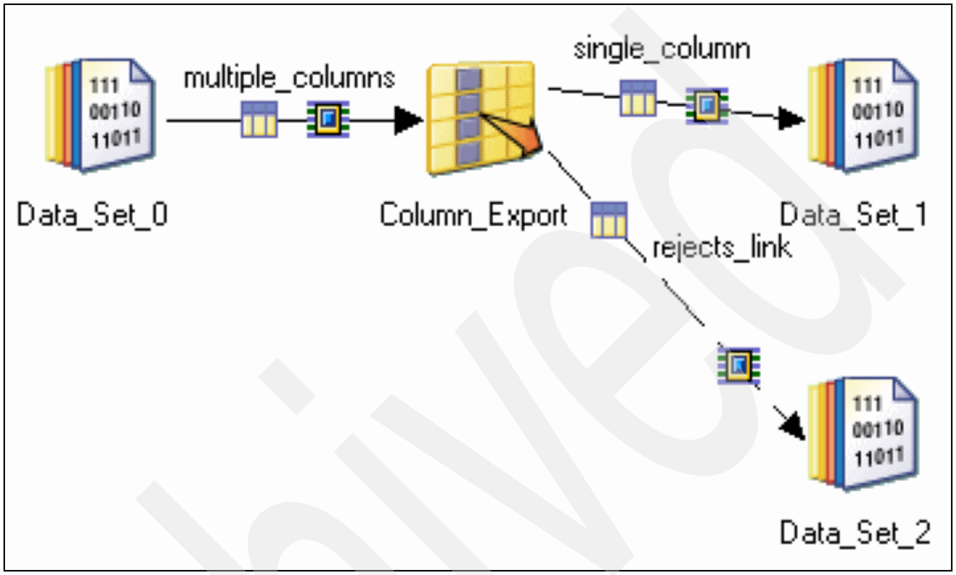


Figure 2-27 Column Export stage

The Column Export stage exports data from a number of columns of different data types into a single column of data type string or binary. It is the complementary stage to Column Import described in 2.4, “Column Import” on page 53.

The input data column definitions determine the order in which the columns are exported to the single output column. You must provide information about how the single column being exported is structured. You can optionally save reject records whose export was rejected. In addition to exporting a column, you can also pass other columns straight through the stage. So, for example, you could pass a key column straight through.

The configuration is an inverse of the configuration corresponding to 2.4, “Column Import” on page 53 and is not repeated here.

2.6 Data Set

The Data Set stage is a file stage. It allows you to read data from or write data to a data set. The stage can have a single input link or a single output link as shown in Figure 2-28. It can be configured to execute in parallel or sequential mode.

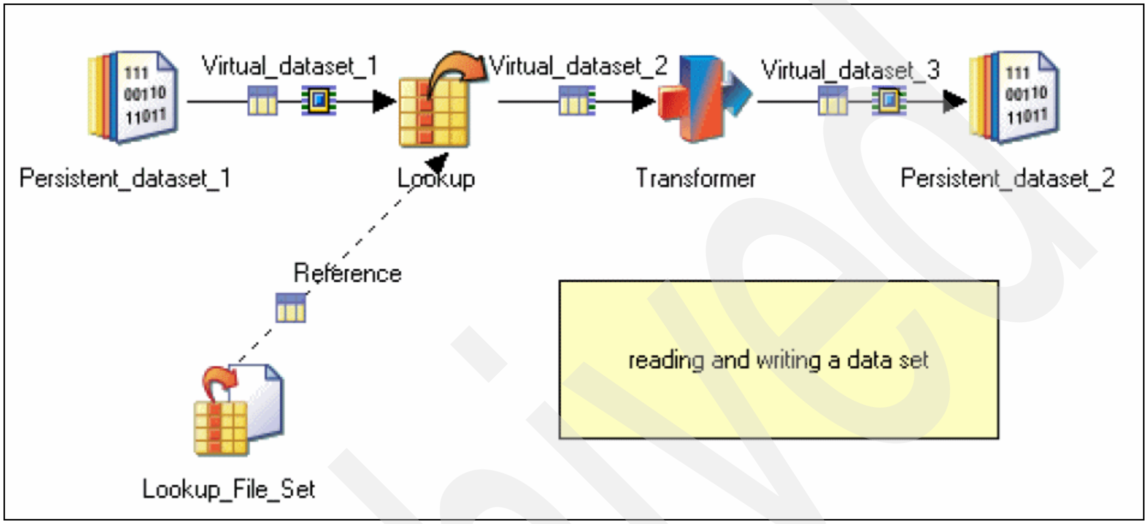


Figure 2-28 Data Set stage

Parallel jobs use data sets to manage data within a job. You can think of each link in a job as carrying a (virtual) data set. The Data Set stage allows you to store data being operated on in a persistent form, which can then be used by other IBM InfoSphere DataStage jobs. Data sets preserve the partitioning and sorting that may have been done on the data.

Data sets are operating system files, each referred to by a control file, which by convention has the suffix .ds. The control file points IBM InfoSphere DataStage to a set of other files that carry the data. The location of these data files is determined by the "resource disk" property in the configuration file used to run the job. Using data sets wisely can be key to good performance in a set of linked jobs. You can also manage data sets independently of a job using the Data Set Management utility, available from the IBM InfoSphere DataStage and QualityStage Designer or Director.

Figure 2-29 on page 62 through Figure 2-31 on page 63 show an example of a write to a Data Set stage in a job ("J04_IL_FTPEmployeeFile" on page 209 in the retail industry scenario described in "Retail industry scenario" on page 140).

The flow is as follows:

1. Figure 2-29 shows the job that extracts Employee data from the mainframe and writes it to a data set. This is described in “J04_IL_FTPEmployeeFile” on page 209 and is not repeated here. Instead, we only focus on the configuration of the Data Set stage in this job.
2. The Input stage allows you to specify details about how the Data Set stage writes data to a data set. Figure 2-30 shows the **Properties** tab in the Input page, which allows you to specify properties for the input link. These dictate how incoming data is written and to what data set.
   – The Update Policy specifies what action will be taken if the data set you are writing to already exists. We chose Overwrite to overwrite any existing data with new data.
3. We let the properties default under the **Partitioning** tab in the Input page, which allows you to specify details about how the incoming data is partitioned or collected before it is written to the data set. It also allows you to specify that the data should be sorted before being written.
4. Figure 2-31 shows the **Columns** tab in the Input page, which specifies the column definitions of the input data.
5. We let all the values default under the **Advanced** tab in the Stage page, which allows you to specify how the stage executes.

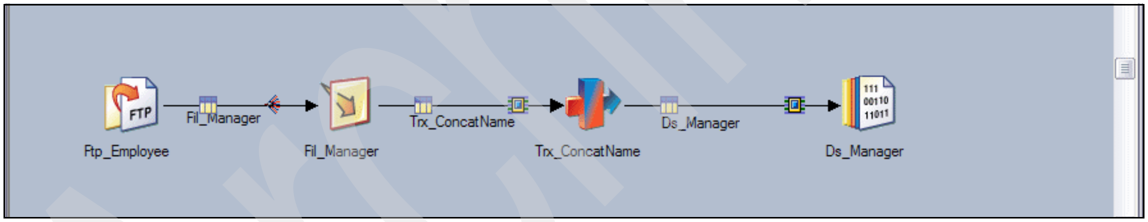


Figure 2-29 Data Set stage example 1/3

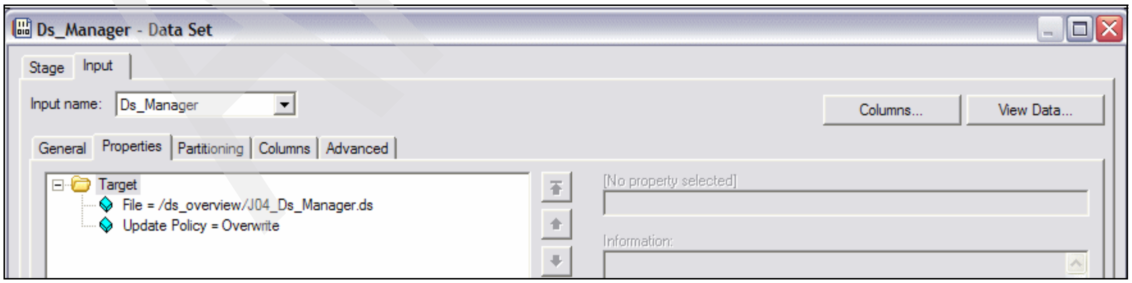


Figure 2-30 Data Set stage example 2/3

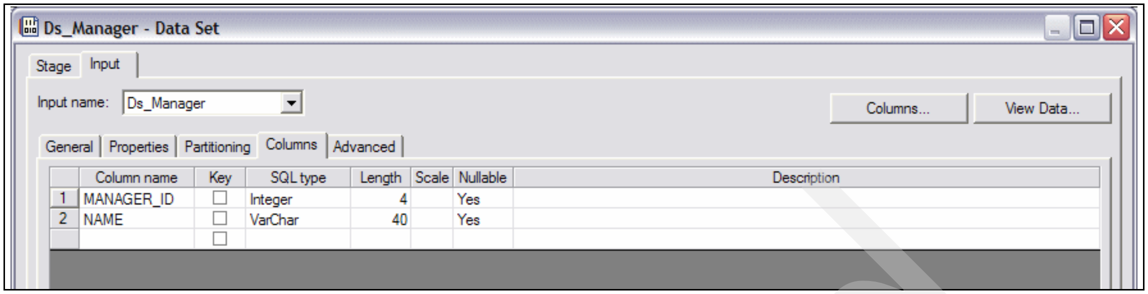


Figure 2-31 Data Set stage example 3/3

2.7 Distributed Transaction (new in Version 8.1)

The connector framework is being enhanced to provide support for distributed two-phased XA transactions in DataStage Enterprise jobs.

> **Note:** At the time of writing this Redbooks publication, DTS is only supported for DB2.

Figure 2-32 shows the main elements involved in exploiting the Distributed Transaction stage.

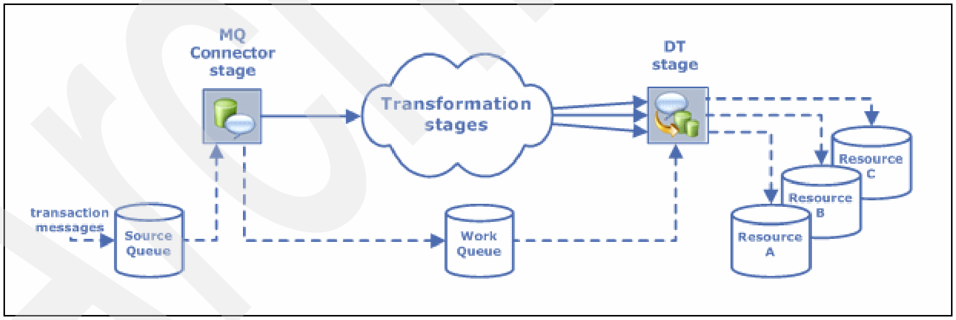


Figure 2-32 Distributed Transaction stage

The flow is as follows:

1. Transaction data is carried by IBM WebSphere MQ messages that arrive at the source queue. Each message can include multiple business transactions[2]. Multiple messages may be grouped together as a single transaction.
2. The MQ Connector stage allows you to configure transaction[3] boundaries. You can specify the number of source messages to include in each transaction, or the time interval after which a transaction boundary is set regardless of the number of source messages received till that moment.

   The MQ Connector uses a specially designated work queue as a temporary buffer storage for source messages that participate in transactions. This is the default, but it may optionally work without the work queue.
3. The retrieved messages may be processed by any number and combination of transformation stages, chosen from a rich palette of stage types provided by IBM InfoSphere DataStage.

   The processed messages from these transformation stages result in rows of data that arrive at the Distributed Transaction Stage (DTS) on one or more input links.

   Each input link on the DTS is associated with one external resource[4]. The rows on each link are sent to the designated resource (as insert, update, or delete operations on the resource).

   > **Note:** Each link can support a combination of insert, update and delete operations.

[2] A business transaction is a set of related records that are provided to IBM InfoSphere DataStage within a single IBM WebSphere MQ message. A business transaction cannot be spread across more than one message, but a single message could contain more than one business transaction. The records that comprise a business transaction are sent to DTS as individual rows of data. It is possible that a record may involve multiple rows, but for simplicity's sake, assume that one record maps to one row.

[3] A transaction corresponds to a unit-of-work and may comprise a number of messages, the actual number determined by the configuration of the MQ Connector stage. As mentioned, this number may be absolute, or may have a time-based component making it a variable number. A unit-of-work is atomic in that all of the records within a unit-of-work are processed as a single indivisible unit — either all records are written to the target, or none are.

[4] They may all be the same resource as well. In other words, we may be updating, deleting, and inserting to the same table.

DTS reads from the input link into the stage and packages into that XA transaction a delete from the work queue (or the source queue if it is configured not use a work queue). If the transaction commits successfully, the message is then removed from the work queue.

The reading of messages and writing to external resources is done in an atomic manner using two-phase XA protocol, with IBM WebSphere MQ Transaction Manager coordinating the XA global transaction.

Figure 2-33 shows a typical flow in DTS. There are two source messages that are processed as a single transaction (unit-of-work). Each of these messages contains three records.

1. The MQ Connector stage moves these two messages to an IBM WebSphere MQ work queue, and sends the data from these two messages to its output link. It then marks the end-of-wave (EOW)[5], since the MQ Connector stage is configured to emit an end-of-wave marker after every two messages.
2. The job logic (typically implemented through Column Import stages) parses out the individual records within the transactional messages, and puts a row on the input link of the DTS for each separate record. The result is a total of six rows on the input link, which is then followed by an EOW marker. The DTS will process all rows up to the end-of-wave as a single XA transaction. The messages in the IBM WebSphere MQ work queue are deleted as part of this XA transaction.

Note: The DTS also supports reading messages directly from a source queue. In this case, the MQ Connector stage will only browse the source queue (non-destructively), rather than destructively getting the source message from the queue and writing them to a work queue. In this case, since there is no work queue, the DTS will (destructively) read directly from the source queue.

[5] A key aspect of the overall architecture is the use of IBM InfoSphere DataStage's end-of-wave markers, which are used to define transactional scopes. The MQ Connector stage can emit EOW markers after it has read a given number of messages, or after a given time period has elapsed. The DTS acts upon these markers to understand the scope of a transaction.

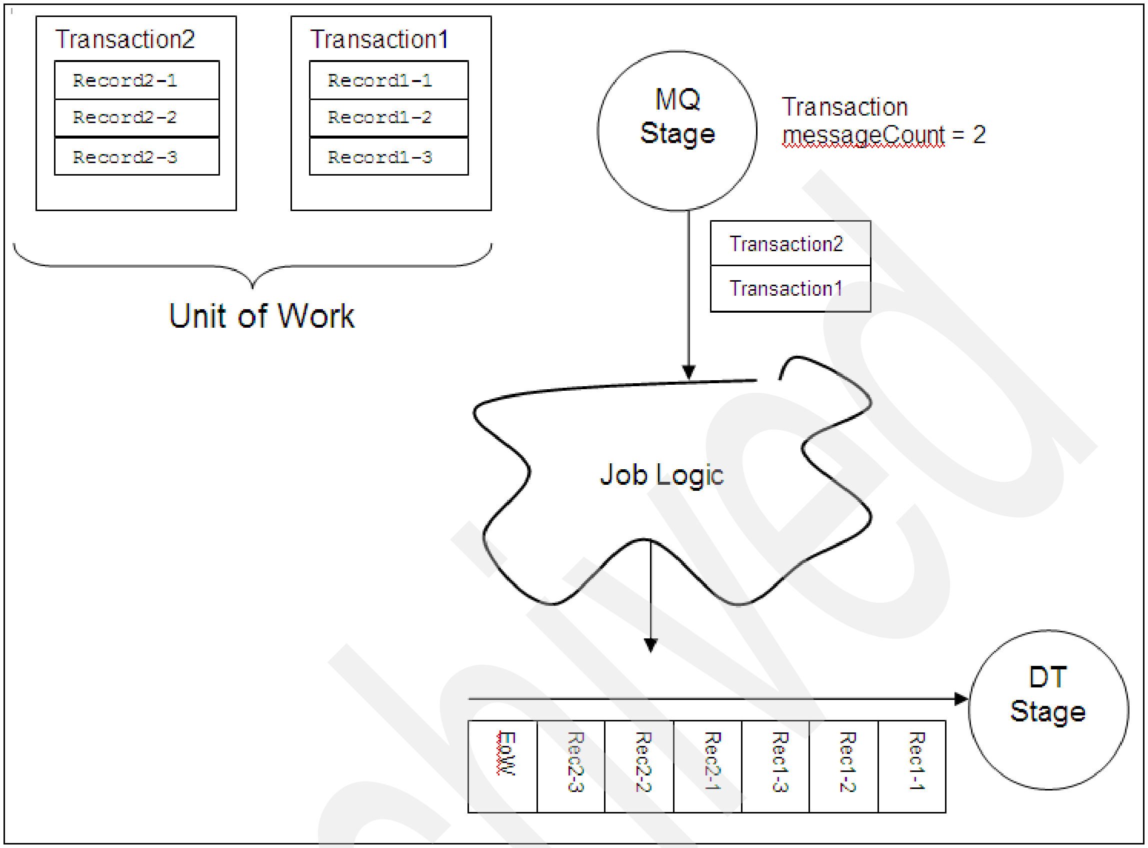


Figure 2-33 DTS flow concepts

There are many ways in which DTS jobs can be deployed. The choice of a particular topology is dependent upon the nature of the source data and how it must be processed, as follows:

- Order

  Whether the messages have to be processed in the order they were written to the source queue. If the order must be maintained, then it is not possible to execute the job in parallel, since there is no co-ordination between player processes on multiple nodes.

  Ordering is specified in the configuration of DTS as the “Order messages” parameter as shown in Figure 2-34.

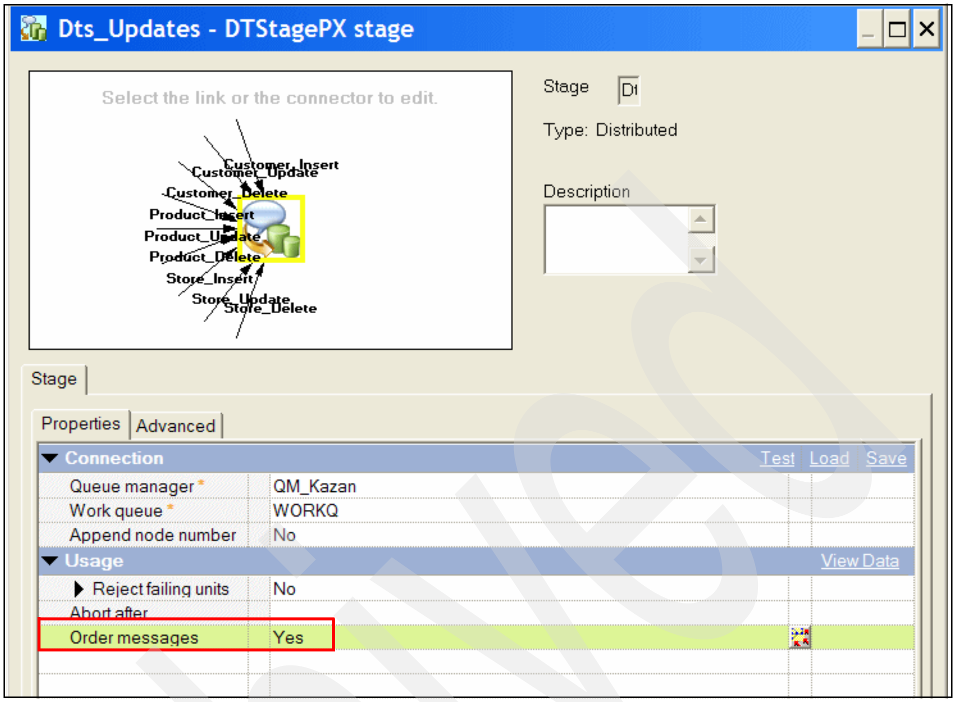


Figure 2-34 Configuring ordering in the DTS

- Relationships

  Whether or not the source messages are related — for example, they have some key field that indicates that they must be processed as a unit. A hash partitioner should then be used to ensure that all messages with a given key are processed by the same node.

The topologies possible when the following conditions apply are described here:

- No order and no relationships between messages

  Since the order of processing of source messages is not important, and there is no relationship between messages, it is possible to run the jobs fully in parallel.

  Figure 2-35 shows what such a topology would look like.

Note: The numbers under the queues represent message sequence numbers and illustrate how messages may be distributed across the queues. The letters represent hash partitioning key fields. The solid arrows show the movement of IBM WebSphere MQ messages to and from the queues. The dashed lines represent the job links. For clarity, only MQ Connector and Distributed Transaction stages are shown here, but in reality there would be other stages in between to implement the business logic of the extract, transformation, and load (ETL) job.

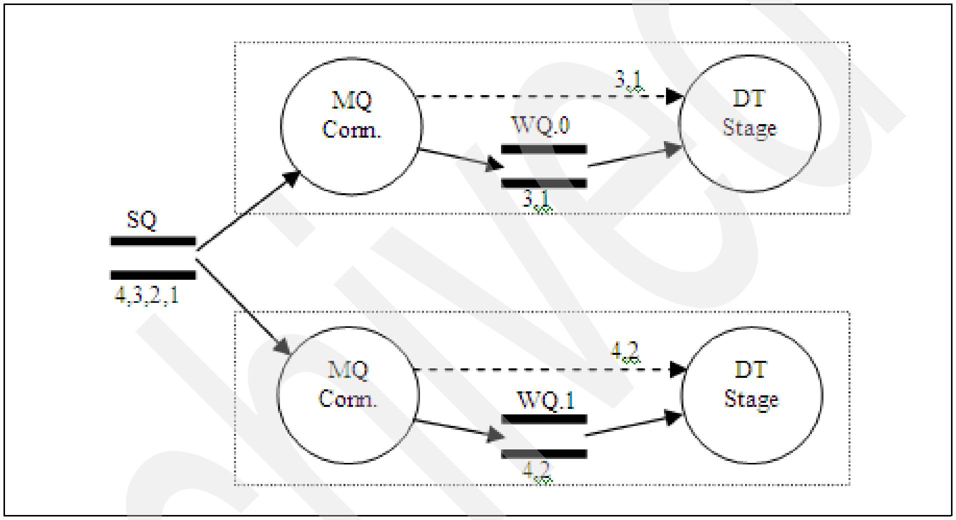


Figure 2-35 No ordering, no relationships

Each node contains an MQ Connector stage, a work queue and a DTS. The MQ Connector stages access a single source queue and distribute these to the nodes. Since the MQ Connector stage instances read the messages destructively off the source queue, there is no contention for messages.

The reason to have multiple work queues is to be able to restart jobs upon catastrophic failure. Multiple work queues also aid performance, since there is no contention for work queues, and the DTS is more likely to find its message from the head of the queue.

- No order, but relationships exist in the messages

  Since it is necessary to ensure that all messages that are related to each other by a shared key value are sent to the same node, a single MQ Connector stage combined with the use of a hash partitioner must be used if parallelism is desired.

  Figure 2-36 shows what such a topology would look like.

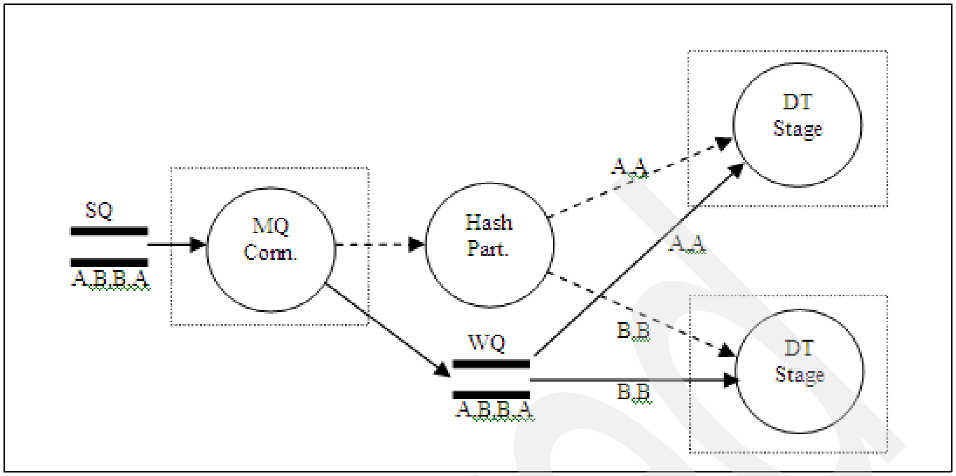


Figure 2-36 No ordering but relationships exist topology

In this scenario, there must be a single work queue, since the MQ Connector cannot determine which node will be targeted for a specific message.

- Ordering is a must

  Since the messages must be processed in the order they arrive on the source queue, it is necessary to execute the entire job sequentially. This is because there is no synchronization between nodes, and therefore distributing messages to multiple nodes cannot guarantee any ordering of messages.

  Figure 2-37 shows what such a topology would look like.

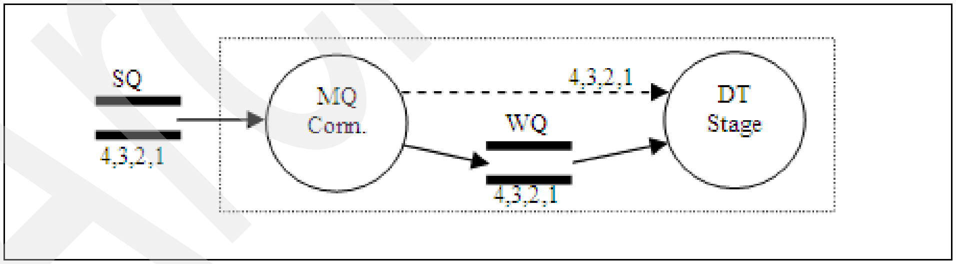


Figure 2-37 Ordering a must topology

Note: These topologies can be modified to support scenarios where work queues bypassed. This is shown in Figure 2-38 and Figure 2-39. By omitting the necessity to write to a work queue, overall performance could possibly be improved.

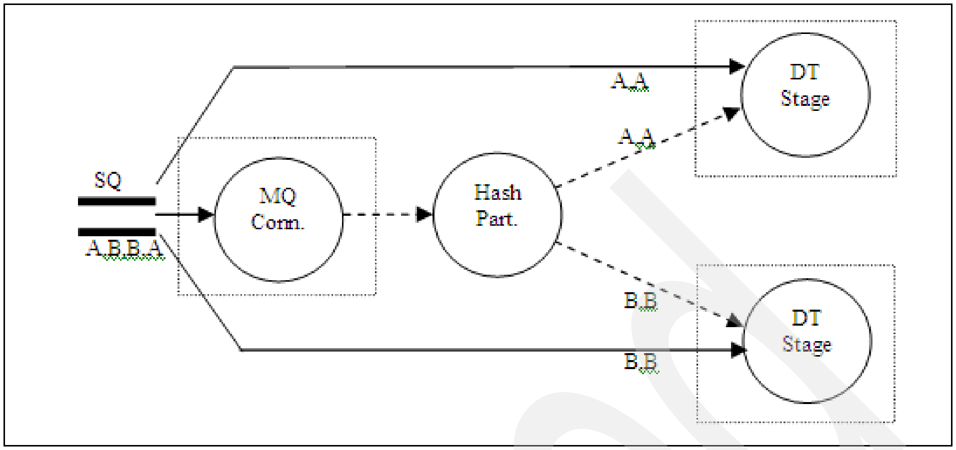


Figure 2-38 No ordering (with no work queue) topology

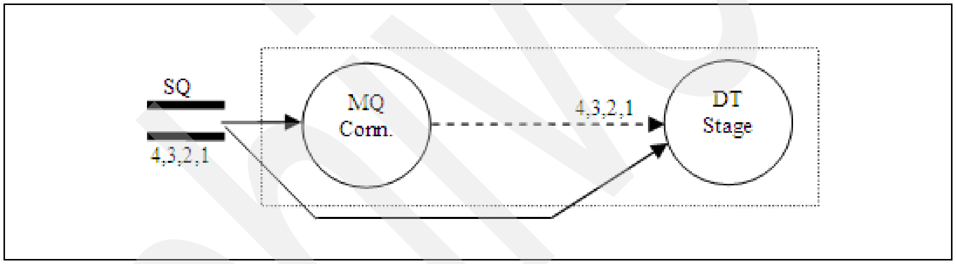


Figure 2-39 Ordering (with no work queue) topology

Figure 2-40 on page 72 through Figure 2-55 on page 85 show an example of a write to a Distributed Transaction stage in a job ("J13_Daily_UpdateLookupDim (Day 1)" on page 356 in the retail industry scenario described in "Retail industry scenario" on page 140).

The flow is as follows:

1. Figure 2-40 on page 72 shows the job that processes changes to attributes of dimension tables arriving via a IBM WebSphere MQ queue. This is described in "J13_Daily_UpdateLookupDim (Day 1)" on page 356 and is not repeated here. Instead, we only focus on the configuration of the Distributed Transaction stage in this job.

2. Figure 2-41 on page 72 shows the **Properties** tab in the Stage page, which allows you to specify connection details (Queue Manager QM_Kazan and Work queue WORKQ) and usage details such as whether the messages across all the input links should be processed in order[6] (Order messages Yes). It shows the nine input links — three (insert, update and delete) for each target (Customer, Product and Store).
3. Figure 2-42 on page 73 through Figure 2-54 on page 84 show the configuration of the Customer_Insert, Customer_Update, and Customer_Delete input links, as follows:
   – Figure 2-42 on page 73 shows the **Properties** tab for the Customer_Insert link that identifies the connection to the target database DSSAMPLE, and the Write mode (Insert). Rather than let the stage generate the insert SQL, the Generate SQL property is set to No to indicate that the SQL will be provided manually (partially seen here).

     The order of processing of the input links is specified under the **Link Ordering** tab as shown in Figure 2-43 on page 74. It is essential to order the input links correctly to ensure parent-child relationships are properly coordinated. Finally, when a link is selected, the 'Connector' drop-down list provides a way to select the target connector for the stage such as DB2 or WebSphere MQ.

     Figure 2-44 on page 75 shows the input column definitions under the **Columns** tab for this link.

     Figure 2-45 on page 76 shows the properties of the **Partitioning** tab, which allows you to specify details about how the incoming data is partitioned or collected on this input link before being processed by the stage. It shows a Hash Partition type and a sort being requested on the DTS_String_Timestamp column (in the Selected pane).
   – Figure 2-46 on page 77 through Figure 2-51 on page 81 are similar to the configuration of the Configuration_Insert link and show the configuration of the Customer_Update link. The manually provided Update SQL statement is shown in Figure 2-48 on page 78.
   – Figure 2-52 on page 82 through Figure 2-55 on page 85 show the corresponding configuration of the Customer_Delete link.

     Note: Similar configurations must be defined for the other two dimension tables Product and Store — not shown here.

[6] We have to process the updates to the various dimension tables in the order in which they occurred in the originating OLTP system to maintain correct versioning. Therefore, this parameter (also called cross link ordering) must be set to Yes.

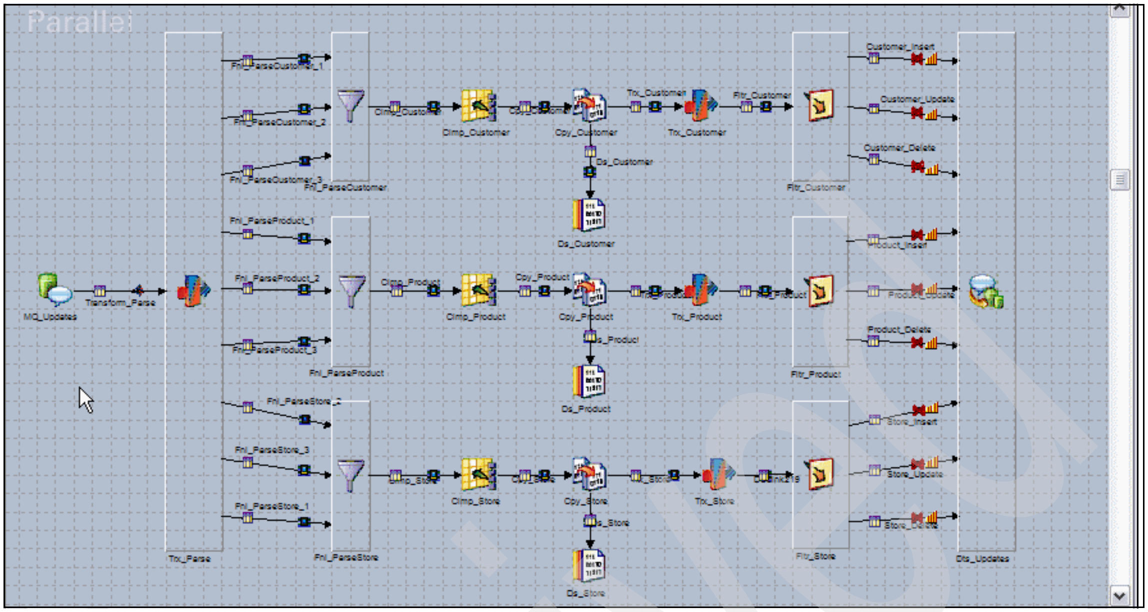


Figure 2-40   DTS example 1/16

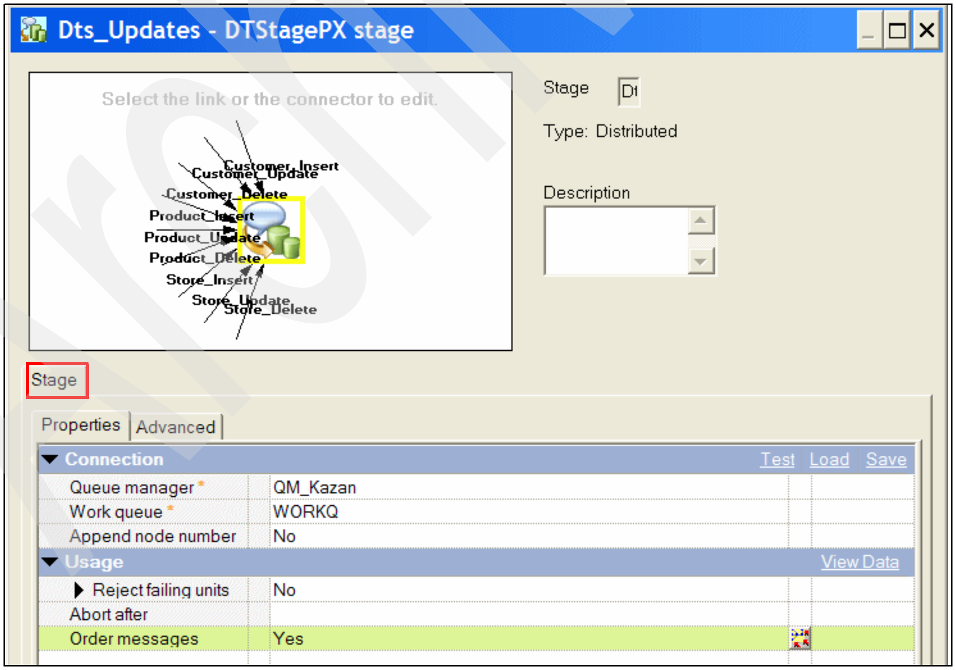


Figure 2-41   DTS example 2/16

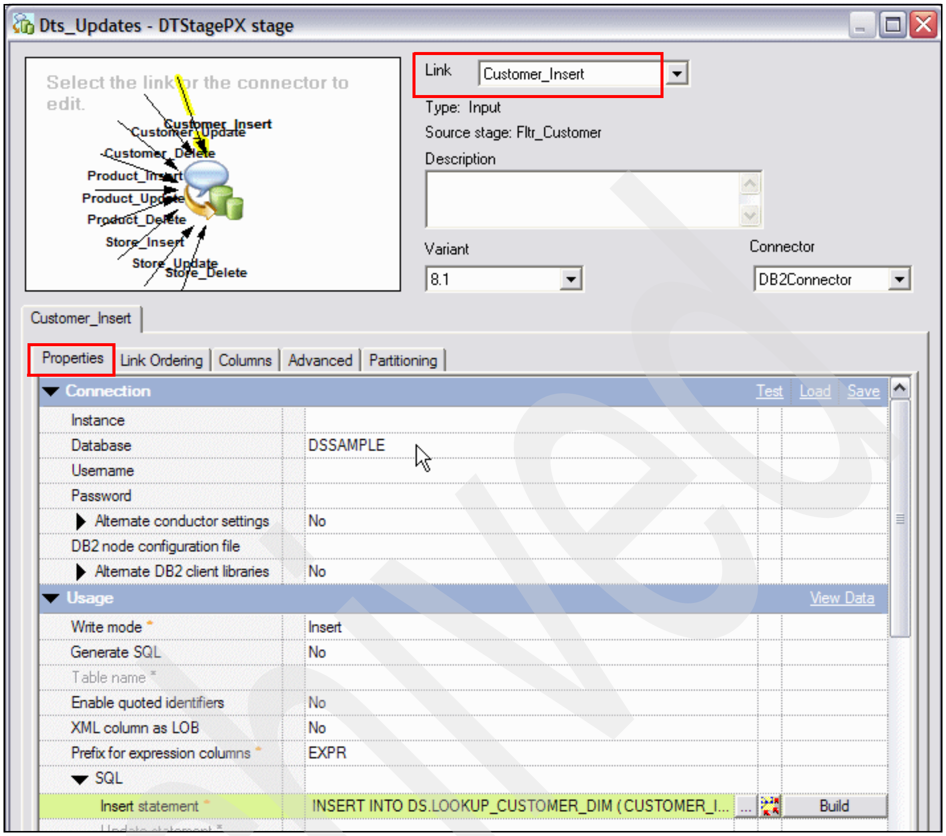


Figure 2-42 DTS example 3/16

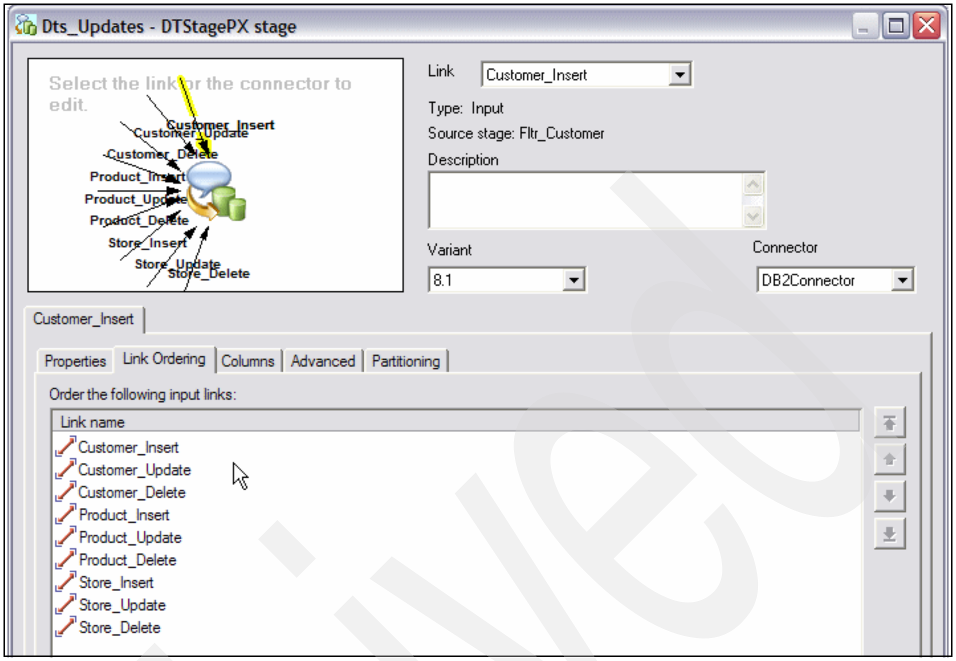


Figure 2-43 DTS example 4/16

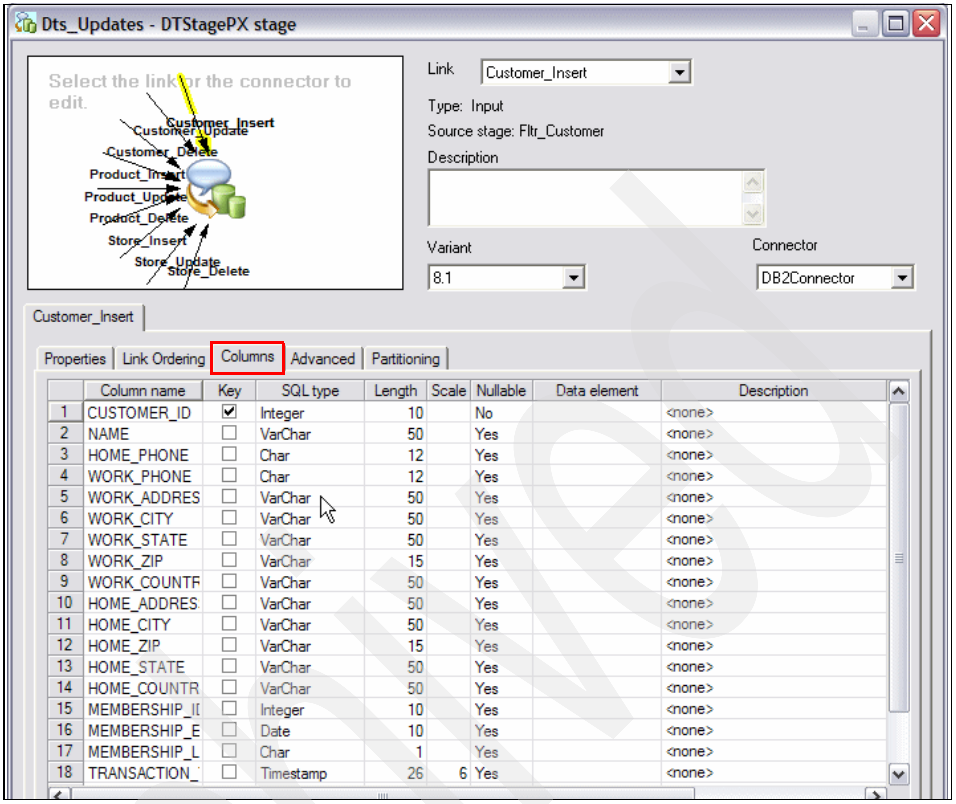


Figure 2-44 DTS example 5/16

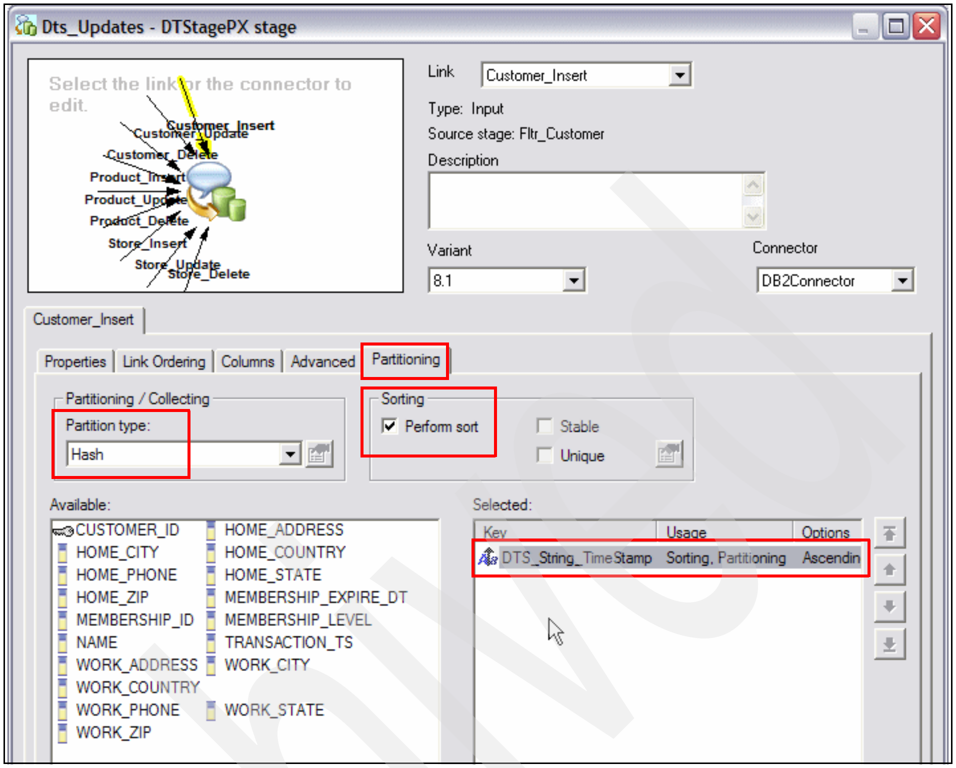


Figure 2-45 DTS example 6/16

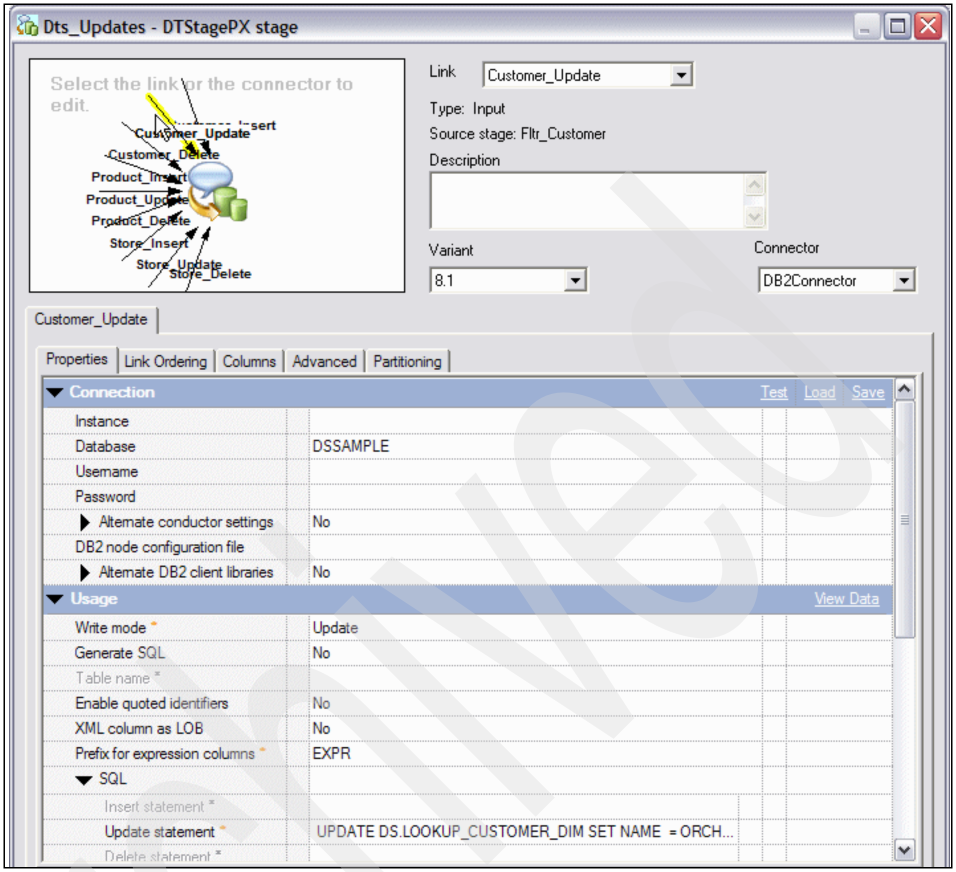


Figure 2-46 DTS example 7/16

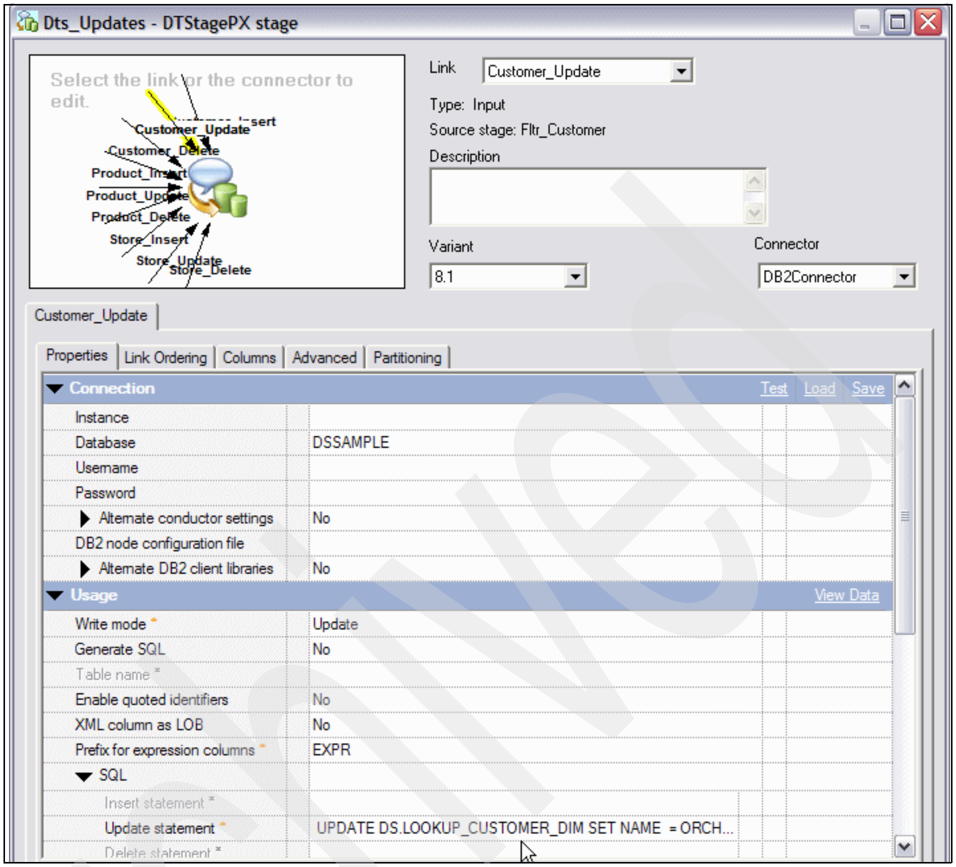


Figure 2-47 DTS example 8/16

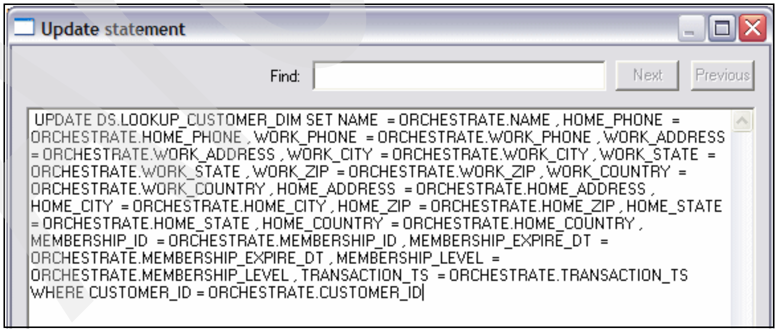


Figure 2-48 DTS example 9/16

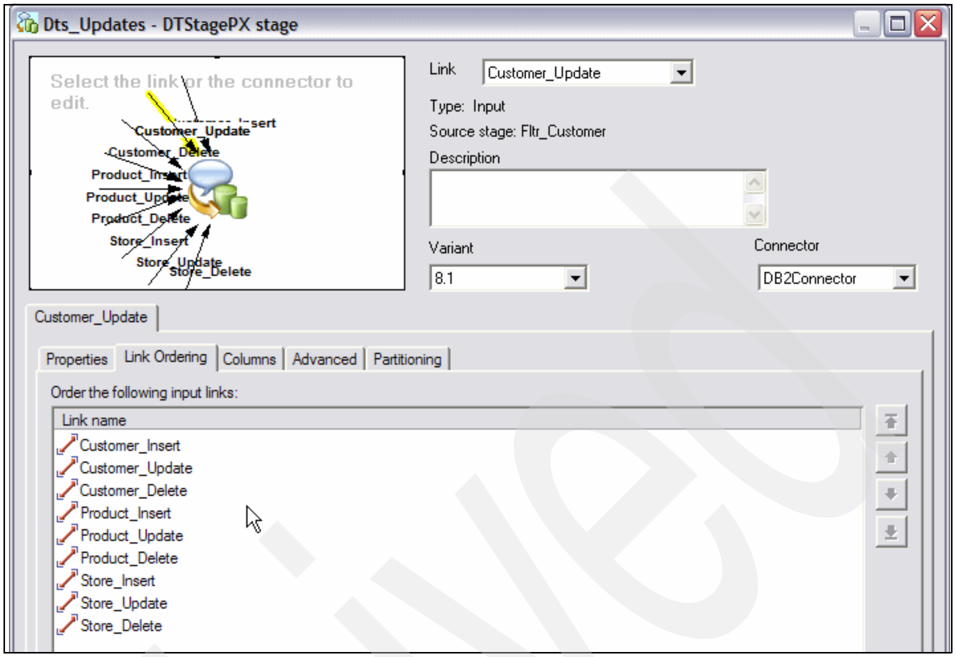


Figure 2-49 DTS example 10/16

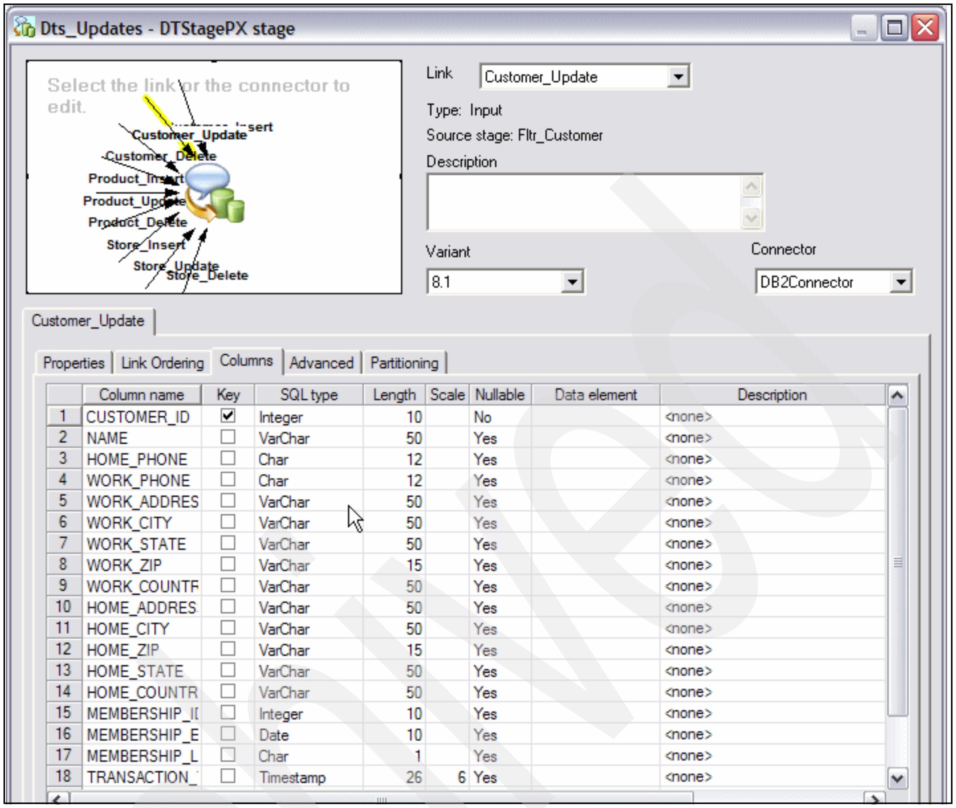


Figure 2-50 DTS example 11/16

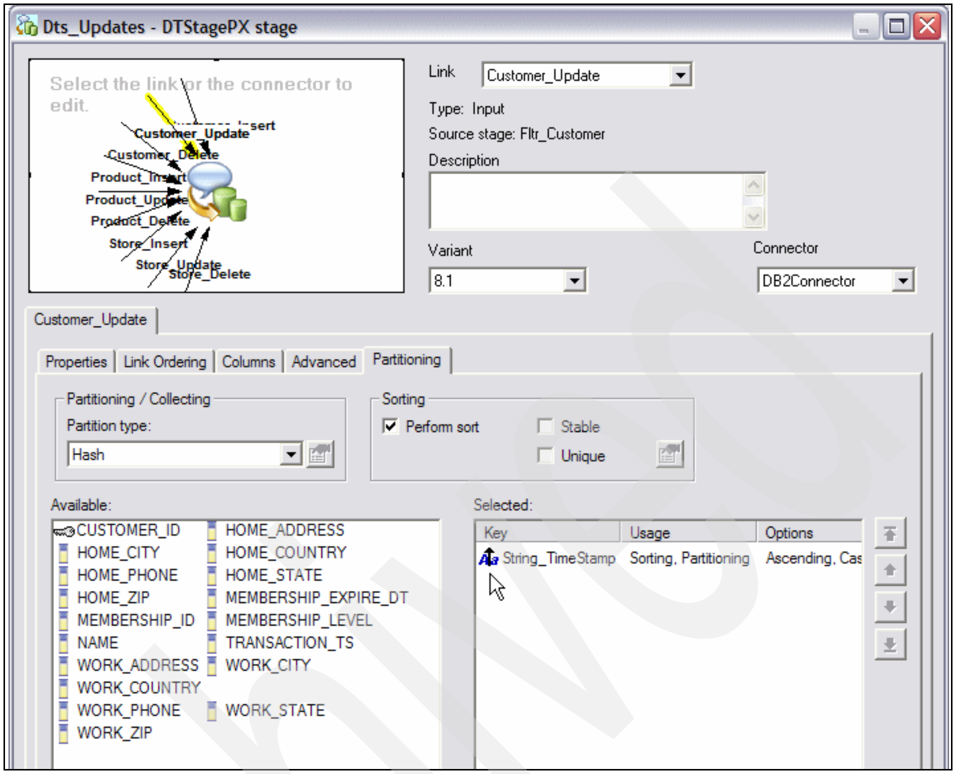


Figure 2-51 DTS example 12/16

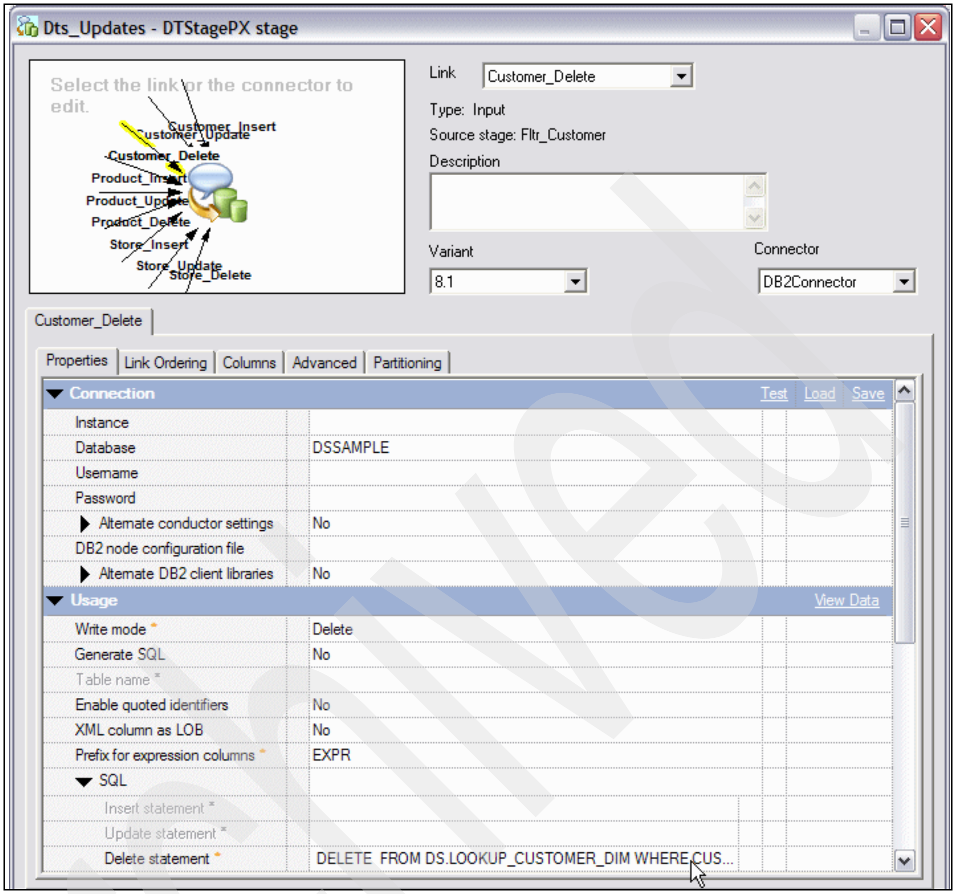


Figure 2-52 DTS example 13/16

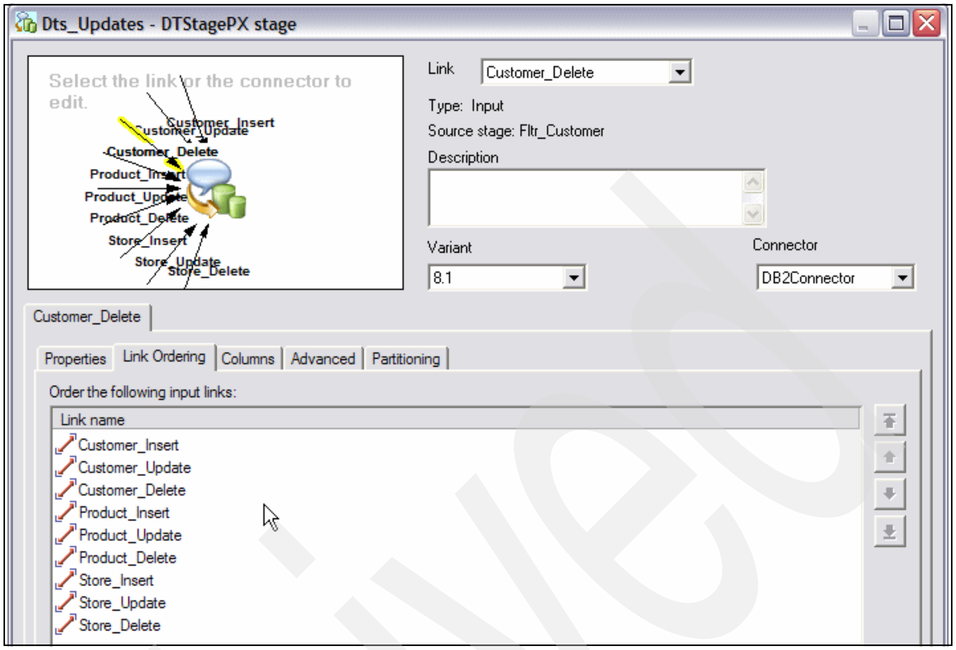


Figure 2-53 DTS example 14/16

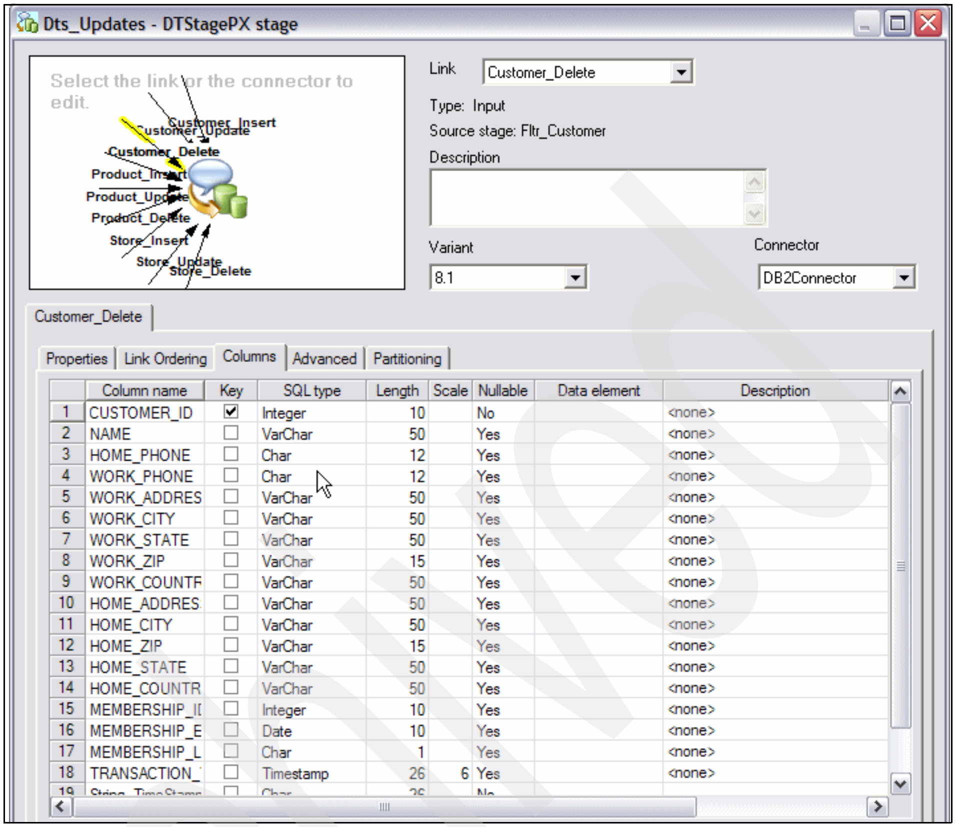


Figure 2-54 DTS example 15/16

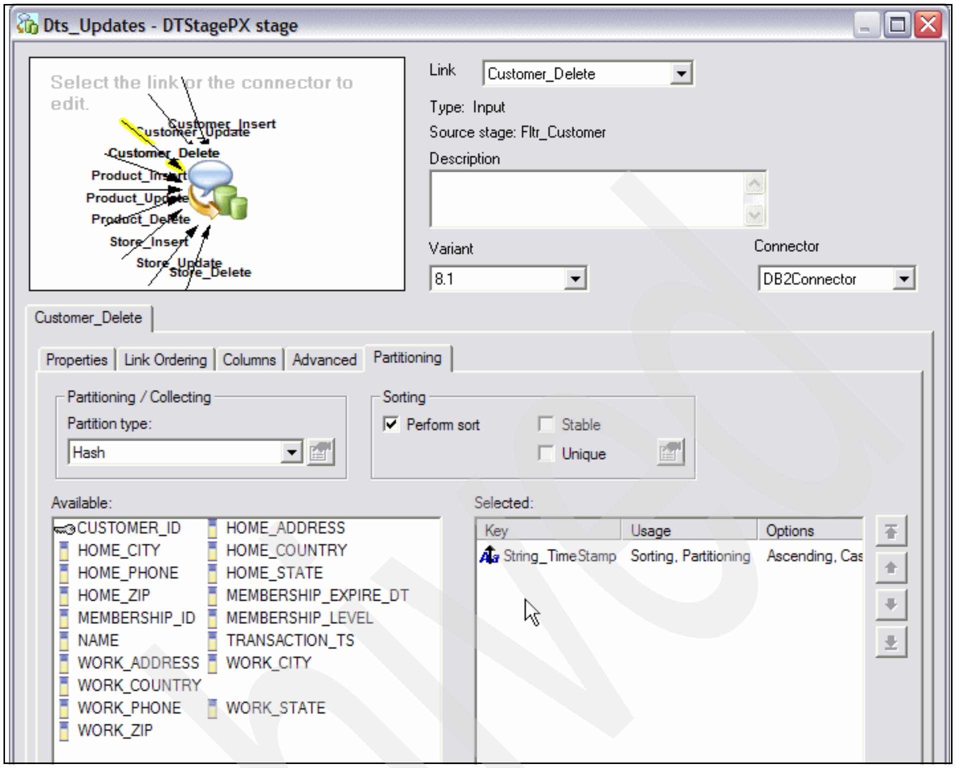


Figure 2-55 DTS example 16/16

2.8 FTP Enterprise

The FTP Enterprise stage transfers multiple files in parallel, as well as a single file. These are sets of files that are transferred from one or more FTP servers into IBM InfoSphere DataStage or from IBM InfoSphere DataStage to one or more FTP servers, as shown in Figure 2-56.

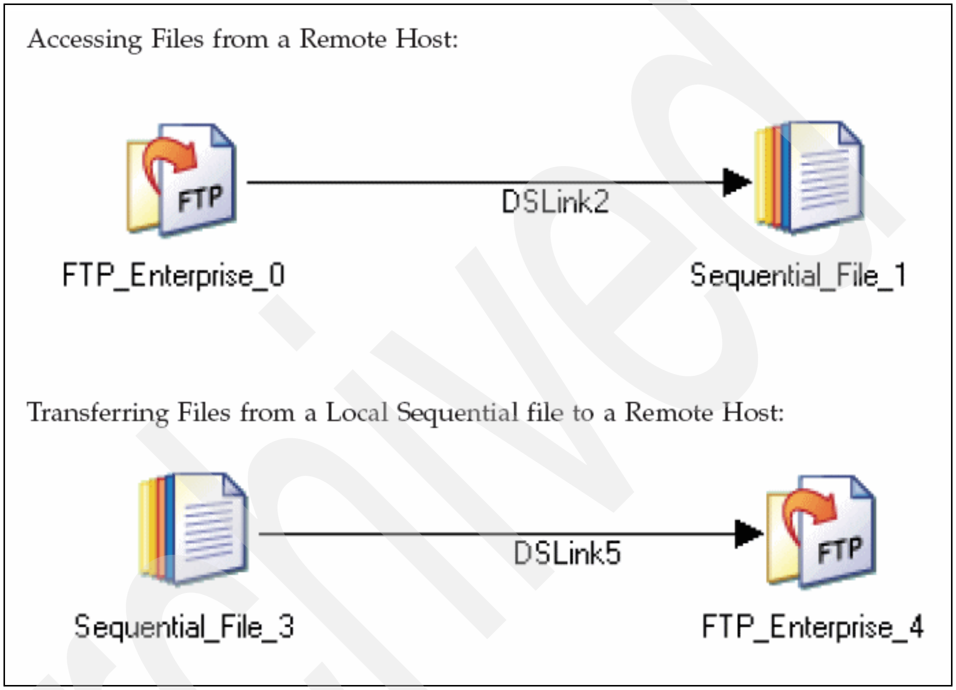


Figure 2-56 FTP Enterprise stage

The source or target for the file is identified by a URI (Universal Resource Identifier). The FTP Enterprise stage invokes an FTP client program and transfers files to or from a remote host using the FTP Protocol.

Figure 2-57 on page 87 through Figure 2-59 on page 88 show an example of the configuration of an FTP Enterprise stage in a job ("J01_IL_FTPCustomerFile" on page 159 in the retail industry scenario described in "Retail industry scenario" on page 140), as follows:

1. Figure 2-57 on page 87 shows the job that transfers a file from the mainframe to a sequential file. This is described in "J01_IL_FTPCustomerFile" on page 159 and is not repeated here. Instead, we only focus on the configuration of the FTP Enterprise stage in this job.

2. The Output page allows you to specify details about how the FTP Enterprise stage transfers one or more files from a remote host using the FTP protocol. Figure 2-58 on page 88 shows the **Properties** tab in the Output page, which allows you to specify properties that determine what the stage actually does. The available properties are displayed in a tree structure. They are divided into categories to help you find your way around them:
   - The Source category property URI specifies the pathname connecting the Stage to a source file on a remote host, which corresponds to the Customer file on the mainframe.
   - The Connection category allows you to specify the User name (nalur1) and Password to access the data source identified by the URI.
   - The Transfer Protocol category Transfer Mode property is FTP.
   - The Options category Transfer Type is Binary.
3. Figure 2-59 on page 88 shows the **Columns** tab in the Output page which identifies a single column definition for this file named Body of VARCHAR (255). Runtime column propagation is not enabled here.

Note: Format tab is used in the same way as in the Column Import or Sequential File stage to add context to how the data can be understood on receipt. This is not shown here.

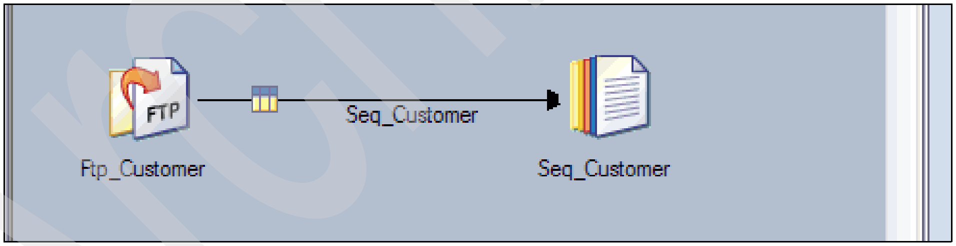


Figure 2-57 FTP Enterprise stage example 1/3

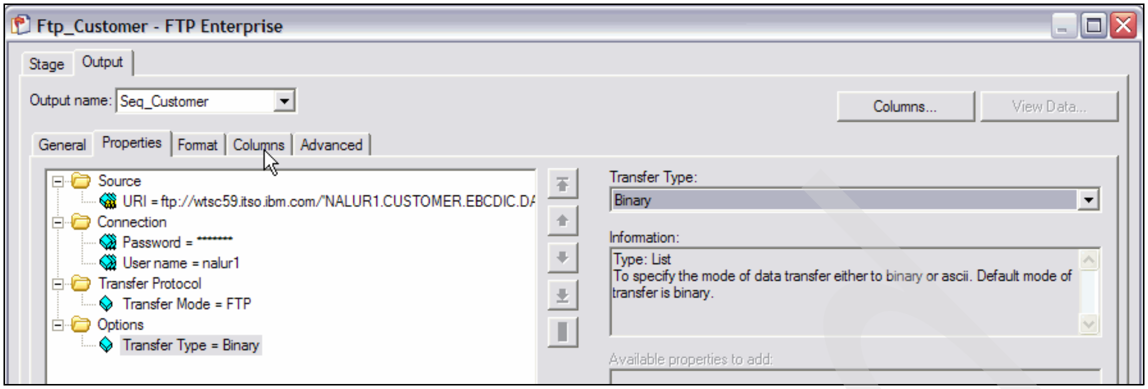


Figure 2-58   FTP Enterprise stage example 2/3

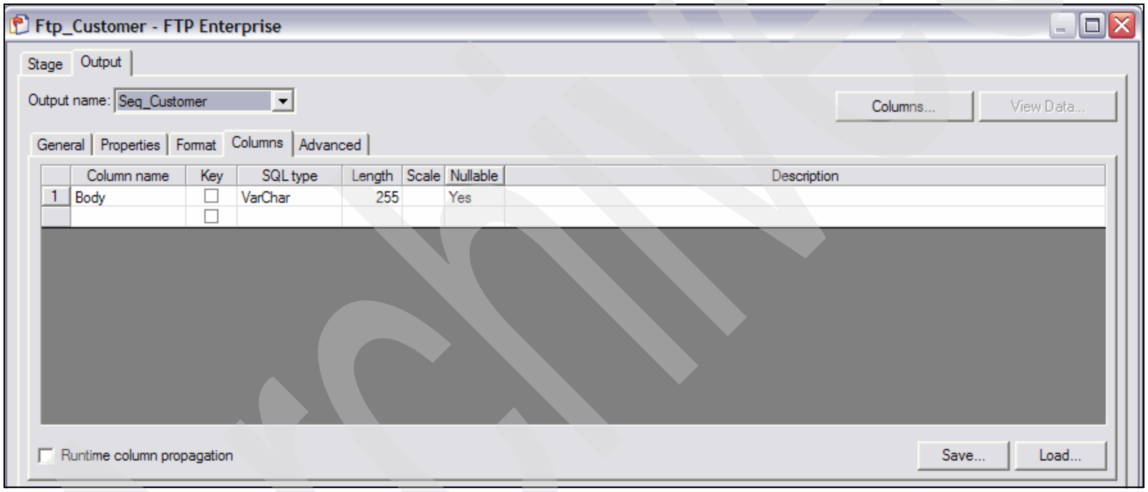


Figure 2-59   FTP Enterprise stage example 3/3

2.9 Funnel

The Funnel stage is a processing stage. It copies multiple input data sets to a single output data set. This operation is useful for combining separate data sets into a single large data set. The stage can have any number of input links and a single output link, as shown in Figure 2-60.

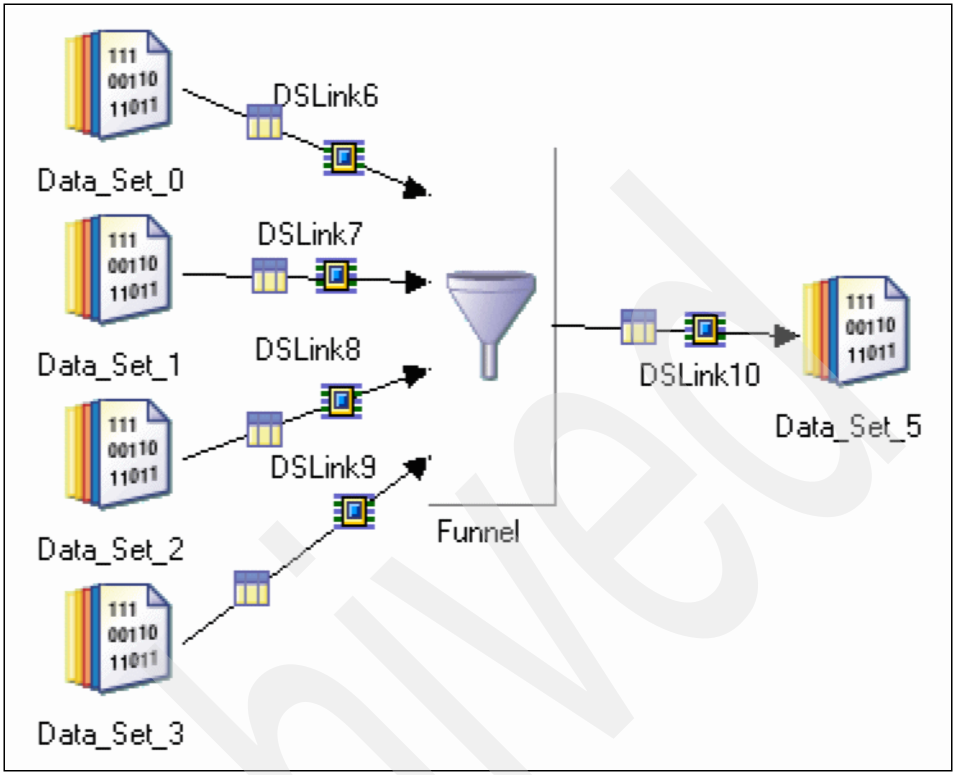


Figure 2-60 Funnel stage

The Funnel stage can operate in one of three modes:

- Continuous Funnel combines the records of the input data in no guaranteed order. It takes one record from each input link in turn. If data is not available on an input link, the stage skips to the next link rather than waiting.
- Sort Funnel combines the input records in the order defined by the value(s) of one or more key columns, and the order of the output records is determined by these sorting keys.

  The sort funnel method has some particular requirements about its input data. All input data sets must be sorted by the same key columns as will be used by the Funnel operation. Typically all input data sets for a sort funnel operation are hash-partitioned before they are sorted — choosing the auto partitioning method will ensure that this is done.

  Hash partitioning guarantees that all records with the same key column values are located in the same partition and so are processed on the same node. If sorting and partitioning are carried out on separate stages before the Funnel stage, this partitioning must be preserved.

The sortfunnel operation allows you to set one primary key and multiple secondary keys. The Funnel stage first examines the primary key in each input record. For multiple records with the same primary key value, it then examines secondary keys to determine the order of records it will output.

- Sequence copies all records from the first input data set to the output data set, then all the records from the second input data set, and so on.

For all methods, the metadata of all input data sets should be identical — mismatched columns are automatically dropped.

Figure 2-61 on page 91 through Figure 2-65 on page 92 show an example of the configuration of a Funnel stage in a job (“J14_Daily_CreateAllSalesStoreDS (Day 1)” on page 385 in the retail industry scenario described in “Retail industry scenario” on page 140), as follows:

1. Figure 2-61 on page 91 shows the job that collects all the sales transactions from three stores into a single Data Set. This is described in “J14_Daily_CreateAllSalesStoreDS (Day 1)” on page 385 and is not repeated here. Instead, we only focus on the configuration of the FTP Enterprise stage in this job.
2. Figure 2-62 on page 91 shows the **Properties** tab in the Stage page, which allows you to specify properties that determine what the stage actually does. We let the Options category property Funnel Type default to Continuous Funnel.
3. We let all the properties default under the **Advanced** tab in the Stage page.
4. We let the properties default under the **Partitioning** tab in the Input page, which allows you to specify details about how the incoming data is partitioned or collected.
5. Figure 2-63 on page 91 shows the **Columns** tab in the Input page, which identifies the input column definitions.
6. The Output page allows you to specify details about data output from the Funnel stage. Figure 2-64 on page 92 shows the **Mapping** tab in the Output page, which allows you to specify how the output columns are derived, that is, what input columns map onto them or how they are generated. In this case we defined a one-to-one mapping between the input and output columns.
7. Figure 2-65 on page 92 shows the **Columns** tab in the Output page, which identifies the output column definitions mapped earlier. Runtime column propagation is not enabled here.

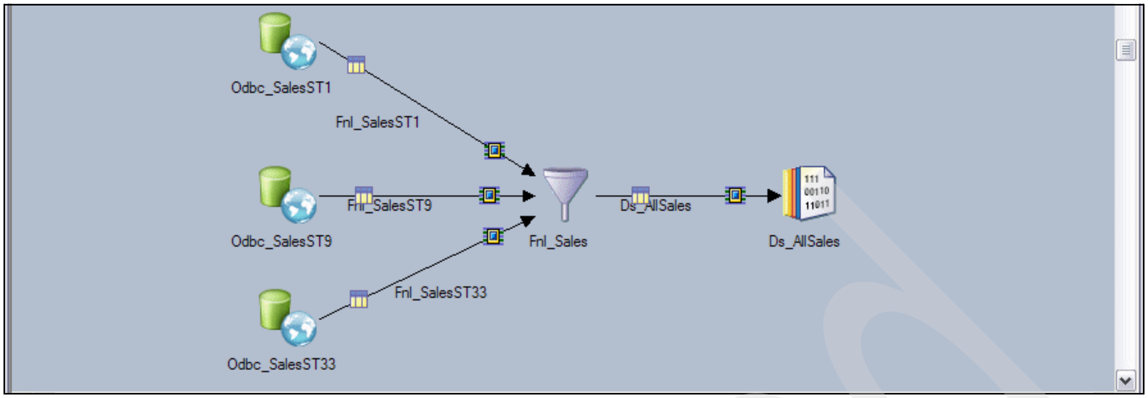


Figure 2-61 Funnel stage example 1/5

Figure 2-62 Funnel stage example 2/5

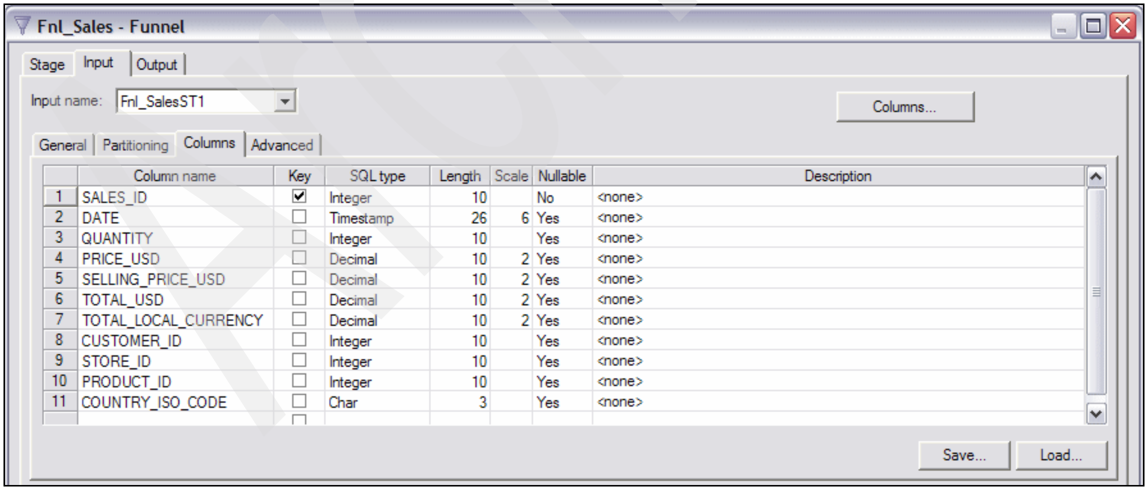


Figure 2-63 Funnel stage example 3/5

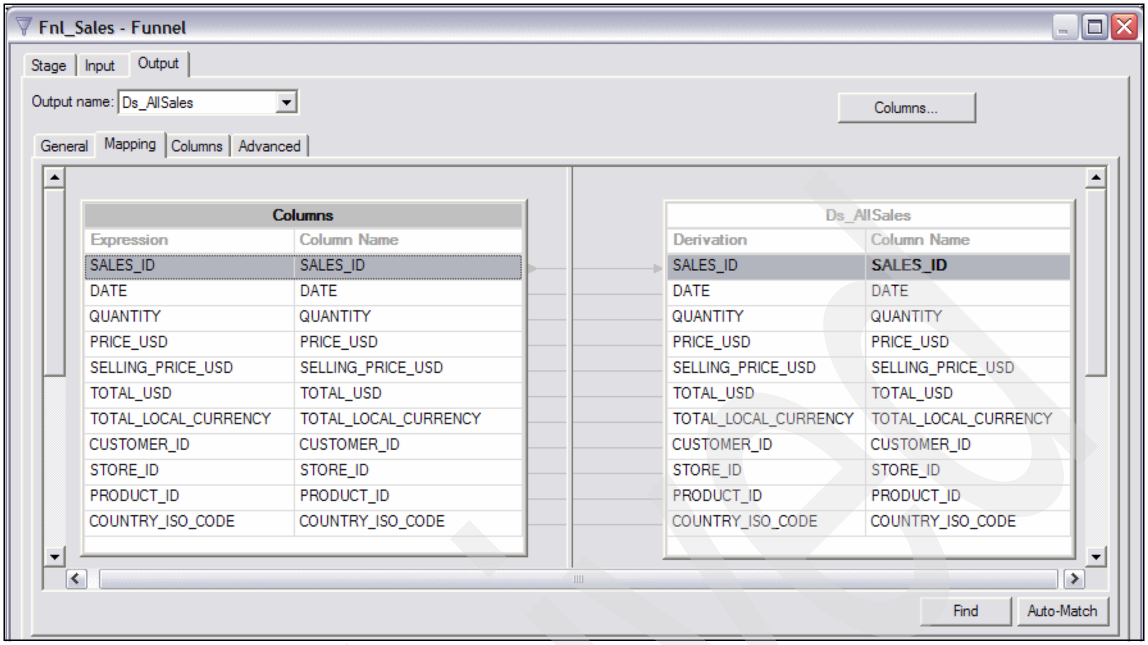


Figure 2-64   Funnel stage example 4/5

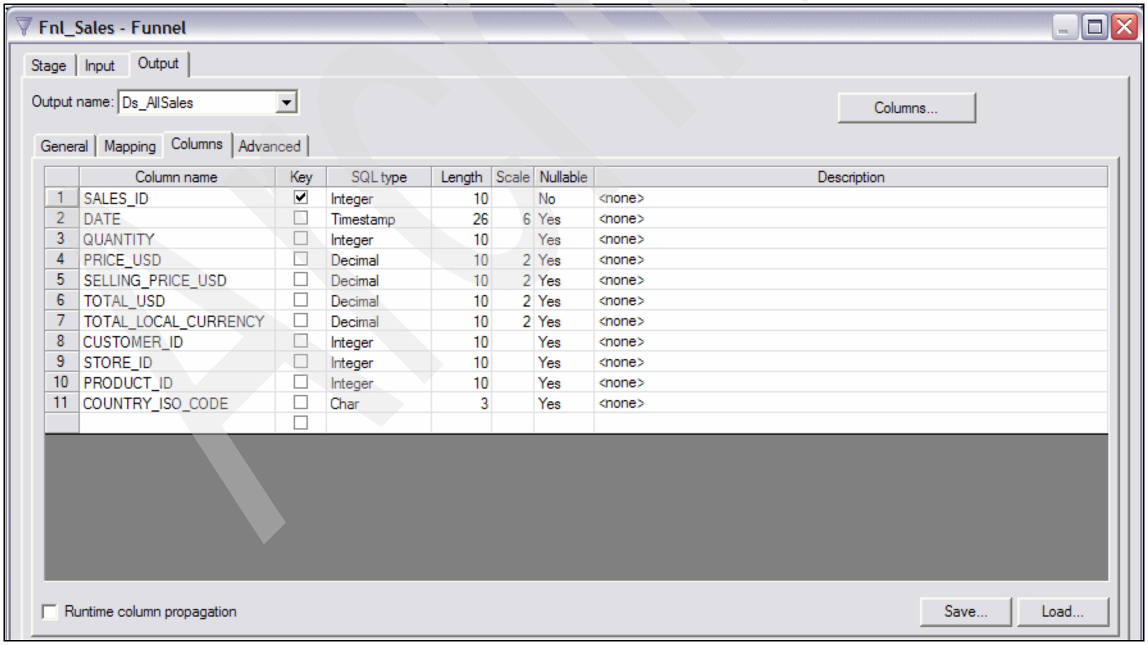


Figure 2-65   Funnel stage example 5/5

2.10 Join

The Join stage is a processing stage. It performs join operations on two or more inputs to the stage and then outputs the resulting data set, as shown in Figure 2-66.

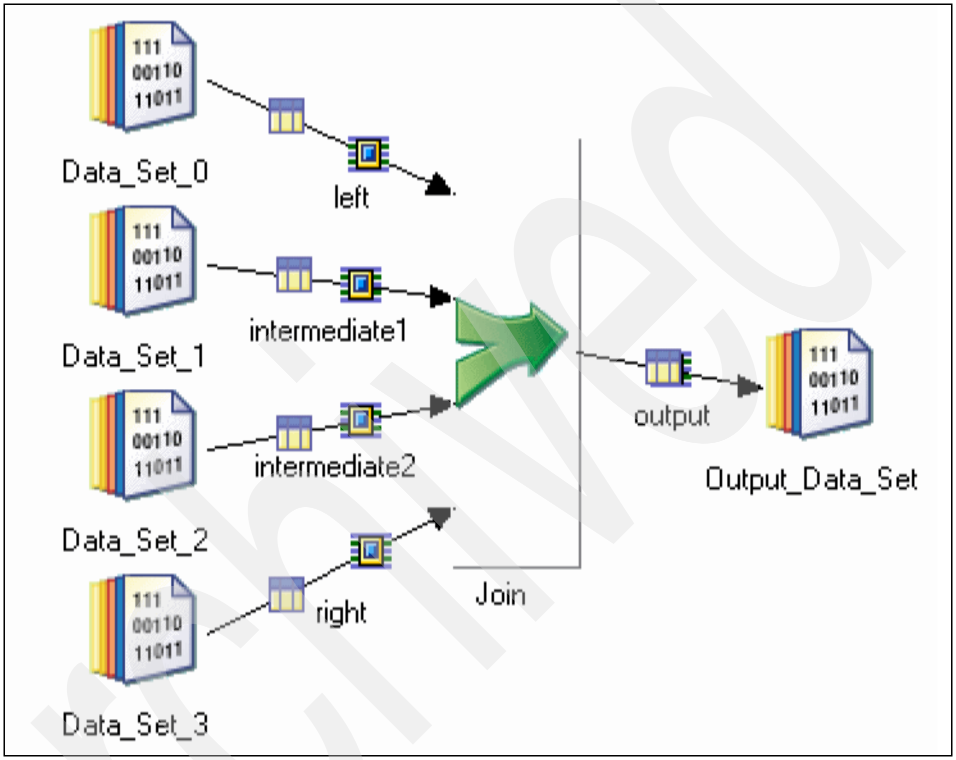


Figure 2-66 Join stage

The Join stage is one of three stages that join tables based on the values of key columns. The other two are the Lookup stage described in 2.11, “Lookup” on page 99 and the Merge stage described in 2.12, “Merge” on page 107.

The three stages differ mainly in the memory they use, the treatment of rows with unmatched keys, and their requirements for data being input (for example, whether it is sorted).

In the Join stage, the input data sets are notionally identified as the “right” set and the “left” set, and “intermediate” sets. You can specify which is which. It has any number (other than 1) of input links and a single output link.

The stage can perform one of four join operations:

- "Inner" transfers records from input data sets whose key columns contain equal values to the output data set. Records whose key columns do not contain equal values are dropped.
- "Left outer" transfers all values from the left data set but transfers values from the right data set and intermediate data sets only where key columns match. The stage drops the key column from the right and intermediate data sets.
- "Right outer" transfers all values from the right data set and transfers values from the left data set and intermediate data sets only where key columns match. The stage drops the key column from the left and intermediate data sets.
- "Full outer" transfers records in which the contents of the key columns are equal from the left and right input data sets to the output data set. It also transfers records whose key columns contain unequal values from both input data sets to the output data set. (Full outer joins do not support more than two input links.)

The data sets input to the Join stage must be key partitioned and sorted. This ensures that rows with the same key column values are located in the same partition and will be processed by the same node. It also minimizes memory requirements because fewer rows have to be in memory at any one time. Choosing the auto partitioning method will ensure that partitioning and sorting is done. If sorting and partitioning are carried out on separate stages before the Join stage, IBM InfoSphere DataStage in auto mode will detect this and not re-partition — alternatively you could explicitly specify the "Same" partitioning method.

Figure 2-67 on page 95 through Figure 2-74 on page 99 show an example of the configuration of a Join stage in a job ("J15_Daily_CreateSalesAggDS (Day 1)" on page 387 in the retail industry scenario described in "Retail industry scenario" on page 140), as follows:

1. Figure 2-67 on page 95 shows the job that prepares the consolidated sales transactions for input to the SCD stage by appending the dimension attributes to each row. This is described in "J15_Daily_CreateSalesAggDS (Day 1)" on page 387 and is not repeated here. Instead, we only focus on the configuration of the Join stage in this job.
2. Figure 2-68 on page 96 shows the **Properties** tab in the Stage page, which allows you to specify properties that determine what the stage actually does.
   - The Join Keys category property Key identifies the join column (CUSTOMER_ID).
   - The Options category property Join Type specifies a Left Outer join.

3. We let all the properties default under the **Advanced** tab in the Stage page.
4. Figure 2-69 on page 96 shows the **Link Ordering** tab in the Stage page, which allows you to specify which input link is regarded as the left link and which link is regarded as the right link, and which links are regarded as intermediate. You can use this tab to reorder the links as required.
5. Figure 2-70 on page 96 shows the **Partitioning** tab in the Input page, which allows you to specify details about how the incoming data is partitioned or collected. We chose the Hash Partition Type and sorted the input on the CUSTOMER_ID column.
6. Figure 2-71 on page 97 shows the **Columns** tab in the Input page, which identifies the input column definitions of the Joi_CustomerDim link.
7. The Output page allows you to specify details about data output from the Join stage. Figure 2-72 on page 97 and Figure 2-73 on page 98 show the **Mapping** tab in the Output page, which allows you to specify how the output columns are derived, that is, what input columns map onto them or how they are generated. In this case we defined a one-to-one mapping between the input and output columns.
8. Figure 2-74 on page 99 shows the **Columns** tab in the Output page, which identifies the output column definitions mapped earlier. Runtime column propagation is not enabled here.

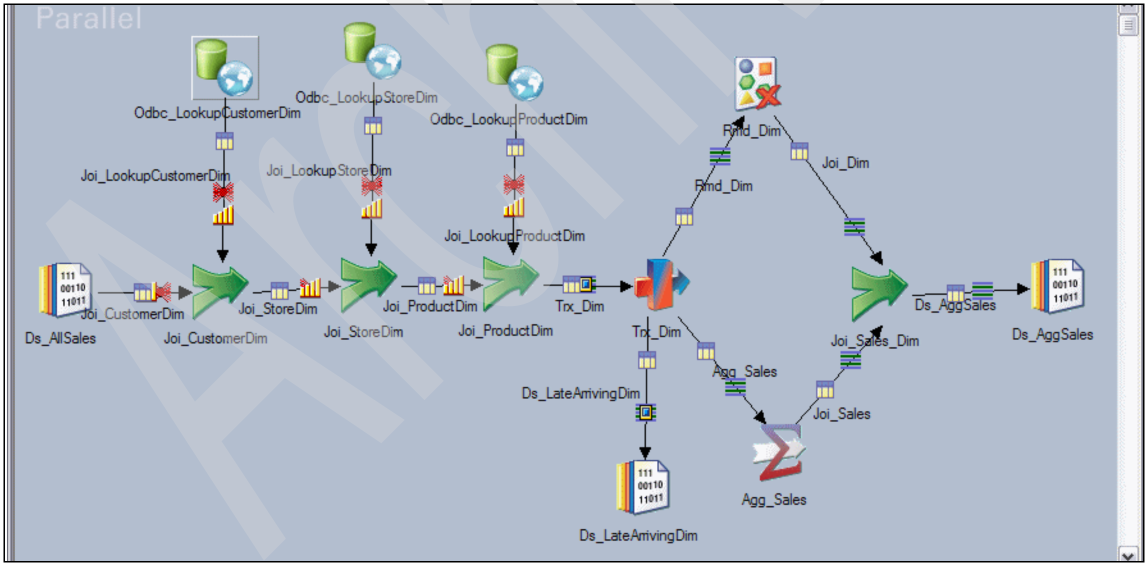


Figure 2-67 Join stage example 1/8

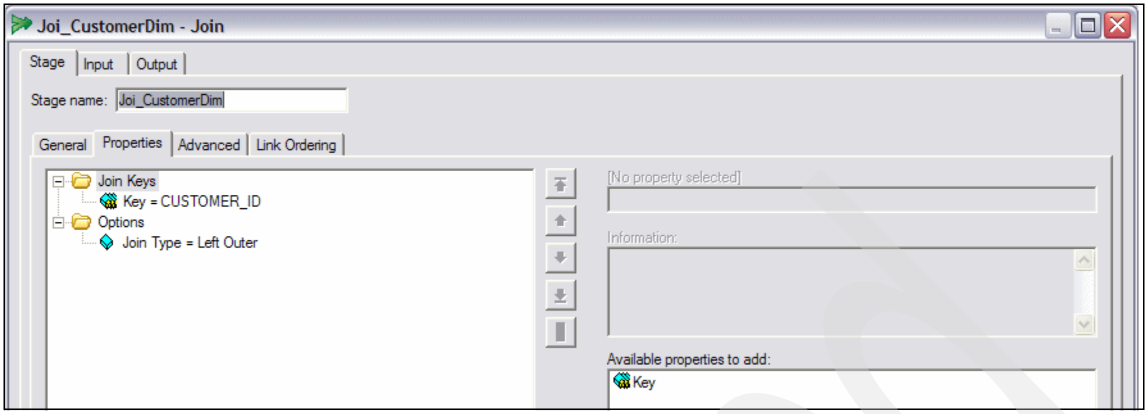


Figure 2-68 Join stage example 2/8

Figure 2-69 Join stage example 3/8

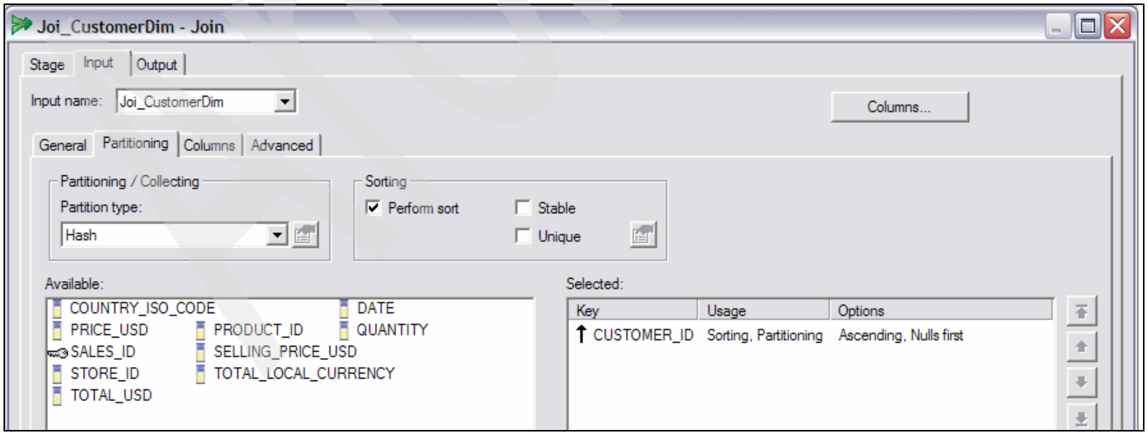


Figure 2-70 Join stage example 4/8

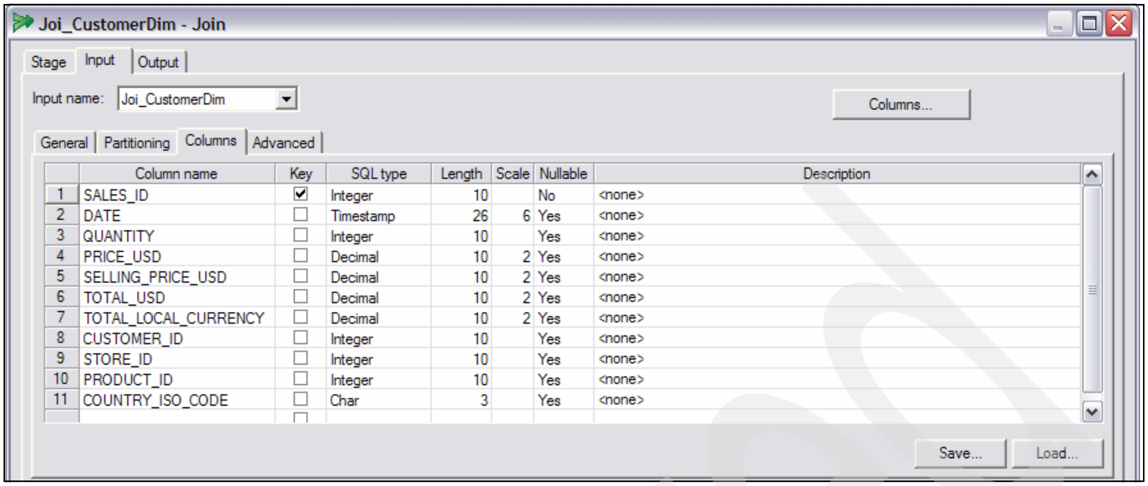


Figure 2-71 Join stage example 5/8

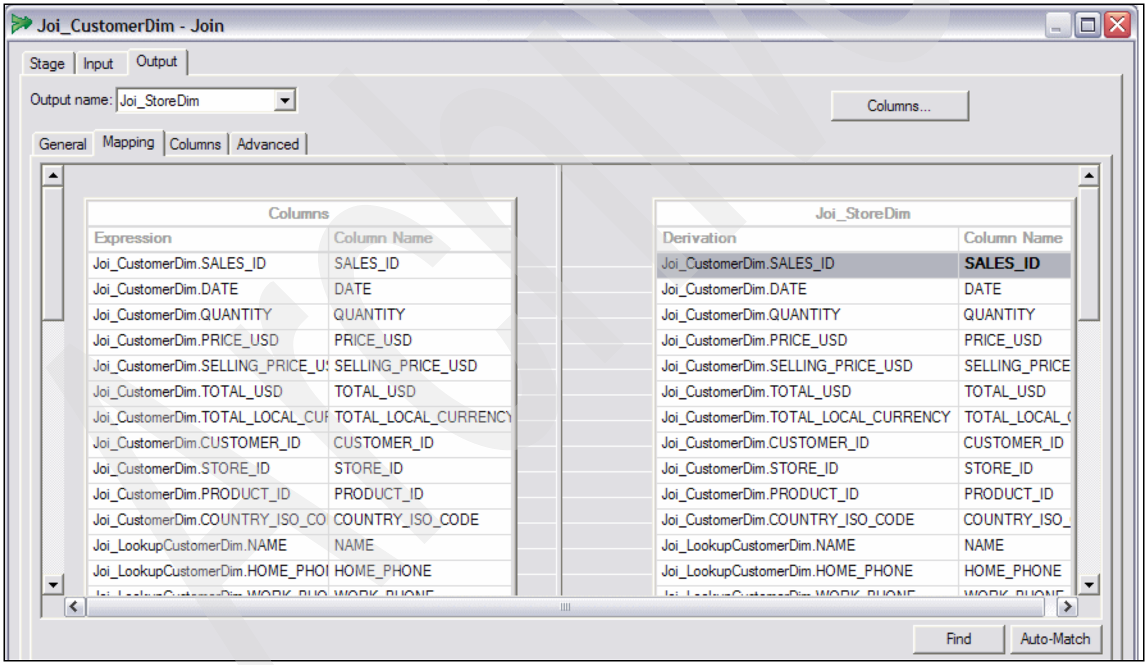


Figure 2-72 Join stage example 6/8

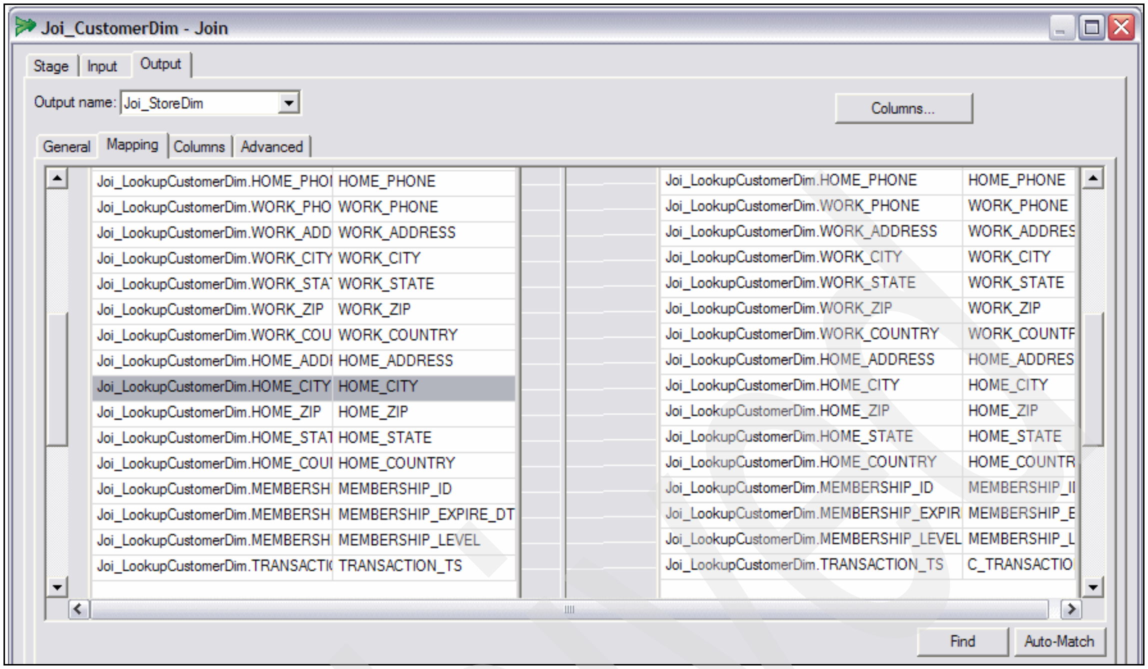
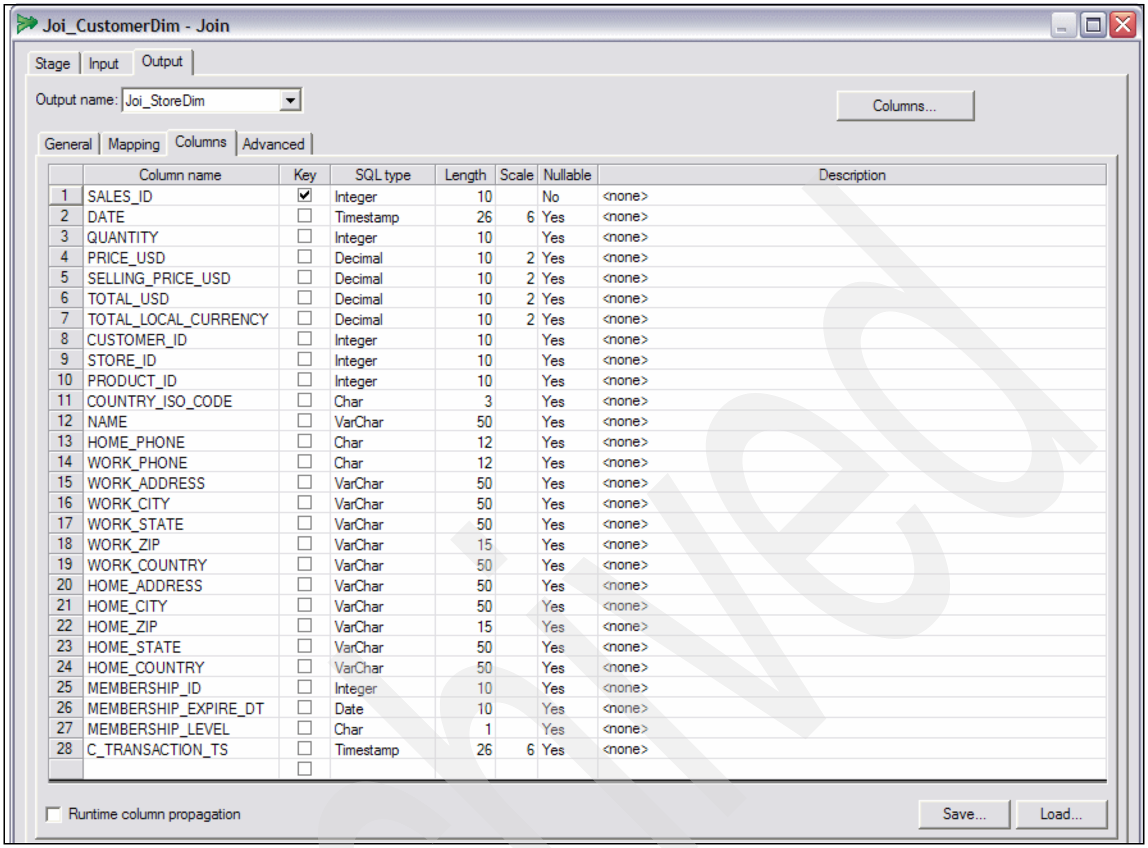


Figure 2-73 Join stage example 7/8



	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	SALES_ID	☑	Integer	10		No	<none>
2	DATE	☐	Timestamp	26	6	Yes	<none>
3	QUANTITY	☐	Integer	10		Yes	<none>
4	PRICE_USD	☐	Decimal	10	2	Yes	<none>
5	SELLING_PRICE_USD	☐	Decimal	10	2	Yes	<none>
6	TOTAL_USD	☐	Decimal	10	2	Yes	<none>
7	TOTAL_LOCAL_CURRENCY	☐	Decimal	10	2	Yes	<none>
8	CUSTOMER_ID	☐	Integer	10		Yes	<none>
9	STORE_ID	☐	Integer	10		Yes	<none>
10	PRODUCT_ID	☐	Integer	10		Yes	<none>
11	COUNTRY_ISO_CODE	☐	Char	3		Yes	<none>
12	NAME	☐	VarChar	50		Yes	<none>
13	HOME_PHONE	☐	Char	12		Yes	<none>
14	WORK_PHONE	☐	Char	12		Yes	<none>
15	WORK_ADDRESS	☐	VarChar	50		Yes	<none>
16	WORK_CITY	☐	VarChar	50		Yes	<none>
17	WORK_STATE	☐	VarChar	50		Yes	<none>
18	WORK_ZIP	☐	VarChar	15		Yes	<none>
19	WORK_COUNTRY	☐	VarChar	50		Yes	<none>
20	HOME_ADDRESS	☐	VarChar	50		Yes	<none>
21	HOME_CITY	☐	VarChar	50		Yes	<none>
22	HOME_ZIP	☐	VarChar	15		Yes	<none>
23	HOME_STATE	☐	VarChar	50		Yes	<none>
24	HOME_COUNTRY	☐	VarChar	50		Yes	<none>
25	MEMBERSHIP_ID	☐	Integer	10		Yes	<none>
26	MEMBERSHIP_EXPIRE_DT	☐	Date	10		Yes	<none>
27	MEMBERSHIP_LEVEL	☐	Char	1		Yes	<none>
28	C_TRANSACTION_TS	☐	Timestamp	26	6	Yes	<none>

Figure 2-74 Join stage example 8/8

2.11 Lookup

The Lookup stage is a processing stage. It is used to perform lookup operations on a data set read into memory from any other Parallel job stage that can output data. It can also perform lookups directly in all DBMSs or in a lookup table contained in a Lookup File Set stage.

The Lookup stage can have a reference link (Ds_rate), a single input link (shared_cont), a single output link (Trx_LocCurrency), and a single reject (Ds_reject) link as shown in Figure 2-75 on page 100.

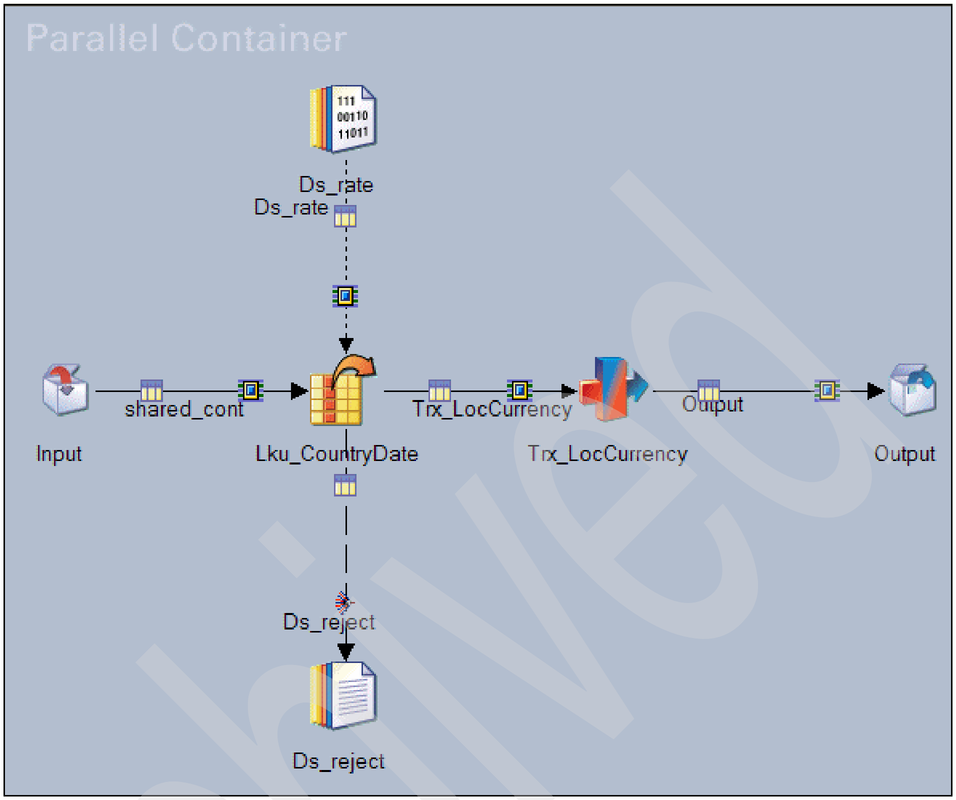


Figure 2-75 Lookup stage

Depending upon the type and setting of the stage(s) providing the lookup information, it can have multiple reference links (where it is directly looking up a DB2 table or Oracle table, it can only have a single reference link). A lot of the setting up of a lookup operation takes place on the stage providing the lookup table.

The input link carries the data from the source data set and is known as the primary link.

For each record of the source data set from the primary link, the Lookup stage performs a table lookup on each of the lookup tables attached by reference links. The table lookup is based on the values of a set of lookup key columns, one set for each table. The keys are defined on the Lookup stage. For lookups of data accessed through the Lookup File Set stage, the keys are specified when you create the lookup file set.

You can specify a condition on each of the reference links, such that the stage will only perform a lookup on that reference link if the condition is satisfied.

Lookup stages do not require data on the input link or reference links to be sorted. Be aware, though, that large in-memory lookup tables will degrade performance because of their paging requirements.

Each record of the output data set contains columns from a source record plus columns from all the corresponding lookup records where corresponding source and lookup records have the same value for the lookup key columns. The lookup key columns do not have to have the same names in the primary and the reference links.

The optional reject link carries source records that do not have a corresponding entry in the input lookup tables.

You can also perform a range lookup, which compares the value of a source column to a range of values between two lookup table columns. If the source column value falls within the required range, a row is passed to the output link. Alternatively, you can compare the value of a lookup column to a range of values between two source columns. Range lookups must be based on column values, not constant values. Multiple ranges are supported.

There are some special partitioning considerations for Lookup stages. You must ensure that the data being looked up in the lookup table is in the same partition as the input data referencing it. One way of doing this is to partition the lookup tables using the Entire method. Another way is to partition it in the same way as the input data.

The most common use for a lookup is to map short codes in the input data set onto expanded information from a lookup table, which is then joined to the incoming data and output. For example, you could have an input data set carrying names and addresses of your U.S. customers. The data as presented identifies state as a two letter U. S. state postal code, but you want the data to carry the full name of the state.

You could define a lookup table that carries a list of codes matched to states, defining the code as the key column. As the Lookup stage reads each line, it uses the key to look up the state in the lookup table. It adds the state to a new column defined for the output link, and so the full state name is added to each address. If any state codes have been incorrectly entered in the data set, the code will not be found in the lookup table, and so that record will be rejected.

Lookups can also be used for validation of a row. If there is no corresponding entry in a lookup table to the key's values, the row is rejected.

Figure 2-76 on page 102 through Figure 2-81 on page 106 show an example of the configuration of a Lookup stage in a job (“J07A_SharedContainerLookupCurrency” on page 273 in the retail industry scenario described in “Retail industry scenario” on page 140), as follows:

1. Figure 2-76 on page 102 shows the job that performs a lookup of the currency conversion rate of the day from a data set that was populated using a Web service. This is described in “J07A_SharedContainerLookupCurrency” on page 273 and is not repeated here. Instead, we only focus on the configuration of the Lookup stage in this job.
2. Figure 2-77 on page 103 through Figure 2-79 on page 105 show the mapping of one column each from the two input links (shared_cont and Ds_rate) to the output link Trx_LocCurrency. The column definitions of each of these links is shown in the bottom pane.
3. Figure 2-80 on page 105 shows the **Link Ordering** tab in the Stage page, which allows you to specify which input link is regarded as the Primary (shared_cont) and which link is regarded as the Lookup (Ds_rate). You can use this tab to reorder the links as required.
4. Figure 2-81 on page 106 shows the **General** tab in the Outputs page, which has Runtime column propagation checked.

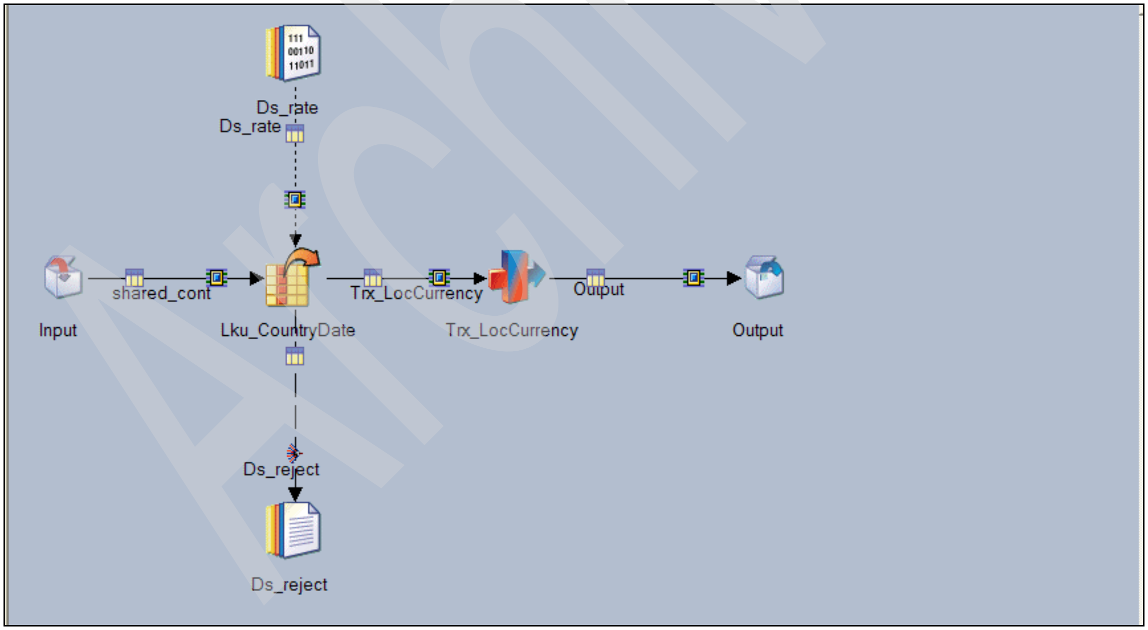


Figure 2-76 Lookup stage example 1/6

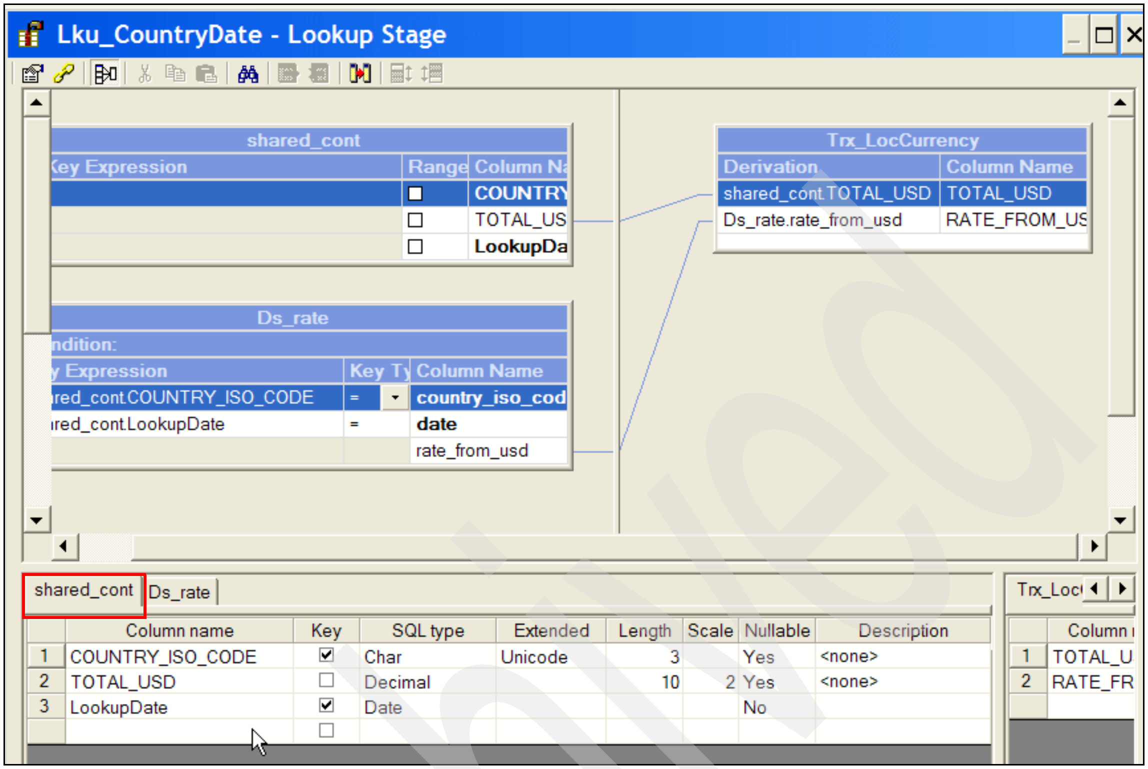


Figure 2-77 Lookup stage example 2/6

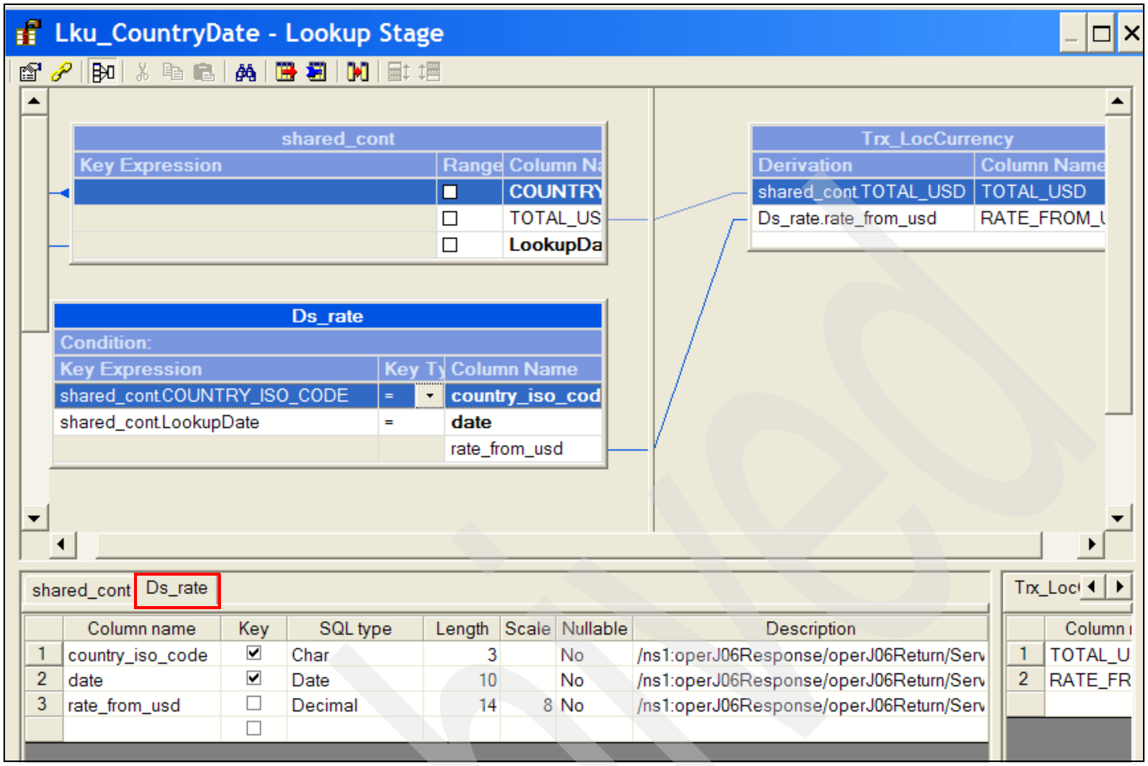


Figure 2-78 Lookup stage example 3/6

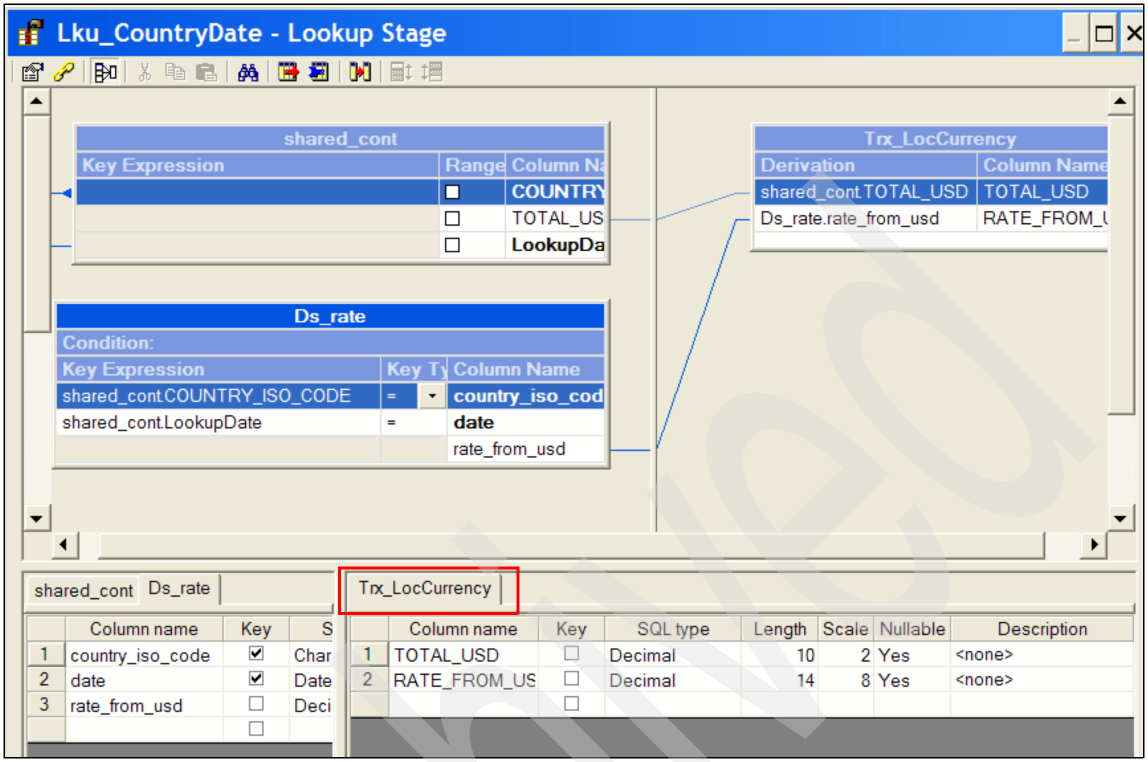


Figure 2-79 Lookup stage example 4/6

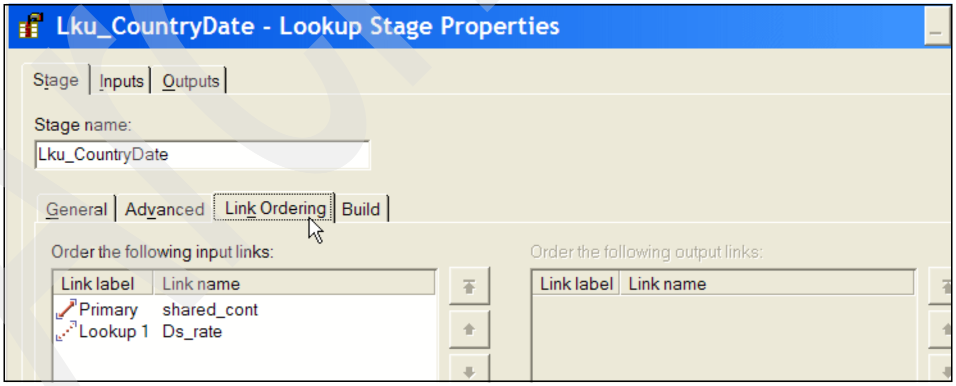


Figure 2-80 Lookup stage example 5/6

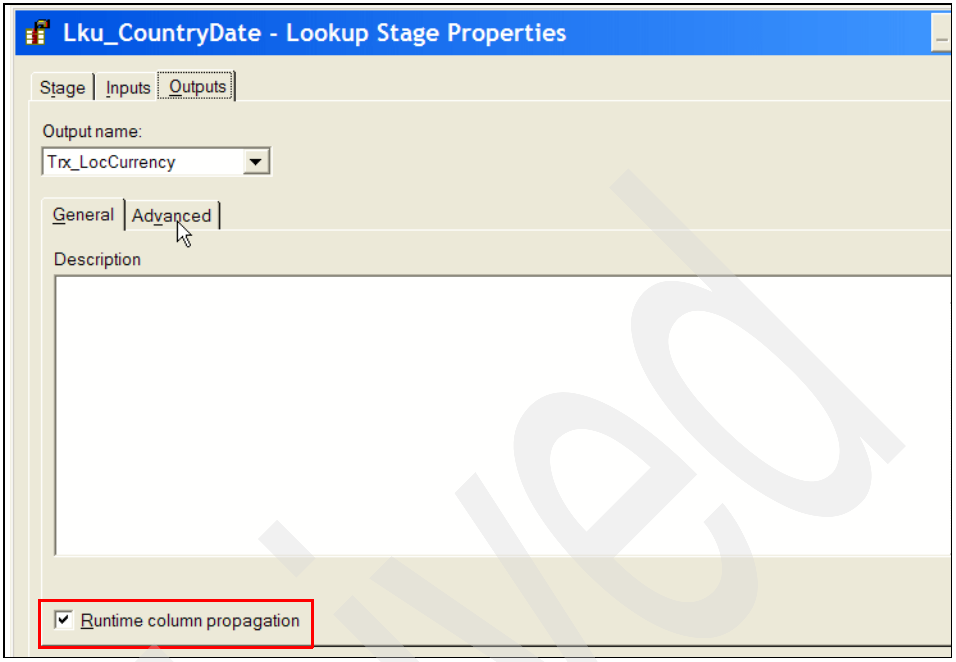


Figure 2-81 Lookup stage example 6/6

2.12 Merge

The Merge stage is a processing stage. It can have any number (more than 1) of input links, a single output link, and the same number of reject links as there are update input links, as shown in Figure 2-82.

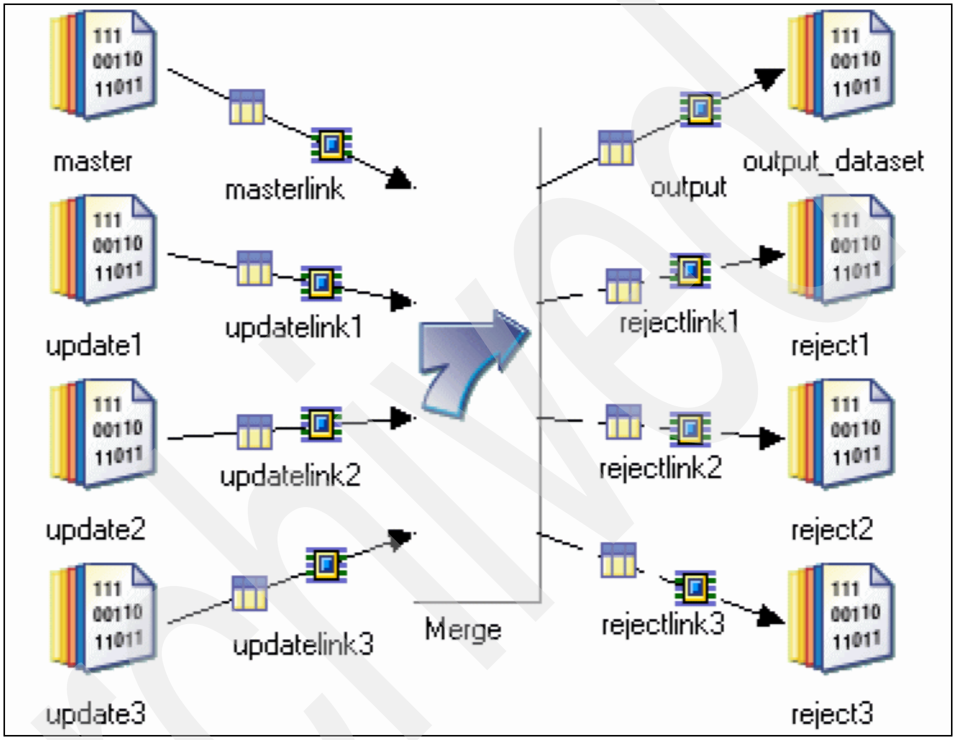


Figure 2-82 Merge stage

As mentioned earlier, the Merge stage is one of three stages that join tables based on the values of key columns. The other two are the Join stage as described in 2.10, “Join” on page 93 and the Lookup stage described in 2.11, “Lookup” on page 99. The three stages differ mainly in the memory they use, the treatment of rows with unmatched keys, and their requirements for data being input (for example, whether it is sorted).

The Merge stage combines a master data set with one or more update data sets. The columns from the records in the master and update data sets are merged so that the output record contains all the columns from the master record plus any additional columns from each update record that are required. A master record and an update record are merged only if both of them have the same values for the merge key column(s) that you specify. Merge key columns are one or more columns that exist in both the master and update records.

The data sets input to the Merge stage must be key partitioned and sorted. This ensures that rows with the same key column values are located in the same partition and will be processed by the same node. It also minimizes memory requirements because fewer rows have to be in memory at any one time. Choosing the auto partitioning method will ensure that partitioning and sorting is done. If sorting and partitioning are carried out on separate stages before the Merge stage, IBM InfoSphere DataStage in auto partition mode will detect this and not re-partition (alternatively you could explicitly specify the Same partitioning method).

As part of preprocessing your data for the Merge stage, you should also remove duplicate records from the master data set. If you have more than one update data set, you must remove duplicate records from the update data sets as well.

Unlike Join stages and Lookup stages, the Merge stage allows you to specify several reject links. You can route update link rows that fail to match a master row down a reject link that is specific for that link. You must have the same number of reject links as you have update links. You can also specify whether to drop unmatched master rows, or output them on the output data link.

2.13 Sequential File

The Sequential File stage is a file stage. It allows you to read data from or write data to one or more flat files as shown in Figure 2-83. The stage can have a single input link or a single output link, and a single rejects link.

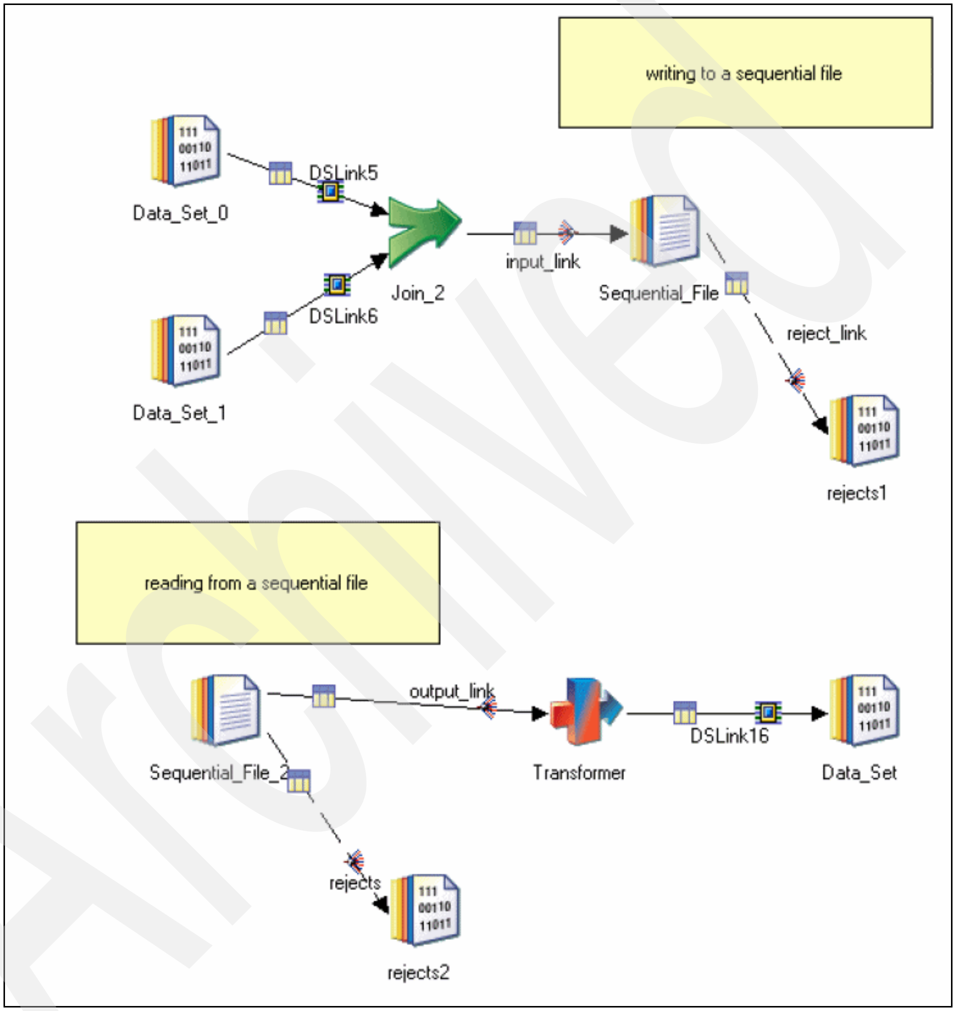


Figure 2-83 Sequential stage

The stage executes in parallel mode by default if reading multiple files but executes sequentially if it is only reading one file. By default, a complete file will be read by a single node (although each node might read more than one file).

For fixed-width files, however, you can configure the stage to behave differently:

- You can specify that single files can be read by multiple nodes. This can improve performance on cluster systems.
- You can specify that a number of readers run on a single node. This means, for example, that a single file can be partitioned as it is read.

These two options are mutually exclusive.

The stage executes in parallel if writing to multiple files, but executes sequentially if writing to a single file.

When reading or writing a flat file, IBM InfoSphere DataStage needs to know something about the format of the file. The information required is how the file is divided into rows and how rows are divided into columns.

Figure 2-84 on page 111 through Figure 2-87 on page 113 show an example of the configuration of a Sequential stage in a job (“J07_IL_Daily_LoadSalesStore” on page 282 in the retail industry scenario described in “Retail industry scenario” on page 138), as follows:

1. Figure 2-84 on page 111 shows the job that reads sales data from a sequential file and performs a lookup to obtain the current exchange rate for the appropriate country code and writes it to a DB2 table. This is described in “J07_IL_Daily_LoadSalesStore” on page 282 and is not repeated here. Instead, we only focus on the configuration of the Sequential File stage in this job.
2. The Output page allows you to specify details about how the Sequential File stage reads data from one or more flat files. Figure 2-85 on page 111 shows the **Properties** tab in the Output page, which allows you to specify properties for the output link. These dictate how incoming data is read from what files.

> **Note:** Information for properties such as File and Schema File is not provided here since we expect to provide it at execution time. You specify a job parameter to represent the missing information, so that when you run the job, you are prompted to supply a value for the job parameter. This is shown in Figure 3-215 on page 291.

Figure 2-86 on page 112 shows the **Format** tab in the Output page, which allows you to supply information about the format of the flat file or files that you are reading. The tab has a similar format to the **Properties** tab. We let the properties default.

Figure 2-87 on page 113 shows the **Columns** tab in the Output page, which specifies the explicitly defined column definitions of the output data. Runtime column propagation is checked to ensure that the metadata of all columns as specified in the schema file are propagated to the next stage.

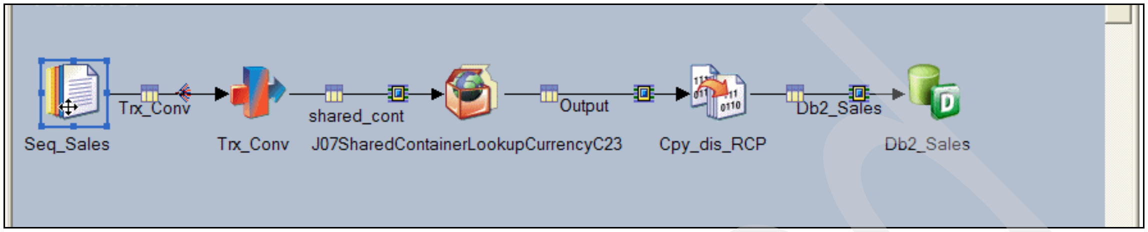


Figure 2-84 Sequential stage example 1/4

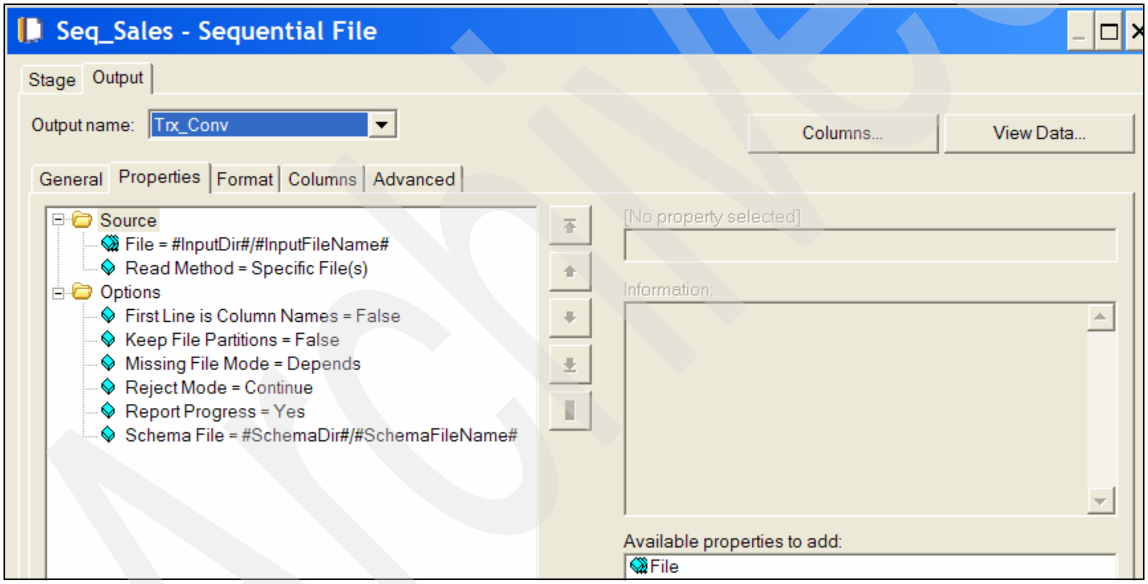


Figure 2-85 Sequential stage example 2/4

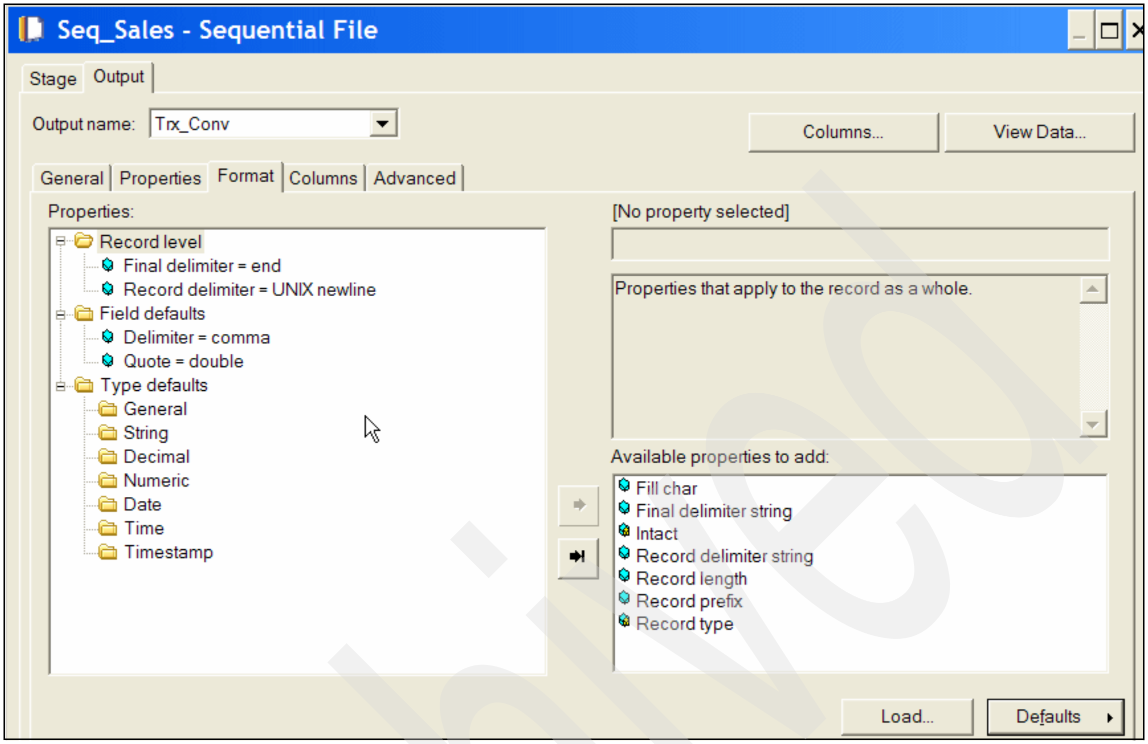


Figure 2-86 Sequential stage example 3/4

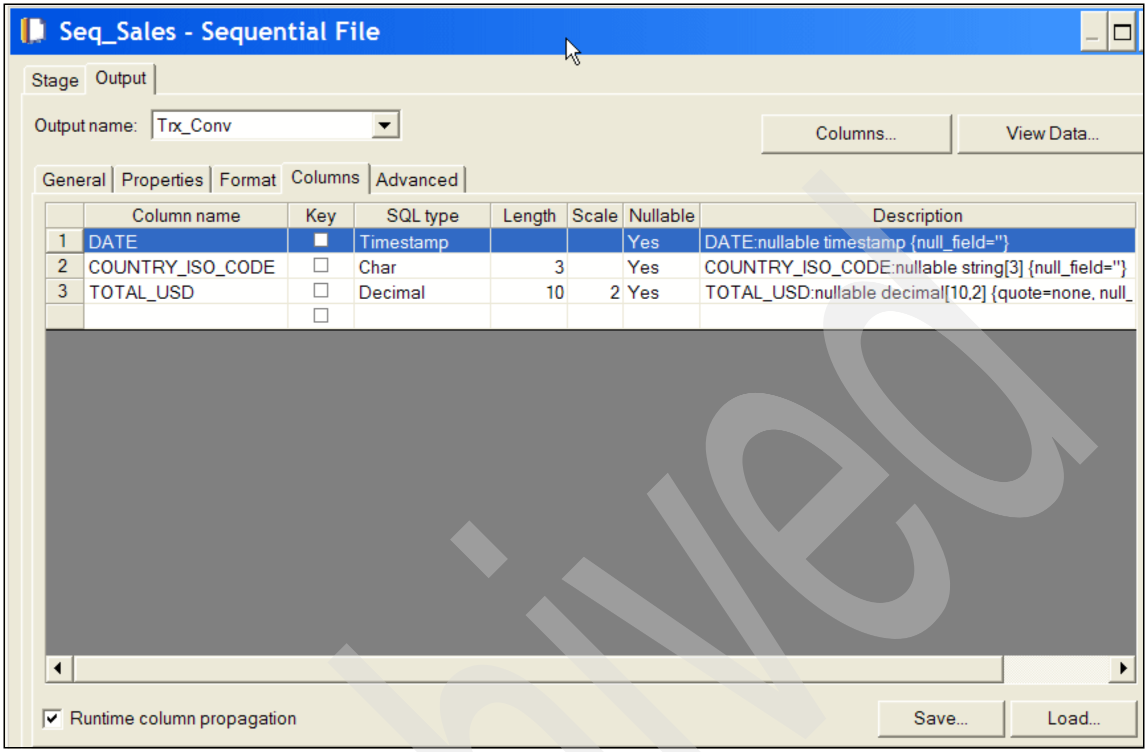


Figure 2-87 Sequential stage example 4/4

2.14 Slowly Changing Dimension

The Slowly Changing Dimension (SCD) stage is a processing stage that works within the context of a star schema database. The SCD stage has a single input link, a single output link, a dimension reference link, and a dimension update link as shown in Figure 2-88 on page 114.

The SCD stage reads source data on the input link, performs a dimension table lookup on the reference link, and writes data on the output link. The output link can pass data to another SCD stage, to a different type of processing stage, or to a fact table. The dimension update link is a separate output link that carries changes for the dimension. You can perform these steps in a single job or a series of jobs, depending on the number of dimensions in your database and your performance requirements.

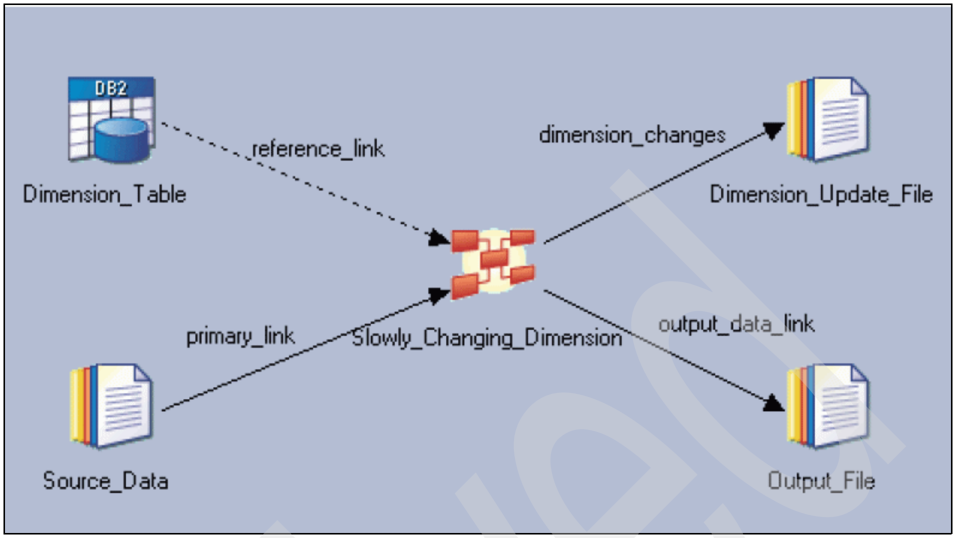


Figure 2-88 SCD stage

SCD stages support both SCD Type 1 and SCD Type 2 processing, as follows:

- SCD Type 1

  Overwrites an attribute in a dimension table.

- SCD Type 2

  Updates the existing row to indicate it expired and adds a new row to the dimension table.

Each SCD stage processes a single dimension and performs lookups by using an equality matching technique. If the dimension is a database table, the stage reads the database to build an in memory lookup table of all the current dimension entries.

- If a match is found, the SCD stage updates[7] rows in the dimension table to reflect the changed data.
- If a match is not found, the stage creates a new row in the dimension table. All of the columns that are needed to create a new dimension row *must be* present in the source data.

[7] As indicated earlier, a Type 1 change results in an update to the existing row; a Type 2 change updates the existing row to indicate it expired and adds a new row to the dimension table.

Important: If both Type 1 and Type 2 changes exist for the same record, the Type 2 change takes precedence over the Type 1 change. This means that a new dimension record is first created, and then the Type 1 changes are applied to the newly created record only. The Type 1 changes in this case are not reflected in the earlier row(s).

Input data to SCD stages must accurately represent the order in which events occurred. You might have to presort your input data by a sequence number or a date field. If a job has multiple SCD stages, you must ensure that the sort order of the input data is correct for each stage.

If the SCD stage is running in parallel, the input data must be hash partitioned by key. Hash partitioning allows all records with the same business key to be handled by the same process. The SCD stage divides the dimension table across processes by building a separate lookup table for each process.

Each SCD stage processes a single dimension, but job design is flexible. You can design one or more jobs to process dimensions, update the dimension table, and load the fact table.

- Processing dimensions

  You can create a separate job for each dimension, one job for all dimensions, or several jobs, each of which has several dimensions.

- Updating dimensions

  You can update the dimension table as the job runs by linking the SCD stage to a database stage, or you can update the dimension table later by sending dimension changes to a flat file that you use in a separate job. Actual dimension changes are applied to the lookup table in memory and are mirrored to the dimension update link, giving you the flexibility to handle a series of changes to the same dimension row.

- Loading the fact table

  You can load the fact table as the final step of the job that updates the last dimension, or in a separate job.

We describe two possible job designs, as follows:

- Figure 2-89 on page 116 through Figure 2-91 on page 117 show a series of jobs, where the first job performs the dimension lookup, the second job performs the dimension table update, and the third job loads the fact table.

The job design shown in these figures minimizes the use of database facilities. The job in Figure 2-89 builds a lookup table in memory for the dimension, so the database connection is active only when the table is being created. Both the output data and the dimension update records are written to flat files.

The jobs in Figure 2-90 and Figure 2-91 use these files to update the dimension table and to load the fact table later. This series of jobs represents a single dimension table. If you have multiple dimensions, each has a job corresponding to Figure 2-89 and Figure 2-90. The output of the last job corresponding to Figure 2-89 is the input to the job corresponding to Figure 2-91.

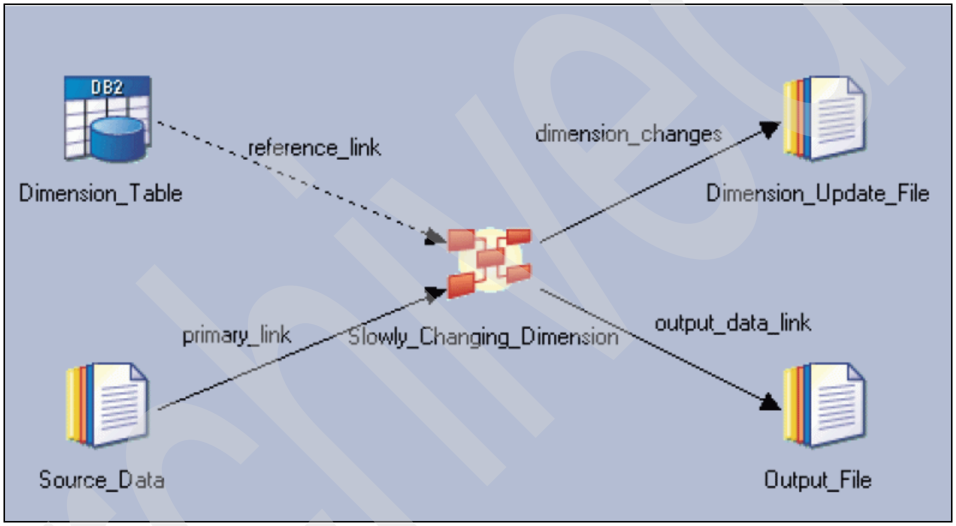


Figure 2-89   SCD job involving 3 stages 1/3

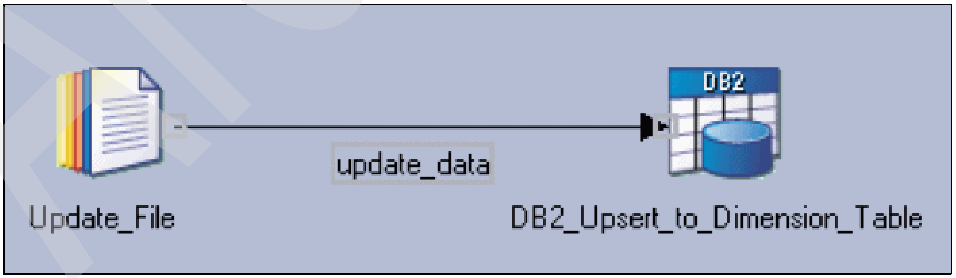


Figure 2-90   SCD job involving 3 stages 2/3

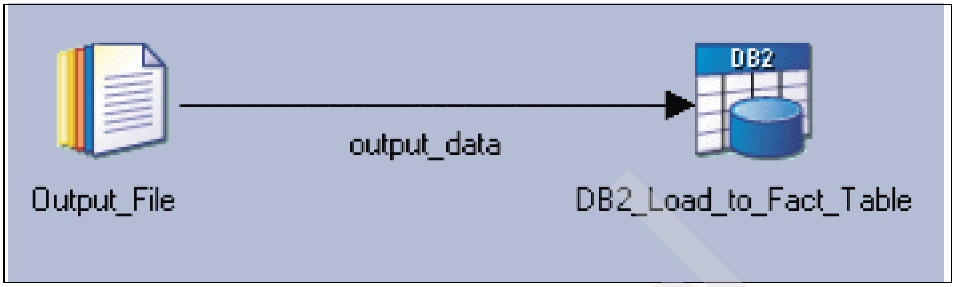


Figure 2-91 SCD job involving 3 stages 3/3

- Figure 2-92 on page 118 shows a strategy that combines jobs corresponding to Figure 2-89 on page 116 and Figure 2-90 on page 116 into a single step.

  Here the SCD stage provides the necessary column information to the database stage so that it can generate the correct INSERT and UPDATE SQL statements to update the dimension table.

  By contrast, the design in Figure 2-89 on page 116 through Figure 2-91 here requires you to save your output columns from the SCD stage in the job corresponding to Figure 2-89 on page 116 as a table definition in the repository. You must then load columns from this table definition into the database stage in the job corresponding to Figure 2-90 on page 116.

> **Note:** The advantage of the approach shown in Figure 2-92 on page 118, over combining all the updates (dimension and fact table) in a single job, is that you can ensure that all the dimension tables are updated correctly before updating the fact table. This allows you to correct any dimension table update failures before running the fact table update so that no failures occur during the fact table load.
>
> Combining all the updates (dimension and fact table) in a single job (not shown here) opens the possibility of a failure of a dimension table update and can cause fact table update failures.

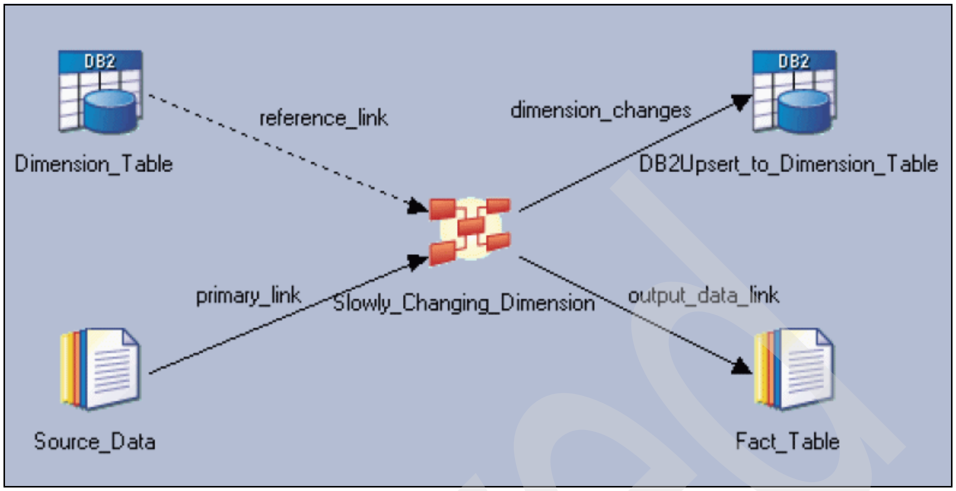


Figure 2-92 SCD job involving a single stage

Purpose codes and surrogate keys are important concepts in SCD processing, as follows:

- Purpose codes are an attribute of dimension columns in SCD stages. Purpose codes are used to build the lookup table, to detect dimension changes, and to update the dimension table.
  - Building the lookup table

    The SCD stage uses purpose codes to determine how to build the lookup table for the dimension lookup. If a dimension has only Type 1 columns, the stage builds the lookup table by using all dimension rows. If any Type 2 columns exist, the stage builds the lookup table by using only the current rows. If a dimension has a Current Indicator column, the stage uses the derivation value of this column on the Dim Update tab to identify the current rows of the dimension table. If a dimension does not have a Current Indicator column, then the stage uses the Expiration Date column and its derivation value to identify the current rows. Any dimension columns that are not needed are not used. This technique minimizes the amount of memory that is required by the lookup table.
  - Detecting dimension changes

    Purpose codes are also used to detect dimension changes. The SCD stage compares Type 1 and Type 2 column values to source column values to determine whether to update an existing row, insert a new row, or expire a row in the dimension table.

– Updating the dimension table

Purpose codes are part of the column metadata that the SCD stage propagates to the dimension update link. You can send this column metadata to a database stage in the same job, or you can save the metadata on the Columns tab and load it into a database stage in a different job. When the database stage uses the auto-generated SQL option to perform inserts and updates, it uses the purpose codes to generate the correct SQL statements.

The SCD stage provides nine purpose codes to support dimension processing, as follows:

– (blank)

The column has no SCD purpose. This purpose code is the default.

– Surrogate Key

The column is a surrogate key that is used to identify dimension records.

– Business Key

The column is a business key that is typically used in the lookup condition.

– Type 1

The column is an SCD Type 1 field. SCD Type 1 column values are always current. When changes occur, the SCD stage overwrites existing values in the dimension table.

– Type 2

The column is an SCD Type 2 field. SCD Type 2 column values represent a point in time. When changes occur, the SCD stage updates the existing row to indicate that it has expired, and adds a new row to the dimension table.

– Current Indicator (Type 2)

The column is the current record indicator for SCD Type 2 processing. Only one Current Indicator column is allowed.

– Effective Date (Type 2)

The column is the effective date for SCD Type 2 processing. Only one Effective Date column is allowed.

– Expiration Date (Type 2)

The column is the expiration date for SCD Type 2 processing. An Expiration Date column is required if there is no Current Indicator column, otherwise it is optional.

  – SK Chain

    The column is used to link a record to the previous record or the next record by using the value of the Surrogate Key column. Only one Surrogate Key column can exist if you have an SK Chain column.

- Surrogate keys are used to join a dimension table to a fact table in a star schema database.

  When the SCD stage performs a dimension lookup:

  – If a matching record is found, it retrieves the value of the existing surrogate key.

  – If a match is not found, the stage obtains a new surrogate key value by using the derivation of the Surrogate Key column on the Dim Update tab.

    - If you want the SCD stage to generate new surrogate keys by using a key source that you created with a Surrogate Key Generator stage as described in “Surrogate Key Generator” on page 132, you must use the NextSurrogateKey function to derive the Surrogate Key column.
    - If you want to use your own method to handle surrogate keys, you should derive the Surrogate Key column from a source column.

  You can replace the dimension information in the source data stream with the surrogate key value by mapping the Surrogate Key column to the output link.

Figure 2-93 on page 122 through Figure 2-99 on page 126 show an example of the configuration of a Slowly Changing Dimension stage in a job (“J07_IL_Daily_LoadSalesStore” on page 282 in the retail industry scenario described in “Retail industry scenario” on page 138), as follows:

1. Figure 2-93 on page 122 shows the job that reads sales transactions with attributes of dimension tables, updates the dimension (Store, Customer, Product and Date) is Type 1or Type 2 changes are present, appends the surrogate key (via a lookup of the appropriate dimension table) to the sales transactions, and generates the enhanced sales transactions file for updating the fact table. This is described in “J07_IL_Daily_LoadSalesStore” on page 282 and is not repeated here. Instead, we only focus on the configuration of the Slowly Changing Dimension stage in this job.

   To edit an SCD stage, you must define how the stage should lookup data in the dimension table, obtain surrogate key values, update the dimension table, and write data to the output link.

2. Figure 2-94 on page 123 shows the Lookup tab in the Input page for the Odbc_StoreDim link, which allows you to define the match condition to use for the dimension lookup. The match condition specifies how the SCD stage should perform the dimension lookup. You may associate one or more pairs of columns. A successful lookup requires all associated pairs of columns to match. In this case, STORE_ID is the match condition.

Figure 2-94 on page 123 also shows the purpose codes, which specify how the SCD stage should process dimension data. Purpose codes apply to columns on the dimension reference link and on the dimension update link. Purpose codes are selected according to the type of columns in a dimension:

- If a dimension contains a Type 2 column, you must select a Current Indicator column, an Expiration Date column, or both. An Effective Date column is optional. You cannot assign Type 2 and Current Indicator to the same column.
- If a dimension contains only Type 1 columns, no Current Indicator, Effective Date, Expiration Date, or SK Chain columns are allowed.

STORE_DIM_KEY is identified with the Surrogate Key purpose code.

CITY_POPULATION and STATE_POPULATION are identified with the Type 1 purpose code.

MANAGER_NAME is identified with the Type 2 purpose code.

CURRENT_IND (Current Indicator (Type 2) purpose code), EFFECTIVE_TS (Effective Date (Type 2) purpose code), and EXPIRATION_TS (Expiration Date (Type 2) purpose code) are the other specifications.

3. Figure 2-95 on page 123 shows the **Surrogate Key** tab in the Input page for the Odbc_StoreDim link, which allows you to specify the source type and source name of the surrogate key generator stage generated file.

   Calls to the key source are made by the NextSurrogateKey function. On the Dim Update tab in the next step, we create a derivation that uses the NextSurrogateKey function for the column that has a purpose code of Surrogate Key. The NextSurrogateKey function returns the value of the next surrogate key when the SCD stage creates a new dimension row.

4. Figure 2-96 on page 124 shows the **Dim_Update** tab in the Output page for the Ds_StoreDimUpdate link, which allows you to specify how to update the dimension table, including the values to use for new records and when records should expire. Every dimension column must have a derivation. Relationship lines show which dimension columns are derived from source columns, either directly or as part of an expression.

   The Derivation columns show the following values:

   - STORE_DIM_KEY has the NextSurrogateKey() function identified in the previous step.
   - CURRENT_IND has the value of “Y”
   - EXPIRATION_TS has the value 2099-12-31-00.00.00.000000

5. Figure 2-97 on page 125 shows the **Output Map** tab in the Output page for the Fri_Store link, which allows you to specify how to write (map) data from the input links to the output link. You can map input data and dimension data to the output link. Dimension data is mapped from the lookup table in memory, so new rows and changed data are available for output.
6. Figure 2-98 on page 126 shows the **Columns** tab in the Output page for the Fri_Store link with the columns definitions corresponding to the mapping defined in Figure 2-97 on page 125.
7. Figure 2-99 on page 126 shows the column definitions for the Ds_StoreDimUpdate link under the **Columns** tab in the Output page. The mapping that resulted in this is not shown here.

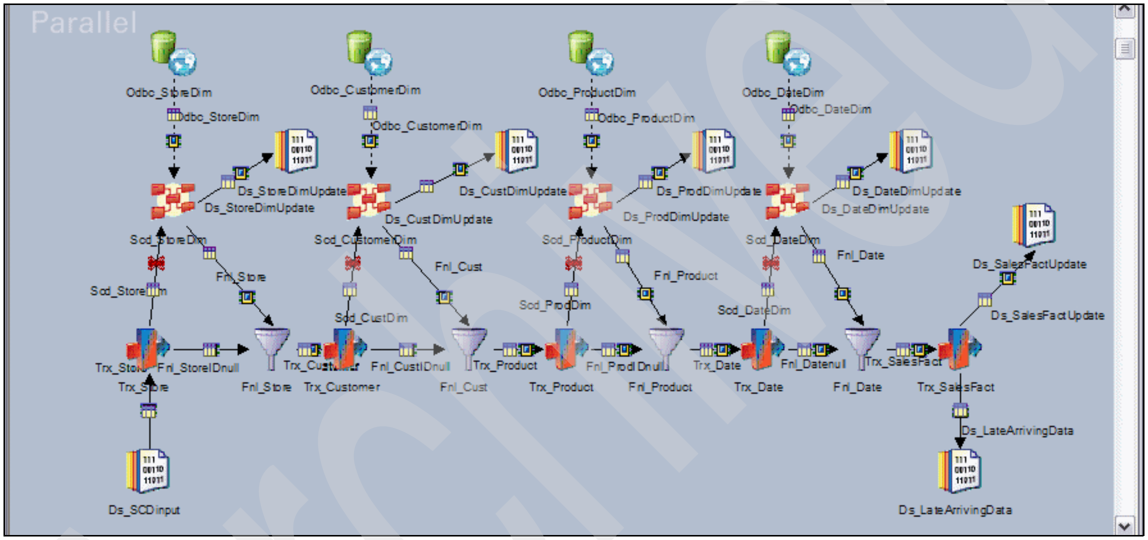


Figure 2-93 SCD stage example 1/7

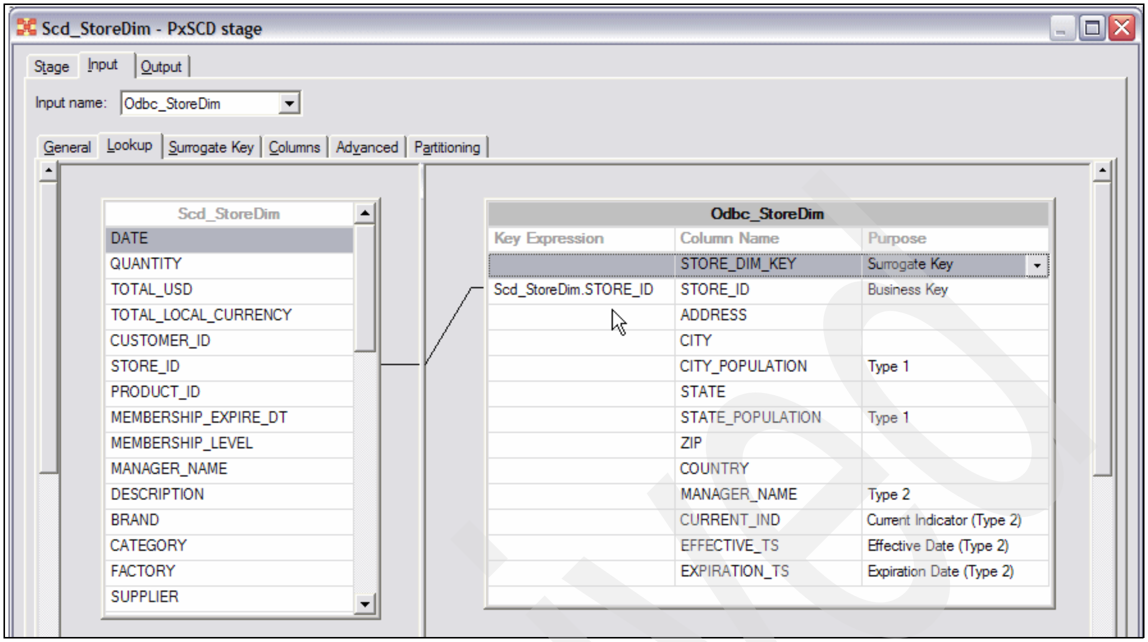


Figure 2-94 SCD stage example 2/7

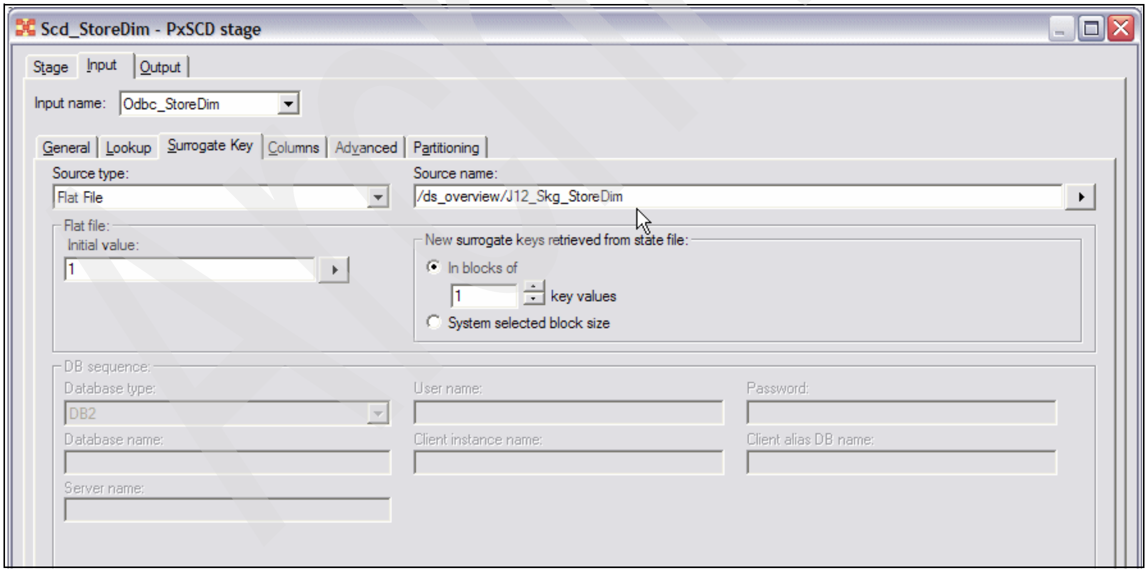


Figure 2-95 SCD stage example 3/7

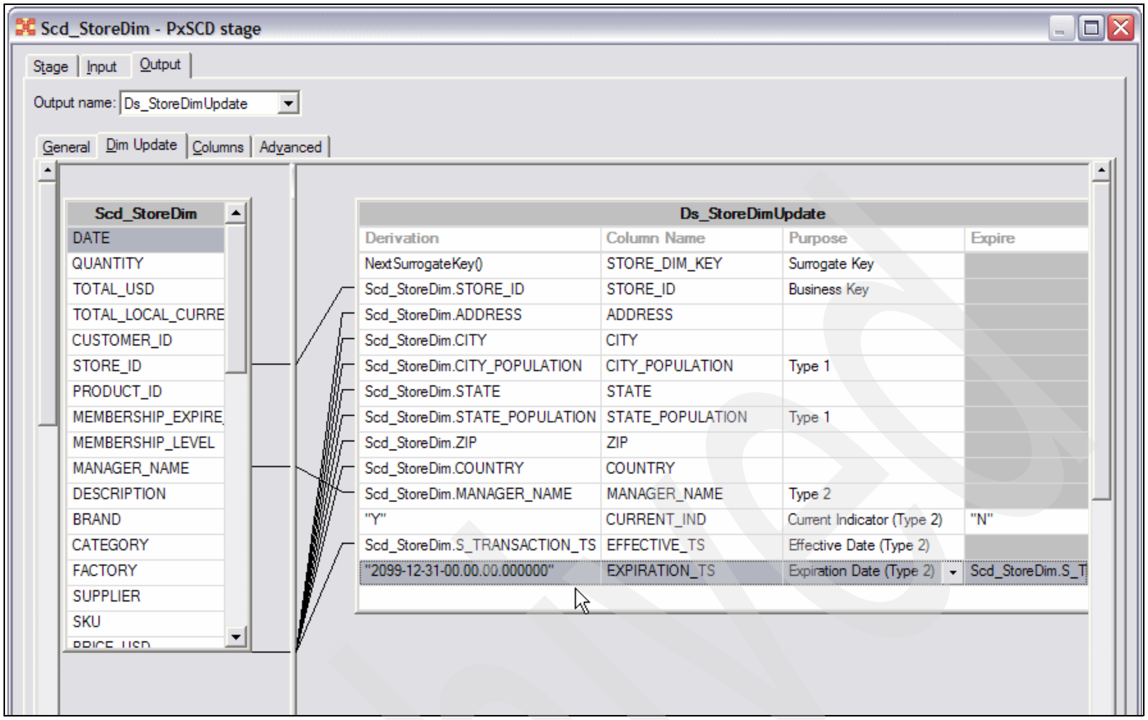


Figure 2-96 SCD stage example 4/7

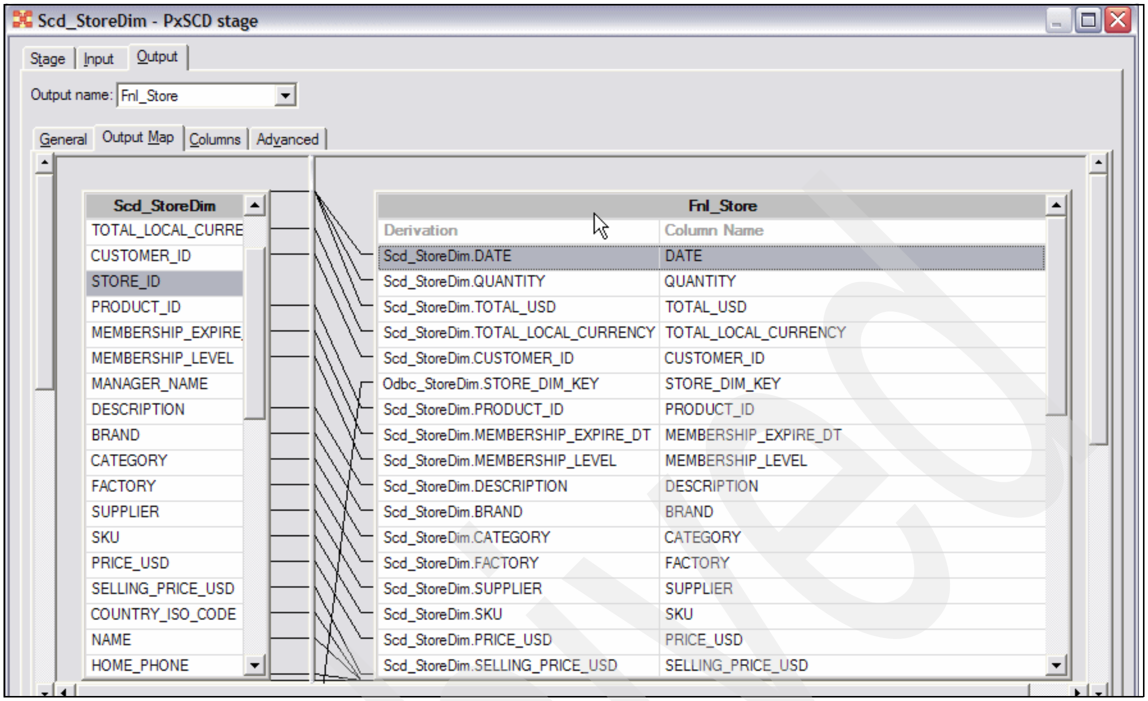


Figure 2-97   SCD stage example 5/7

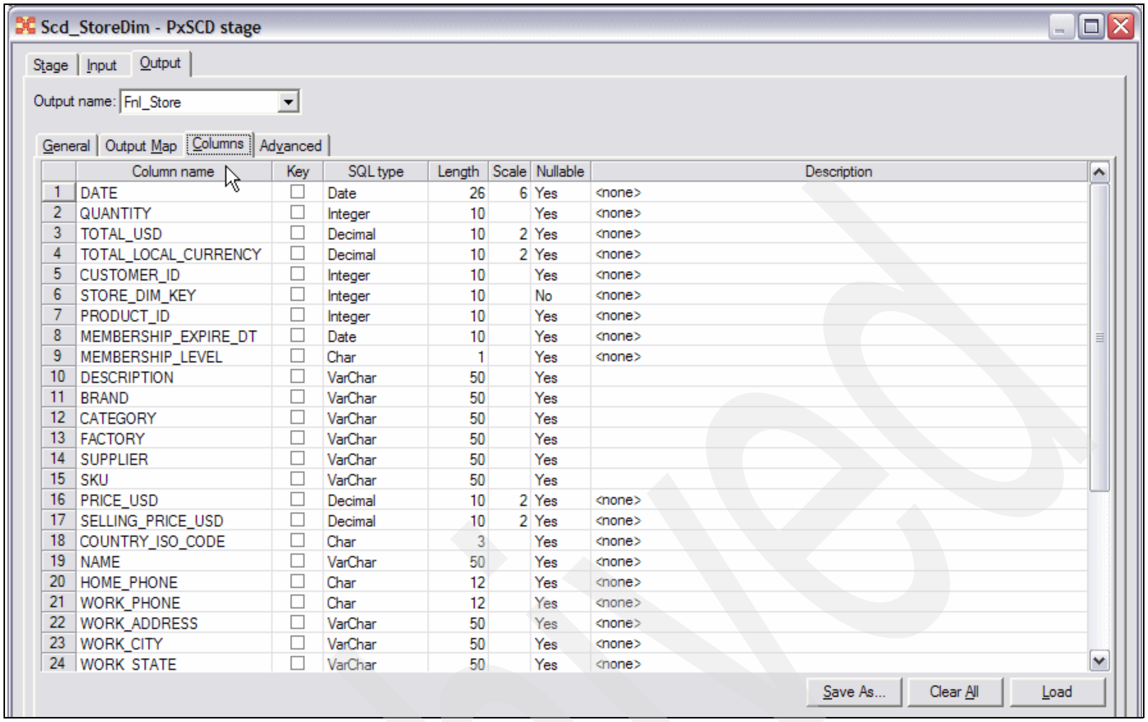


Figure 2-98 SCD stage example 6/7

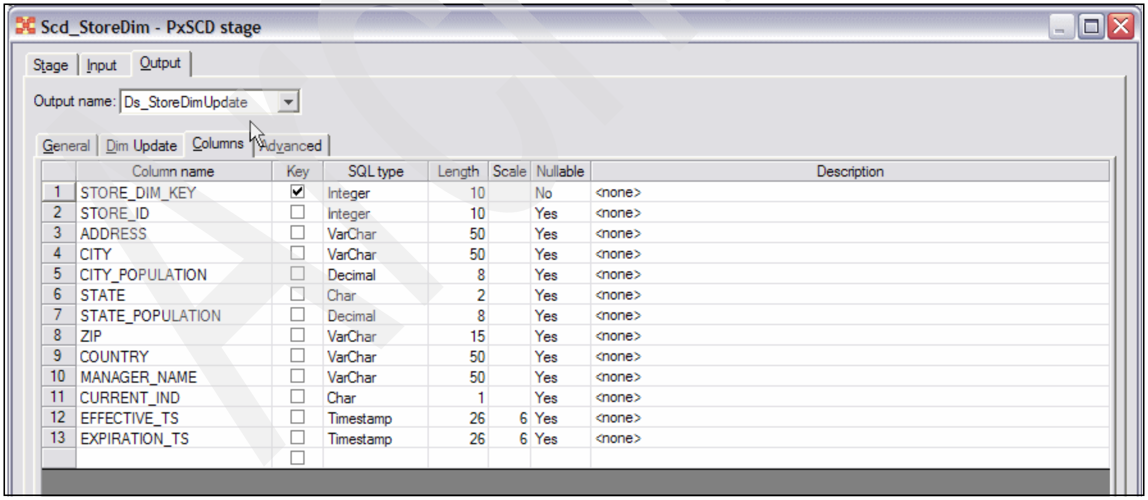


Figure 2-99 SCD stage example 7/7

2.15 Sort

The Sort stage is a processing stage. It is used to perform more complex sort operations than can be provided for on the Input page Partitioning tab of parallel job stage editors. You can also use it to insert a more explicit sort operation where you want to make your job easier to understand.

The Sort stage has a single input link that carries the data to be sorted, and a single output link carrying the sorted data as shown in Figure 2-100.

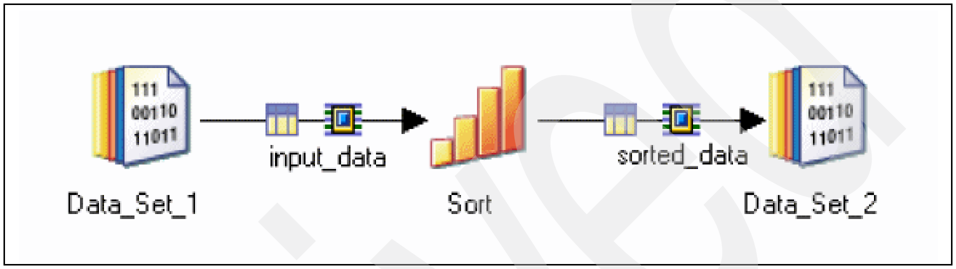


Figure 2-100 Sort stage

You specify sorting keys as the criteria on which to perform the sort. A key is a column on which to sort the data, for example, if you had a name column you might specify that as the sort key to produce an alphabetical list of names. The first column you specify as a key to the stage is the primary key, but you can specify additional secondary keys.

If multiple rows have the same value for the primary key column, then IBM InfoSphere DataStage uses the secondary columns to sort these rows. You can sort in sequential mode to sort data in its entirety or in parallel mode to sort data within partitions,

Many types of processing, such as re-partitioning, can destroy the sort order of a sorted data set. For example, assume you sorted a data set on a system with four processing nodes and stored the results to a data set stage. The data set will therefore have four partitions. You then use that data set as input to a stage executing on a different number of nodes, possibly due to node constraints.

IBM InfoSphere DataStage automatically re-partitions a data set to spread out the data set to all nodes in the system, unless you tell it not to, thereby possibly destroying the sort order of the data. You could avoid this by specifying the "Same" partitioning method. The stage then does not perform any re-partitioning as it reads the input data set, and the original partitions are preserved.

You must also be careful when using a stage operating sequentially to process a sorted data set. A sequential stage executes on a single processing node to perform its action. Sequential stages will collect the data where the data set has more than one partition, which may also destroy the sorting order of its input data set. You can avoid this if you specify the collection method as follows:

- If the data was range partitioned before being sorted, you should use the ordered collection method to preserve the sort order of the data set. Using this collection method causes all the records from the first partition of a data set to be read first, then all records from the second partition, and so on.
- If the data was hash partitioned before being sorted, you should use the sort merge collection method specifying the same collection keys as the data was partitioned on.

By default, the stage will sort with the native IBM InfoSphere DataStage sorter, but you direct it to use the UNIX sort command.

Figure 2-101 on page 129 through Figure 2-106 on page 131 show an example of the configuration of a Sort stage in a job ("J09_IL_LoadLookupCustomerDim" on page 320 in the retail industry scenario described in "Retail industry scenario" on page 138), as follows:

1. Figure 2-101 on page 129 shows the job that creates an intermediate dimension lookup table and involves the use of a sort. This is described in "J09_IL_LoadLookupCustomerDim" on page 320 and is not repeated here. Instead, we only focus on the configuration of the Sort stage in this job.
2. Figure 2-102 on page 129 shows the **Properties** tab in the Stage page, which allows you to specify properties that determine what the stage actually does.
   - The Sorting Keys category Key property identifies columns CUSTOMER_ID and EFFECTIVE_TS as the sorting keys with an Ascending for the Sort Order property.
   - The Options category properties were allowed to default as shown.
3. The Input page for the Srt_CustomerDim link allows you to specify details about the data coming in to be sorted.
   - Figure 2-103 on page 130 shows the **Partitioning** tab, which allows you to specify details about how the incoming data is partitioned or collected before the sort is performed. A sort is performed using the CUSTOMER_ID as the sort key.

     > **Note:** We should have marked CUSTOMER_ID as having been sorted previously. This was an error on our part.

   - Figure 2-104 on page 130 shows the **Columns** tab with the column definitions of the input data.

4. The Output page for the Rmd_CustomerDim allows you to specify details about data output from the Sort stage.

   - Figure 2-105 on page 131 shows the **Mapping** tab allows you to specify how the output columns are derived, i.e., what input columns map onto them. In this case the data has been mapped directly across from the input.
   - Figure 2-106 on page 131 shows the **Columns** tab with the column definitions of the output data based on the mapping in the earlier step.

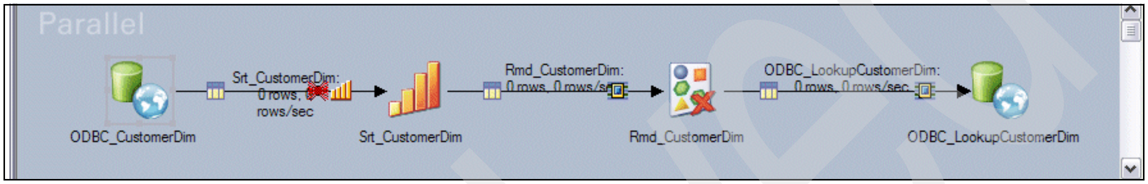


Figure 2-101 Sort stage example 1/6

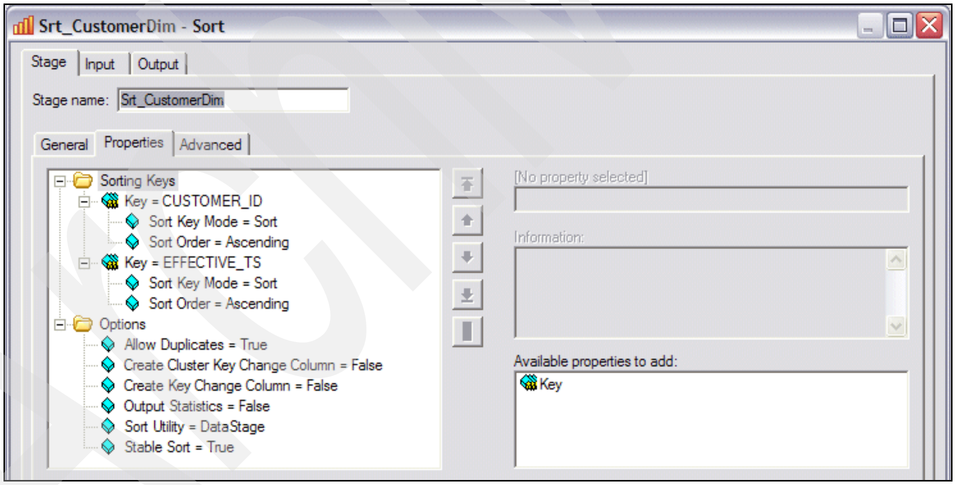


Figure 2-102 Sort stage example 2/6

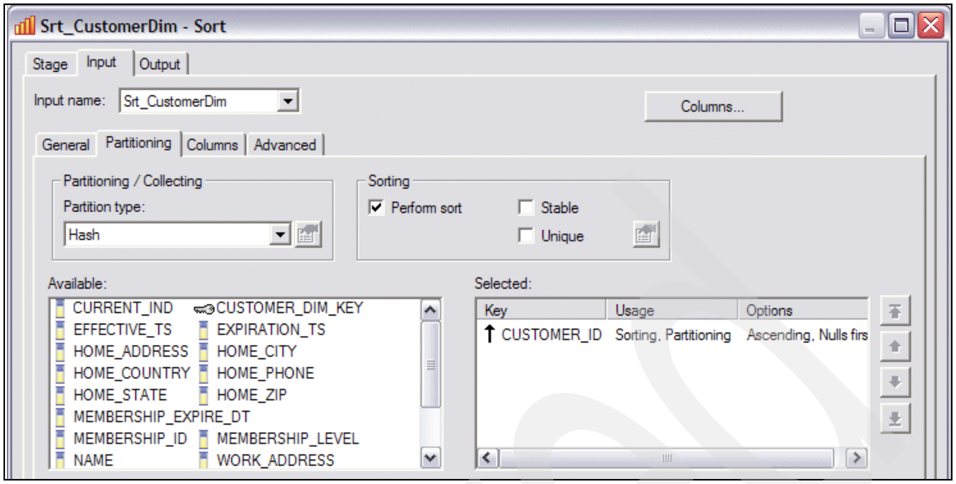


Figure 2-103 Sort stage example 3/6

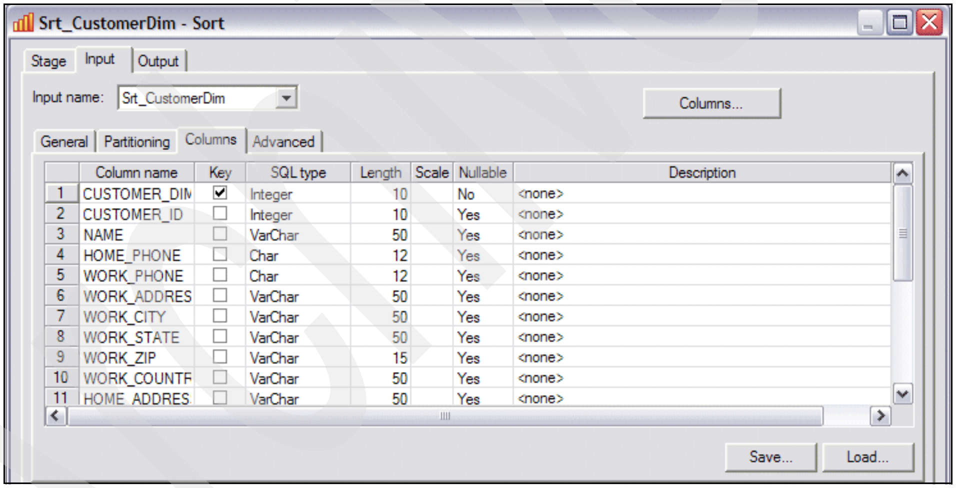


Figure 2-104 Sort stage example 4/6

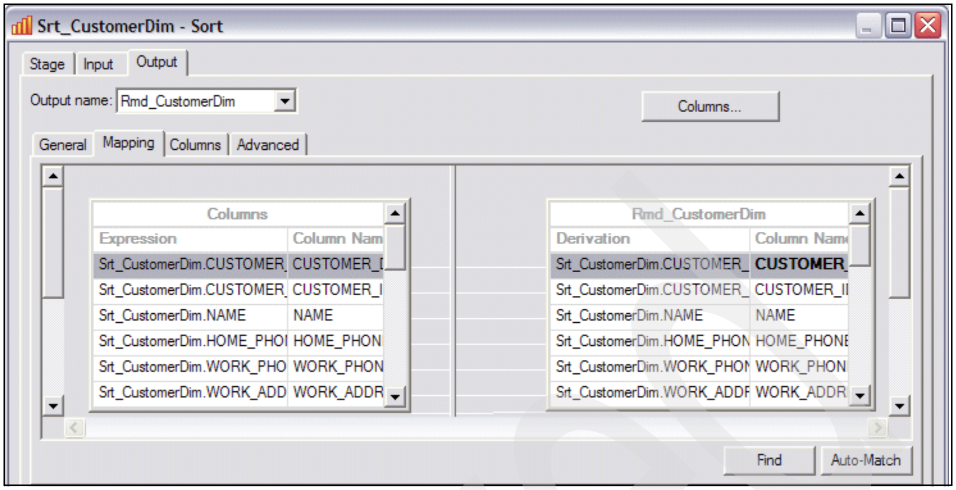


Figure 2-105   Sort stage example 5/6

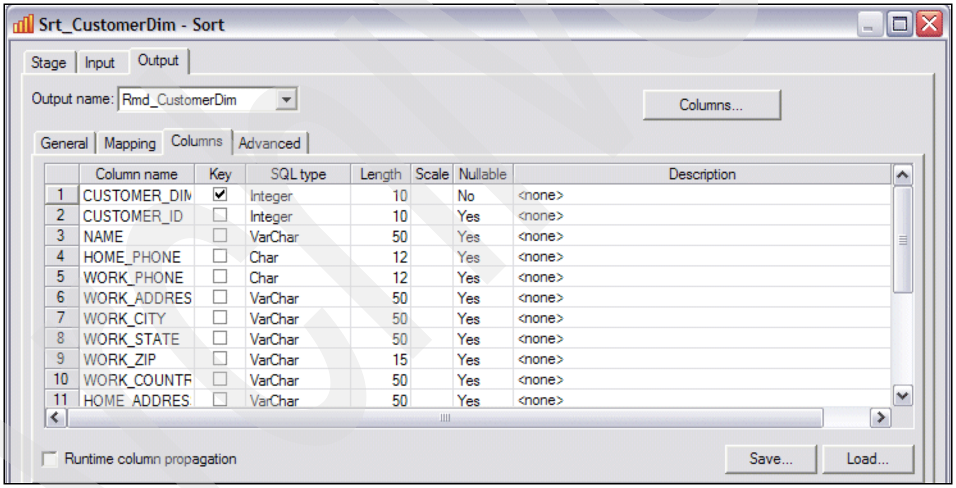


Figure 2-106   Sort stage example 6/6

2.16 Surrogate Key Generator

The Surrogate Key Generator stage is a processing stage that generates surrogate key columns and maintains the key source.

A surrogate key is a unique primary key that is not derived from the data that it represents, therefore changes to the data will not change the primary key. In a star schema database, surrogate keys are used to join a fact table to a dimension table.

The Surrogate Key Generator stage can have a single input link, a single output link, both an input link and an output link, or no links.

Job design depends on the purpose of the stage. You can use a Surrogate Key Generator stage to perform the following tasks:

- Create or delete the key source before other jobs run
- Update a state file with a range of key values
- Generate surrogate key columns and pass them to the next stage in the job
- View the contents of the state file

Generated keys are unsigned 64-bit integers. The key source can be a state file or a database sequence.

> **Note:** You can use the Surrogate Key Generator stage to update a state file, but not a database sequence. Sequences must be modified with database tools.

Figure 2-107 through Figure 2-109 show an example of the configuration of a Surrogate Key Generator stage in a job ("J12_IL_GenerateSurrogateKey" on page 335 in the retail industry scenario described in "Retail industry scenario" on page 138), as follows:

1. Figure 2-107 shows the job that creates a surrogate key source and updates its state. This is described in "J12_IL_GenerateSurrogateKey" on page 335 and is not repeated here. Instead, we only focus on the configuration of the Surrogate Key Generator stage in this job.

Figure 2-108 shows the **Properties** tab in the Stage page, which allows you to specify properties that determine what the stage actually does.

The Key Source category Input Column Name property identifies the input column to update the state file. This column usually is the surrogate key column — PRODUCT_DIM_KEY in this case.

The Key Source Update Action property is set to Create and Update since the state file does not exist in our case and we want to create it.

The Source Name property identifies the file and set the Source Type property to Flat File.

We let the **Advanced** tab and **Partitioning** tab properties default — this is not shown here.

2. Figure 2-109 shows the **Columns** tab in the Input page for the Skg_ProductDim link Input page, which allows you to provide the column definitions of the input data.

Figure 2-107 Surrogate Key Generator stage example 1/3

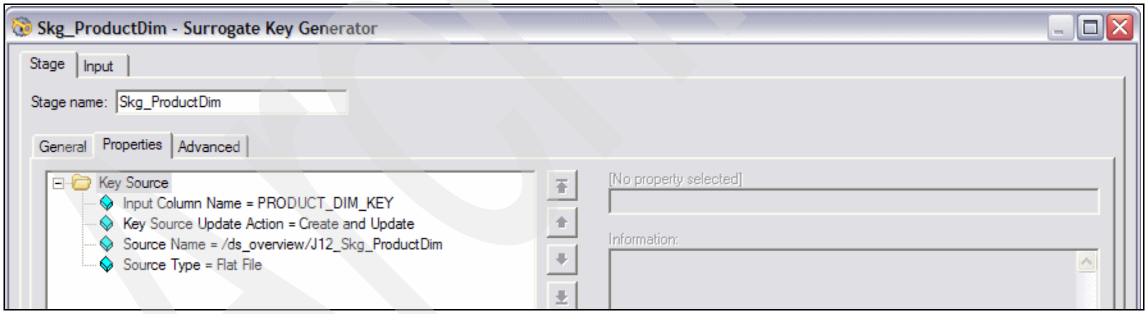


Figure 2-108 Surrogate Key Generator stage example 2/3

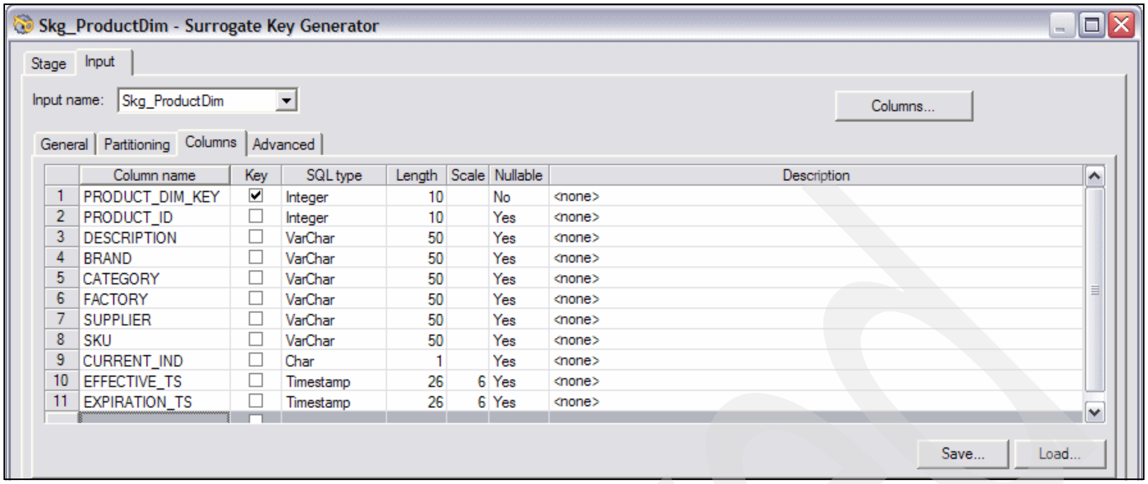


Figure 2-109 Surrogate Key Generator stage example 3/3

2.17 Transformer

The Transformer stage is a processing stage. It appears under the processing category in the tool palette. Transformer stages allow you to create transformations to apply to your data. These transformations can be simple or complex and can be applied to individual columns in your data. Transformations are specified using a powerful set of functions such as date & time, logical, mathematical, null handling, number, raw, string, vector, type conversions, type casting, and utility functions. For complete details of these functions, refer to *IBM WebSphere DataStage and QualityStage Parallel Job Developer Guide*, SC18-9891-00.

Transformer stages can have a single input and any number of outputs. It can also have a reject link, which takes any rows that have not been written to any of the outputs links by reason of a write failure or expression evaluation failure. This is shown in Figure 2-110.

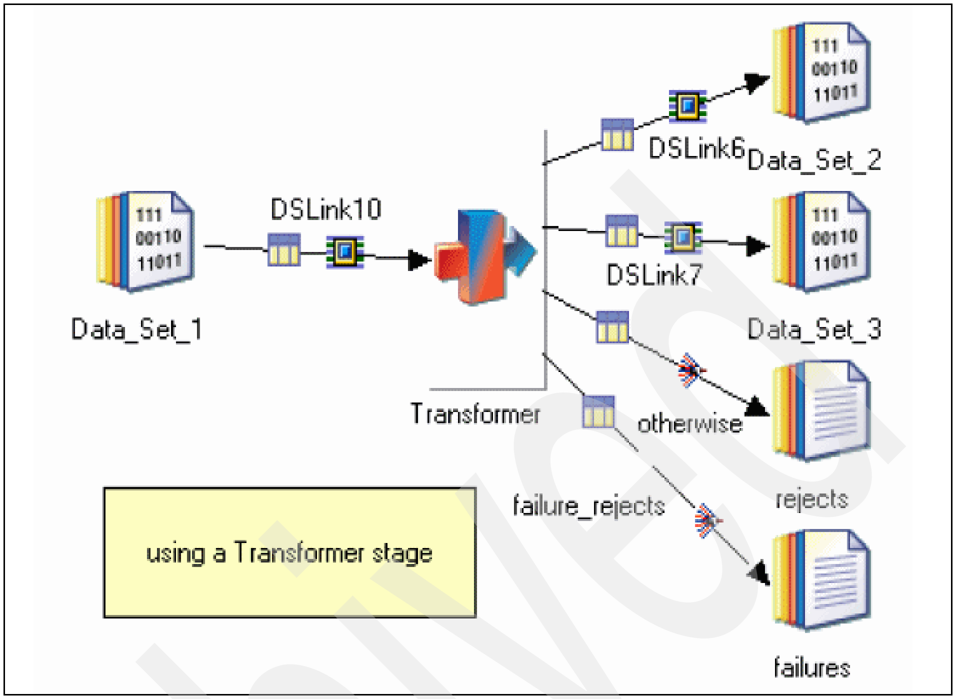


Figure 2-110   Transformer stage

You might want to pass some data straight through the Transformer stage unaltered, but it is likely that you will want to transform data from some input columns before outputting it from the Transformer stage.

You can specify such an operation by entering a transform expression. The source of an output link column is defined in that column's Derivation cell within the Transformer Editor. You can use the Expression® Editor to enter expressions in this cell. You can also simply drag an input column to an output column's Derivation cell, to pass the data straight through the Transformer stage.

In addition to specifying derivation details for individual output columns, you can also specify constraints that operate on entire output links. A constraint is an expression that specifies criteria that data must meet before it can be passed to the output link. You can also specify a constraint otherwise link, which is an output link that carries all the data not output on other links, that is, columns that have not met the criteria.

Each output link is processed in turn. If the constraint expression evaluates to TRUE for an input row, the data row is output on that link. Conversely, if a constraint expression evaluates to FALSE for an input row, the data row is not output on that link.

Constraint expressions on different links are independent. If you have more than one output link, an input row may result in a data row being output from some, none, or all of the output links.

You can also specify another output link, which takes rows that have not been written to any other links because of write failure or expression evaluation failure. This is specified outside the stage by adding a link and converting it to a reject link. This link is not shown in the Transformer metadata grid, and derives its metadata from the input link. Its column values are those in the input row that failed to be written.

> **Note:** If you have enabled Runtime Column Propagation for an output link, you do not have to specify metadata for that link. IBM InfoSphere DataStage is flexible about metadata. It can cope with the situation where metadata is not fully defined. You can define part of your schema and specify that, if your job encounters extra columns that are not defined in the metadata when it actually runs, it will adopt these extra columns and propagate them through the rest of the job. This is known as runtime column propagation (RCP).
>
> This can be enabled for a project via the IBM InfoSphere DataStage and QualityStage Admin, and set for individual links via the Output Page Columns tab for most stages, or in the Output page General tab for Transformer stages. You should always ensure that runtime column propagation is turned on if you want to use schema files to define column metadata.

Figure 2-111 and Figure 2-112 show an example of the configuration of a Transformer stage in a job (“J03_IL_LoadProductDim” on page 202 in the retail industry scenario described in “Retail industry scenario” on page 138), as follows:

1. Figure 2-111 shows the job that initially loads the Product dimension table. This is described in “J03_IL_LoadProductDim” on page 202 and is not repeated here. Instead, we only focus on the configuration of the Transformer stage in this job.
2. Figure 2-112 shows the Trim function being used to remove trailing blanks in all the input columns before being written to the output link Odbc_ProductDim.

   However, for the SKU column, a constraint[8] is defined that the raw length of the value in this field must exceed 5 bytes before it can be passed to the output link.

[8] A constraint is an expression that specifies criteria that data must meet before it can be passed to the output link.

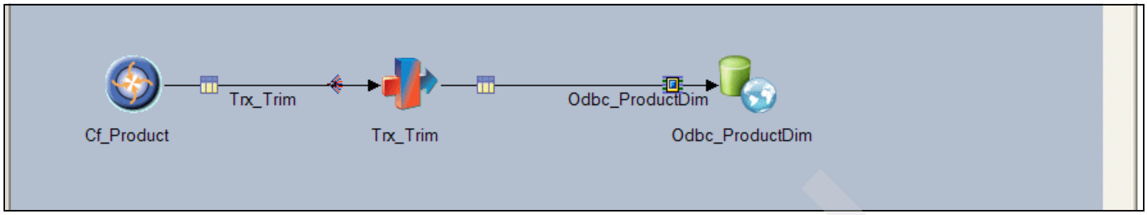


Figure 2-111 Transformer stage example 1/2

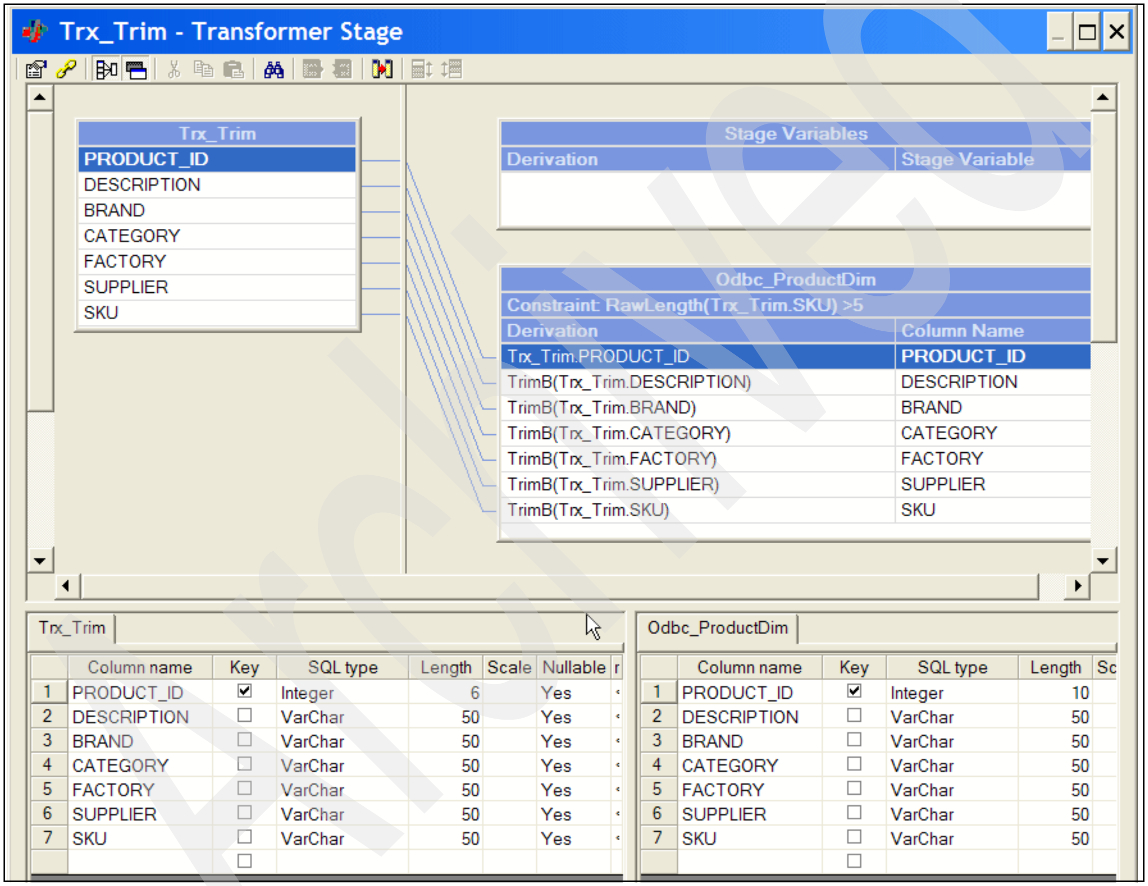


Figure 2-112 Transformer stage example 2/2

3

Retail industry scenario

In this chapter we use a “real world” retail industry scenario to demonstrate a typical star-schema data warehousing flow using IBM InfoSphere DataStage. Included in the flow are the Complex Flat File, Distributed Transaction Stage, and Slowly Changing Dimension stage.

3.1 Retail industry scenario

In this scenario, we use a "real world" retail industry scenario to demonstrate a typical star-schema data warehousing flow using IBM InfoSphere DataStage. Hopefully, you can then extrapolate/customize this process flow to address the unique star-schema data warehousing requirements of your organization.

Our scenario assumes a fictitious national department store named WantThatStuff, which decides to build a star-schema based sales analysis data warehouse with the dimensions of customer, store, product, and date. Over time, dimension attributes are expected to change, and the requirement is to preserve versions of these changes in the star-schema data warehouse in order to deliver accurate results with queries that relate to prior versions of dimension attributes.

The data source for the star-schema is an OLTP system on a z/OS platform. Customer information, Employee information and SALESTRANS information (one SALESTRANS file per store) are stored in sequential files, while product information and store information are stored in VSAM files.

> **Note:** In the real world, the mainframe OLTP systems is more likely to be DBMS based (such as IMS™ or DB2 for z/OS). However, we wanted to showcase the sequential file and VSAM access capabilities of this solution and hence chose them as the OLTP sources in this scenario.

Figure 3-1 provides an overview of the retail industry scenario environment. It starts with WantThatStuff having two stores initially and then expanding with another store (indicated by a dotted line) sometime after the star-schema data warehouse is populated.

Each store's sales transactions is assumed to be collected locally and then transferred to the mainframe at the end of the store's business day, presumably using file transfer protocol. These files are subsequently moved (using file transfer protocol) to the IBM InfoSphere DataStage server for processing.

A number of processes are required to extract and transform the data in the source OLTP systems before it can be used to update the star-schema data warehouse. These processes are collectively grouped together in Figure 3-1 as "Pre-process data on Linux platform prior to updating star-schema data warehouse".

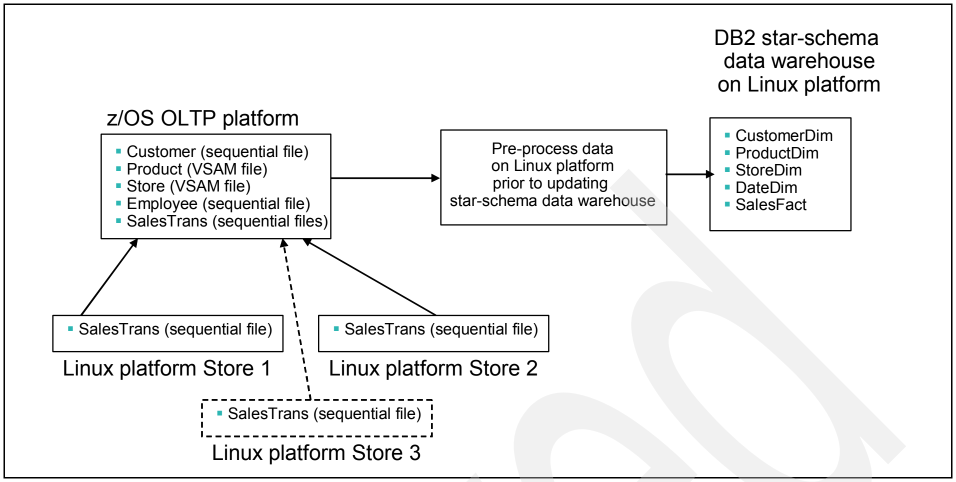


Figure 3-1 Retail industry scenario overview for WANTTHATSTUFF

A description of the data model of the CUSTOMER, PRODUCT, STORE, EMPLOYEE, and SALESTRANS entities is shown here in Figure 3-2, while that of the star-schema is shown in Figure 3-3 on page 144.

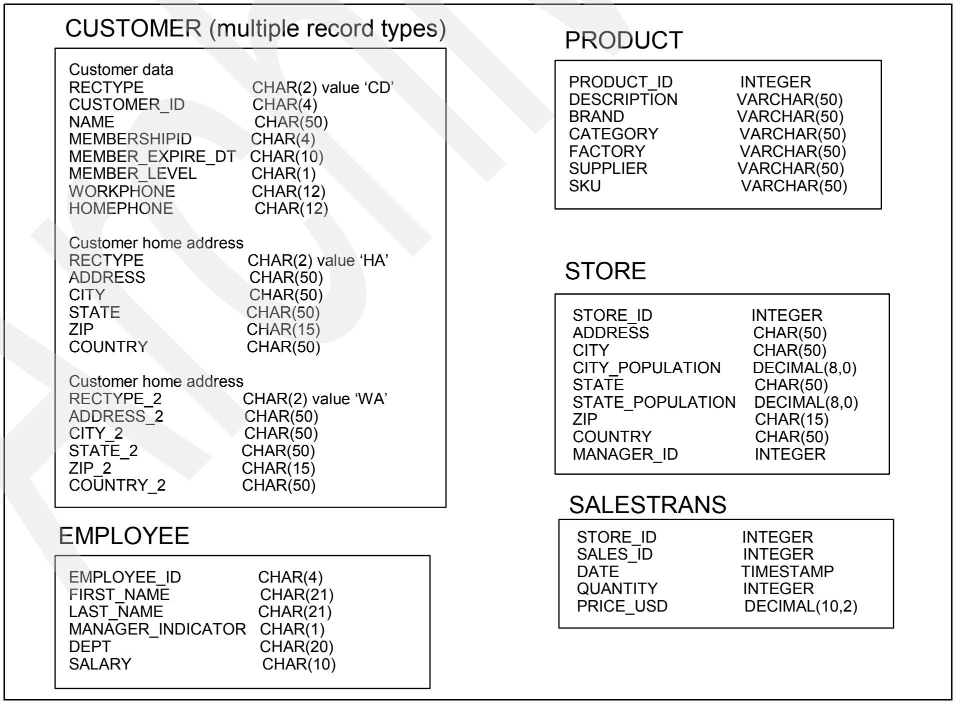


Figure 3-2 WantThatStuff source OLTP data model

We make the following assumptions about this retail industry scenario:

- The star-schema data warehouse is updated daily.
- Changes to dimension tables are captured in the relevant operational systems and fed into an IBM WebSphere MQ message queue.
- Transactions associated with Late Arriving Dimensions (LAD) are to be rejected and written to a separate file for analysis and subsequent re-processing.

  LAD corresponds to a scenario where a transaction contains a dimension business key (such as a customer number or a store id) that has not yet been inserted into the corresponding dimension table.
- Late Arriving or Non-Arriving Data (LANAD) changes should be processed — in other words, the dimension tables should reflect these changes even if there are no transactions referencing them.

  LANAD corresponds to a scenario where changes are processed for a dimension key or attribute, and there are no transactions corresponding to these dimension changes.
- Dimension table changes may arrive in the following combinations for a given business key:
  - Changes to a single dimension table may contain more than one Type 1 attribute change.
  - Changes to a single dimension table may contain more than one Type 2 attribute change.
  - Changes to a single dimension table may contain a mix of Type 1 and Type 2 attribute changes.

  > **Note:** The norm is more likely to be the absence of any dimension table changes during a particular daily update cycle.

The following tasks must be performed to achieve WantThatStuff's business objectives:

- One time tasks (Day 0) involve the following actions:

  a. Designing the star-schema.

  b. Populating the dimension tables and fact table.

  > **Note:** The loading of the DATE dimension table is not shown here, because the input for it does not come from the OLTP systems, but is generated directly from a calendar.

  c. Setting up for the recurring tasks.

- Recurring tasks (Day 1, Day 2, and Day 3) involve capturing dimension table changes and the sales transactions and preparing the information for updating the dimension tables and fact table over multiple update cycles as follows:

  a. Capture dimension table changes occurring in the operational OLTP systems.

  b. Collect sales transactions from the stores from the operational OLTP systems.

  c. Prepare the changes to the dimension table for updating the dimension tables.

  d. Prepare the sales transactions for updating the fact table.

  e. Update the dimension tables.

  f. Update the fact table.

  As mentioned earlier, the update cycle is daily.

> **Attention:** As mentioned earlier, in all the following sections, to avoid overburdening you with excessive screen captures, we have *not* included all the panels that you would typically navigate through in order to perform the desired function. Instead we have focused on including select screen captures (and in some cases, just portions of them) that highlight the key items of interest, thereby skipping both initial screen captures, as well as some intervening ones in the process. Screen captures involving default values are not shown here either. And finally, also not covered is a discussion of each property of the stages, since they are all well described in the *IBM WebSphere DataStage and QualityStage Parallel Job Developer Guide*, SC18-9891-00.

3.1.1 One time tasks (Day 0)

WantThatStuff designed the following series of steps to perform these tasks:

1. The star-schema for the data warehouse is shown in Figure 3-3. It shows four dimension tables CUSTOMER, PRODUCT, STORE and DATE, and the SALES fact table.

   The Type 1 and Type 2 columns were identified as follows:

   – Type 1

     • Customer dimension table columns HOME_PHONE, WORK_PHONE, NAME, HOMEADDRESS, and WORKADDRESS

     • Store dimension table columns CITY_POPULATION and STATE_POPULATION

– Type 2

  - Customer dimension table columns MEMBERSHIP_EXPIRE_DT and MEMBERSHIP_LEVEL
  - Store dimension table column MANAGER_NAME

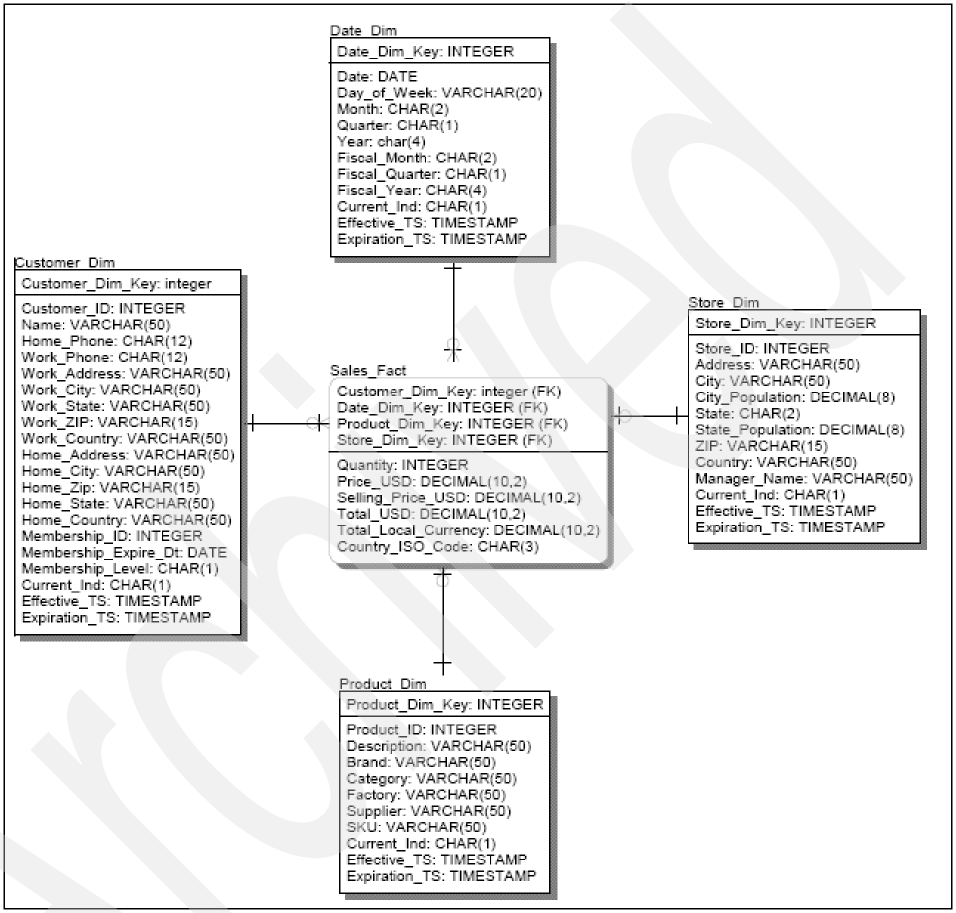


Figure 3-3 Star-schema of WantThatStuff's data warehouse

2. All applications, operations, and services are associated with a project as shown in Figure 3-4 on page 147. Therefore, you first have to create a project before you can define any applications, operations or services. A project is a collaborative environment that you use to design applications, services, and operations.

   "J0A_Create a project" on page 147 performs this step by creating the DS_Overview project.

3. After the DS_Overview project has been created, you have to import all the table definitions required by the IBM InfoSphere DataStage jobs into the IBM Information Server metadata repository. These include the star-schema tables, and the some of the intermediate tables used in the retail industry scenario.

   “J0B_Import table definitions into repository from DB2 using ODBC” on page 154 performs this step.

4. A number of other prerequisites must be installed and configured prior to the initial load of the star-schema database such as these:

   – Create an IBM WebSphere MQ queue manager for use by the Distributed Transaction stage. This is described in “Create the Queue Manager” on page 580.

   – Set up the XA parameters on the Queue Manager for use by the Distributed Transaction stage. This is described in “Set up the XA parameters on Queue Manager” on page 587.

   – Use the Classic Data Architect (CDA) of IBM Information Integrator Classic Federation to configure access to the VSAM files on the mainframe as relational tables on the Linux platform where IBM InfoSphere DataStage is installed. This is described in “Configuration of Classic Data Architect” on page 574.

5. Table 3-1 lists the IBM InfoSphere DataStage jobs we created to perform the one time tasks identified earlier. These were performed on November 5th, 2007.

Table 3-1 One time tasks jobs

Job name	Brief description
“J01_IL_FTPCustomerFile” on page 159	Transfers Customer file from the mainframe to the Linux platform
“J02_IL_LoadCustomerDim” on page 184	Loads the Customer dimension table
“J03_IL_LoadProductDim” on page 202	Loads the Product dimension table
“J04_IL_FTPEmployeeFile” on page 209	Transfers Employee file from the mainframe to the Linux platform
“J05_IL_LoadStoreDim” on page 219	Loads the Store dimension table
“J06_IL_Daily_CreateCurrencyLookup_Service” on page 227	Performs a lookup of the daily currency exchange rate
“J07_IL_Daily_LoadSalesStore” on page 282	Loads the daily sales transactions of a store to a table
“J08_IL_LoadSalesFact” on page 292	Loads the Sales fact table

Job name	Brief description
“J09_IL_LoadLookupCustomerDim” on page 320	Creates interim table for Customer dimension keys
“J10_IL_LoadLookupProductDim” on page 327	Creates interim table for Product dimension keys
“J11_IL_LoadLookupStoreDim” on page 330	Creates interim table for Store dimension keys
“J12_IL_GenerateSurrogateKey” on page 335	Creates the surrogate key files SCD handling

Each of these jobs is briefly described here:

– The Complex Flat File stage processes single and multiple record type sequential files, but the restriction is that the sequential file must be on the same server as the IBM InfoSphere DataStage server. Therefore, in order to process the Customer and Employee files (that reside on the mainframe), we musthave to first transfer these EBCDIC files from the mainframe on to the Linux platform using the FTP Enterprise stage as described in “J01_IL_FTPCustomerFile” on page 159 and “J04_IL_FTPEmployeeFile” on page 209.

– Once the Customer and Employee files have been transferred to the Linux platform, we can use the Complex Flat File stage to process its contents. We have to consolidate the information from the multiple record types (relating to a single customer) into a single record and the single record type of the Employee file for populating the CustomerDim and StoreDim tables respectively. “J02_IL_LoadCustomerDim” on page 184, “J03_IL_LoadProductDim” on page 202 and “J05_IL_LoadStoreDim” on page 219 perform the steps of loading the dimension tables.

– Some customers use non-US credit cards to purchase products at the various WantThatStuff stores. The individual sales transactions captured at the individual stores are in $US, but the foreign currency equivalent must be determined and then loaded into an interim DB2 table for subsequent loading into the Sales fact table. These steps are performed by “J06_IL_Daily_CreateCurrencyLookup_Service” on page 227, “J07_IL_Daily_LoadSalesStore” on page 282, and “J07A_SharedContainerLookupCurrency” on page 273.

– The sales transactions (in the interim DB2 tables) from the various stores are then merged, aggregated, and assigned the appropriate surrogate key (corresponding to the business key) before being loaded into the Sales fact table as described in “J08_IL_LoadSalesFact” on page 292.

– As mentioned earlier, the sales transactions must have surrogate keys assigned to them before being loaded into the Sales fact table. Dimension lookup tables and surrogate key files must be generated to provide this information. “J09_IL_LoadLookupCustomerDim” on page 320, “J10_IL_LoadLookupProductDim” on page 327, “J11_IL_LoadLookupStoreDim” on page 330, and “J12_IL_GenerateSurrogateKey” on page 335 performs these steps.

At the completion of these one time tasks on November 5th, 2007, you can proceed to processing the recurring tasks as described in 3.1.2, “Recurring tasks” on page 341.

J0A_Create a project

IBM Information Server is a project-based development environment as shown in Figure 3-4.

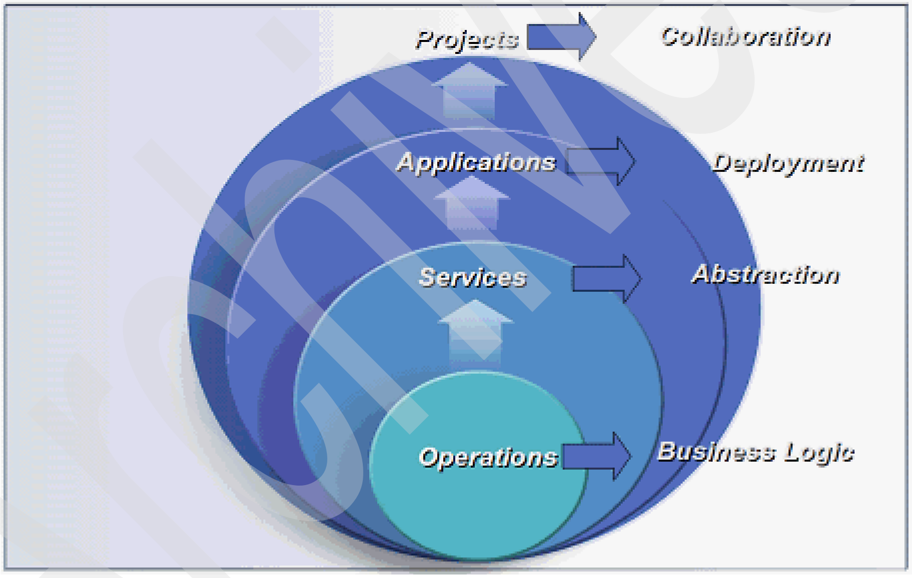


Figure 3-4 IBM Information Server development paradigm

All applications, operations, and services are associated with a project as shown in Figure 3-4. Therefore, you first have to create a project before you can define any applications, operations, or services. A project is a collaborative environment that you use to design applications, services, and operations. All project information that you create is saved in the common metadata repository so that it can easily be shared among other IBM Information Server components. For our retail industry scenario, we created a project named DS_Overview.

Jobs define the sequence of steps that determine how IBM Information Server performs its work. After they are designed, jobs are compiled and run on the parallel processing engine.

Each time that an IBM InfoSphere DataStage job is validated, run, or scheduled, you can set options to change parameters, override default limits for row processing, assign invocation IDs, and set tracing options. When you have a large number of jobs that run with the same parameters, it is more efficient to create a parameter set object once and have it reused by all the jobs.

Figure 3-5 on page 149 through Figure 3-14 on page 153 describe the steps in creating the DS_Overview project in IBM Information Server, as follows:

1. Launch the IBM InfoSphere DataStage and QualityStage Administrator program by clicking **Start** → **All Programs** → **IBM Information Server** → **IBM WebSphere DataStage and QualityStage Administrator** as shown in Figure 3-5 on page 149.
2. Attach to the DataStage server KAZAN.ITSOSJ.SANJOSE.IBM.COM at domain 9.43.86.77:9080 with username (admin) and appropriate password as shown in Figure 3-6 on page 149. Click **OK**.
3. Under the Projects tab, click **Add** to add a new project as shown in Figure 3-7 on page 150.
4. Provide the Name (DS_Overview) in Figure 3-8 on page 150 and click **OK**.
5. Figure 3-9 on page 151 through Figure 3-14 on page 153 show the definition of user-defined environment variables for this DS_Overview project and the project's successful creation.

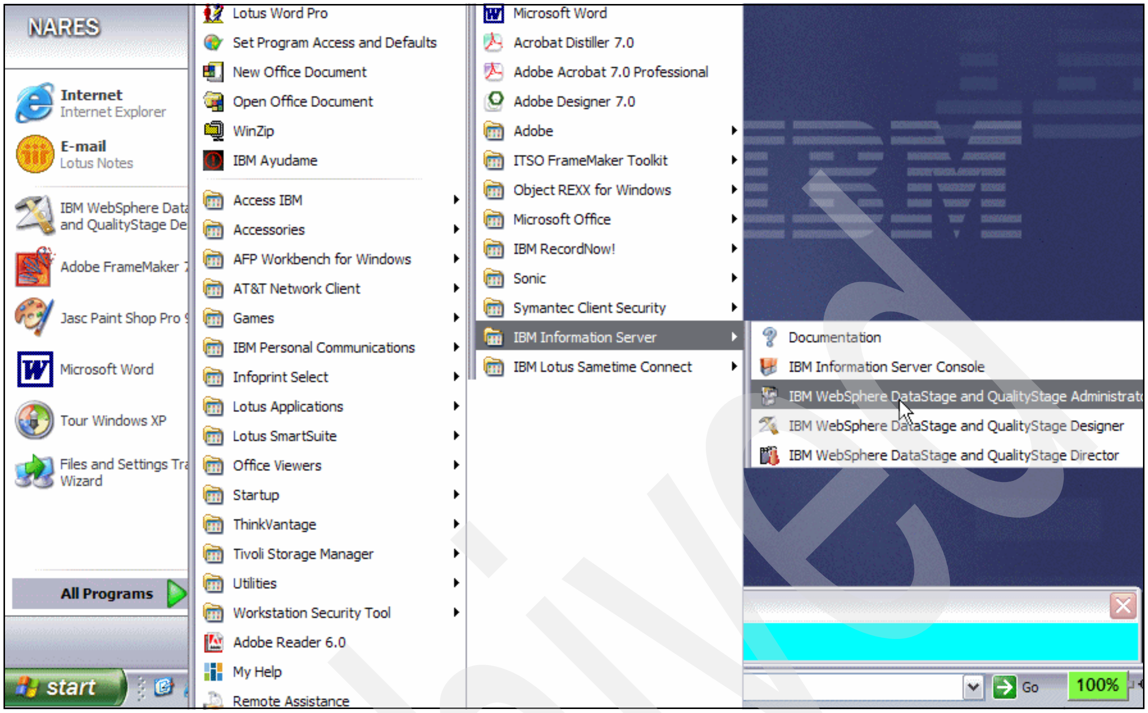


Figure 3-5 Create the DS_Overview project 1/10

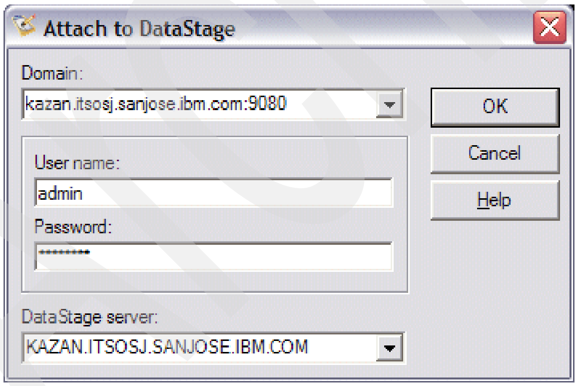


Figure 3-6 Create the DS_Overview project 2/10

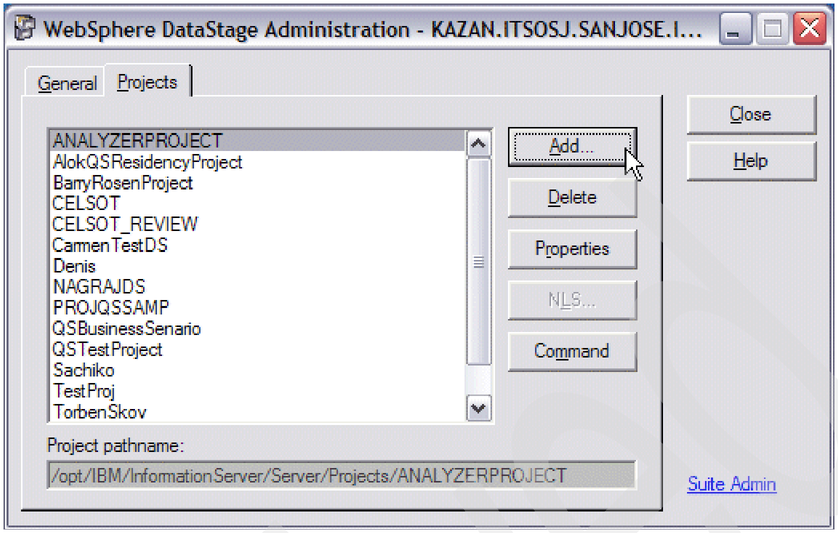


Figure 3-7 Create the DS_Overview project 3/10

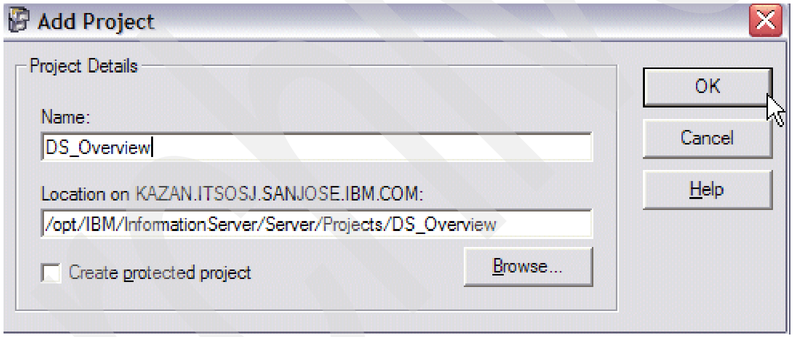


Figure 3-8 Create the DS_Overview project 4/10

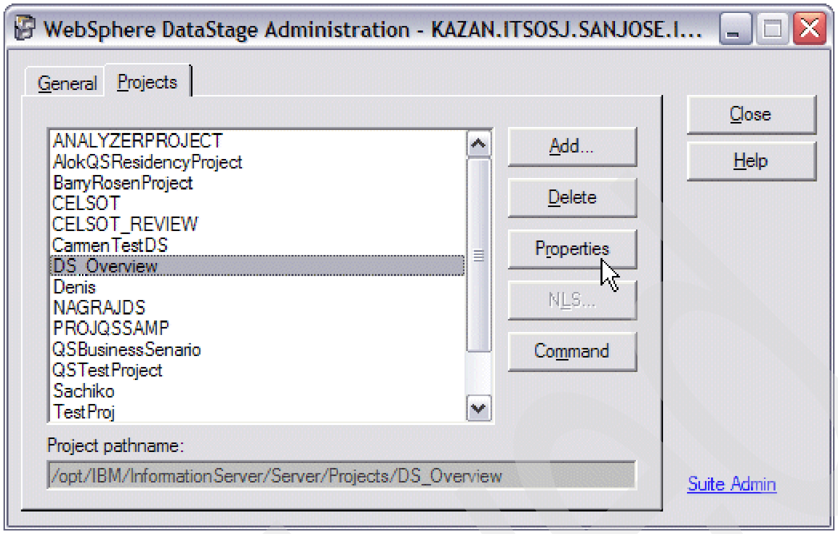


Figure 3-9 Create the DS_Overview project 5/10

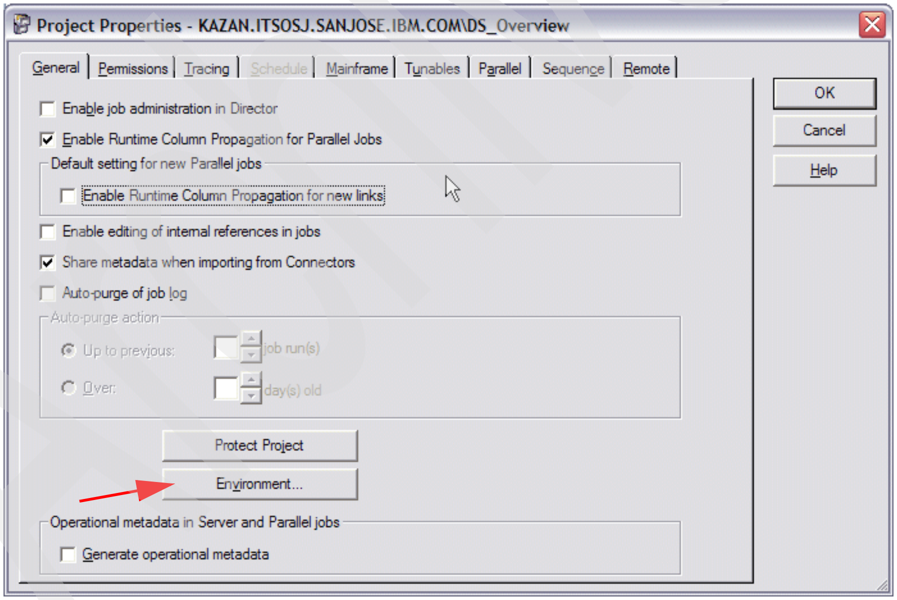


Figure 3-10 Create the DS_Overview project 6/10

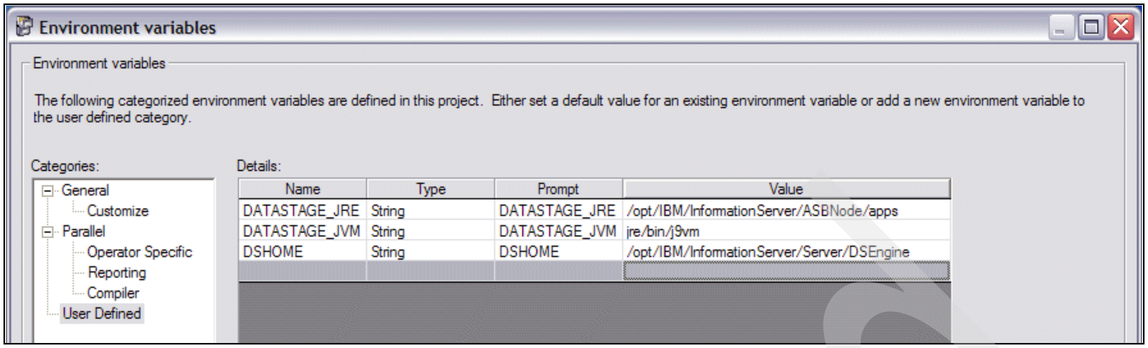


Figure 3-11   Create the DS_Overview project 7/10

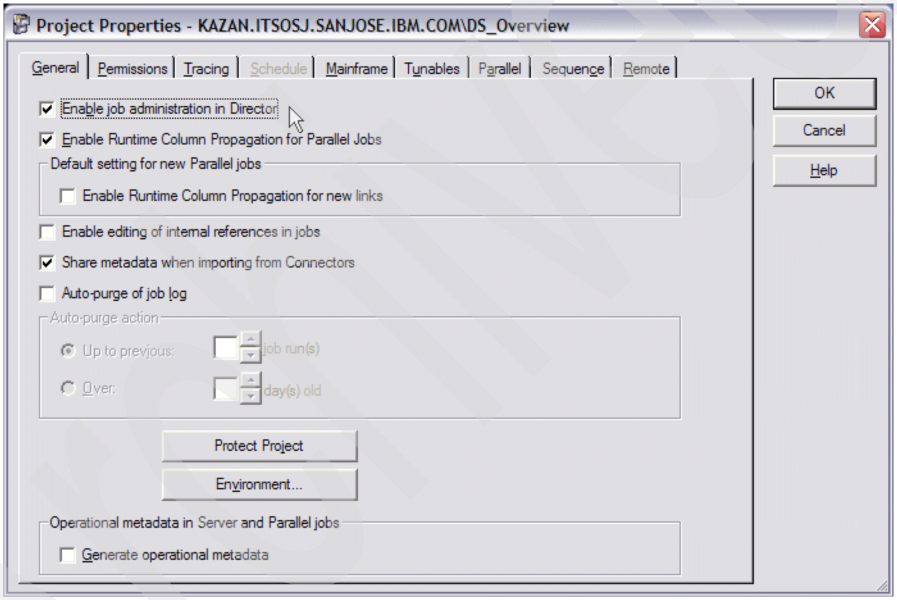


Figure 3-12   Create the DS_Overview project 8/10

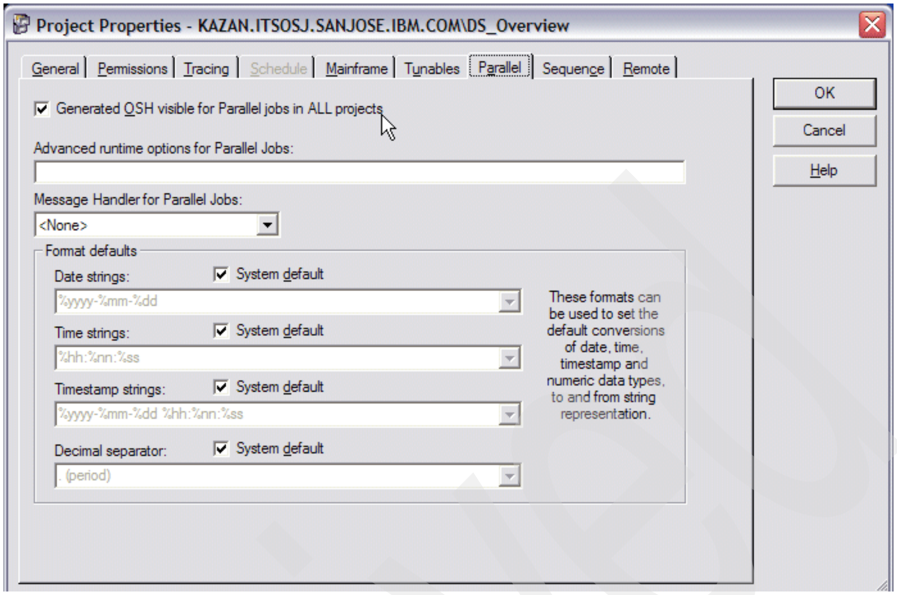


Figure 3-13 Create the DS_Overview project 9/10

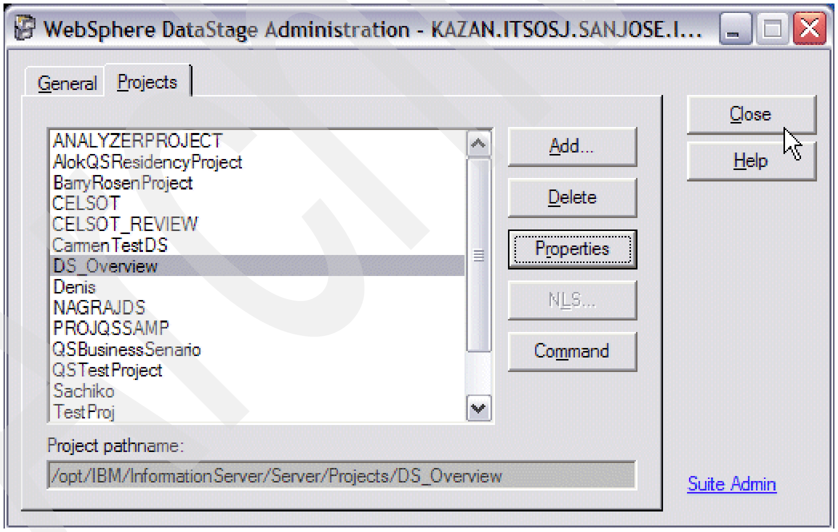


Figure 3-14 Create the DS_Overview project 10/10

J0B_Import table definitions into repository from DB2 using ODBC

You must import metadata into the metadata repository[1] for use in all IBM InfoSphere DataStage projects. You can import all, or selected tables, files, or columns in a schema/directory.

Figure 3-15 on page 155 through Figure 3-21 on page 159 describe the import of the required star-schema and other table definitions using ODBC from the DSSAMPLE database into the metadata repository as follows:

1. After launching the IBM InfoSphere DataStage and QualityStage Designer (similar to that shown in Figure 3-5 on page 149 but selecting **IBM WebSphere DataStage and QualityStage Designer** instead) for the DS_Overview project in the KAZAN.ITSOSJ.SANJOSE.IBM.COM server, select **Import** → **Table Definitions** → **ODBC Table Definitions** from the main menu bar as shown in Figure 3-15 on page 155.
2. In the **Import Meta Data (ODBC)** window, provide access details of the database (DSSAMPLE) containing the tables of interest and click **OK** as shown in Figure 3-16 on page 155.
3. Select all the tables whose definitions you want to import, provide the target folder in the To folder field (\Table Definitions\ODBC\DSSAMPLE) and click **Import** as shown in Figure 3-17 on page 156. Figure 3-18 on page 156 shows the progress of the import.
4. Figure 3-19 on page 157 through Figure 3-21 on page 159 show the properties of the imported table definition of the DS.CUSTOMER_DIM table.

You can now proceed to file transfer the mainframe files to the Linux platform as described in "J01_IL_FTPCustomerFile" on page 159.

[1] The metadata repository (XMETA database) stores imported metadata, project configurations, reports, and results for all components of IBM Information Server.

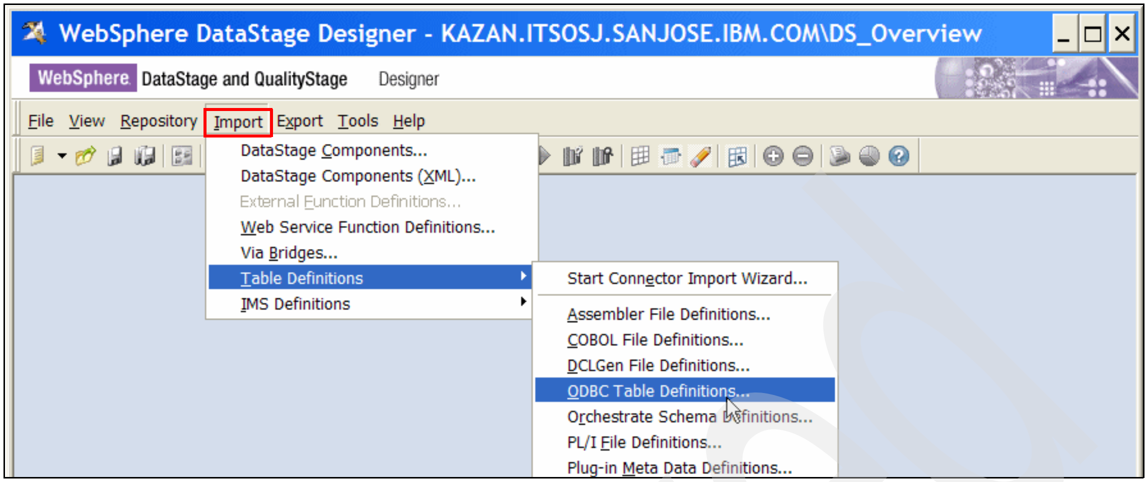


Figure 3-15 Create J0_Import table definitions to repository from DB2: ODBC 1/7

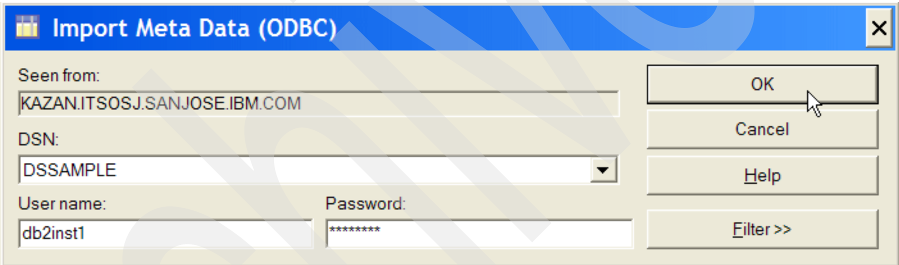


Figure 3-16 Create J0_Import table definitions to repository from DB2: ODBC 2/7

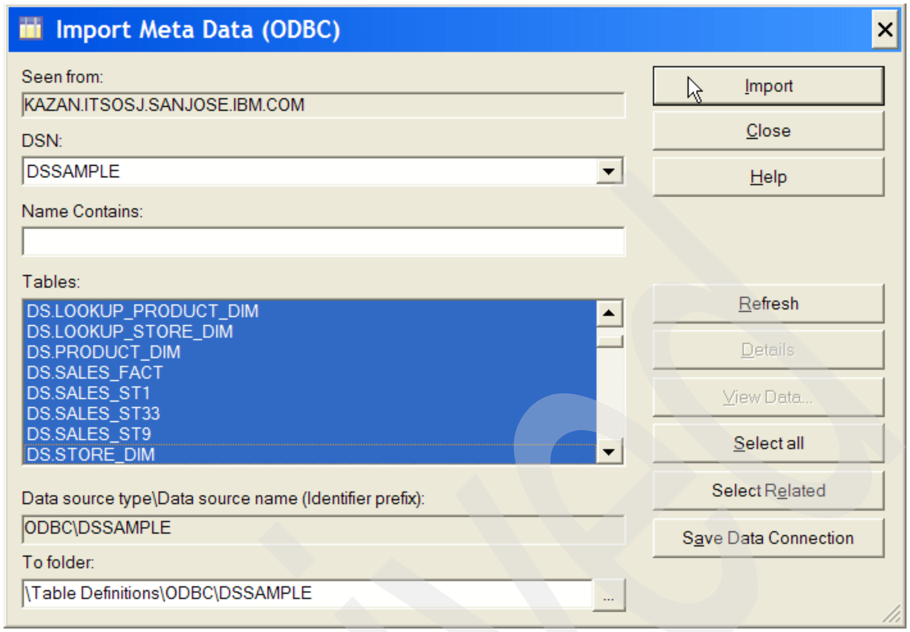


Figure 3-17 Create J0_Import table definitions to repository from DB2: ODBC 3/7

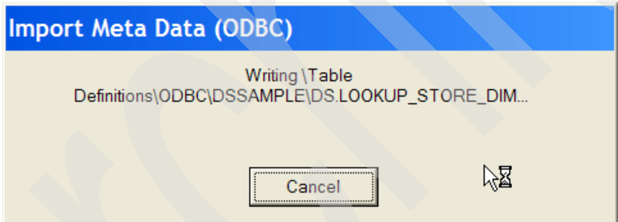


Figure 3-18 Create J0_Import table definitions to repository from DB2: ODBC 4/7

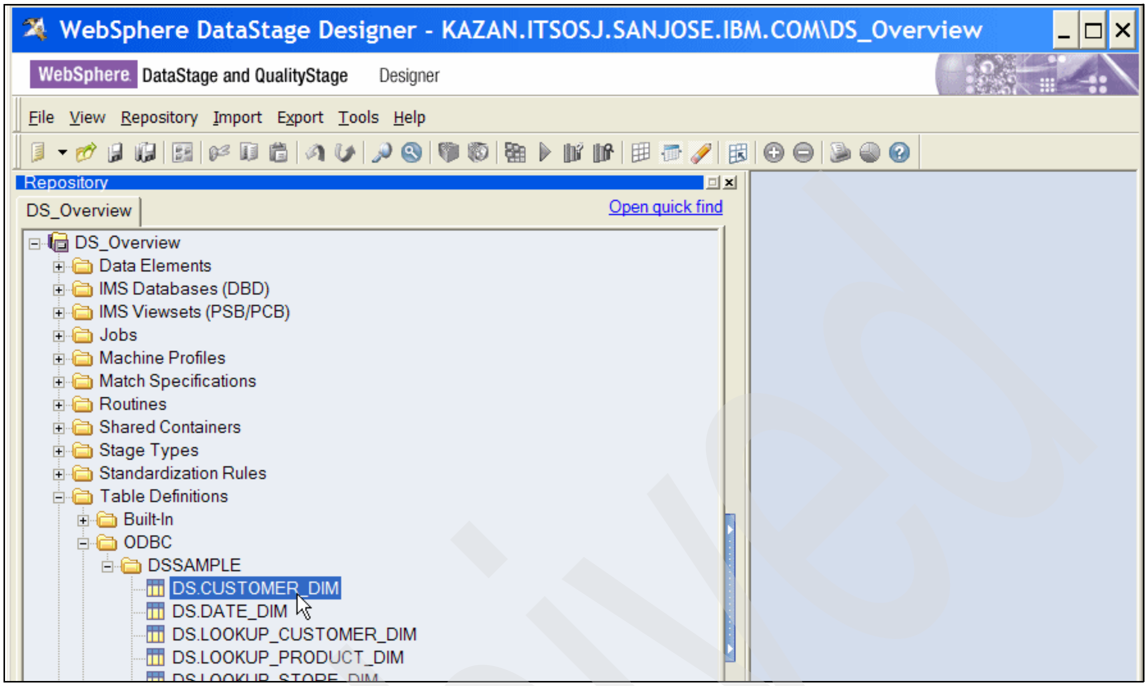


Figure 3-19 Create J0_Import table definitions to repository from DB2: ODBC 5/7

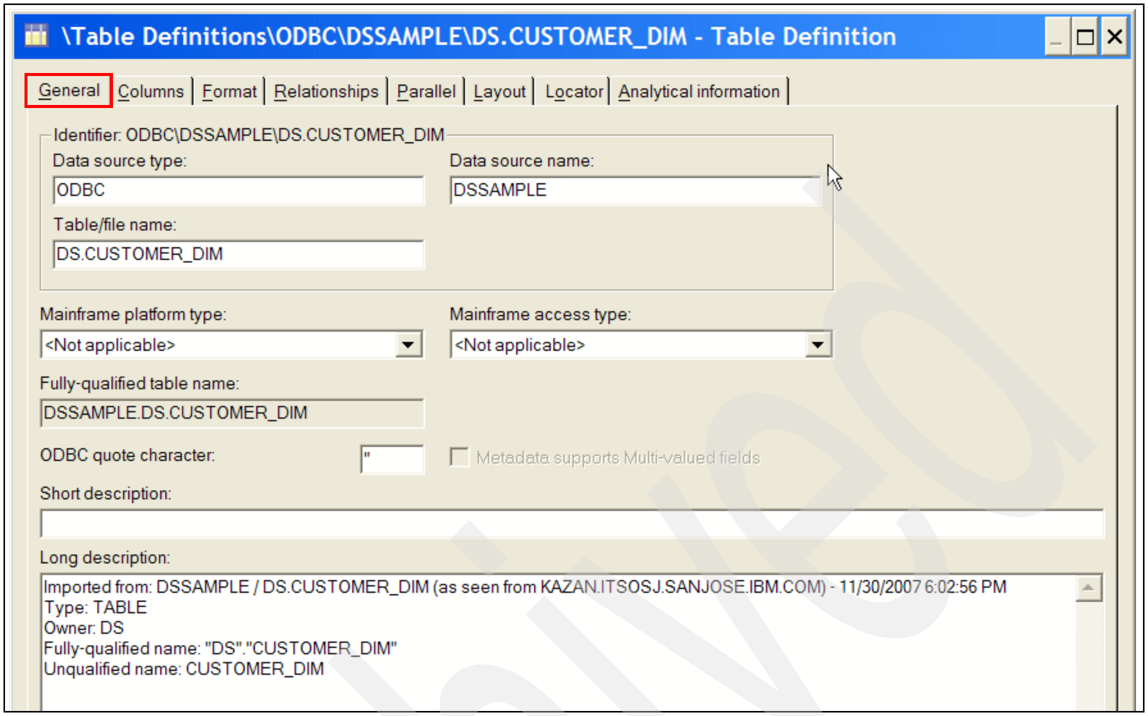
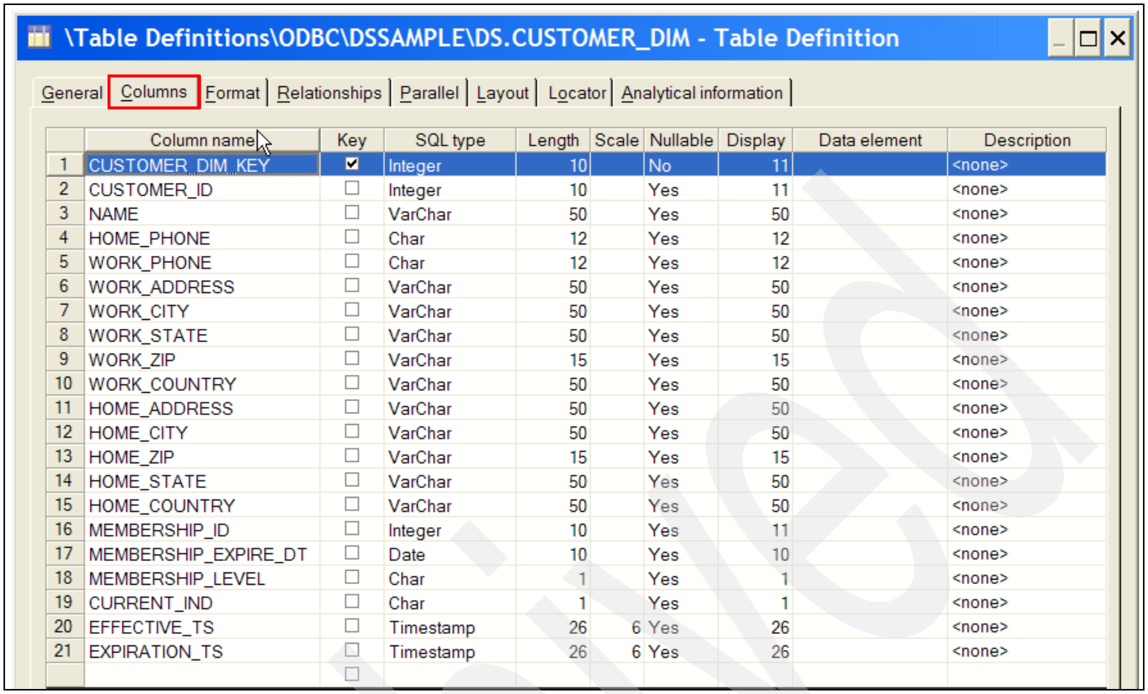


Figure 3-20 Create J0_Import table definitions to repository from DB2: ODBC 6/7



	Column name	Key	SQL type	Length	Scale	Nullable	Display	Data element	Description
1	CUSTOMER_DIM_KEY	☑	Integer	10		No	11		<none>
2	CUSTOMER_ID	☐	Integer	10		Yes	11		<none>
3	NAME	☐	VarChar	50		Yes	50		<none>
4	HOME_PHONE	☐	Char	12		Yes	12		<none>
5	WORK_PHONE	☐	Char	12		Yes	12		<none>
6	WORK_ADDRESS	☐	VarChar	50		Yes	50		<none>
7	WORK_CITY	☐	VarChar	50		Yes	50		<none>
8	WORK_STATE	☐	VarChar	50		Yes	50		<none>
9	WORK_ZIP	☐	VarChar	15		Yes	15		<none>
10	WORK_COUNTRY	☐	VarChar	50		Yes	50		<none>
11	HOME_ADDRESS	☐	VarChar	50		Yes	50		<none>
12	HOME_CITY	☐	VarChar	50		Yes	50		<none>
13	HOME_ZIP	☐	VarChar	15		Yes	15		<none>
14	HOME_STATE	☐	VarChar	50		Yes	50		<none>
15	HOME_COUNTRY	☐	VarChar	50		Yes	50		<none>
16	MEMBERSHIP_ID	☐	Integer	10		Yes	11		<none>
17	MEMBERSHIP_EXPIRE_DT	☐	Date	10		Yes	10		<none>
18	MEMBERSHIP_LEVEL	☐	Char	1		Yes	1		<none>
19	CURRENT_IND	☐	Char	1		Yes	1		<none>
20	EFFECTIVE_TS	☐	Timestamp	26	6	Yes	26		<none>
21	EXPIRATION_TS	☐	Timestamp	26	6	Yes	26		<none>

Figure 3-21 Create J0_Import table definitions to repository from DB2: ODBC 7/7

J01_IL_FTPCustomerFile

In this job, we use the IBM InfoSphere DataStage FTP Enterprise stage to file transfer the CUSTOMER sequential (EBCDIC) file from the mainframe to the Linux platform.

Figure 3-22 on page 161 through Figure 3-66 on page 183 describe the steps using Designer Client to build and execute the DataStage job to perform this task:

The steps are as follows:

1. After launching the IBM InfoSphere DataStage and QualityStage Designer (similar to that shown in Figure 3-5 on page 149, but selecting **IBM WebSphere DataStage and QualityStage Designer** instead), attach to the DS_Overview project in the kazan.itsosj.sanjose.ibm.com server as shown in Figure 3-22 on page 161. Click **OK**.
2. Figure 3-23 on page 162 through Figure 3-31 on page 168 show the creation a parallel job (using drag and drop in the Designer canvas) using the FTP Enterprise stage to transfer a sequential file from one platform to another. The renaming of these stages is also shown here.

3. Figure 3-32 on page 169 through Figure 3-46 on page 175 show the configuration of the FTP Enterprise stage. The Output page allows you to specify details about how the FTP Enterprise stage transfers one or more files from a remote host using the FTP protocol. Figure 3-33 on page 169 through Figure 3-41 on page 173 show the **Properties** tab in the Output page, which allows you to specify properties that determine what the stage actually does.
   - The Source category property URI specifies the pathname connecting the Stage to a source file on a remote host, which corresponds to the Customer file on the mainframe.
   - The Connection category allows you to specify the User name (nalur1) and Password to access the data source identified by the URI.
   - The Transfer Protocol category Transfer Mode property is FTP.
   - The Options category Transfer Type is Binary.
4. Figure 3-42 on page 174 through Figure 3-46 on page 175 show the **Columns** tab in the Output page, which identifies a single column definition for this file named Body of VARCHAR (255). Runtime column propagation is not enabled here.
5. Figure 3-47 on page 176 through Figure 3-54 on page 179 show the configuration of the sequential file to which the FTP Enterprise stage writes. The Input page allows you to specify details about how the Sequential File stage writes data to one or more flat files.
   - The **Properties** tab allows you to specify details of exactly what the link does as shown in Figure 3-48 on page 176. The File property in the Target category defines the flat file that the incoming data will be written to. The File Update Mode property specifies Overwrite to overwrite existing files,
   - The **Formats** tab gives information about the format of the files being written as shown in Figure 3-49 on page 177 through Figure 3-53 on page 178.
     - The Record level properties define details about how data records are formatted in the flat file. The Final delimiter value of end (default) is removed.
     - The Field defaults properties defines the default properties for columns written to the file. These are applied to all columns written, but can be overridden for individual columns from the **Columns** tab. The Quote specifies that variable length fields are enclosed in a double quote.
   - The **Columns** tab specifies the column definitions of data being written. A single column named Body with a VarChar of length 255 is defined as shown in Figure 3-54 on page 179.

6. Clicking the **Run** taskbar to execute this job prompts you to save this job (Figure 3-55 on page 179 through Figure 3-59 on page 181) before execution begins.
7. The execution of this job can be tracked by selecting **Tools** → **Run Director** in the menu as shown in Figure 3-60 on page 181 and Figure 3-61 on page 182. Selecting the J01_IL_FTPCustomerFile job, you can view its log by clicking the **Log** icon in the toolbar as shown in Figure 3-62 on page 182. The successful execution of the job is shown in Figure 3-63 on page 182.
8. The contents of the sequential file can be viewed by right-clicking the sequential file stage and selecting **View Seq_Customer data** as shown in Figure 3-64 on page 183 through Figure 3-66 on page 183. The contents are undecipherable because it is EBCDIC.

The contents of the CUSTOMER file is used to load the CUSTOMER_DIM table as described in “J02_IL_LoadCustomerDim” on page 184.

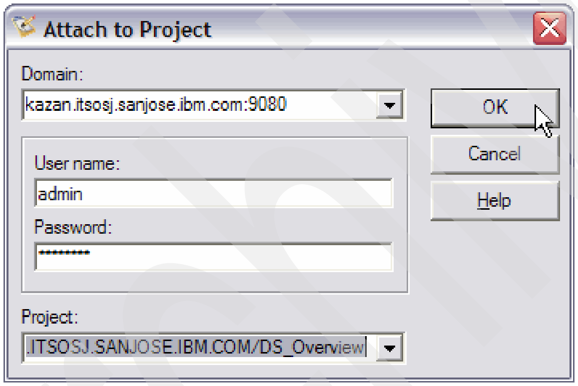


Figure 3-22 Create the J01_IL_FTPCustomerFile job 1/45

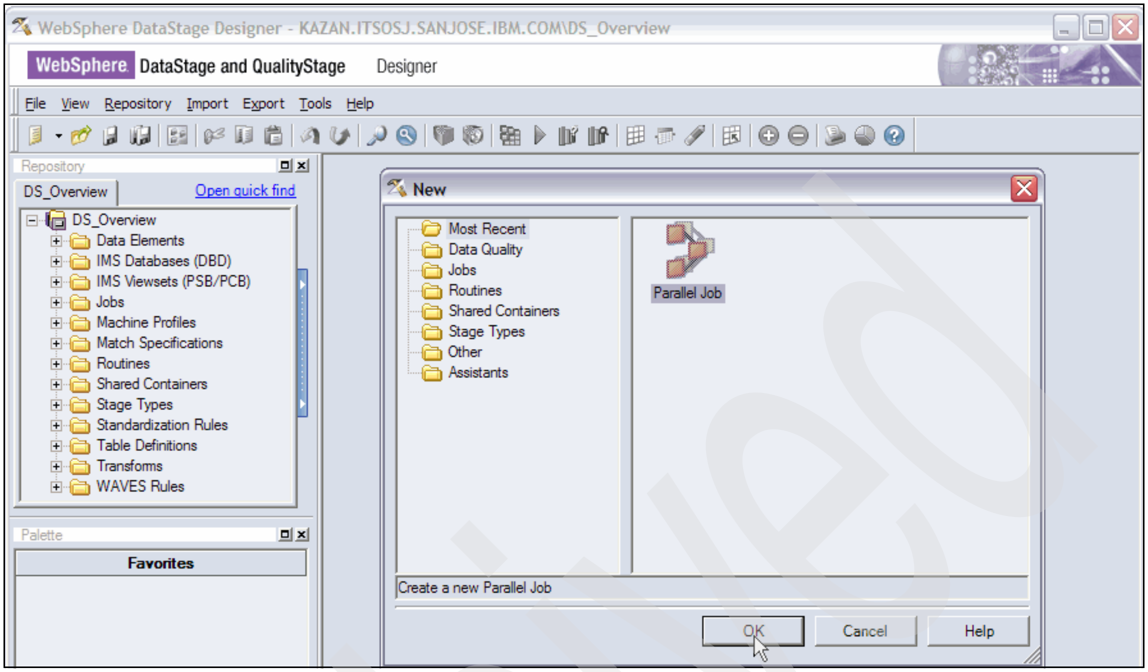


Figure 3-23 Create the J01_IL_FTPCustomerFile job 2/45

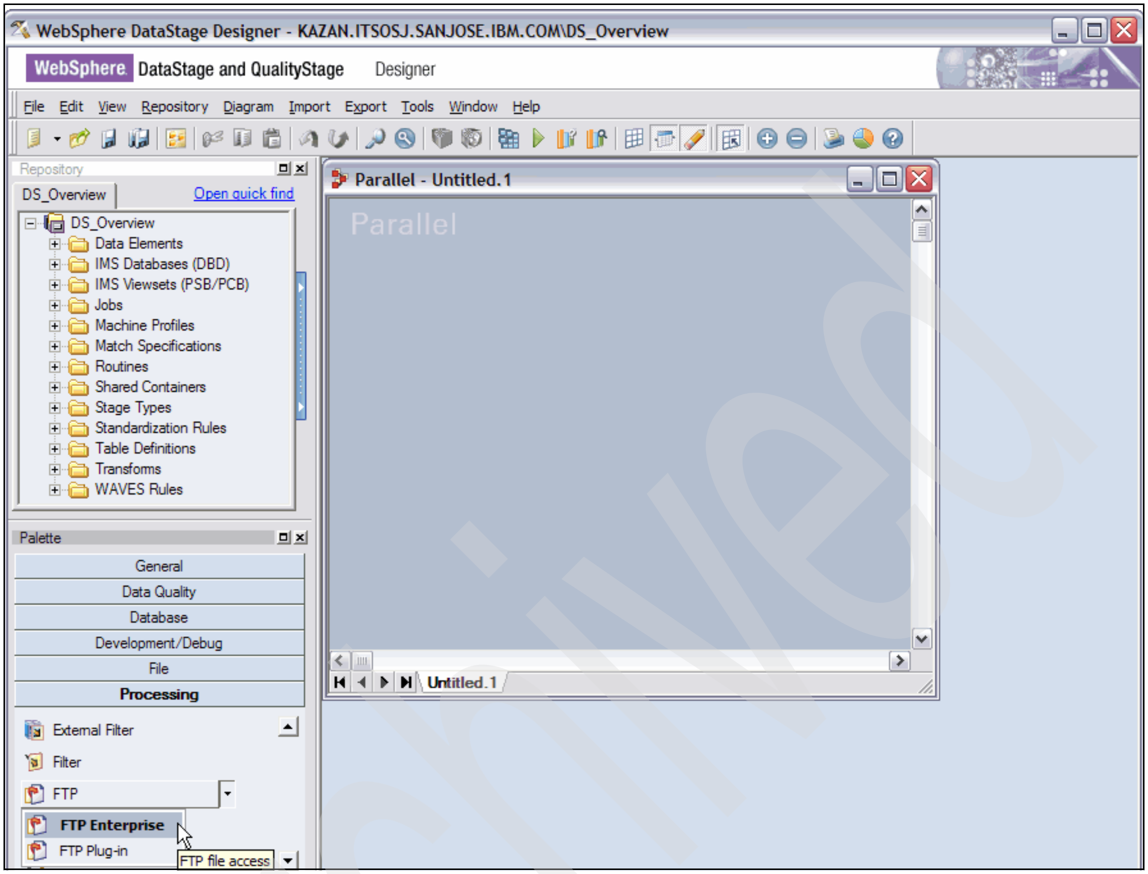


Figure 3-24 Create the J01_IL_FTPCustomerFile job 3/45

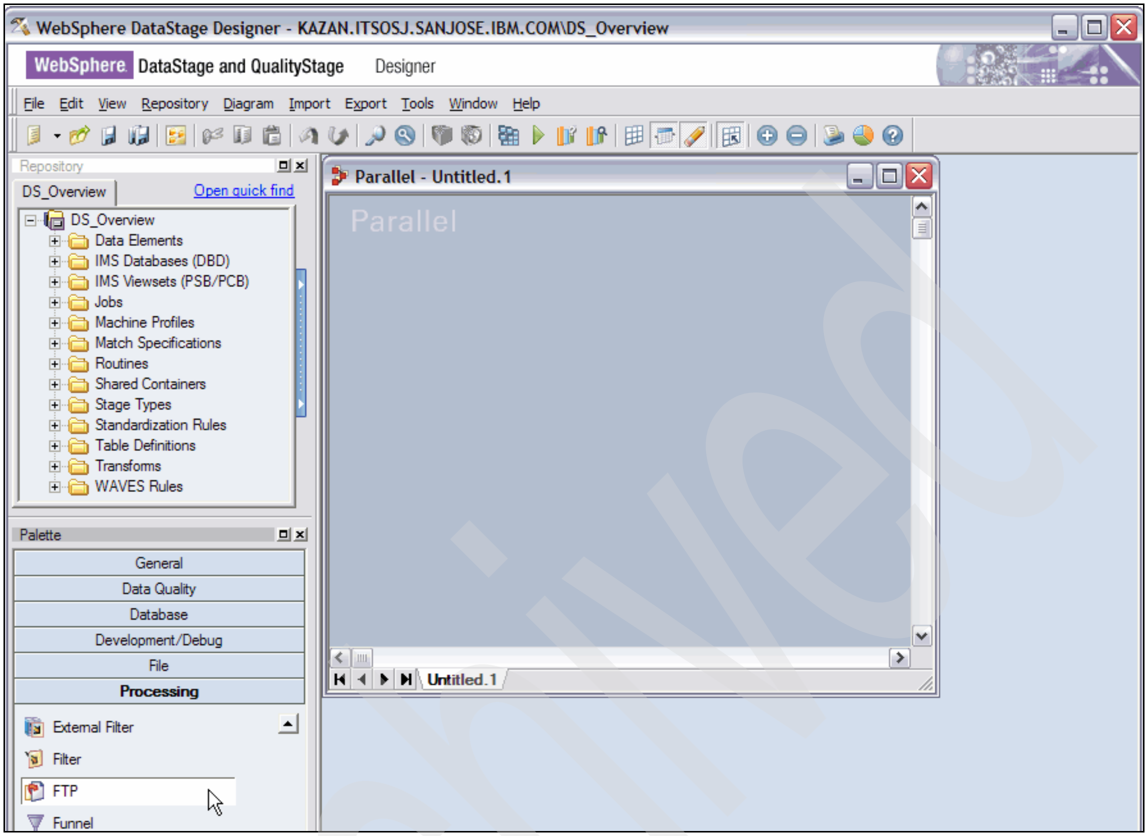


Figure 3-25 Create the J01_IL_FTPCustomerFile job 4/45

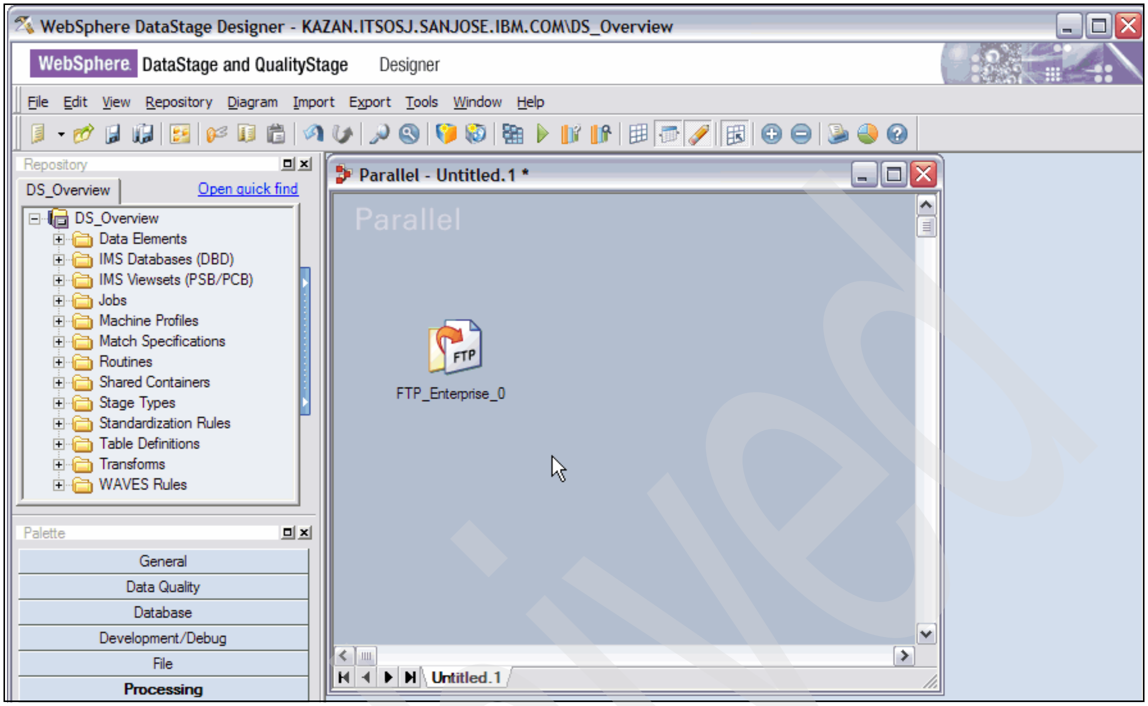


Figure 3-26 Create the J01_IL_FTPCustomerFile job 5/45

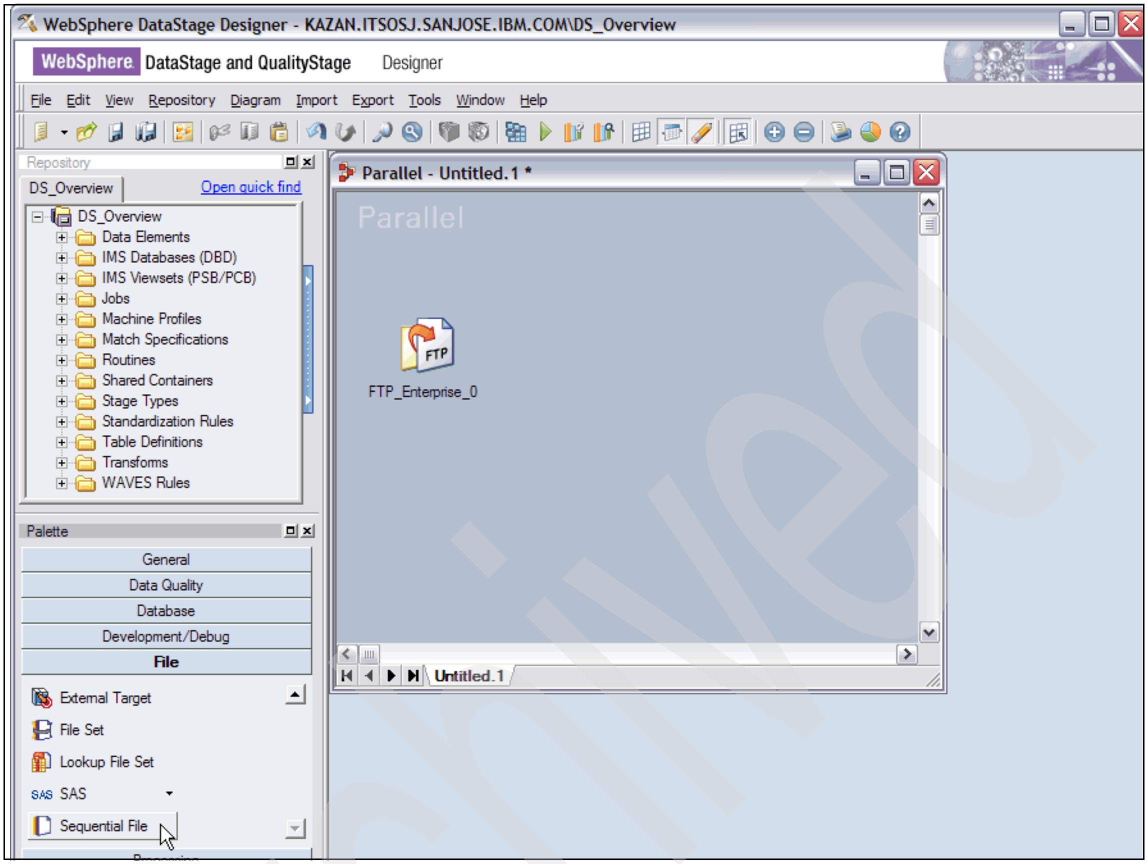


Figure 3-27 Create the J01_IL_FTPCustomerFile job 6/45

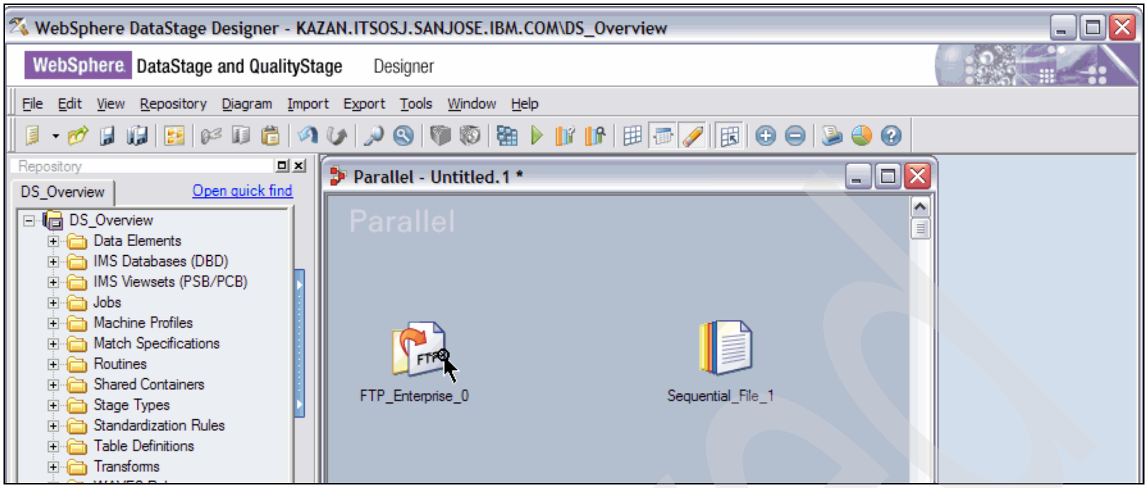


Figure 3-28 Create the J01_IL_FTPCustomerFile job 7/45

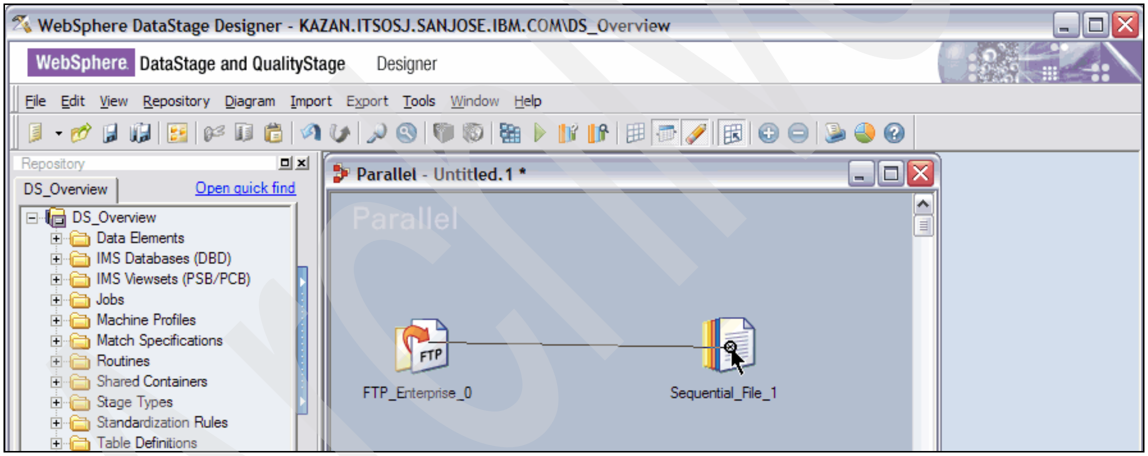


Figure 3-29 Create the J01_IL_FTPCustomerFile job 8/45

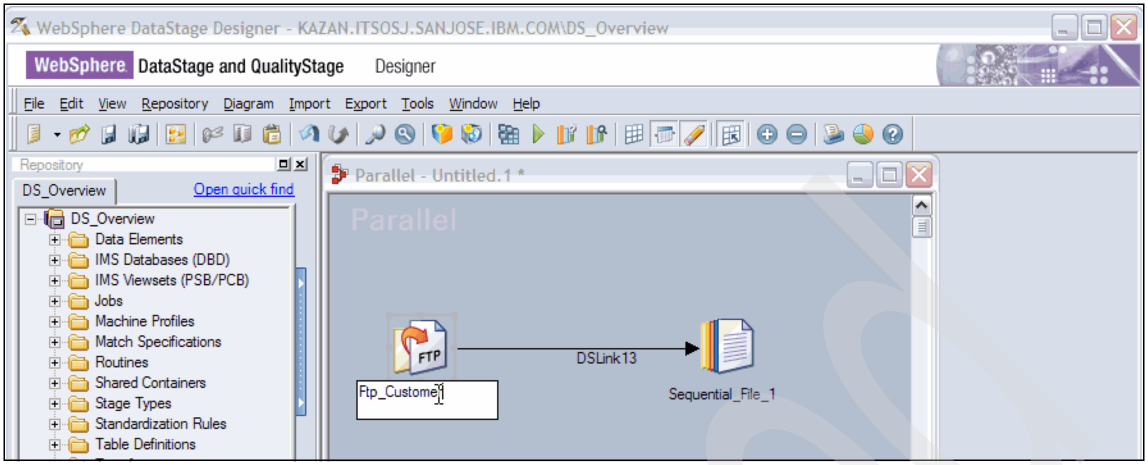


Figure 3-30 Create the J01_IL_FTPCustomerFile job 9/45

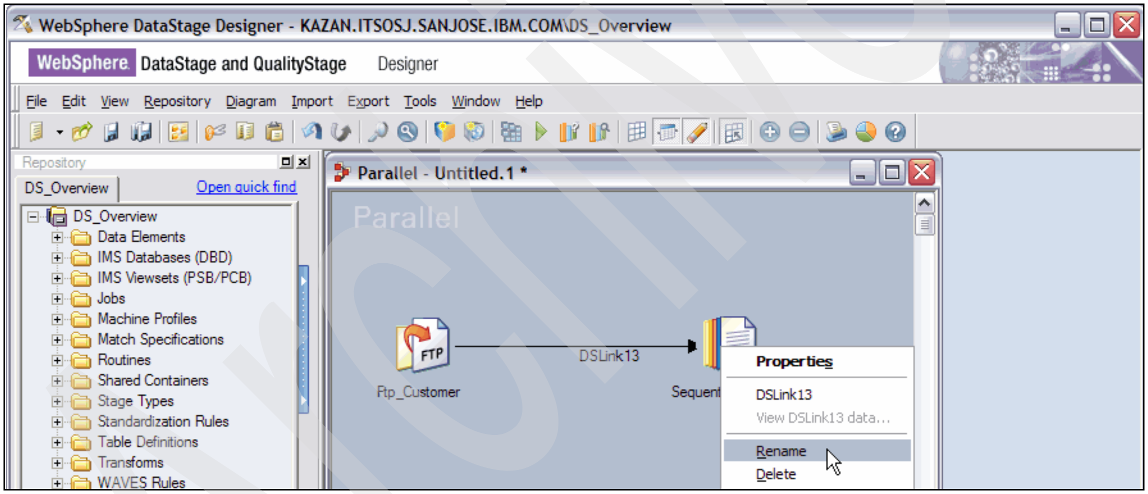


Figure 3-31 Create the J01_IL_FTPCustomerFile job 10/45

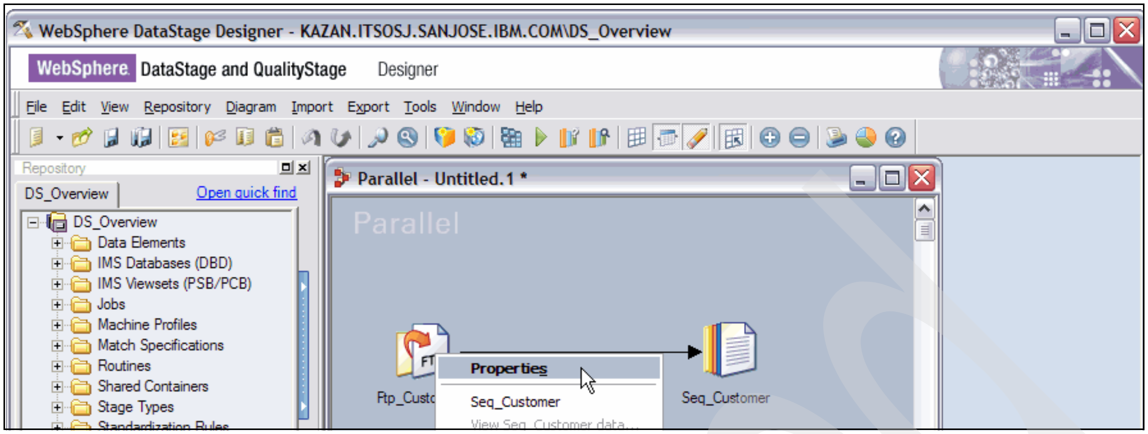


Figure 3-32 Create the J01_IL_FTPCustomerFile job 11/45

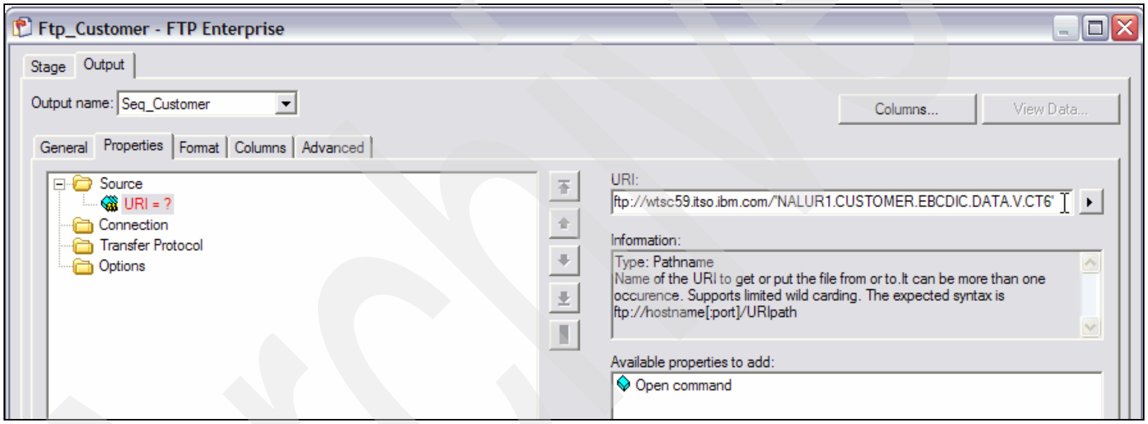


Figure 3-33 Create the J01_IL_FTPCustomerFile job 12/45

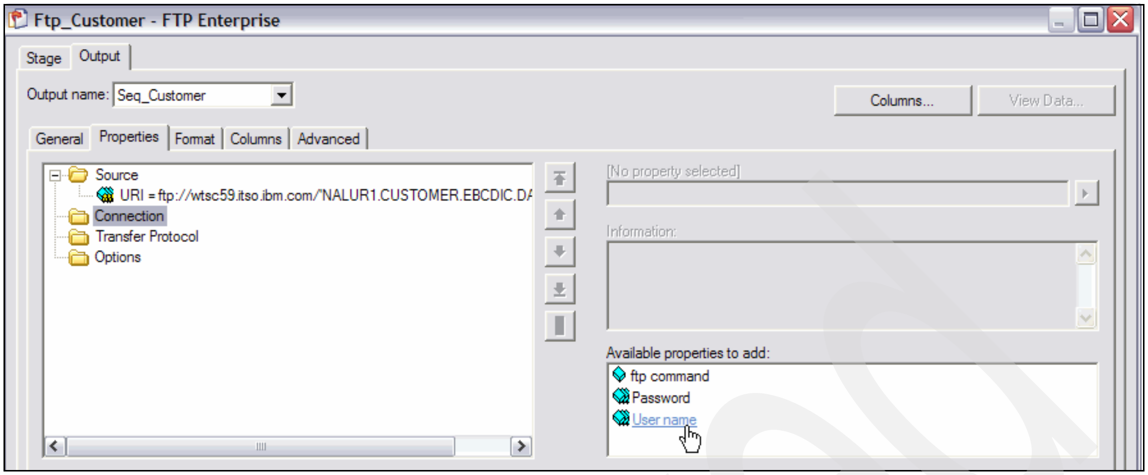


Figure 3-34 Create the J01_IL_FTPCustomerFile job 13/45

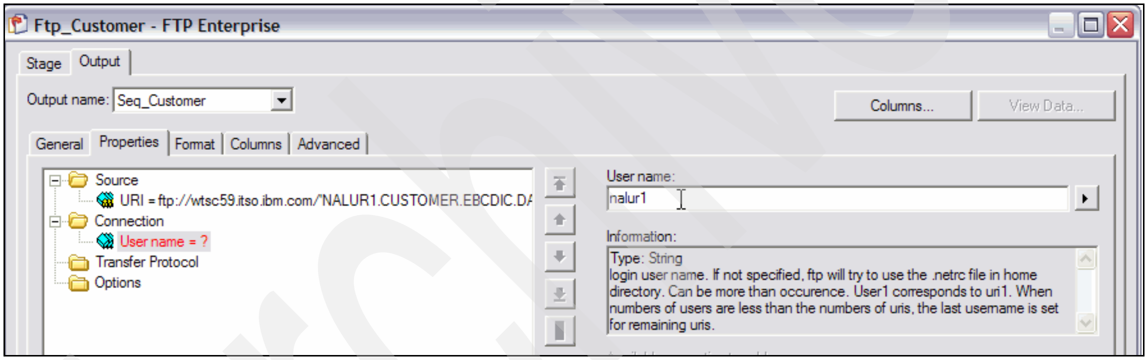


Figure 3-35 Create the J01_IL_FTPCustomerFile job 14/45

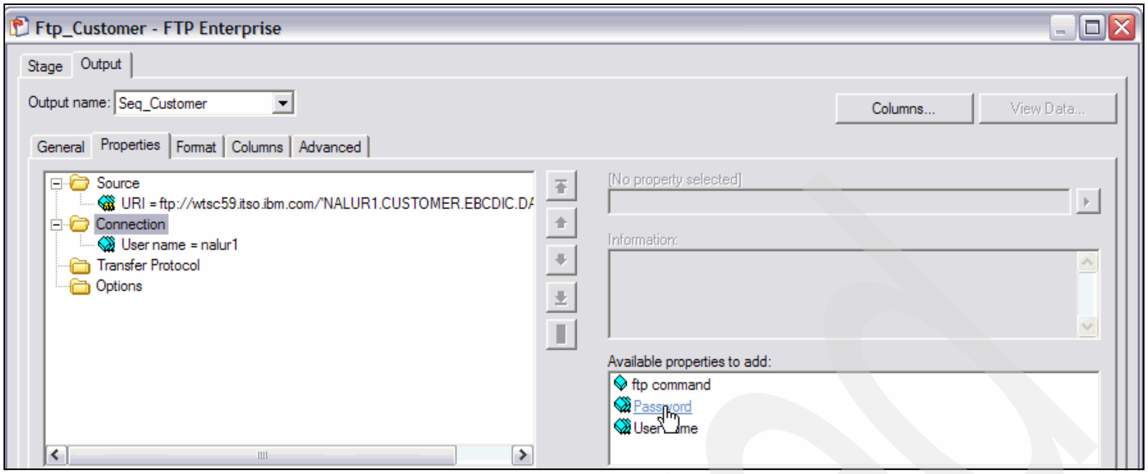


Figure 3-36 Create the J01_IL_FTPCustomerFile job 15/45

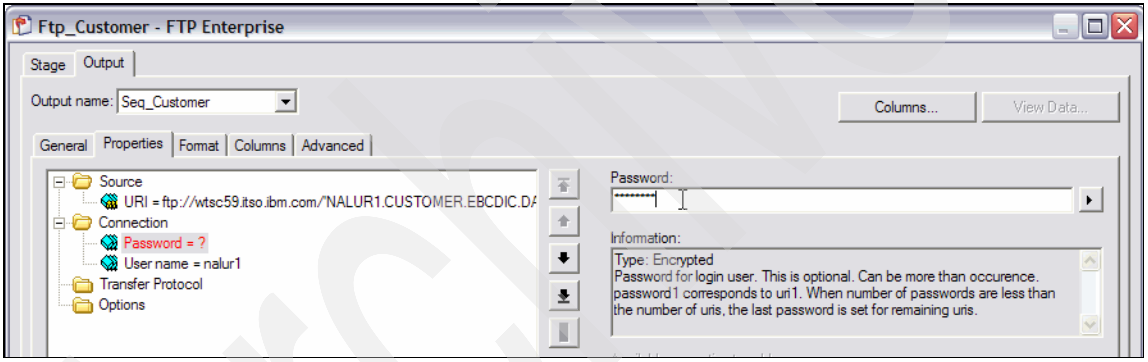


Figure 3-37 Create the J01_IL_FTPCustomerFile job 16/45

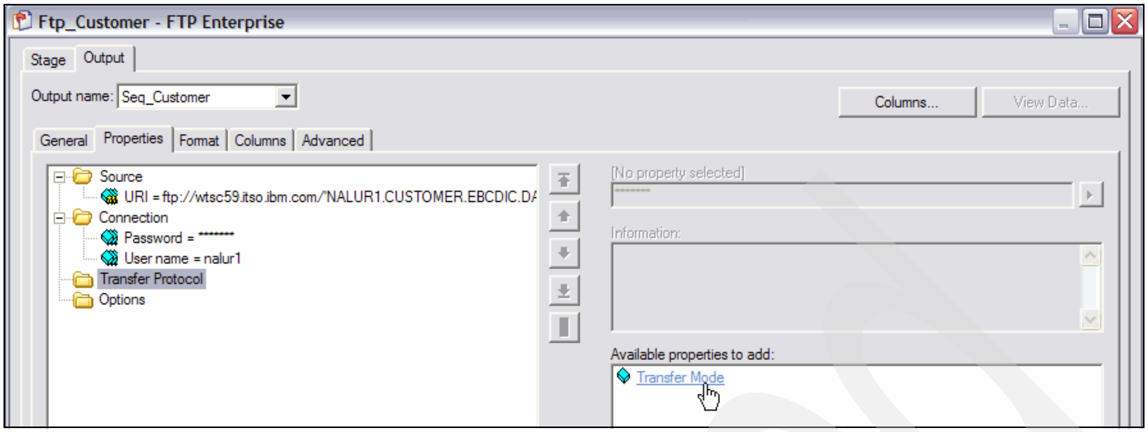


Figure 3-38 Create the J01_IL_FTPCustomerFile job 17/45

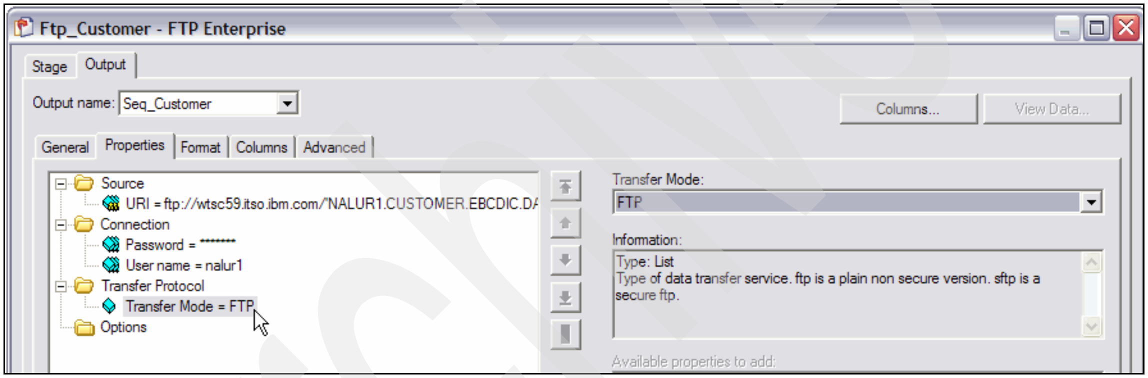


Figure 3-39 Create the J01_IL_FTPCustomerFile job 18/45

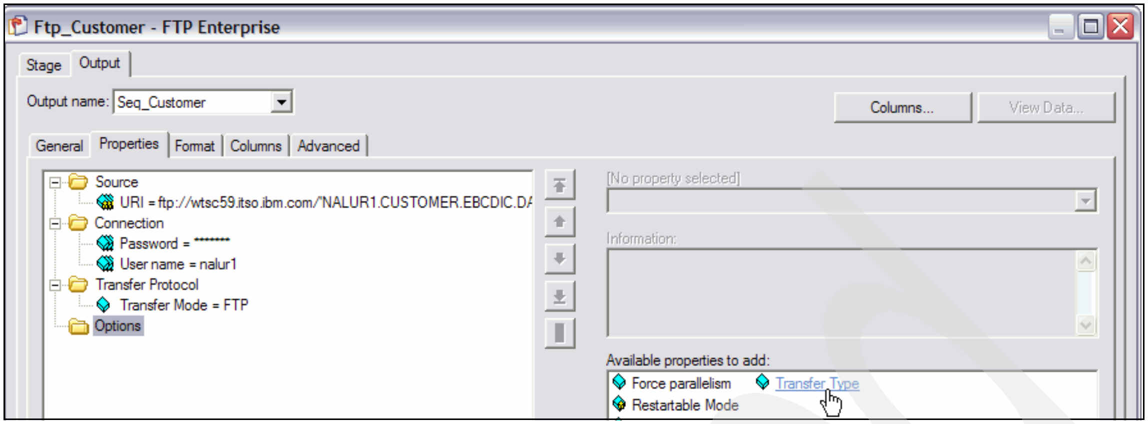


Figure 3-40 Create the J01_IL_FTPCustomerFile job 19/45

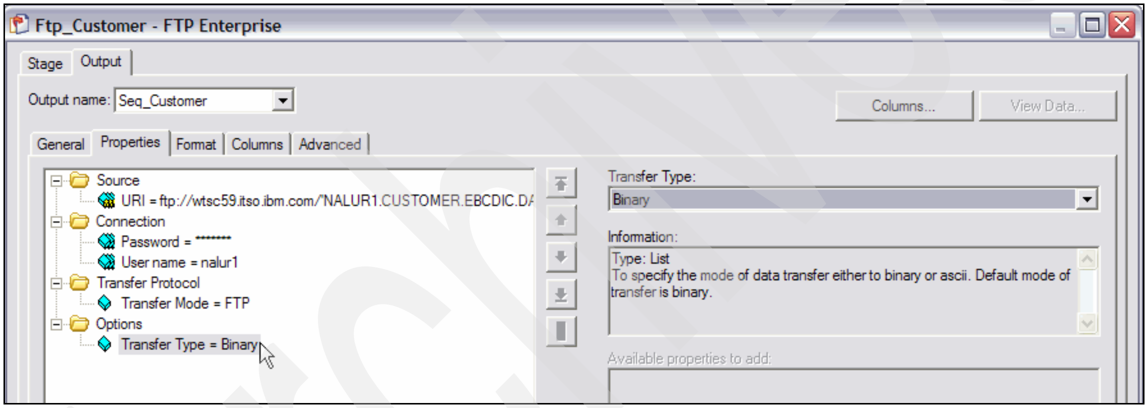


Figure 3-41 Create the J01_IL_FTPCustomerFile job 20/45

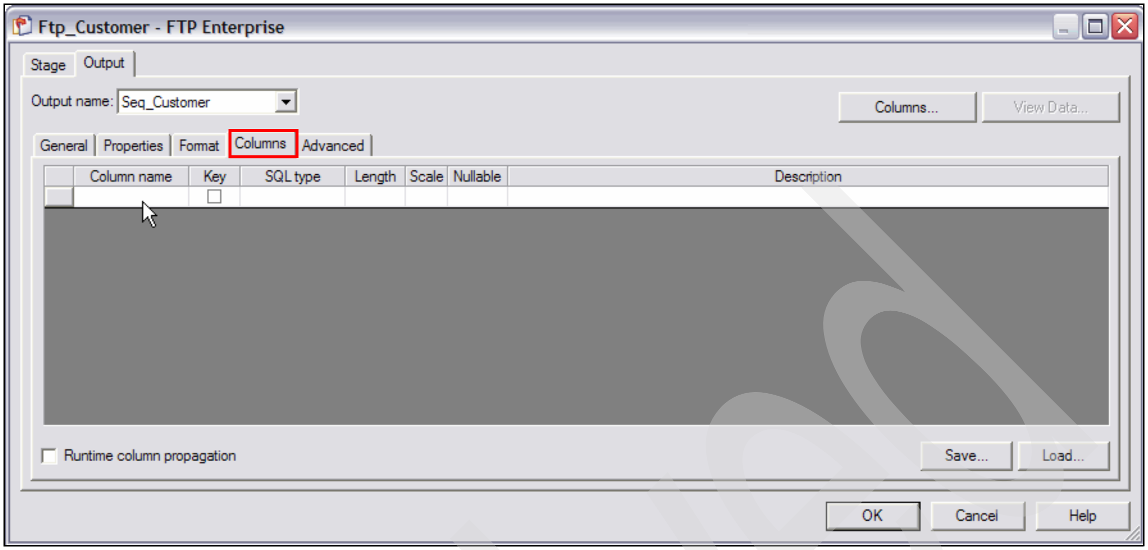


Figure 3-42 Create the J01_IL_FTPCustomerFile job 21/45

Figure 3-43 Create the J01_IL_FTPCustomerFile job 22/45

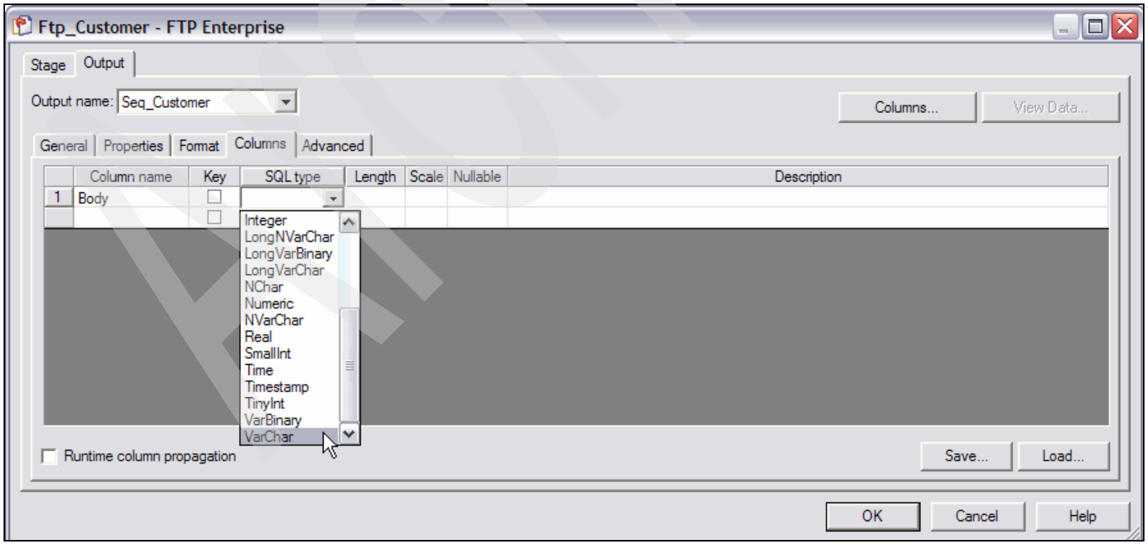


Figure 3-44 Create the J01_IL_FTPCustomerFile job 23/45

v

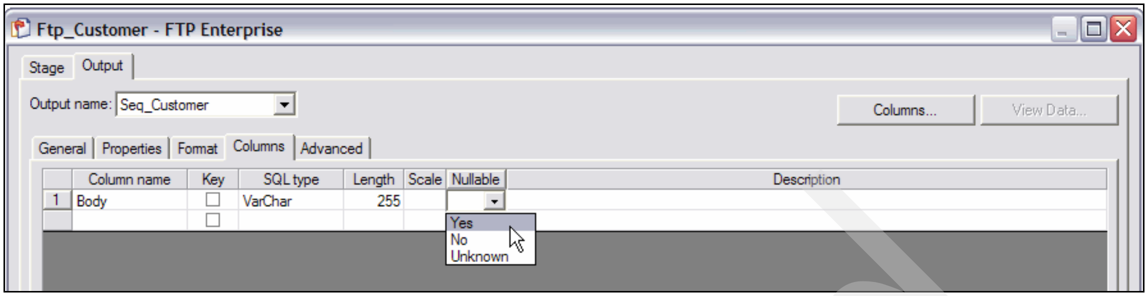


Figure 3-45 Create the J01_IL_FTPCustomerFile job 24/45

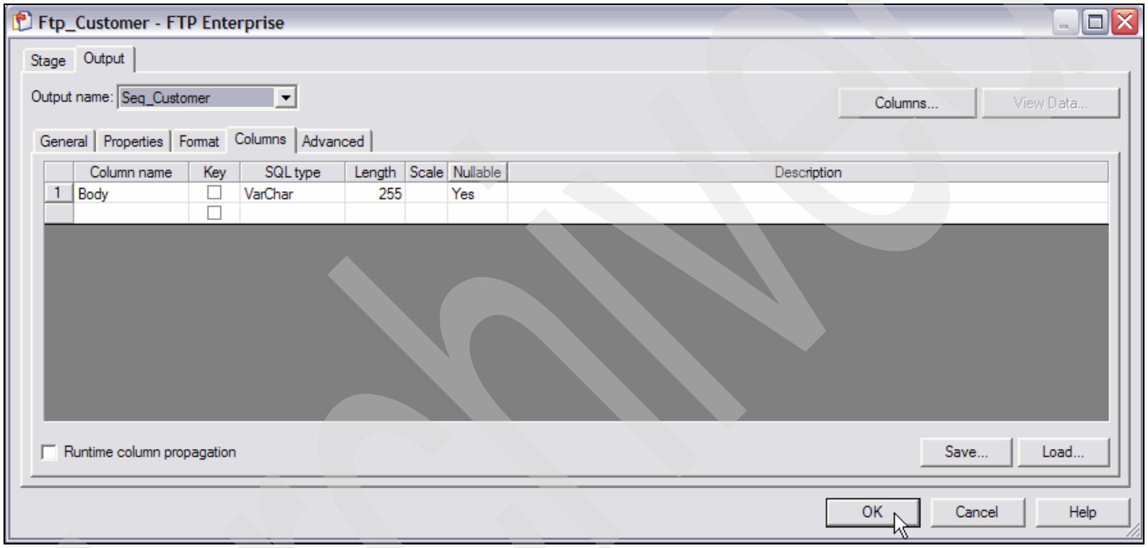


Figure 3-46 Create the J01_IL_FTPCustomerFile job 25/45

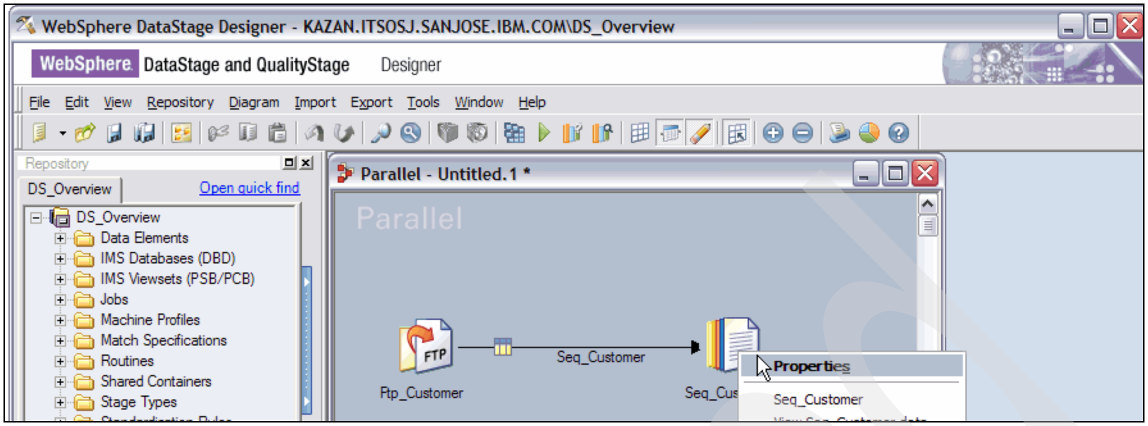


Figure 3-47 Create the J01_IL_FTPCustomerFile job 26/45

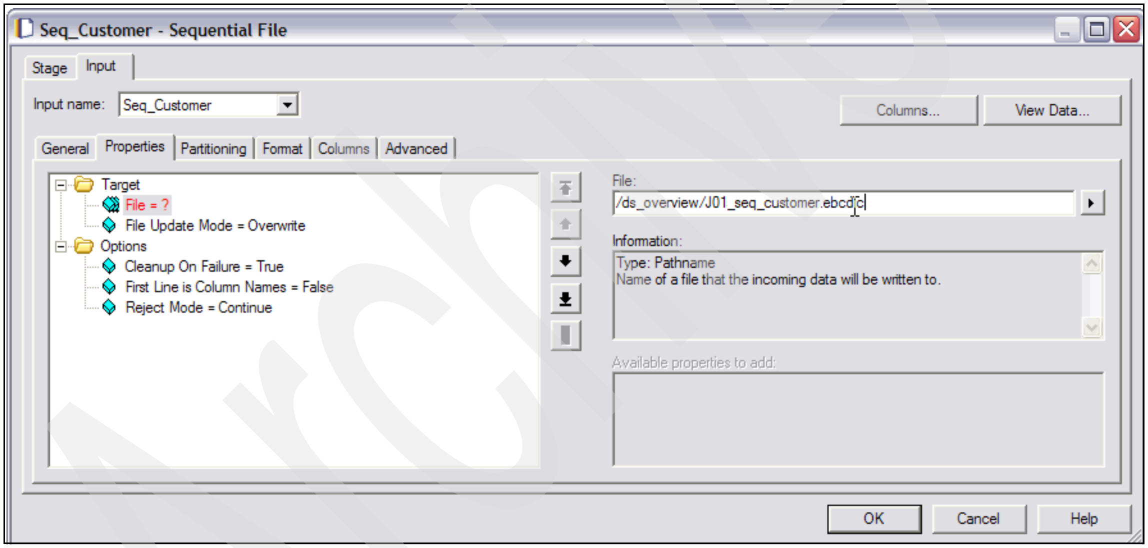


Figure 3-48 Create the J01_IL_FTPCustomerFile job 27/45

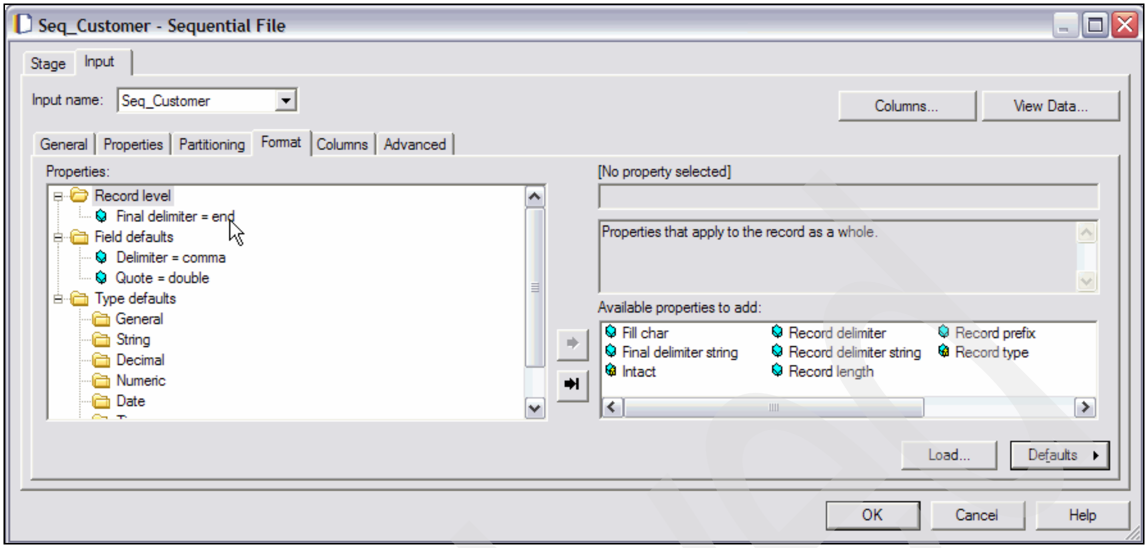


Figure 3-49 Create the J01_IL_FTPCustomerFile job 28/45

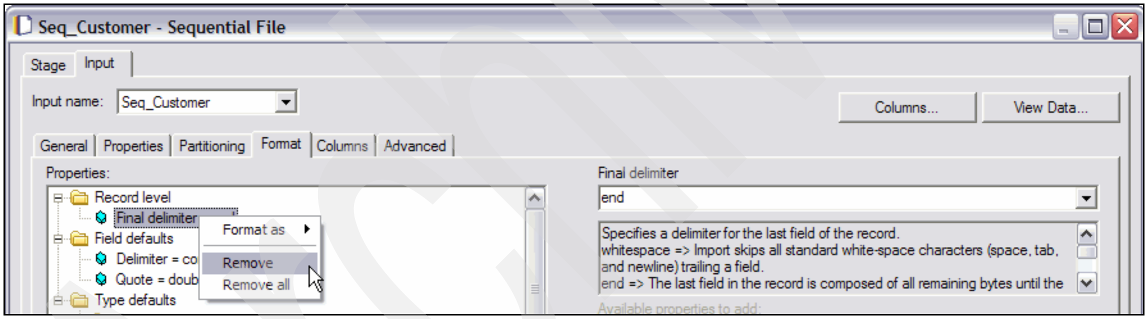


Figure 3-50 Create the J01_IL_FTPCustomerFile job 29/45

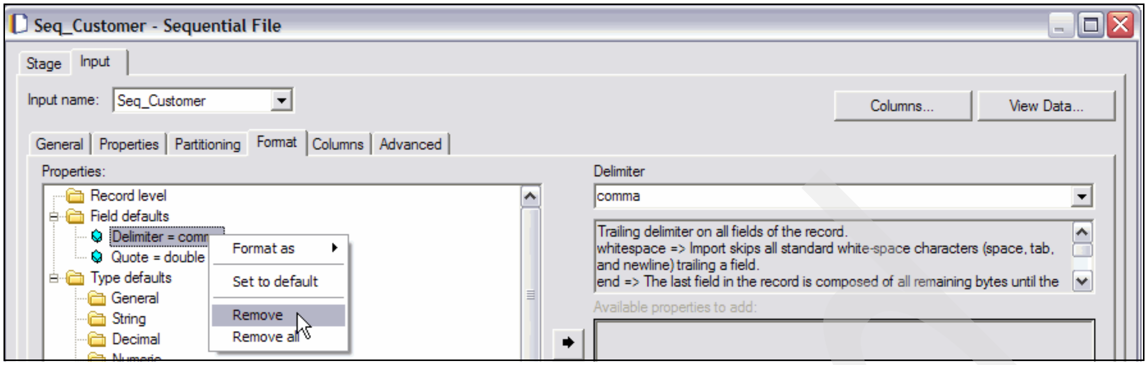


Figure 3-51 Create the J01_IL_FTPCustomerFile job 30/45

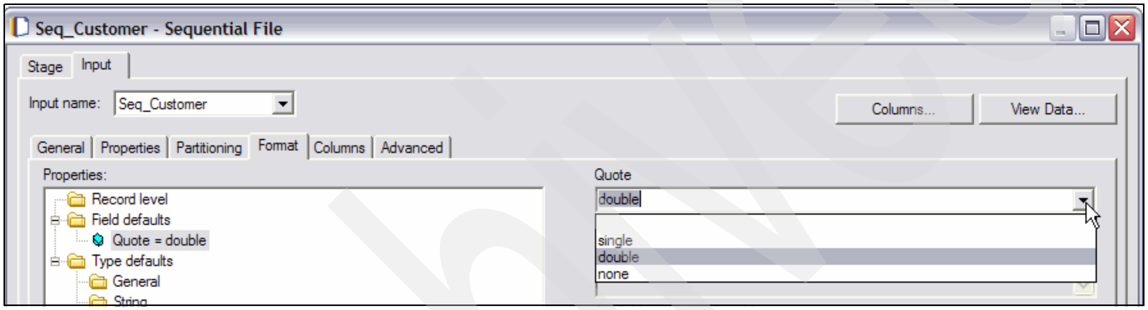


Figure 3-52 Create the J01_IL_FTPCustomerFile job 31/45

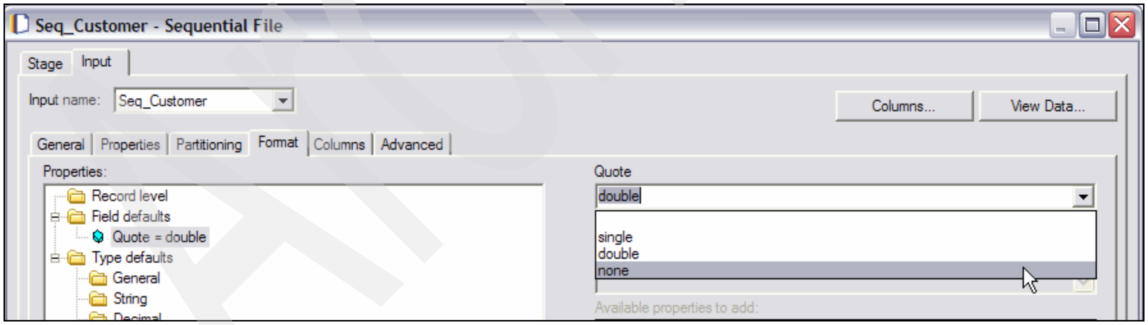


Figure 3-53 Create the J01_IL_FTPCustomerFile job 32/45

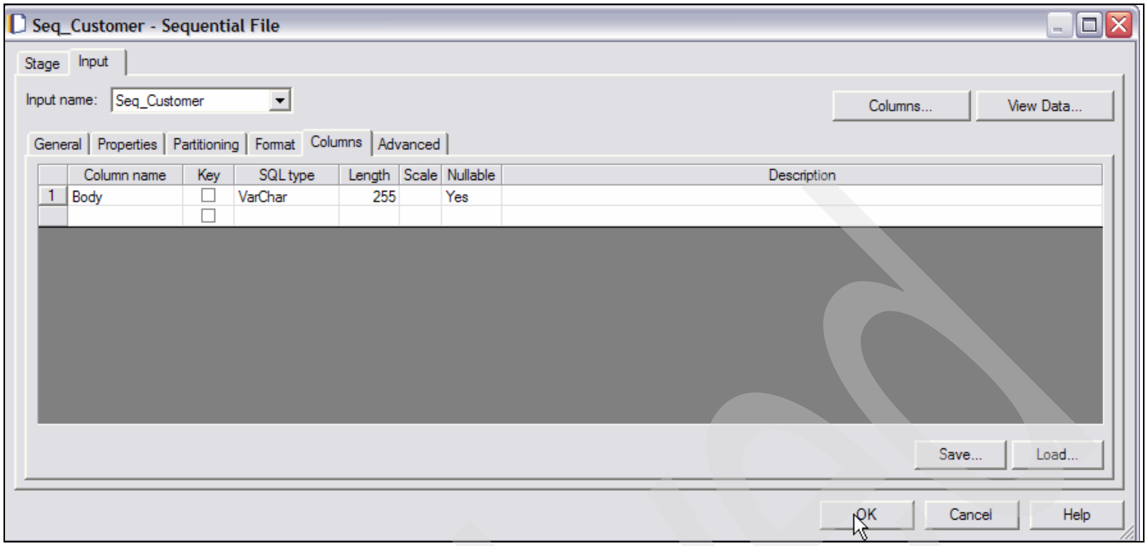


Figure 3-54 Create the J01_IL_FTPCustomerFile job 33/45

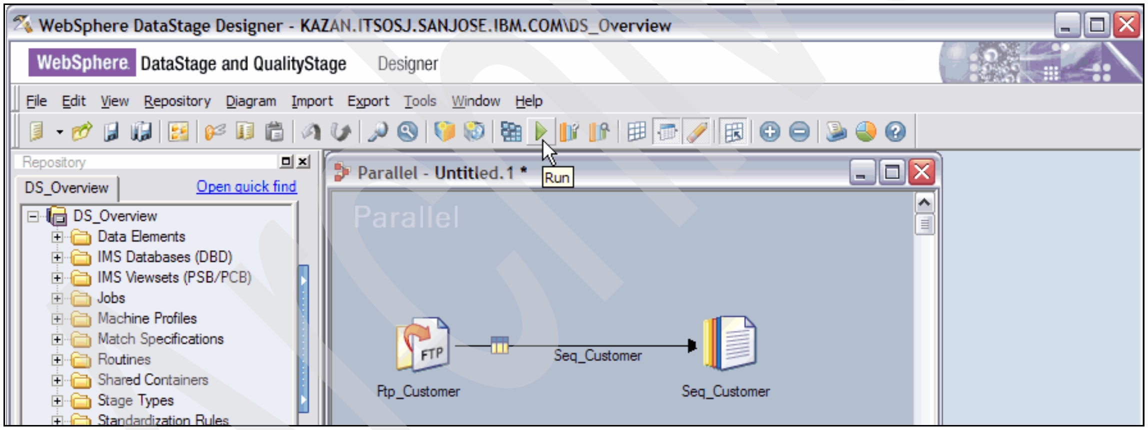


Figure 3-55 Create the J01_IL_FTPCustomerFile job 34/45

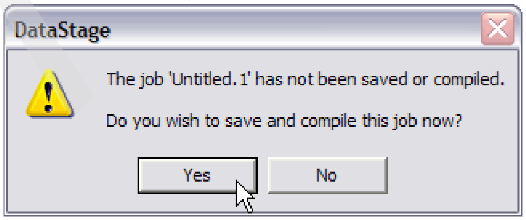


Figure 3-56 Create the J01_IL_FTPCustomerFile job 35/45

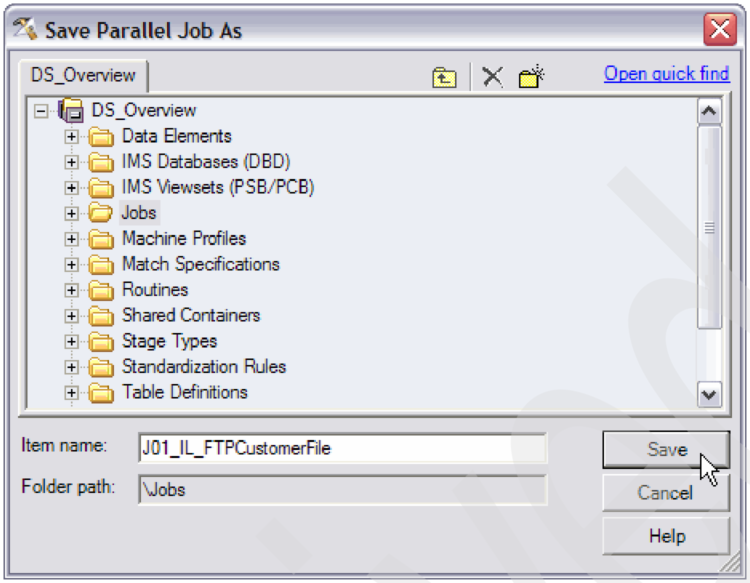


Figure 3-57 Create the J01_IL_FTPCustomerFile job 36/45

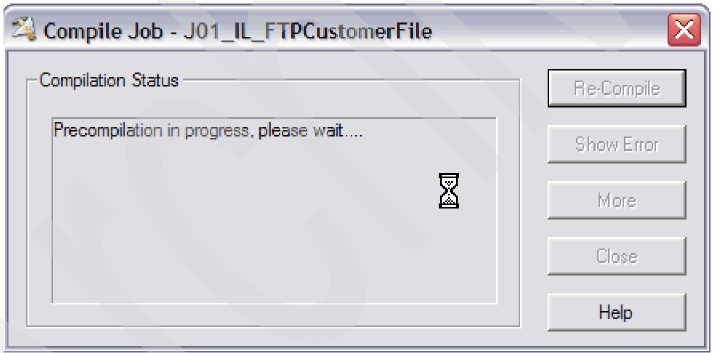


Figure 3-58 Create the J01_IL_FTPCustomerFile job 37/45

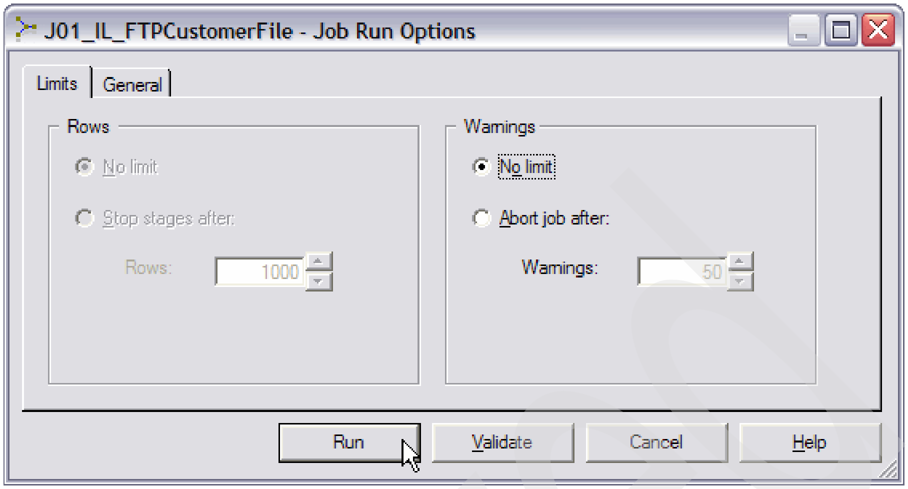


Figure 3-59 Create the J01_IL_FTPCustomerFile job 38/45

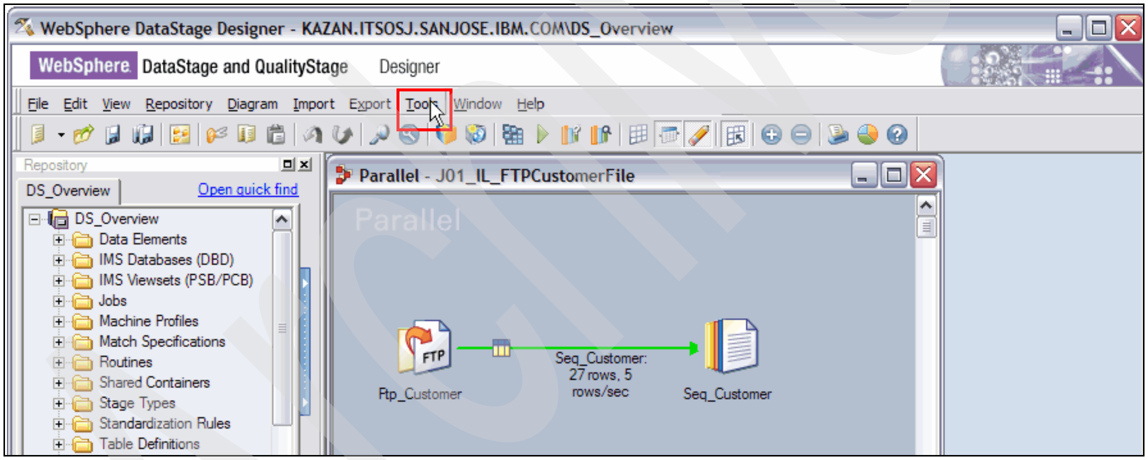


Figure 3-60 Create the J01_IL_FTPCustomerFile job 39/45

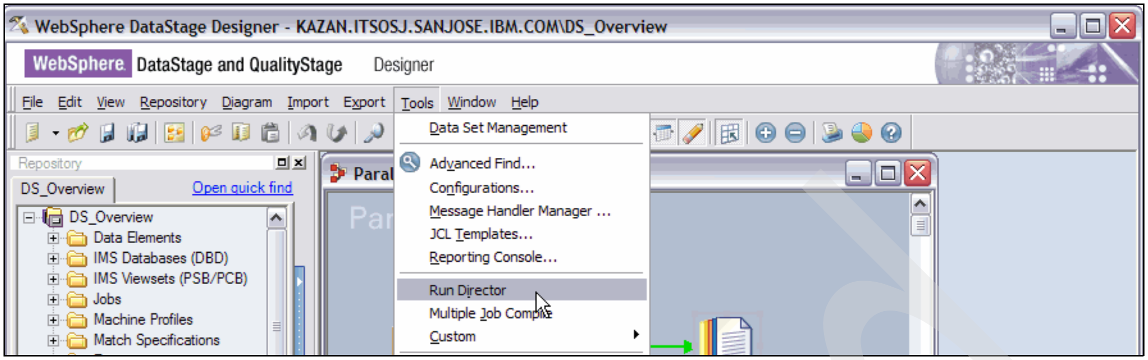


Figure 3-61 Create the J01_IL_FTPCustomerFile job 40/45

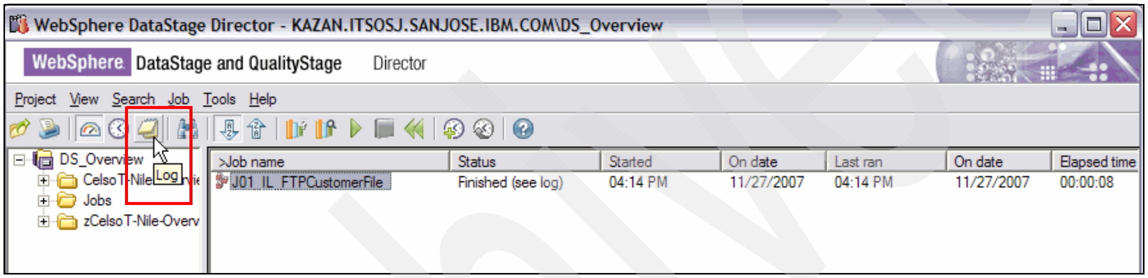


Figure 3-62 Create the J01_IL_FTPCustomerFile job 41/45

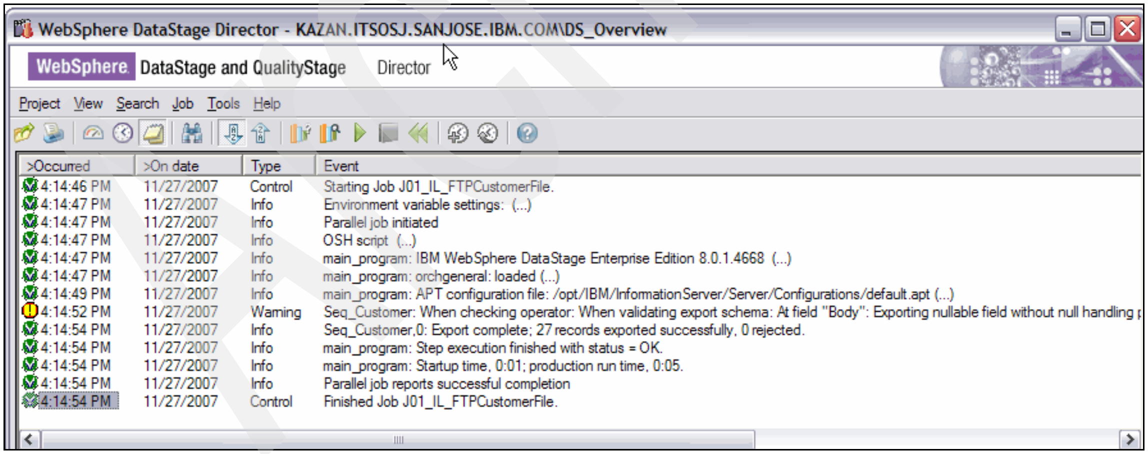


Figure 3-63 Create the J01_IL_FTPCustomerFile job 42/45

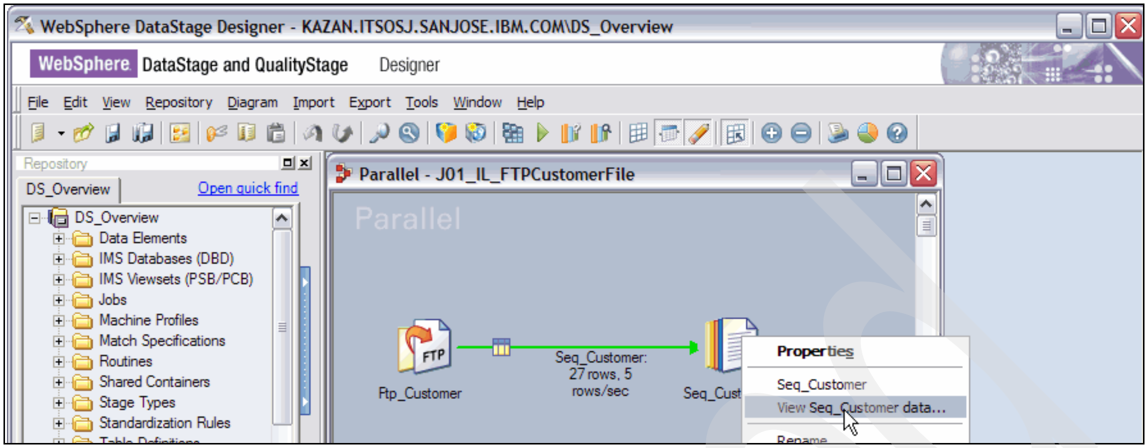


Figure 3-64 Create the J01_IL_FTPCustomerFile job 43/45

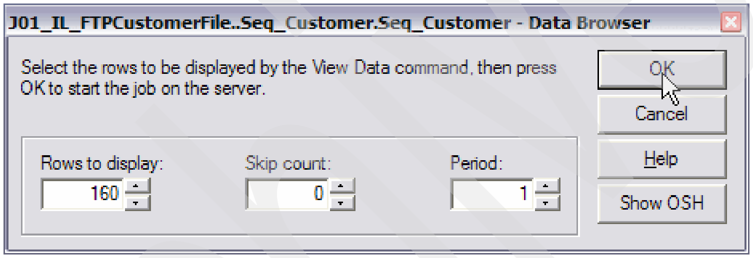


Figure 3-65 Create the J01_IL_FTPCustomerFile job 44/45

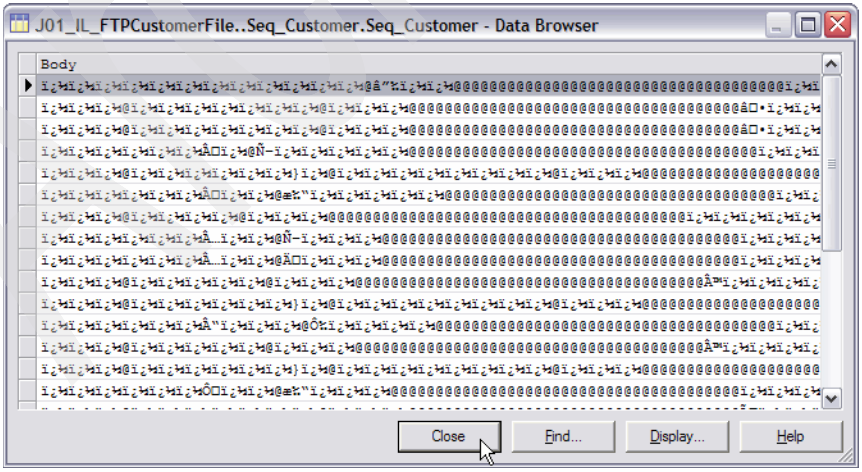


Figure 3-66 Create the J01_IL_FTPCustomerFile job 45/45

J02_IL_LoadCustomerDim

In this job, we extract relevant attributes from the Customer file and load them into the CUSTOMER_DIM dimension table.

Figure 3-67 on page 186 through Figure 3-92 on page 201 describe the steps using Designer Client to build and execute the DataStage job to perform this task.

The steps are as follows:

1. Figure 3-67 on page 186 shows the various stages used in this job — it includes the Data Set created in "J01_IL_FTPCustomerFile" on page 159, a Complex Flat File stage, a Transformer stage, a Remove Duplicates stage, and an ODBCConnector stage. The names of the stages were modified as shown.
2. Figure 3-68 on page 187 through Figure 3-77 on page 193 show the configuration of the Complex Flat File stage that extracts and processes customer information from a file that contains multiple record types for loading into the dimension table.

   In the CFF stage, you must provide details about the file that the stage will read, create record definitions for the data, define the column metadata, specify record ID constraints, and select output columns.

   – Figure 3-68 on page 187 shows the **File options** tab in the Stage page, which provides details about the file (J01_seq_customer.ebcdic) that the stage will read.

   – Figure 3-69 on page 187 shows the **Record options** tab in the Stage page, which describes the format of the data in the file. Specifically, the Character set (EBCDIC), Data format (Binary), and Record delimiter (UNIX Newline) are of interest, corresponding to the file transferred by the FTP Enterprise stage in "J01_IL_FTPCustomerFile" on page 159.

   – Since the stage will be reading a file containing multiple record types, we have to create the record definitions of the data. Figure 3-70 on page 188 through Figure 3-72 on page 190 show the **Records** tab in the Stage page, which identify the three (CUSTOMER, HOMEADDRESS, and WORKADDRESS) record definitions in the customer file by either typing or loading column definitions from the repository.

   – Figure 3-73 on page 190 through Figure 3-75 on page 191 define the record ID constraint for each record (CUSTOMER record type with a value 'CD', HOMEADDRESS record type with a value 'HA', and WORKADDRESS record type with a value 'WA') on the **Records ID** tab.

– Figure 3-76 on page 192 shows the **Selection** tab in the Output page, which specifies how to read data from the source file. It shows the selection of multiple columns (excluding only the RECTYPE, RECTYPE_2, and RECTYPE_3 columns from the input) for the Trx_Customer output link.

> **Note:** By selecting output columns, you specify which columns from the source file the CFF stage should pass to the output links. You can select columns from multiple record types to output from the stage. If you do not select columns to output on each link, the CFF stage automatically propagates all of the stage columns except group columns to each empty output link when you click **OK** to exit the stage.

– Figure 3-77 on page 193 shows the **Constraint** tab in the Output page, which filters the rows (based on the values 'CD', 'HA', and 'WA' in the record type columns in this case) on the output.

> **Note:** You must specify a record ID constraint to identify the format of each record. Columns that are identified in the record ID clause must be in the same physical storage location across records. The constraint must be a simple equality expression, where a column equals a value.

3. Figure 3-78 on page 193 through Figure 3-79 on page 194 show the contents of the output file of the CFF stage. It shows multiple records for the same customer corresponding to each record type — some customers have only one record type (Beel Jones); others have two record types (Barn Williams); and some have all three record types (Archana Smith).

> **Note:** The fields of only the last instance of a particular customer record have all the information from all the record types, which is why Duplicate To Retain = Last option is used in the following Remove Duplicates stage.

4. The Transformer stage is used to trim the trailing blanks in the various fields using the TRIM function as shown in Figure 3-80 on page 195.

5. The Remove Duplicates stage is required to eliminate the multiple occurrences of the same customer. Each record instance in the CFF stage output has columns populated from the different record types depending upon the sequence of arrival of each record type. The record instance corresponding to the last record type arrival (in the input file) for a customer has the consolidated information from all the record types associated with that customer. This is the record instance of the customer that must be preserved in the Remove Duplicates stage with the Duplicate To Retain = Last option as shown in Figure 3-81 on page 195 through Figure 3-84 on page 197.
6. The ODBCConnectorPX stage does a simple SQL INSERT of the cleansed and consolidated customer information into the CUSTOMER_DIM dimension table as shown in Figure 3-85 on page 198 and Figure 3-86 on page 198. The SQL INSERT statement is manually coded rather than being automatically generated.
7. The execution result of this job is shown in Figure 3-87 on page 199 and Figure 3-88 on page 199. It shows 27 records from the CFF stage being reduced to 11 records in the Remove Duplicates stage, which are then inserted into the CUSTOMER_DIM table.
8. Figure 3-89 on page 200 through Figure 3-92 on page 201 show the 11 records that are input to the ODBCConnectorPX stage.

We then proceeded to load the Product dimension table as described in "J03_IL_LoadProductDim" on page 202.

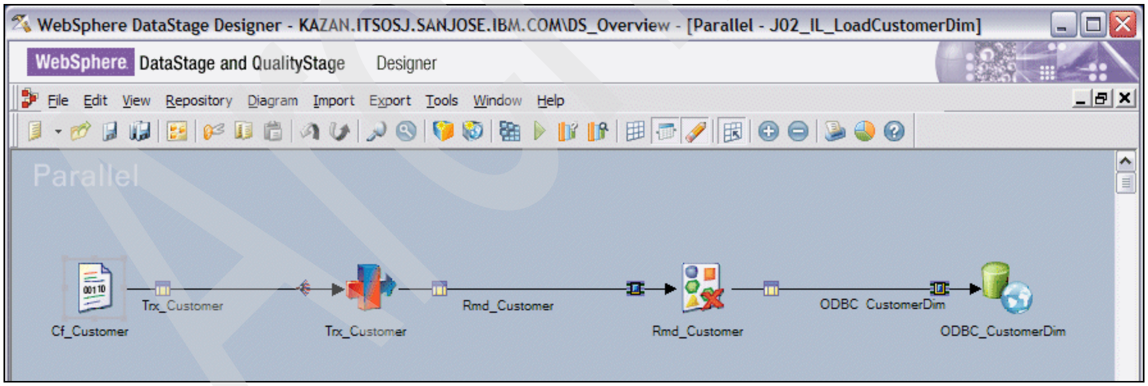


Figure 3-67 Create the J02_IL_LoadCustomerDim job 1/26

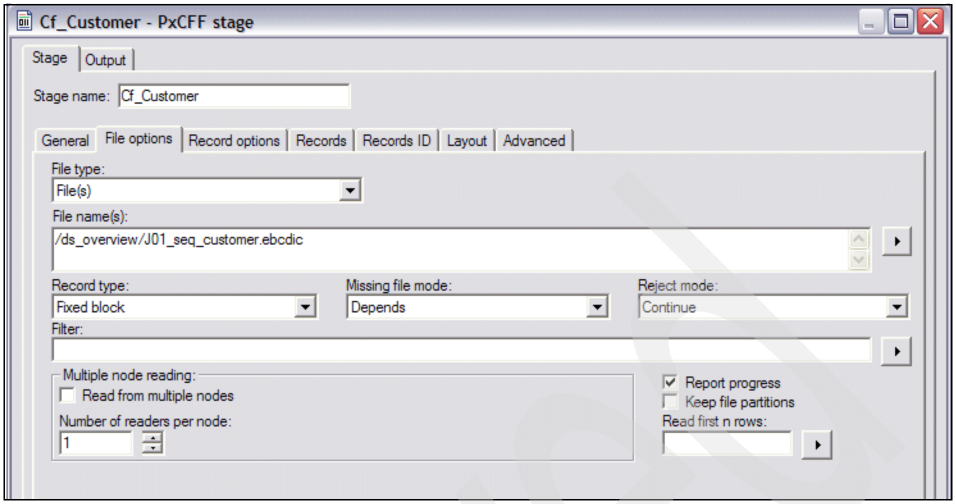


Figure 3-68 Create the J02_IL_LoadCustomerDim job 2/26

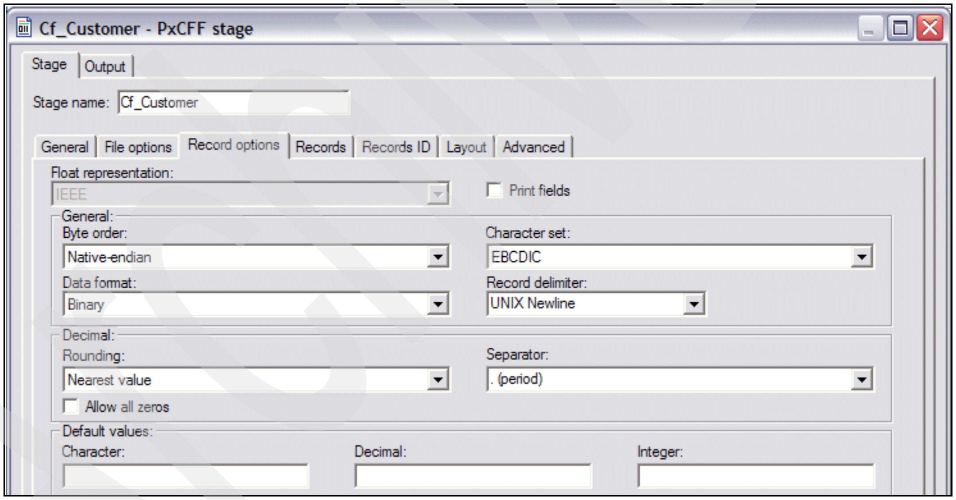


Figure 3-69 Create the J02_IL_LoadCustomerDim job 3/26

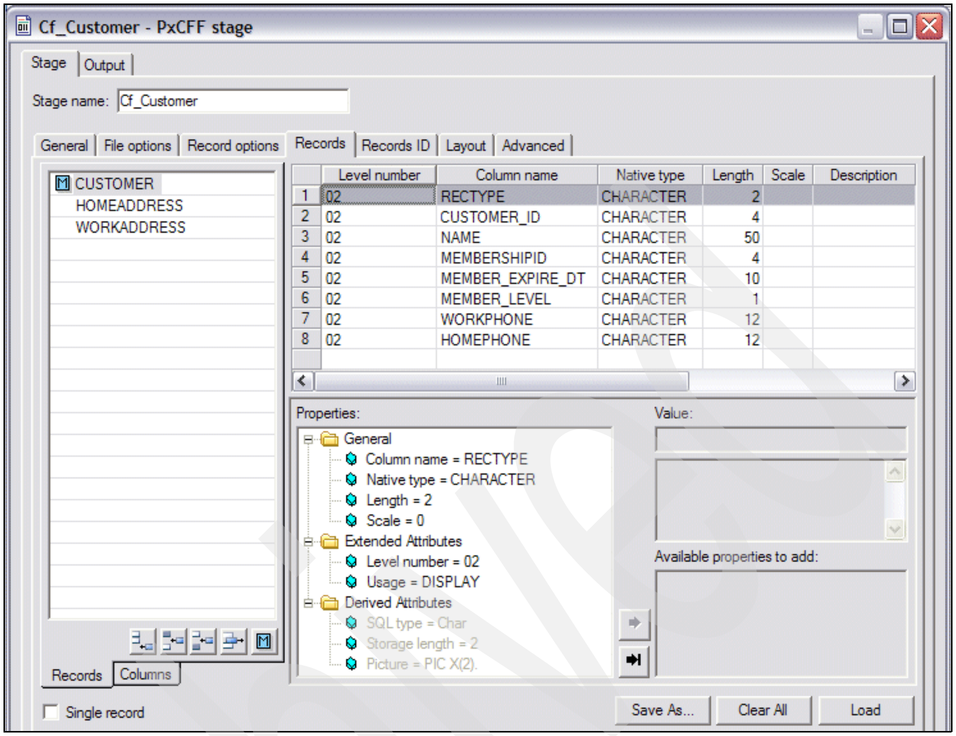


Figure 3-70 Create the J02_IL_LoadCustomerDim job 4/26

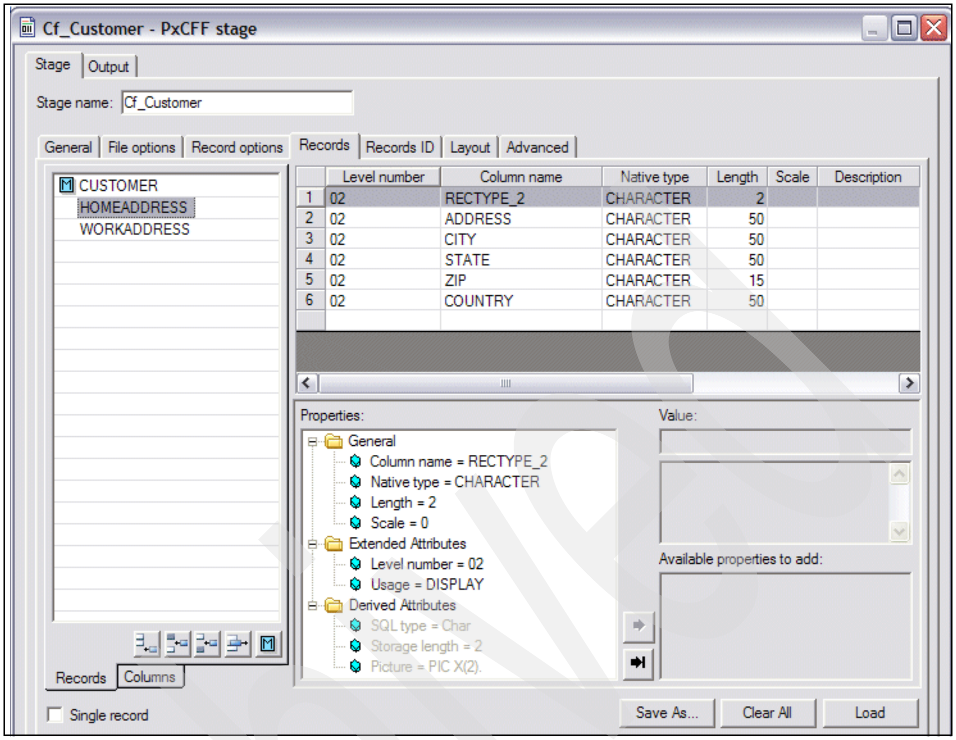


Figure 3-71 Create the J02_IL_LoadCustomerDim job 5/26

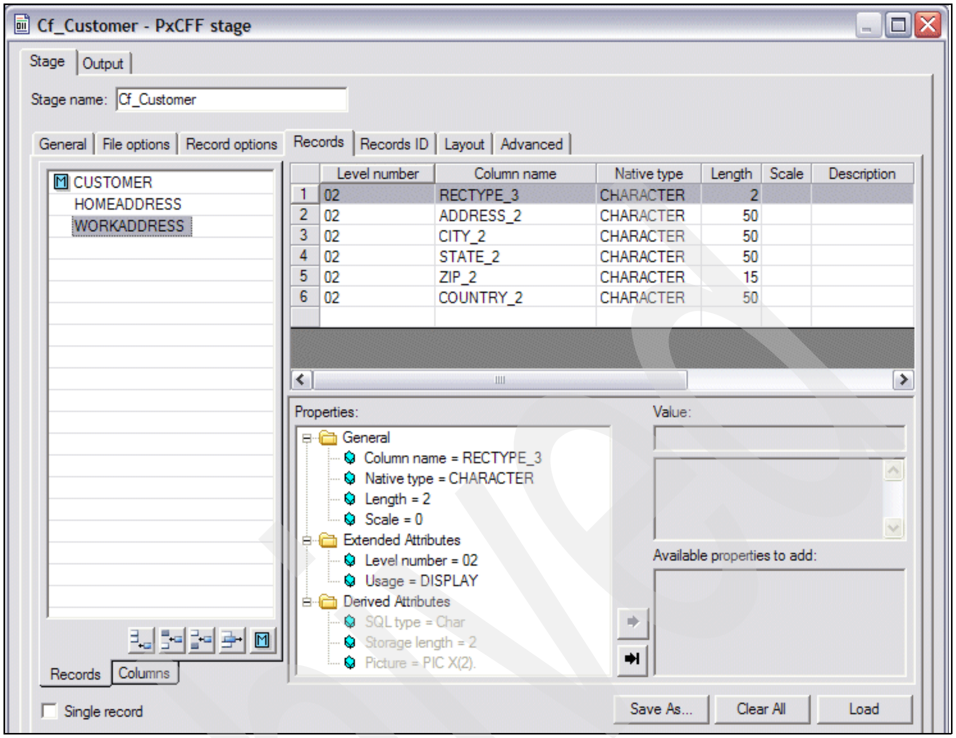


Figure 3-72 Create the J02_IL_LoadCustomerDim job 6/26

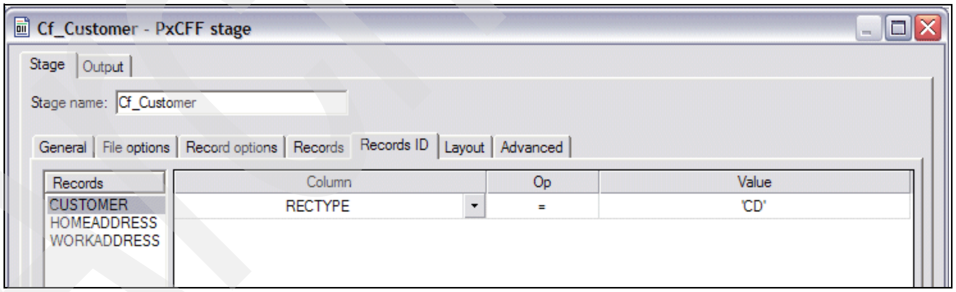


Figure 3-73 Create the J02_IL_LoadCustomerDim job 7/26

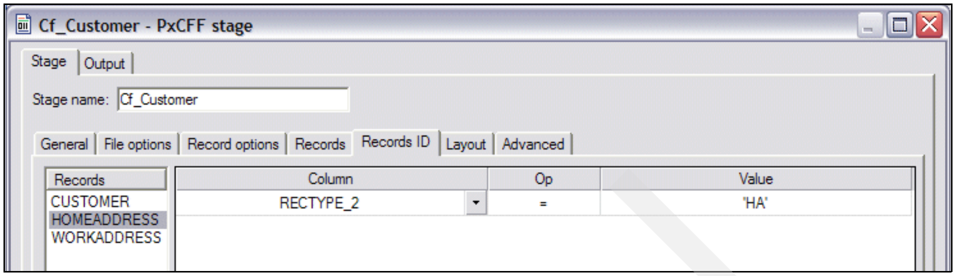


Figure 3-74 Create the J02_IL_LoadCustomerDim job 8/26

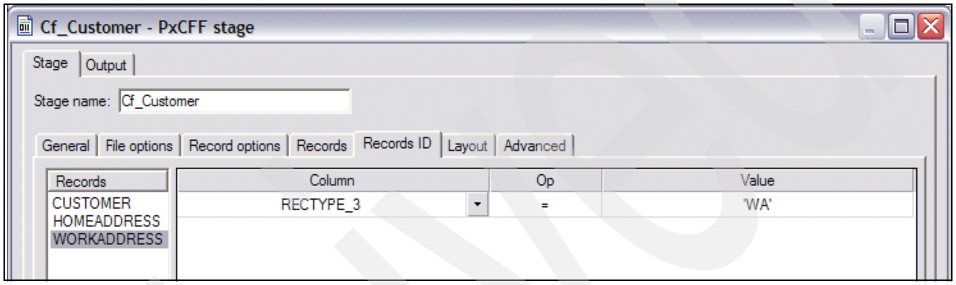


Figure 3-75 Create the J02_IL_LoadCustomerDim job 9/26

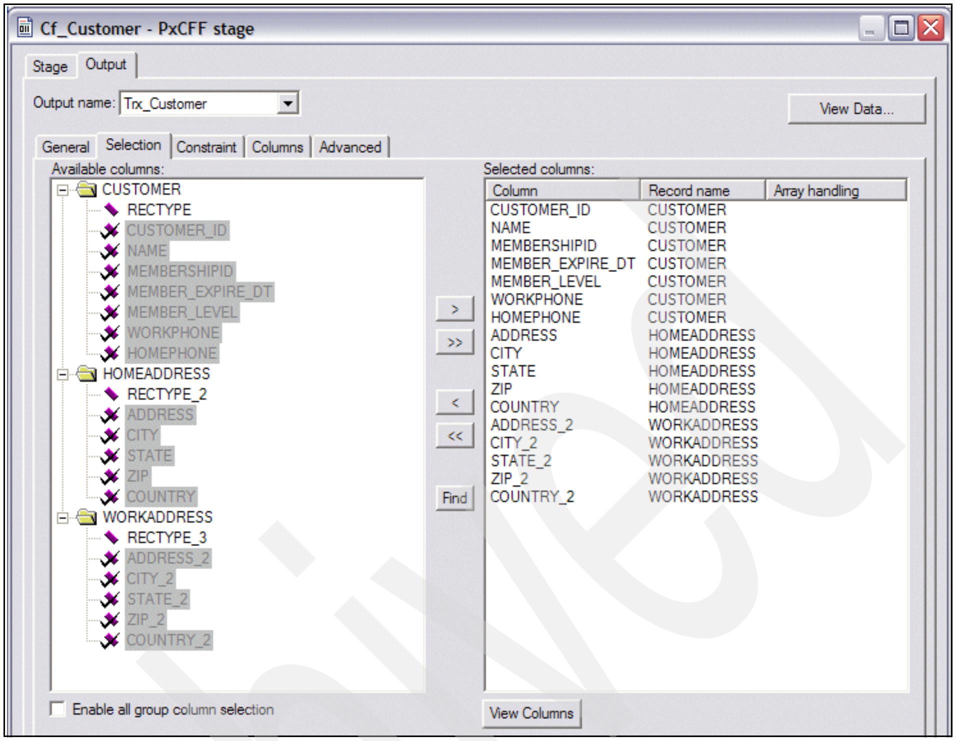


Figure 3-76 Create the J02_IL_LoadCustomerDim job 10/26

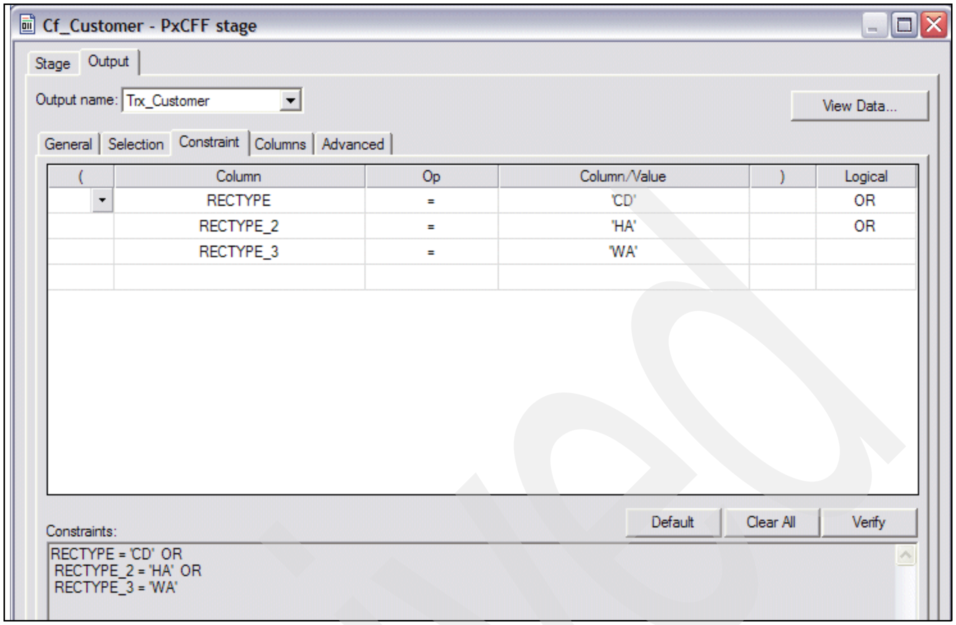
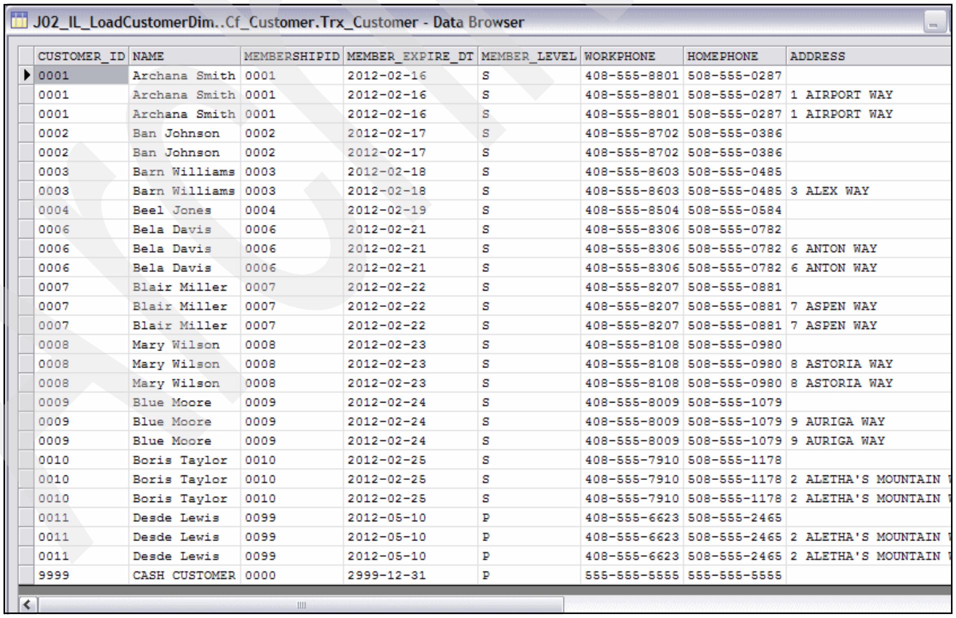
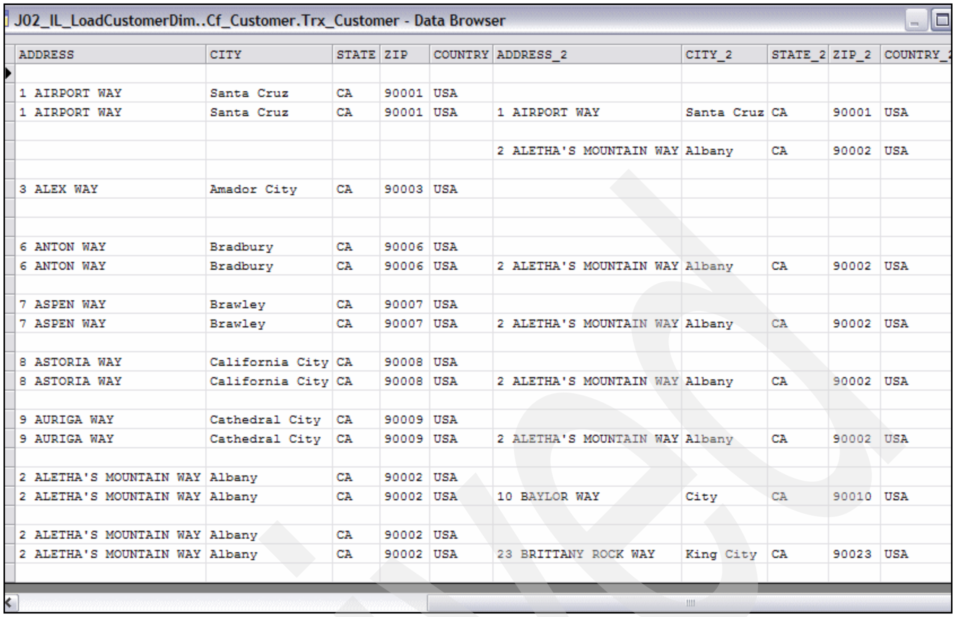


Figure 3-77 Create the J02_IL_LoadCustomerDim job 11/26



CUSTOMER_ID	NAME	MEMBERSHIPID	MEMBER_EXPIRE_DT	MEMBER_LEVEL	WORKPHONE	HOMEPHONE	ADDRESS
0001	Archana Smith	0001	2012-02-16	S	408-555-8801	508-555-0287	
0001	Archana Smith	0001	2012-02-16	S	408-555-8801	508-555-0287	1 AIRPORT WAY
0001	Archana Smith	0001	2012-02-16	S	408-555-8801	508-555-0287	1 AIRPORT WAY
0002	Ban Johnson	0002	2012-02-17	S	408-555-8702	508-555-0386	
0002	Ban Johnson	0002	2012-02-17	S	408-555-8702	508-555-0386	
0003	Barn Williams	0003	2012-02-18	S	408-555-8603	508-555-0485	
0003	Barn Williams	0003	2012-02-18	S	408-555-8603	508-555-0485	3 ALEX WAY
0004	Beel Jones	0004	2012-02-19	S	408-555-8504	508-555-0584	
0006	Bela Davis	0006	2012-02-21	S	408-555-8306	508-555-0782	
0006	Bela Davis	0006	2012-02-21	S	408-555-8306	508-555-0782	6 ANTON WAY
0006	Bela Davis	0006	2012-02-21	S	408-555-8306	508-555-0782	6 ANTON WAY
0007	Blair Miller	0007	2012-02-22	S	408-555-8207	508-555-0881	
0007	Blair Miller	0007	2012-02-22	S	408-555-8207	508-555-0881	7 ASPEN WAY
0007	Blair Miller	0007	2012-02-22	S	408-555-8207	508-555-0881	7 ASPEN WAY
0008	Mary Wilson	0008	2012-02-23	S	408-555-8108	508-555-0980	
0008	Mary Wilson	0008	2012-02-23	S	408-555-8108	508-555-0980	8 ASTORIA WAY
0008	Mary Wilson	0008	2012-02-23	S	408-555-8108	508-555-0980	8 ASTORIA WAY
0009	Blue Moore	0009	2012-02-24	S	408-555-8009	508-555-1079	
0009	Blue Moore	0009	2012-02-24	S	408-555-8009	508-555-1079	9 AURIGA WAY
0009	Blue Moore	0009	2012-02-24	S	408-555-8009	508-555-1079	9 AURIGA WAY
0010	Boris Taylor	0010	2012-02-25	S	408-555-7910	508-555-1178	
0010	Boris Taylor	0010	2012-02-25	S	408-555-7910	508-555-1178	2 ALETHA'S MOUNTAIN
0010	Boris Taylor	0010	2012-02-25	S	408-555-7910	508-555-1178	2 ALETHA'S MOUNTAIN
0011	Desde Lewis	0099	2012-05-10	P	408-555-6623	508-555-2465	
0011	Desde Lewis	0099	2012-05-10	P	408-555-6623	508-555-2465	2 ALETHA'S MOUNTAIN
0011	Desde Lewis	0099	2012-05-10	P	408-555-6623	508-555-2465	2 ALETHA'S MOUNTAIN
9999	CASH CUSTOMER	0000	2999-12-31	P	555-555-5555	555-555-5555	

Figure 3-78 Create the J02_IL_LoadCustomerDim job 12/26



J02_IL_LoadCustomerDim..Cf_Customer.Trx_Customer - Data Browser

ADDRESS	CITY	STATE	ZIP	COUNTRY	ADDRESS_2	CITY_2	STATE_2	ZIP_2	COUNTRY_2
1 AIRPORT WAY	Santa Cruz	CA	90001	USA					
1 AIRPORT WAY	Santa Cruz	CA	90001	USA	1 AIRPORT WAY	Santa Cruz	CA	90001	USA
					2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA
3 ALEX WAY	Amador City	CA	90003	USA					
6 ANTON WAY	Bradbury	CA	90006	USA					
6 ANTON WAY	Bradbury	CA	90006	USA	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA
7 ASPEN WAY	Brawley	CA	90007	USA					
7 ASPEN WAY	Brawley	CA	90007	USA	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA
8 ASTORIA WAY	California City	CA	90008	USA					
8 ASTORIA WAY	California City	CA	90008	USA	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA
9 AURIGA WAY	Cathedral City	CA	90009	USA					
9 AURIGA WAY	Cathedral City	CA	90009	USA	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA
2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA					
2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	10 BAYLOR WAY	City	CA	90010	USA
2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA					
2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	23 BRITTANY ROCK WAY	King City	CA	90023	USA

Figure 3-79 Create the J02_IL_LoadCustomerDim job 13/26

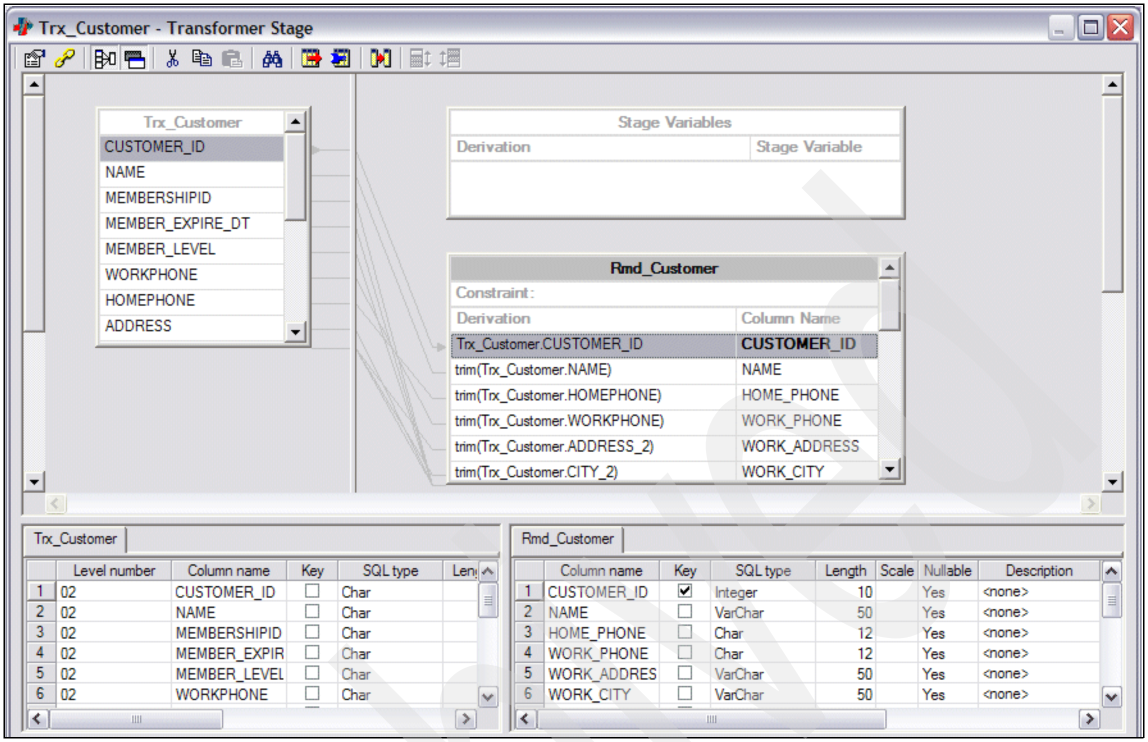


Figure 3-80 Create the J02_IL_LoadCustomerDim job 14/26

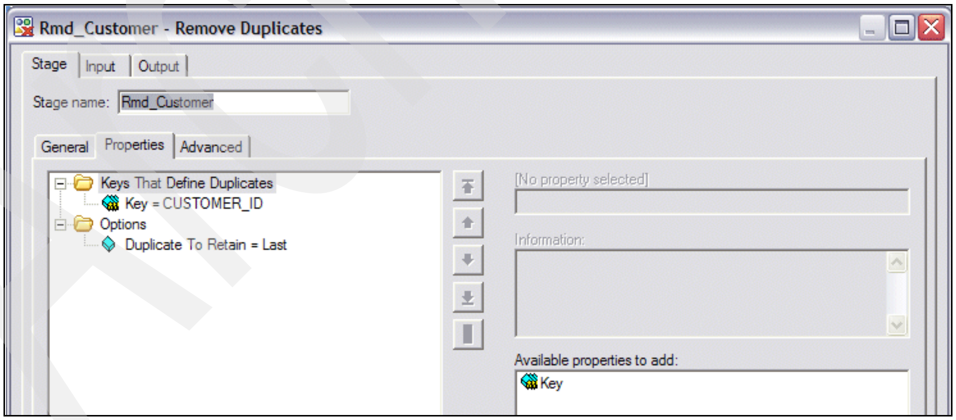


Figure 3-81 Create the J02_IL_LoadCustomerDim job 15/26

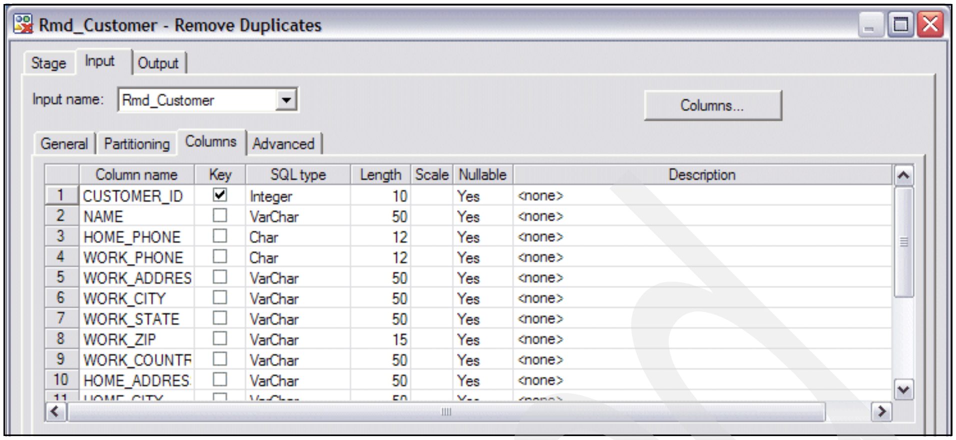


Figure 3-82 Create the J02_IL_LoadCustomerDim job 16/26

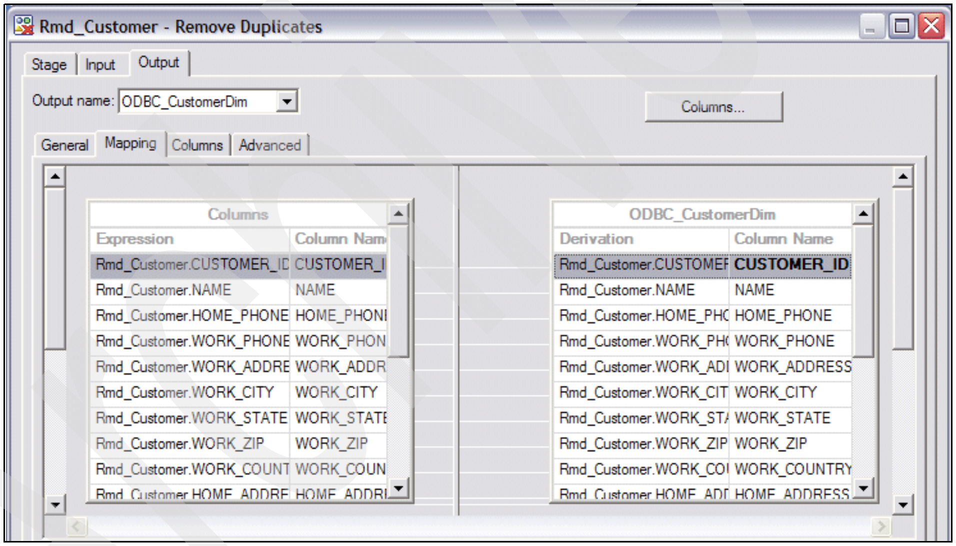


Figure 3-83 Create the J02_IL_LoadCustomerDim job 17/26

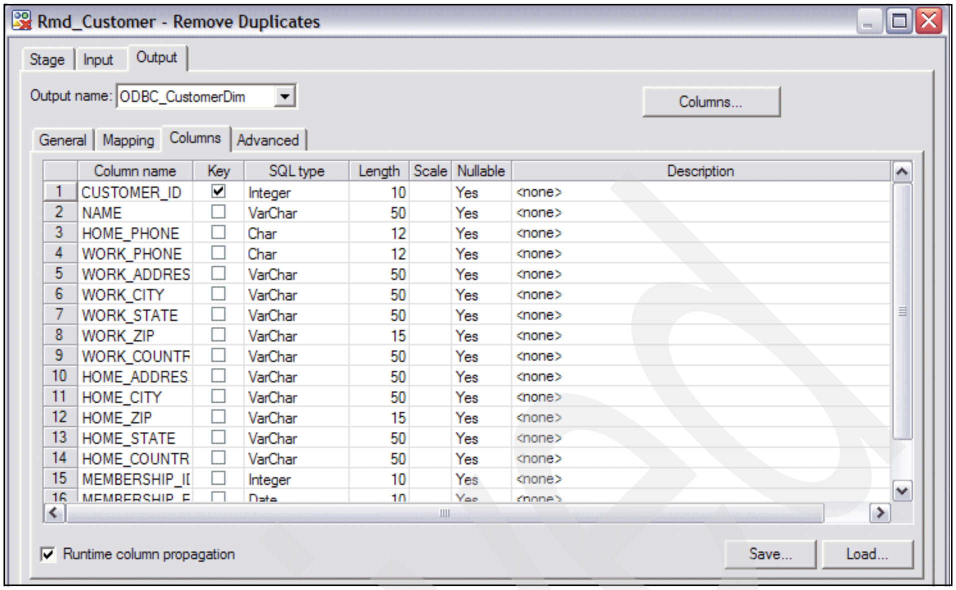


Figure 3-84 Create the J02_IL_LoadCustomerDim job 18/26

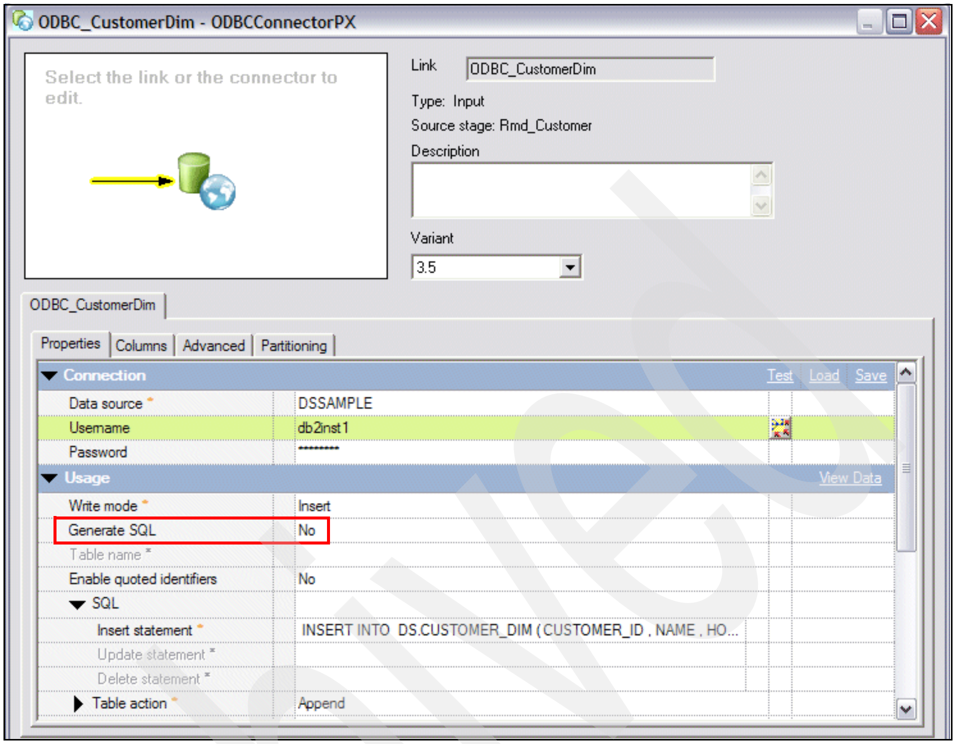


Figure 3-85 Create the J02_IL_LoadCustomerDim job 19/26

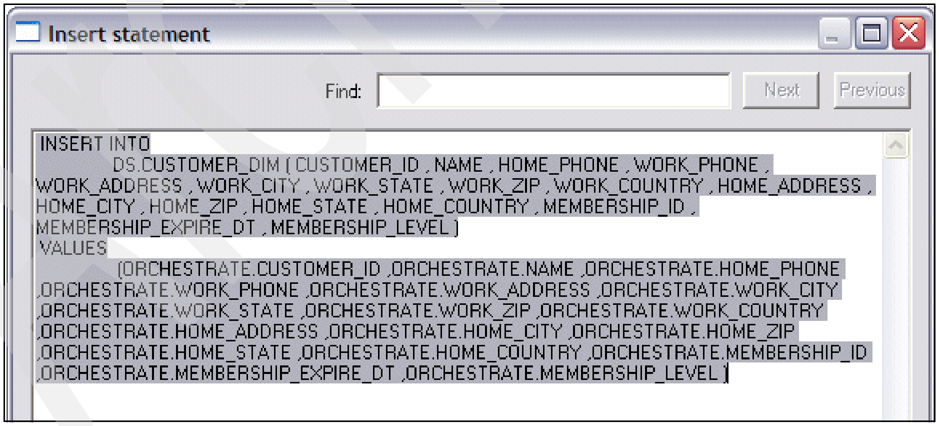


Figure 3-86 Create the J02_IL_LoadCustomerDim job 20/26

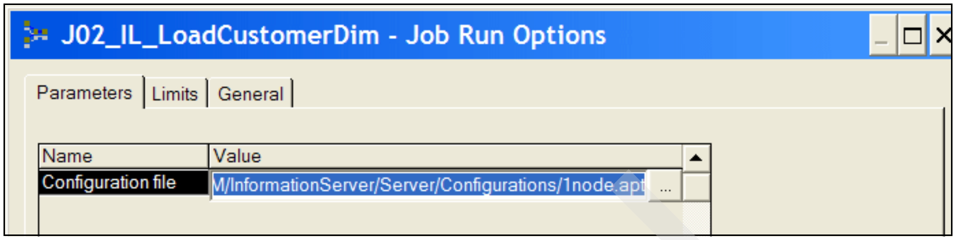


Figure 3-87 Create the J02_IL_LoadCustomerDim job 21/26

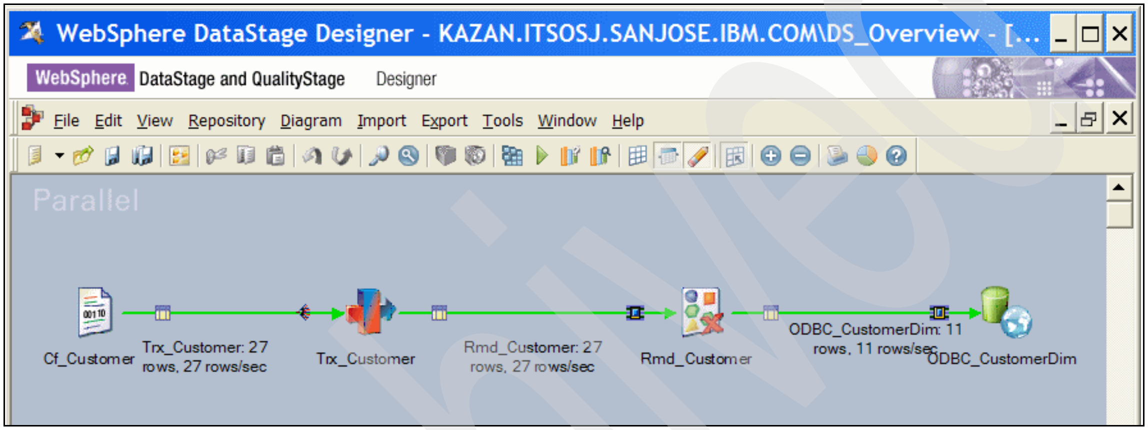


Figure 3-88 Create the J02_IL_LoadCustomerDim job 22/26

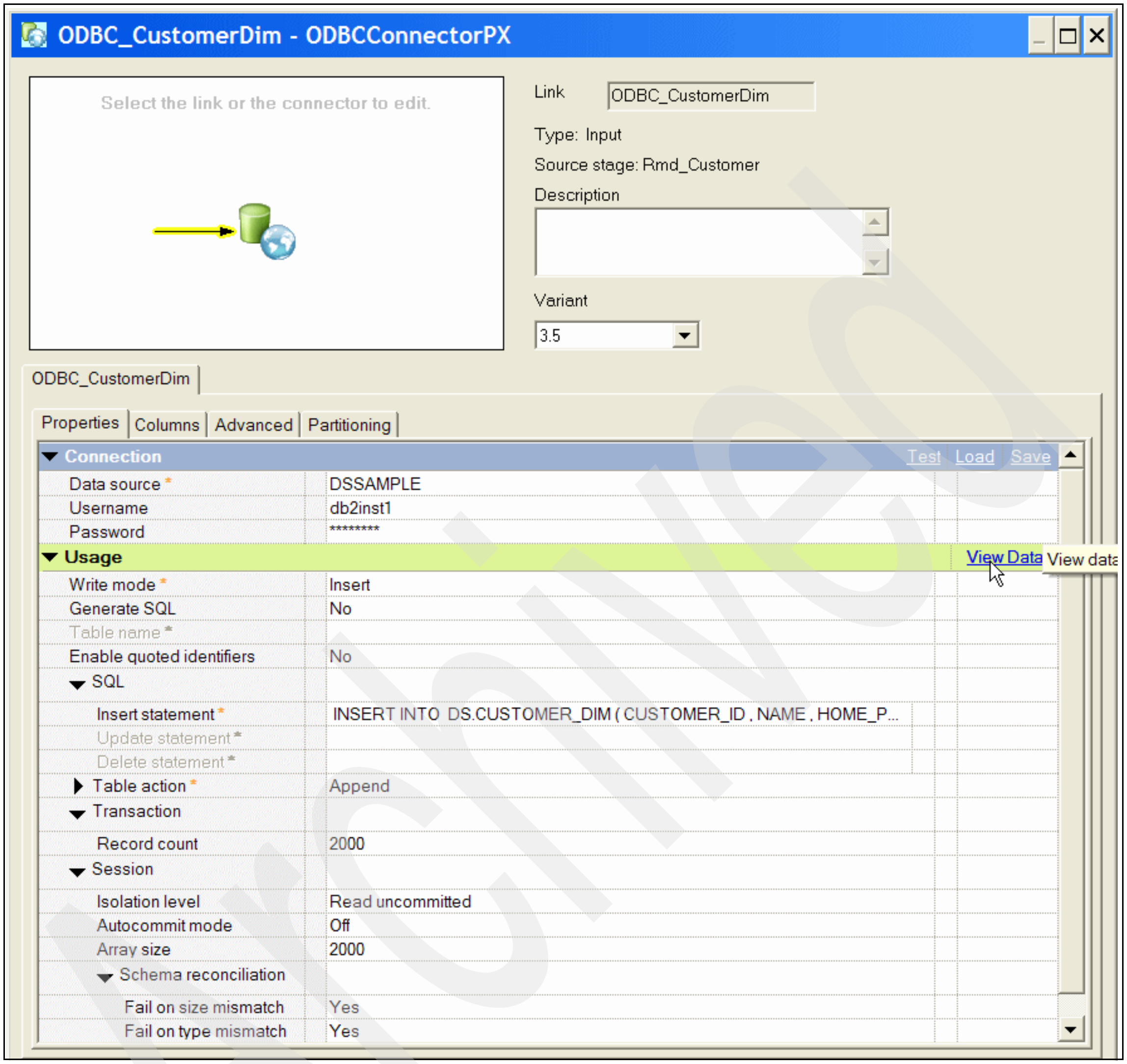


Figure 3-89 Create the J02_IL_LoadCustomerDim job 23/26

View Data

C...	CU...	NAME	HOME_PH...	WORK_PH...	WORK_ADDRESS	WORK_C...	W...	WO...	WO...	HOME_ADI
832	1	Archana Smith	508-555-0287	408-555-8801	1 AIRPORT WAY	Santa Cruz	CA	90001	USA	1 AIRPORT
833	2	Ban Johnson	508-555-0386	408-555-8702	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	
834	3	Barn Williams	508-555-0485	408-555-8603						3 ALEX WAY
835	4	Beel Jones	508-555-0584	408-555-8504						
836	6	Bela Davis	508-555-0782	408-555-8306	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	6 ANTON W
837	7	Blair Miller	508-555-0881	408-555-8207	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	7 ASPEN W.
838	8	Mary Wilson	508-555-0980	408-555-8108	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	8 ASTORIA
839	9	Blue Moore	508-555-1079	408-555-8009	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	9 AURIGA W
840	10	Boris Taylor	508-555-1178	408-555-7910	10 BAYLOR WAY	City	CA	90010	USA	2 ALETHA'S
841	11	Desde Lewis	508-555-2465	408-555-6623	23 BRITTANY ROCK WAY	King City	CA	90023	USA	2 ALETHA'S
842	9999	CASH CUSTOMER	555-555-5555	555-555-5555						

Figure 3-90 Create the J02_IL_LoadCustomerDim job 24/26

View Data

HOME_ADDRESS	HOME_CITY	HO...	H...	HO...	M...	MEMBERSHIP_EXPIRE_DT	M.	C.	EFFECTIVE_TS
1 AIRPORT WAY	Santa Cruz	90001	CA	USA	1	Thursday, February 16, 2012	S	Y	Monday, November 5, 2007 12:0
					2	Friday, February 17, 2012	S	Y	Monday, November 5, 2007 12:0
3 ALEX WAY	Amador City	90003	CA	USA	3	Saturday, February 18, 2012	S	Y	Monday, November 5, 2007 12:0
					4	Sunday, February 19, 2012	S	Y	Monday, November 5, 2007 12:0
6 ANTON WAY	Bradbury	90006	CA	USA	6	Tuesday, February 21, 2012	S	Y	Monday, November 5, 2007 12:0
7 ASPEN WAY	Brawley	90007	CA	USA	7	Wednesday, February 22, 2012	S	Y	Monday, November 5, 2007 12:0
8 ASTORIA WAY	California City	90008	CA	USA	8	Thursday, February 23, 2012	S	Y	Monday, November 5, 2007 12:0
9 AURIGA WAY	Cathedral City	90009	CA	USA	9	Friday, February 24, 2012	S	Y	Monday, November 5, 2007 12:0
2 ALETHA'S MOUNTAIN WAY	Albany	90002	CA	USA	10	Saturday, February 25, 2012	S	Y	Monday, November 5, 2007 12:0
2 ALETHA'S MOUNTAIN WAY	Albany	90002	CA	USA	99	Thursday, May 10, 2012	P	Y	Monday, November 5, 2007 12:0
					0	Tuesday, December 31, 2999	P	Y	Monday, November 5, 2007 12:0

Figure 3-91 Create the J02_IL_LoadCustomerDim job 25/26

View Data

)...	M..	MEMBERSHIP_EXPIRE_DT	M.	C.	EFFECTIVE_TS	EXPIRATION_TS
A	1	Thursday, February 16, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
	2	Friday, February 17, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
A	3	Saturday, February 18, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
	4	Sunday, February 19, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
A	6	Tuesday, February 21, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
A	7	Wednesday, February 22, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
A	8	Thursday, February 23, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
A	9	Friday, February 24, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
A	10	Saturday, February 25, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
A	99	Thursday, May 10, 2012	P	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
	0	Tuesday, December 31, 2999	P	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT

Figure 3-92 Create the J02_IL_LoadCustomerDim job 26/26

J03_IL_LoadProductDim

In this job, we extract relevant attributes from the Product VSAM file and load them into the PRODUCT_DIM dimension table. Since the Product information is stored in a VSAM file on the mainframe, we used the Classic Federation stage to access and retrieve the contents of this file.

Our objective in storing Product information in a VSAM file on the mainframe was to showcase the Classic Federation stage of IBM InfoSphere DataStage.

Figure 3-93 on page 203 through Figure 3-104 on page 209 describe the steps using Designer Client to build and execute the DataStage job to perform this task.

The steps are as follows:

1. Figure 3-93 on page 203 shows the various stages used in this job — it includes a Classic Federation stage, a Transformer stage, and an ODBCConnector stage. The names of the stages were modified as shown.
2. Figure 3-94 on page 204 and Figure 3-95 on page 204 show the configuration of the Classic Federation stage. The Output page allows you to specify details about how the Classic Federation stage accesses data from a remote host and writes it to an output link.
   - Figure 3-94 on page 204 shows the **Properties** tab in the Output page, which allows you to specify properties that determine what the stage actually does:
     - The Source category property Read Method = Table specifies a relational table that is identified by the Table = CAC.PRODUCT property.

       > **Note:** The IBM InfoSphere Classic Federation configuration of the Product VSAM file (to be accessed as a relational table) was done using Classic Data Architect as described in "Configuration of Classic Data Architect" on page 574.

     - The Connection category allows you to specify the User name (nalur1) and Password to access the CAC.PRODUCT table.
   - Figure 3-95 on page 204 shows the **Columns** tab in the Output page where you identify all the columns associated with this table.
3. The Transformer stage is used to trim the trailing blanks in the various fields using the TRIM function as shown in Figure 3-96 on page 205.

4. The ODBCConnectorPX stage does a simple SQL INSERT of the product information into the PRODUCT_DIM dimension table as shown in Figure 3-97 on page 206. The SQL INSERT statement is automatically generated.
5. The execution results of this job is shown in Figure 3-98 on page 207. It shows 4 records from the Classic Federation stage being inserted into the PRODUCT_DIM table.
6. Figure 3-99 on page 207 through Figure 3-102 on page 208 show the 4 records that are input to the ODBCConnectorPX stage.
7. Figure 3-103 on page 208 and Figure 3-104 on page 209 show the rows in the PRODUCT_DIM table using the DB2 Control Center.

We then proceeded to FTP the Employee file from the mainframe as described in “J04_IL_FTPEmployeeFile” on page 209.

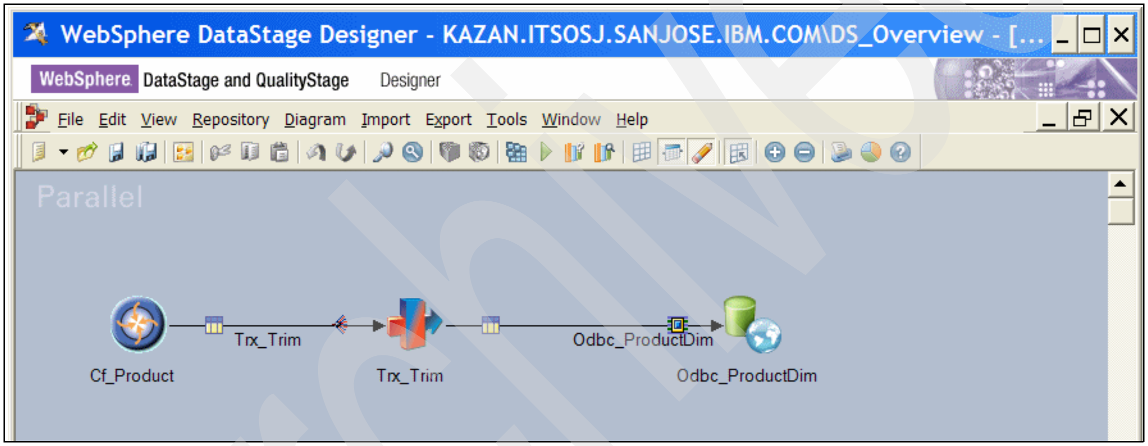


Figure 3-93 Create the J03_IL_LoadProductDim job 1/12

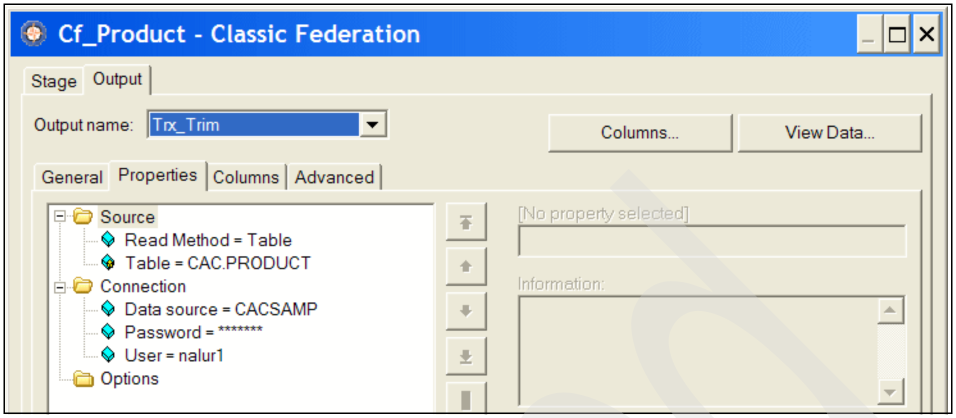


Figure 3-94 Create the J03_IL_LoadProductDim job 2/12

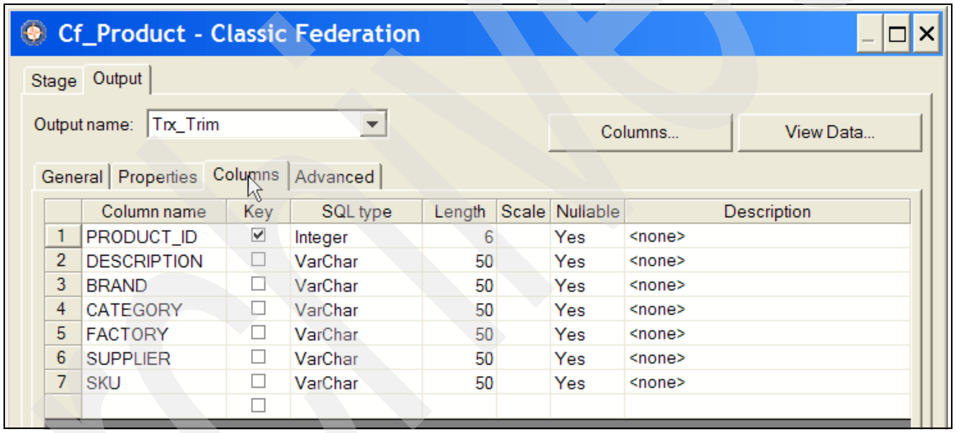


Figure 3-95 Create the J03_IL_LoadProductDim job 3/12

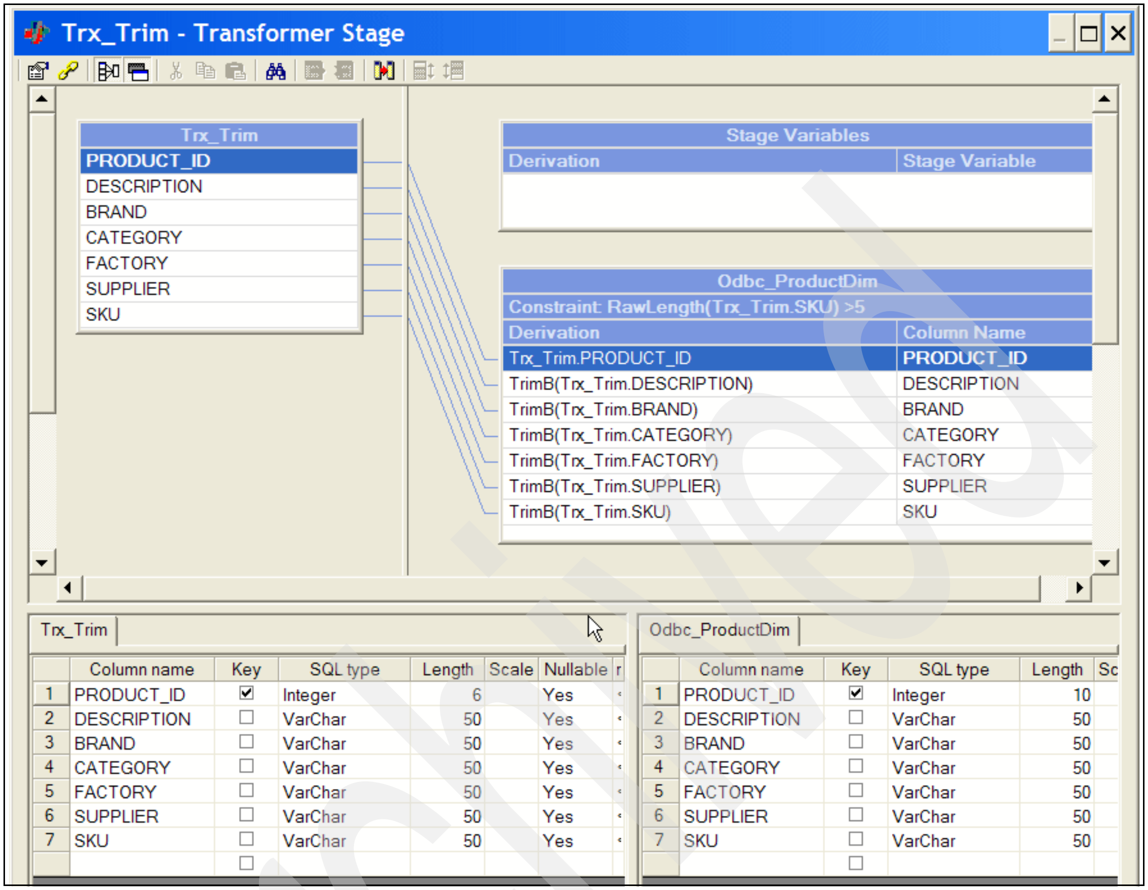


Figure 3-96 Create the J03_IL_LoadProductDim job 4/12

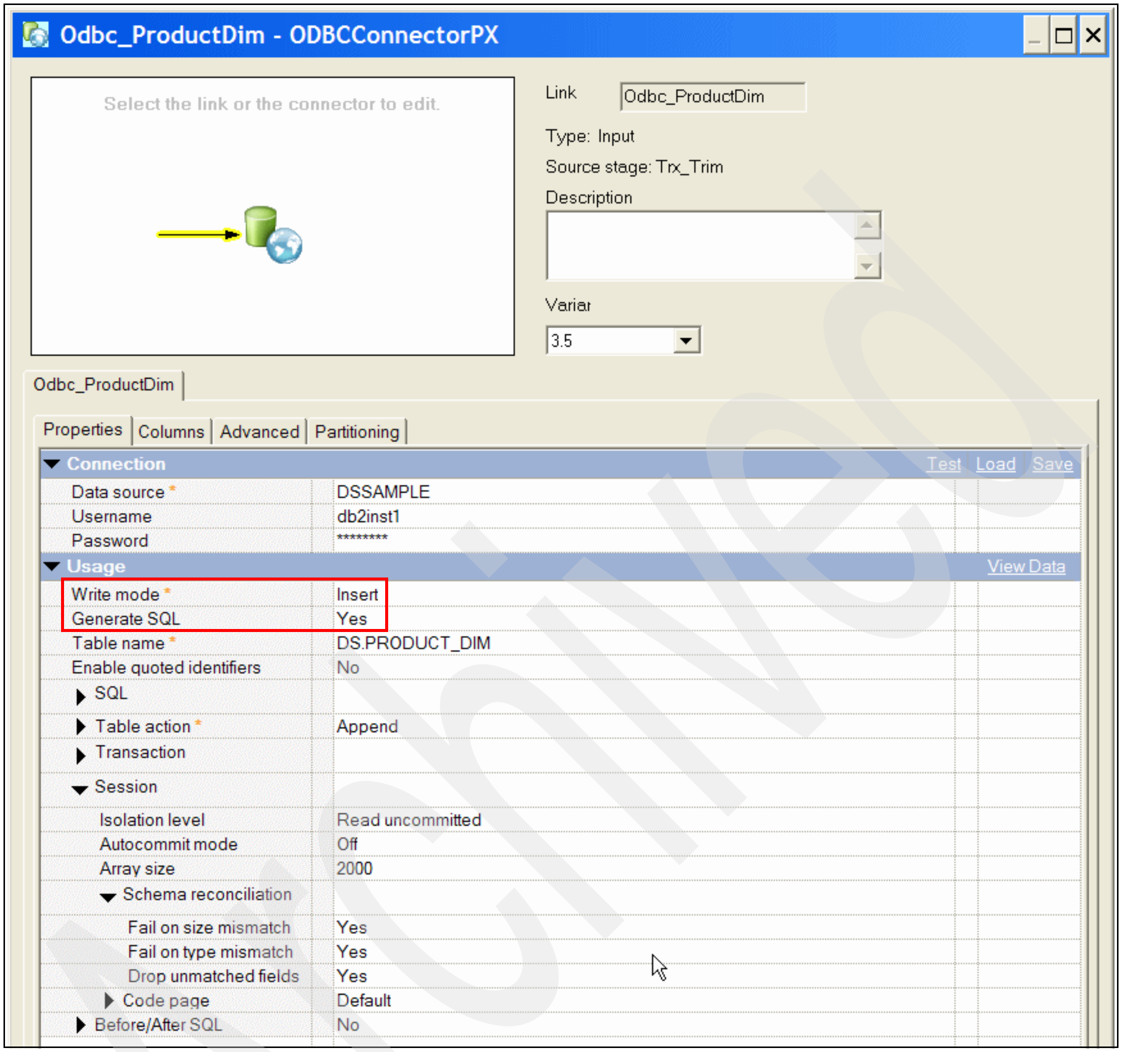


Figure 3-97 Create the J03_IL_LoadProductDim job 5/12

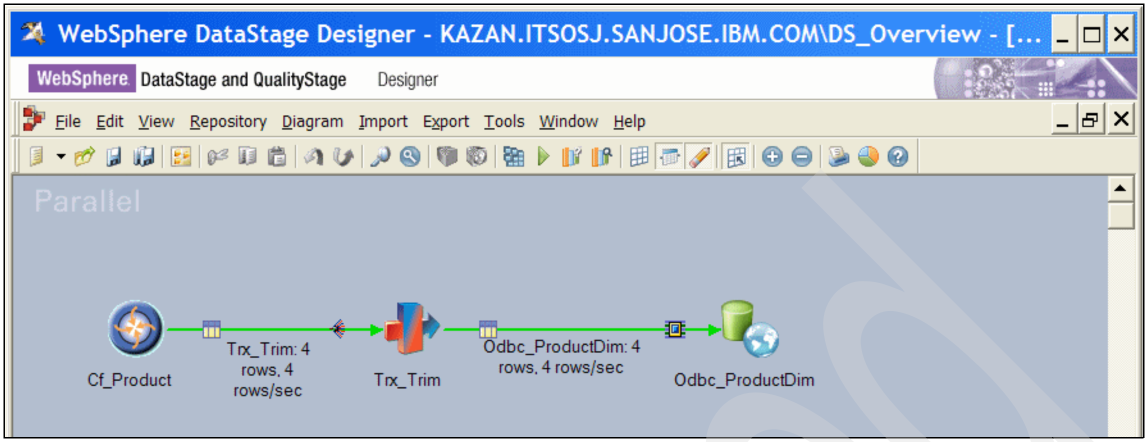


Figure 3-98 Create the J03_IL_LoadProductDim job 6/12

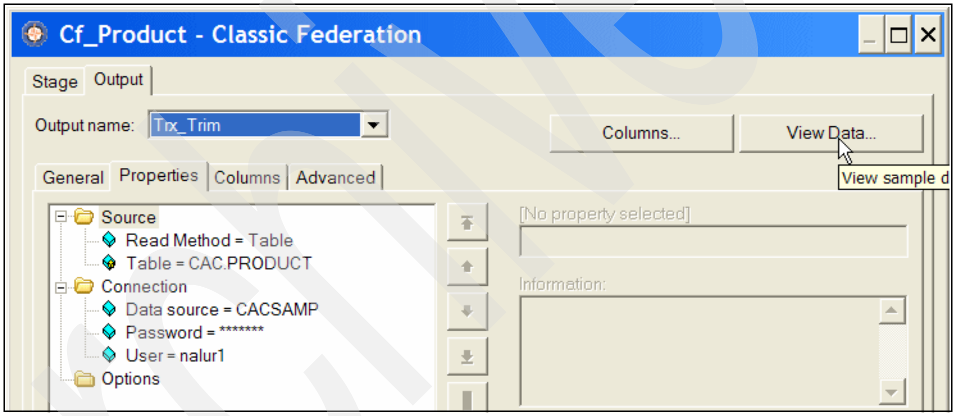


Figure 3-99 Create the J03_IL_LoadProductDim job 7/12

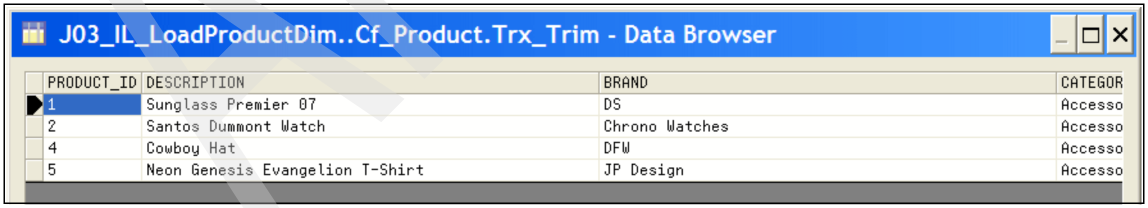


Figure 3-100 Create the J03_IL_LoadProductDim job 8/12

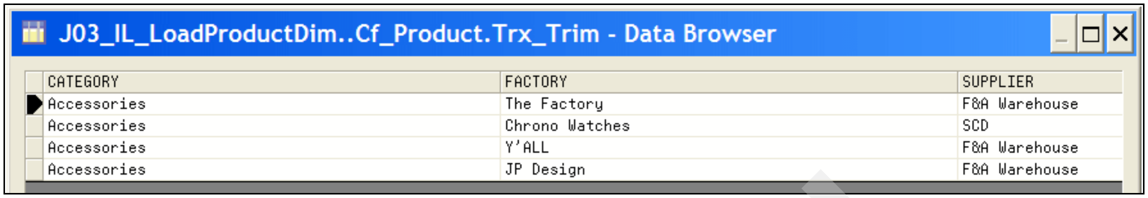


Figure 3-101 Create the J03_IL_LoadProductDim job 9/12

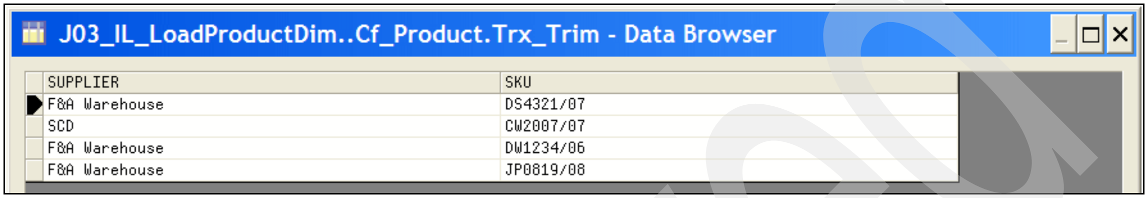


Figure 3-102 Create the J03_IL_LoadProductDim job 10/12

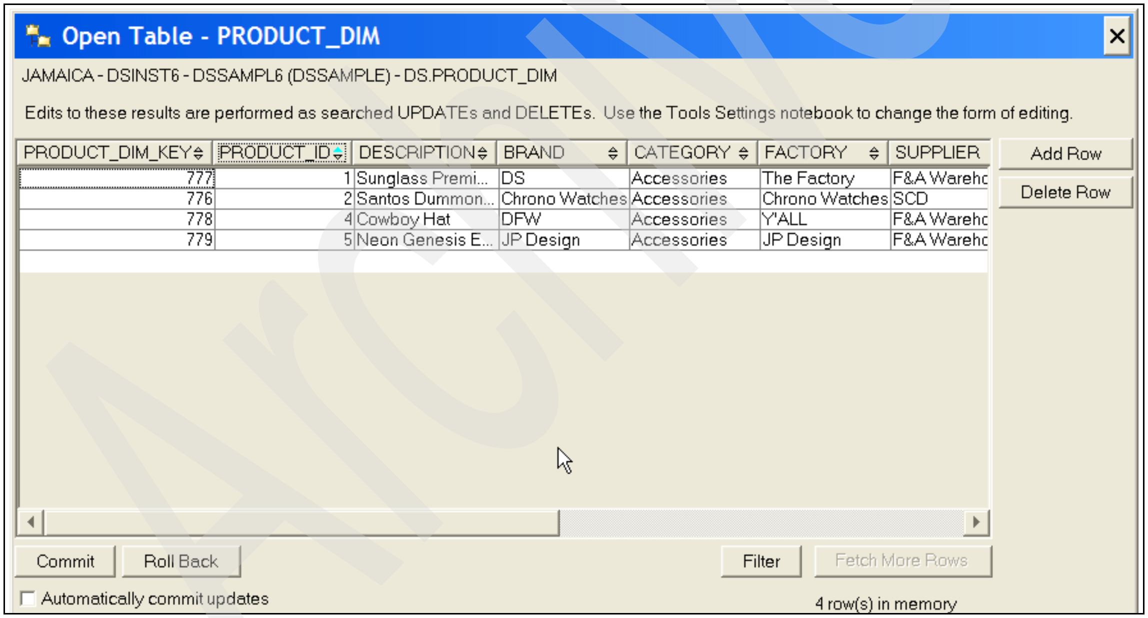


Figure 3-103 Create the J03_IL_LoadProductDim job 11/12

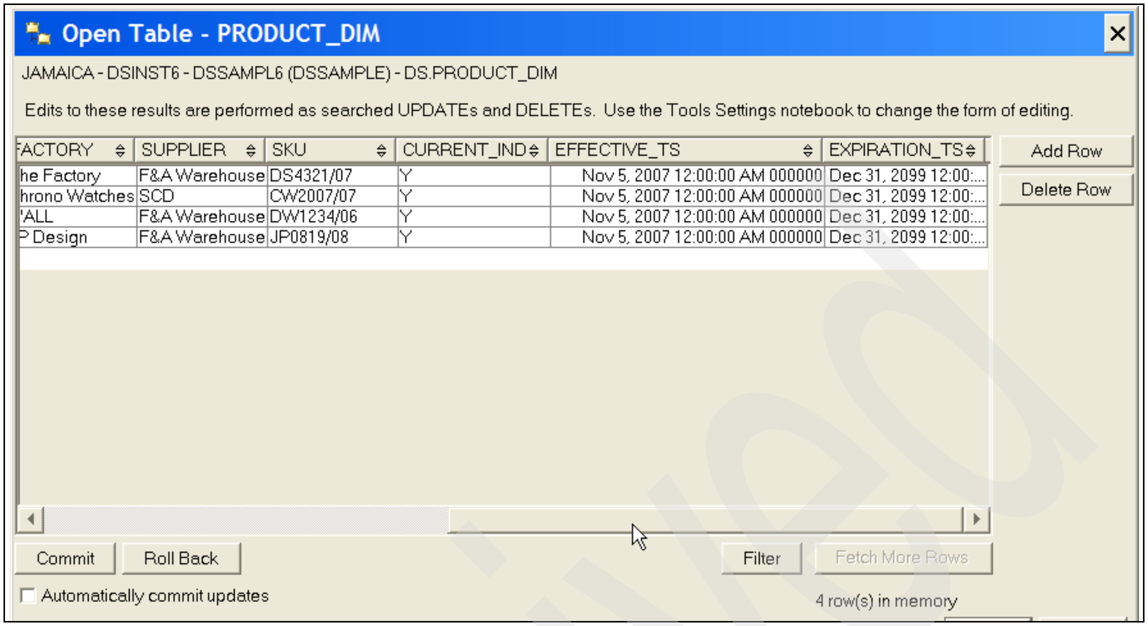


Figure 3-104 Create the J03_IL_LoadProductDim job 12/12

J04_IL_FTPEmployeeFile

In this job, we use the IBM InfoSphere DataStage FTP Enterprise stage to file transfer the EMPLOYEE sequential (EBCDIC) file from the mainframe to the Linux platform. Only the employees (manager id, first name, and last name) that are managers are extracted from this file using a Filter stage, and the first name and last name is then concatenated into a single column using a Transformer stage.

This manager information (manager id, first name and last name) is extracted in this step, so that it can be associated with store information (that only has a manager id associated with it) for populating the STORE_DIM dimension table. This association and loading of the STORE_DIM table is described in "J05_IL_LoadStoreDim" on page 219.

Figure 3-105 on page 211 through Figure 3-121 on page 218 describe the steps using Designer Client to build and execute the DataStage job to perform this task:

The steps are as follows:

1. Figure 3-105 on page 211 shows the various stages used in this job — it includes an FTP Enterprise stage, a Filter stage, Transformer stage, and a Data Set stage. The names of the stages were modified as shown.

2. Figure 3-106 on page 212 through Figure 3-108 on page 213 show the configuration of the FTP Enterprise stage. The Output page allows you to specify details about how the FTP Enterprise stage transfers one or more files from a remote host using the FTP protocol.
   - Figure 3-106 on page 212 shows the **Properties** tab in the Output page, which allows you to specify properties that determine what the stage actually does.
     - The Source category property URI specifies the pathname connecting the Stage to a source file on a remote host which corresponds to the Employee file on the mainframe.
     - The Connection category allows you to specify the User name (nalur1) and Password to access the data source identified by the URI.
     - The Transfer Protocol category Transfer Mode property is FTP.
     - The Options category Transfer Type is Binary.
   - Figure 3-107 on page 212 shows the **Format** tab in the Output page, which gives information about the format of the output.
   - Figure 3-108 on page 213 shows the **Columns** tab in the Output page where you identify the column definitions for this file. Runtime column propagation is not enabled here.
3. Figure 3-109 on page 213 shows the 20 rows retrieved from the Employee table which contains manager and non-manager records (Manager_Indicator of Y or N).
4. Figure 3-110 on page 214 through Figure 3-114 on page 216 show the configuration of the Filter stage that extracts only the manager records containing manager id, first name, and last name and writes it to the output link:
   - Figure 3-110 on page 214 shows the **Properties** tab in the Stage page, which specifies the predicate (MANAGER_INDICATOR='Y') to filter only managers as shown in the Predicates category Where Clause property.
   - Figure 3-111 on page 214 shows the **Link Ordering** tab in the Stage page that specifies the mapping of the qualifying rows to the output link (Trx_ConcatName).
   - Figure 3-112 on page 215 shows the **Columns** tab in the Input page that specifies the column definitions of the data being read.
   - Figure 3-113 on page 215 shows the **Mapping** tab in the Output page that specifies how the input columns are mapped to the Output name (Trx_ConcatName). Only the MANAGER_ID, FIRST_NAME, and LAST_NAME are mapped from the input to the output.

5. Figure 3-114 on page 216 shows the **Columns** tab in the Output page identifying the three output columns. Runtime column propagation is not selected.
6. The first name and last name columns are concatenated together into a single column using a Transformer stage as shown in Figure 3-115 on page 216 and Figure 3-116 on page 217.
7. The transformed data is written to a data set (with two columns MANAGER_ID and NAME containing the concatenated first name and last name values) as shown in the configuration of the Data Set stage in Figure 3-117 on page 217 and Figure 3-118 on page 217.
8. The results of the execution of this job is shown in Figure 3-119 on page 218, which shows 6 manager records out of a total of 20 employee records. Figure 3-120 on page 218 shows the 20 employee records, while Figure 3-121 on page 218 shows the 6 manager records and their concatenated name data that is written to the output data set.

This information is merged with store information (that only has the manager id of the employee but not the first name and last name information of the manager) to load the STORE_DIM table (that has a column for name information) as described in “J05_IL_LoadStoreDim” on page 219.

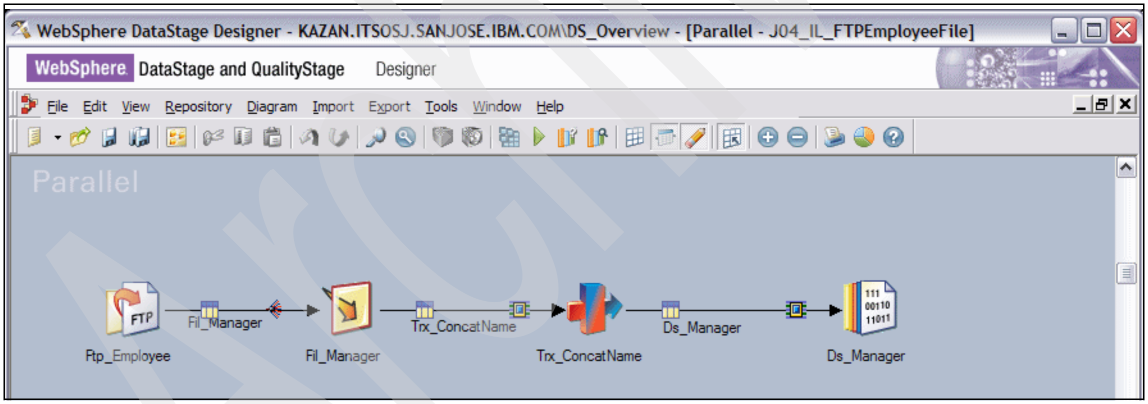


Figure 3-105 Create the J04_IL_FTPEmployeeFile job 1/17

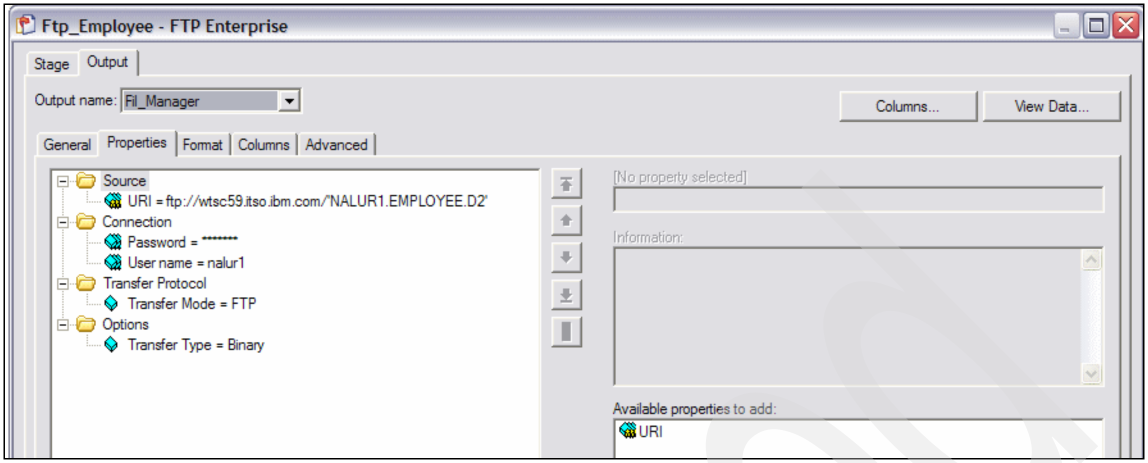


Figure 3-106   Create the J04_IL_FTPEmployeeFile job 2/17

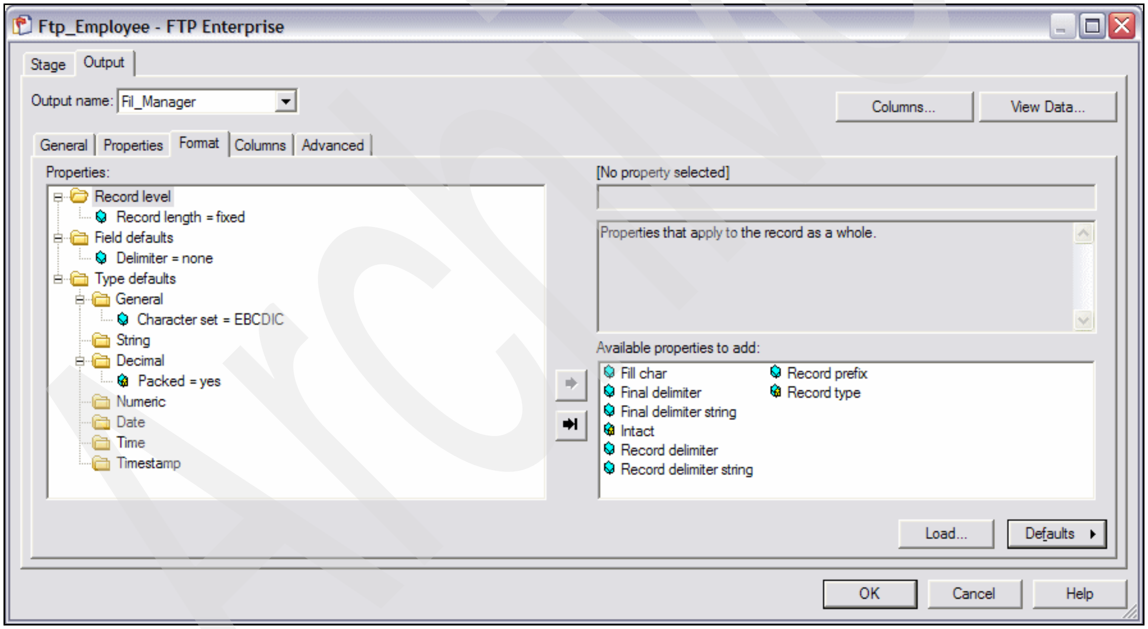
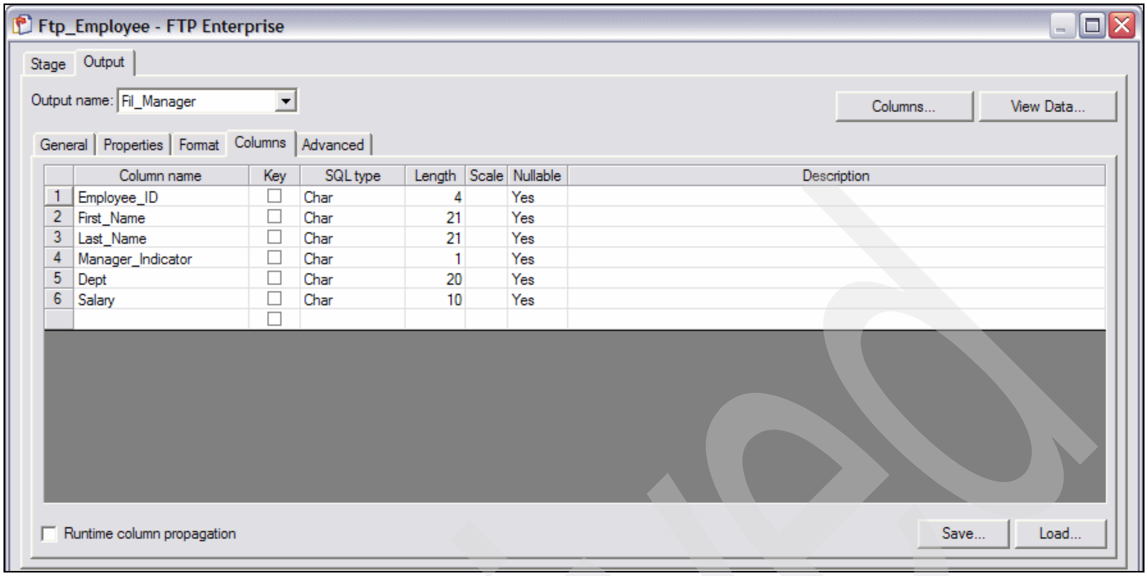
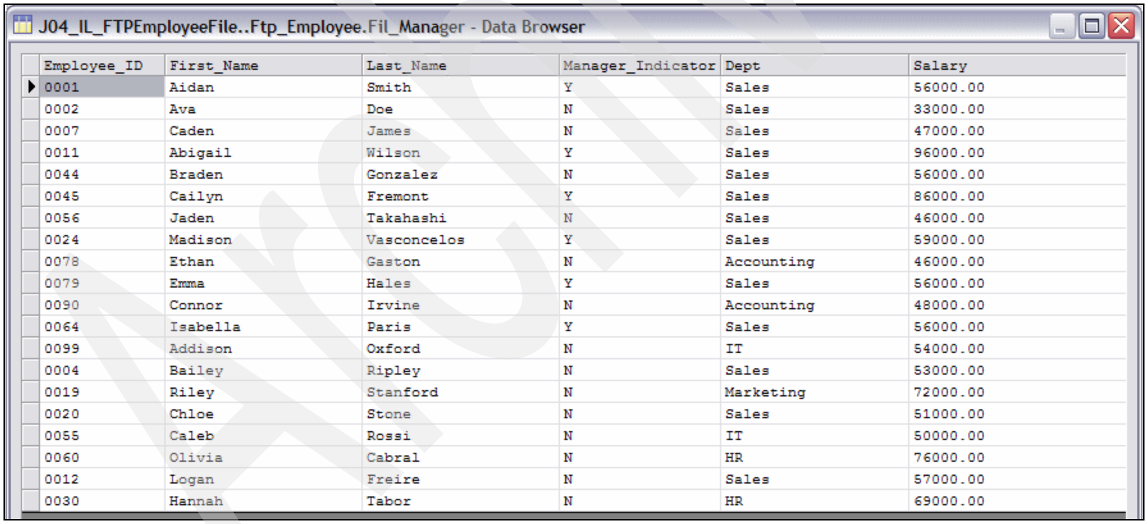


Figure 3-107   Create the J04_IL_FTPEmployeeFile job 3/17



	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	Employee_ID		Char	4		Yes	
2	First_Name		Char	21		Yes	
3	Last_Name		Char	21		Yes	
4	Manager_Indicator		Char	1		Yes	
5	Dept		Char	20		Yes	
6	Salary		Char	10		Yes	

Figure 3-108 Create the J04_IL_FTPEmployeeFile job 4/17



Employee_ID	First_Name	Last_Name	Manager_Indicator	Dept	Salary
0001	Aidan	Smith	Y	Sales	56000.00
0002	Ava	Doe	N	Sales	33000.00
0007	Caden	James	N	Sales	47000.00
0011	Abigail	Wilson	Y	Sales	96000.00
0044	Braden	Gonzalez	N	Sales	56000.00
0045	Cailyn	Fremont	Y	Sales	86000.00
0056	Jaden	Takahashi	N	Sales	46000.00
0024	Madison	Vasconcelos	Y	Sales	59000.00
0078	Ethan	Gaston	N	Accounting	46000.00
0079	Emma	Hales	Y	Sales	56000.00
0090	Connor	Irvine	N	Accounting	48000.00
0064	Isabella	Paris	Y	Sales	56000.00
0099	Addison	Oxford	N	IT	54000.00
0004	Bailey	Ripley	N	Sales	53000.00
0019	Riley	Stanford	N	Marketing	72000.00
0020	Chloe	Stone	N	Sales	51000.00
0055	Caleb	Rossi	N	IT	50000.00
0060	Olivia	Cabral	N	HR	76000.00
0012	Logan	Freire	N	Sales	57000.00
0030	Hannah	Tabor	N	HR	69000.00

Figure 3-109 Create the J04_IL_FTPEmployeeFile job 5/17

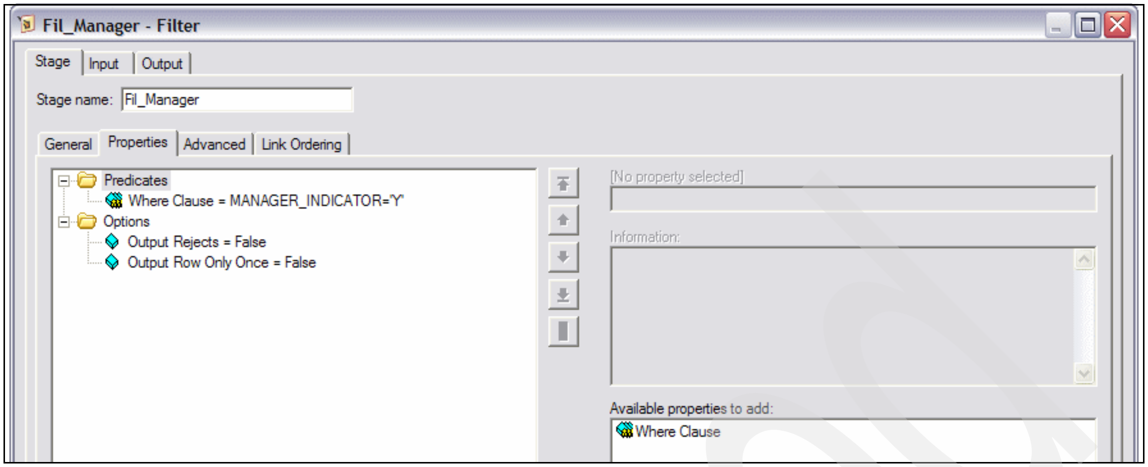


Figure 3-110   Create the J04_IL_FTPEmployeeFile job 6/17

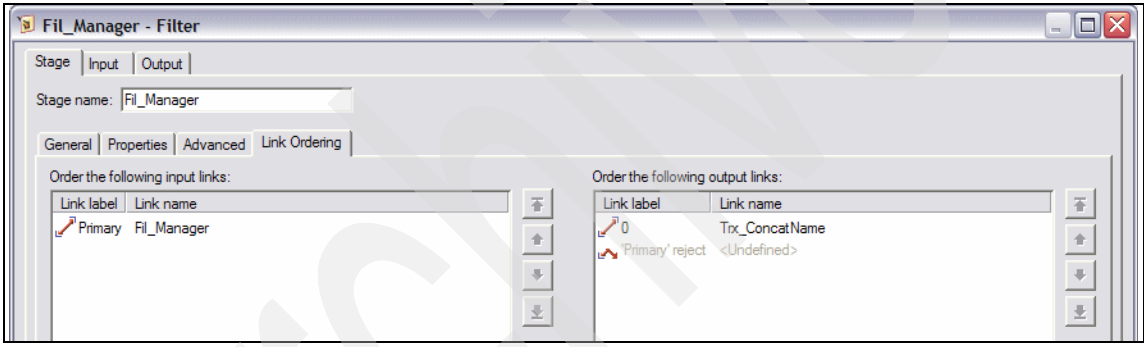


Figure 3-111   Create the J04_IL_FTPEmployeeFile job 7/17

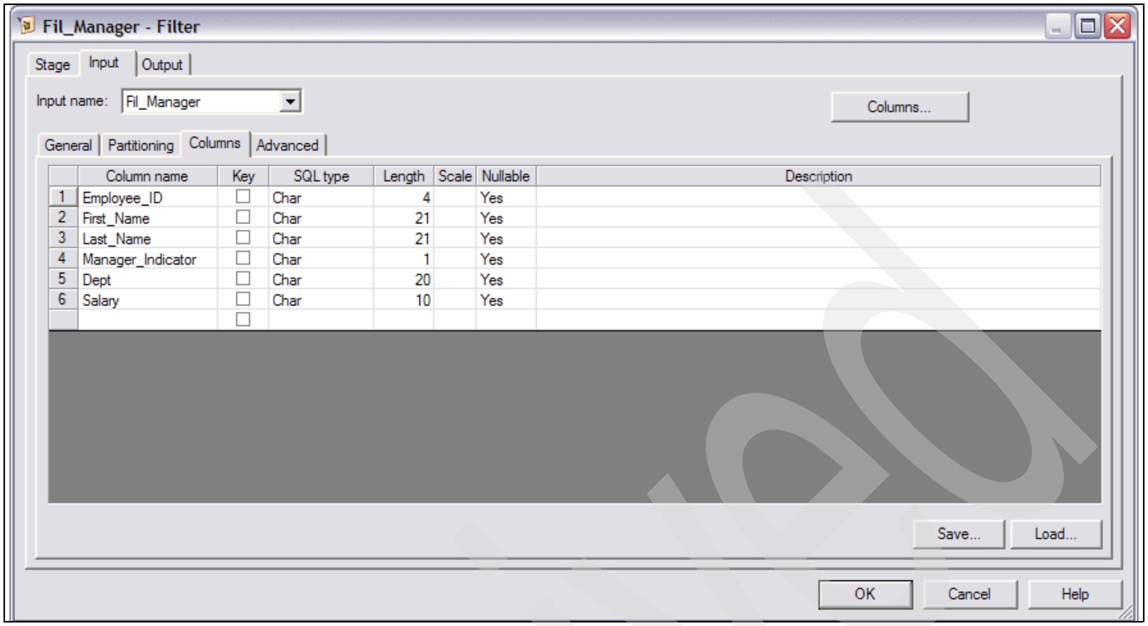


Figure 3-112 Create the J04_IL_FTPEmployeeFile job 8/17

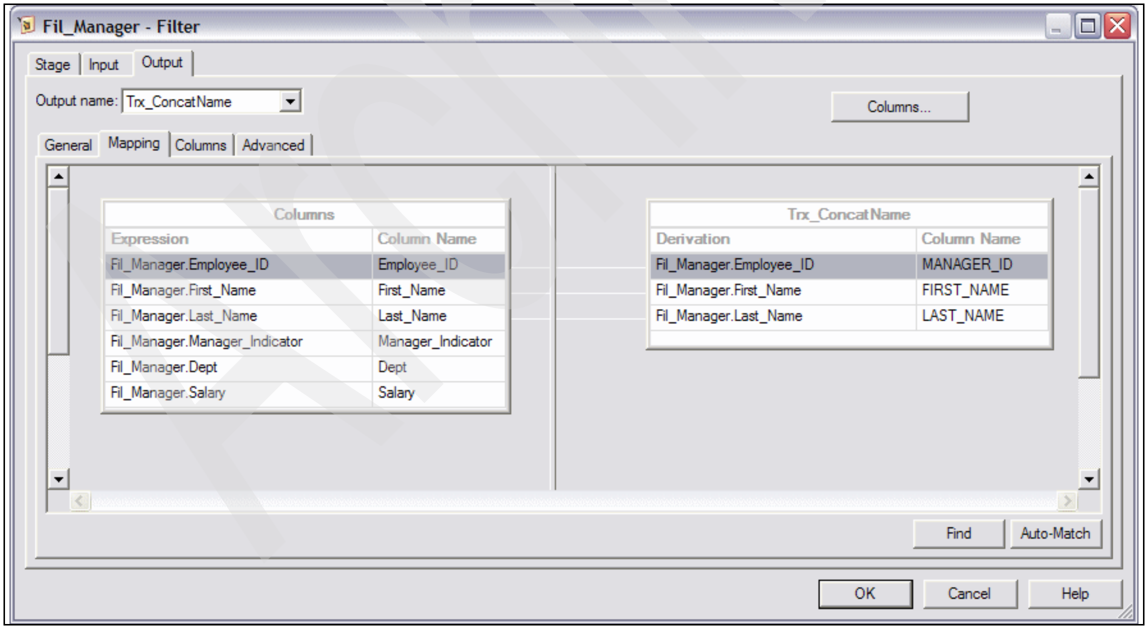


Figure 3-113 Create the J04_IL_FTPEmployeeFile job 9/17

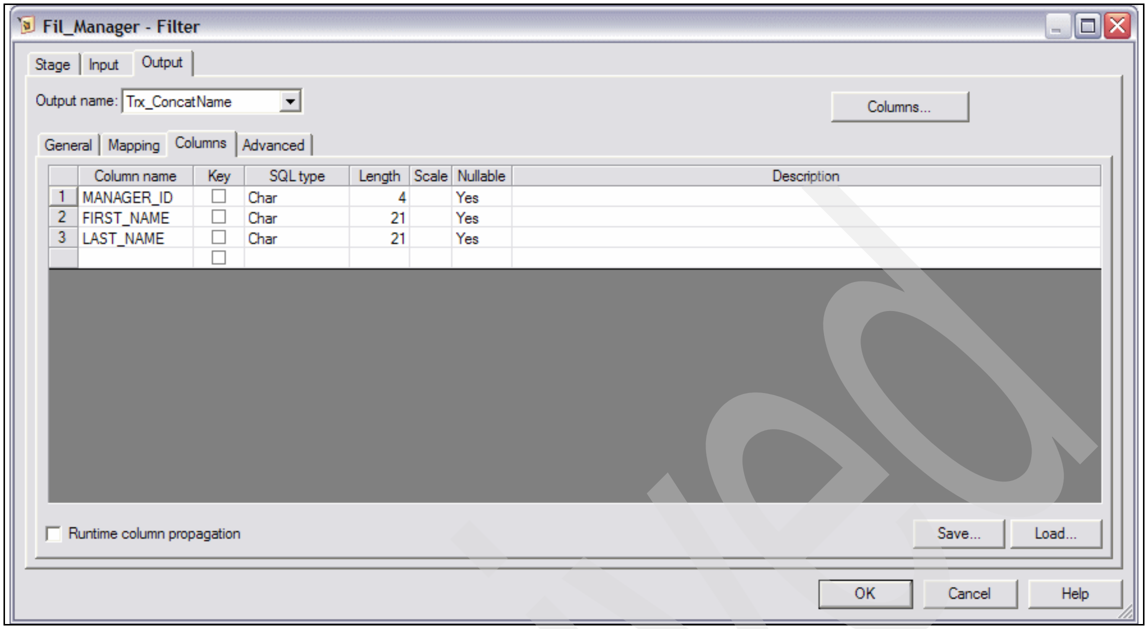


Figure 3-114 Create the J04_IL_FTPEmployeeFile job 10/17

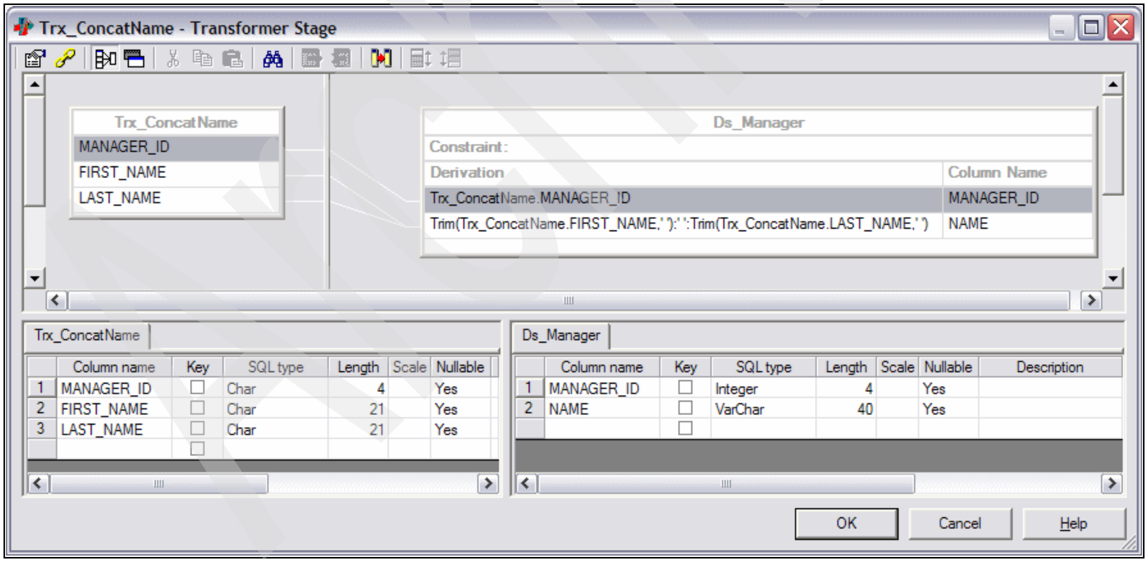


Figure 3-115 Create the J04_IL_FTPEmployeeFile job 11/17

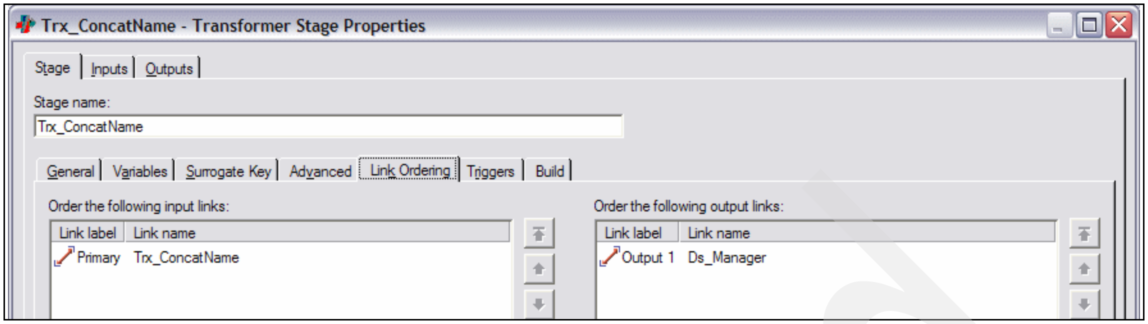


Figure 3-116   Create the J04_IL_FTPEmployeeFile job 12/17

Figure 3-117   Create the J04_IL_FTPEmployeeFile job 13/17

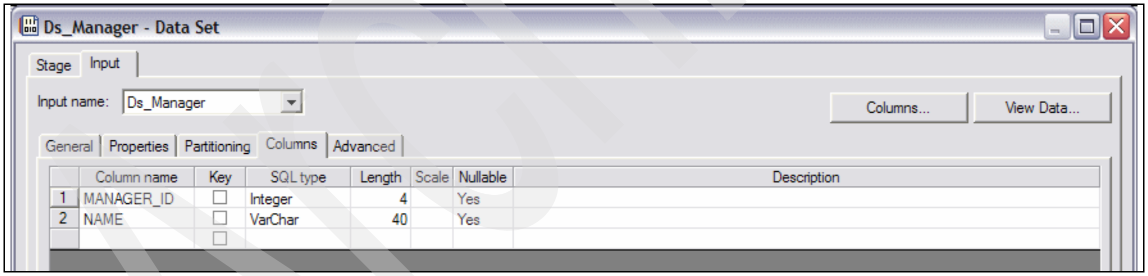


Figure 3-118   Create the J04_IL_FTPEmployeeFile job 14/17

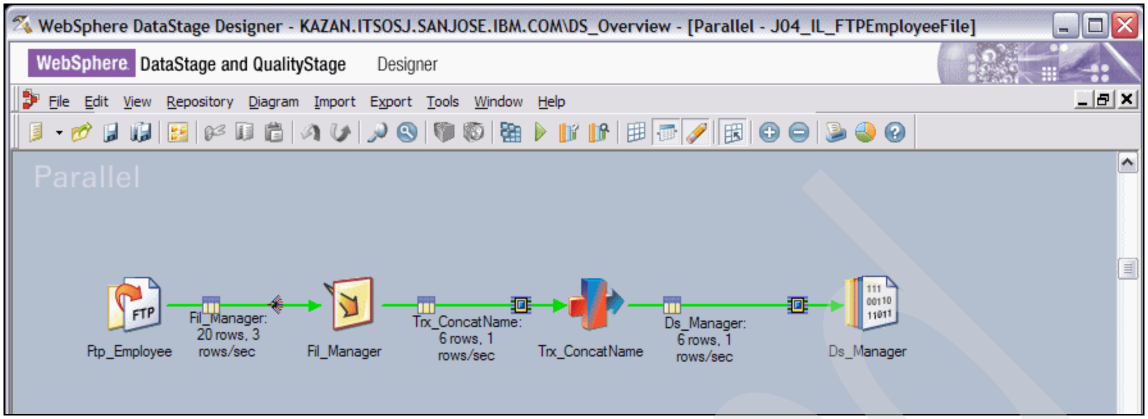
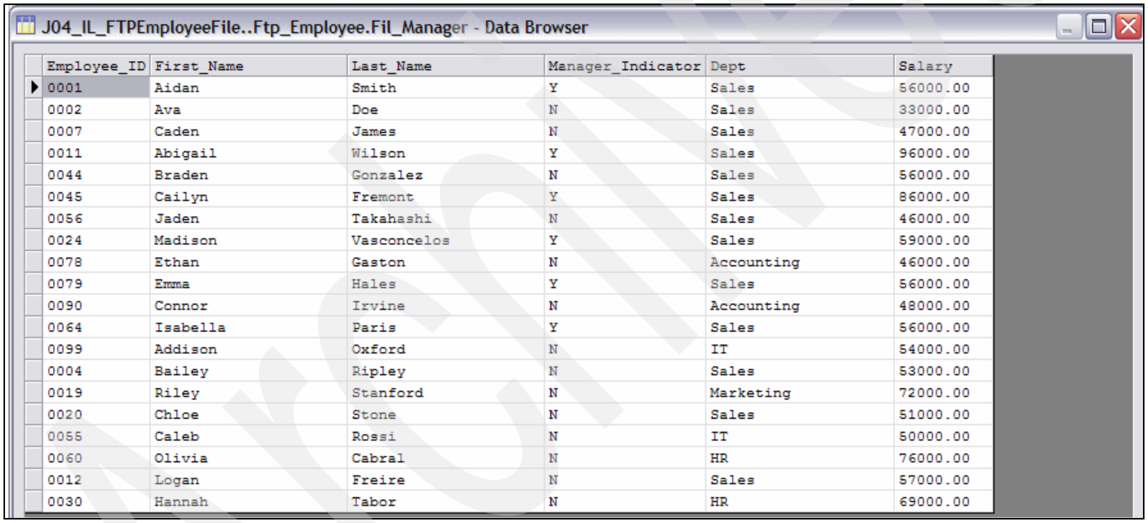
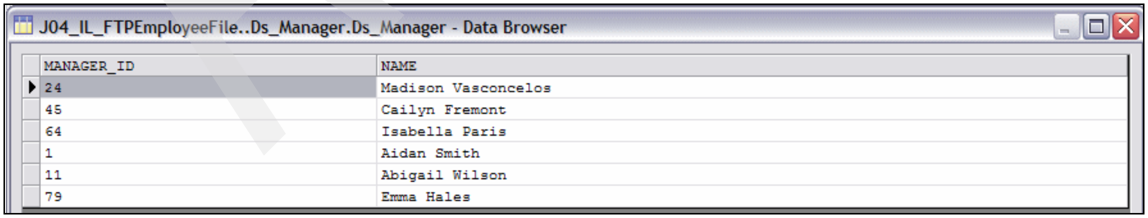


Figure 3-119 Create the J04_IL_FTPEmployeeFile job 15/17


J04_IL_FTPEmployeeFile..Ftp_Employee.Fil_Manager - Data Browser

Employee_ID	First_Name	Last_Name	Manager_Indicator	Dept	Salary
0001	Aidan	Smith	Y	Sales	56000.00
0002	Ava	Doe	N	Sales	33000.00
0007	Caden	James	N	Sales	47000.00
0011	Abigail	Wilson	Y	Sales	96000.00
0044	Braden	Gonzalez	N	Sales	56000.00
0045	Cailyn	Fremont	Y	Sales	86000.00
0056	Jaden	Takahashi	N	Sales	46000.00
0024	Madison	Vasconcelos	Y	Sales	59000.00
0078	Ethan	Gaston	N	Accounting	46000.00
0079	Emma	Hales	Y	Sales	56000.00
0090	Connor	Irvine	N	Accounting	48000.00
0064	Isabella	Paris	Y	Sales	56000.00
0099	Addison	Oxford	N	IT	54000.00
0004	Bailey	Ripley	N	Sales	53000.00
0019	Riley	Stanford	N	Marketing	72000.00
0020	Chloe	Stone	N	Sales	51000.00
0055	Caleb	Rossi	N	IT	50000.00
0060	Olivia	Cabral	N	HR	76000.00
0012	Logan	Freire	N	Sales	57000.00
0030	Hannah	Tabor	N	HR	69000.00

Figure 3-120 Create the J04_IL_FTPEmployeeFile job 16/17


J04_IL_FTPEmployeeFile..Ds_Manager.Ds_Manager - Data Browser

MANAGER_ID	NAME
24	Madison Vasconcelos
45	Cailyn Fremont
64	Isabella Paris
1	Aidan Smith
11	Abigail Wilson
79	Emma Hales

Figure 3-121 Create the J04_IL_FTPEmployeeFile job 17/17

J05_IL_LoadStoreDim

In this job, we extract relevant attributes from the Store VSAM file and join it with manager name information retrieved in the “J04_IL_FTPEmployeeFile” on page 209 job from the Employee file before loading the STORE_DIM dimension table. Since Store information is stored in a VSAM file on the mainframe, we used the Classic Federation stage to access and retrieve the contents of this file.

Our objective in storing Store information in a VSAM file on the mainframe was to showcase the Classic Federation stage of IBM InfoSphere DataStage.

Figure 3-122 on page 221 through Figure 3-137 on page 226 describe the steps using Designer Client to build and execute the DataStage job to perform this task.

The steps are as follows:

1. Figure 3-122 on page 221 shows the various stages used in this job — it includes a Classic Federation stage, a Data Set stage, a Join stage, and an ODBCConnector stage. The names of the stages were modified as shown.
2. Figure 3-123 on page 221 and Figure 3-124 on page 222 show the configuration of the Classic Federation stage. The Output page allows you to specify details about how the Classic Federation stage accesses data from a remote host and writes it to an output link.
   - Figure 3-123 on page 221 shows the **Properties** tab in the Output page, which allows you to specify properties that determine what the stage actually does.
     - The Source category property Read Method = Table specifies a relational table that is identified by the Table = CAC.STORE property.

       > **Note:** The IBM InfoSphere Classic Federation configuration of the Product VSAM file (to be accessed as a relational table) was done using Classic Data Architect as described in “Configuration of Classic Data Architect” on page 574.

     - The Connection category allows you to specify the User name (nalur1) and Password to access the CAC.STORE table.
   - Figure 3-124 on page 222 shows the **Columns** tab in the Output page where you identify all the columns associated with this Store file.
3. Figure 3-125 on page 222 shows the contents (2 records) of the Store file (table) as stored on the mainframe.

4. Figure 3-126 on page 222 and Figure 3-127 on page 223 show the configuration of the data set created in “J04_IL_FTPEmployeeFile” on page 209. They identify the file and the column definitions. Figure 3-128 on page 223 shows the contents of this data set.
5. Figure 3-129 on page 223 through Figure 3-131 on page 224 describe the configuration of the Join stage that joins the Store file retrieved from the mainframe in the Classic Federation stage with the filtered manager information from the Employee file generated in “J04_IL_FTPEmployeeFile” on page 209 on the MANAGER_ID column. The output of the join includes selected columns from the two input sources.
   - Figure 3-129 on page 223 shows the **Properties** tab in the Stage page that identifies the Key property in the Join Keys category as MANAGER_ID and the Join Type property as Inner (join).
   - Figure 3-130 on page 224 shows the **Link Ordering** tab in the Stage page that identifies the left and right links in the join. This is not relevant for an inner join, but would be relevant had a left or right outer join been chosen.
   - Figure 3-131 on page 224 shows the **Mapping** tab in the Output page that identifies the columns that will be mapped to the output from the two input sources being joined. It shows most of the columns in the Store file and the concatenated (manager) name, but excludes the manager id.
6. The ODBCConnectorPX stage generates a simple SQL INSERT of the store information into the STORE_DIM dimension table as shown in Figure 3-132 on page 225. The SQL INSERT statement is manually generated as shown in Figure 3-133 on page 225.
7. The execution results of this job is shown in Figure 3-134 on page 226. It shows 2 records from the Join stage being inserted into the STORE_DIM table.
8. Figure 3-135 on page 226 shows the 2 records that are input to the Join stage.
9. Figure 3-136 on page 226 and Figure 3-137 on page 226 show the rows inserted into the STORE_DIM dimension table.

We then proceeded to FTP the Employee file from the mainframe as described in “J04_IL_FTPEmployeeFile” on page 209.

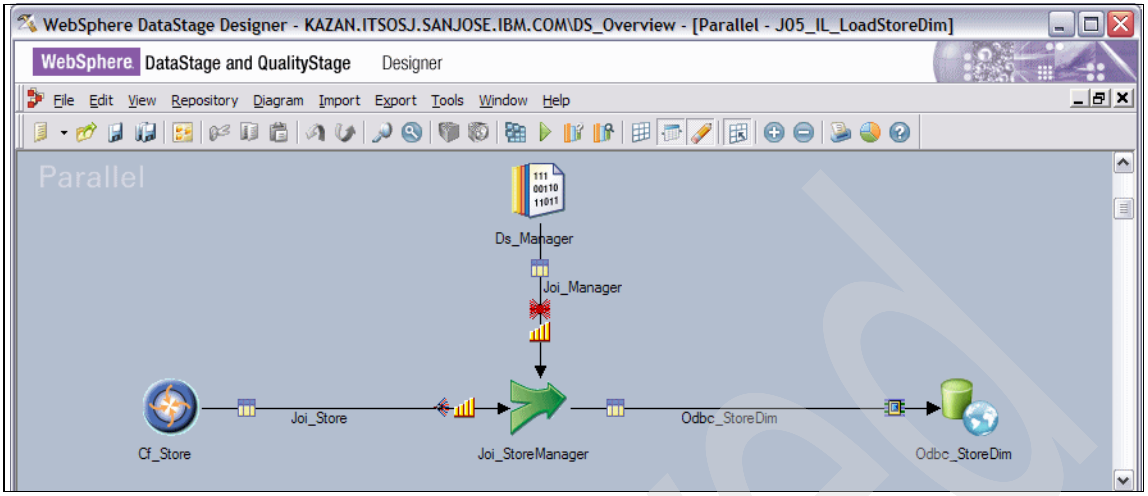


Figure 3-122 Create the J05_IL_LoadStoreDim job 1/16

Figure 3-123 Create the J05_IL_LoadStoreDim job 2/16

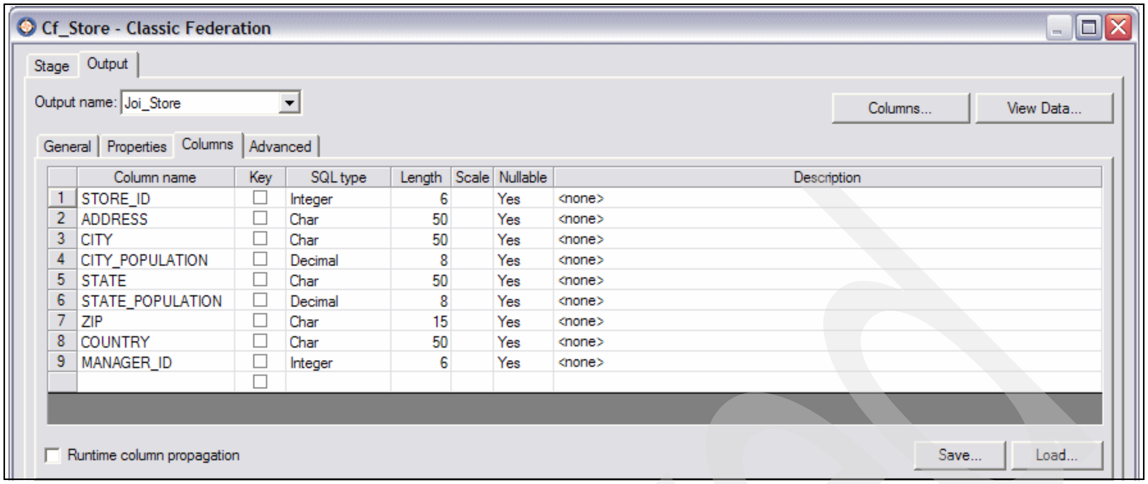


Figure 3-124 Create the J05_IL_LoadStoreDim job 3/16

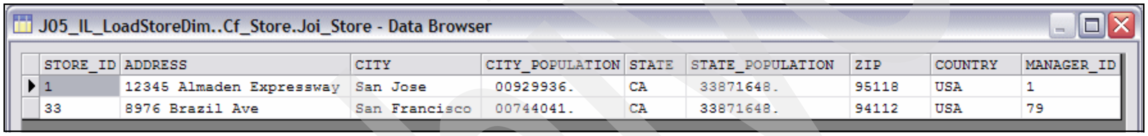


Figure 3-125 Create the J05_IL_LoadStoreDim job 4/16

Figure 3-126 Create the J05_IL_LoadStoreDim job 5/16

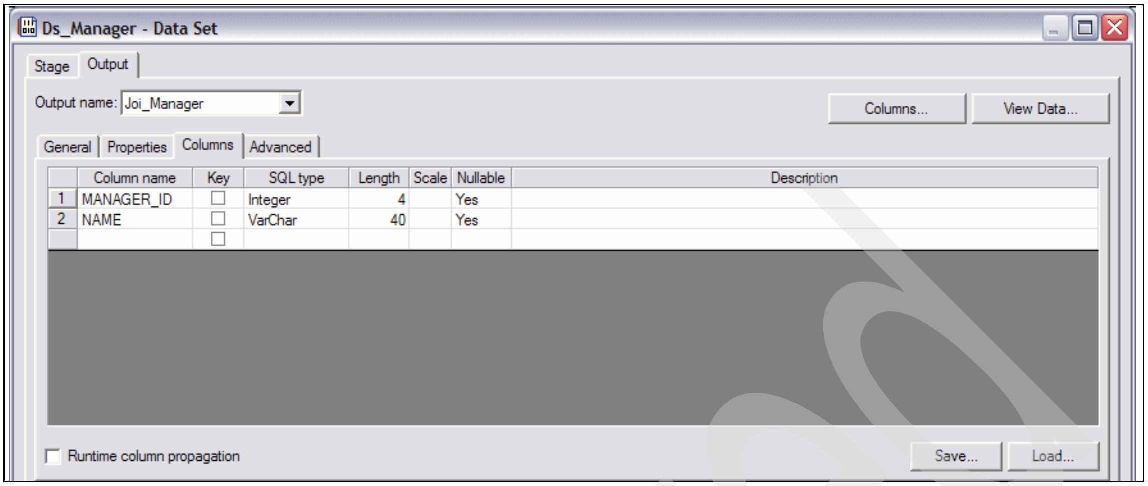


Figure 3-127 Create the J05_IL_LoadStoreDim job 6/16

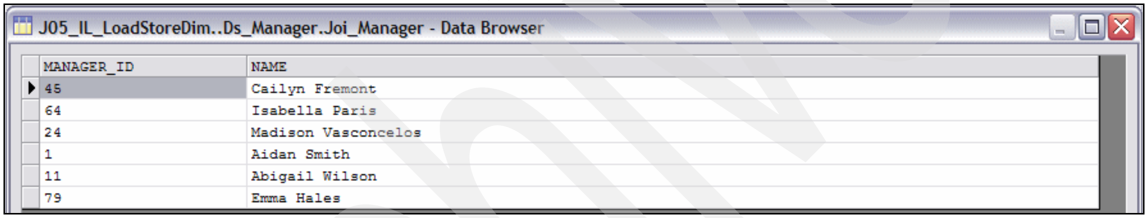


Figure 3-128 Create the J05_IL_LoadStoreDim job 7/16

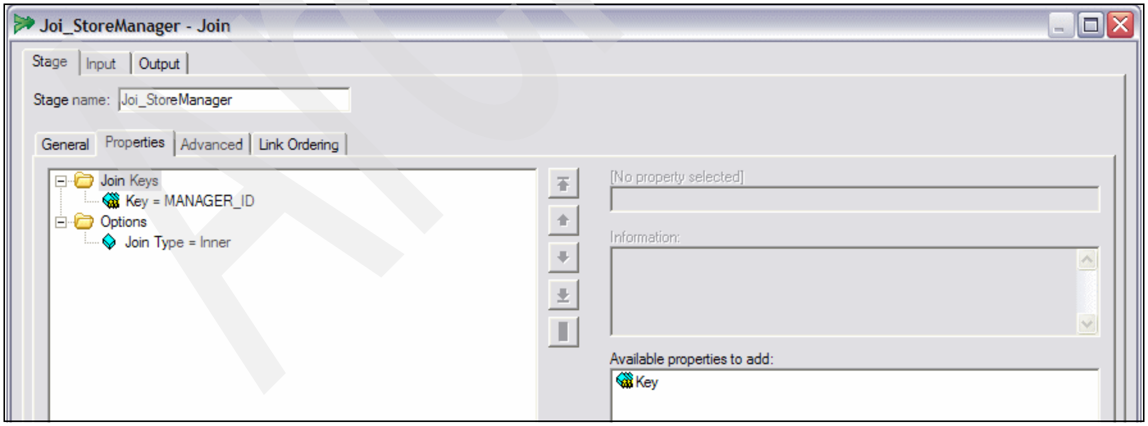


Figure 3-129 Create the J05_IL_LoadStoreDim job 8/16

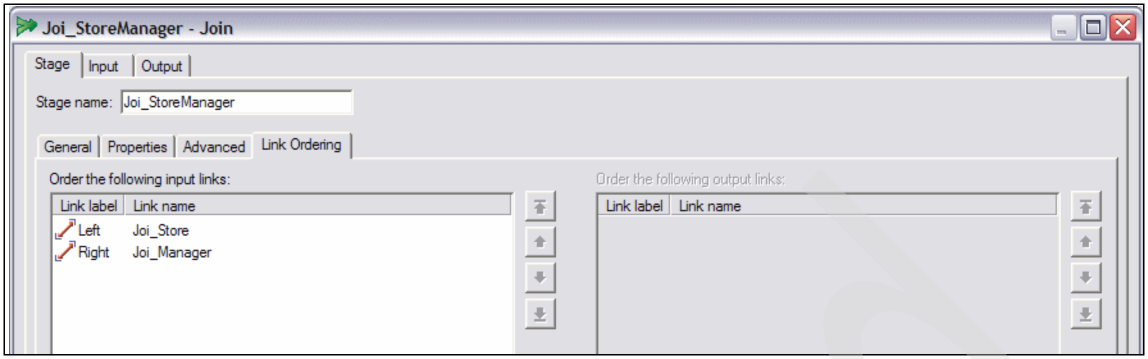


Figure 3-130   Create the J05_IL_LoadStoreDim job 9/16

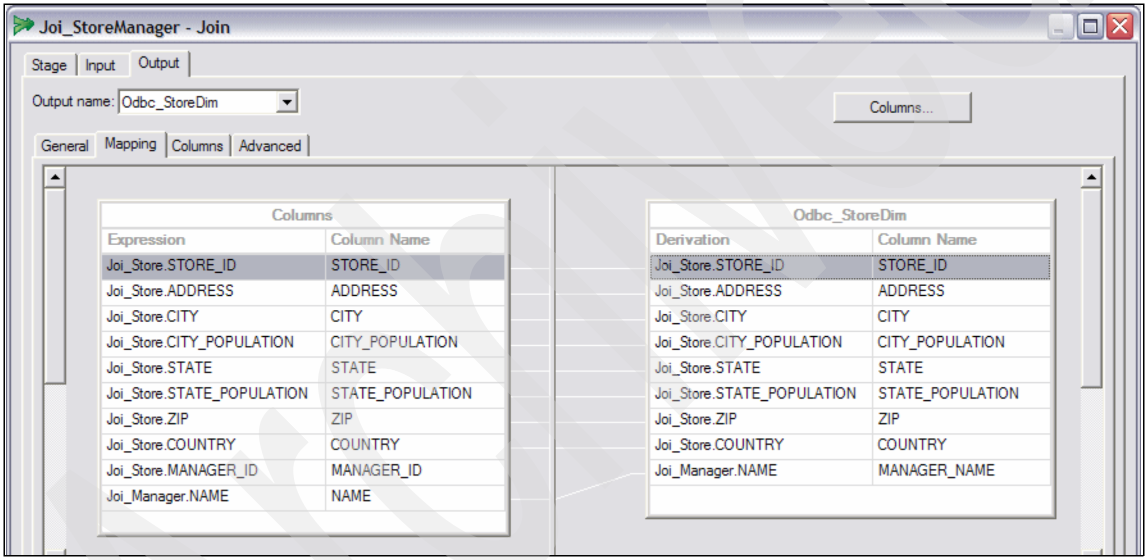


Figure 3-131   Create the J05_IL_LoadStoreDim job 10/16

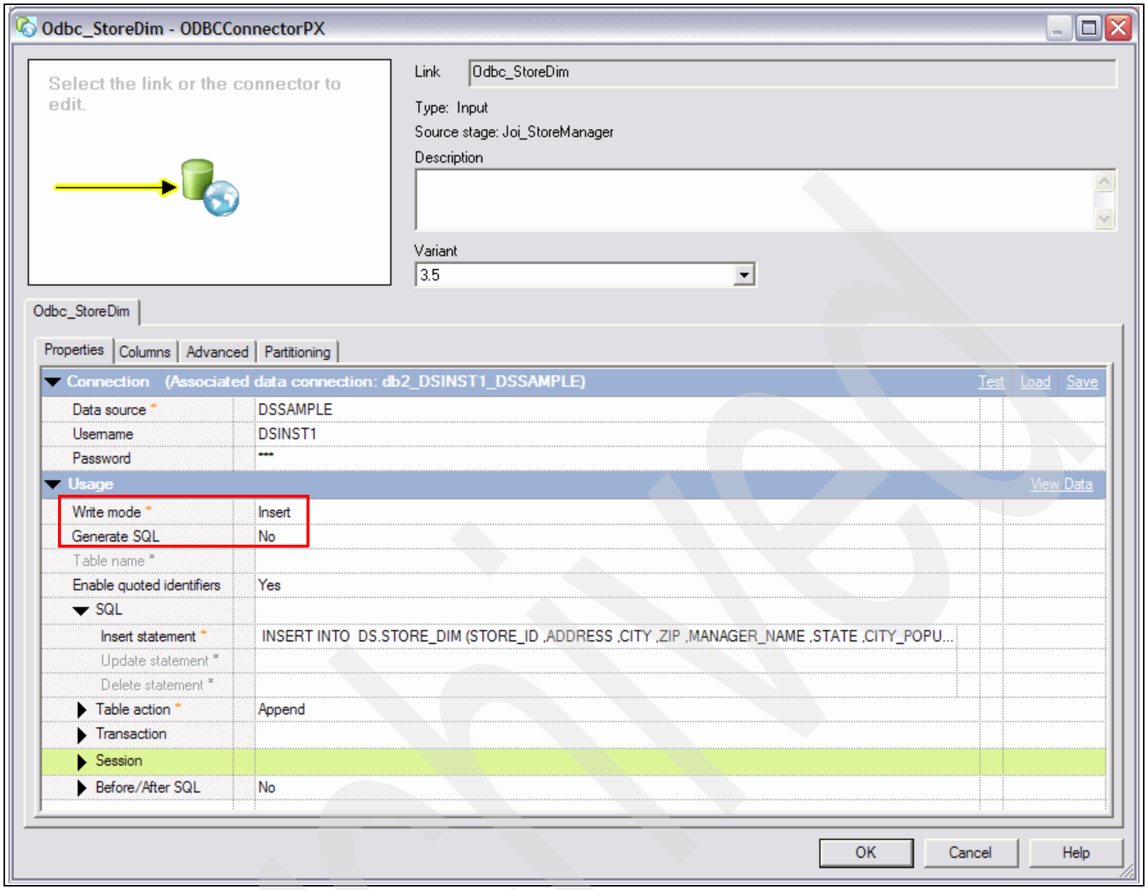


Figure 3-132 Create the J05_IL_LoadStoreDim job 11/16

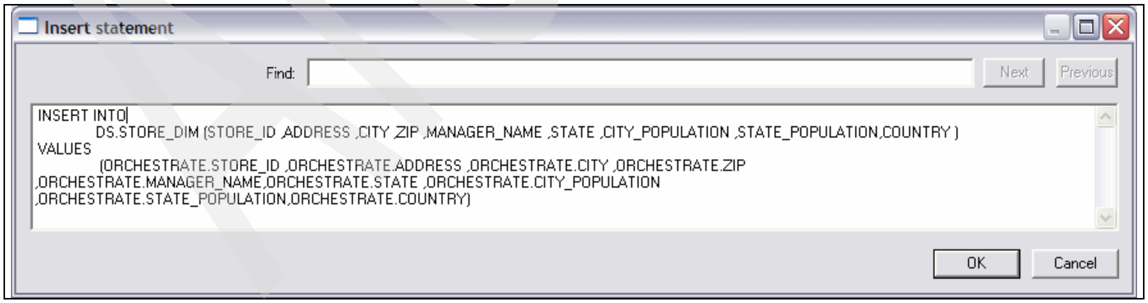


Figure 3-133 Create the J05_IL_LoadStoreDim job 12/16

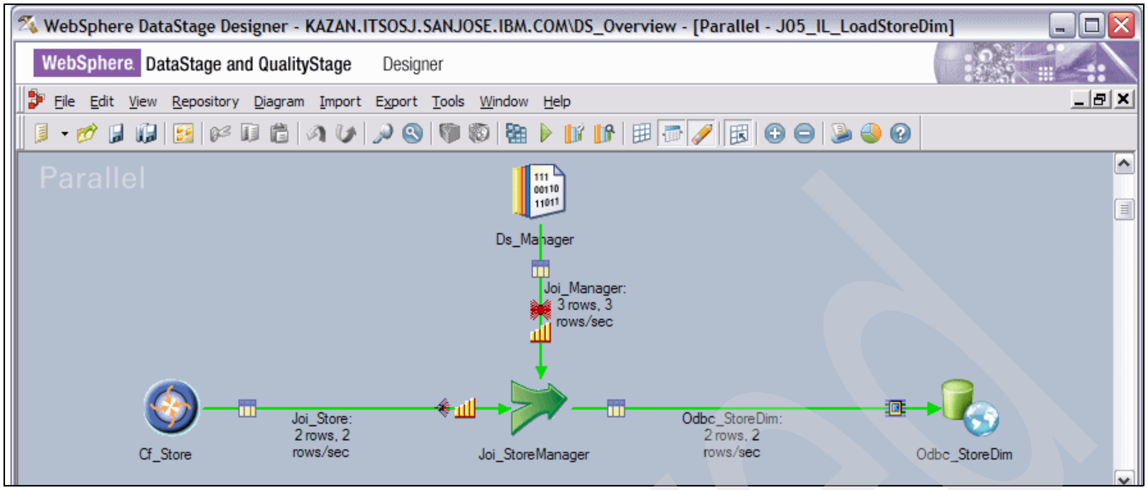


Figure 3-134 Create the J05_IL_LoadStoreDim job 13/16

J05_IL_LoadStoreDim..Cf_Store.Joi_Store - Data Browser

STORE_ID	ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	MANAGER_ID
1	12345 Almaden Expressway	San Jose	00929936.	CA	33871648.	95118	USA	1
33	8976 Brazil Ave	San Francisco	00744041.	CA	33871648.	94112	USA	79

Figure 3-135 Create the J05_IL_LoadStoreDim job 14/16

View Data

STORE_DIM_KEY	STORE_ID	ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY
742	33	8976 Brazil Ave ...	San Francisco ...	744041	CA	33871648	94112 ...	USA ...
743	1	12345 Almaden Expressway ...	San Jose ...	929936	CA	33871648	95118 ...	USA ...

Figure 3-136 Create the J05_IL_LoadStoreDim job 15/16

View Data

ION	ZIP	COUNTRY	MANAGER_NAME	CURRENT_IND	EFFECTIVE_TS	EXPIRATION_TS
	94112 ...	USA ...	Emma Hales	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
	95118 ...	USA ...	Aidan Smith	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT

Figure 3-137 Create the J05_IL_LoadStoreDim job 16/16

J06_IL_Daily_CreateCurrencyLookup_Service

As mentioned earlier, some of the sales transactions at stores occur with a credit card from a foreign country. While the sales transaction is in $US, it must be converted into the foreign currency equivalent before it can be loaded into the sales fact table.

In order to showcase the Web service capabilities of IBM InfoSphere DataStage, we specified that the daily exchange rates would be available as a Web service that would be looked up for each sales transaction involving foreign currency.

> **Note:** For performance reasons, we assumed that the exchange rates by country (ISOCODE) would be downloaded at the beginning of each day and written to a data set that would then serve as the lookup source for each sales transaction involving foreign currency.

In this step, we create a Web service using IBM Information Server on an SQL query that retrieves daily currency exchange rates stored in a CURRENCY table in the CURRENCY database. In a subsequent step, we create a shared container of the lookup of the currency exchange as described in “J07A_SharedContainerLookupCurrency” on page 273. This shared container is incorporated in the job that prepares the daily sales transactions for updating the sales fact table as described in “J07_IL_Daily_LoadSalesStore” on page 282.

Figure 3-138 shows the main steps in creating an SOA service using IBM Information Server. These are described in more detail as follows:

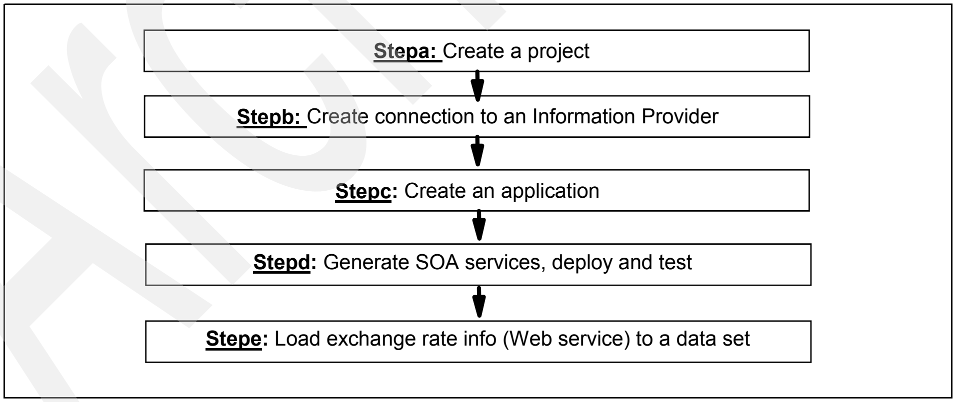


Figure 3-138 Steps in creating SOA services

Stepa: Create a project

To create an SOA service using IBM Information Server, you have to create a project and an application. While the application is a deployable unit[2], a project is a mechanism for grouping SOA services together in a logical unit. A project is a collaborative environment that you use to design applications, services, and operations.

The main steps are as follows:

1. After logging in to the IBM Information Server Console, click **New Project** as shown in Figure 3-139.
2. Provide the Name (Proj_J06) in Figure 3-140 and click **OK**.
3. Provide a Description (Proj_J06) and click **Save** to complete the definition of the project — this is not shown here.

We can now proceed to create a connection to an Information Provider as described in “Stepb: Create connection to an Information Provider” on page 229.

Figure 3-139 Create an SOA project 1/2

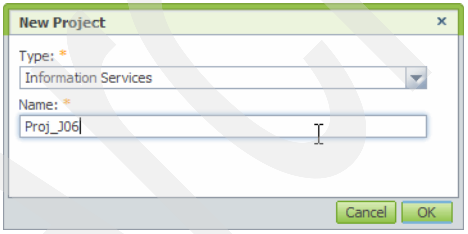


Figure 3-140 Create an SOA project 2/2

[2] An application becomes a “.ear” file that gets deployed on the WebSphere Application Server associated with IBM Information Server.

Stepb: Create connection to an Information Provider

You must create a connection to an Information Provider before being able to generate an SOA service. There are two types of Information Providers — a “DataStage and QualityStage” type for DataStage and QualityStage jobs, and a “DB2 or Federation Server” type for stored procedures and federated queries.

Figure 3-141 on page 230 through Figure 3-148 on page 236 describe the creation and testing of a “DB2 or Federation Server” type of information provider:

1. Launch the IBM Information Server console by clicking **Start** → **Programs** → **IBM Information Server** → **IBM Information Server Console** and then provide login information — this is not shown here.
2. Click **Home**, expand **Configuration**, and click **Information Services Connections** as shown in Figure 3-141 on page 230.
3. Then click **New** under Tasks column in Figure 3-142 on page 230 to create a new Information Services connection.
4. Provide details of the Connection Name (we chose Connection_J06), Information Provider Type (“DB2 or Federation Server” from the drop-down list) as shown in Figure 3-143 on page 231.
5. Then provide details of the Agent Host (KAZAN from the drop-down list which is where IBM Information Server is installed), Database Host (KAZAN where the database is installed) and its Port (50001) as shown in Figure 3-143 on page 231.
6. Click **Add** to add databases to the list of databases. Provide database details (currency) along with the User Name and Password to access it, and click **OK** as shown in Figure 3-144 on page 232.

   Note: JDBC™ Connection Properties such as isolation levels may be specified in Figure 3-143 on page 231 to override defaults used by IBM Information Server.

7. Highlight the currency database in the Database box, and click **Test** to ensure that the database has been configured correctly as shown in Figure 3-145 on page 233 and Figure 3-146 on page 234. Click **OK** in Figure 3-146 on page 234.
8. After successful validation, click **Save & Enable and Close** to complete the definition of the Information Provider as shown in Figure 3-147 on page 235 and Figure 3-148 on page 236.

We can now proceed to create an application under this project as described in “Stepc: Create an application” on page 237.

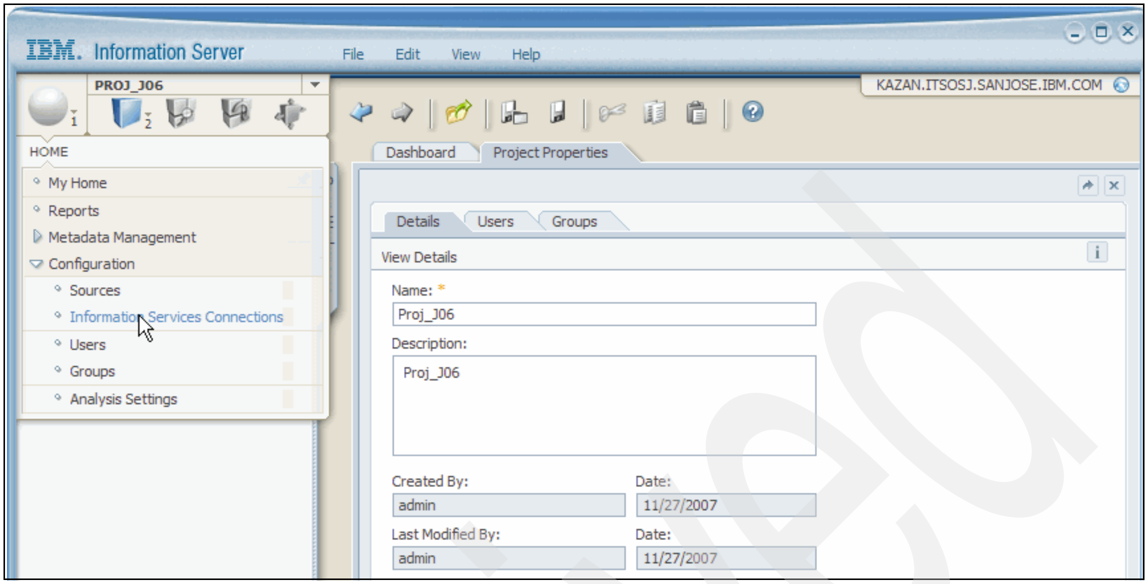


Figure 3-141   Create connection to an Information Provider 1/8

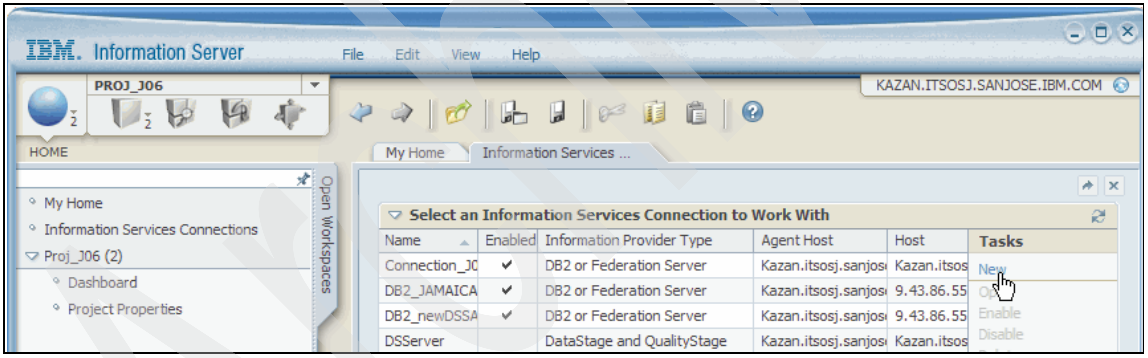


Figure 3-142   Create connection to an Information Provider 2/8

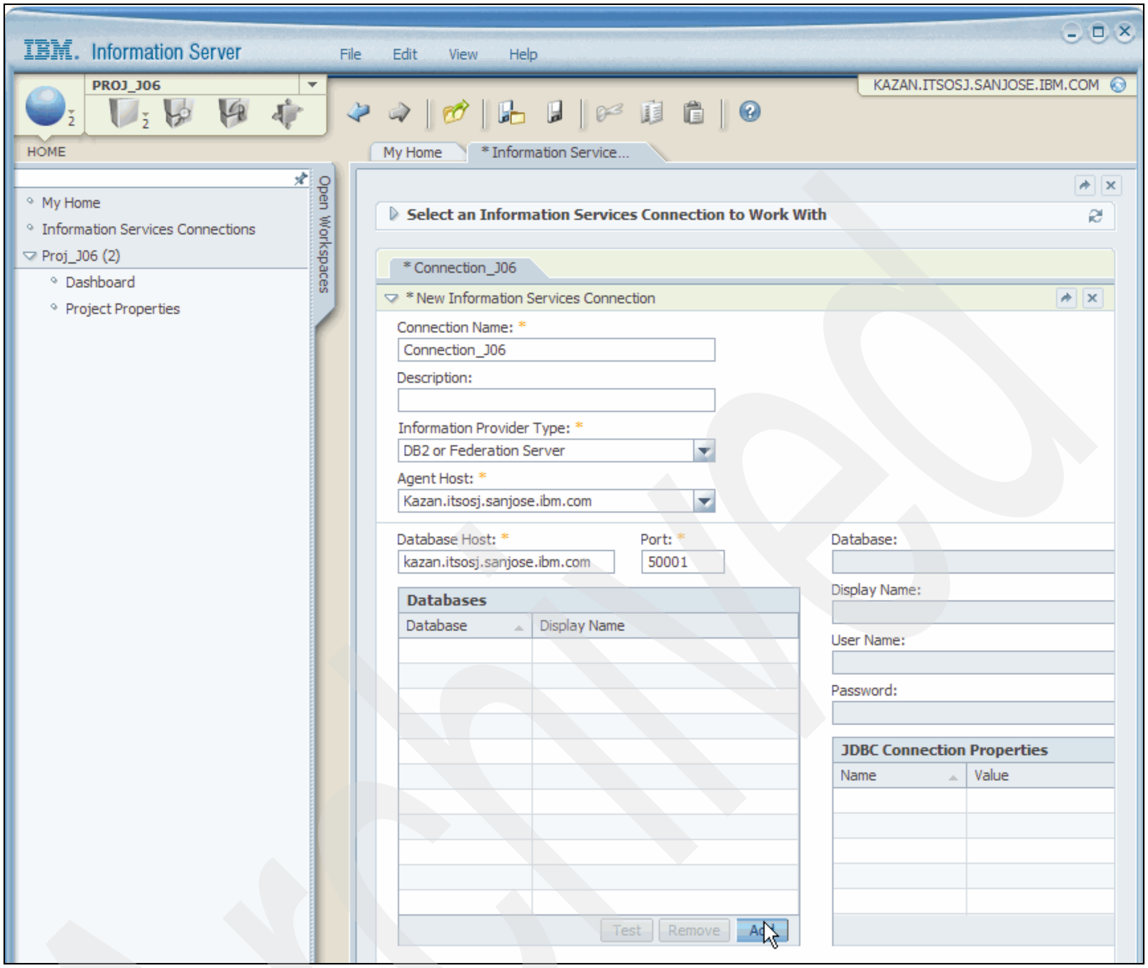


Figure 3-143 Create connection to an Information Provider 3/8

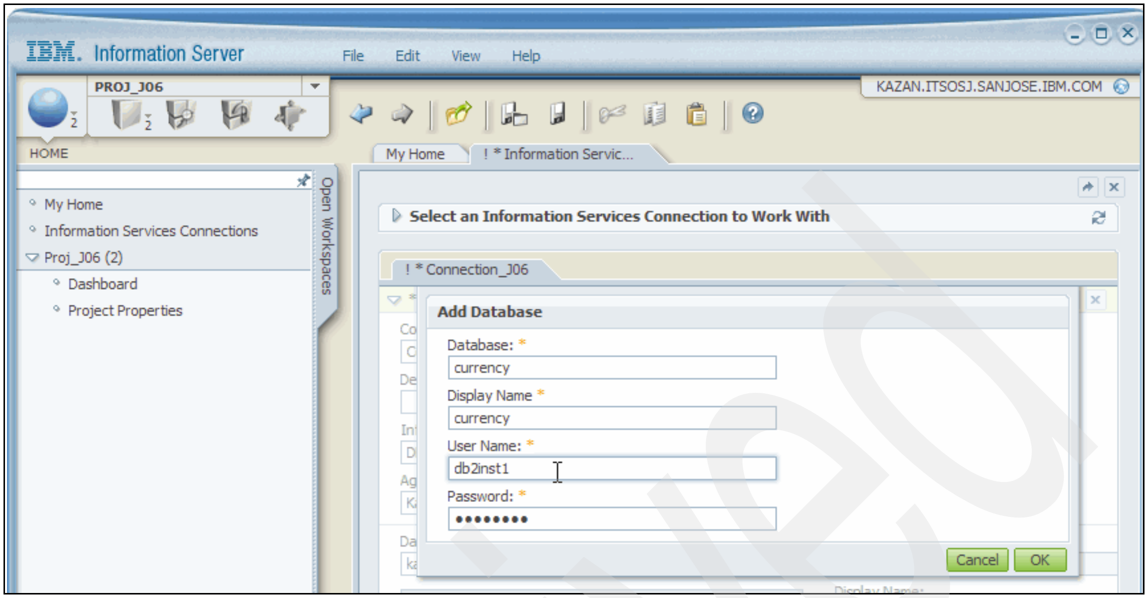


Figure 3-144 Create connection to an Information Provider 4/8

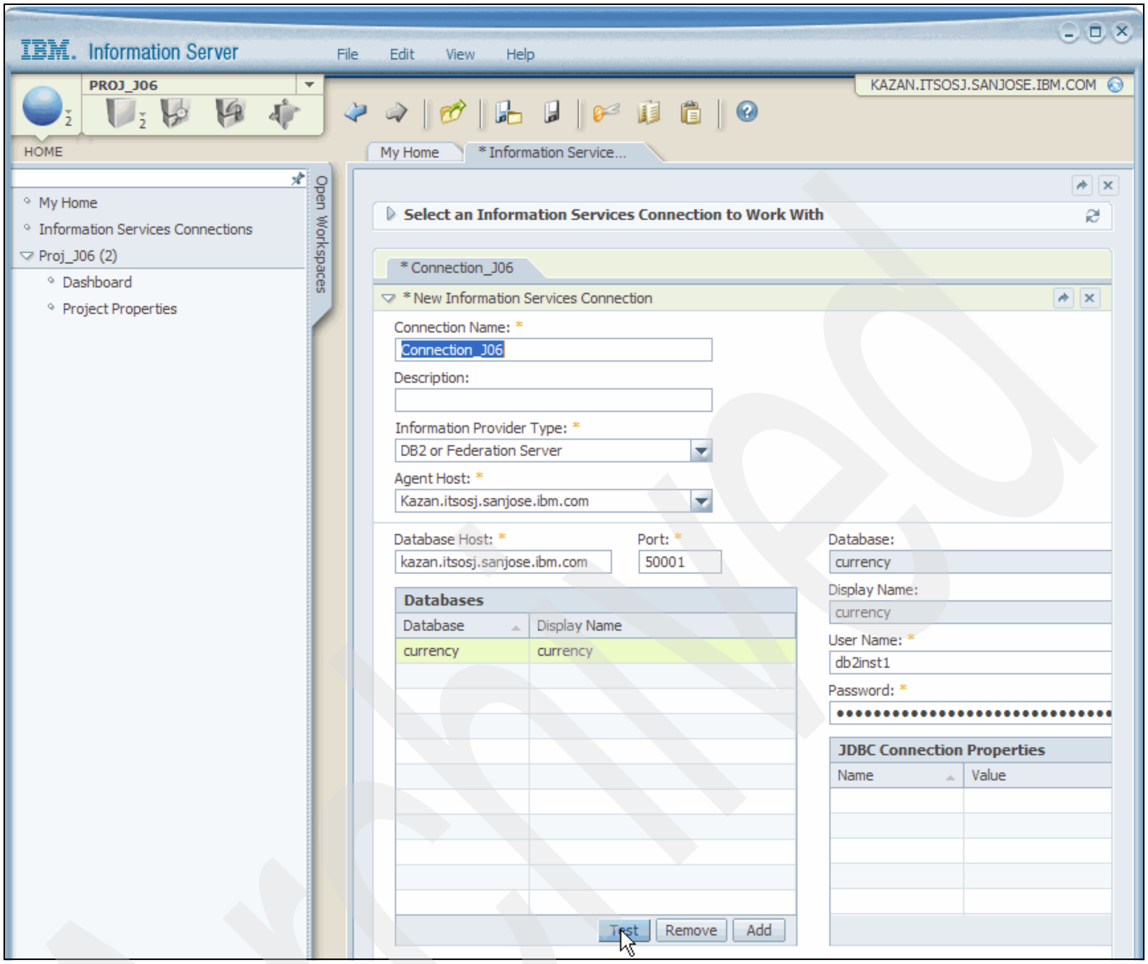


Figure 3-145 Create connection to an Information Provider 5/8

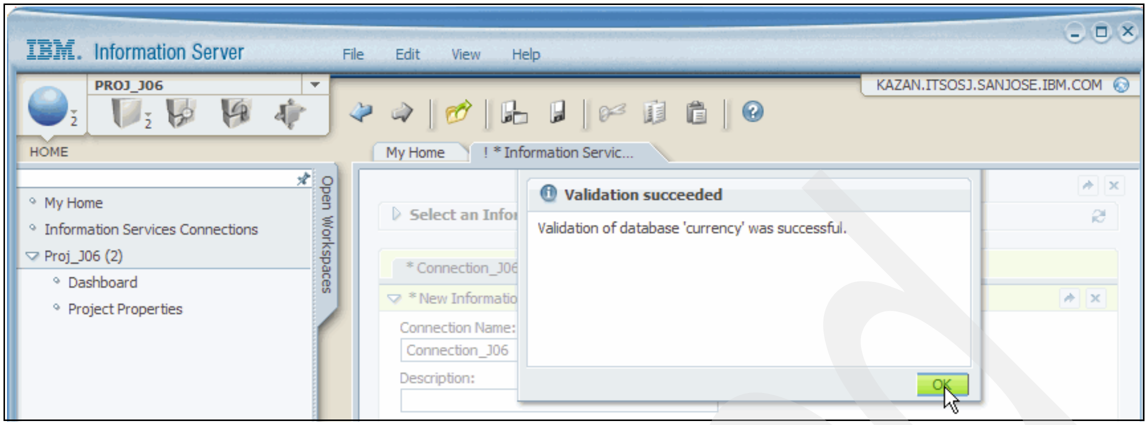


Figure 3-146 Create connection to an Information Provider 6/8

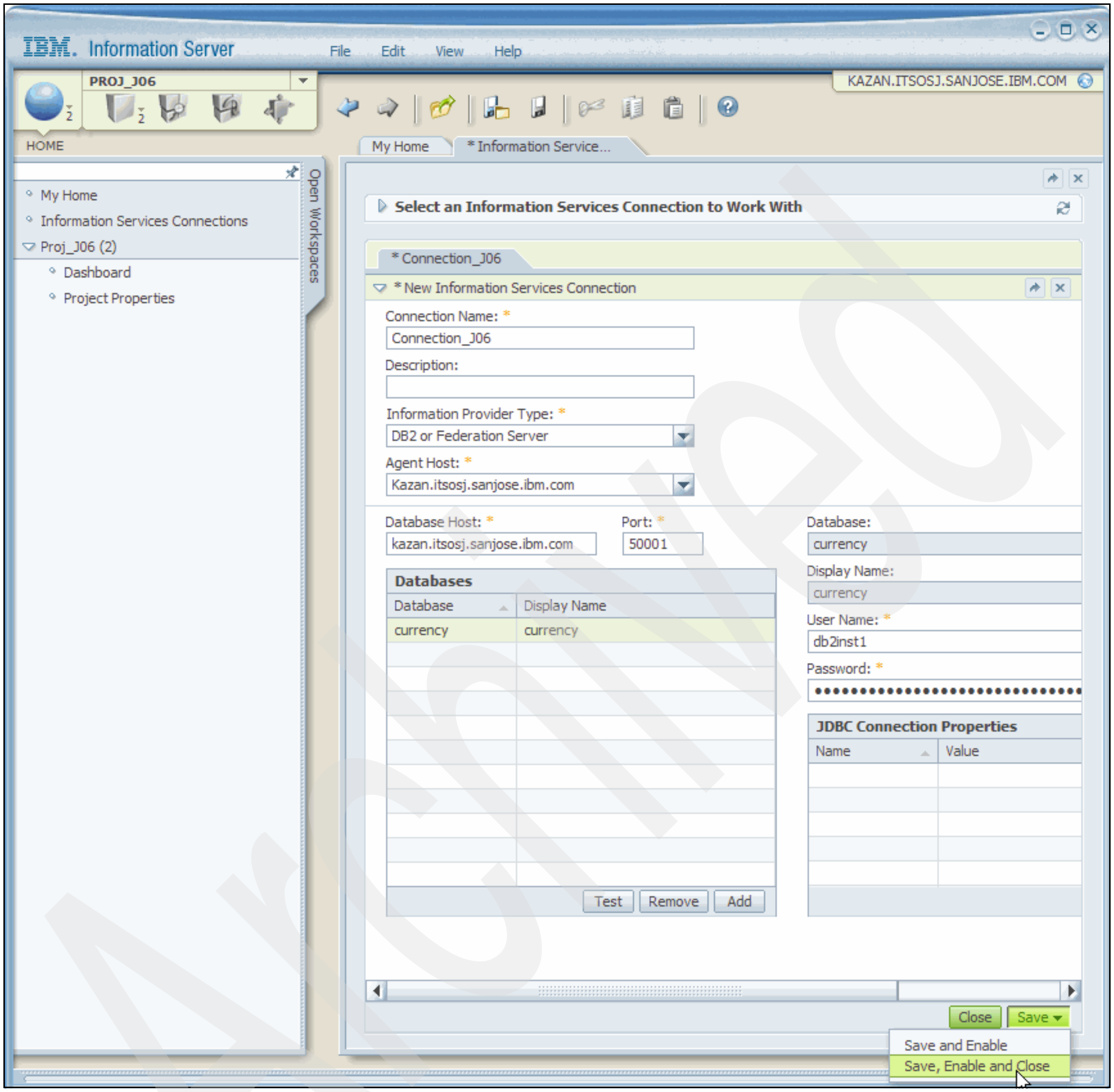


Figure 3-147   *Create connection to an Information Provider 7/8*

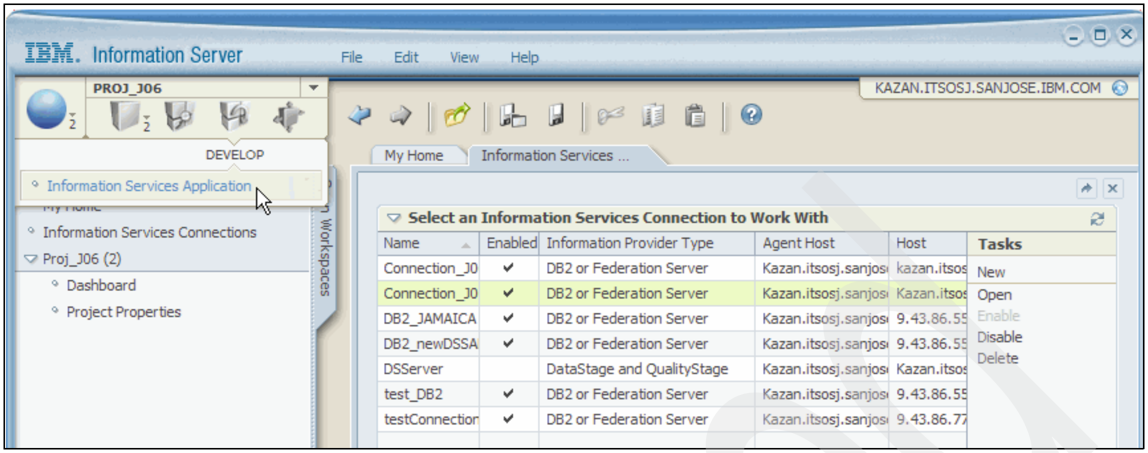


Figure 3-148 Create an application 8/8

Stepc: Create an application

The main steps in creating an application are as follows:

1. For the PROJ_J06 project, click **Information Services Application** under the Develop icon. Click **New** under the Tasks column to create a new Information Services Application to work as shown in Figure 3-149.
2. Provide details of the application such as Name (Appl_J06) and Description (WebServices Select Currency for DataStage job) and click **Save Application** to complete the definition as shown in Figure 3-150.

> **Note:** An application is a deployable unit, in that a ".ear" file is created for each application, and appears as an installed application when viewed from the WebSphere Application Server Administrative Console.

We can now proceed to generate the various SOA services, deploy them, and test them, as described in "Step 3d: Generate SOA services, deploy, and test" on page 78.

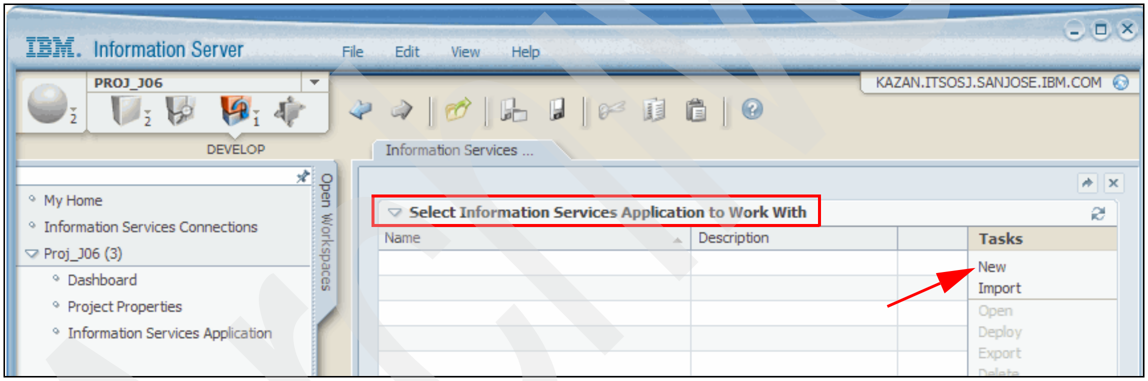


Figure 3-149 Create an application 1/2

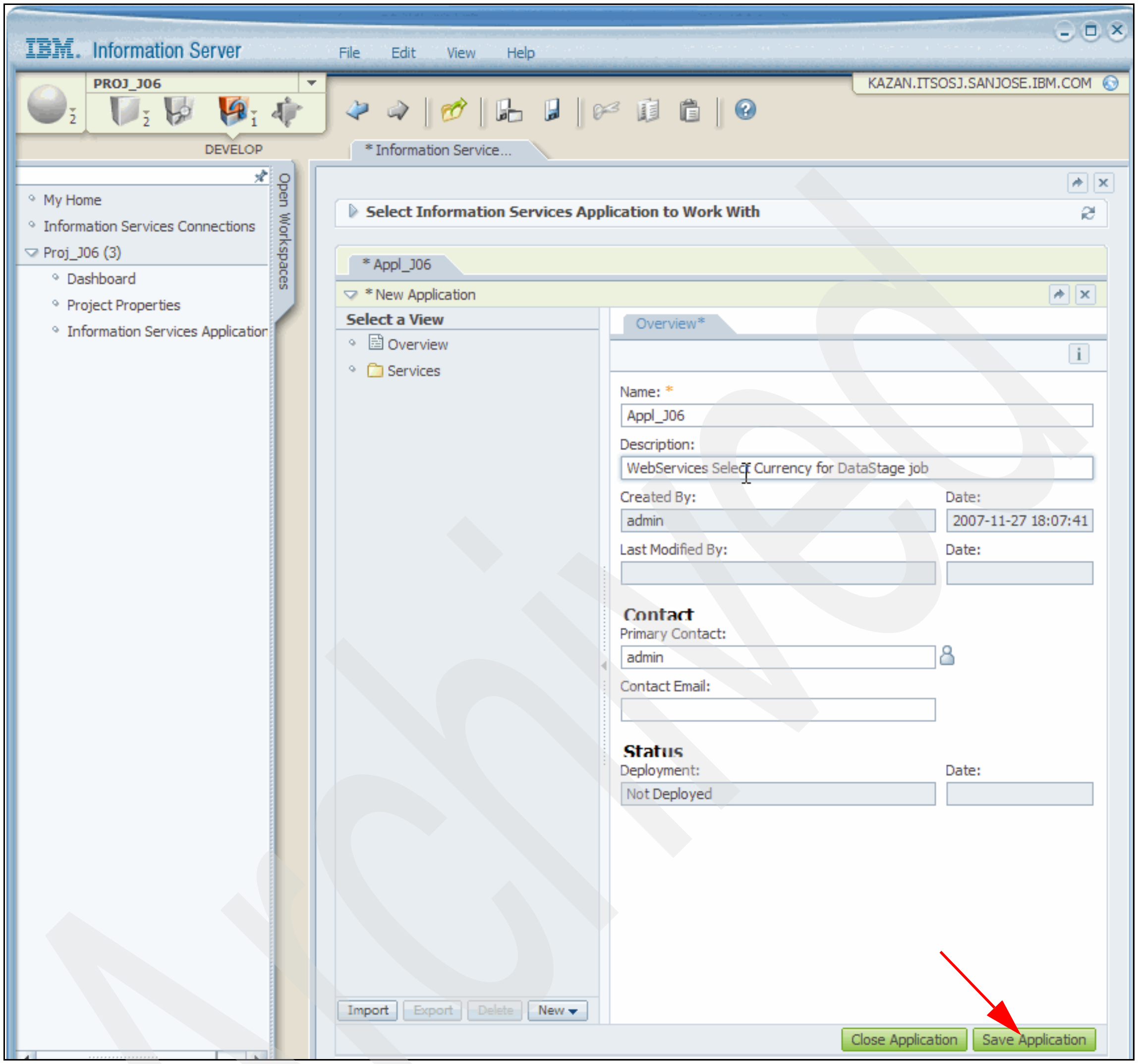


Figure 3-150 Create an application 2/2

Stepd: Generate SOA services, deploy, and test

In this section, we show the definition of service involving a federated query and its deployment and test that involves SOAP over HTTP and EJB™ bindings.

Figure 3-151 on page 241 through Figure 3-171 on page 259 describe some of the steps involved in generating an SOA service of a federated query.

The main steps are as follows:

1. After creating an application, expand Services and select **New** → **Service** for the Appl_J06 application as shown in Figure 3-151 on page 241.
2. Provide details such as the Service Name (Serv_J06) and optionally the Description and click **Save Application** as shown in Figure 3-152 on page 242.
3. Expand Operations, double-click **newOperation1** (in Figure 3-153 on page 243) and then modify the Name field to operJ06 and optional Description field as shown in Figure 3-154 on page 244. Click **Select** to select an information provider.
4. Select **DB2 or Federation Server** as the (Information Provider) Type from the drop-down list, **SQL Statement** from the Subtype drop-down list, and **Create SQL Statement** from the Action drop-down list. Select currency in the Select a Database column. Then type the SQL statement as shown and click **OK** in Figure 3-155 on page 245.
5. For the operJ06 operation, the **Inputs** (none in this case, since this service just returns the daily exchange rates for all the countries) are shown in Figure 3-156 on page 246, **Outputs** (which include the country_iso_code, date, and rate_from_usd columns) are shown in Figure 3-157 on page 247, the **SQL Statement** is shown in Figure 3-158 on page 248, the **Provider Properties** are shown in Figure 3-159 on page 249, and the **Default Settings** are shown in Figure 3-160 on page 250. Click **Save Application** in Figure 3-160 on page 250 to save the changes.
6. Next specify the bindings for the service. Double-click **Bindings** for the Serv_J06 as shown in Figure 3-161 on page 251. Then click **Attach Bindings** and select **SOAP over HTTP**.

   Note: Multiple bindings can be defined depending upon the environments (J2EE and/or .NET) in which client applications consuming these services operate. We chose only the SOAP over HTTP binding here.

7. Click **Save Application** to save these changes as shown in Figure 3-162 on page 252.

1. The next step is to deploy the federated query service. from the To deploy the saved Appl_J06 application, select it in the Select Information Services Application to Work With under the **Information Services Application** tab and select **Deploy** as shown in Figure 3-163 on page 253.
2. Confirm the services, bindings and operations to include in this deployment by checking the appropriate boxes, and click **Deploy** as shown in Figure 3-164 on page 254.
3. When deployment is completed, the status of the Appl_J06 has a Deployment Status of Deployed as shown in Figure 3-165 on page 255.
4. All deployed services can be viewed from IBM Information Server console follows:
   a. For the PROJ_J06 project, click **OPERATE** and then **Deployed Information Services Applications** as shown in Figure 3-166 on page 255 through
   b. Expand Appl_J06 and select operJ06 as shown in Figure 3-167 on page 256 to see the overview of this operation's features.
   c. Select the name of the service Serv_J06 and click **View Service in Catalog** as shown in Figure 3-168 on page 257 to view details of this service as seen in Figure 3-169 on page 258.
   d. Select **Bindings** in Figure 3-169 on page 258 to view the SOAP over HTTP binding that was chosen for this service. The WSDL[3] document for this service can be viewed by clicking **Open WSDL Document** in Figure 3-169 on page 258**.** Its contents can be seen in Figure 3-171 on page 259.
5. This service has to be tested before making it available to the user community. A number of freeware products are available to test a SOAP over HTTP service. We have not included them here. Refer to the Redbooks publication, *SOA Solutions Using IBM Information Server*, SG24-7402 for full details on generating, deploying, and testing SOA services using IBM Information Server.

[3] A Web Services Description Language (WSDL) is an XML format document that is used to exchange interface information between a Web service producer and Web service consumers. A WSDL description allows a consumer (client application) to utilize a Web service's capabilities without having to know the technologies used to implement the Web service.

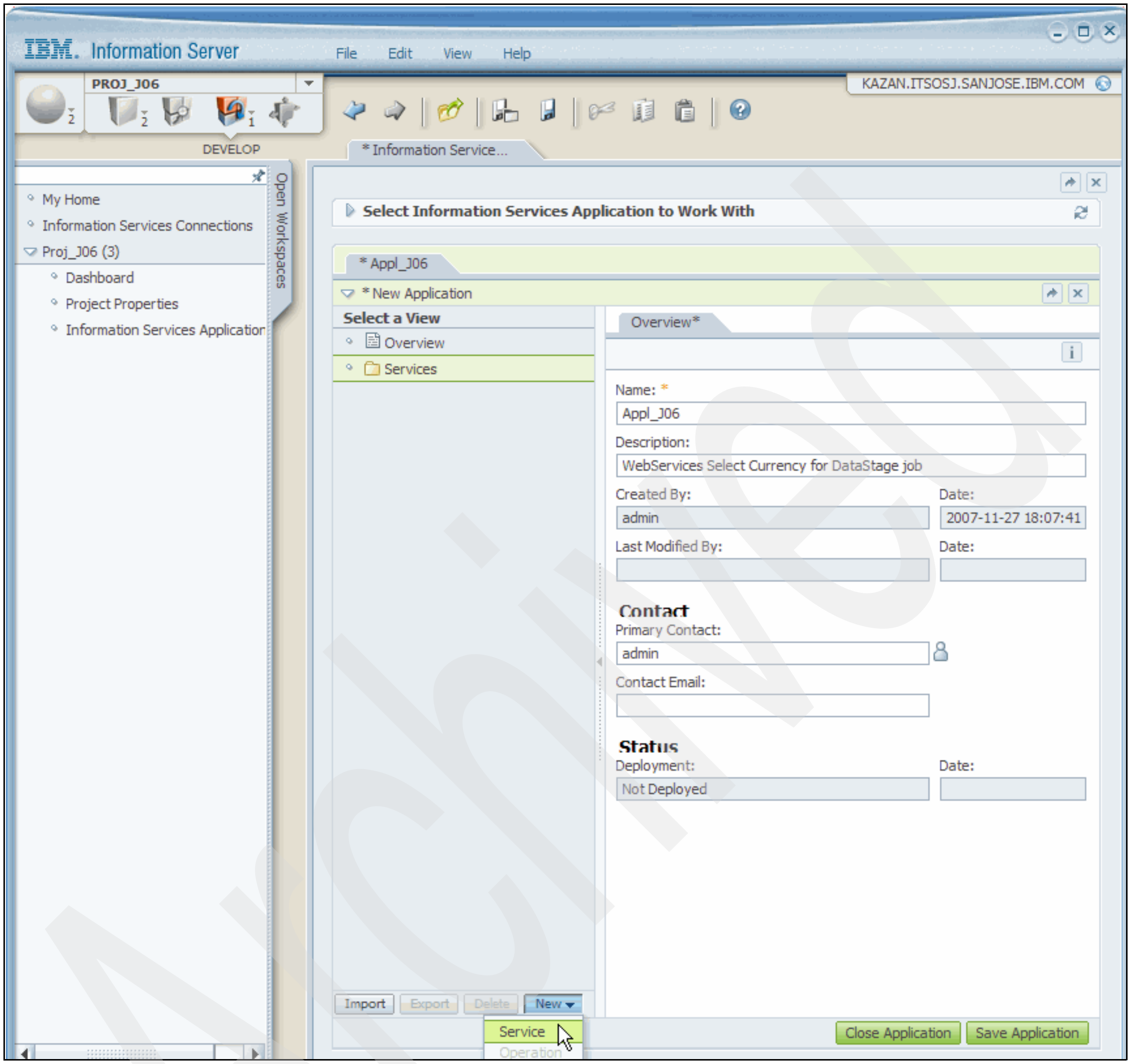


Figure 3-151   Generate SOA services, deploy, and test 1/21

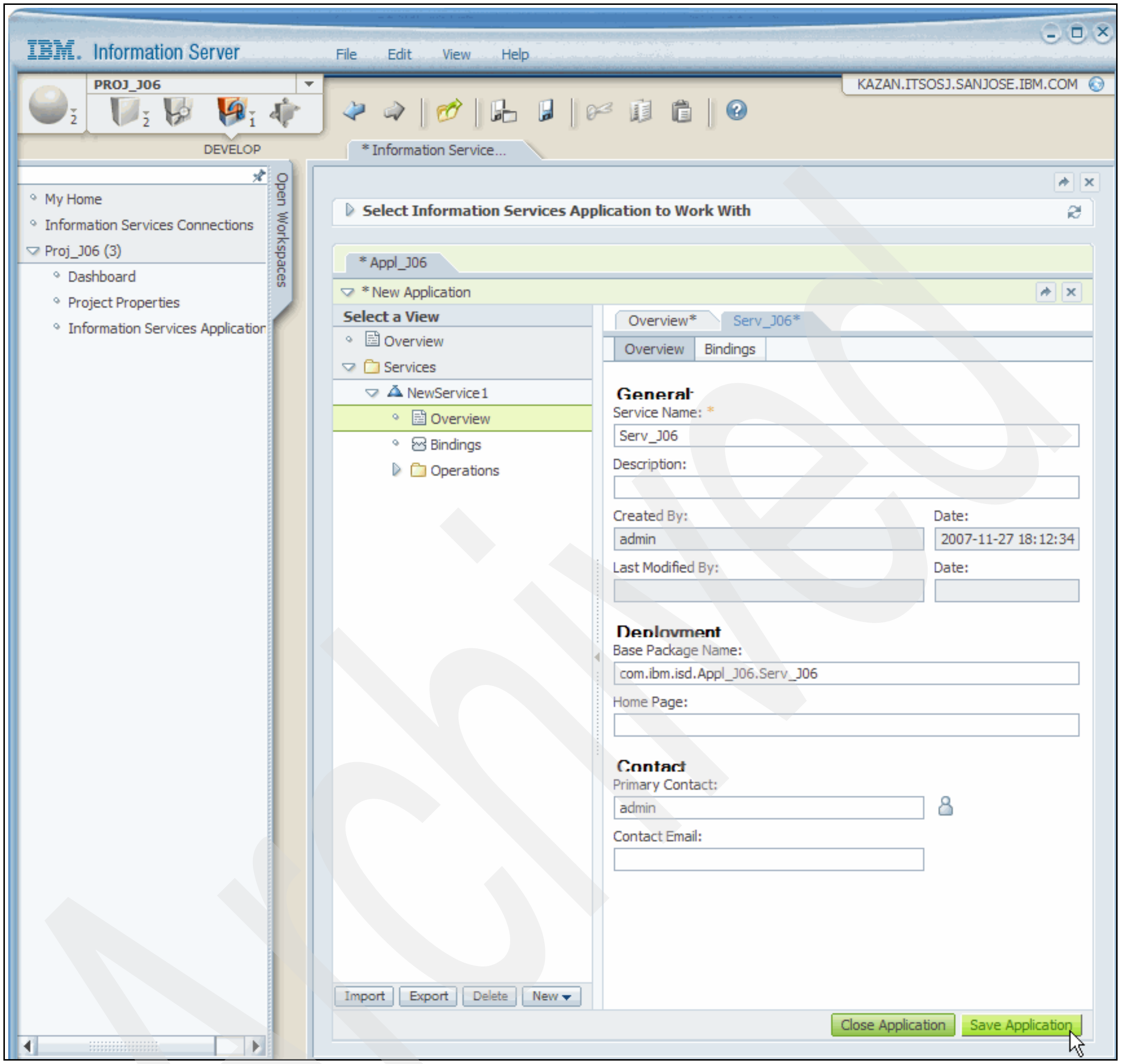


Figure 3-152 Generate SOA services, deploy, and test 2/21

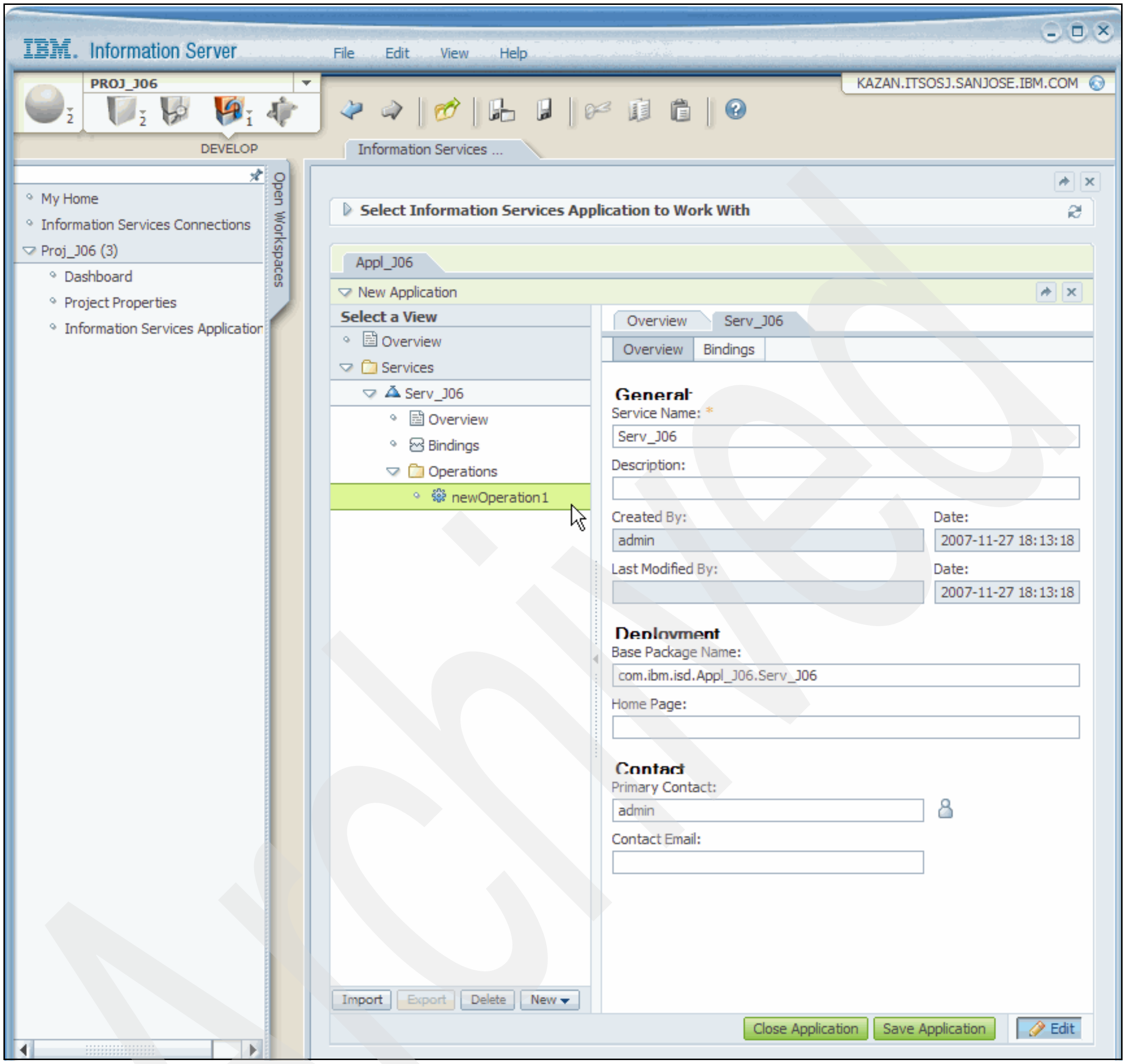


Figure 3-153 Generate SOA services, deploy, and test 3/21

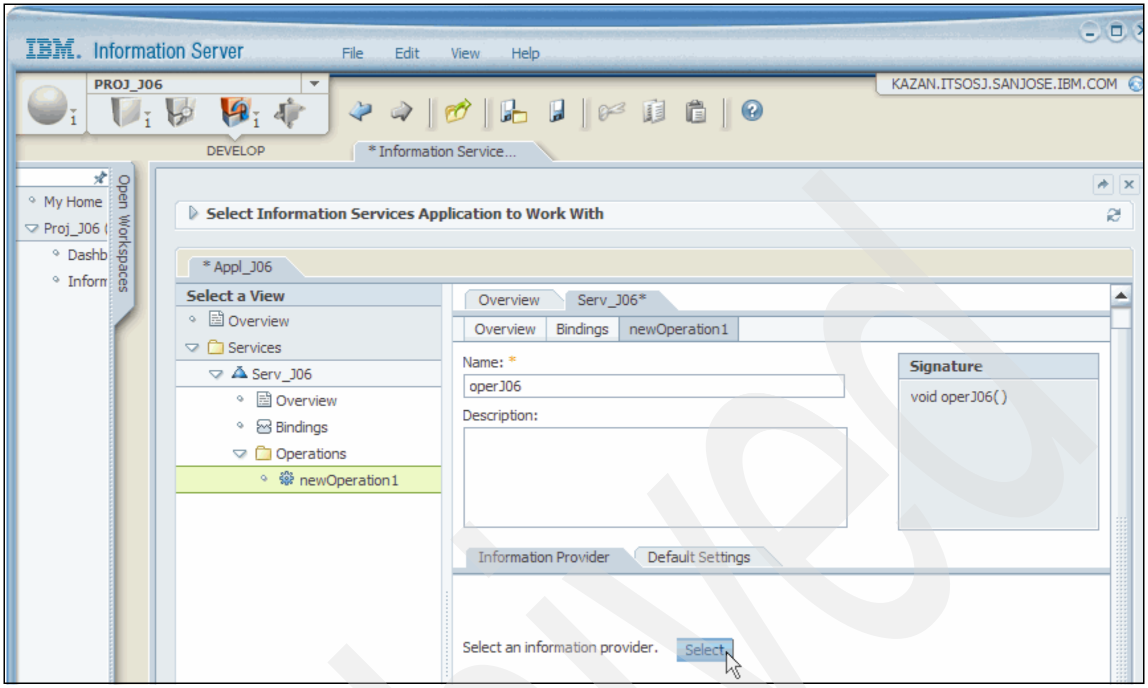


Figure 3-154 Generate SOA services, deploy, and test 4/21

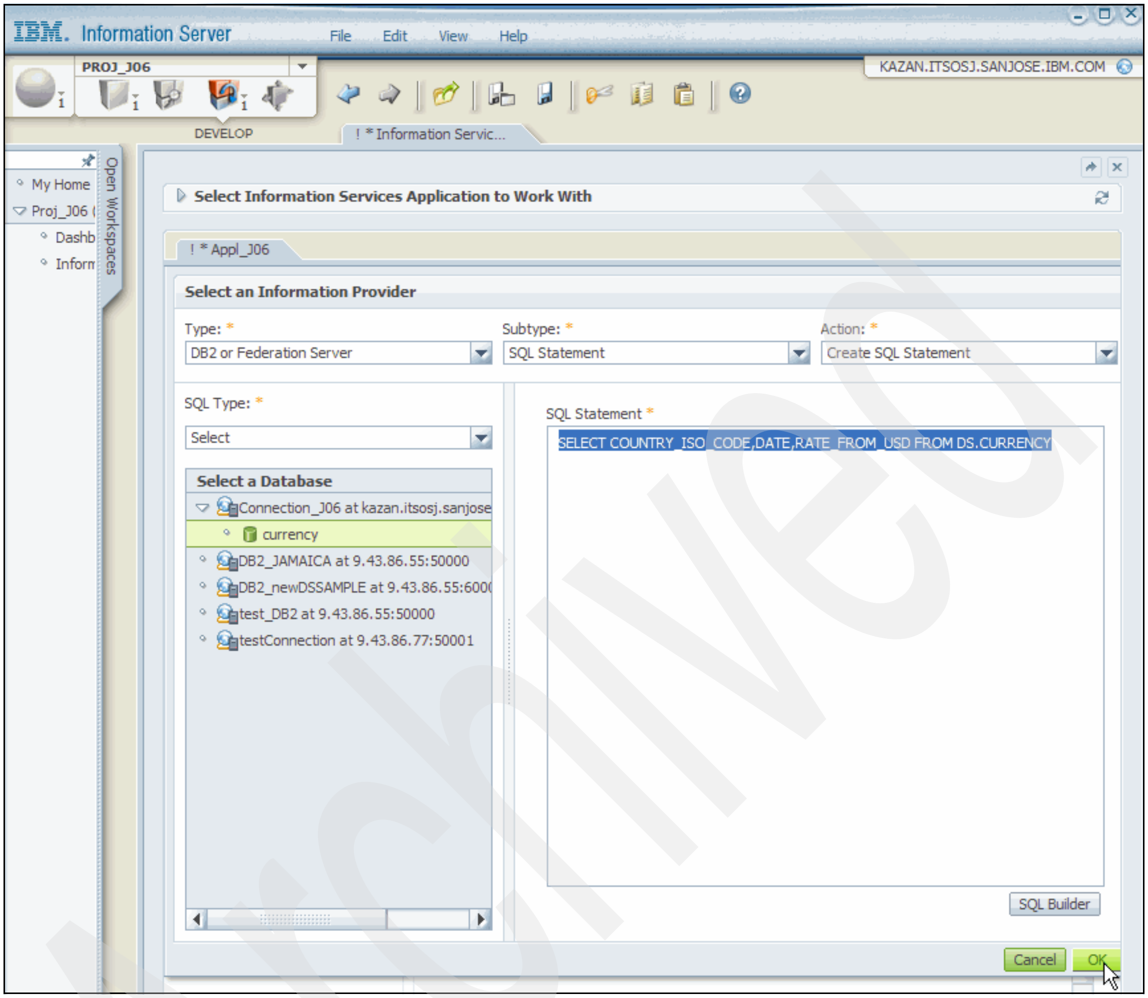


Figure 3-155 Generate SOA services, deploy, and test 5/21

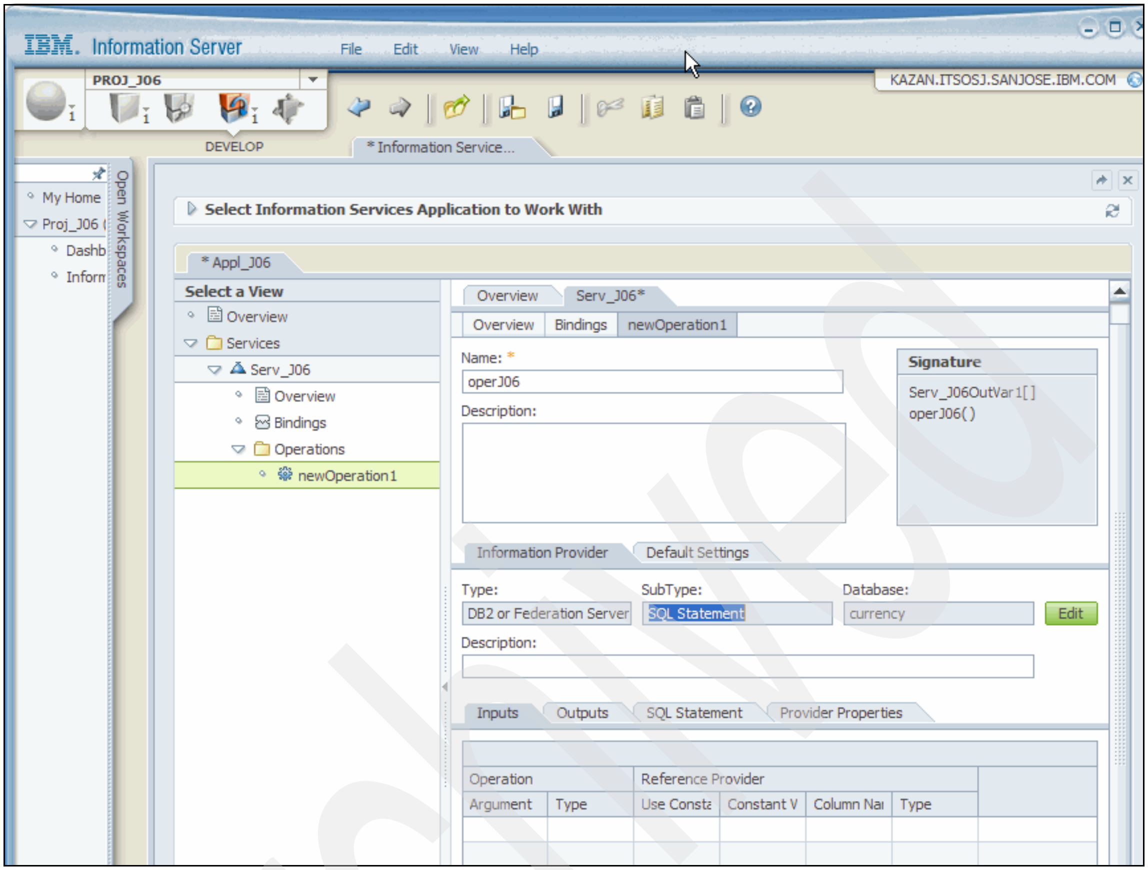


Figure 3-156 Generate SOA services, deploy, and test 6/21

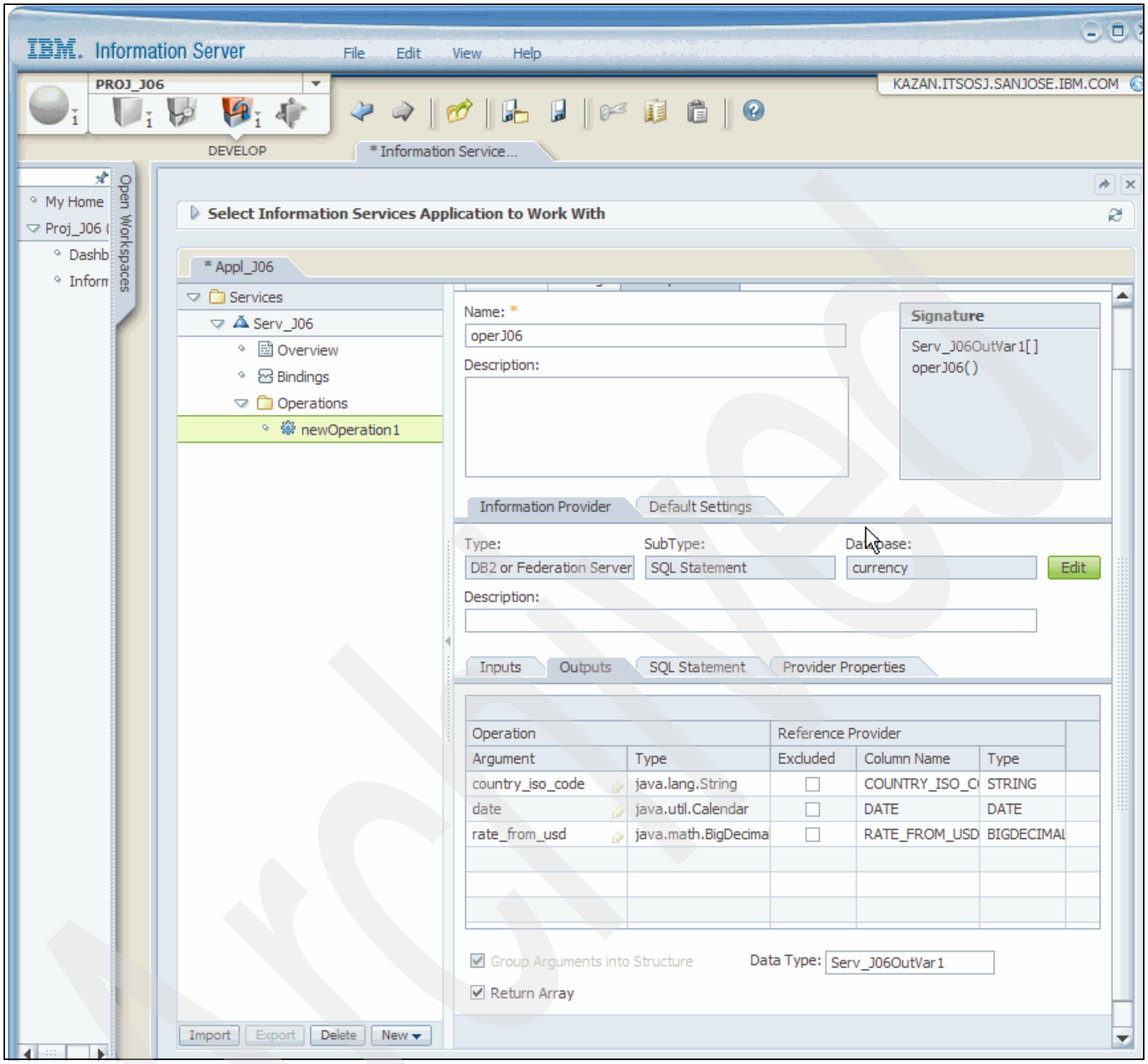


Figure 3-157 Generate SOA services, deploy, and test 7/21

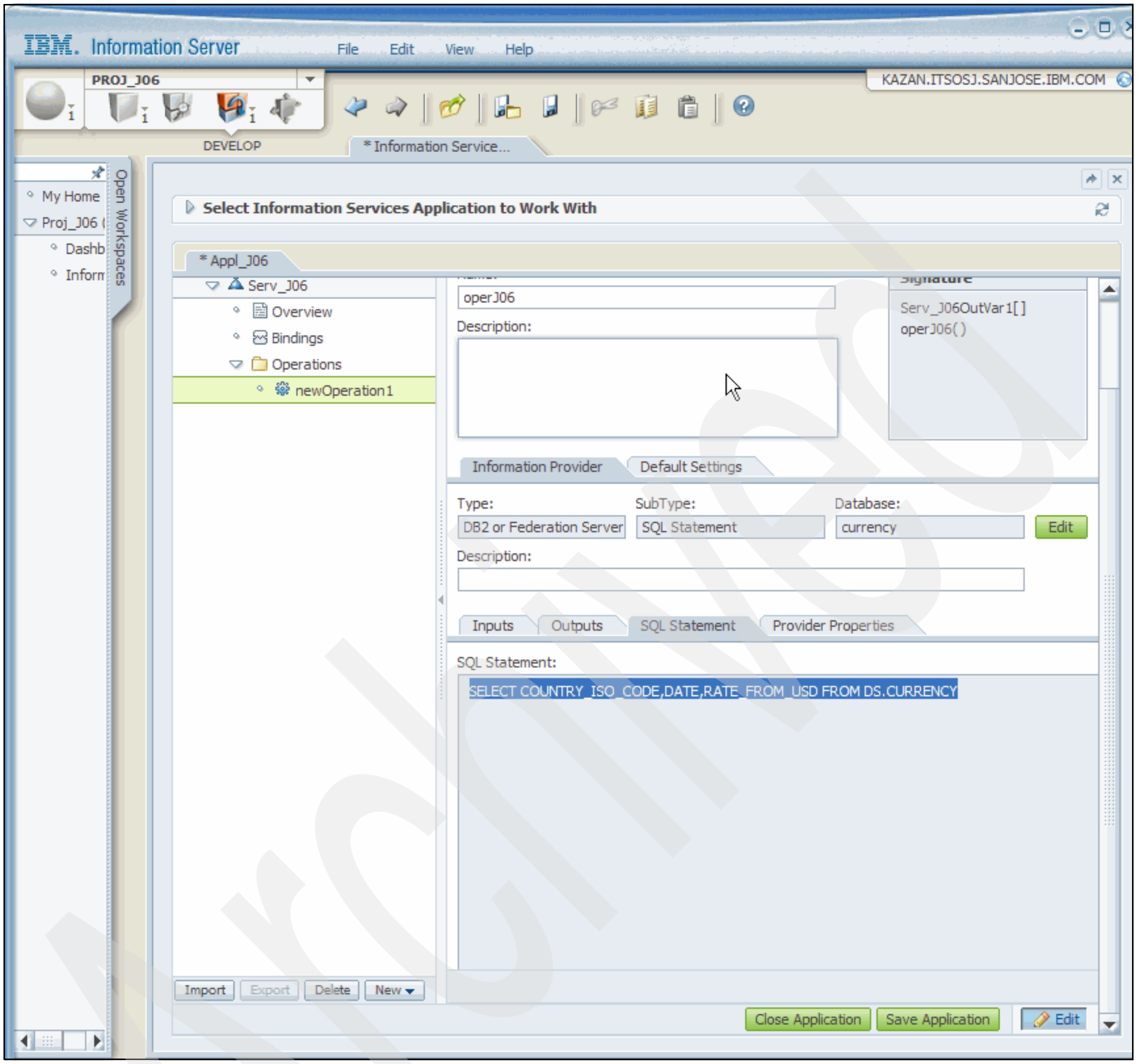


Figure 3-158 Generate SOA services, deploy, and test 8/21

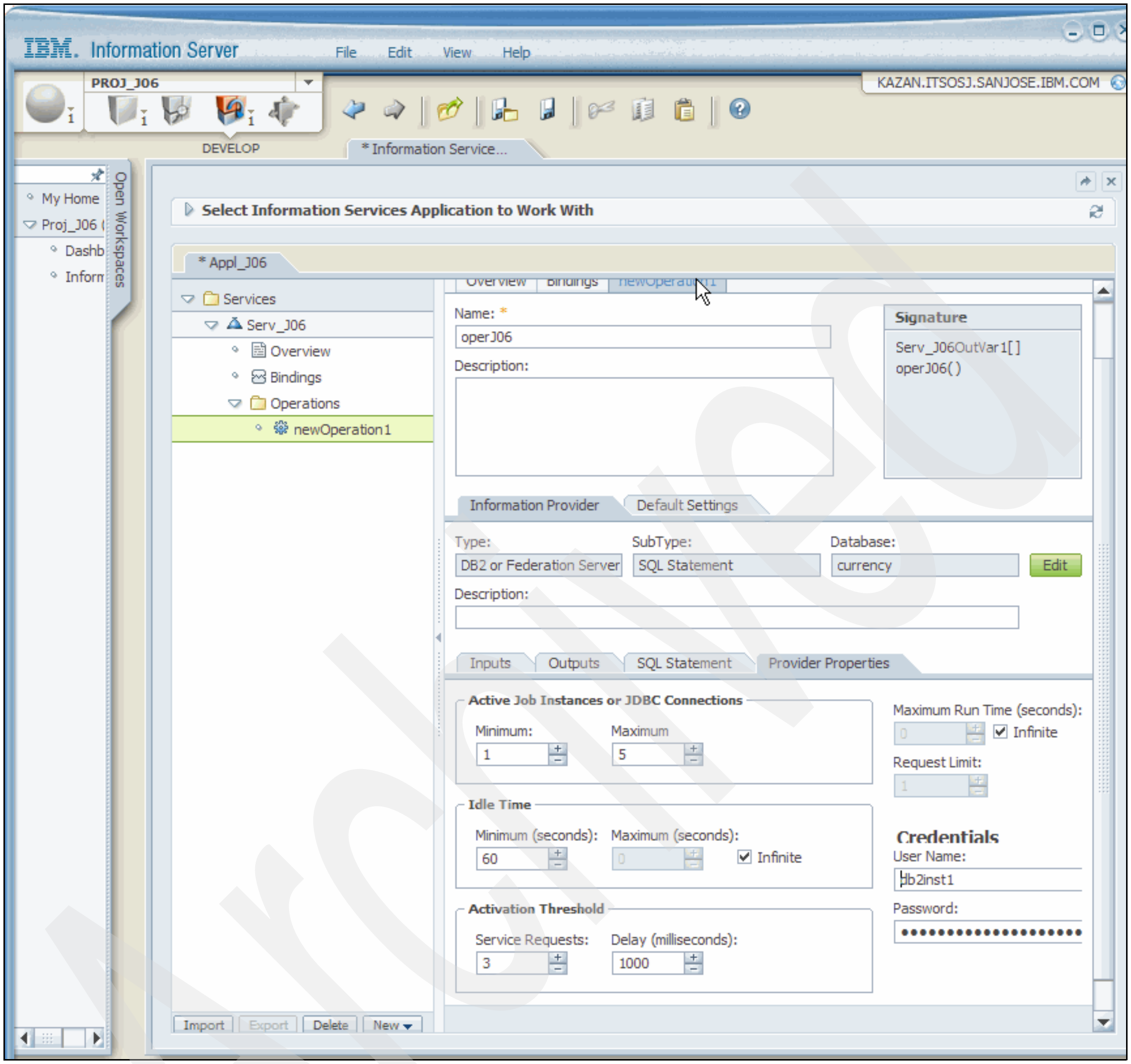


Figure 3-159 Generate SOA services, deploy, and test 9/21

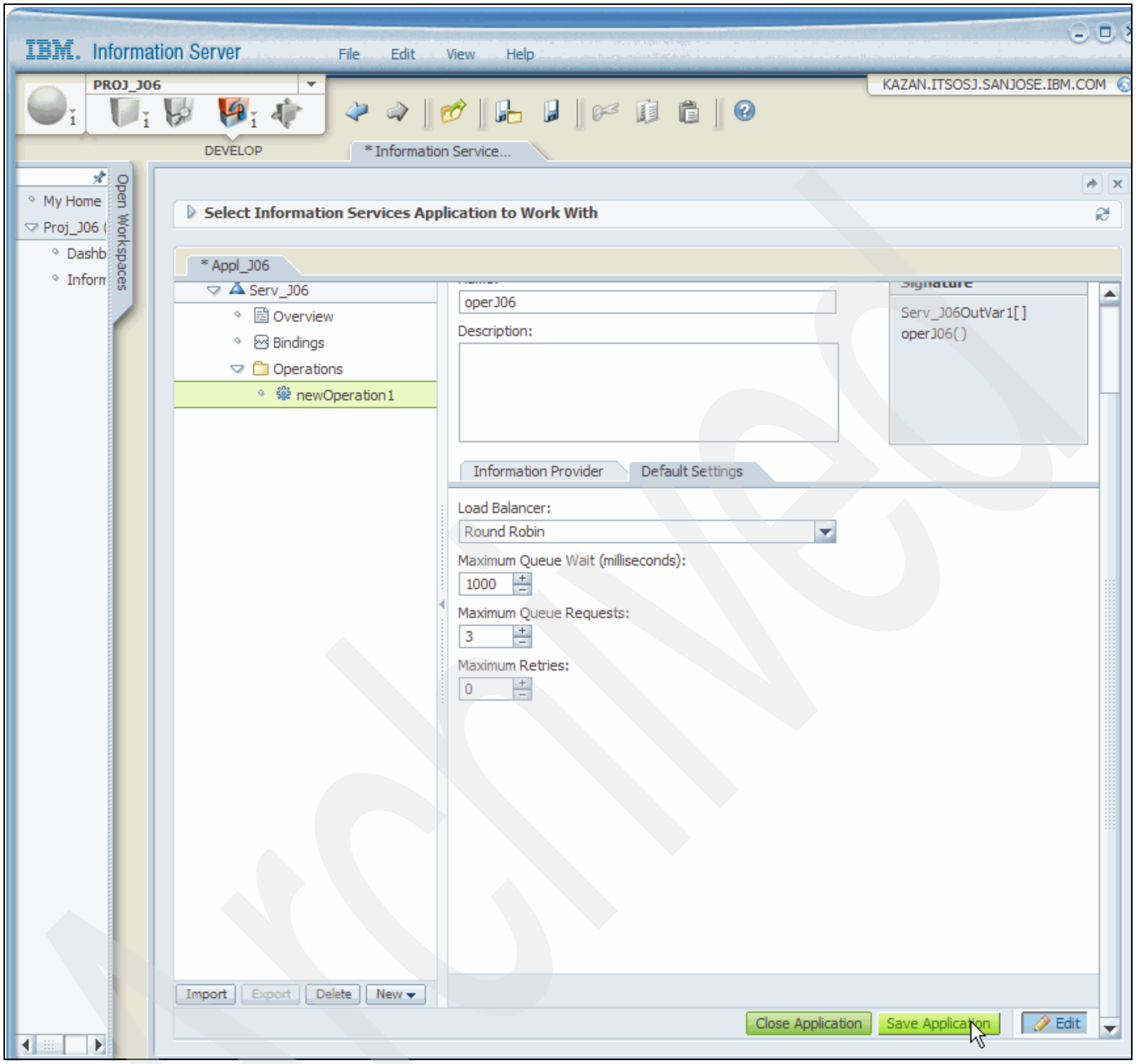


Figure 3-160 Generate SOA services, deploy, and test 10/21

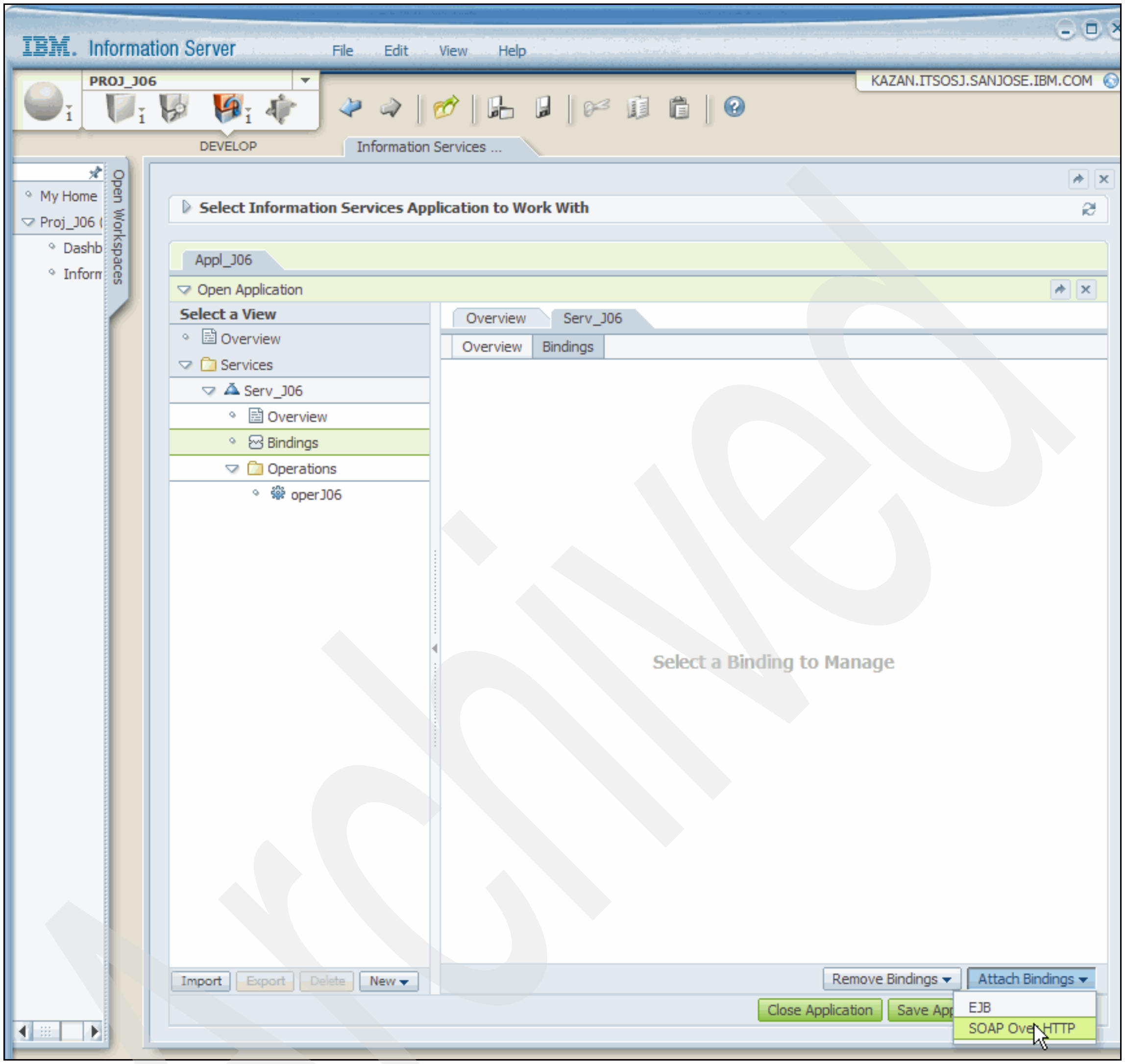


Figure 3-161 Generate SOA services, deploy, and test 11/21

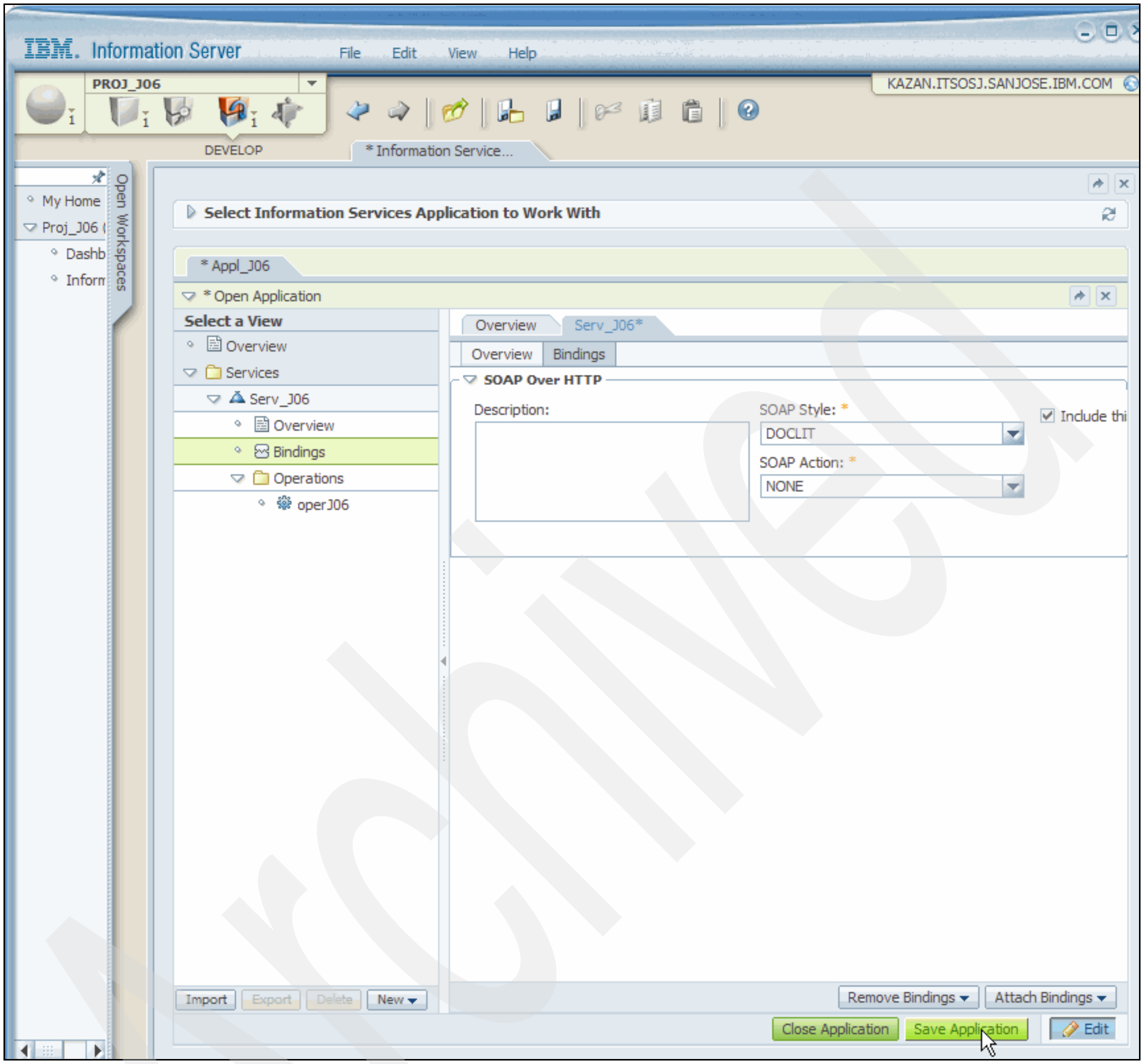


Figure 3-162 Generate SOA services, deploy, and test 12/21

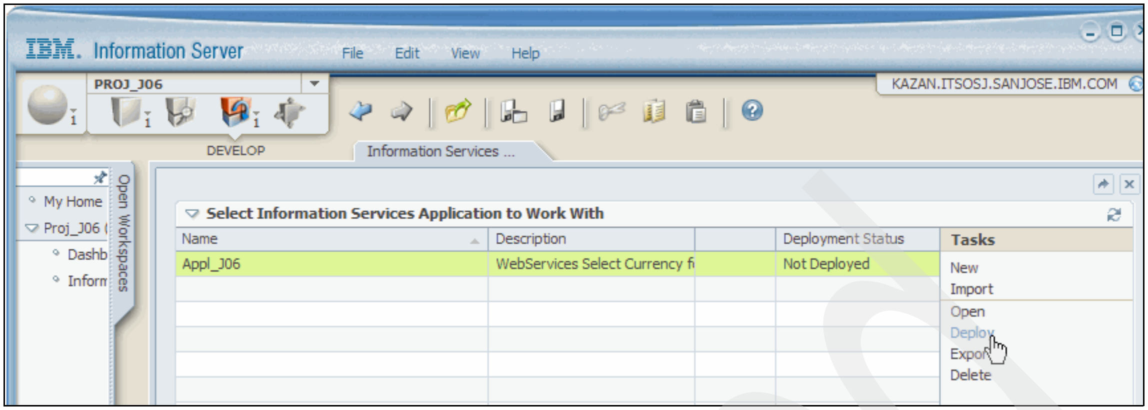


Figure 3-163 Generate SOA services, deploy, and test 13/21

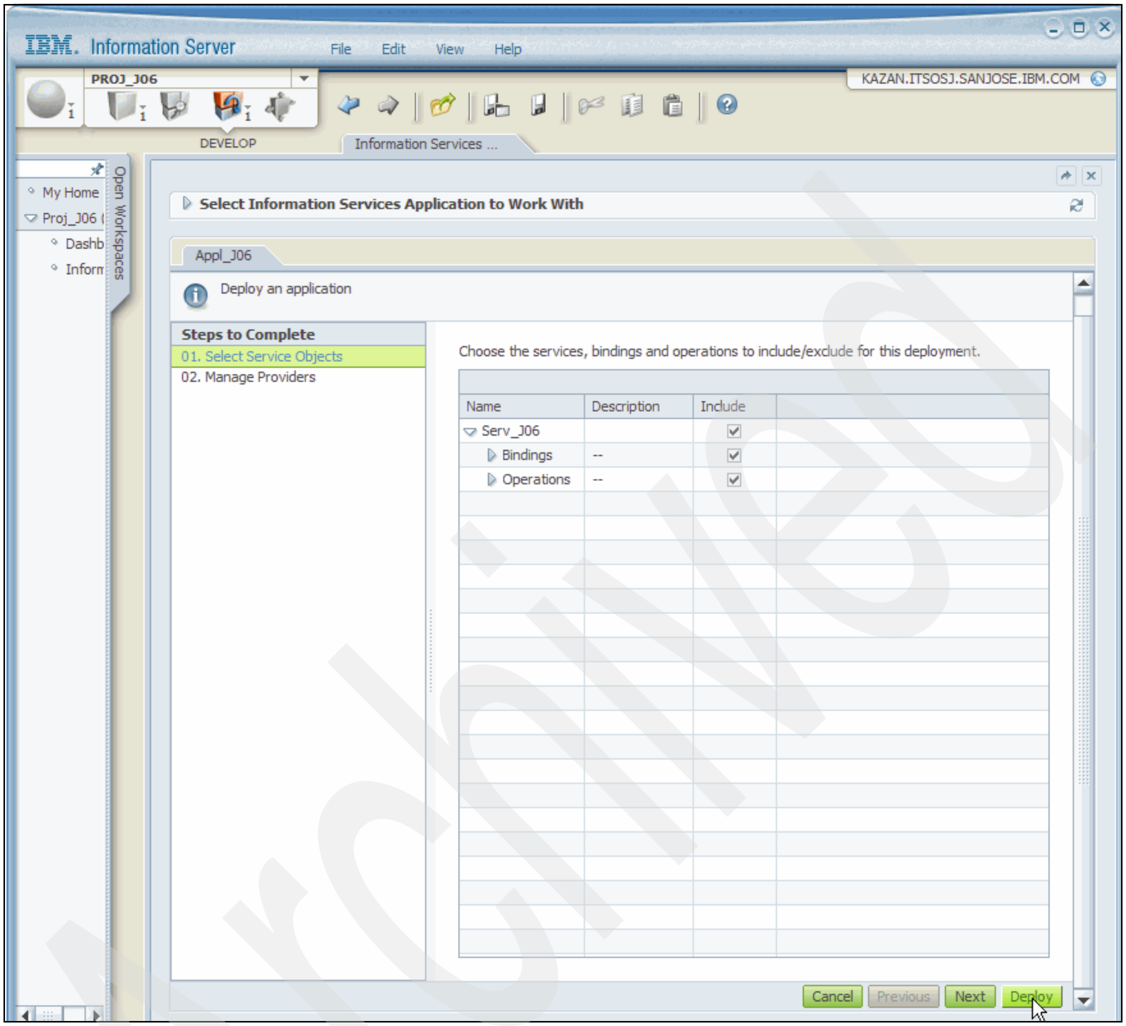


Figure 3-164 Generate SOA services, deploy, and test 14/21

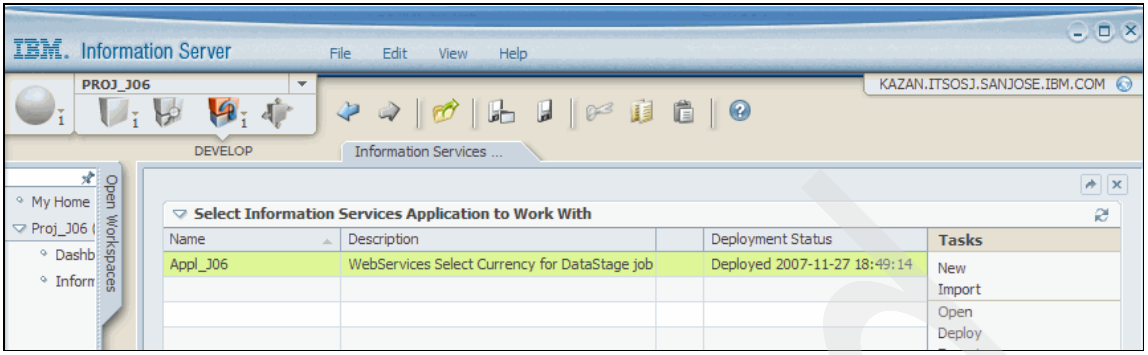


Figure 3-165 Generate SOA services, deploy, and test 15/21

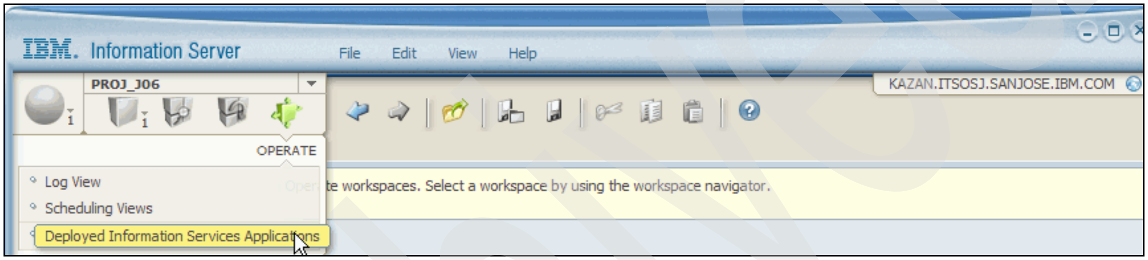


Figure 3-166 Generate SOA services, deploy, and test 16/21

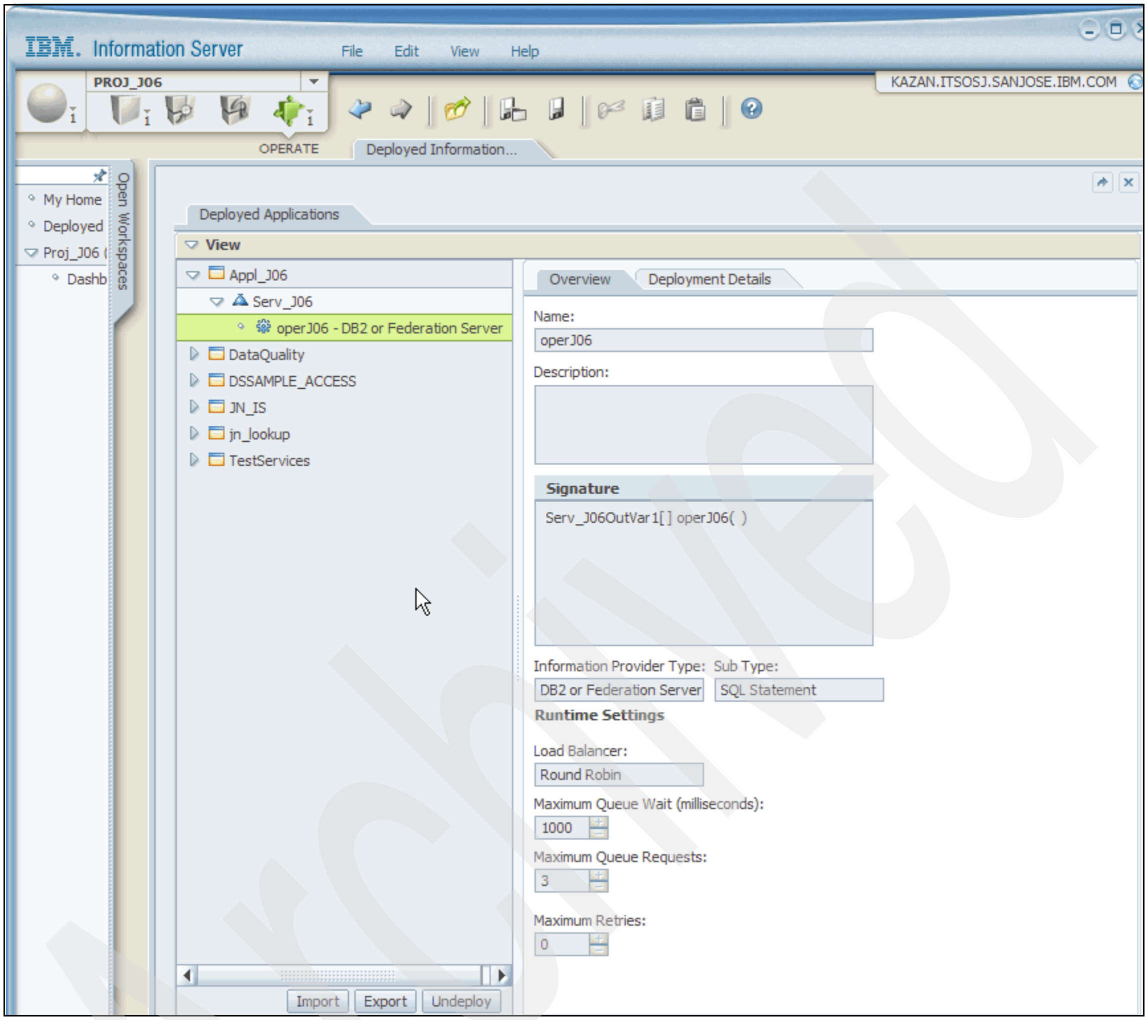


Figure 3-167 Generate SOA services, deploy, and test 17/21

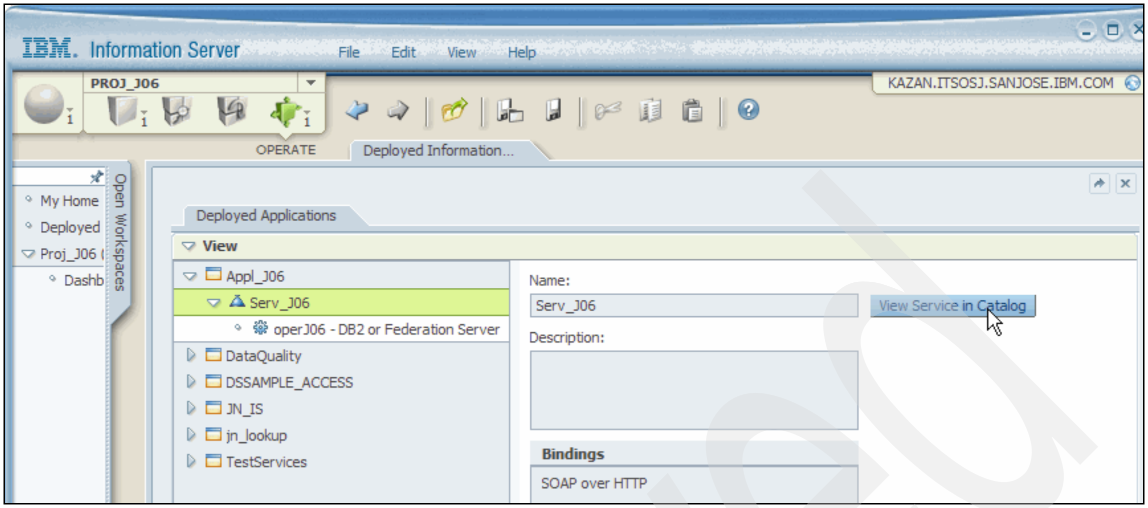


Figure 3-168 Generate SOA services, deploy, and test 18/21

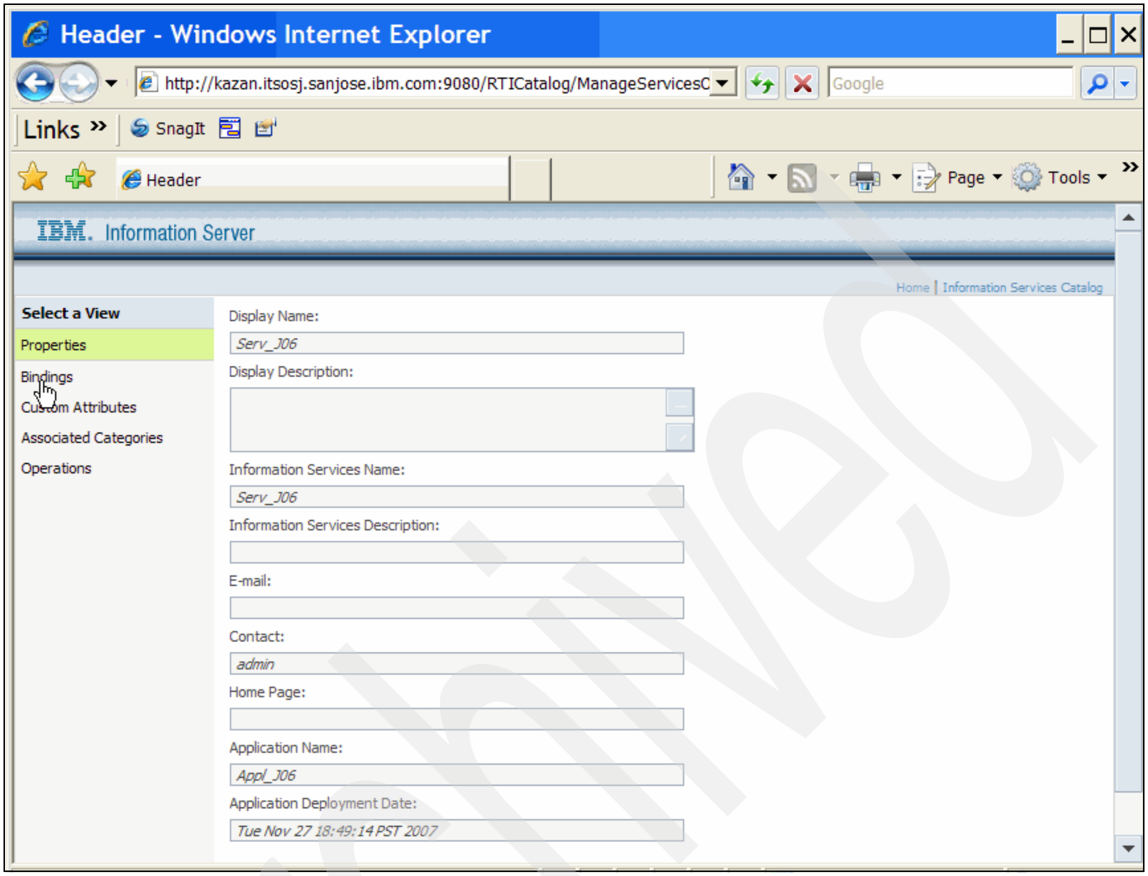


Figure 3-169 Generate SOA services, deploy, and test 19/21

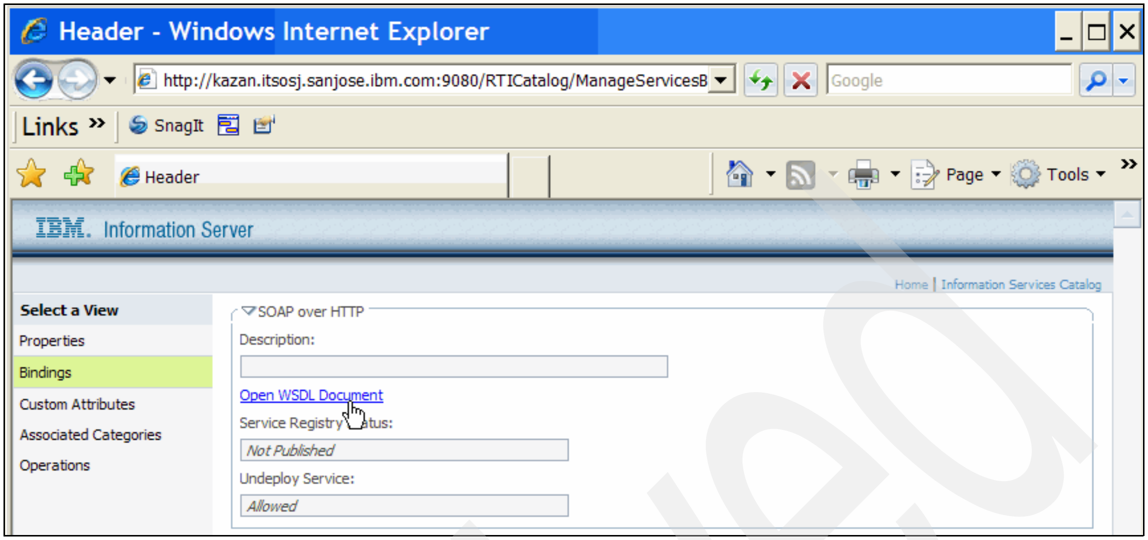


Figure 3-170 Generate SOA services, deploy, and test 20/21

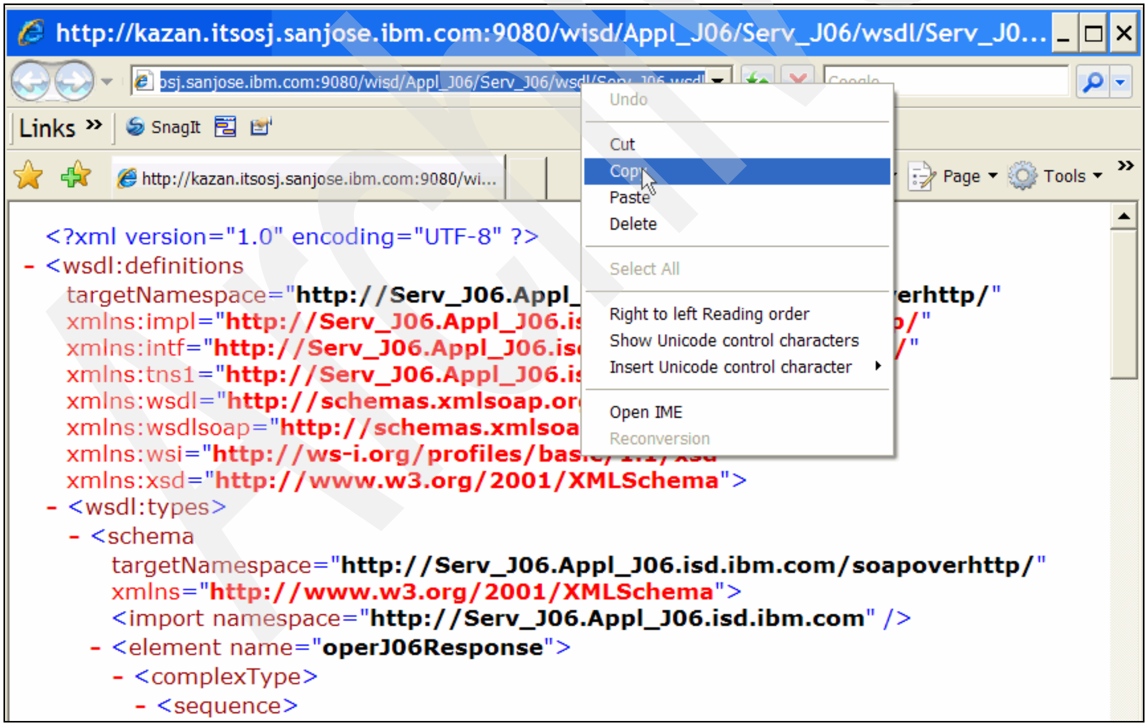


Figure 3-171 Generate SOA services, deploy, and test 21/21

Stepe: Load exchange rate info (Web service) to a data set

In this section, we access the exchange rates using the Web service created earlier and write its contents to a data set.

Figure 3-172 on page 263 through Figure 3-191 on page 272 describe some of the steps involved in accessing the exchange rates using the Web service and writing it in XML format to a data set. This involves importing the WSDL document for this Web service, and using the WSClientPX stage, an XMLInputPX stage, and a Data Set stage as shown in Figure 3-177 on page 266.

The main steps are as follows:

1. After launching the IBM InfoSphere DataStage and QualityStage Designer, click **Import** →**Table Definitions** →**Web Services WSDL Definitions..** as shown in Figure 3-172 on page 263.
2. Paste the URL for the WSDL document (that was copied in Figure 3-171 on page 259) in the Address field as shown in Figure 3-173 on page 263. Click **Import** in Figure 3-174 on page 264 to import the WSDL document into the Designer tool. Figure 3-175 on page 264 shows the successful import message.
3. Figure 3-176 on page 265 shows the partial contents of the CURRENCY table that stores currency exchange rates by ISO country code and date which is accessed by the Web service.
4. Figure 3-177 on page 266 shows the various stages used in this job — it includes a WSClientPX stage, an XMLInputPX stage, and a Data Set stage. The names of the stages were modified as shown.
5. Figure 3-178 on page 266 through Figure 3-181 on page 269 show the configuration of the Web Service Client stage that retrieves the exchange rates stored in a database via the Web service created in “Stepd: Generate SOA services, deploy, and test” on page 239.

   The Web Services Client stage is used when you need the Web service to act as either a data source or a data target during an operation. The Web Services Client stage encodes requests as SOAP messages and decodes responses from SOAP messages, using metadata that is defined for a Web service operation in its Web Services Description Language (WSDL).

   – Figure 3-178 on page 266 shows the Web service and operation to be accessed under the **General** tab in the Stage page.

   – Figure 3-179 on page 267 shows the **InputArguments** tab in the Output page, which identifies namespace information and input parameters for the Web service operation listed as a Stage property in Figure 3-178 on page 266. The output is XML content.

The Input Arguments page is used to:

- Load the namespace, input parameters, and other table definition information for the Web service that you specify on the General page of the Stage properties page. This information is used to create the SOAP message for a Web service request.
- Specify constants or job parameters (#param#) for each input parameter.
- Supply input SOAP header elements.
- Indicate whether or not a reference link supplies an input parameter.

– Figure 3-180 on page 268 shows the **OutputMessage** tab in the Output page, which contains message information from the Web service. Select the User-Defined Message check box, and select the column SOAPbody (of the linked stage that will receive the output message) in the drop-down list of the Choose the Column Receiving the User Message field.

The Output Message page is used to perform one of these actions:

- Load namespace information and output parameters from the table definition that contains WSDL information. The Web Services Client stage uses this information to create an output message.
- Specify the column on the output link that receives the response from the Web service.

– Figure 3-181 on page 269 shows the **Columns** tab in the Output page, which contains the column definition (SOAPbody) for the data being output.

Use the Columns page to:

- Inspect the definitions of output values.
- Load another table definition.

6. Figure 3-182 on page 269 through Figure 3-186 on page 271 show the configuration of the XMLInputPX stage that is used to convert XML data (generated by the WSClientPX stage in its output link Xml_Currency) to flat relational tables.

– Figure 3-182 on page 269 shows the **XML Source** tab in the Input page, which specifies the input column (SOAPbody) that contains the XML document.

– Figure 3-183 on page 269 shows the **Columns** tab in the Input page, which describes the input, including the location of the XML document that is transformed. It specifies the column definitions for the data written to the table or file on the chosen link.

- Figure 3-184 on page 270 shows the **Transformation Settings** tab in the Output page. It is used to:
    - Indicate that the output link inherits properties from the Stage page.
    - Replace missing elements and attributes with empty values.
    - Replace empty elements and empty values with NULLs.
    - Load namespaces from a table definition created with the XML Meta Data Importer.
    - Supply namespace declarations manually.
- Figure 3-185 on page 270 shows the **Columns** tab in the Output page, which specifies the output columns, including columns that receive the transformed output. The Derivation column identifies the source of each column in the output.

There are two major steps in using XML Input, as follows:

- Create mappings between XML and relational data.

  You create mappings for XML Input using the XML Meta Data Importer. The output is a table definition that contains a set of XML XPath expressions. These XPath expressions specify how to extract information from the XML document to a relational database format.

  You can also manually create XPath expressions through the XML Input stage.
- Add the XML Input stage to a server job.

  Drag-and-drop the XML Input stage to your server job, and set up properties within the stage.

7. Figure 3-187 on page 271 through Figure 3-191 on page 272 show the configuration of the Ds_Currency Data Set stage that and its partial contents after execution:
    - Figure 3-187 on page 271 shows the **Properties** tab in the Input page that identifies the location and name (J06_Dst_Currency.ds) of the data set.
    - Figure 3-188 on page 271 shows the **Columns** tab in the Input page that identifies the incoming input columns.
    - Figure 3-189 on page 272 shows the execution results of this job, indicating 135 records being written to the data set.
    - Figure 3-190 on page 272 and Figure 3-191 on page 272 show the partial contents of the data set created.

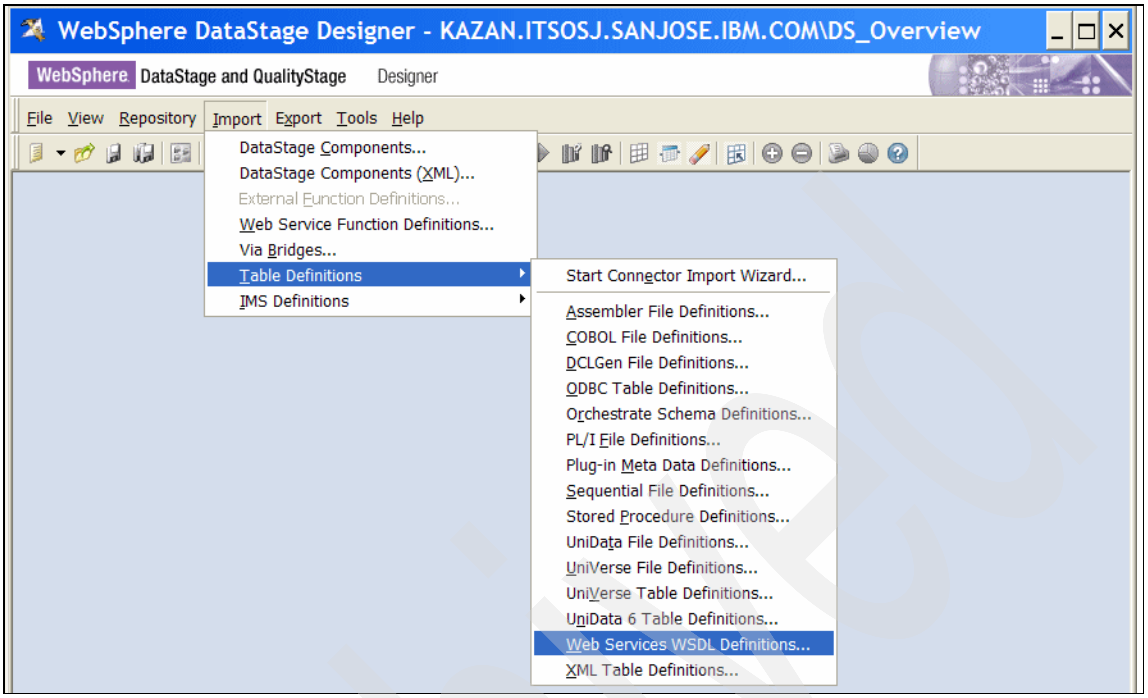


Figure 3-172 Load exchange rate information (Web service) to a data set 1/20

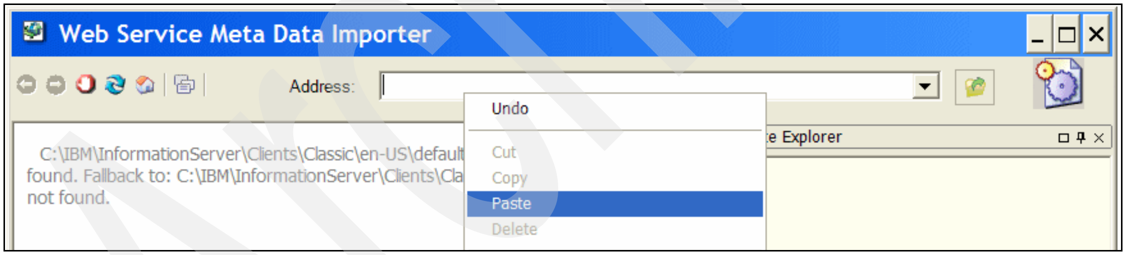


Figure 3-173 Load exchange rate information (Web service) to a data set 2/20

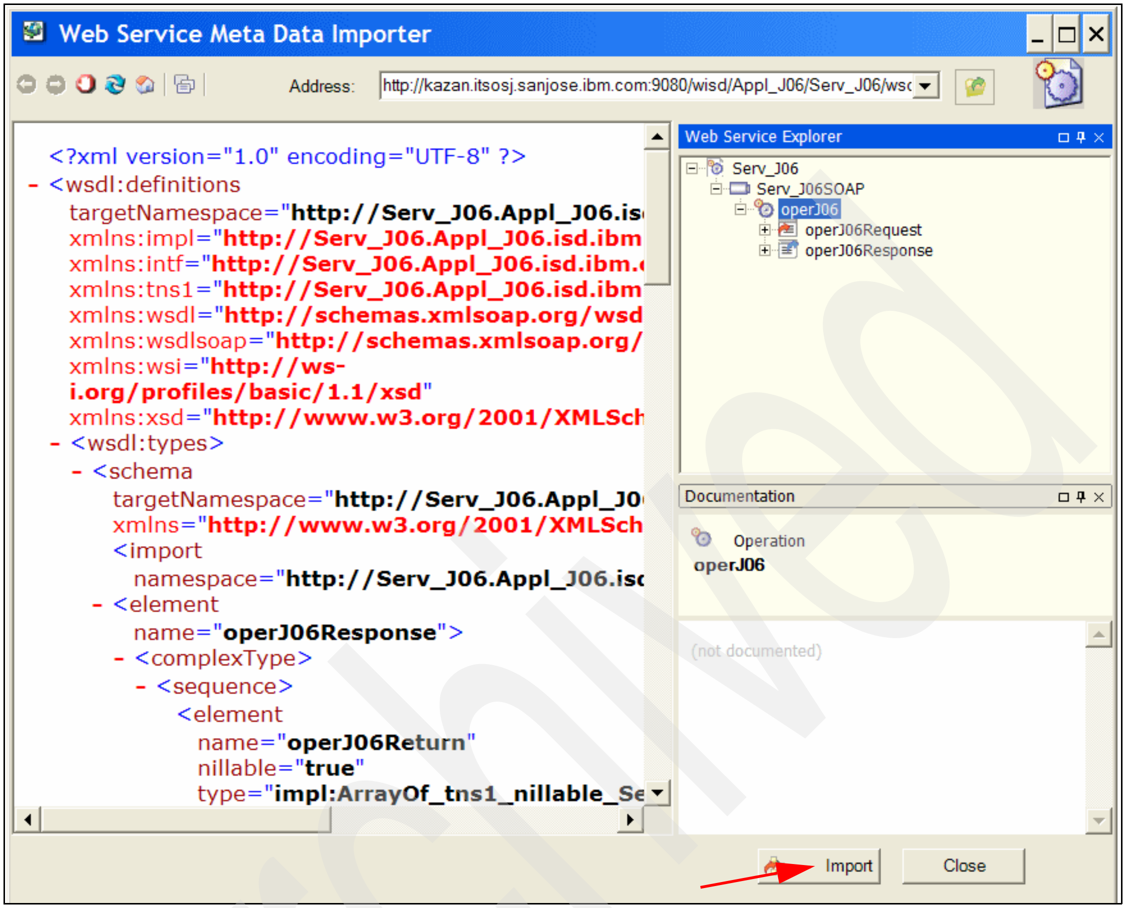


Figure 3-174 Load exchange rate information (Web service) to a data set 3/20

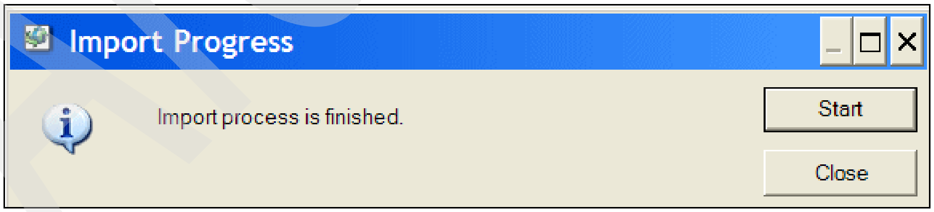
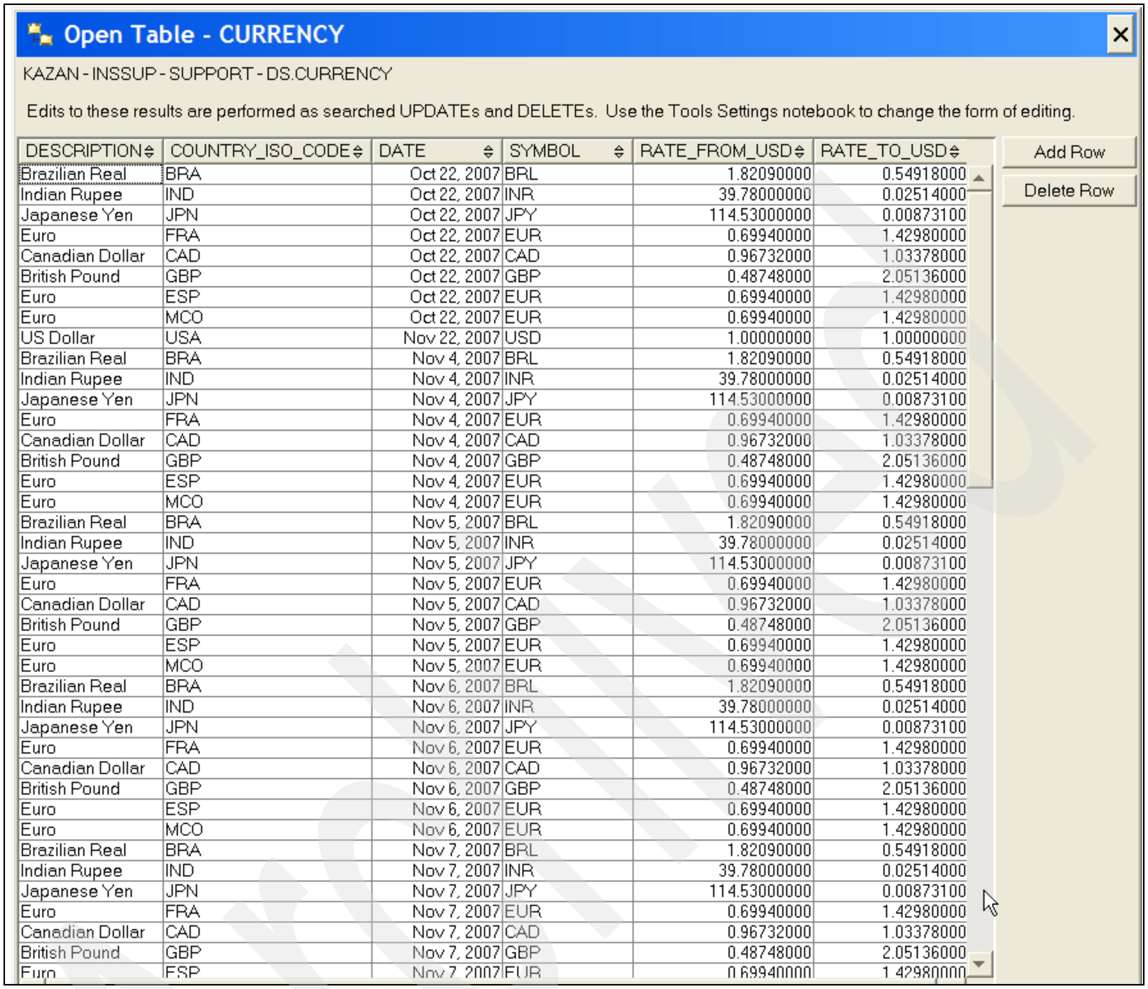


Figure 3-175 Load exchange rate information (Web service) to a data set 4/20



Open Table - CURRENCY

KAZAN - INSSUP - SUPPORT - DS.CURRENCY

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

DESCRIPTION	COUNTRY_ISO_CODE	DATE	SYMBOL	RATE_FROM_USD	RATE_TO_USD
Brazilian Real	BRA	Oct 22, 2007	BRL	1.82090000	0.54918000
Indian Rupee	IND	Oct 22, 2007	INR	39.78000000	0.02514000
Japanese Yen	JPN	Oct 22, 2007	JPY	114.53000000	0.00873100
Euro	FRA	Oct 22, 2007	EUR	0.69940000	1.42980000
Canadian Dollar	CAD	Oct 22, 2007	CAD	0.96732000	1.03378000
British Pound	GBP	Oct 22, 2007	GBP	0.48748000	2.05136000
Euro	ESP	Oct 22, 2007	EUR	0.69940000	1.42980000
Euro	MCO	Oct 22, 2007	EUR	0.69940000	1.42980000
US Dollar	USA	Nov 22, 2007	USD	1.00000000	1.00000000
Brazilian Real	BRA	Nov 4, 2007	BRL	1.82090000	0.54918000
Indian Rupee	IND	Nov 4, 2007	INR	39.78000000	0.02514000
Japanese Yen	JPN	Nov 4, 2007	JPY	114.53000000	0.00873100
Euro	FRA	Nov 4, 2007	EUR	0.69940000	1.42980000
Canadian Dollar	CAD	Nov 4, 2007	CAD	0.96732000	1.03378000
British Pound	GBP	Nov 4, 2007	GBP	0.48748000	2.05136000
Euro	ESP	Nov 4, 2007	EUR	0.69940000	1.42980000
Euro	MCO	Nov 4, 2007	EUR	0.69940000	1.42980000
Brazilian Real	BRA	Nov 5, 2007	BRL	1.82090000	0.54918000
Indian Rupee	IND	Nov 5, 2007	INR	39.78000000	0.02514000
Japanese Yen	JPN	Nov 5, 2007	JPY	114.53000000	0.00873100
Euro	FRA	Nov 5, 2007	EUR	0.69940000	1.42980000
Canadian Dollar	CAD	Nov 5, 2007	CAD	0.96732000	1.03378000
British Pound	GBP	Nov 5, 2007	GBP	0.48748000	2.05136000
Euro	ESP	Nov 5, 2007	EUR	0.69940000	1.42980000
Euro	MCO	Nov 5, 2007	EUR	0.69940000	1.42980000
Brazilian Real	BRA	Nov 6, 2007	BRL	1.82090000	0.54918000
Indian Rupee	IND	Nov 6, 2007	INR	39.78000000	0.02514000
Japanese Yen	JPN	Nov 6, 2007	JPY	114.53000000	0.00873100
Euro	FRA	Nov 6, 2007	EUR	0.69940000	1.42980000
Canadian Dollar	CAD	Nov 6, 2007	CAD	0.96732000	1.03378000
British Pound	GBP	Nov 6, 2007	GBP	0.48748000	2.05136000
Euro	ESP	Nov 6, 2007	EUR	0.69940000	1.42980000
Euro	MCO	Nov 6, 2007	EUR	0.69940000	1.42980000
Brazilian Real	BRA	Nov 7, 2007	BRL	1.82090000	0.54918000
Indian Rupee	IND	Nov 7, 2007	INR	39.78000000	0.02514000
Japanese Yen	JPN	Nov 7, 2007	JPY	114.53000000	0.00873100
Euro	FRA	Nov 7, 2007	EUR	0.69940000	1.42980000
Canadian Dollar	CAD	Nov 7, 2007	CAD	0.96732000	1.03378000
British Pound	GBP	Nov 7, 2007	GBP	0.48748000	2.05136000
Euro	ESP	Nov 7, 2007	EUR	0.69940000	1.42980000

Figure 3-176 Load exchange rate information (Web service) to a data set 5/20

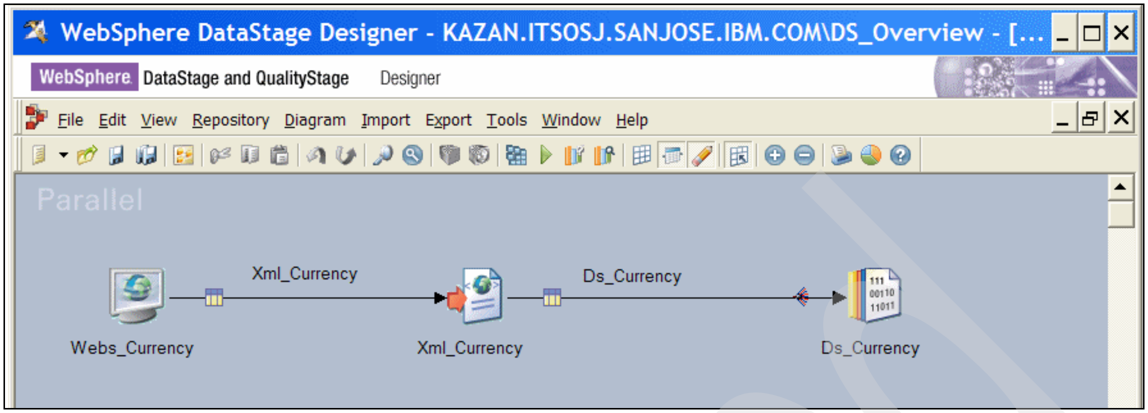


Figure 3-177 Load exchange rate information (Web service) to a data set 6/20

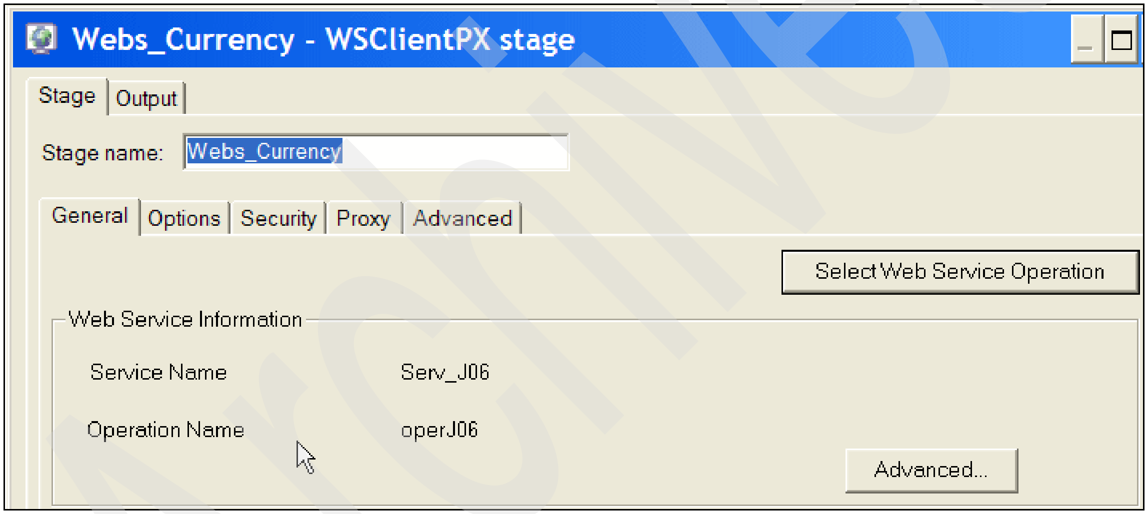


Figure 3-178 Load exchange rate information (Web service) to a data set 7/20

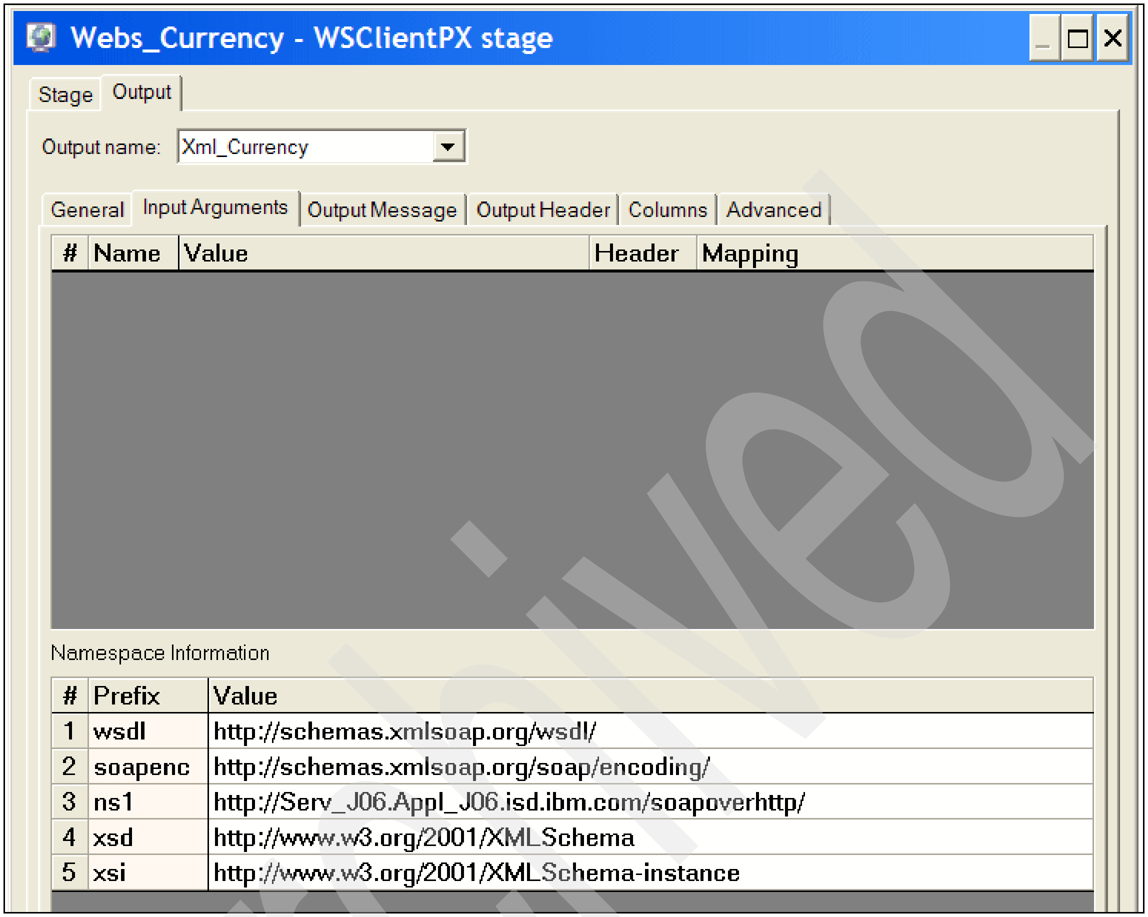


Figure 3-179 Load exchange rate information (Web service) to a data set 8/20

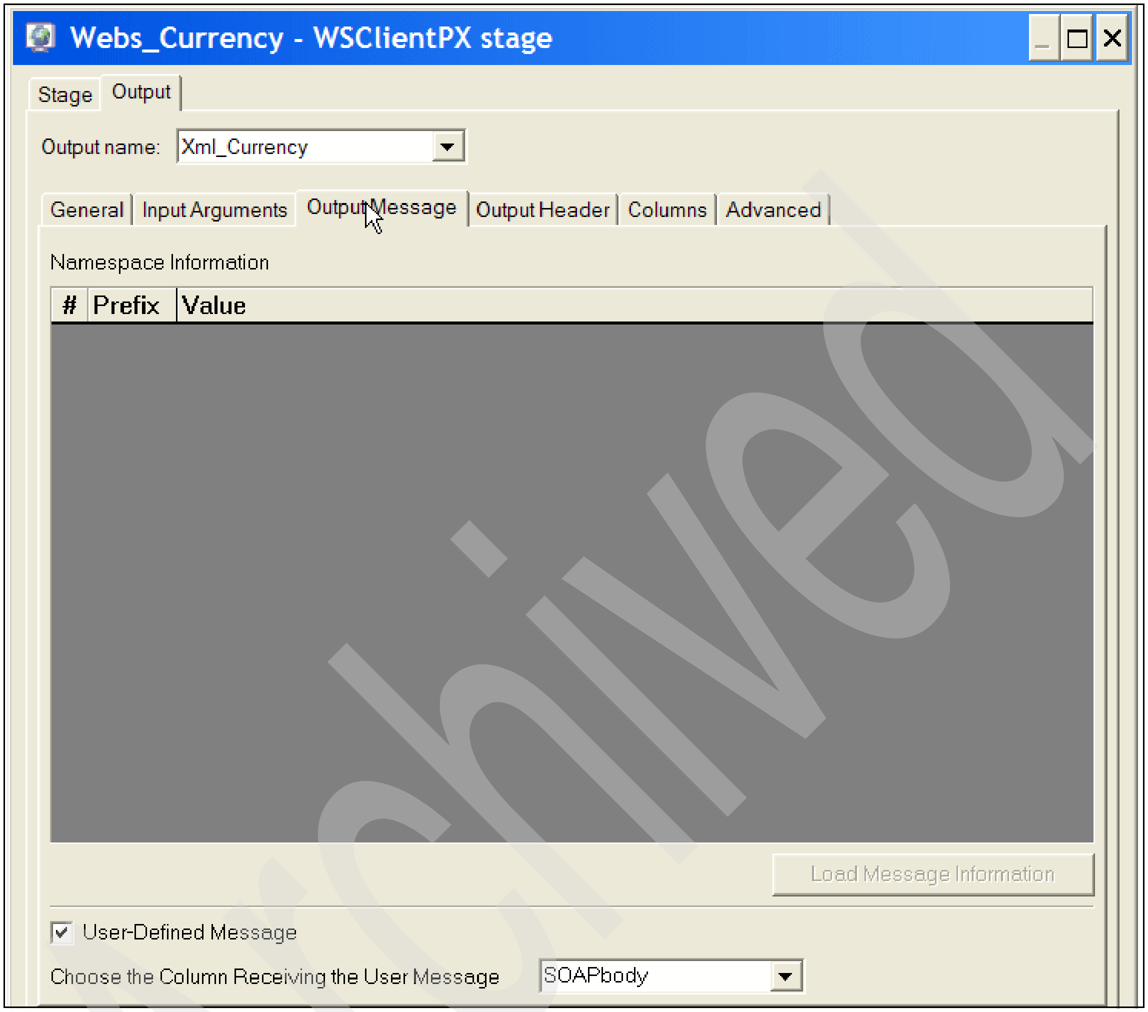


Figure 3-180 Load exchange rate information (Web service) to a data set 9/20

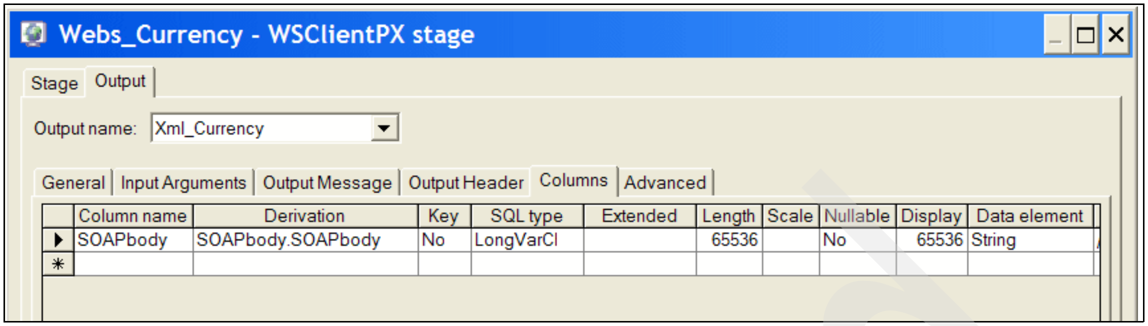


Figure 3-181 Load exchange rate information (Web service) to a data set 10/20

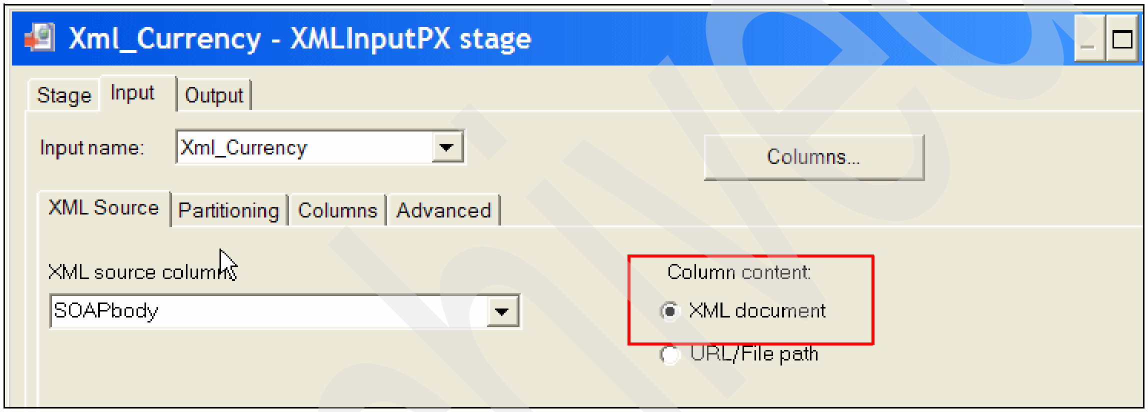


Figure 3-182 Load exchange rate information (Web service) to a data set 11/20

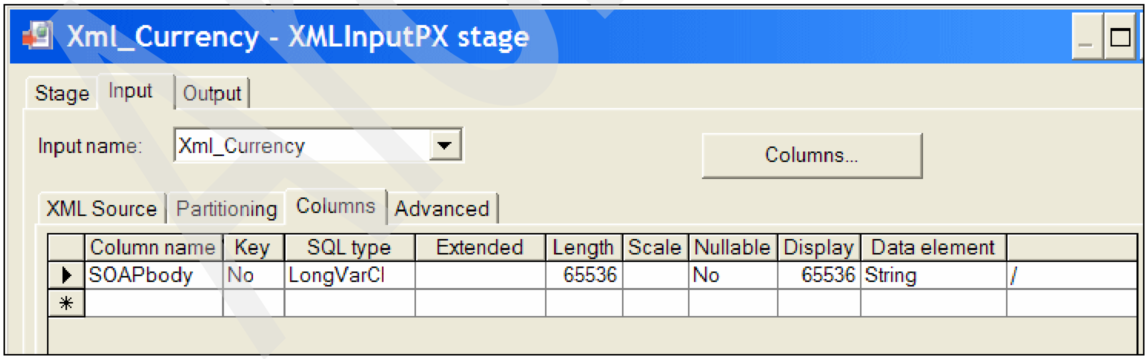


Figure 3-183 Load exchange rate information (Web service) to a data set 12/20

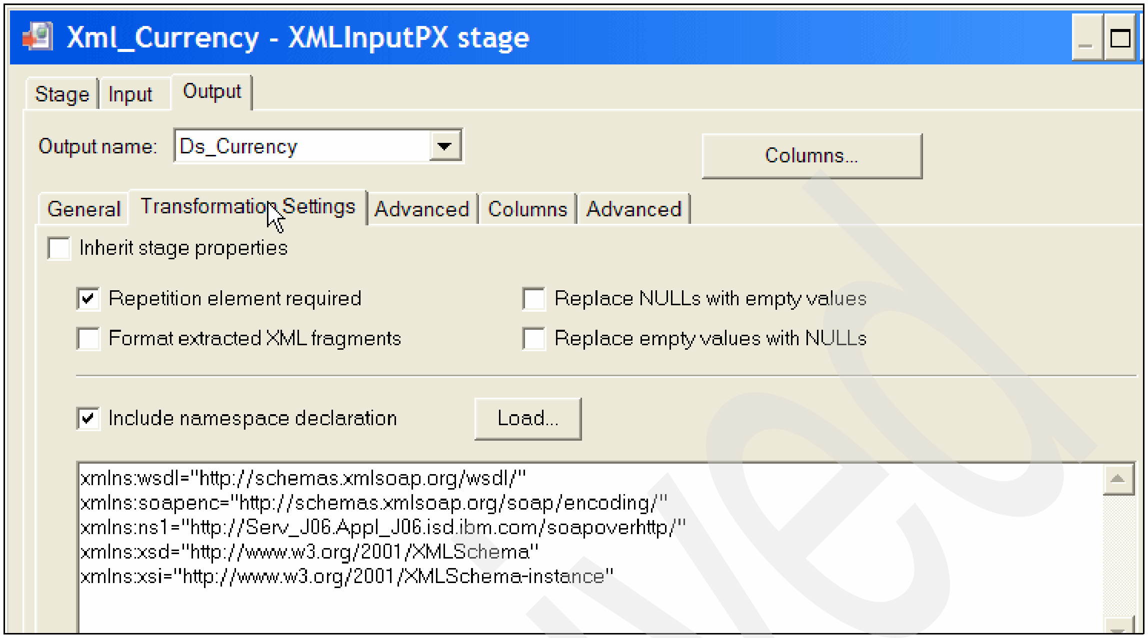


Figure 3-184 Load exchange rate information (Web service) to a data set 13/20

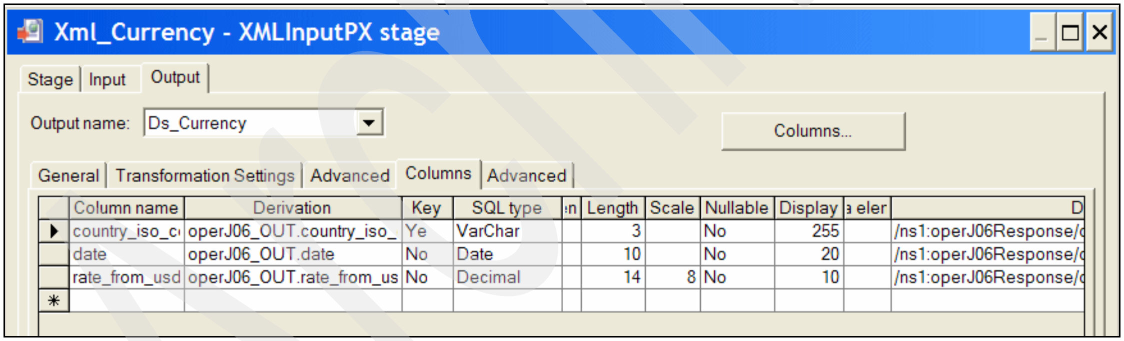


Figure 3-185 Load exchange rate information (Web service) to a data set 14/20

Figure 3-186 Load exchange rate information (Web service) to a data set 15/20

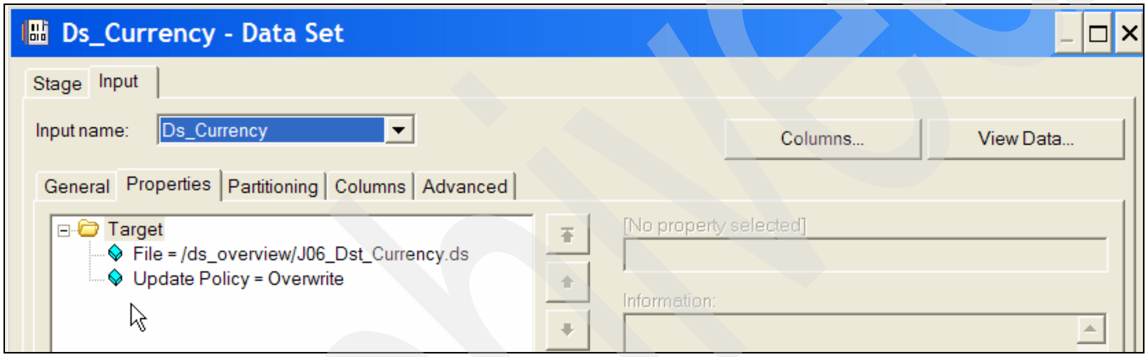


Figure 3-187 Load exchange rate information (Web service) to a data set 16/20

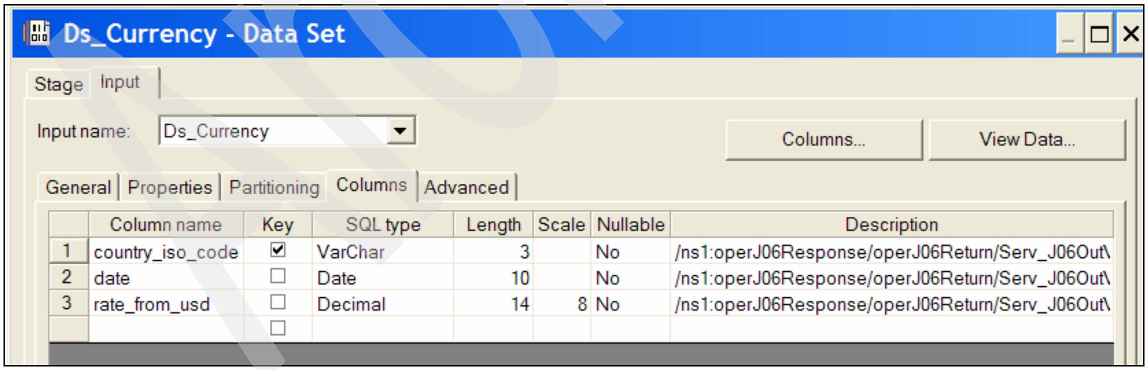


Figure 3-188 Load exchange rate information (Web service) to a data set 17/20

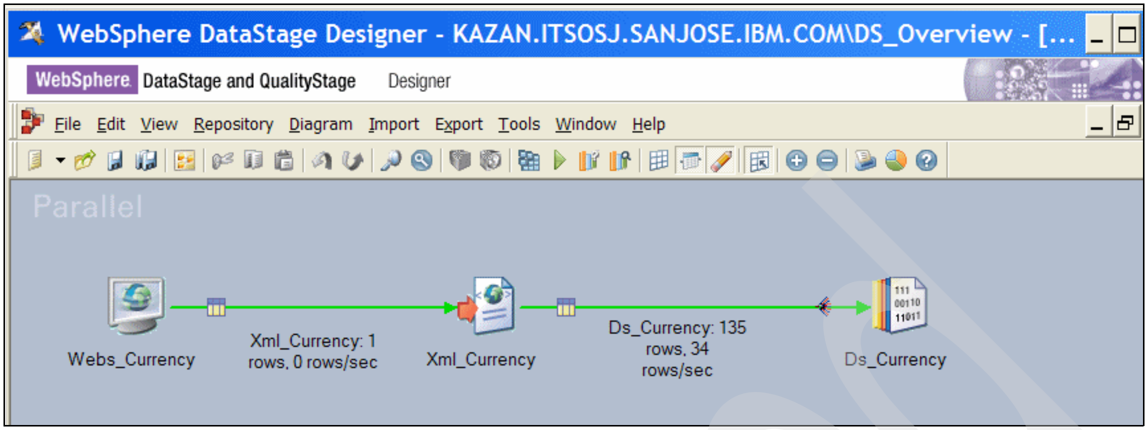
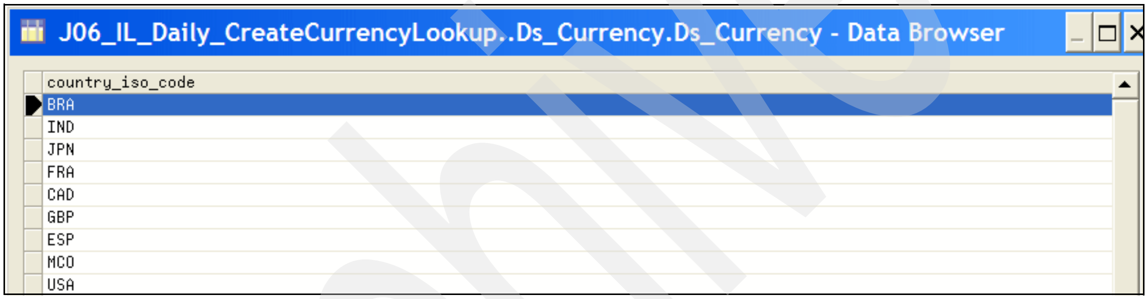
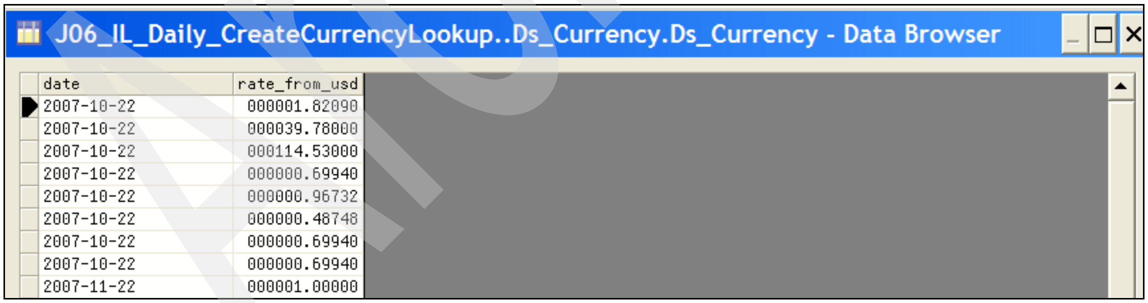


Figure 3-189 Load exchange rate information (Web service) to a data set 18/20


J06_IL_Daily_CreateCurrencyLookup..Ds_Currency.Ds_Currency - Data Browser

country_iso_code
BRA
IND
JPN
FRA
CAD
GBP
ESP
MCO
USA

Figure 3-190 Load exchange rate information (Web service) to a data set 19/20


J06_IL_Daily_CreateCurrencyLookup..Ds_Currency.Ds_Currency - Data Browser

date	rate_from_usd
2007-10-22	000001.82090
2007-10-22	000039.78000
2007-10-22	000114.53000
2007-10-22	000000.69940
2007-10-22	000000.96732
2007-10-22	000000.48748
2007-10-22	000000.69940
2007-10-22	000000.69940
2007-11-22	000001.00000

Figure 3-191 Load exchange rate information (Web service) to a data set 20/20

J07A_SharedContainerLookupCurrency

"J06_IL_Daily_CreateCurrencyLookup_Service" on page 227 described the creation of a Web service to retrieve the daily exchange rate for foreign currency vis-a-vis the US dollar.

In this section, we create a shared container[4] where input consists of the iso country code, date (date), and the total US dollar value (of the sales transaction). This input is processed using a Lookup stage and a Transformer stage to produce an output that corresponds to the corresponding foreign currency (local to the credit card) equivalent of the US dollar transaction. The objective here was to showcase the shared container capability of IBM InfoSphere DataStage.

Figure 3-192 on page 275 through Figure 3-202 on page 282 describe the main steps in creating a shared container that converts a $US (sales transaction) amount into the equivalent amount in the foreign currency of a given iso country. This shared container is used in the "J07_IL_Daily_LoadSalesStore" on page 282 job.

1. Figure 3-192 on page 275 shows the various stages in the Parallel Container — it includes a Container Input interface, a Container Output interface, a Data Set stage, a Lookup stage, a Sequential file stage, and a Transformer stage. The names of the stages were modified as shown.
2. Figure 3-193 on page 276 shows the **Columns** tab in the Output page of the Data Set stage which includes the country_iso_code, date and rate_from_usd columns. This data set is created daily (as shown in Figure 3-177 on page 266) from the Web service and is used here for performance reasons, since accessing the Web service for each incoming sales transaction would

[4] Instances of a shared container can be reused in different parallel jobs. You can use shared containers to make common job components available throughout the project. You can create a shared container from a stage and associated metadata and add the shared container to the palette to make this pre-configured stage available to other jobs. Shared containers comprise groups of stages and links and are stored in the metadata repository like IBM InfoSphere DataStage jobs. When you insert a shared container into a job, IBM InfoSphere DataStage places an instance of that container into the design. When you compile the job containing an instance of a shared container, the code for the container is included in the compiled job. You can use the InfoSphere DataStage debugger on instances of shared containers used within server jobs. When you add an instance of a shared container to a job, you will have to map metadata for the links into and out of the container, as these may vary in each job in which you use the shared container. If you change the contents of a shared container, you will have to recompile those jobs that use the container in order for the changes to take effect. For parallel shared containers, you can take advantage of runtime column propagation to avoid having to map the metadata. If you enable runtime column propagation, then, when the job runs, metadata will be automatically propagated across the boundary between the shared container and the stage(s) to which it connects in the job. You can create a shared container from scratch, or place a set of existing stages and links within a shared container.

be expensive. This data set contains all the exchange rates for all dates for all iso countries. Runtime column propagation is enabled so that any extra columns that are not defined in the metadata when it actually runs, will be adopted and propagated through the rest of the job.

3. Figure 3-194 on page 277 through Figure 3-197 on page 279 show the configuration of the Lookup stage. For each record of the source data set from the primary link (shared_cont), the Lookup stage performs a table lookup on the lookup table attached by reference link (Ds_rate).

   – The table lookup is based on the values of a set of lookup key columns (COUNTRY_ISO_CODE and LookupDate in the Ds_rate link and the country_iso_code and date in the shared_cont link). These are identified in Figure 3-194 on page 277 through Figure 3-196 on page 279. You can specify a condition on the reference link such that the stage will only perform a lookup on that reference link if the condition is satisfied. The equality condition is used here as shown in Figure 3-194 on page 277 through Figure 3-196 on page 279.

   – Each record of the output link (Trx_LocCurrency) contains columns from the source plus columns from all the corresponding lookup record where corresponding source and lookup record have the same value for the lookup key columns. The lookup key columns do not have to have the same names in the primary and the reference links. The TOTALUSD column from the shared_cont primary link and the rate_from_usd column from the Ds_rate reference link are copied to the output link (Trx_LocCurrency). This is shown in Figure 3-194 on page 277 through Figure 3-196 on page 279.

   – Figure 3-197 on page 279 shows the **Link Ordering** tab in the Stage page, which identifies the Primary (link) as being the shared_cont link and the Lookup (Reference link) as being the Ds_rate link.

   – The optional reject link Ds_reject carries source records that do not have a corresponding entry in the input lookup table. Figure 3-198 on page 280 through Figure 3-200 on page 281 identify the Properties (such as the name), Format, and Columns of the sequential file to which the rejected records are written.

4. Figure 3-201 on page 281 shows the configuration of the Transformer stage that computes the value of equivalent of the $US value in foreign currency by multiplying the $US amount by the exchange rate.

5. Figure 3-202 on page 282 shows the contents of the Ds_reject link which has an invalid ISO code CHN in the source.

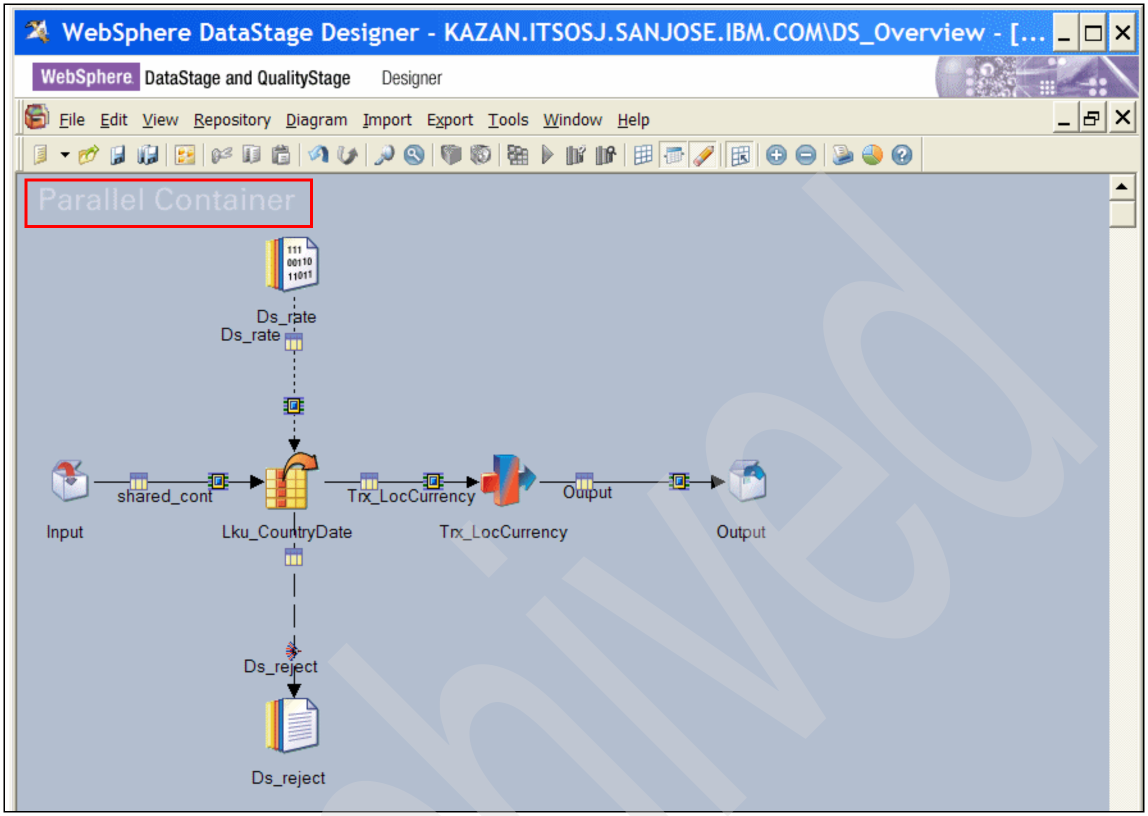


Figure 3-192 Create the J07A_SharedContainerLookupCurrency job 1/11

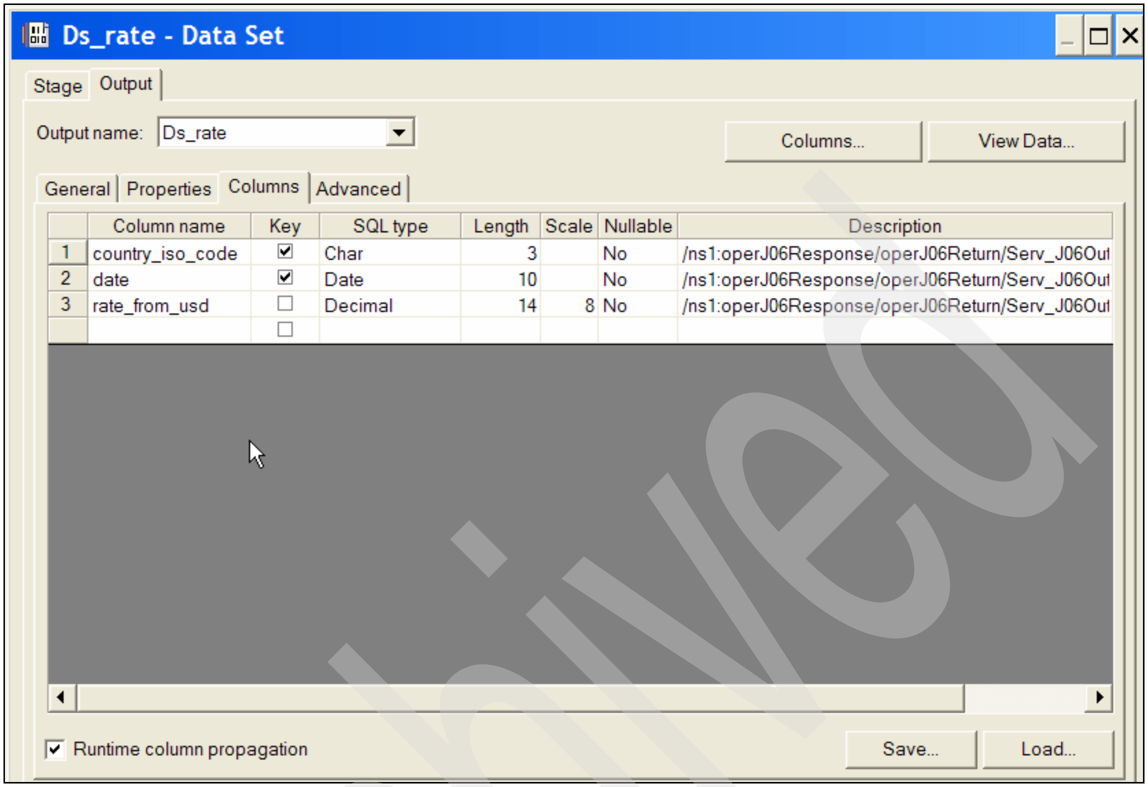


Figure 3-193 Create the J07A_SharedContainerLookupCurrency job 2/11

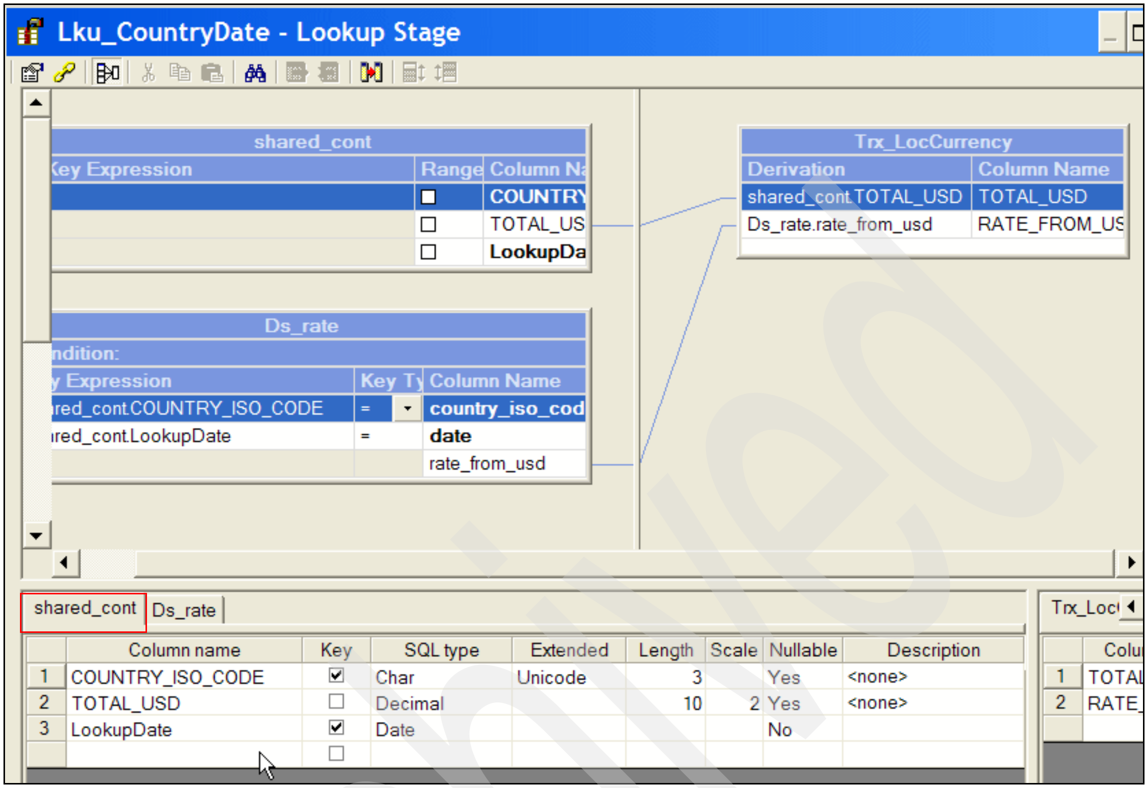


Figure 3-194 Create the J07A_SharedContainerLookupCurrency job 3/11

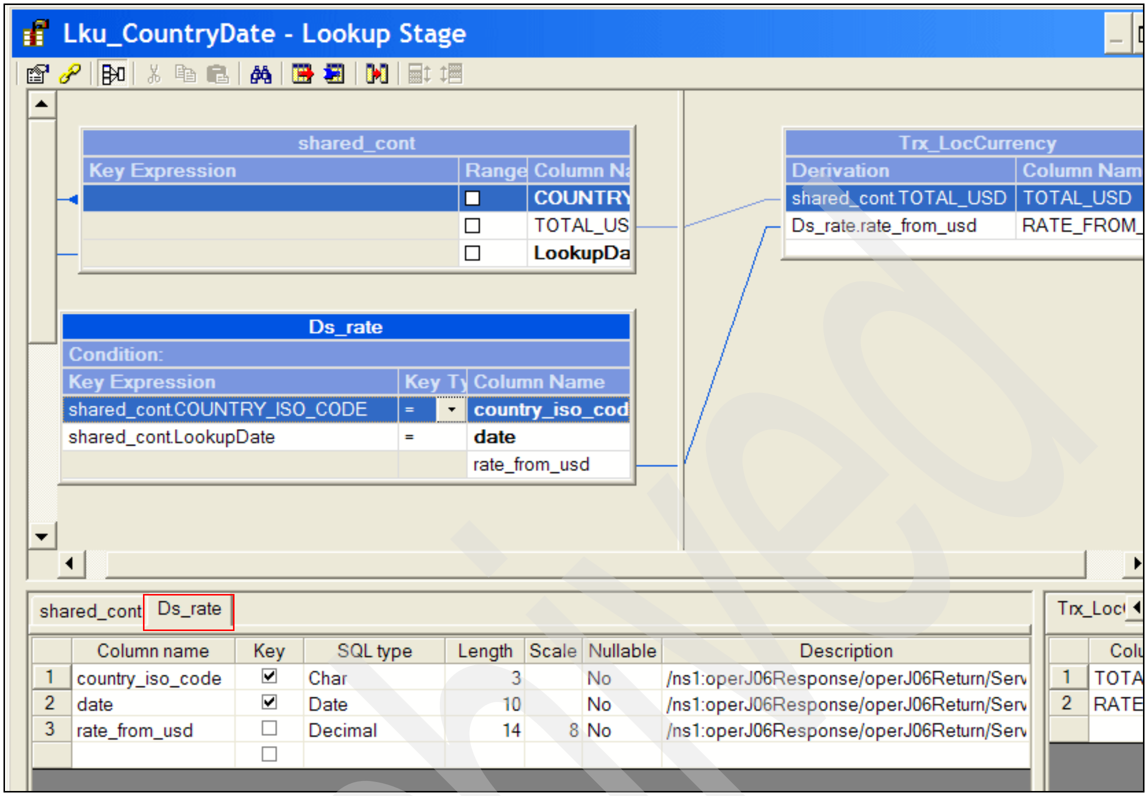


Figure 3-195 Create the J07A_SharedContainerLookupCurrency job 4/11

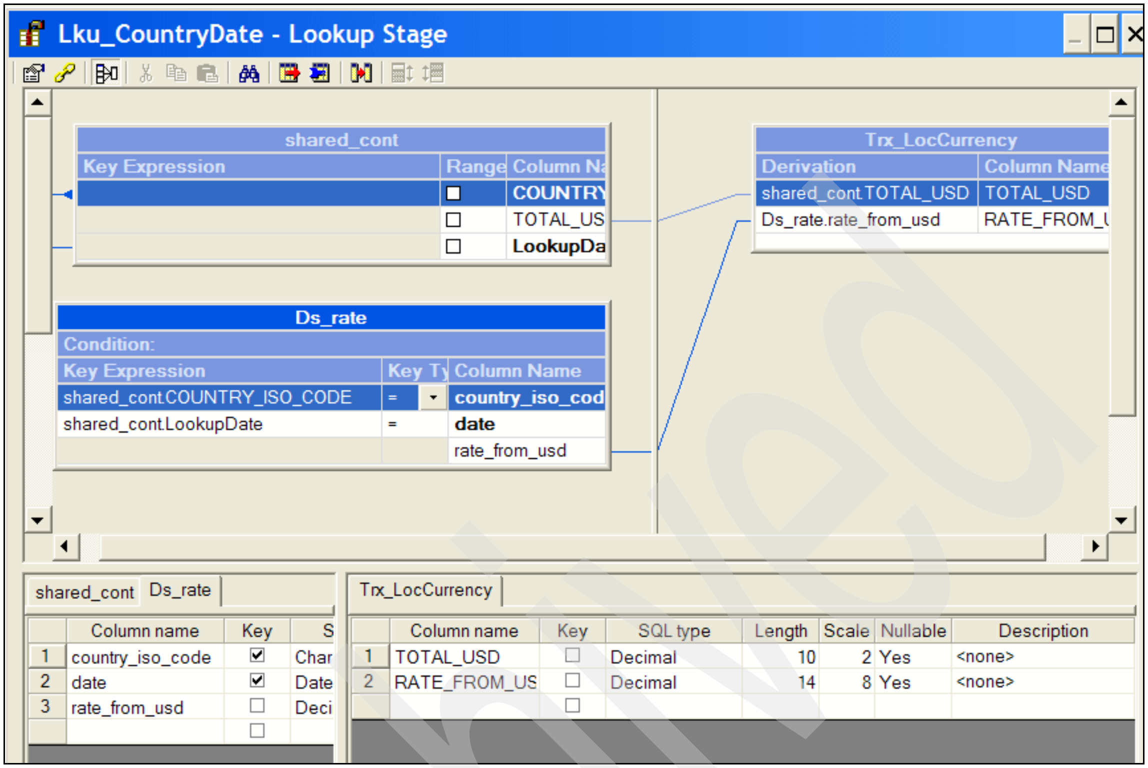


Figure 3-196   Create the J07A_SharedContainerLookupCurrency job 5/11

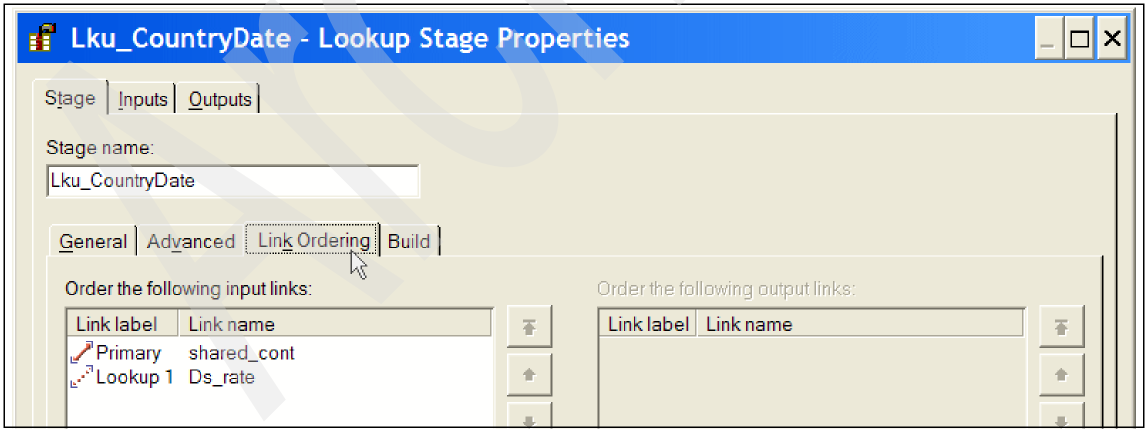


Figure 3-197   Create the J07A_SharedContainerLookupCurrency job 6/11

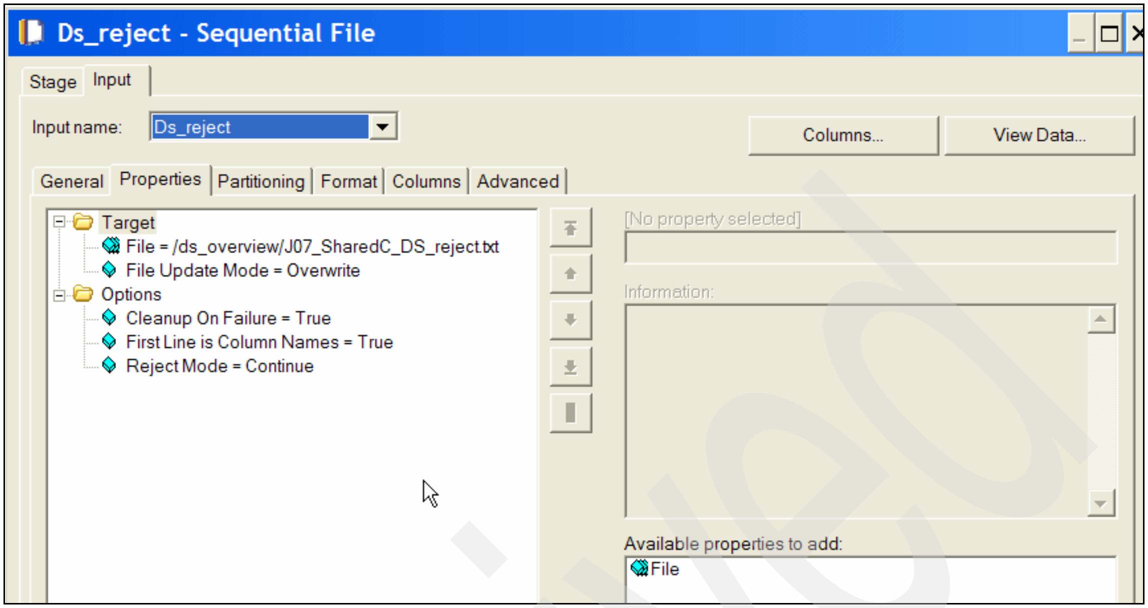


Figure 3-198 Create the J07A_SharedContainerLookupCurrency job 7/11

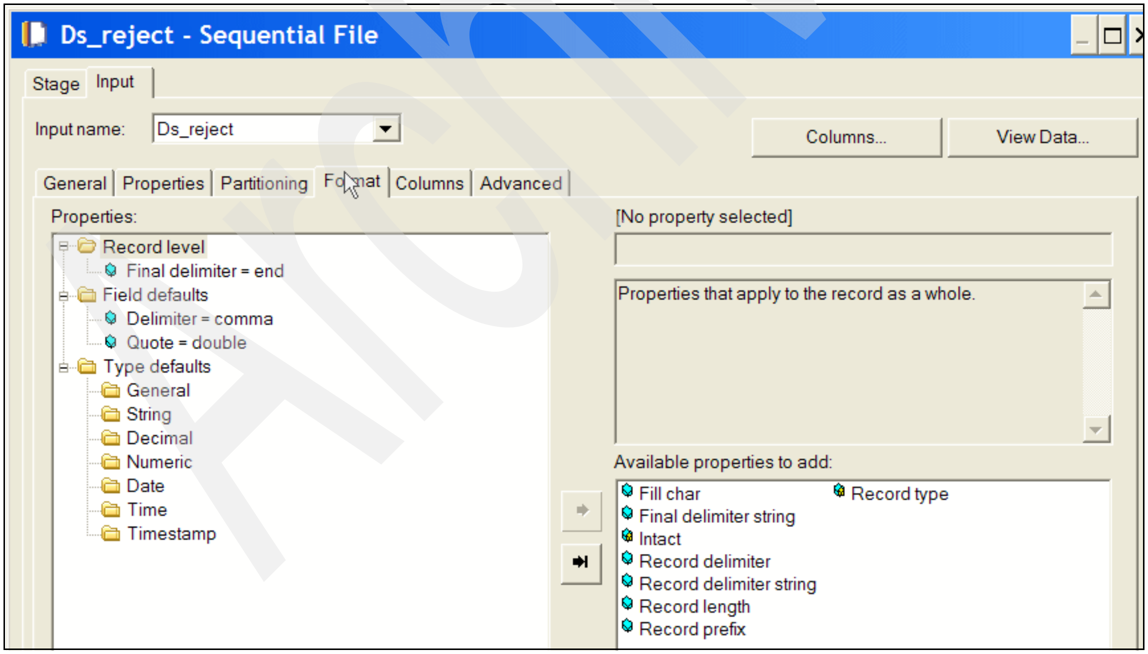


Figure 3-199 Create the J07A_SharedContainerLookupCurrency job 8/11

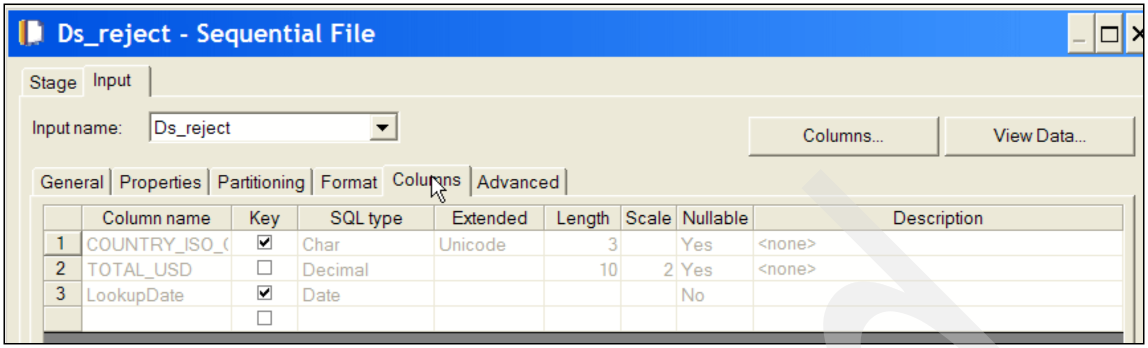


Figure 3-200 Create the J07A_SharedContainerLookupCurrency job 9/11

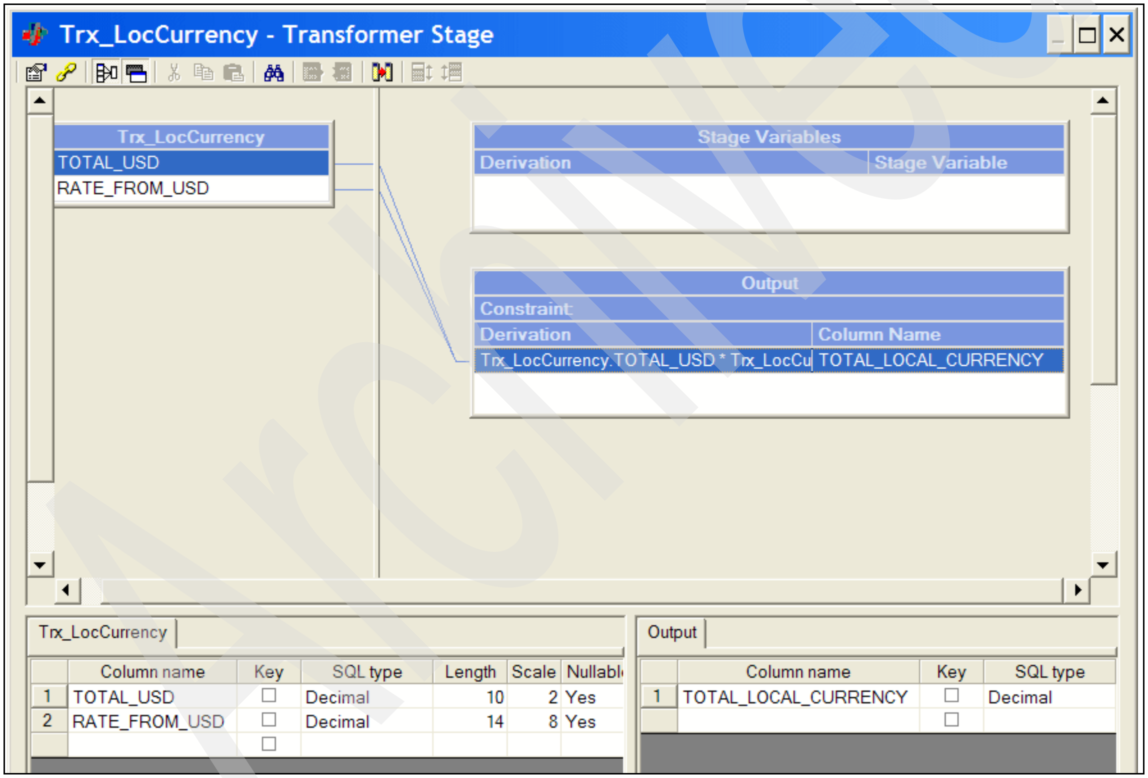


Figure 3-201 Create the J07A_SharedContainerLookupCurrency job 10/11

Figure 3-202 Create the J07A_SharedContainerLookupCurrency job 11/11

J07_IL_Daily_LoadSalesStore

As mentioned earlier, some customers use non-US credit cards to purchase products at the various WantThatStuff stores. The individual sales transactions captured at the individual stores are in $US, but the foreign currency equivalent must be determined and then loaded into an interim DB2 table for subsequent loading into the Sales fact table.

In this job, we compute the foreign currency equivalent for a sales transaction involving a non-US credit card using the shared container stage created in "J07A_SharedContainerLookupCurrency" on page 273 and write it to an interim DB2 table for subsequent processing prior to being loaded into the sales fact table.

Figure 3-203 on page 283 through Figure 3-211 on page 289 describe the main steps processing sales transactions from the stores and generating the foreign currency equivalent of the $US amount before writing it to a DB2 table.

1. Figure 3-203 on page 283 shows the various stages in the job — it includes a Sequential file stage, a Transformer stage, a shared container stage, a Copy stage and an ODBCConnectorPX stage. The names of the stages were modified as shown.
2. Figure 3-204 on page 284 shows the configuration of the Sequential file containing the sales transactions of an individual store. It shows the **Columns** tab in the Output page, which identifies all the columns associated with a sales transaction. Note in particular the Timestamp data type of the DATE column, and the Runtime column propagation box being checked.
3. The COUNTRY_ISO_CODE, DATE, and TOTAL_USD columns are required input to the shared container stage described in "J07A_SharedContainerLookupCurrency" on page 273. However, the shared container requires a DATE data type and not TIMESTAMP. Therefore an intervening Transformer stage (Trx_Conv) is required to convert the TIMESTAMP data type to a DATE data type using the TimestampToDate function as shown in Figure 3-205 on page 285.

4. The output of the shared container stage is then written to a Data Set involving a one-to-one mapping of the columns using a Copy stage as shown in Figure 3-206 on page 286 and Figure 3-207 on page 286. This stage was introduced to disable Runtime column propagation (as shown in Figure 3-208 on page 287) so that only the columns of interest (as identified in Figure 3-208 on page 287) are passed to the ODBCConnectorPX stage.
5. The output of the Copy stage is then loaded into a DB2 table using an ODBCConnectorPX stage as shown in Figure 3-209 on page 288. The INSERT SQL statement is automatically generated as shown.
6. This job is then executed twice — once for store transactions corresponding to store ST1 and the second corresponding to store ST33. Note the enablement of runtime column propagation as shown in Figure 3-211 on page 289:
   - Figure 3-210 on page 289 through Figure 3-214 on page 290 show the job properties and execution results associated with store ST1 which has 5 sales transactions. These 5 sales transactions are shown in Figure 3-217 on page 291 and Figure 3-218 on page 292. Note the foreign currency equivalents of the $US amounts and the country iso code associated with each sales transaction.
   - Figure 3-215 on page 291 and Figure 3-216 on page 291 show the job properties and execution results associated with store ST33 which has 2 sales transactions. These 2 sales transactions are shown in Figure 3-219 on page 292 and Figure 3-220 on page 292. Note the foreign currency equivalents of the $US amounts and the country iso code associated with each sales transaction.

You can now proceed to load the sales fact table with the sales transactions in the interim DB2 tables as described in “J08_IL_LoadSalesFact” on page 292.

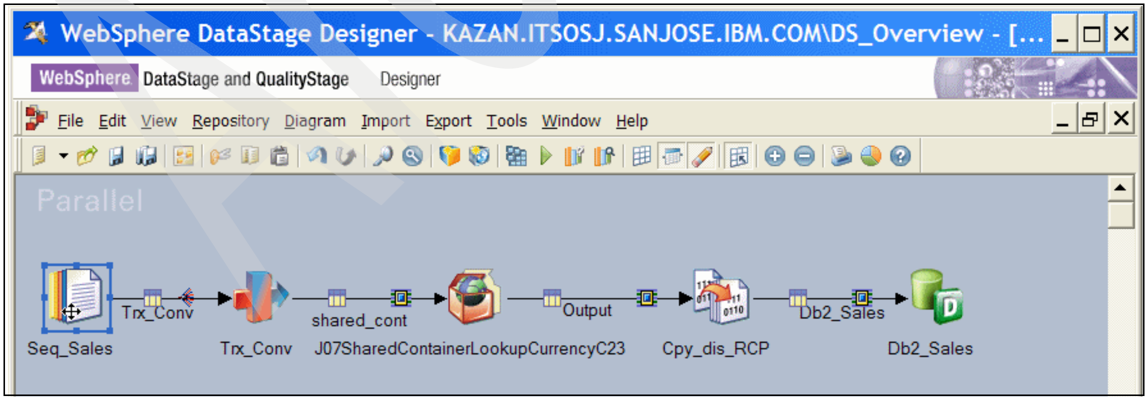


Figure 3-203 Create the J07_IL_Daily_LoadSalesStore job 1/18

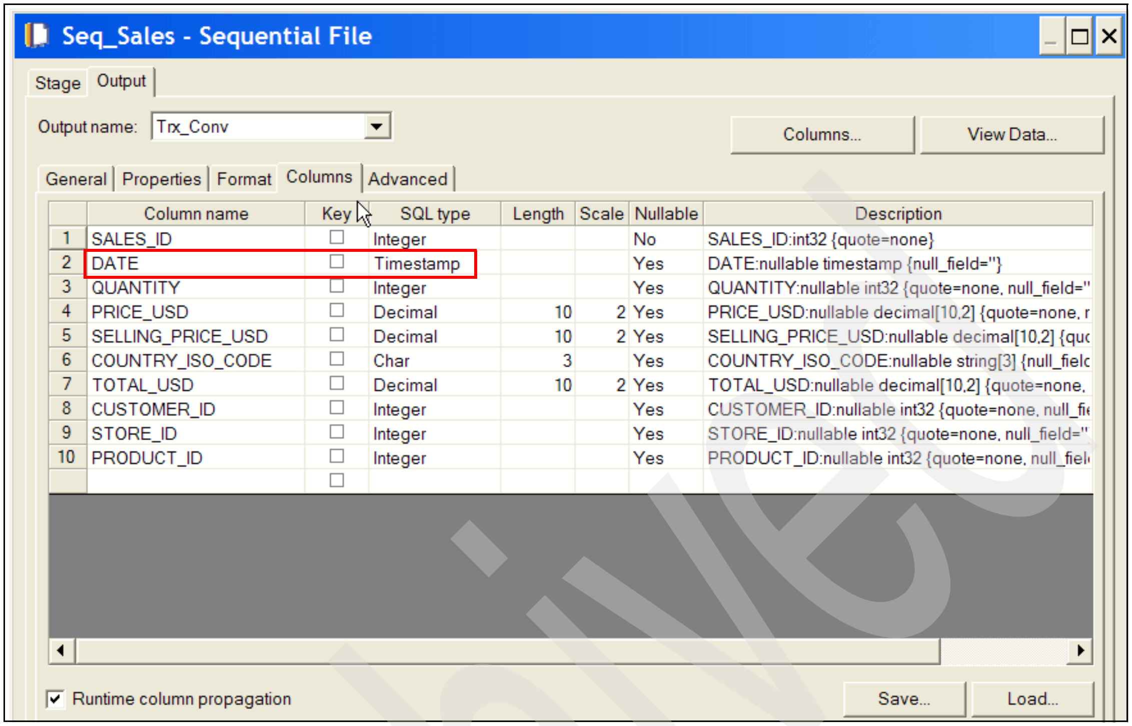


Figure 3-204 Create the J07_IL_Daily_LoadSalesStore job 2/18

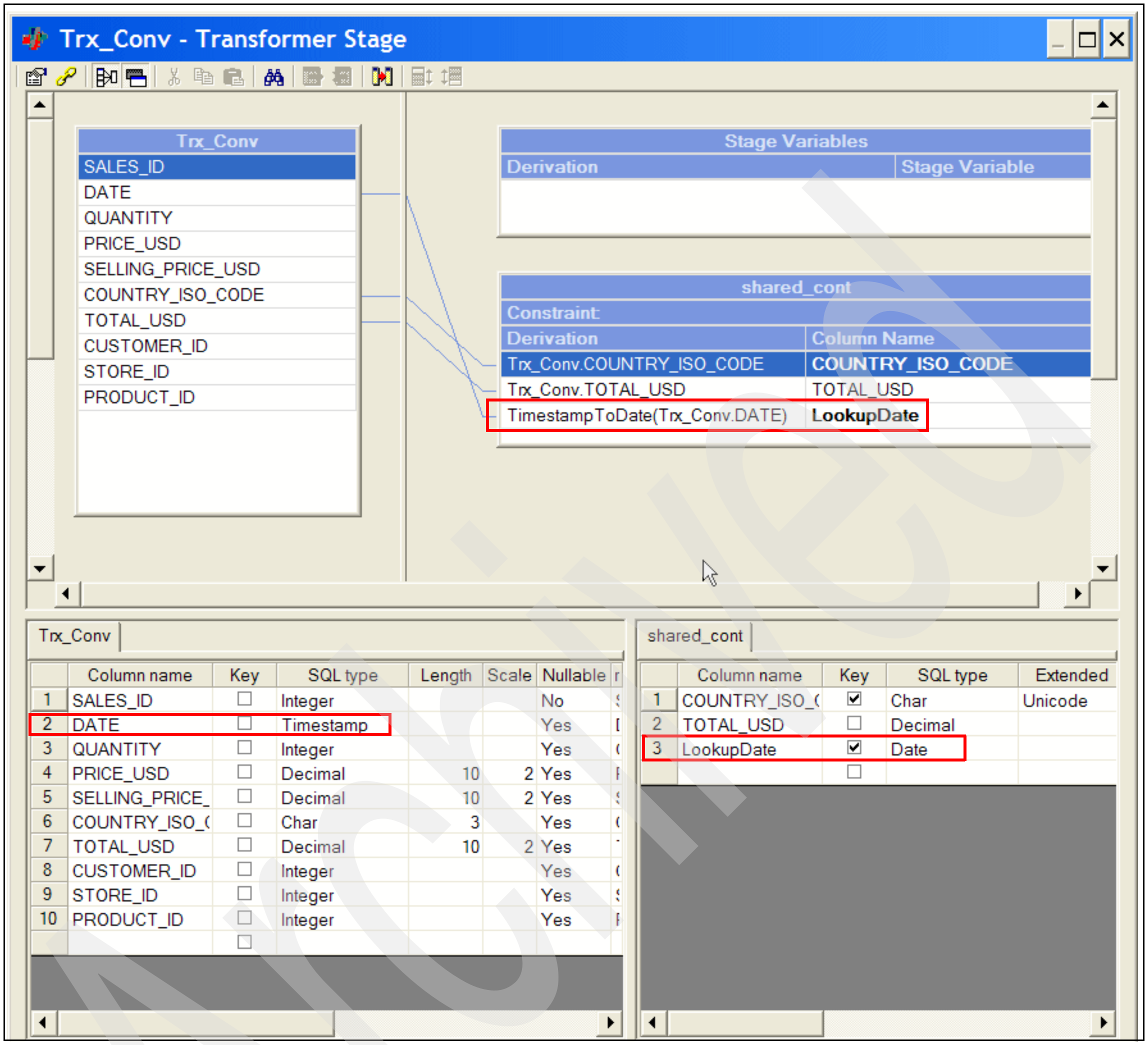


Figure 3-205 Create the J07_IL_Daily_LoadSalesStore job 3/18

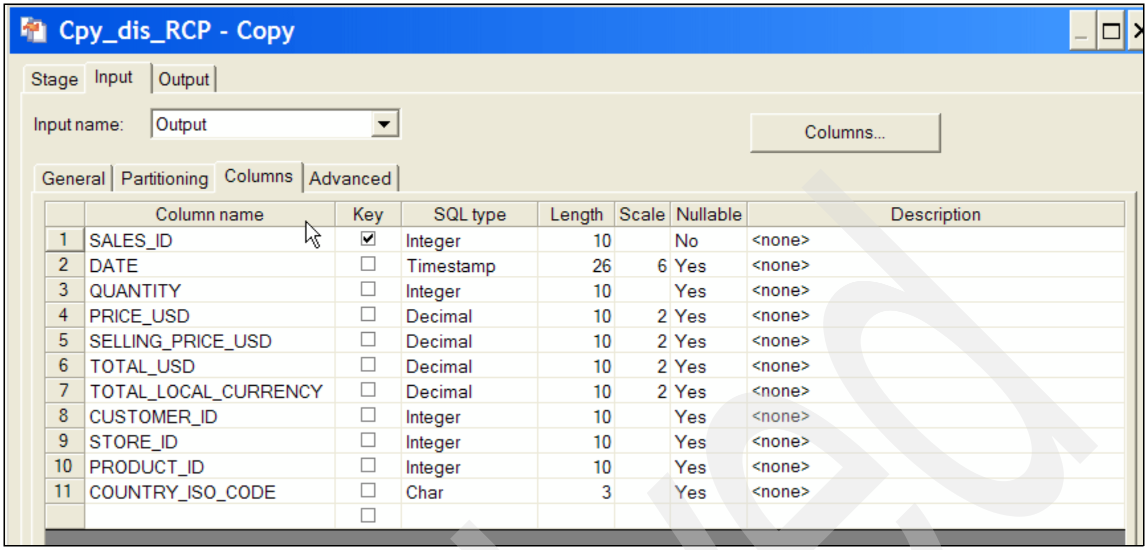


Figure 3-206 Create the J07_IL_Daily_LoadSalesStore job 4/18

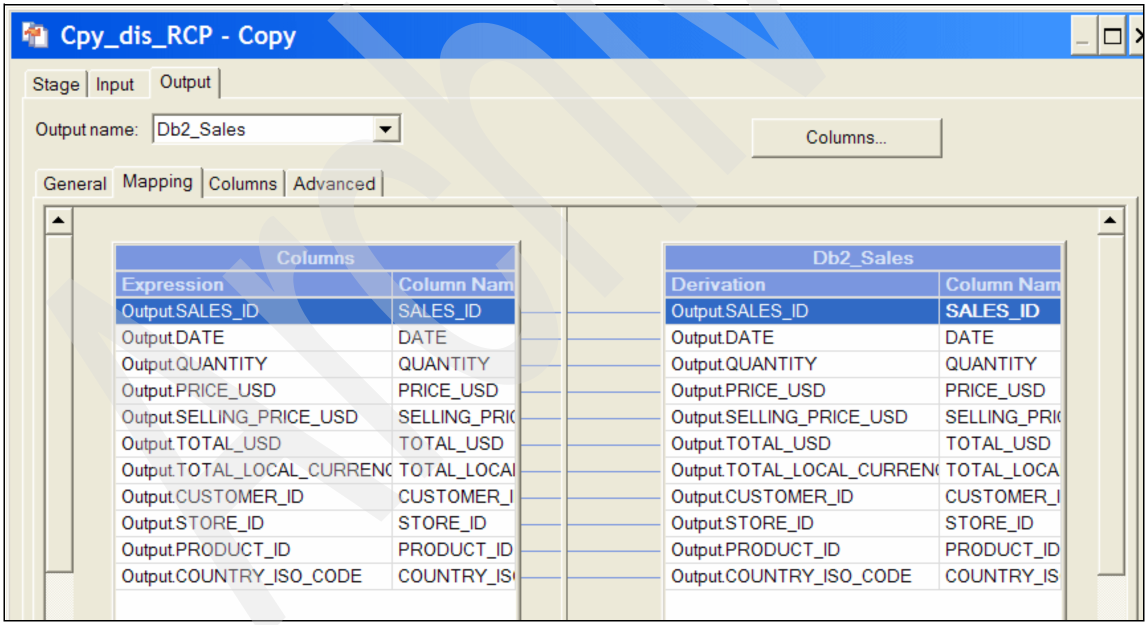


Figure 3-207 Create the J07_IL_Daily_LoadSalesStore job 5/18

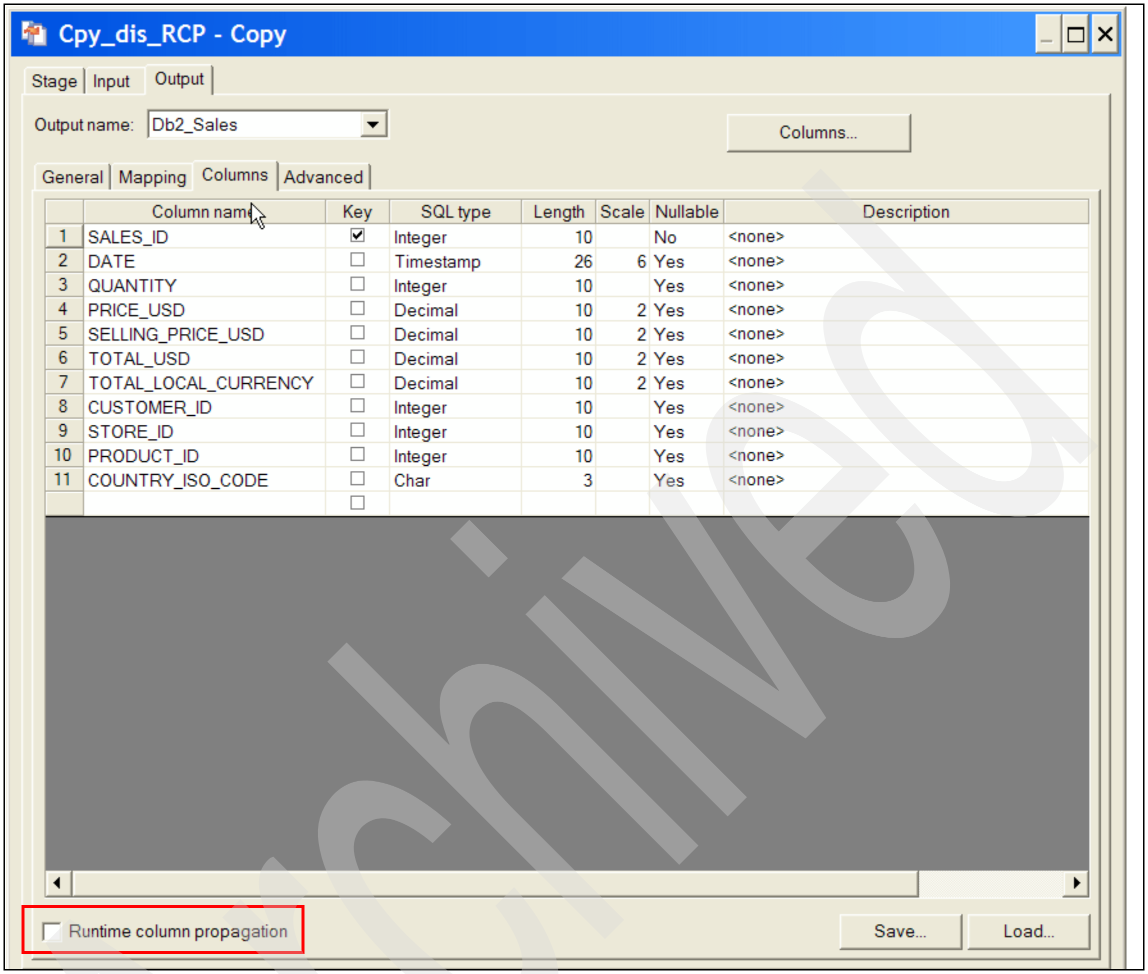


Figure 3-208 Create the J07_IL_Daily_LoadSalesStore job 6/18

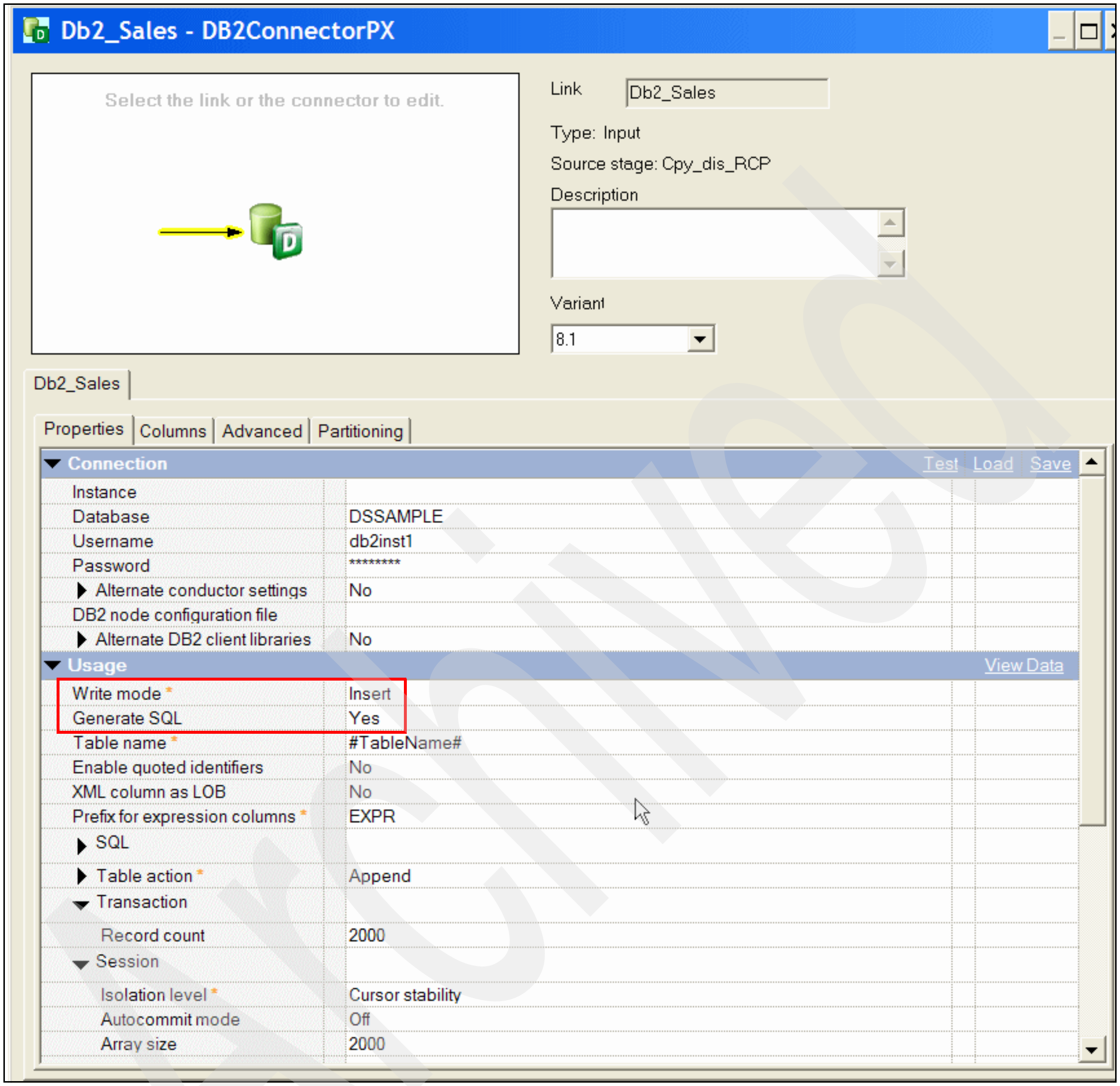


Figure 3-209 Create the J07_IL_Daily_LoadSalesStore job 7/18

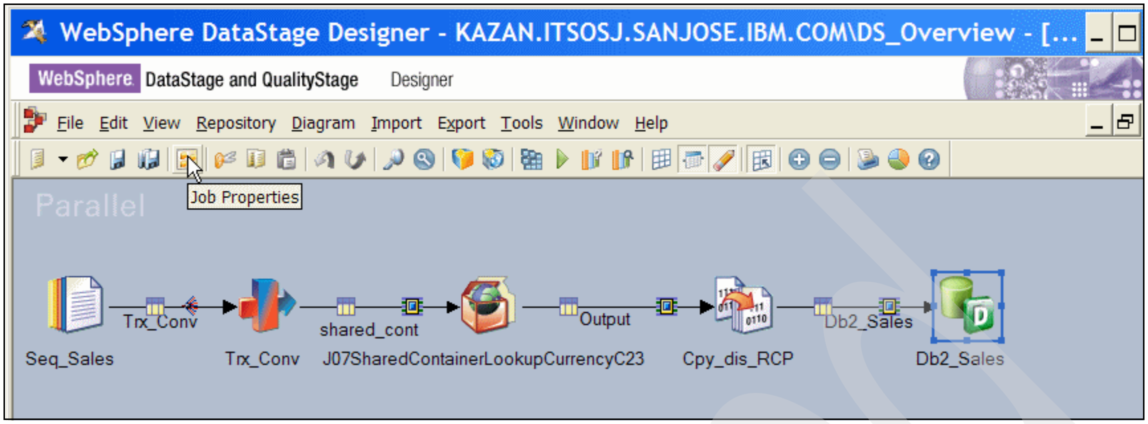


Figure 3-210 Create the J07_IL_Daily_LoadSalesStore job 8/18

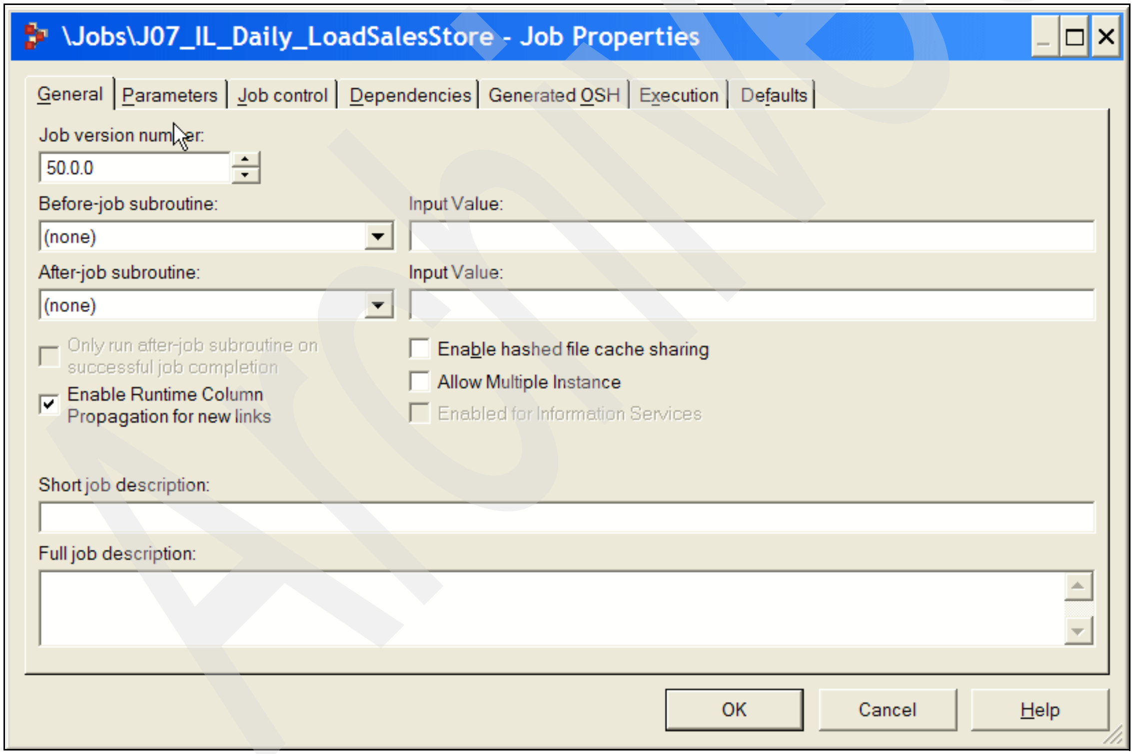


Figure 3-211 Create the J07_IL_Daily_LoadSalesStore job 9/18

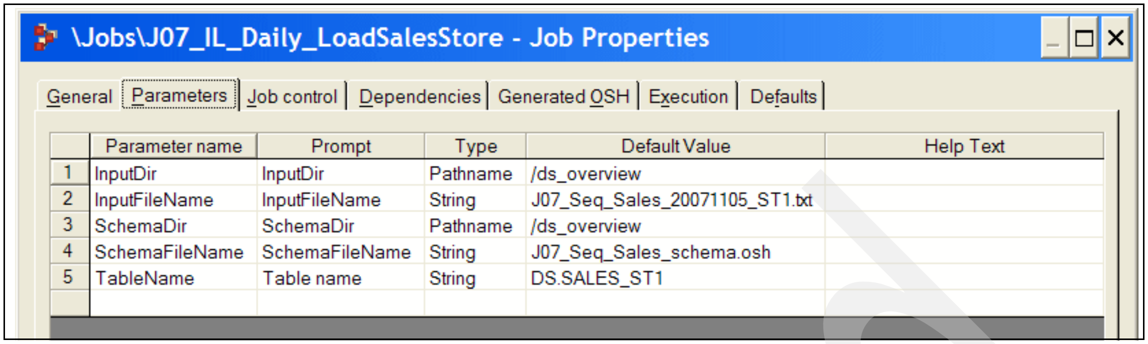


Figure 3-212 Create the J07_IL_Daily_LoadSalesStore job 10/18

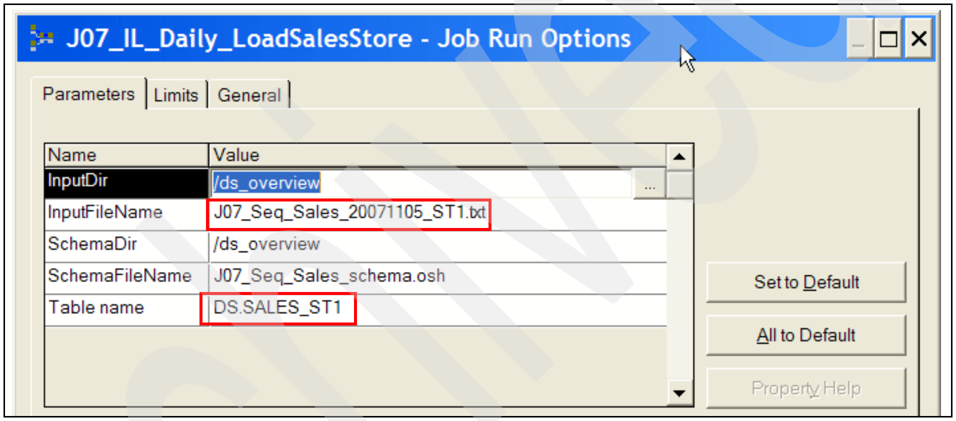


Figure 3-213 Create the J07_IL_Daily_LoadSalesStore job 11/18

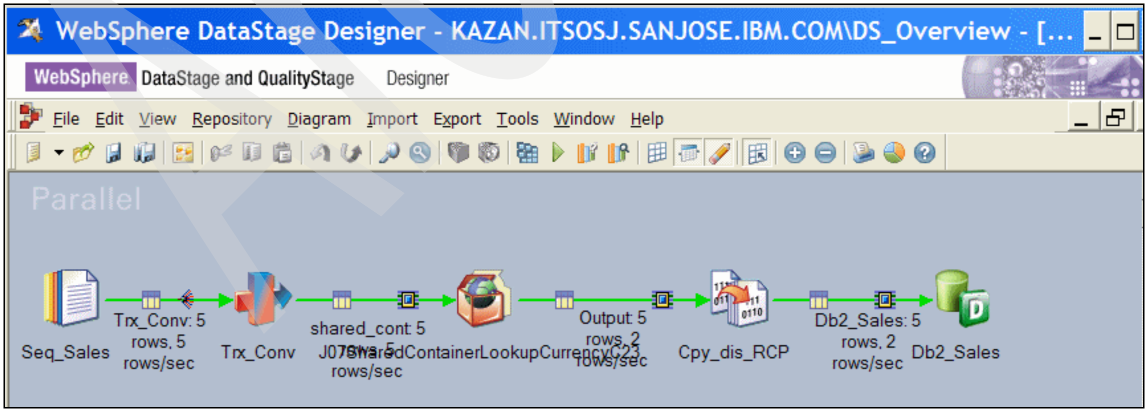


Figure 3-214 Create the J07_IL_Daily_LoadSalesStore job 12/18

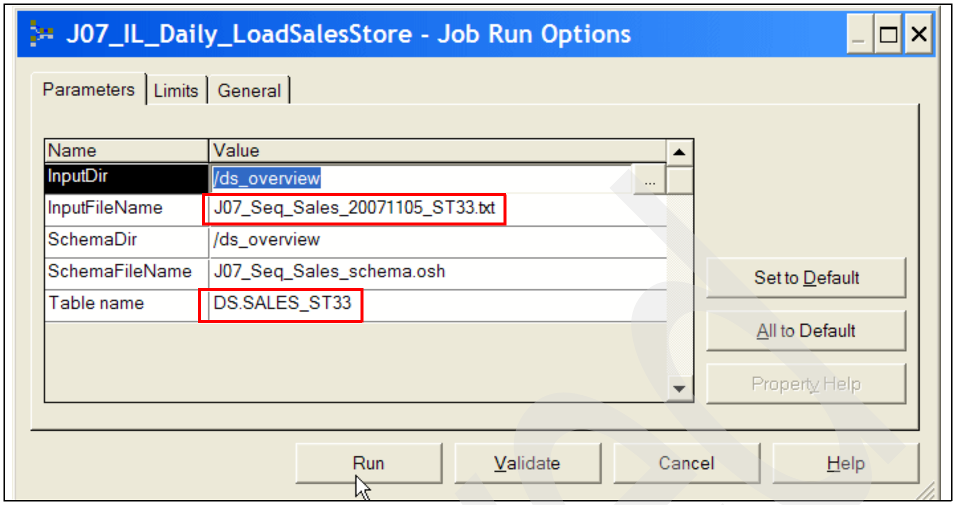


Figure 3-215 Create the J07_IL_Daily_LoadSalesStore job 13/18

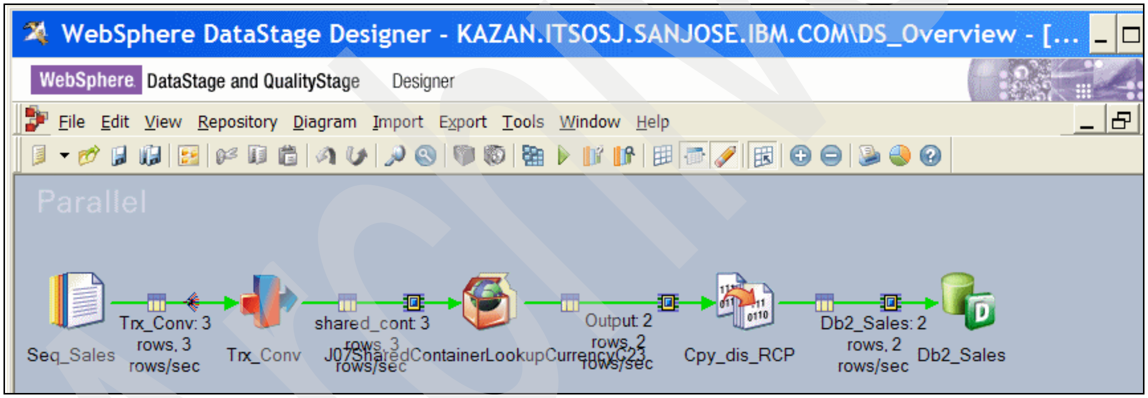
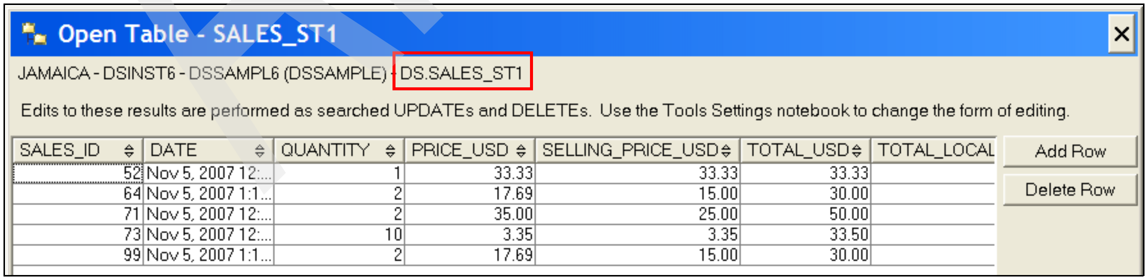
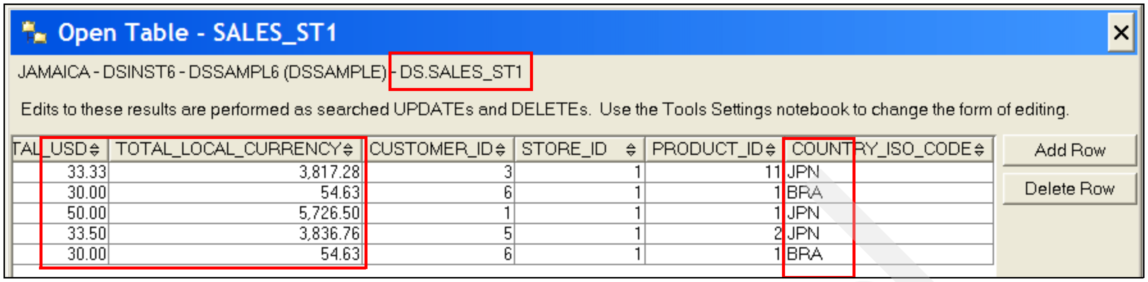


Figure 3-216 Create the J07_IL_Daily_LoadSalesStore job 14/18



SALES_ID	DATE	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	TOTAL_LOCAL
52	Nov 5, 2007 12:...	1	33.33	33.33	33.33	
64	Nov 5, 2007 1:1...	2	17.69	15.00	30.00	
71	Nov 5, 2007 12:...	2	35.00	25.00	50.00	
73	Nov 5, 2007 12:...	10	3.35	3.35	33.50	
99	Nov 5, 2007 1:1...	2	17.69	15.00	30.00	

Figure 3-217 Create the J07_IL_Daily_LoadSalesStore job 15/18



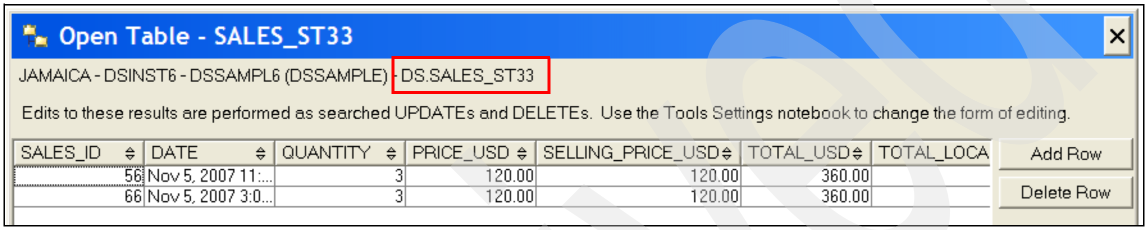
Open Table - SALES_ST1

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.SALES_ST1

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

TAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	COUNTRY_ISO_CODE
33.33	3,817.28	3	1	11	JPN
30.00	54.63	6	1	1	BRA
50.00	5,726.50	1	1	1	JPN
33.50	3,836.76	5	1	2	JPN
30.00	54.63	6	1	1	BRA

Figure 3-218 Create the J07_IL_Daily_LoadSalesStore job 16/18



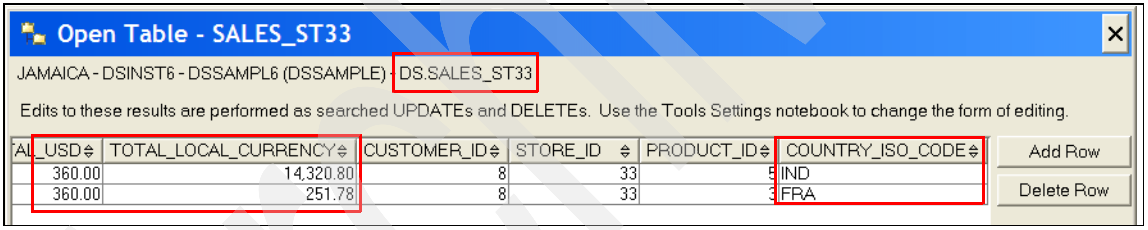
Open Table - SALES_ST33

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.SALES_ST33

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

SALES_ID	DATE	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	TOTAL_LOCA
56	Nov 5, 2007 11:...	3	120.00	120.00	360.00	
66	Nov 5, 2007 3:0...	3	120.00	120.00	360.00	

Figure 3-219 Create the J07_IL_Daily_LoadSalesStore job 17/18



Open Table - SALES_ST33

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.SALES_ST33

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

AL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	COUNTRY_ISO_CODE
360.00	14,320.80	8	33	5	IND
360.00	251.78	8	33	3	FRA

Figure 3-220 Create the J07_IL_Daily_LoadSalesStore job 18/18

J08_IL_LoadSalesFact

In this job, all the sales transactions (in the interim DB2 tables) from the various stores are merged, aggregated, and assigned the appropriate surrogate key (corresponding to the business key) before being loaded into the Sales fact table.

The Sales fact table does not contain the raw sales transactions, but aggregated summaries of the sales transactions. Figure 3-221 on page 295 through Figure 3-254 on page 320 describe the main steps in processing the sales transactions prior to loading the Sales fact table.

1. Figure 3-221 on page 295 shows the various stages in the job — it includes seven ODBCConnectorPX stages, a Funnel stage, a Modify stage, an Aggregator stage, a Lookup stage, a Filter stage, and a Sequential file stage. The names of the stages were modified as shown.

2. Figure 3-222 on page 296 shows an ODBCConnectorPX stage that retrieves sales transactions from the interim DB2 table corresponding to the ST1 store, while Figure 3-223 on page 297 shows the corresponding ODBCConnectorPX stage that retrieves sales transactions from the interim DB2 table corresponding to the ST33 store. The SQL to access these tables are generated automatically. The rows from these two tables are then unioned using a Funnel stage (Fnl_Sales).
3. Figure 3-224 on page 298 and Figure 3-225 on page 298 show the configuration of the Funnel stage including the mapping of columns in the output.
4. In the output of the Funnel stage, the DATE column is a TIMESTAMP data type. In order to aggregate the sales transactions on multiple columns including the date, we first have to create a Modify stage that converts the TIMESTAMP data type to a DATE for all the sales transactions. This is shown in Figure 3-226 on page 299.
5. After the conversion of the date columns in the sales transactions in the Modify stage as shown in Figure 3-226 on page 299, we can aggregate the sales transactions' QUANTITY (number of units of the product sold), TOTAL_USD (total cost of the units in $US) and TOTAL_LOCAL_CURRENCY (equivalent total cost of the units in the foreign currency) columns based on the grouping columns CUSTOMER_ID, PRODUCT_ID, STORE_ID, DATE, COUNTRY_ISO_CODE, PRICE_USD, and SELLING_PRICE_USD.
   - Figure 3-227 on page 299 shows the **Properties** tab in the Stage page, which identifies the Grouping Keys, and the Aggregations details such as the sum calculation.
   - Figure 3-228 on page 300 shows the **Mapping** tab in the Output page that identifies the columns mapped to the output Lku_Dim link. It includes the grouping columns as well as the aggregated columns.
6. Figure 3-229 on page 301 through Figure 3-240 on page 312 show the configuration of the Lookup stage. For each record of the source data set from the primary link (Lku_Dim), the Lookup stage performs a table lookup on the four lookup tables attached by reference links (Odbc_Customer, Odbc_Product, Odbc_Store, and Odbc_Date).
   - Figure 3-229 on page 301 through Figure 3-235 on page 307 identify the access to each of the four reference links using the ODBCConnectorPX stage using manually generated SQL SELECT statements that retrieve all the business key and surrogate key pairs.
   - The table lookups are based on the values of a set of lookup key columns as identified in Figure 3-236 on page 308 through Figure 3-240 on page 312. You can specify a condition on each reference link such that the

stage will only perform a lookup on that reference link if the condition is satisfied. The equality condition is used here as shown in Figure 3-236 on page 308 through Figure 3-240 on page 312.

- Each record of the output link (filter) contains columns from the source plus columns from all the corresponding lookup records where the corresponding source and lookup records have the same value for the lookup key columns. The lookup key columns do not have to have the same names in the primary and the reference links. This is shown in Figure 3-241 on page 313.
- Figure 3-242 on page 314 shows the **Link Ordering** tab in the Stage page, which identifies the Primary (link) as being the Lku_Dim link and the Lookups (Reference links) as being the Odbc_Customer, Odbc_Product, Odbc_Store, and Odbc_Date.

  We chose not to define the optional reject link for this stage.

7. The output of the Lookup stage is then input to a Filter stage to only accept records that have a non-zero value in the surrogate keys (Figure 3-243 on page 314 and Figure 3-244 on page 315) and write them out to the output link Odbc_Fact, and write the rejects (those that do not qualify per the predicate) to the Seq_reject link.

   Figure 3-245 on page 315 shows the **Link Ordering** tab in the Stage page that directs the records that qualify to the Odbc_Fact link, while the rejects are directed to the Seq_reject link.

   Figure 3-246 on page 316 shows the **Mapping** tab in the Output page that copies all columns from the input to the output.

   Figure 3-247 on page 316 (**Properties** tab in the Input page) and Figure 3-248 on page 317 (**Format** tab in the Input page) show the configuration of the sequential file containing the reject records
8. Figure 3-249 on page 318 shows the ODBCConnectorPX stage that is used to insert the sales transactions into the SALES_FACT table. The SQL INSERT statement is automatically generated. The Write mode is Delete then insert to ensure that no insert failures can occur.
9. Figure 3-250 on page 319 shows the results of executing this job. Three rows are inserted into the SALES_FACT table, while three rows are written to the reject file.

   Figure 3-251 on page 319 and Figure 3-252 on page 319 show the rows rejected because of at least one of the dimension keys has a zero value.

   Figure 3-253 on page 320 and Figure 3-254 on page 320 show the rows successfully inserted into the SALES_FACT table.

This concludes the initial load of the sales fact table and the dimension tables.

Before you can commence the recurring tasks (update of the sales fact table with sales transactions, and the update of the dimension tables with new business keys or changes to attributes), you have to create interim lookup dimension tables and surrogate key files the dimension tables as described in "J09_IL_LoadLookupCustomerDim" on page 320, "J10_IL_LoadLookupProductDim" on page 327, "J11_IL_LoadLookupStoreDim" on page 330, and "J12_IL_GenerateSurrogateKey" on page 335.

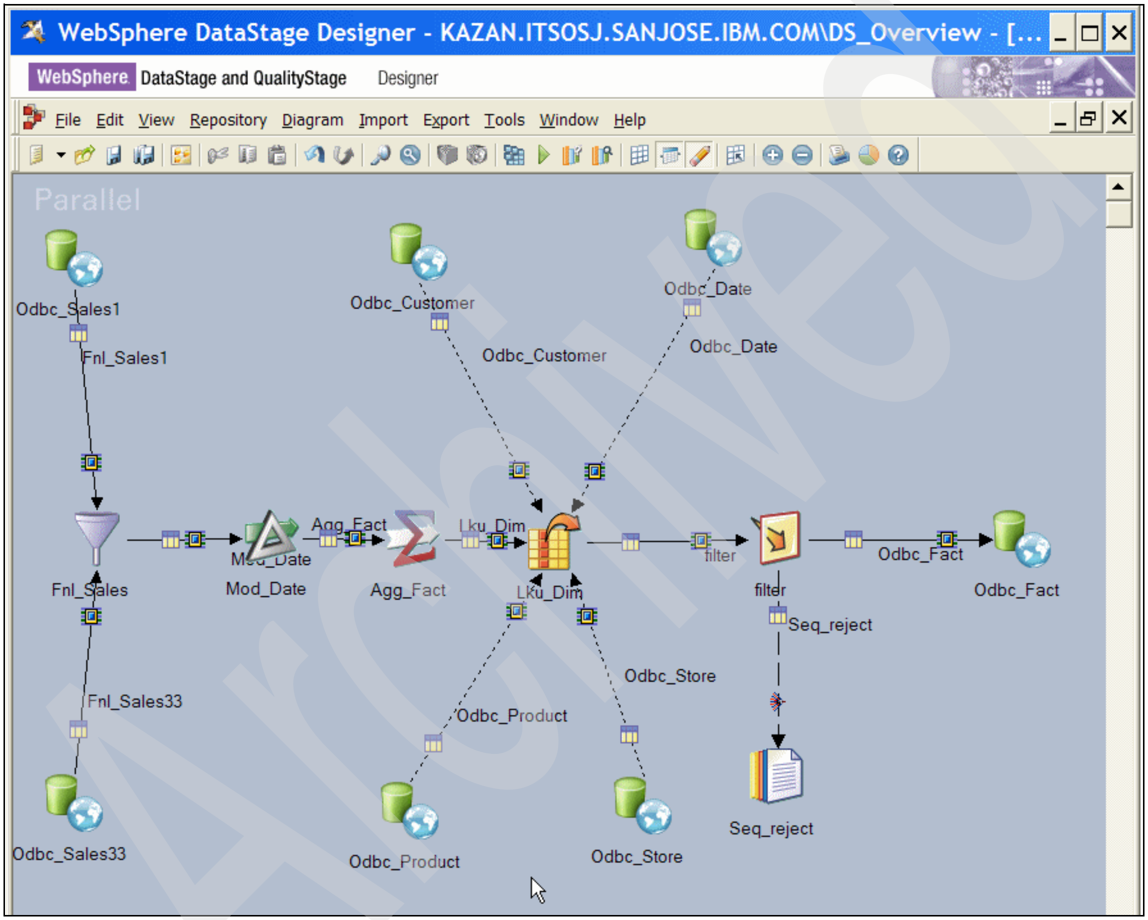


Figure 3-221 Create the J08_IL_LoadSalesFact job 1/34

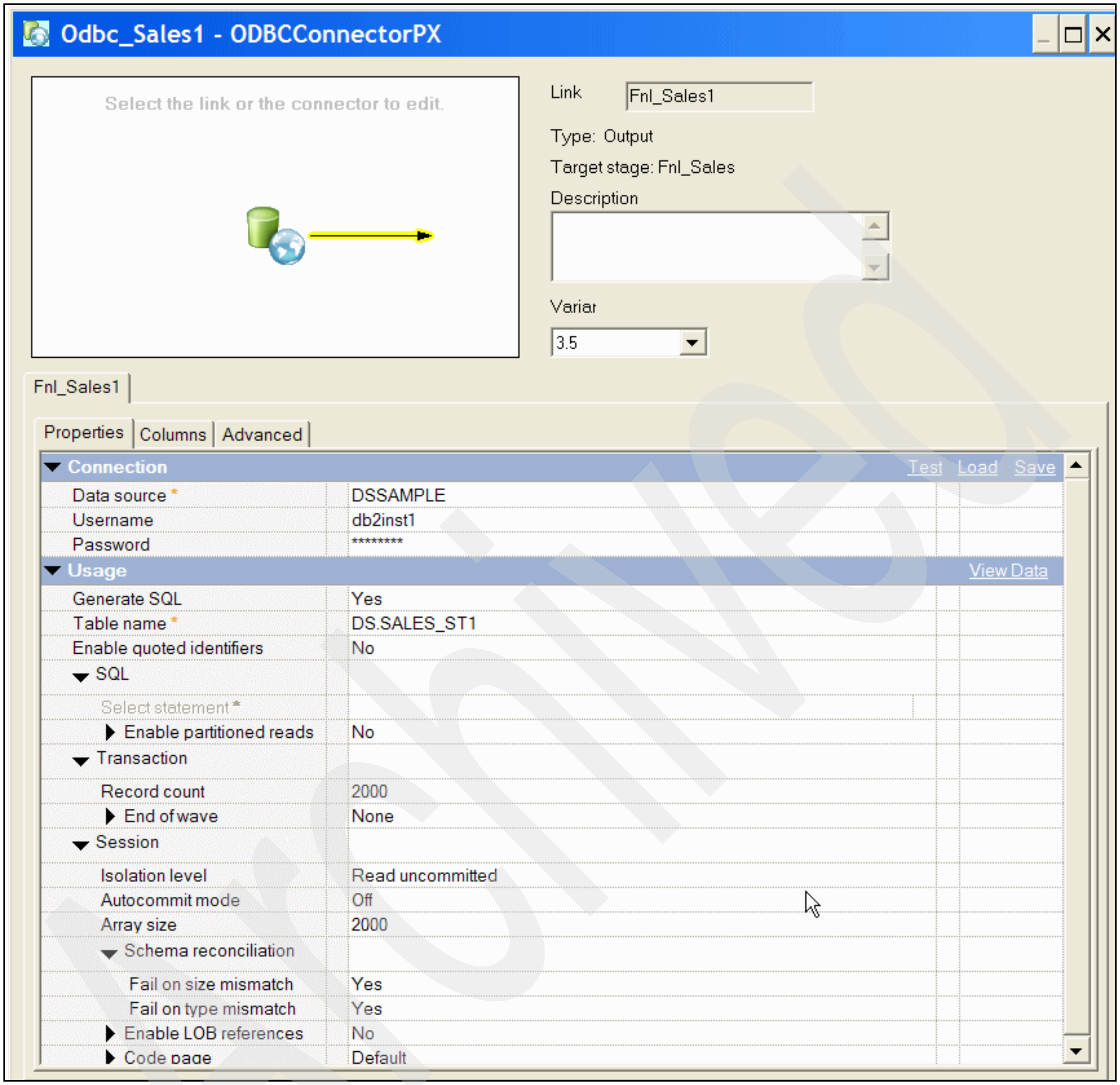


Figure 3-222 Create the J08_IL_LoadSalesFact job 2/34

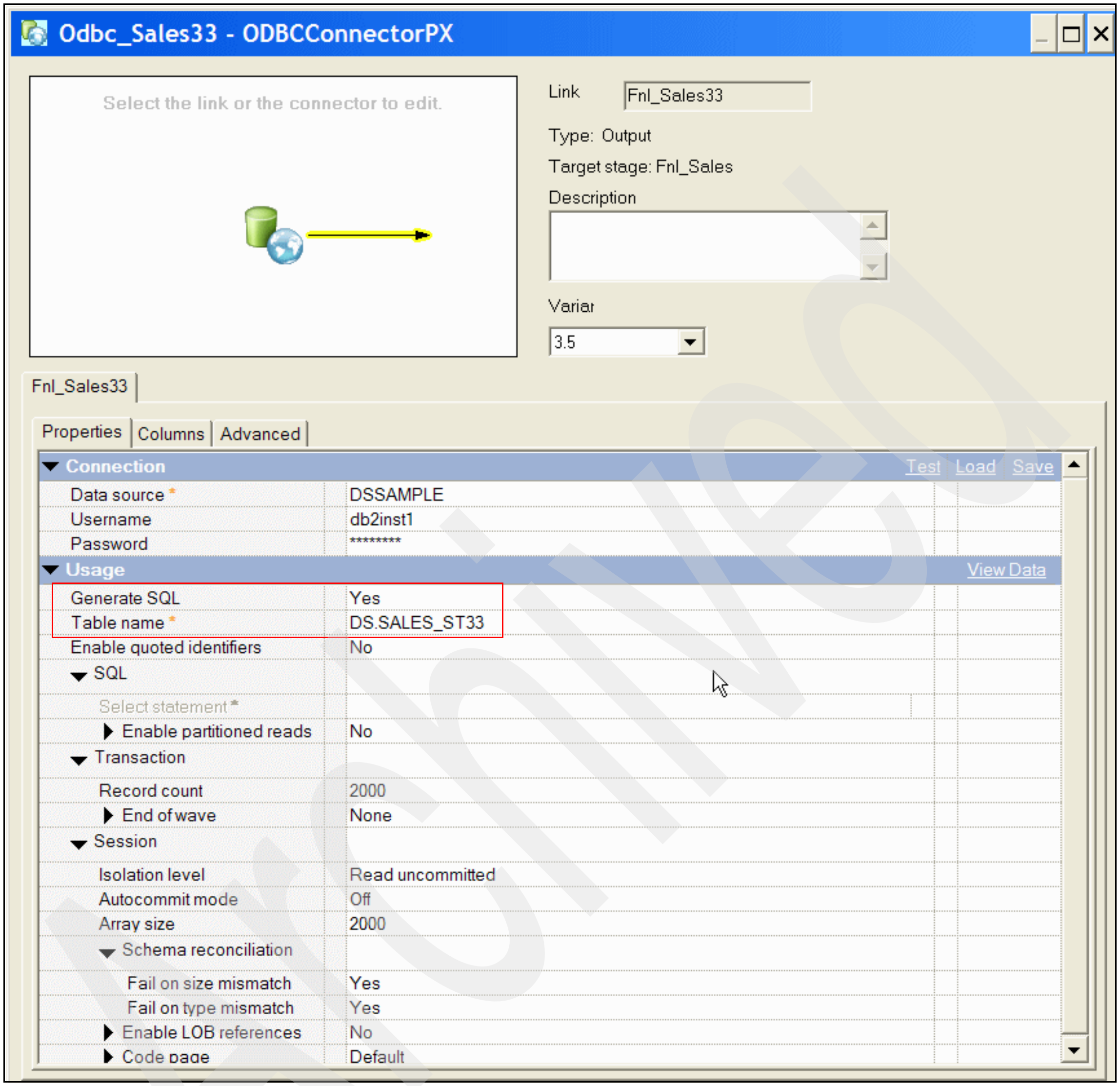


Figure 3-223 Create the J08_IL_LoadSalesFact job 3/34

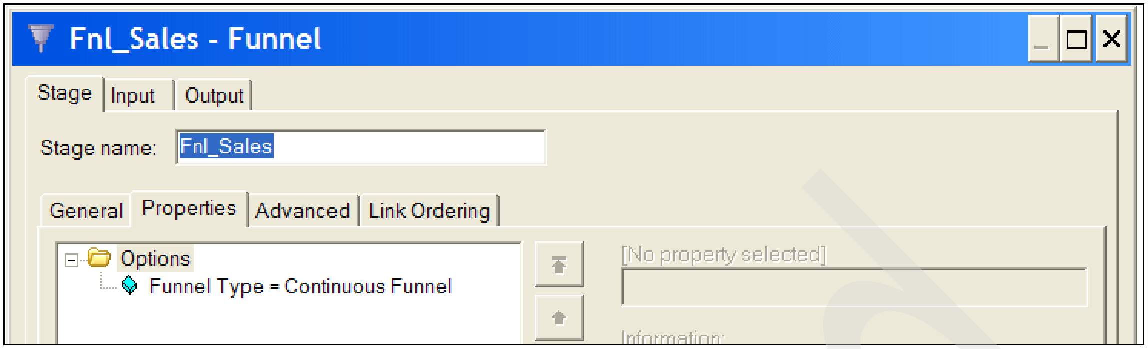


Figure 3-224 Create the J08_IL_LoadSalesFact job 4/34

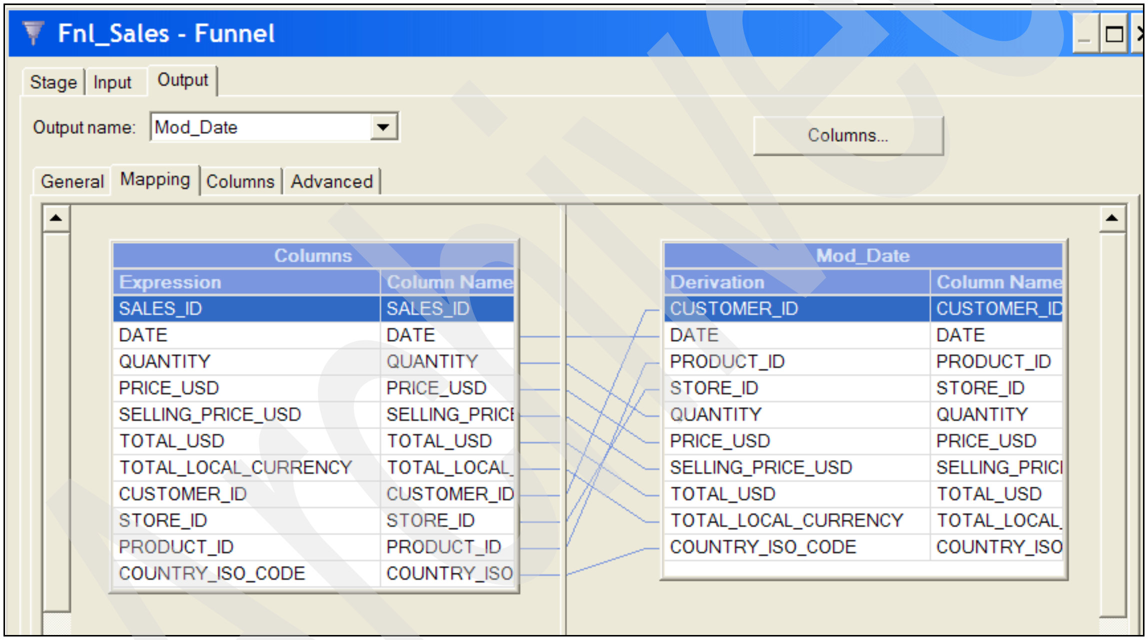


Figure 3-225 Create the J08_IL_LoadSalesFact job 5/34

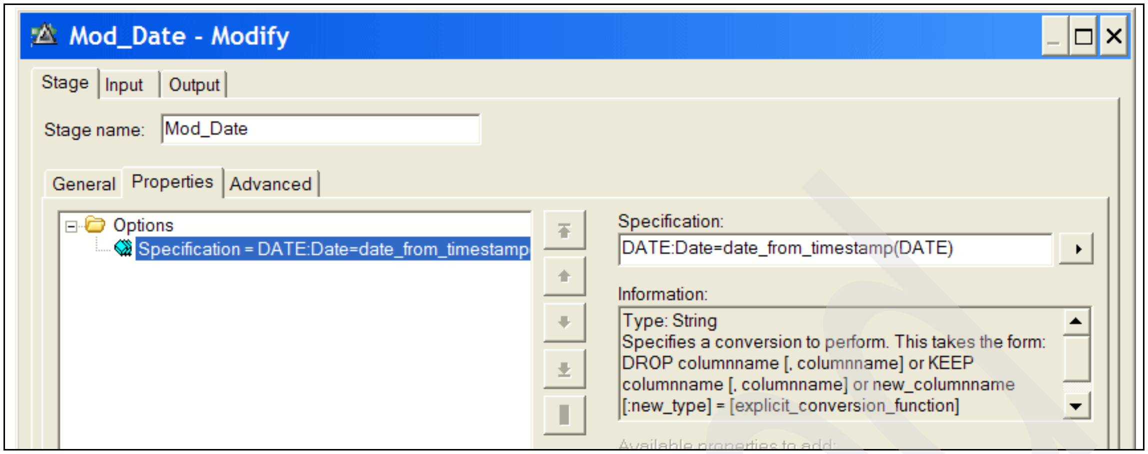


Figure 3-226 Create the J08_IL_LoadSalesFact job 6/34

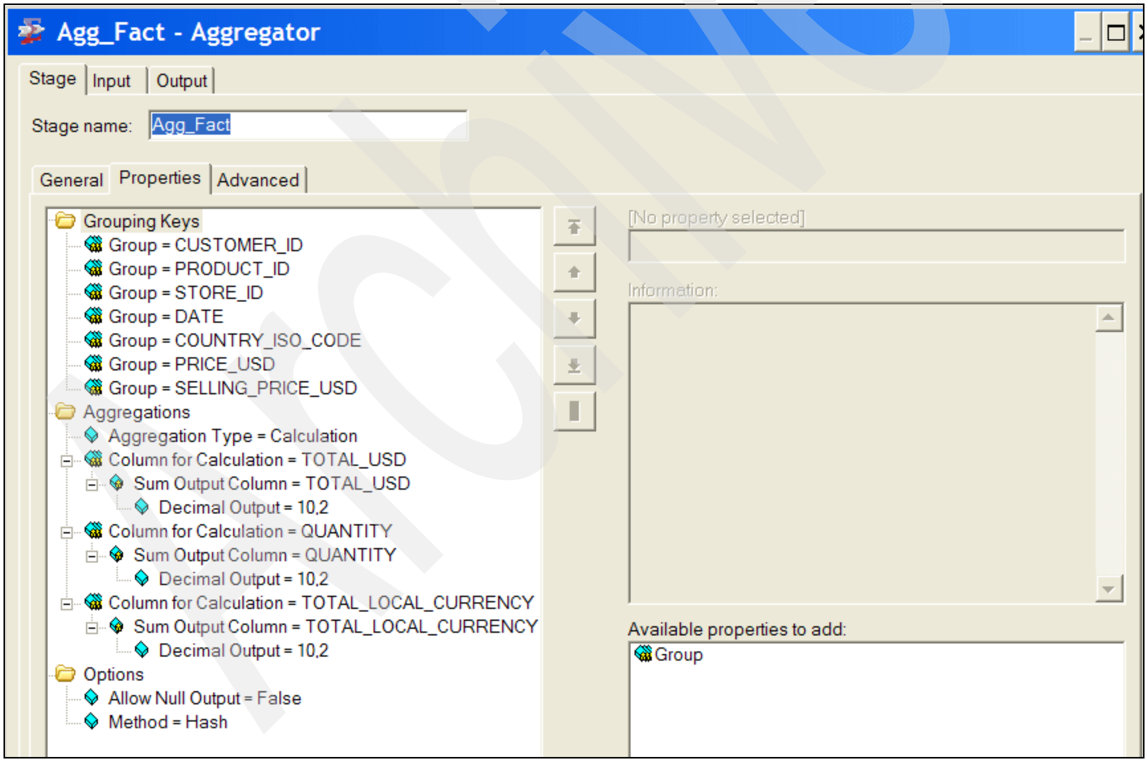


Figure 3-227 Create the J08_IL_LoadSalesFact job 7/34

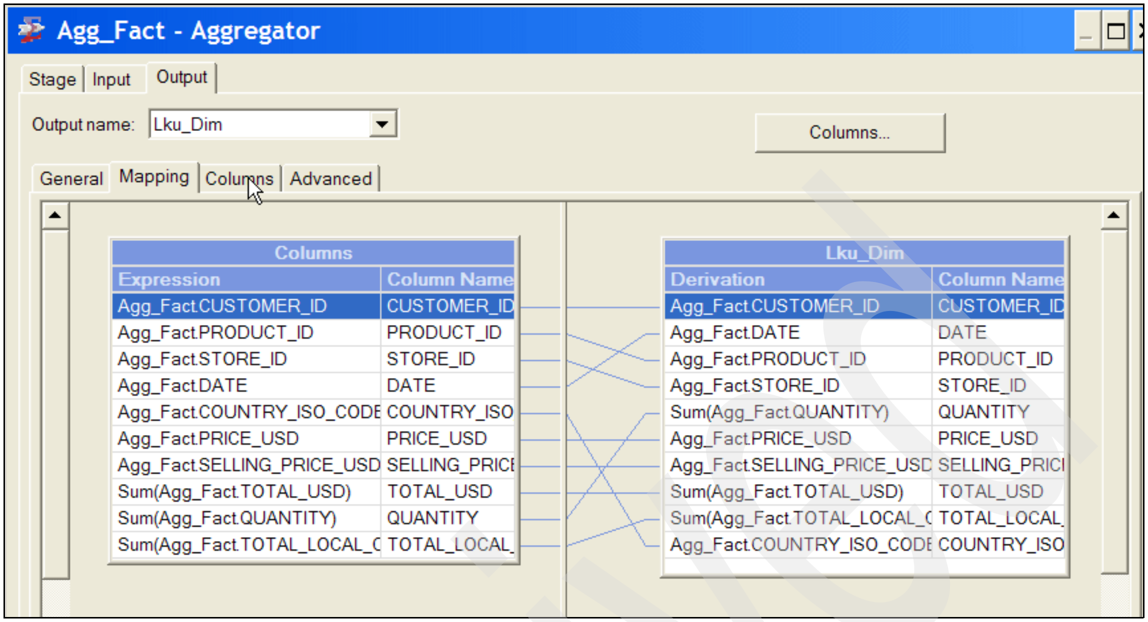


Figure 3-228 Create the J08_IL_LoadSalesFact job 8/34

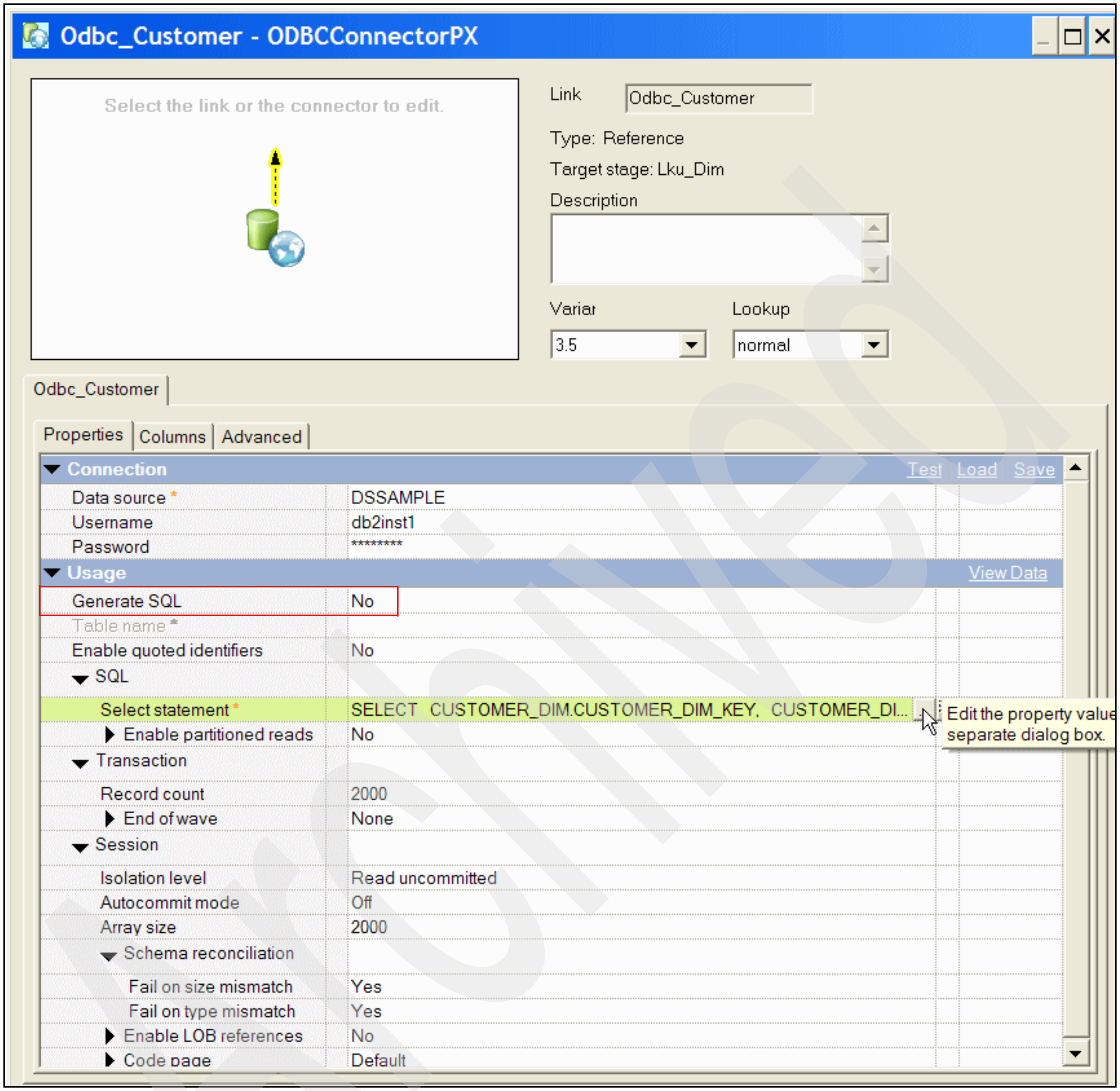


Figure 3-229 Create the J08_IL_LoadSalesFact job 9/34

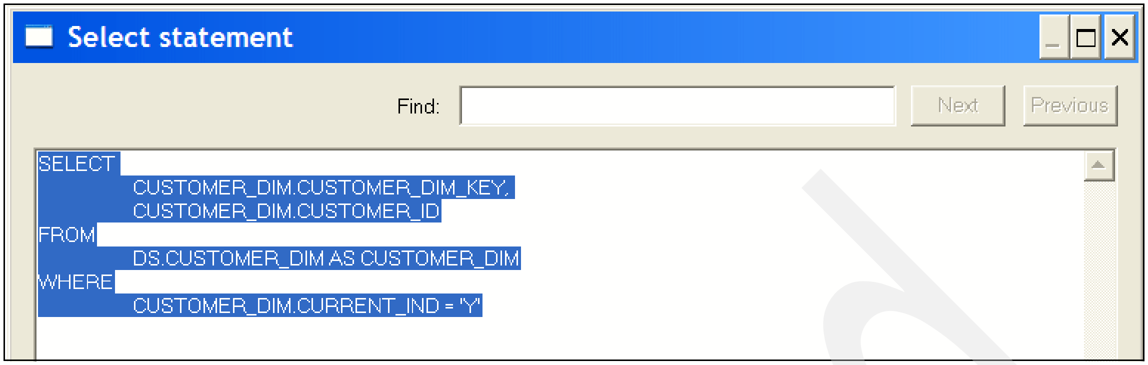


Figure 3-230 Create the J08_IL_LoadSalesFact job 10/34

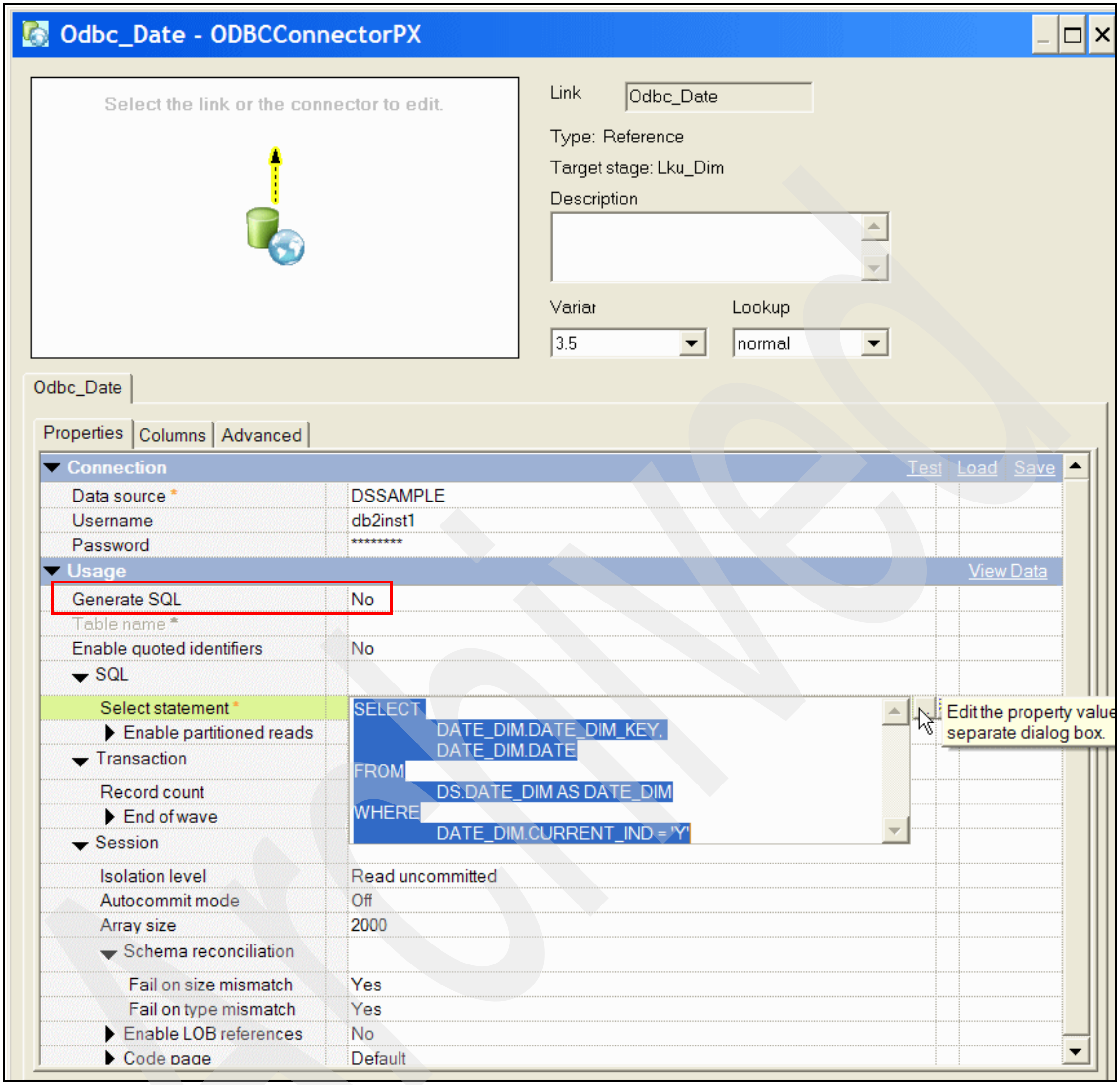


Figure 3-231 Create the J08_IL_LoadSalesFact job 11/34

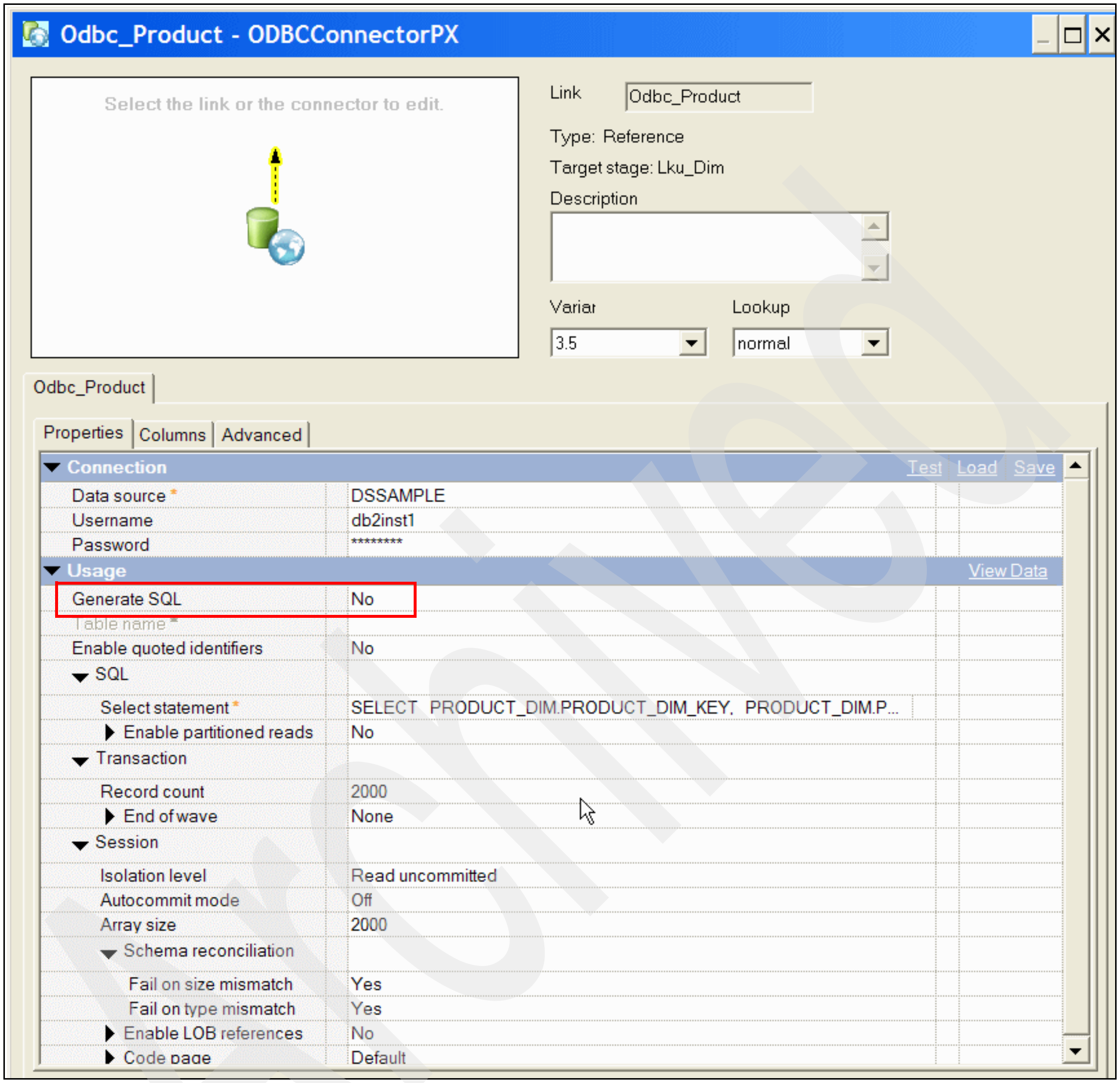


Figure 3-232 Create the J08_IL_LoadSalesFact job 12/34

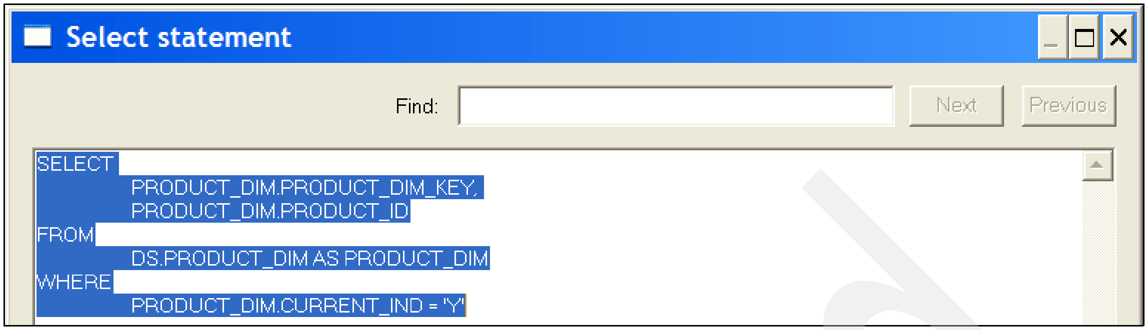


Figure 3-233 Create the J08_IL_LoadSalesFact job 13/34

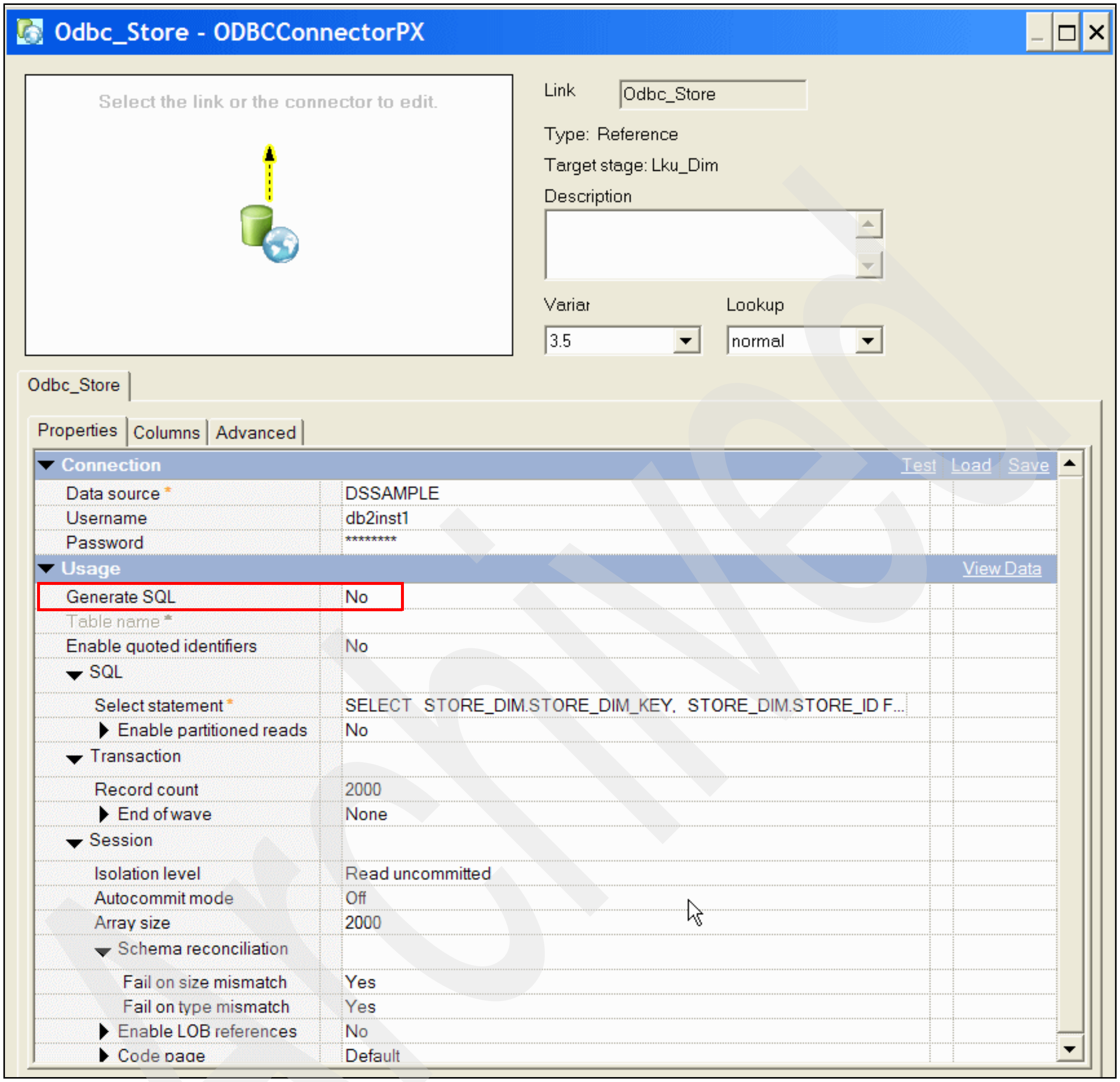


Figure 3-234 Create the J08_IL_LoadSalesFact job 14/34

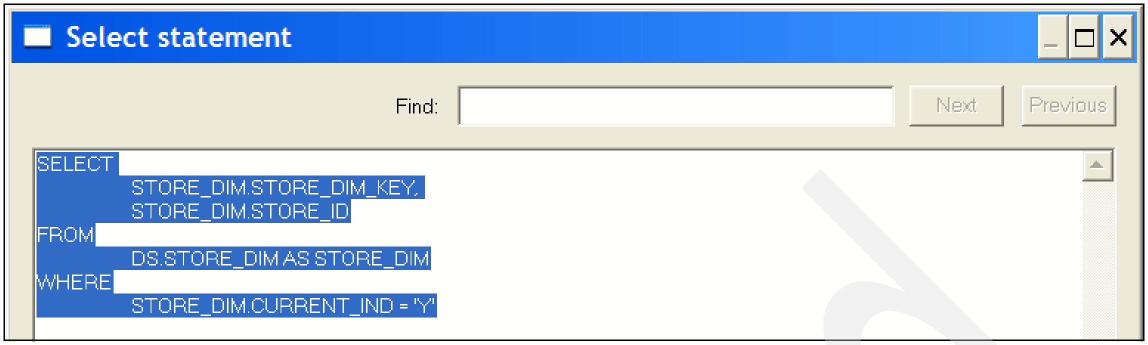


Figure 3-235 Create the J08_IL_LoadSalesFact job 15/34

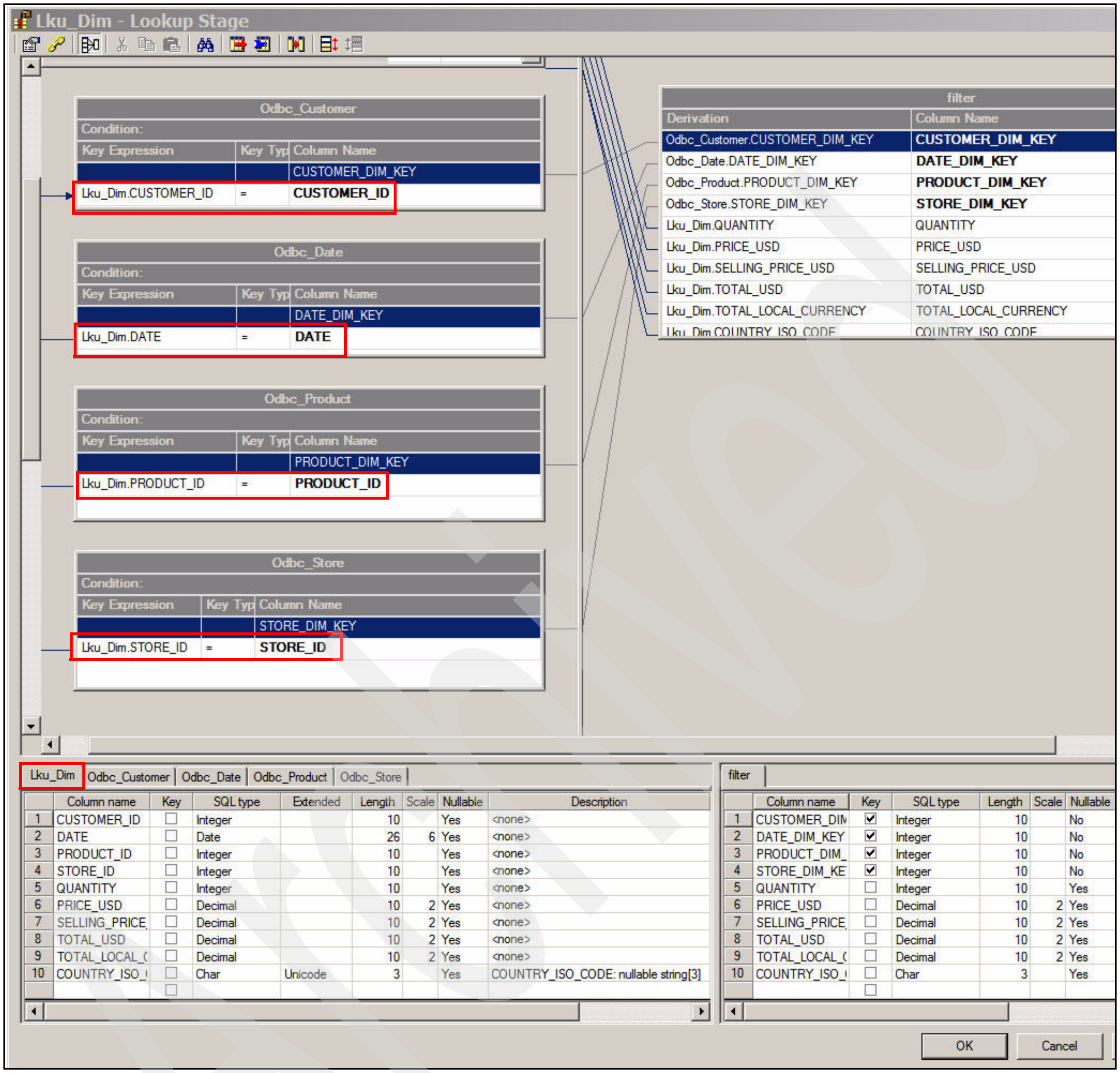


Figure 3-236 Create the J08_IL_LoadSalesFact job 16/34

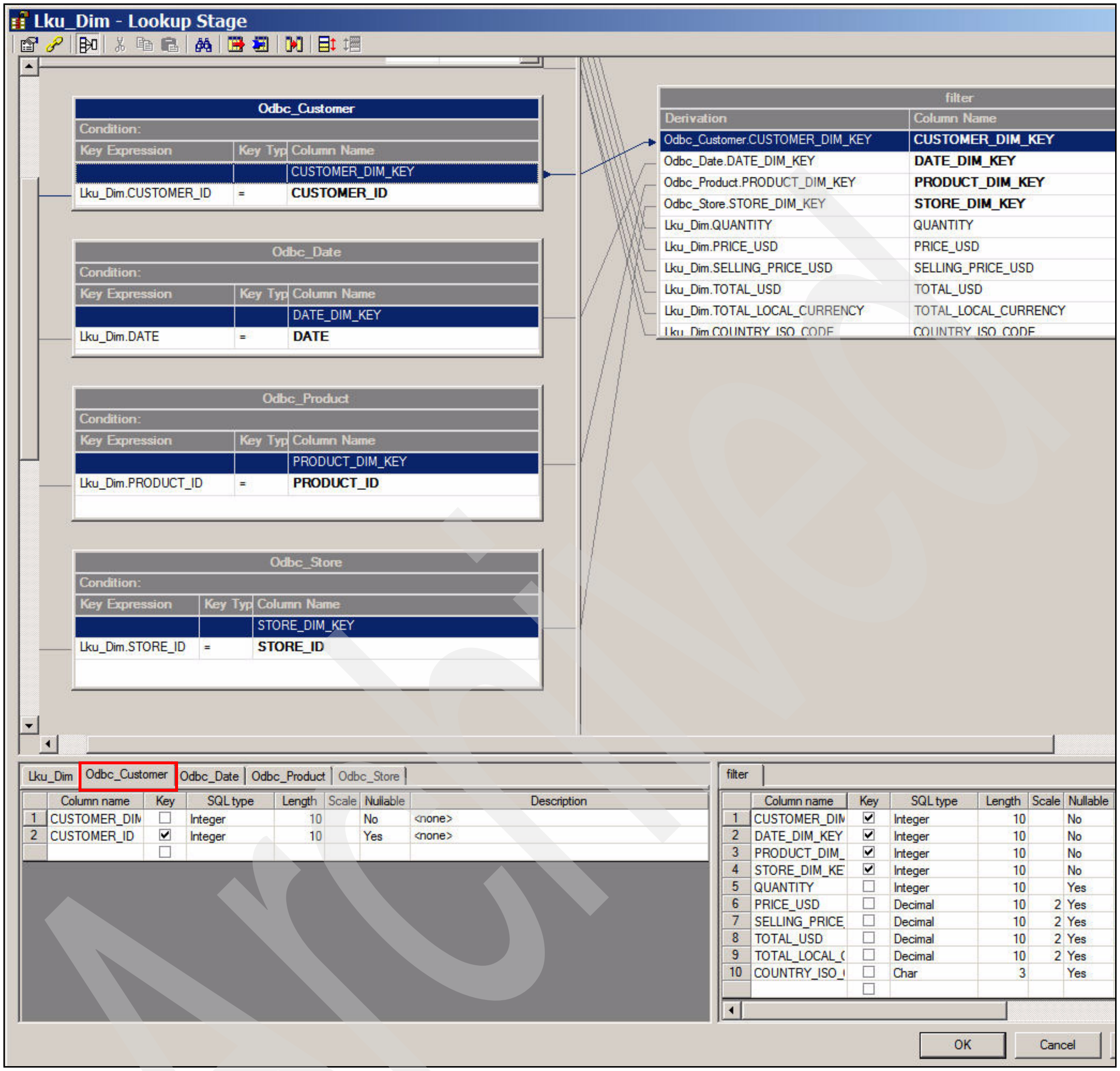


Figure 3-237 Create the J08_IL_LoadSalesFact job 17/34

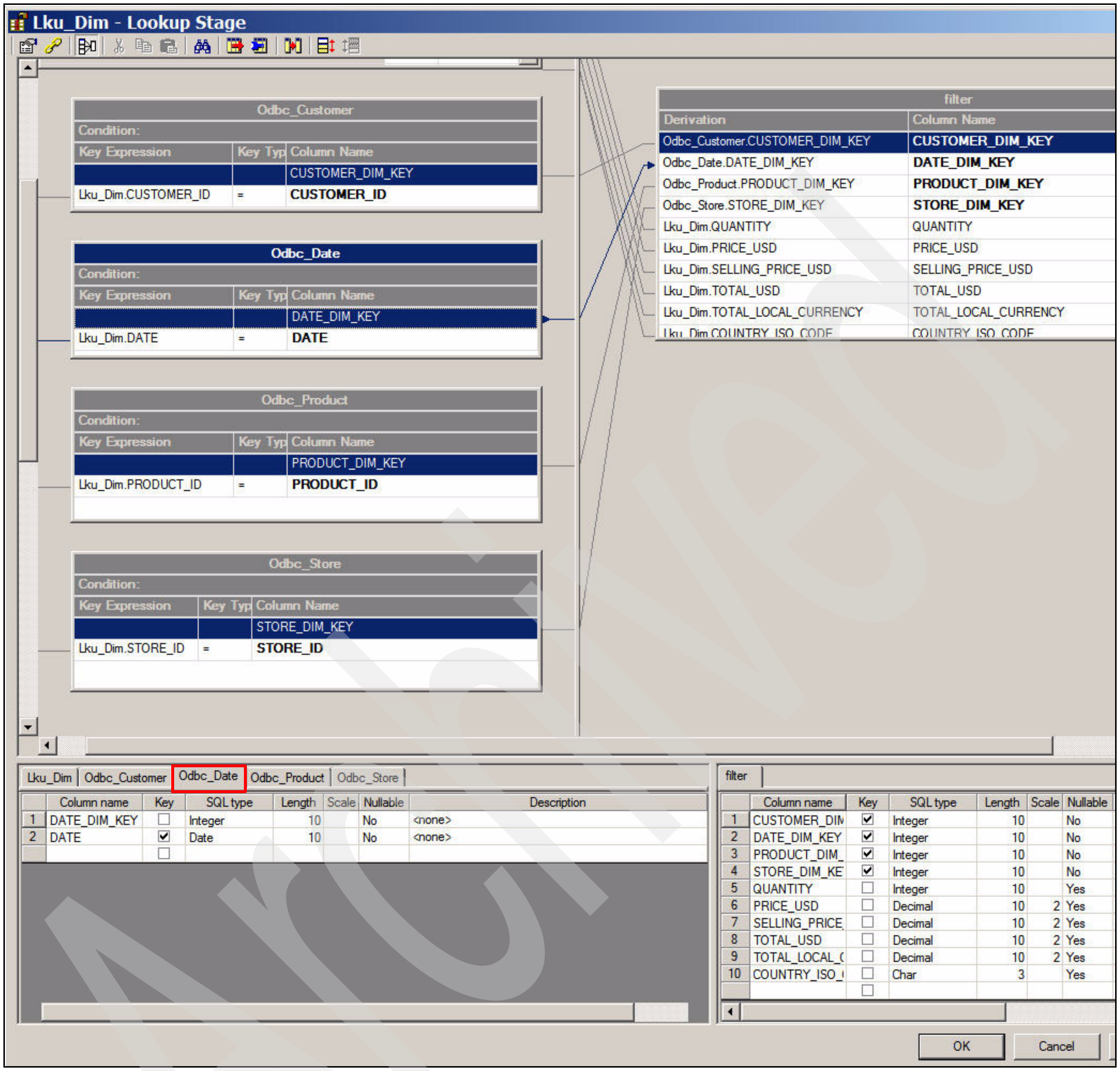


Figure 3-238 Create the J08_IL_LoadSalesFact job 18/34

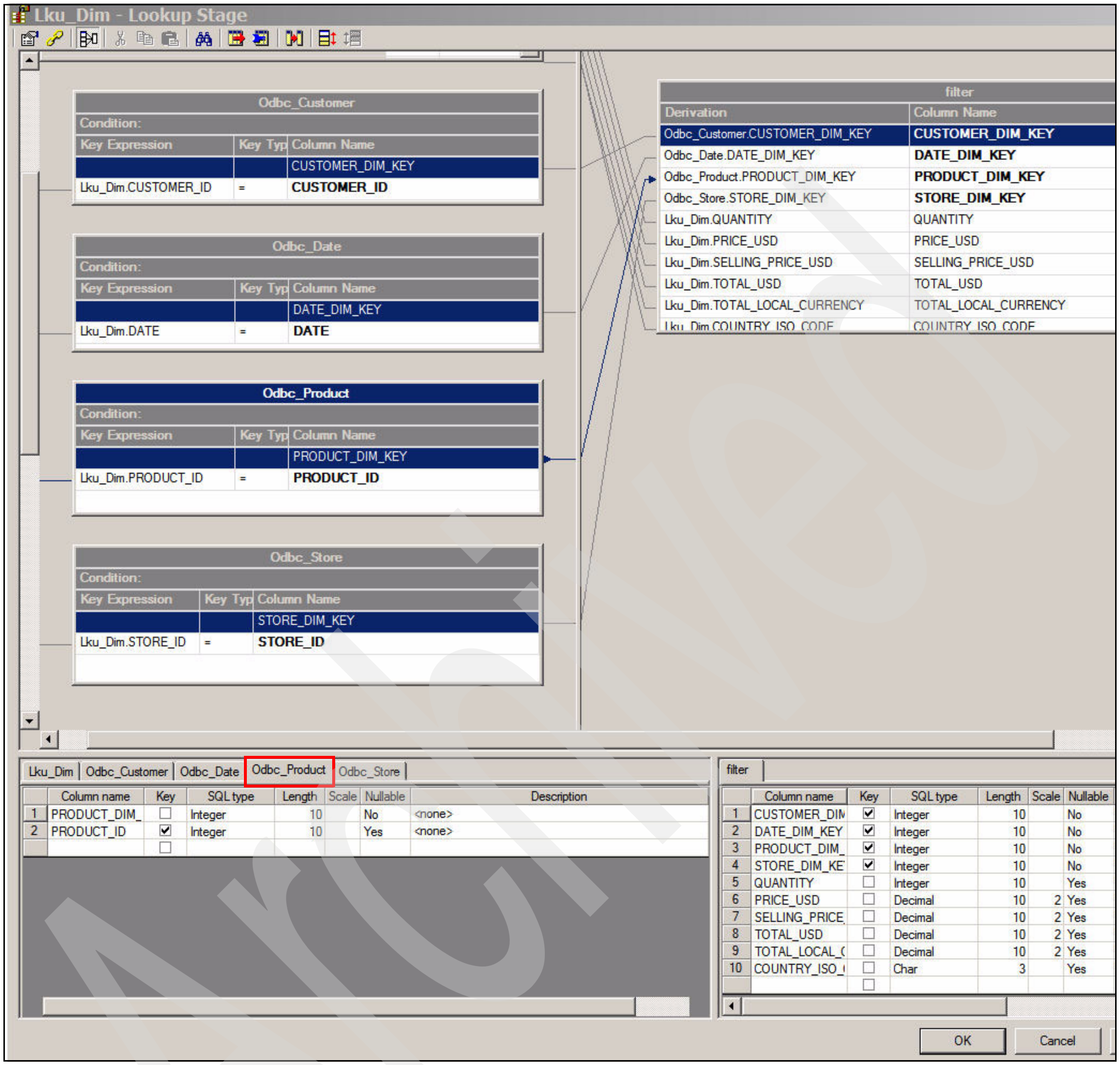


Figure 3-239 Create the J08_IL_LoadSalesFact job 19/34

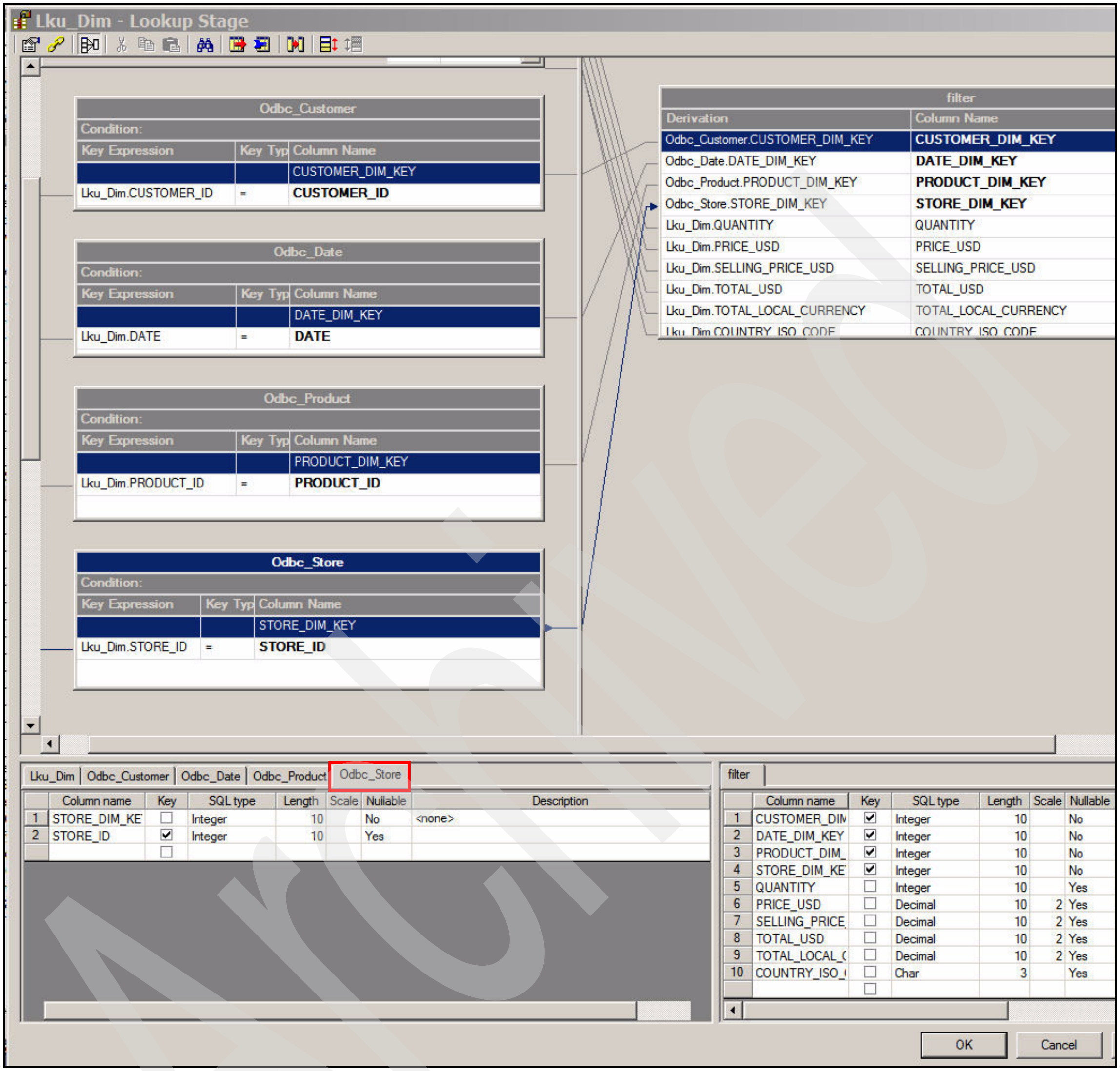


Figure 3-240   Create the J08_IL_LoadSalesFact job 20/34

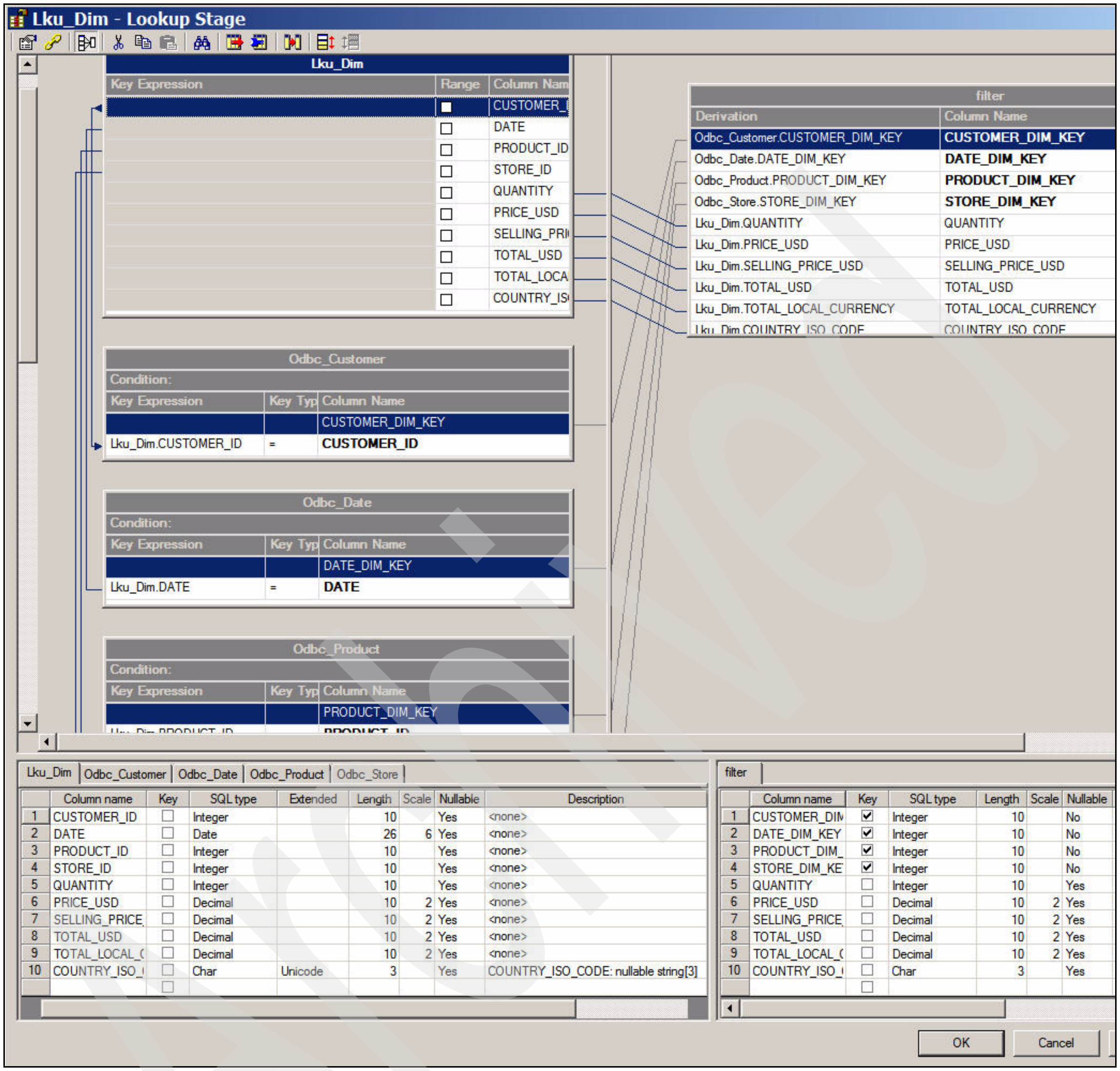


Figure 3-241 Create the J08_IL_LoadSalesFact job 21/34

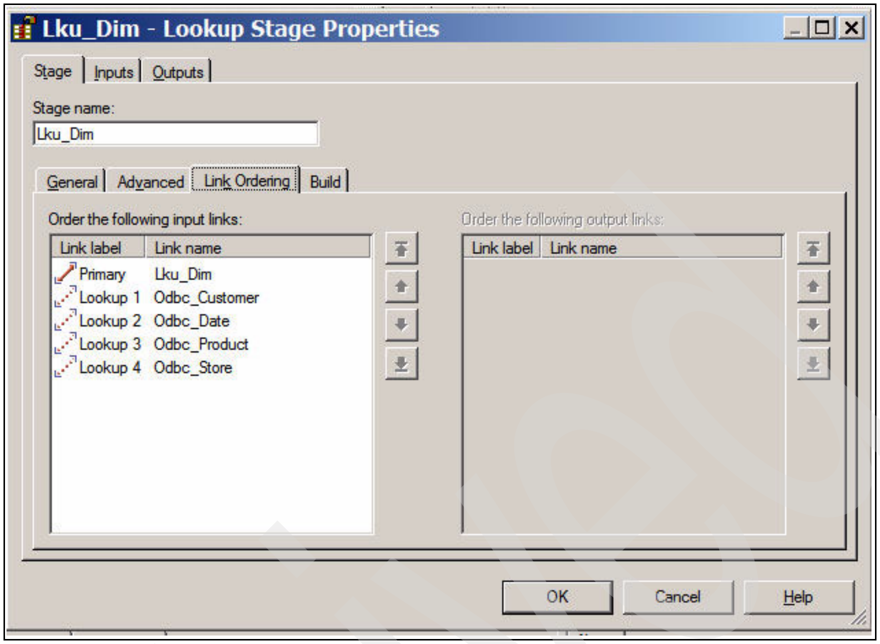


Figure 3-242 Create the J08_IL_LoadSalesFact job 22/34

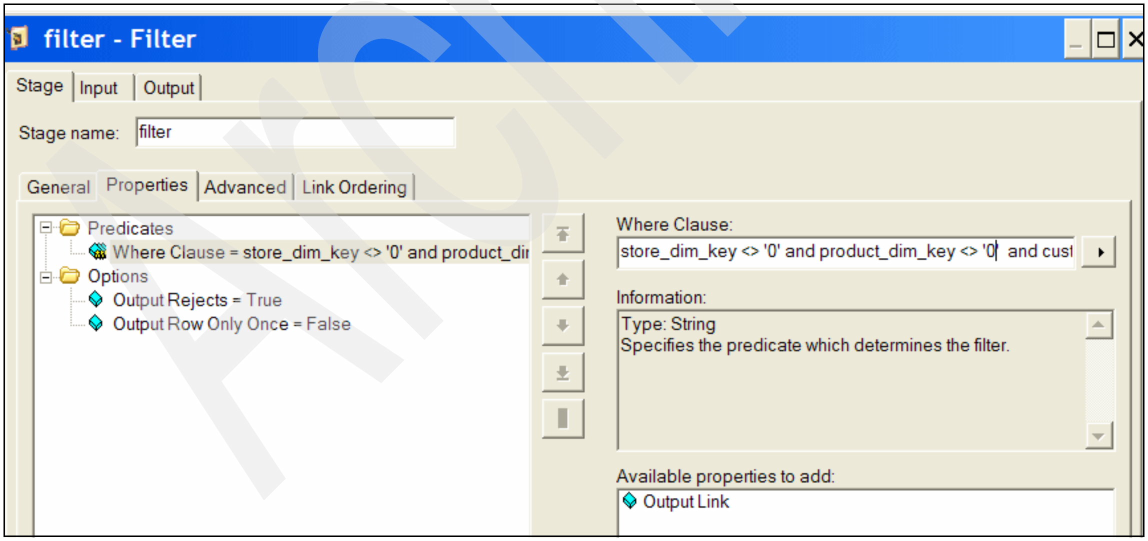


Figure 3-243 Create the J08_IL_LoadSalesFact job 23/34

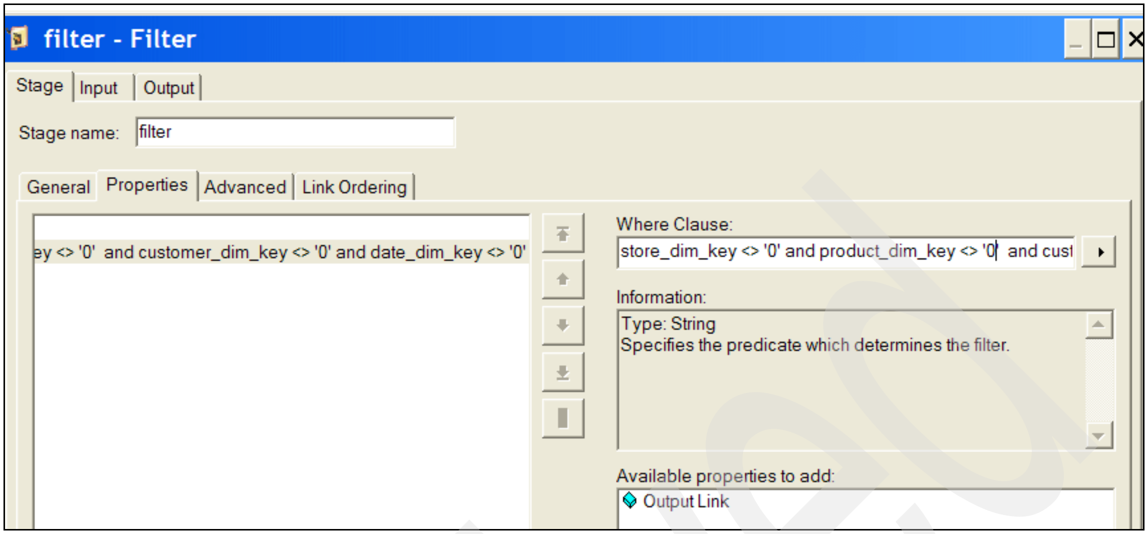


Figure 3-244 Create the J08_IL_LoadSalesFact job 24/34

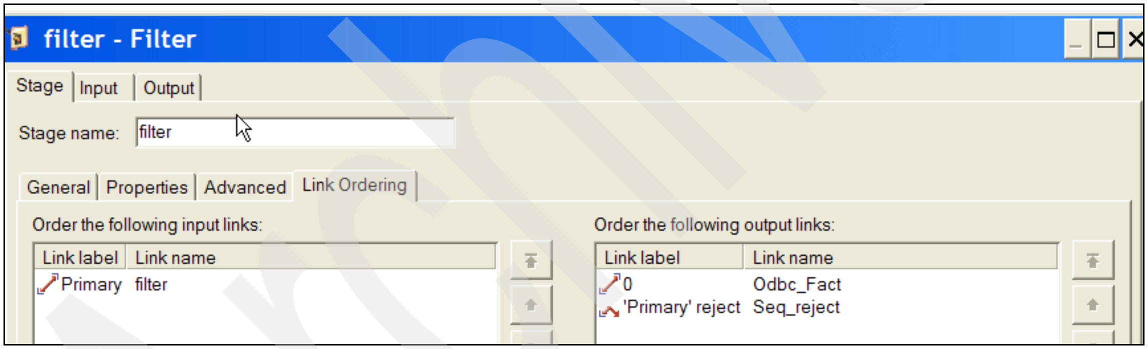


Figure 3-245 Create the J08_IL_LoadSalesFact job 25/34

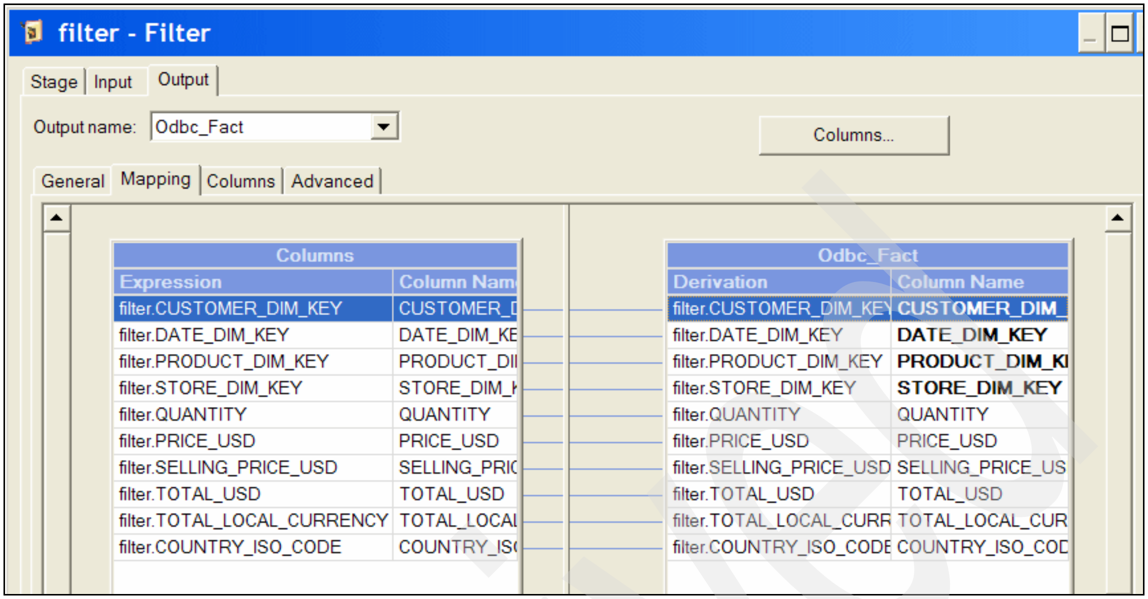


Figure 3-246 Create the J08_IL_LoadSalesFact job 26/34

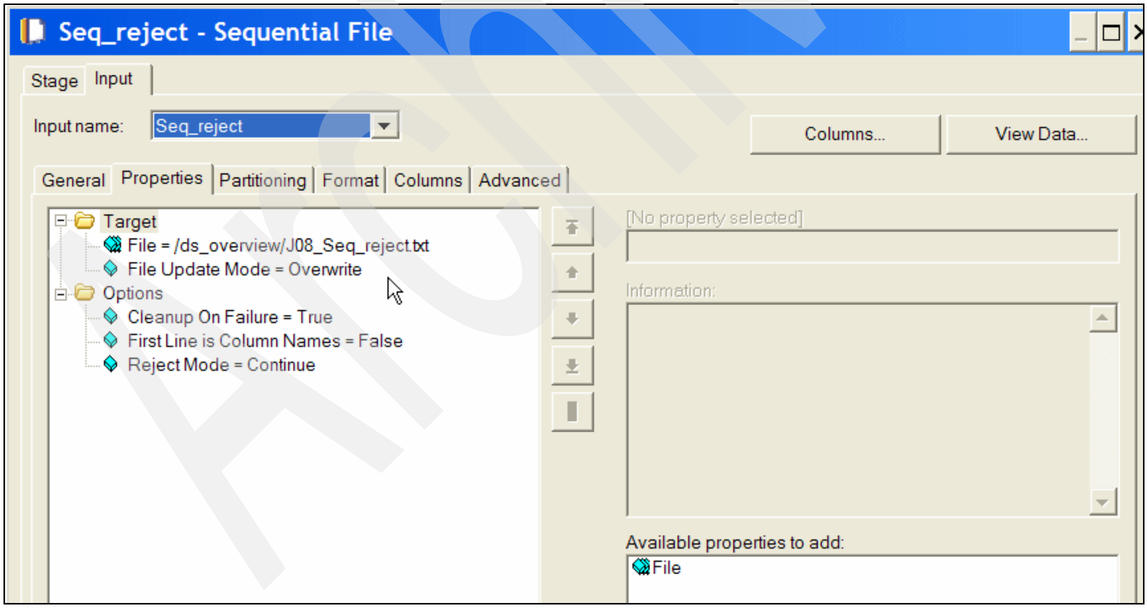


Figure 3-247 Create the J08_IL_LoadSalesFact job 27/34

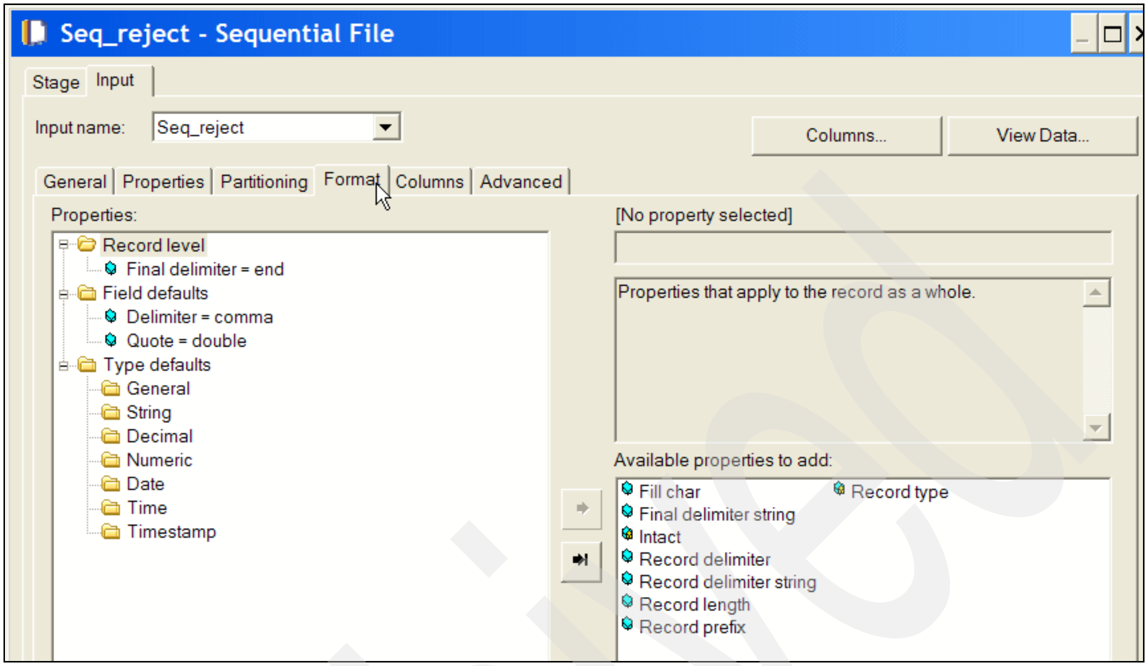


Figure 3-248 Create the J08_IL_LoadSalesFact job 28/34

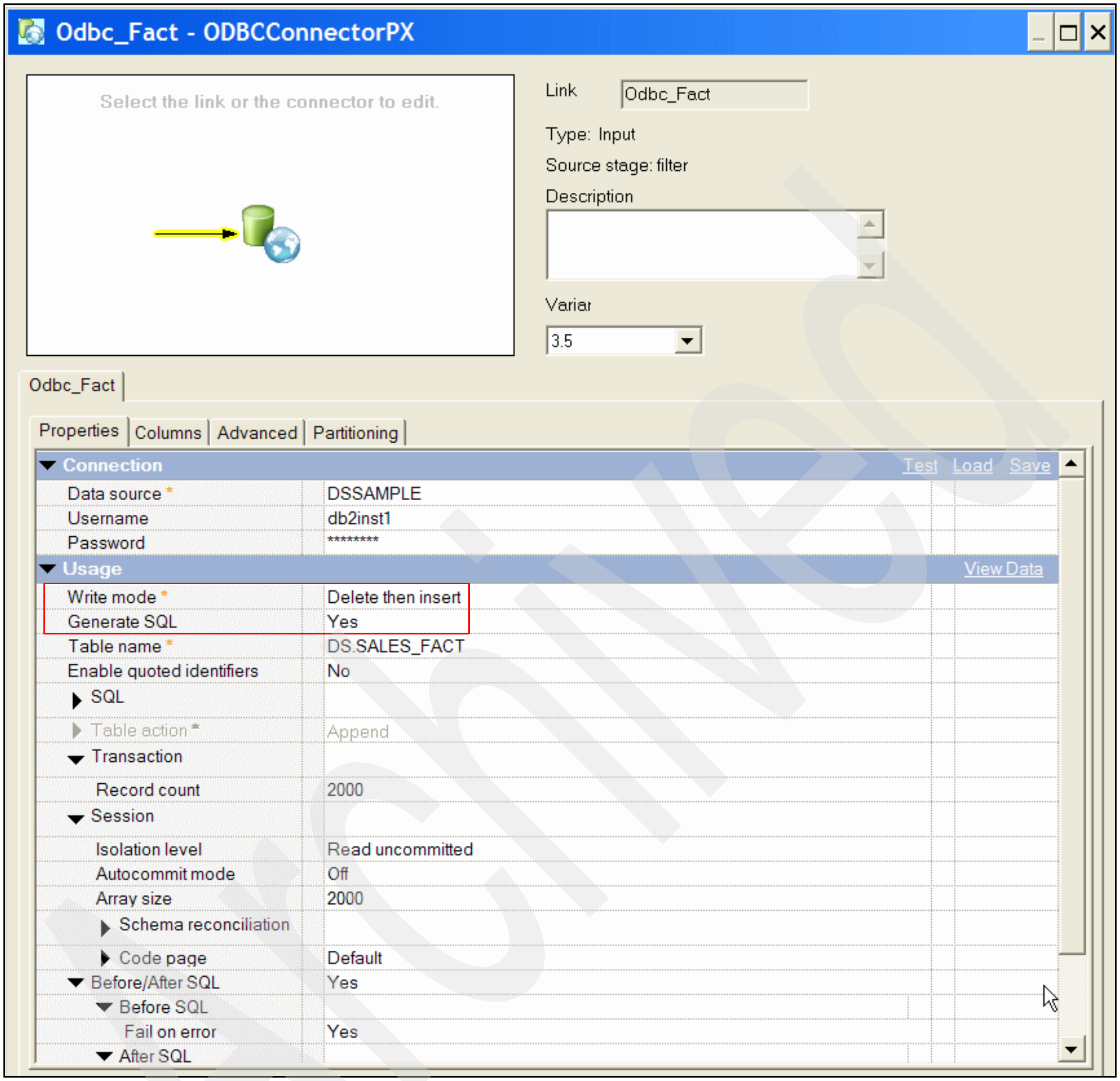


Figure 3-249   Create the J08_IL_LoadSalesFact job 29/34

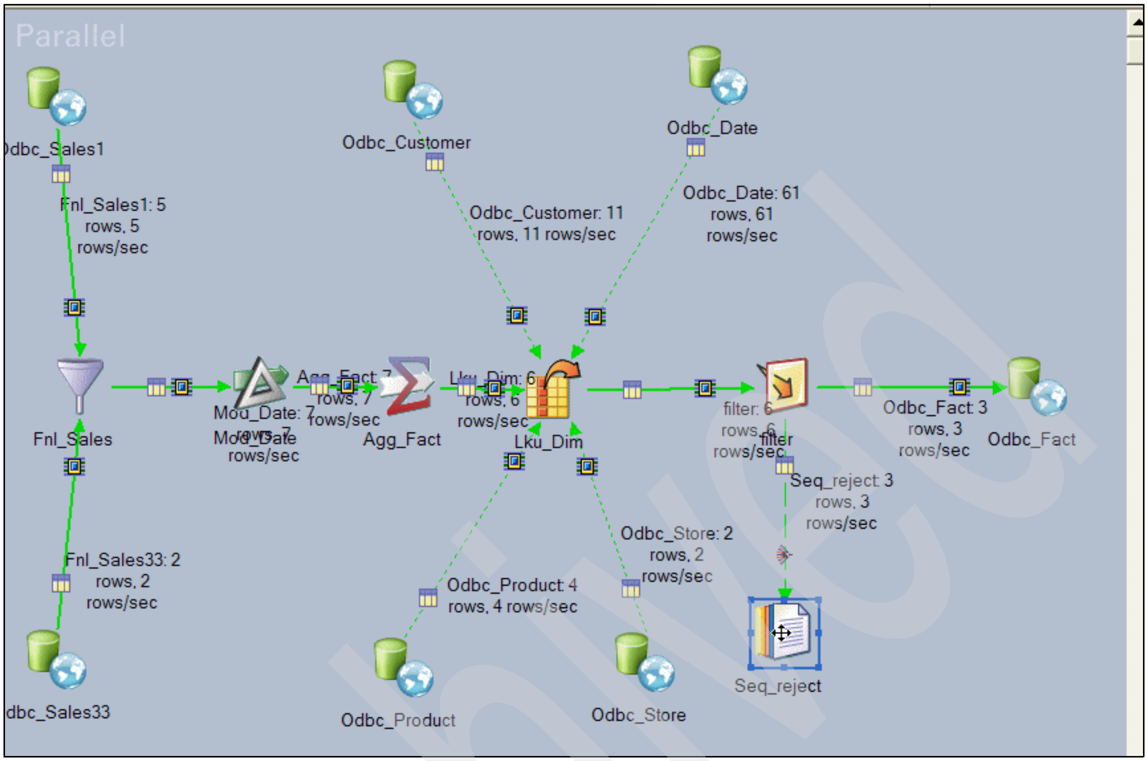


Figure 3-250 Create the J08_IL_LoadSalesFact job 30/34

J08_IL_LoadSalesFact..Seq_reject.Seq_reject - Data Browser

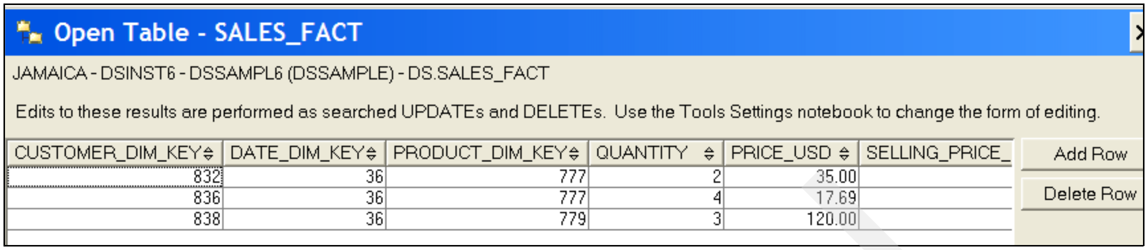
CUSTOMER_DIM_KEY	DATE_DIM_KEY	PRODUCT_DIM_KEY	STORE_DIM_KEY	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	TO
838	36	0	742	3	00000120.00	00000120.00	00000360.00	0
834	36	0	743	1	00000033.33	00000033.33	00000033.33	0
0	36	776	743	10	00000003.35	00000003.35	00000033.50	0

Figure 3-251 Create the J08_IL_LoadSalesFact job 31/34

J08_IL_LoadSalesFact..Seq_reject.Seq_reject - Data Browser

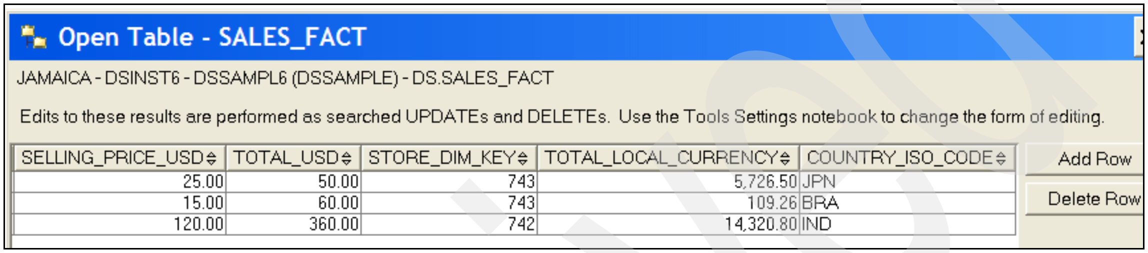
STORE_DIM_KEY	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURRENCY	COUNTRY_ISO_CODE
742	3	00000120.00	00000120.00	00000360.00	00000251.78	FRA
743	1	00000033.33	00000033.33	00000033.33	00003817.28	JPN
743	10	00000003.35	00000003.35	00000033.50	00003836.76	JPN

Figure 3-252 Create the J08_IL_LoadSalesFact job 32/34



CUSTOMER_DIM_KEY	DATE_DIM_KEY	PRODUCT_DIM_KEY	QUANTITY	PRICE_USD	SELLING_PRICE_
832	36	777	2	35.00	
836	36	777	4	17.69	
838	36	779	3	120.00	

Figure 3-253 Create the J08_IL_LoadSalesFact job 33/34



SELLING_PRICE_USD	TOTAL_USD	STORE_DIM_KEY	TOTAL_LOCAL_CURRENCY	COUNTRY_ISO_CODE
25.00	50.00	743	5,726.50	JPN
15.00	60.00	743	109.26	BRA
120.00	360.00	742	14,320.80	IND

Figure 3-254 Create the J08_IL_LoadSalesFact job 34/34

J09_IL_LoadLookupCustomerDim

When multiple versions of a business key are maintained in a dimension table, each of the entries associated with a particular business is associated with an effective date range and a surrogate key. The process that maintains multiple versions of a business key (Slowly Changing Dimension in our case) is responsible for maintaining the effective date and generating a surrogate key for the current version of a business key.

Before an incoming sales transaction can be loaded into the SALES_FACT table, it has to be aggregated per the grouping columns, and then associated with the surrogate key of that business key corresponding to the date of the sales transaction. Typically, an incoming sales transaction would correspond to the current version of the business key in the dimension table unless delays caused late arriving data that corresponds to an earlier version of the business key. A lookup table must be generated for each dimension table that corresponds to the current version of a business key that specifies the effective date.

In this job, we extract all the current version of the business keys from the Customer_Dim table and write it to an interim LOOKUP_CUSTOMER_DIM table. All the attributes of the CUSTOMER_DIM table are written to this lookup table excepting the surrogate key.

Figure 3-255 on page 322 through Figure 3-266 on page 327 describe the main steps in creating a Customer lookup dimension table as follows:

1. Figure 3-255 on page 322 shows the various stages in the job — it includes a source ODBCConnectorPX stage, a Sort, a Remove Duplicates stage, and a target ODBCConnectorPX stage. The names of the stages were modified as shown.
1. Figure 3-256 on page 322 shows an ODBCConnectorPX stage that retrieves records from the CUSTOMER_DIM table using automatically generated SQL SELECT statements.
2. The extracted rows from the previous stage are written to the output link Srt_CustomerDim to be sorted on ascending sequence of CUSTOMER_ID (business key) and EFFECTIVE_TS (effective timestamp) as shown in the Properties tab of the Stage page in Figure 3-257 on page 323.

   Figure 3-258 on page 323 shows the **Mapping** tab in the Output page, which maps all the input columns to the output.
3. When multiple versions exist for a particular business key, there will be duplicates of the same business key (CUSTOMER_ID) value. To ensure that only the current version is selected (corresponding to the row with the latest effective timestamp), the output of the previous stage is fed to a Remove Duplicates stage with the specification Duplicate To Retain = Last option selected. This ensures that only the business key with the highest effective timestamp is retained in the output link ODBC_LookupCustomerDim. This is shown in Figure 3-259 on page 324.
4. Figure 3-260 on page 324 shows the **Mapping** tab in the Output page, which maps all the input columns to the output except the surrogate key CUSTOMER_DIM_KEY.
5. Figure 3-261 on page 325 shows the ODBCConnectorPX stage that is used to update/insert the current version of the business key into the LOOKUP_CUSTOMER_DIM table. The Write mode is Update then Insert, since an insert will fail if the business key already exists. The SQL INSERT (Figure 3-262 on page 325) and UPDATE (Figure 3-263 on page 326) are manually generated as shown. statement is automatically generated.
6. Figure 3-264 on page 326 shows the results of the job execution, where a total of eleven rows are generated and inserted into the LOOKUP_CUSTOMER_DIM table. Figure 3-265 on page 327 and Figure 3-266 on page 327 show the eleven rows inserted into the LOOKUP_CUSTOMER_DIM table.

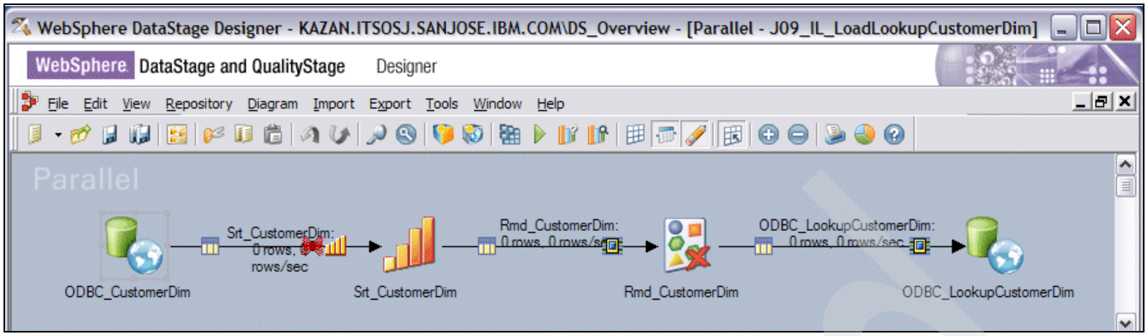


Figure 3-255 Create the J09_IL_LoadLookupCustomerDim job 1/12

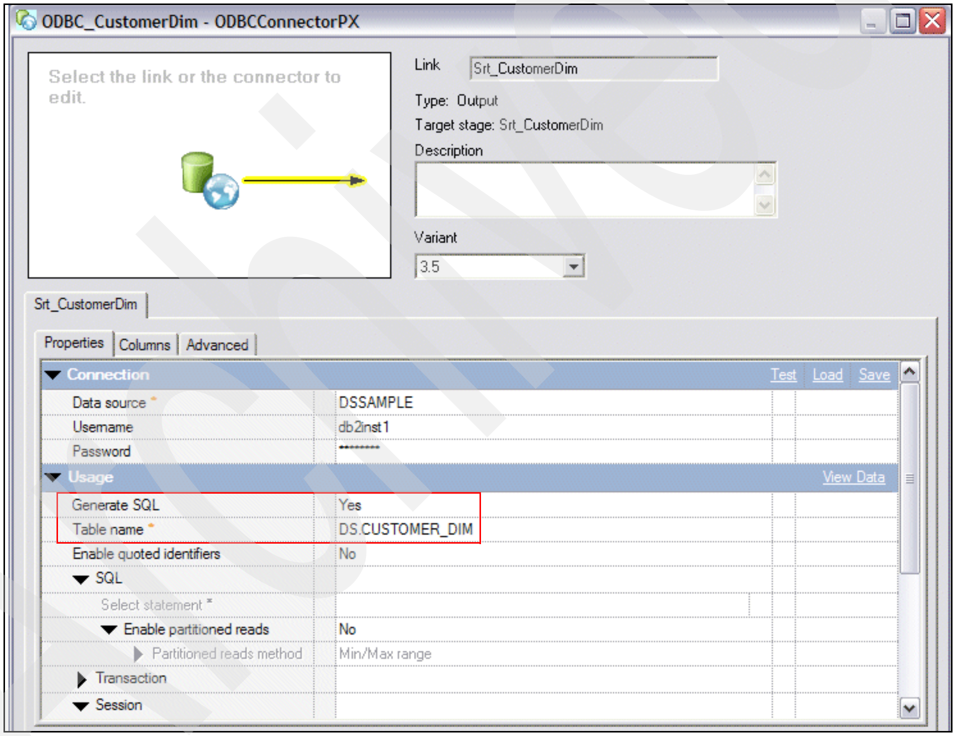


Figure 3-256 Create the J09_IL_LoadLookupCustomerDim job 2/12

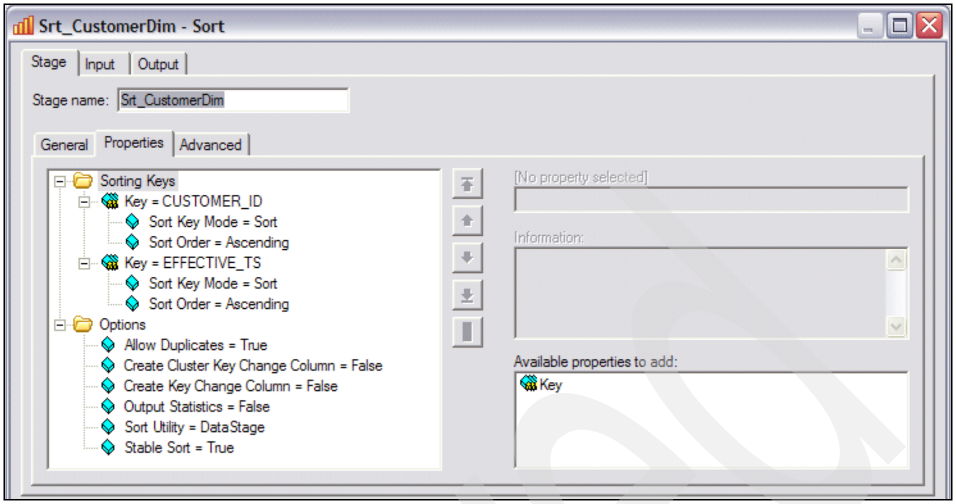


Figure 3-257 Create the J09_IL_LoadLookupCustomerDim job 3/12

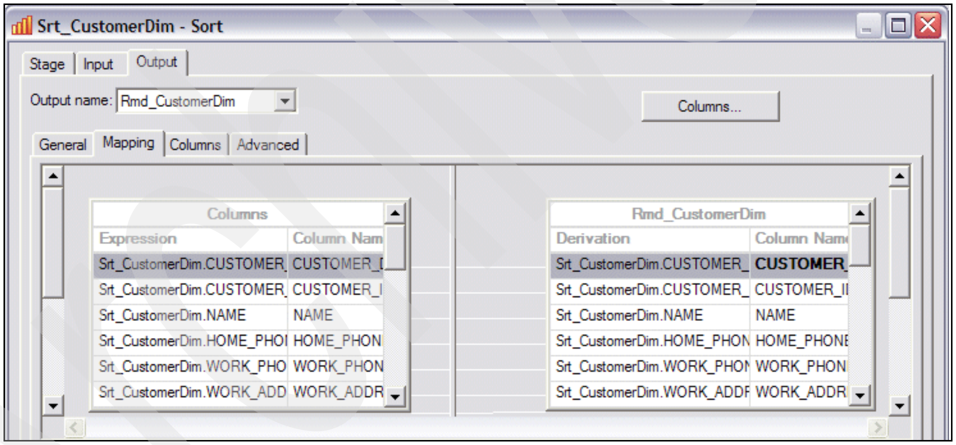


Figure 3-258 Create the J09_IL_LoadLookupCustomerDim job 4/12

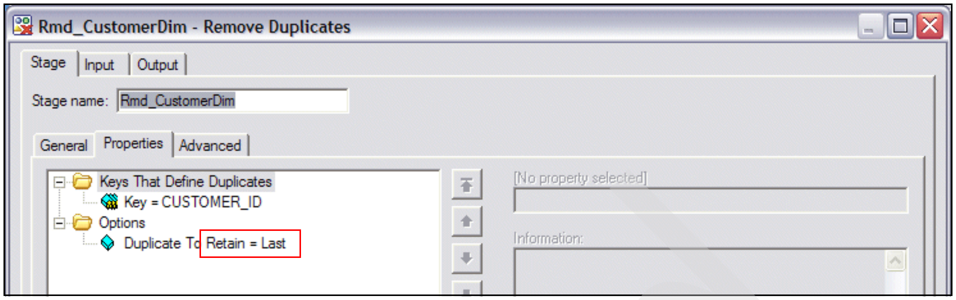


Figure 3-259   Create the J09_IL_LoadLookupCustomerDim job 5/12

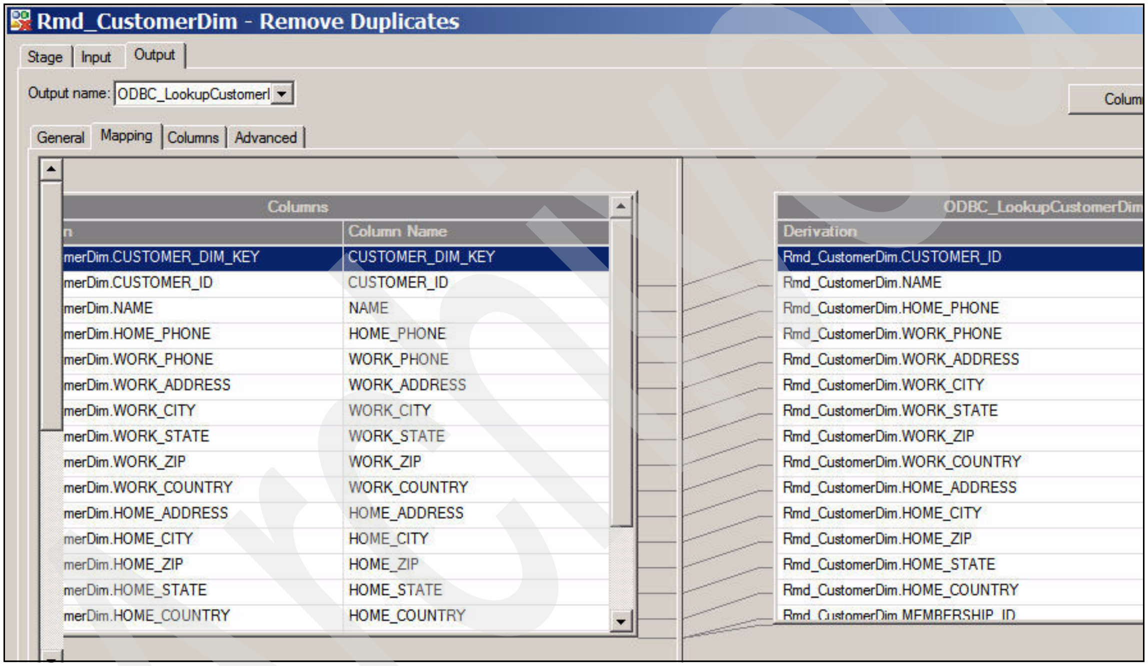


Figure 3-260   Create the J09_IL_LoadLookupCustomerDim job 6/12

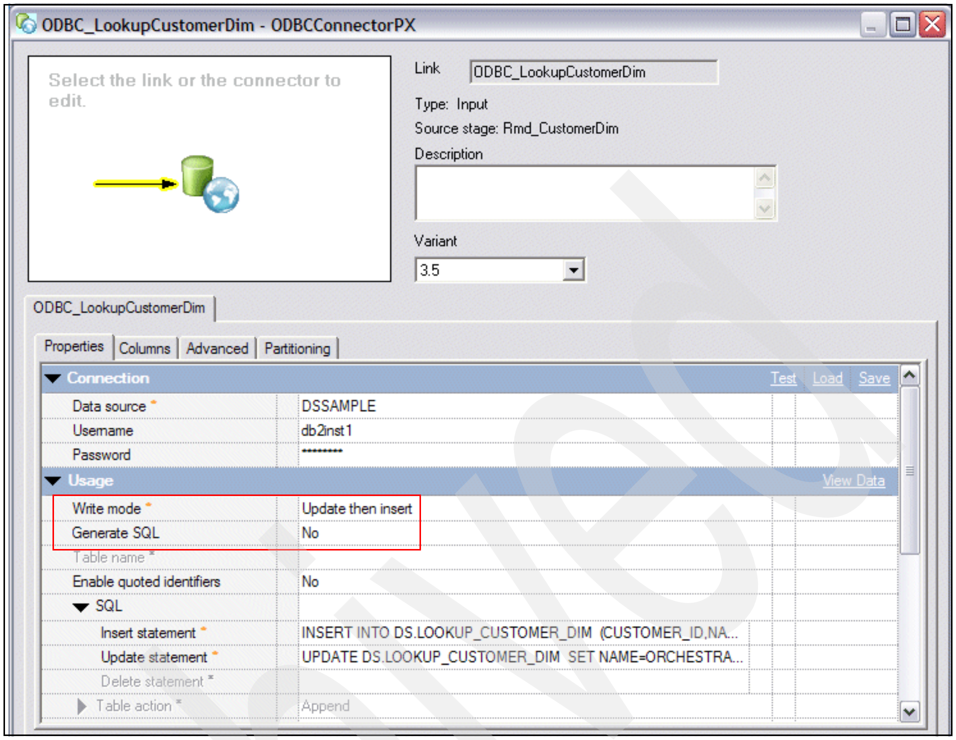


Figure 3-261 Create the J09_IL_LoadLookupCustomerDim job 7/12

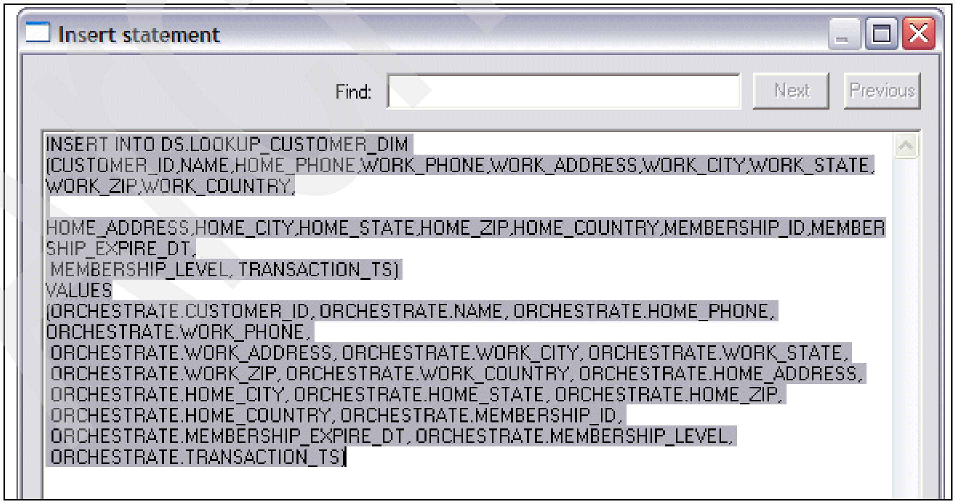


Figure 3-262 Create the J09_IL_LoadLookupCustomerDim job 8/12

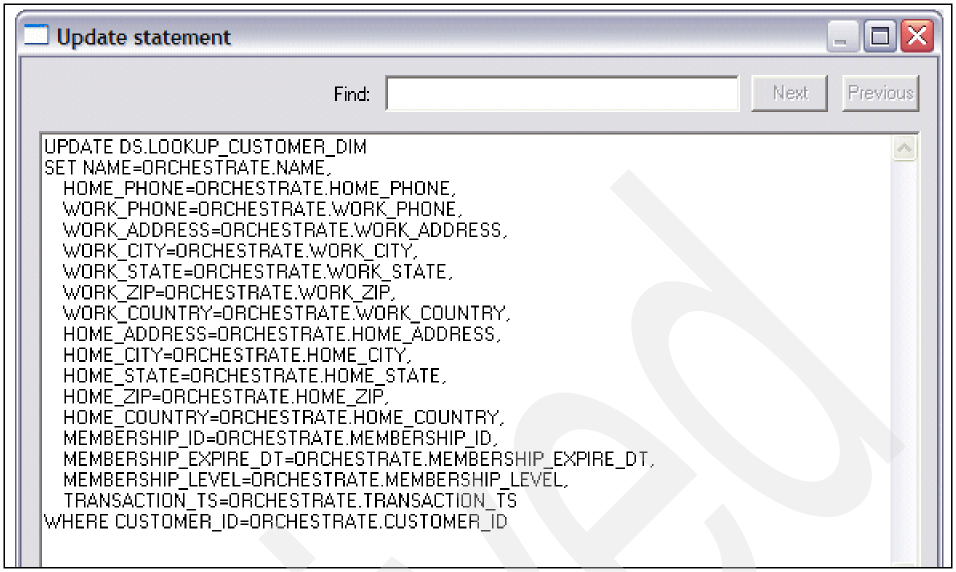


Figure 3-263 Create the J09_IL_LoadLookupCustomerDim job 9/12

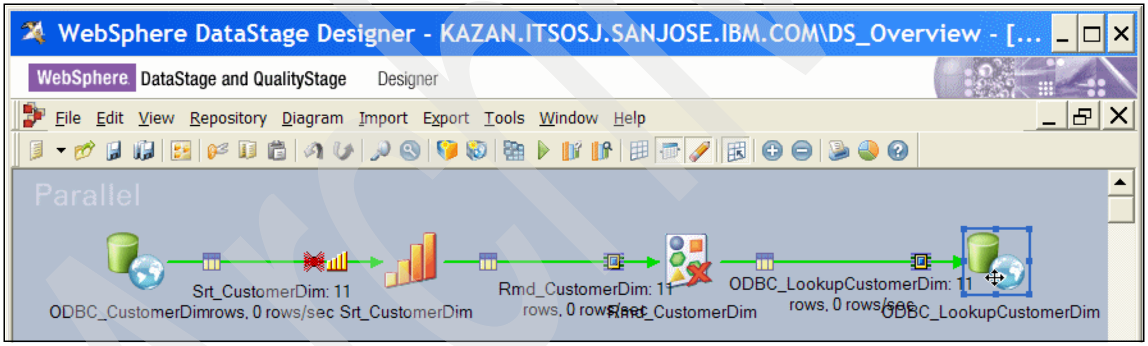
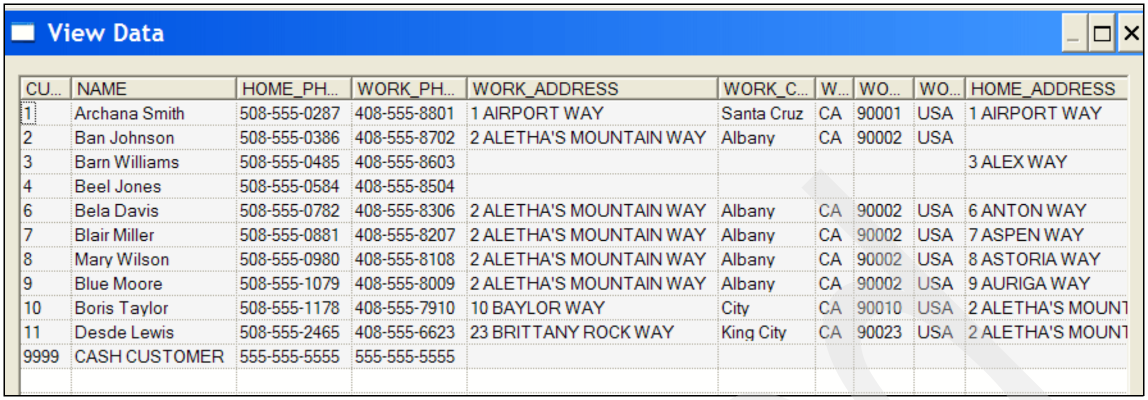


Figure 3-264 Create the J09_IL_LoadLookupCustomerDim job 10/12



CU...	NAME	HOME_PH...	WORK_PH...	WORK_ADDRESS	WORK_C...	W...	WO...	WO...	HOME_ADDRESS
1	Archana Smith	508-555-0287	408-555-8801	1 AIRPORT WAY	Santa Cruz	CA	90001	USA	1 AIRPORT WAY
2	Ban Johnson	508-555-0386	408-555-8702	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	
3	Barn Williams	508-555-0485	408-555-8603						3 ALEX WAY
4	Beel Jones	508-555-0584	408-555-8504						
6	Bela Davis	508-555-0782	408-555-8306	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	6 ANTON WAY
7	Blair Miller	508-555-0881	408-555-8207	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	7 ASPEN WAY
8	Mary Wilson	508-555-0980	408-555-8108	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	8 ASTORIA WAY
9	Blue Moore	508-555-1079	408-555-8009	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	9 AURIGA WAY
10	Boris Taylor	508-555-1178	408-555-7910	10 BAYLOR WAY	City	CA	90010	USA	2 ALETHA'S MOUNT
11	Desde Lewis	508-555-2465	408-555-6623	23 BRITTANY ROCK WAY	King City	CA	90023	USA	2 ALETHA'S MOUNT
9999	CASH CUSTOMER	555-555-5555	555-555-5555						

Figure 3-265 Create the J09_IL_LoadLookupCustomerDim job 11/12

DRESS	HOME_CITY	HO...	H...	HO...	M...	MEMBERSHIP_EXPIRE_DT	M...	TRANSACTION_TS
WAY	Santa Cruz	90001	CA	USA	1	Thursday, February 16, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
					2	Friday, February 17, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
Y	Amador City	90003	CA	USA	3	Saturday, February 18, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
					4	Sunday, February 19, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
VAY	Bradbury	90006	CA	USA	6	Tuesday, February 21, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
/AY	Brawley	90007	CA	USA	7	Wednesday, February 22, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
WAY	California City	90008	CA	USA	8	Thursday, February 23, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
VAY	Cathedral City	90009	CA	USA	9	Friday, February 24, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
S MOUNTAIN WAY	Albany	90002	CA	USA	10	Saturday, February 25, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
S MOUNTAIN WAY	Albany	90002	CA	USA	99	Thursday, May 10, 2012	P	Monday, November 5, 2007 12:00:00 AM GMT
					0	Tuesday, December 31, 2999	P	Monday, November 5, 2007 12:00:00 AM GMT

Figure 3-266 Create the J09_IL_LoadLookupCustomerDim job 12/12

J10_IL_LoadLookupProductDim

In this job, we load the LOOKUP_PRODUCT_DIM table from the PRODUCT_DIM dimension table. Figure 3-267 on page 328 through Figure 3-273 on page 330 show some of the main steps in loading this table. Since this is similar to the process described in "J09_IL_LoadLookupCustomerDim" on page 320, it is not repeated here.

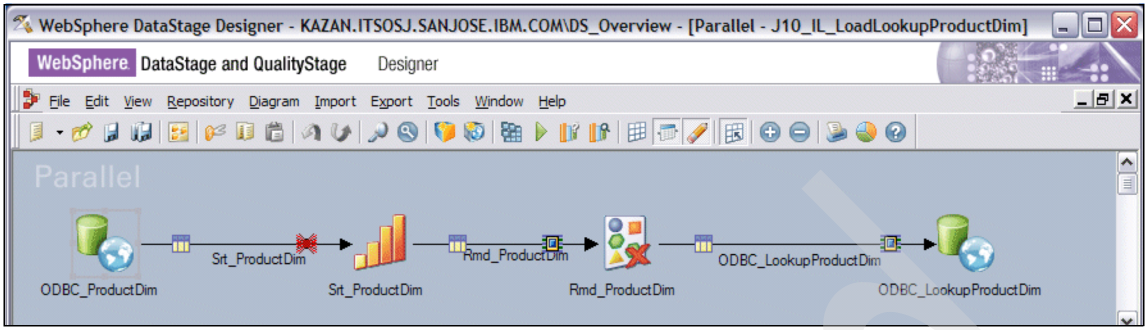


Figure 3-267 Create the J10_IL_LoadLookupProductDim job 1/7

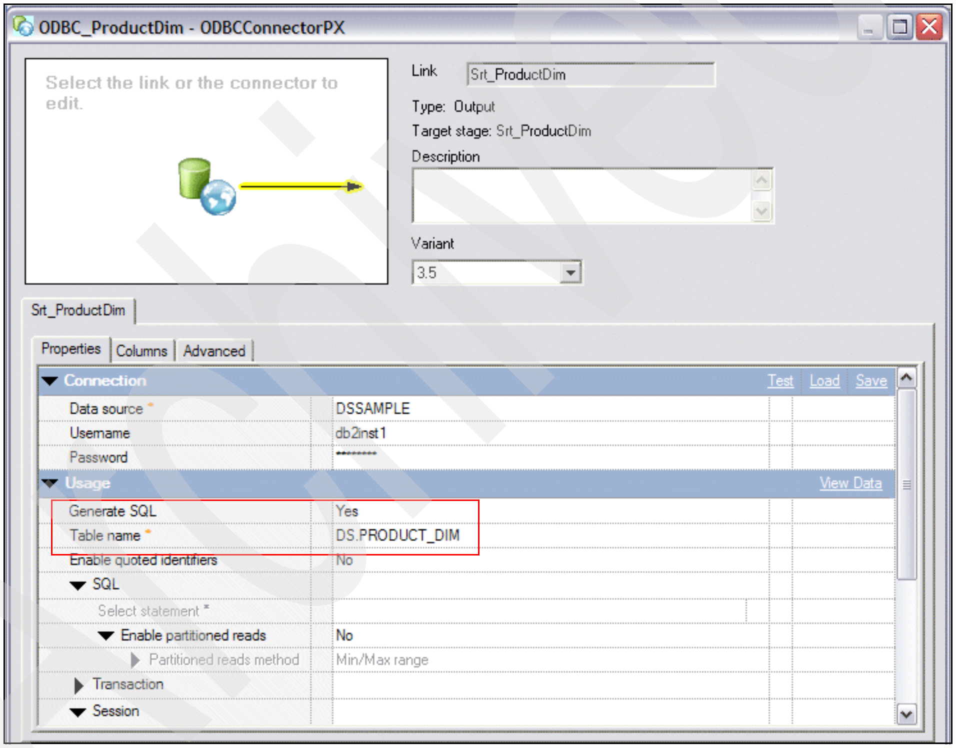


Figure 3-268 Create the J10_IL_LoadLookupProductDim job 2/7

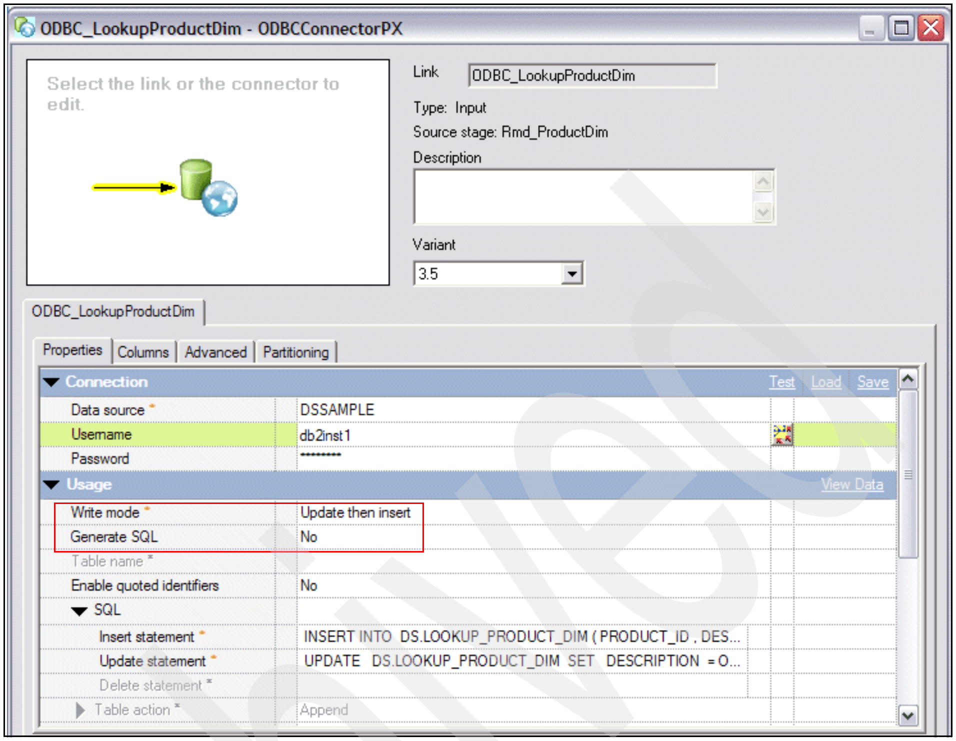


Figure 3-269 Create the J10_IL_LoadLookupProductDim job 3/7

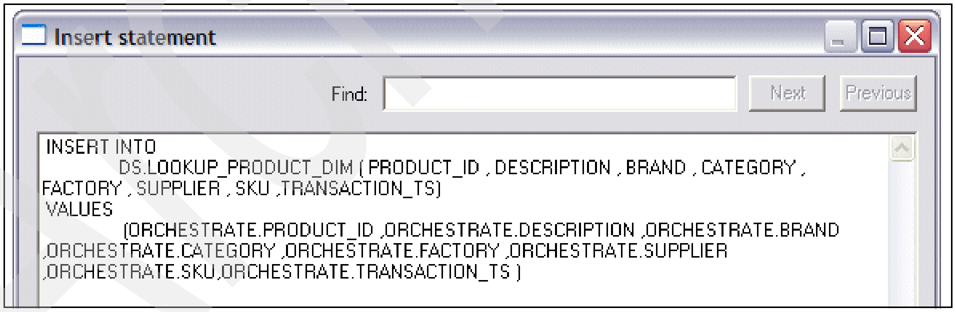


Figure 3-270 Create the J10_IL_LoadLookupProductDim job 4/7

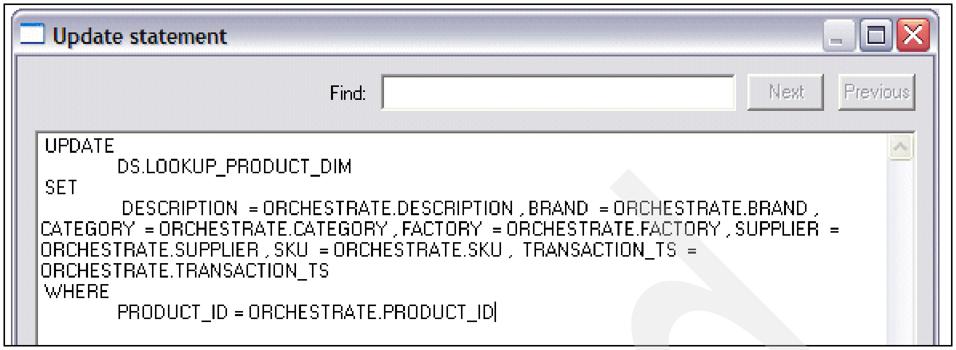


Figure 3-271 Create the J10_IL_LoadLookupProductDim job 5/7

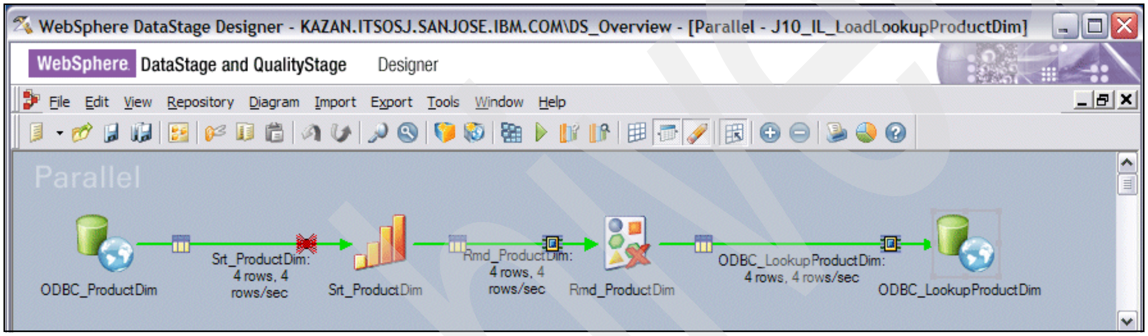
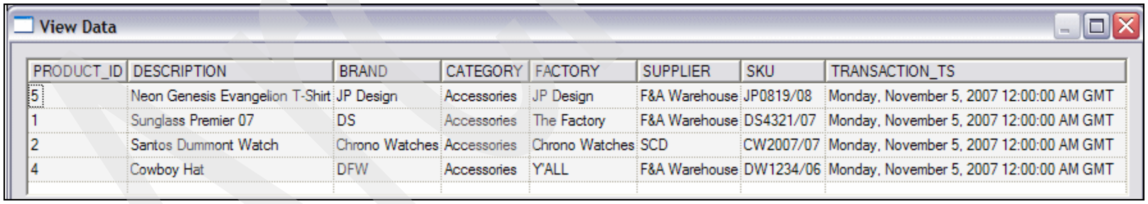


Figure 3-272 Create the J10_IL_LoadLookupProductDim job 6/7


View Data

PRODUCT_ID	DESCRIPTION	BRAND	CATEGORY	FACTORY	SUPPLIER	SKU	TRANSACTION_TS
5	Neon Genesis Evangelion T-Shirt	JP Design	Accessories	JP Design	F&A Warehouse	JP0819/08	Monday, November 5, 2007 12:00:00 AM GMT
1	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	Monday, November 5, 2007 12:00:00 AM GMT
2	Santos Dummont Watch	Chrono Watches	Accessories	Chrono Watches	SCD	CW2007/07	Monday, November 5, 2007 12:00:00 AM GMT
4	Cowboy Hat	DFW	Accessories	Y'ALL	F&A Warehouse	DW1234/06	Monday, November 5, 2007 12:00:00 AM GMT

Figure 3-273 Create the J10_IL_LoadLookupProductDim job 7/7

J11_IL_LoadLookupStoreDim

In this job, we load the LOOKUP_STORE_DIM table from the STORE_DIM dimension table. Figure 3-274 on page 331 through Figure 3-284 on page 335 show some of the main steps in loading this table. Since this is similar to the process described in "J09_IL_LoadLookupCustomerDim" on page 320, it is not repeated here.

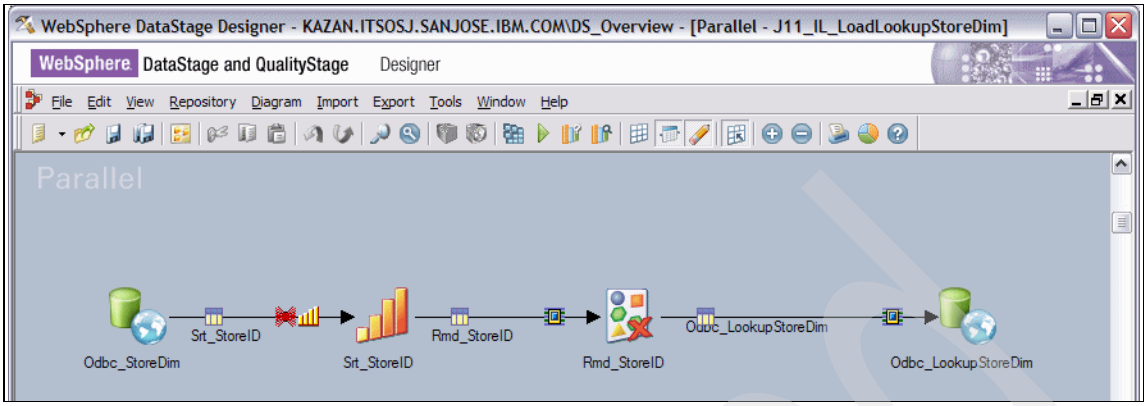


Figure 3-274 Create the J11_IL_LoadLookupStoreDim job 1/11

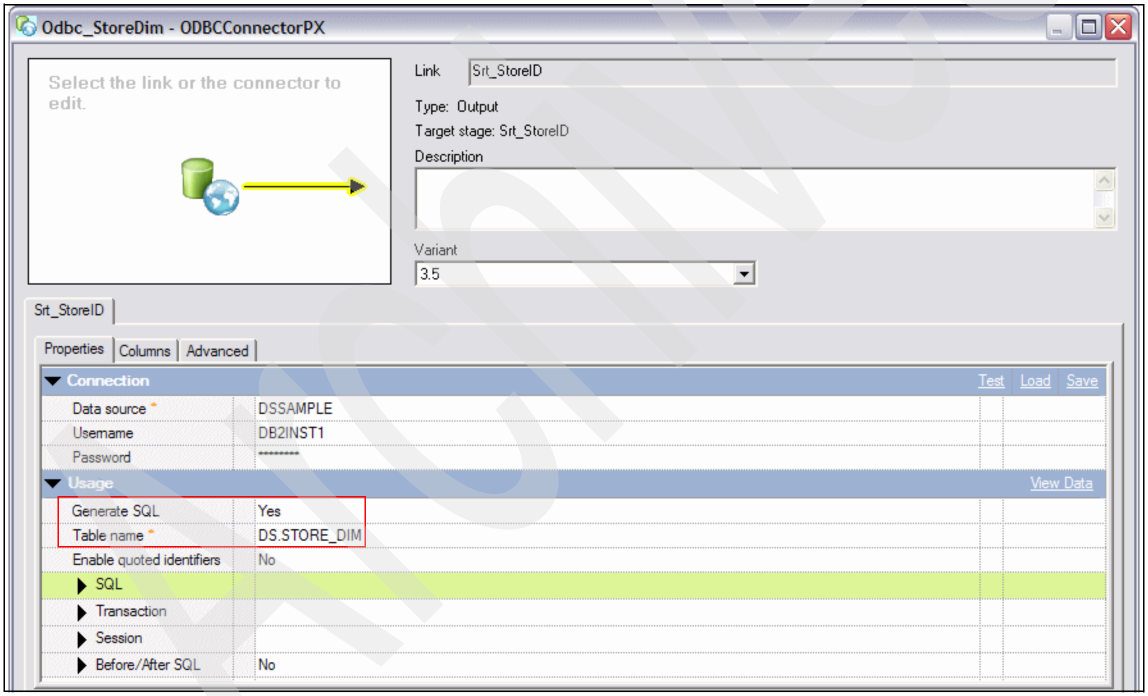


Figure 3-275 Create the J11_IL_LoadLookupStoreDim job 2/11

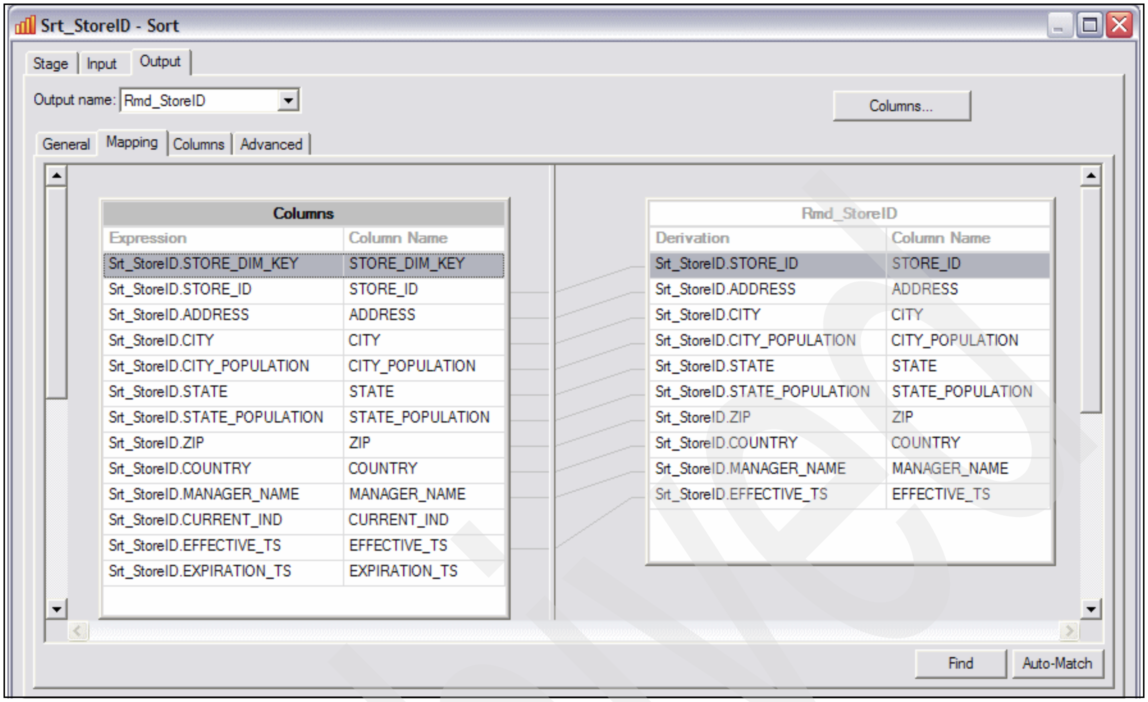


Figure 3-276 Create the J11_IL_LoadLookupStoreDim job 3/11

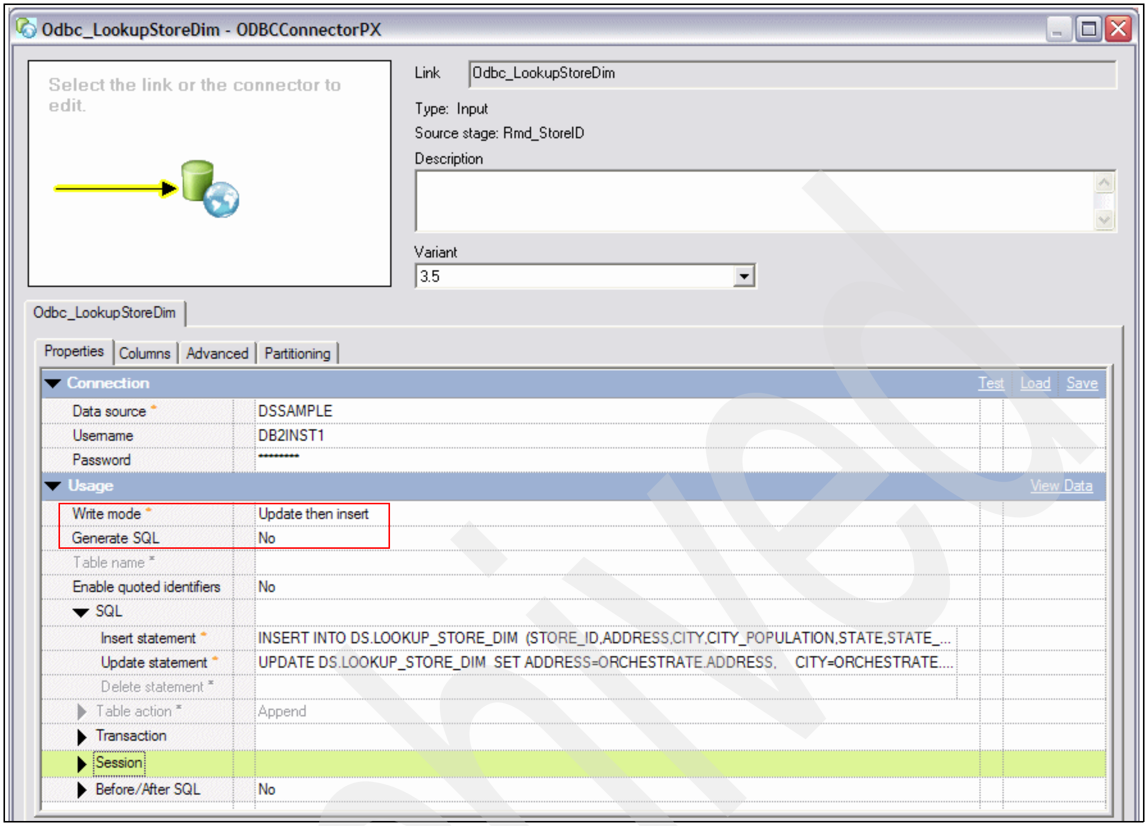


Figure 3-277 Create the J11_IL_LoadLookupStoreDim job 4/11

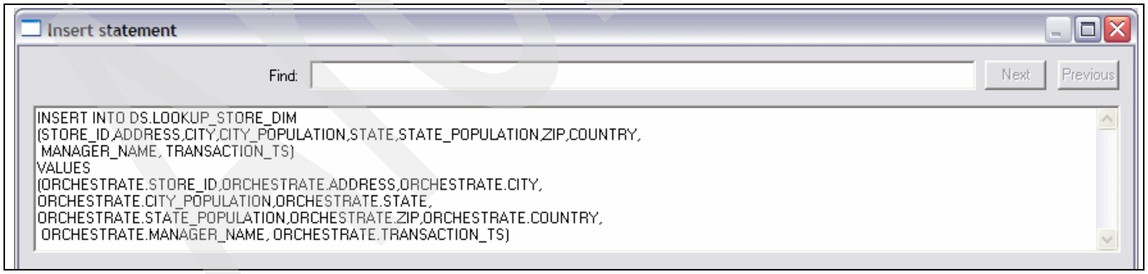


Figure 3-278 Create the J11_IL_LoadLookupStoreDim job 5/11

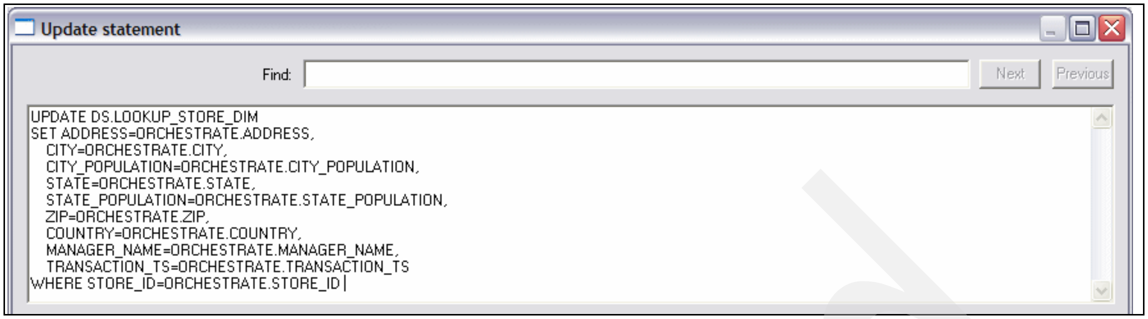


Figure 3-279 Create the J11_IL_LoadLookupStoreDim job 6/11

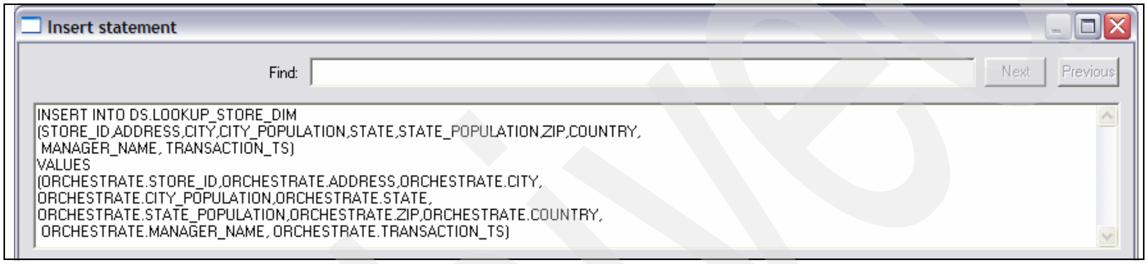


Figure 3-280 Create the J11_IL_LoadLookupStoreDim job 7/11

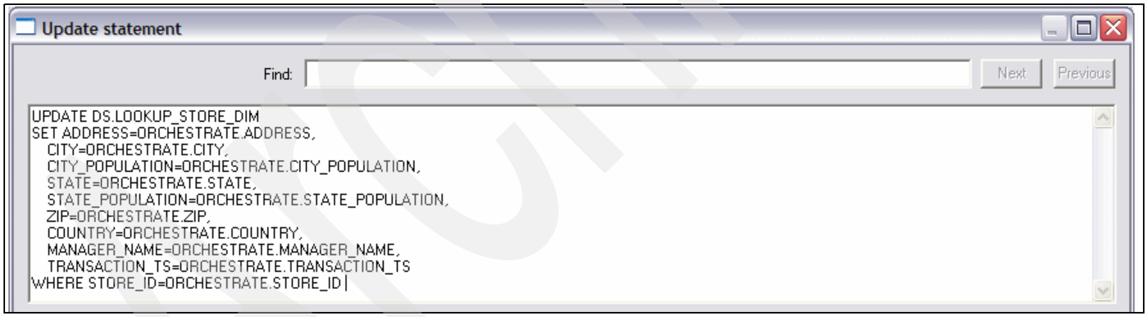


Figure 3-281 Create the J11_IL_LoadLookupStoreDim job 8/11

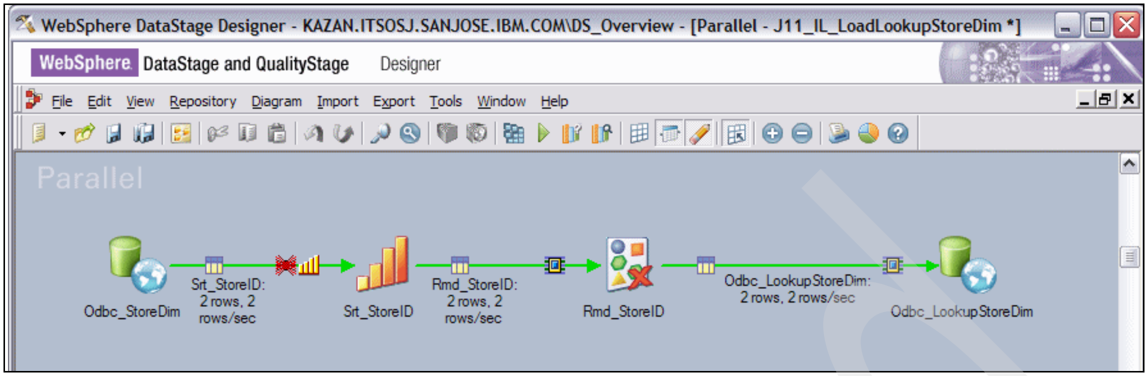


Figure 3-282 Create the J11_IL_LoadLookupStoreDim job 9/11

View Data

STORE_ID	ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP
1	12345 Almaden Expressway	San Jose	929936	CA	33871648	95118
33	8976 Brazil Ave	San Francisco	744041	CA	33871648	94112

Figure 3-283 Create the J11_IL_LoadLookupStoreDim job 10/11

View Data

PULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	MANAGER_NAME	TRANSACTION_TS
	CA	33871648	95118	USA	Aidan Smith	Monday, November 5, 2007 12:00:00 AM GMT
	CA	33871648	94112	USA	Emma Hales	Monday, November 5, 2007 12:00:00 AM GMT

Figure 3-284 Create the J11_IL_LoadLookupStoreDim job 11/11

J12_IL_GenerateSurrogateKey

As described in "Slowly Changing Dimension" on page 113, when the SCD stage performs a dimension lookup, and a match is not found, the stage obtains a new surrogate key value by using the derivation of the Surrogate Key column on the **Dim Update** tab. Since we want the SCD stage to generate new surrogate keys by using a key source that you create with a Surrogate Key Generator stage as described in "Surrogate Key Generator" on page 132, you must use the NextSurrogateKey function to derive the Surrogate Key column.

In this job, we create a surrogate key source for each of the four dimension tables using the Surrogate Key Generator stage using the surrogate key value initially loaded into the individual dimension tables.

Figure 3-285 on page 336 through Figure 3-293 on page 340 describe main steps for creating a surrogate key source for each of the four dimension tables.

The flow is as follows:

1. Figure 3-285 here shows the various stages in the job — it includes a source ODBCConnectorPX stage, and a Surrogate Key Generator stage for each of the four dimension tables. The names of the stages were modified as shown.
1. Figure 3-286 on page 337 shows an ODBCConnectorPX stage that retrieves records from the PRODUCT_DIM table using an automatically generated SQL SELECT statement. Figure 3-287 on page 337 shows the **Properties** tab in the Stage page, which identifies the Key Source's Input Column Name = PRODUCT_DIM_KEY for priming the surrogate key file. The Source Name identifies the name of the surrogate key source file.
2. Figure 3-288 on page 338 through Figure 3-293 on page 340 show the equivalent configurations for the STORE_DIM, CUSTOMER_DIM, and DATE_DIM dimension tables.

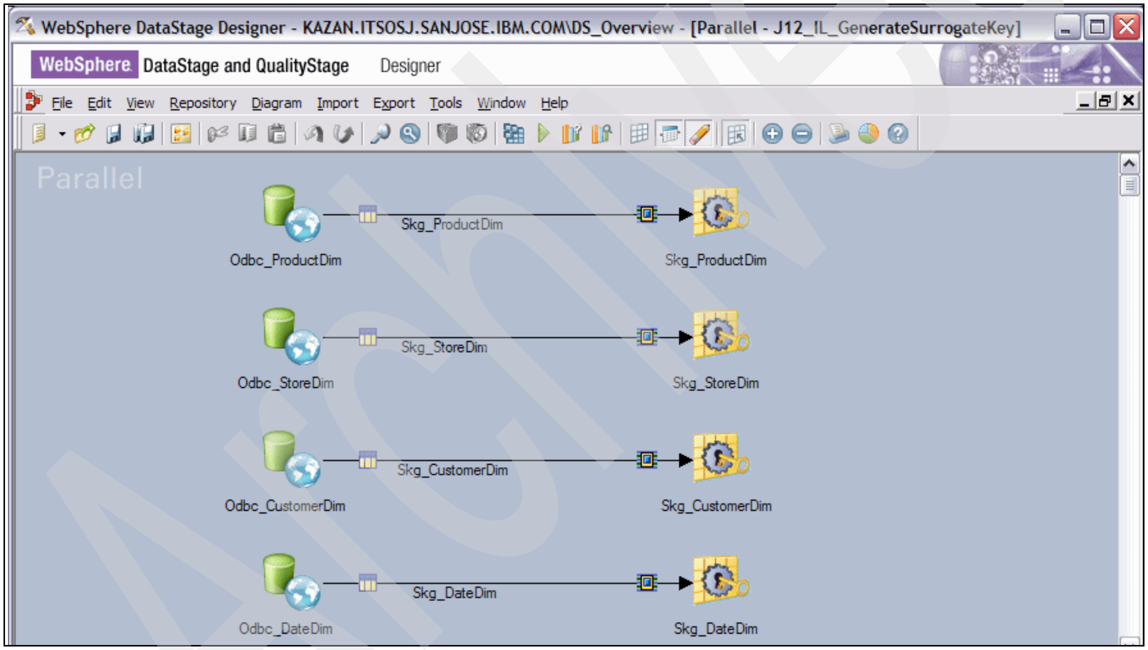


Figure 3-285 Create the J12_IL_GenerateSurrogateKey job 1/9

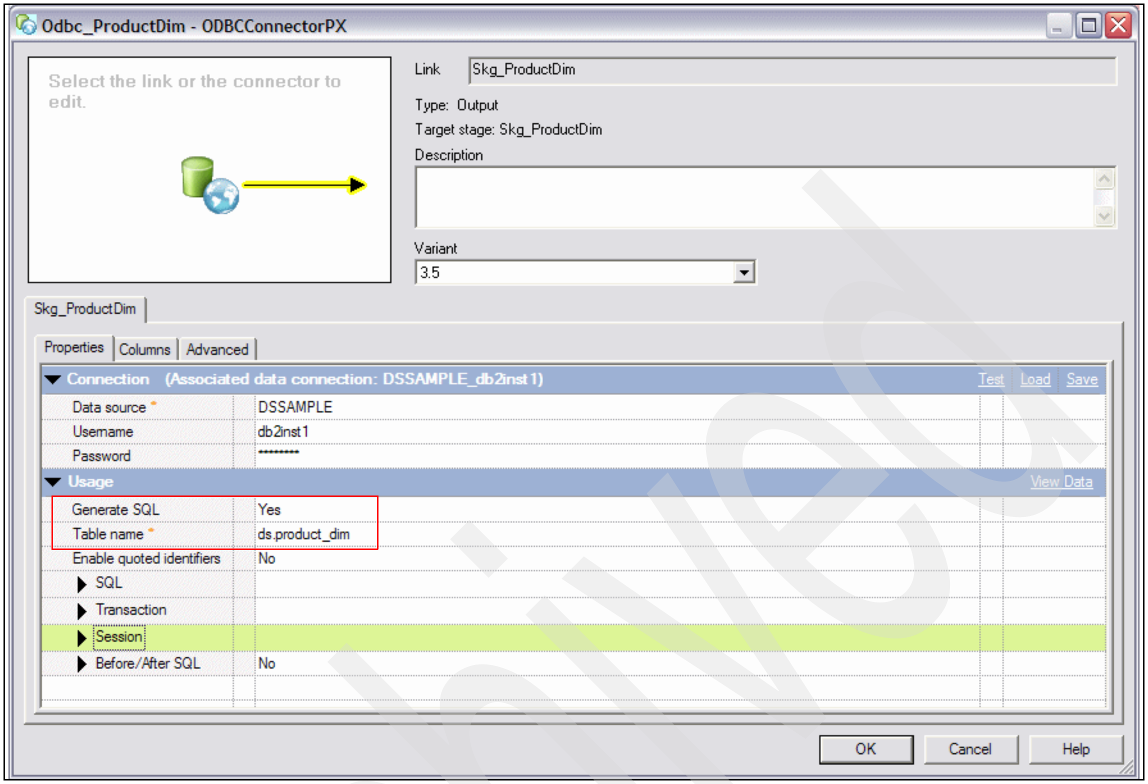


Figure 3-286 Create the J12_IL_GenerateSurrogateKey job 2/9

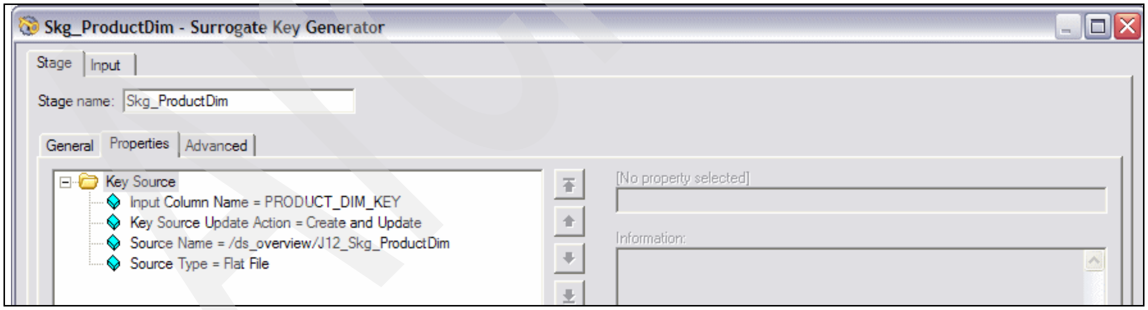


Figure 3-287 Create the J12_IL_GenerateSurrogateKey job 3/9

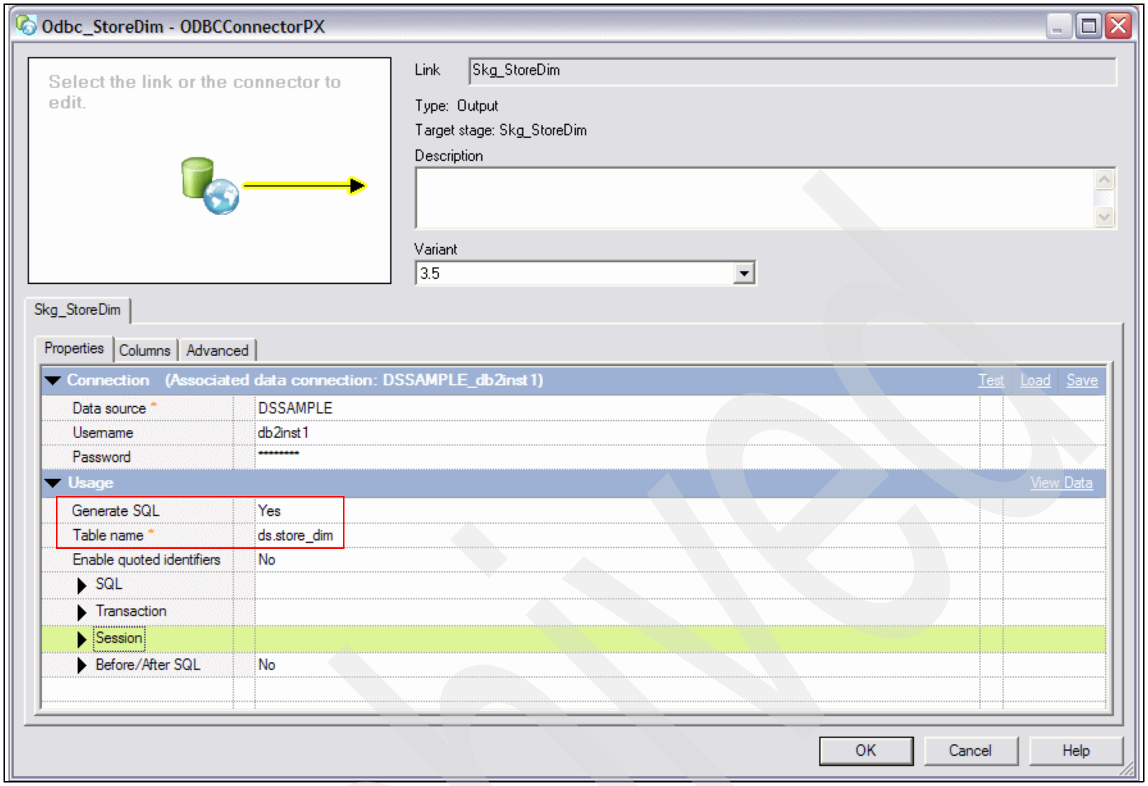


Figure 3-288 Create the J12_IL_GenerateSurrogateKey job 4/9

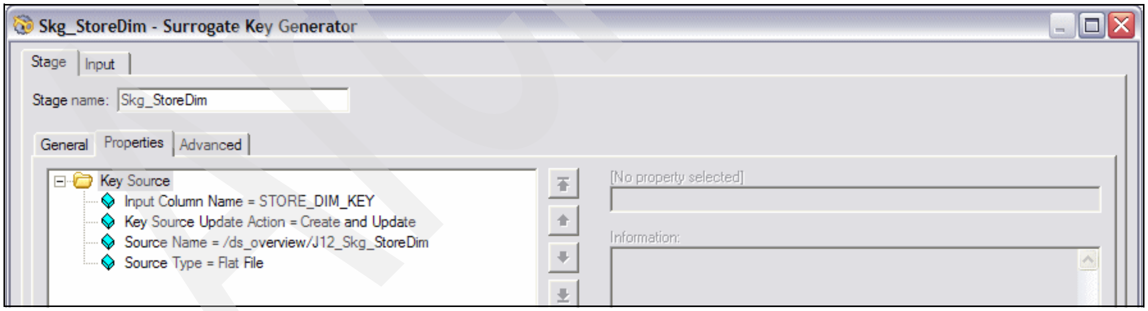


Figure 3-289 Create the J12_IL_GenerateSurrogateKey job 5/9

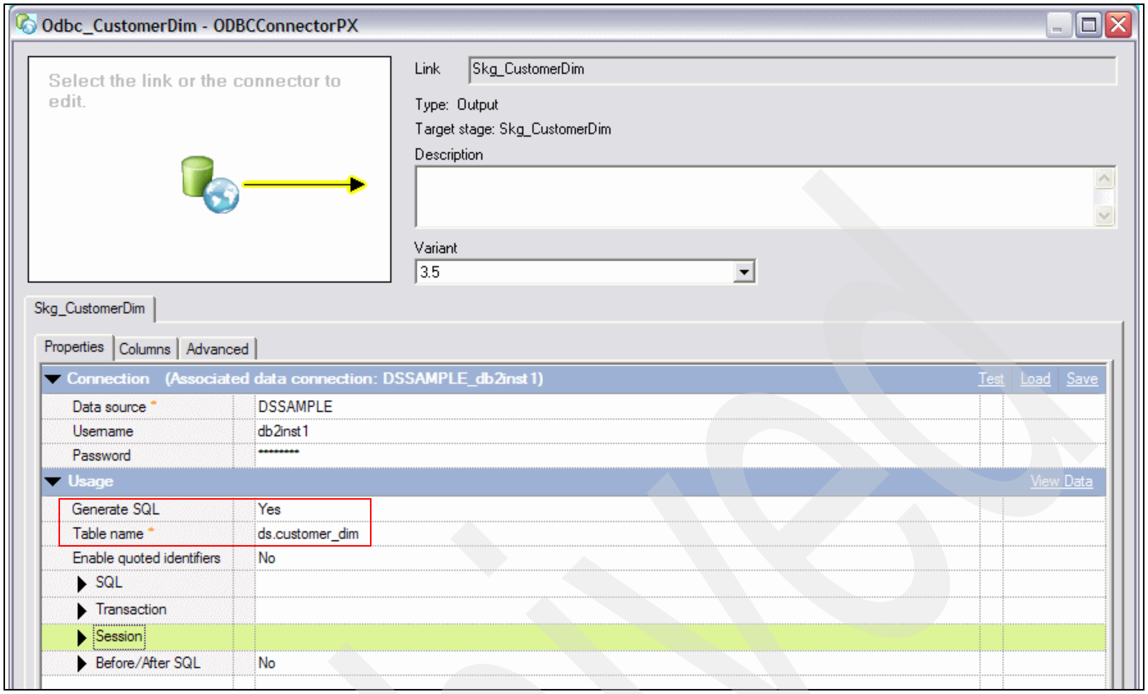


Figure 3-290 Create the J12_IL_GenerateSurrogateKey job 6/9

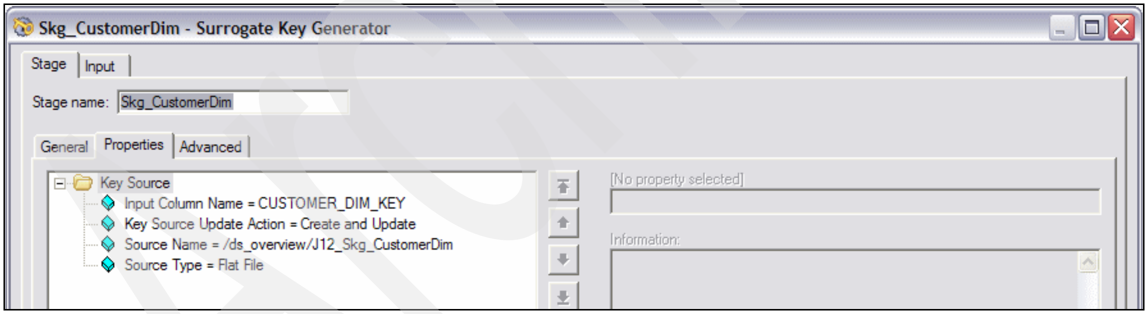


Figure 3-291 Create the J12_IL_GenerateSurrogateKey job 7/9

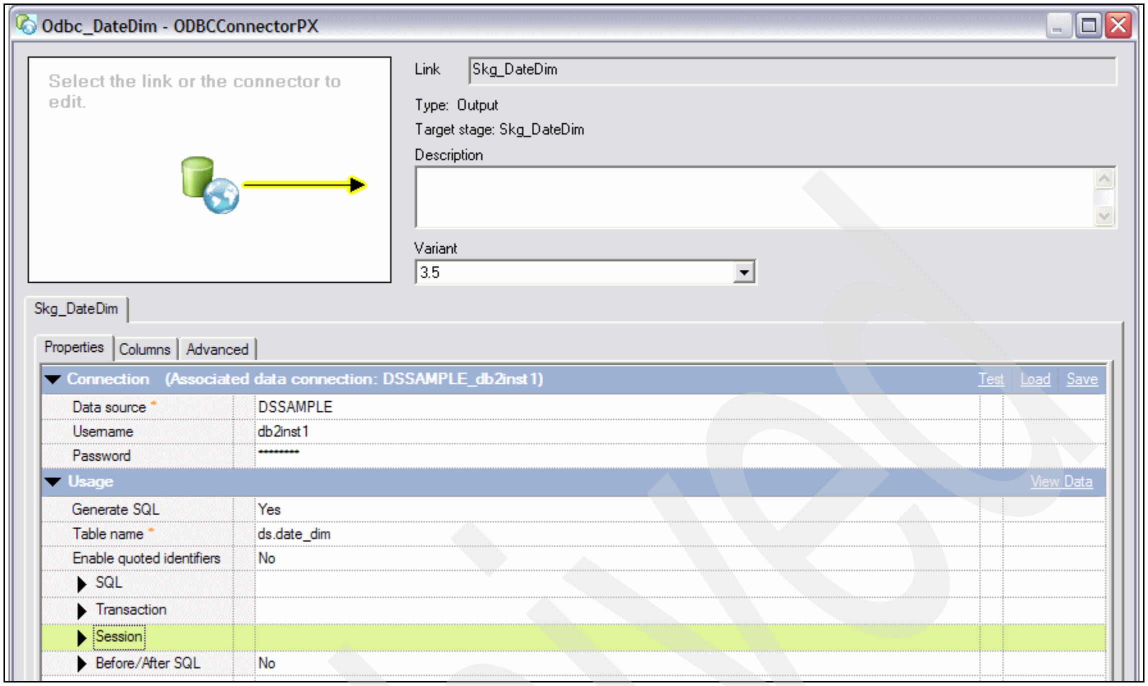


Figure 3-292 Create the J12_IL_GenerateSurrogateKey job 8/9

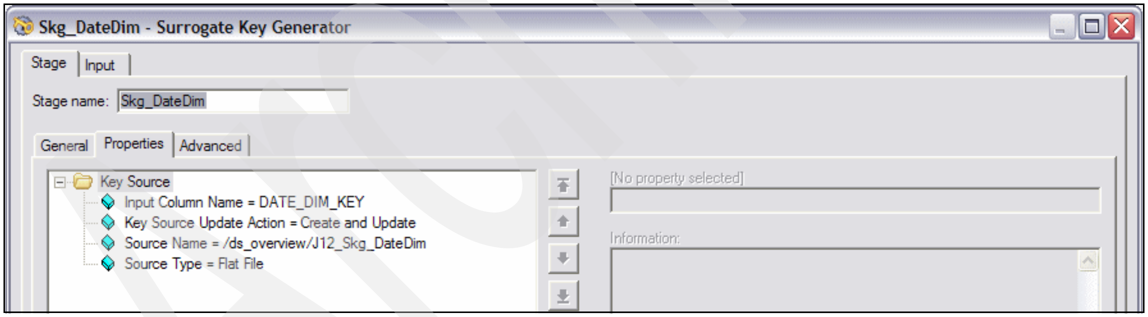


Figure 3-293 Create the J12_IL_GenerateSurrogateKey job 9/9

3.1.2 Recurring tasks

As mentioned earlier, the recurring (daily) tasks involve capturing dimension table changes and the sales transactions and preparing the information for updating the dimension tables and fact table over multiple update cycles, as follows:

1. Capture dimension table changes occurring in the operational OLTP systems.
2. Collect sales transactions from the stores from the operational OLTP systems.
3. Prepare the changes (to the dimension tables) for updating the dimension tables.
4. Prepare the sales transactions for updating the fact table.
5. Update the dimension tables.
6. Update the fact table.

In the following sections, we describe the jobs processing Day 1 on November 6th, 2007. In subsequent sections, we describe Day 2 processing on November 7th, 2007 (3.1.4, "Recurring tasks (Day 2)" on page 507) and Day 3 processing on November 8th, 2007 (3.1.5, "Recurring tasks (Day 3)" on page 537).

Note: We chose three (daily) processing cycles in order to showcase various scenarios as follows:

- Update to dimension tables that include Type 1 changes only, Type 2 changes only, and a combination of Type 1 and Type 2 changes.
- Sales transactions belonging to a previous version of the business key, that is, they do not correspond to the current version of the business key.
- Sales transactions that have some business keys that have no correspondence in the dimension tables,
- Late arriving dimensions (sales transactions with no corresponding business key entries in the dimension tables). This is a slight variation of the previous scenario.
- Dimension table changes with no corresponding sales transactions.

Table 3-2 lists the IBM InfoSphere DataStage jobs that we created to perform the recurring tasks identified earlier.

Table 3-2 Recurring (daily) tasks jobs

Job name	Brief description
"J06_IL_Daily_CreateCurrencyLookup_Service" on page 227	Downloads the daily exchange rate by country iso codes vis-a-vis the $US
"J07_IL_Daily_LoadSalesStore" on page 282	Loads the daily sales transactions of a store to a table
"J13_Daily_UpdateLookupDim (Day 1)" on page 356	Updates the dimension lookup tables with incoming Type 1 and/or Type 2 attribute changes
"J14_Daily_CreateAllSalesStoreDS (Day 1)" on page 385	Merge the sales transactions from all the stores
"J15_Daily_CreateSalesAggDS (Day 1)" on page 387	Associates dimension attributes from the lookup tables with the sales transactions, and aggregates sales transactions by quantity, foreign currency and US currency using the grouping of customer, produce, store, and date.
"J16_Daily_CreateScdInputDS (Day 1)" on page 421	Merges the aggregated sales transactions created in the "J15_Daily_CreateSalesAggDS (Day 1)" on page 387 job with the dimension table updates data sets (with nulls in the sales transaction columns) created in the "J13_Daily_UpdateLookupDim (Day 1)" on page 356 job. The result is in the format required as input to the SCD stage.
"J17_DailyCreateSalesFactDS (Day1)" on page 433	Creates the files to update dimension tables and the sales fact table in the star-schema using the SCD stage. Late arriving data is identified and written to a reject file.
"J18_Daily_UpdateStoreDim (Day 1)" on page 478	Updates the Store dimension table using the file created in the "J17_DailyCreateSalesFactDS (Day1)" on page 433 job
"J19_Daily_UpdateCustomerDim (Day 1)" on page 485	Updates the Customer dimension table using the file created in the "J17_DailyCreateSalesFactDS (Day1)" on page 433 job
"J20_Daily_UpdateProductDim (Day 1)" on page 494	Updates the Product dimension table using the file created in the "J17_DailyCreateSalesFactDS (Day1)" on page 433 job

Job name	Brief description
"J21_Daily_UpdateDateDim (Day 1)" on page 499	Updates the Date dimension table using the file created in the "J17_DailyCreateSalesFactDS (Day1)" on page 433 job
"J22_Daily_UpdateSalesFact (Day 1)" on page 502	Updates the Sales fact table using the file created in the "J17_DailyCreateSalesFactDS (Day1)" on page 433 job

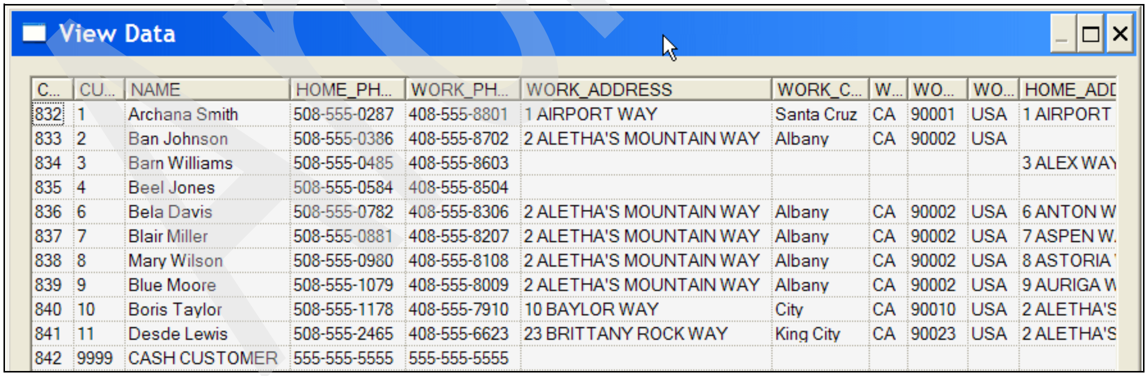
Here, we briefly describe each of these jobs:

- As described earlier in "J06_IL_Daily_CreateCurrencyLookup_Service" on page 227 job, this job describes the creation of a data set containing the exchange rates for different ISO country codes using a Web service.
- As described earlier in "J07_IL_Daily_LoadSalesStore" on page 282, this job computes the foreign currency equivalent for a sales transaction involving a non-US credit card and write it to an interim DB2 table.
- The "J13_Daily_UpdateLookupDim (Day 1)" on page 356 job retrieves changes to customer, product, and store attributes (Type 1 and Type 2) from an IBM WebSphere MQ queue and updates the dimension lookup tables (created in "J09_IL_LoadLookupCustomerDim" on page 320, "J10_IL_LoadLookupProductDim" on page 327, and "J11_IL_LoadLookupStoreDim" on page 330 jobs). It also creates a data set for each dimension table (with nulls in the sales transaction portion of the records — more on this later) for input to the SCD stage in the "J17_DailyCreateSalesFactDS (Day1)" on page 433 job.
- The "J14_Daily_CreateAllSalesStoreDS (Day 1)" on page 385 job merges the sales transactions from the individual stores into a single data set for subsequent processing to update the star-schema database.
- The "J15_Daily_CreateSalesAggDS (Day 1)" on page 387 job associates dimension attributes from the lookup tables with the sales transactions, and aggregates sales transactions by quantity, foreign currency, and US currency using the grouping of customer, produce, store, and date. Sales transactions corresponding to late arriving dimension updates and invalid business keys are identified and written to a reject file in this job.
- The "J16_Daily_CreateScdInputDS (Day 1)" on page 421 job creates a data set in the format required as input to the SCD stage in the "J17_DailyCreateSalesFactDS (Day1)" on page 433 job, by merging the aggregated sales transactions created in the J15_Daily_CreateSalesAggDS job with the dimension table updates data sets (with nulls in the sales transaction columns) created in the J13_Daily_UpdateLookupDim job.

- The “J17_DailyCreateSalesFactDS (Day1)” on page 433 job creates the files to update dimension tables and the sales fact table in the star-schema using the SCD stage. Late arriving data is identified and written to a reject file.
- The “J18_Daily_UpdateStoreDim (Day 1)” on page 478 job updates the Store dimension table with the DBCConnectorPX stage using the file created in the J17_Daily_CreateSalesFactDS job.
- The “J19_Daily_UpdateCustomerDim (Day 1)” on page 485 job updates the Customer dimension table with the DBCConnectorPX stage using the file created in the J17_Daily_CreateSalesFactDS job.
- The “J20_Daily_UpdateProductDim (Day 1)” on page 494 job updates the Product dimension table with the ODBCConnectorPX stage using the file created in the J17_Daily_CreateSalesFactDS job.
- The “J21_Daily_UpdateDateDim (Day 1)” on page 499 job updates the Date dimension table with the DBCConnectorPX stage using the file created in the J17_Daily_CreateSalesFactDS job.
- The “J22_Daily_UpdateSalesFact (Day 1)” on page 502 job updates the Sales fact table with the DBCConnectorPX stage using the file created in the J17_Daily_CreateSalesFactDS job.

The content of the dimension tables (excluding the Date dimension), the dimension lookup tables (excluding the Date dimension), and the Sales fact tables after the initial load (and just prior to the recurring daily cycle) is as follows:

- Dimension tables content
  – Customer dimension table (11 rows) is shown in Figure 3-294 through Figure 3-296.



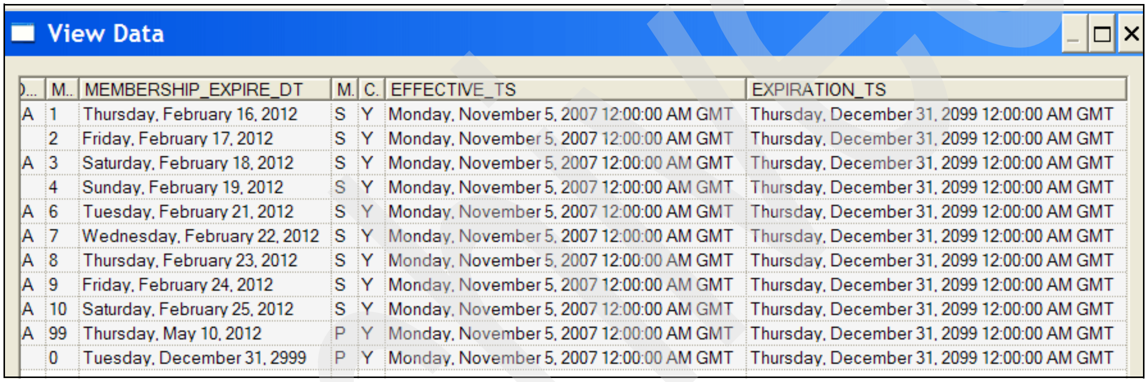
C...	CU...	NAME	HOME_PH...	WORK_PH...	WORK_ADDRESS	WORK_C...	W...	WO...	WO...	HOME_ADI
832	1	Archana Smith	508-555-0287	408-555-8801	1 AIRPORT WAY	Santa Cruz	CA	90001	USA	1 AIRPORT
833	2	Ban Johnson	508-555-0386	408-555-8702	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	
834	3	Barn Williams	508-555-0485	408-555-8603						3 ALEX WAY
835	4	Beel Jones	508-555-0584	408-555-8504						
836	6	Bela Davis	508-555-0782	408-555-8306	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	6 ANTON W
837	7	Blair Miller	508-555-0881	408-555-8207	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	7 ASPEN W.
838	8	Mary Wilson	508-555-0980	408-555-8108	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	8 ASTORIA '
839	9	Blue Moore	508-555-1079	408-555-8009	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	9 AURIGA W
840	10	Boris Taylor	508-555-1178	408-555-7910	10 BAYLOR WAY	City	CA	90010	USA	2 ALETHA'S
841	11	Desde Lewis	508-555-2465	408-555-6623	23 BRITTANY ROCK WAY	King City	CA	90023	USA	2 ALETHA'S
842	9999	CASH CUSTOMER	555-555-5555	555-555-5555						

Figure 3-294 Customer dimension table 1/3

View Data

HOME_ADDRESS	HOME_CITY	HO...	H...	HO...	M...	MEMBERSHIP_EXPIRE_DT	M.	C.	EFFECTIVE_TS
1 AIRPORT WAY	Santa Cruz	90001	CA	USA	1	Thursday, February 16, 2012	S	Y	Monday, November 5, 2007 12:0
					2	Friday, February 17, 2012	S	Y	Monday, November 5, 2007 12:0
3 ALEX WAY	Amador City	90003	CA	USA	3	Saturday, February 18, 2012	S	Y	Monday, November 5, 2007 12:0
					4	Sunday, February 19, 2012	S	Y	Monday, November 5, 2007 12:0
6 ANTON WAY	Bradbury	90006	CA	USA	6	Tuesday, February 21, 2012	S	Y	Monday, November 5, 2007 12:0
7 ASPEN WAY	Brawley	90007	CA	USA	7	Wednesday, February 22, 2012	S	Y	Monday, November 5, 2007 12:0
8 ASTORIA WAY	California City	90008	CA	USA	8	Thursday, February 23, 2012	S	Y	Monday, November 5, 2007 12:0
9 AURIGA WAY	Cathedral City	90009	CA	USA	9	Friday, February 24, 2012	S	Y	Monday, November 5, 2007 12:0
2 ALETHA'S MOUNTAIN WAY	Albany	90002	CA	USA	10	Saturday, February 25, 2012	S	Y	Monday, November 5, 2007 12:0
2 ALETHA'S MOUNTAIN WAY	Albany	90002	CA	USA	99	Thursday, May 10, 2012	P	Y	Monday, November 5, 2007 12:0
					0	Tuesday, December 31, 2999	P	Y	Monday, November 5, 2007 12:0

Figure 3-295 Customer dimension table 2/3

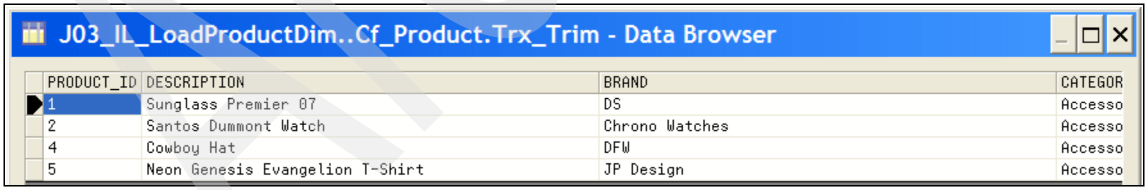


View Data

)...	M..	MEMBERSHIP_EXPIRE_DT	M.	C.	EFFECTIVE_TS	EXPIRATION_TS
A	1	Thursday, February 16, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
	2	Friday, February 17, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
A	3	Saturday, February 18, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
	4	Sunday, February 19, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
A	6	Tuesday, February 21, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
A	7	Wednesday, February 22, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
A	8	Thursday, February 23, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
A	9	Friday, February 24, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
A	10	Saturday, February 25, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
A	99	Thursday, May 10, 2012	P	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
	0	Tuesday, December 31, 2999	P	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT

Figure 3-296 Customer dimension table 3/3

– Product dimension table is shown in Figure 3-297 through Figure 3-299.



J03_IL_LoadProductDim..Cf_Product.Trx_Trim - Data Browser

PRODUCT_ID	DESCRIPTION	BRAND	CATEGOR
1	Sunglass Premier 07	DS	Accesso
2	Santos Dummont Watch	Chrono Watches	Accesso
4	Cowboy Hat	DFW	Accesso
5	Neon Genesis Evangelion T-Shirt	JP Design	Accesso

Figure 3-297 Product dimension 1/3

J03_IL_LoadProductDim..Cf_Product.Trx_Trim - Data Browser

CATEGORY	FACTORY	SUPPLIER
Accessories	The Factory	F&A Warehouse
Accessories	Chrono Watches	SCD
Accessories	Y'ALL	F&A Warehouse
Accessories	JP Design	F&A Warehouse

Figure 3-298 Product dimension 2/3

J03_IL_LoadProductDim..Cf_Product.Trx_Trim - Data Browser

SUPPLIER	SKU
F&A Warehouse	DS4321/07
SCD	CW2007/07
F&A Warehouse	DW1234/06
F&A Warehouse	JP0819/08

Figure 3-299 Product dimension 3/3

– Store dimension table (2 rows) is shown in Figure 3-300.

J05_IL_LoadStoreDim..Cf_Store.Joi_Store - Data Browser

STORE_ID	ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	MANAGER_ID
1	12345 Almaden Expressway	San Jose	00929936.	CA	33871648.	95118	USA	1
33	8976 Brazil Ave	San Francisco	00744041.	CA	33871648.	94112	USA	79

Figure 3-300 Store dimension

- Sales fact table contents

  Sales fact table (3 rows) is shown in Figure 3-301 and Figure 3-302.

Open Table - SALES_FACT

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.SALES_FACT

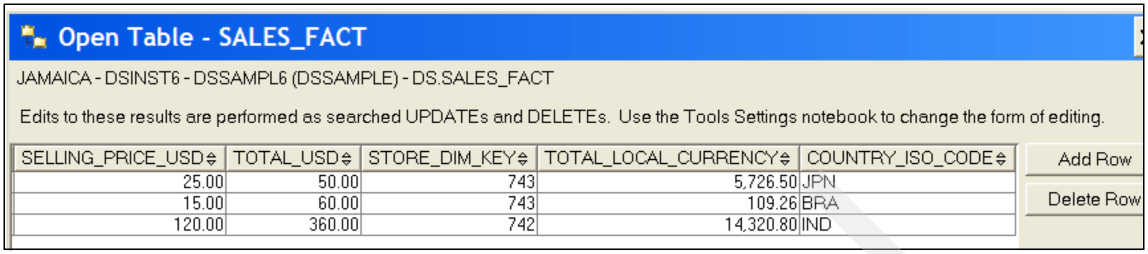
Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

CUSTOMER_DIM_KEY	DATE_DIM_KEY	PRODUCT_DIM_KEY	QUANTITY	PRICE_USD	SELLING_PRICE_
832	36	777	2	35.00	
836	36	777	4	17.69	
838	36	779	3	120.00	

Add Row

Delete Row

Figure 3-301 Sales fact table 1/2



Open Table - SALES_FACT

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.SALES_FACT

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

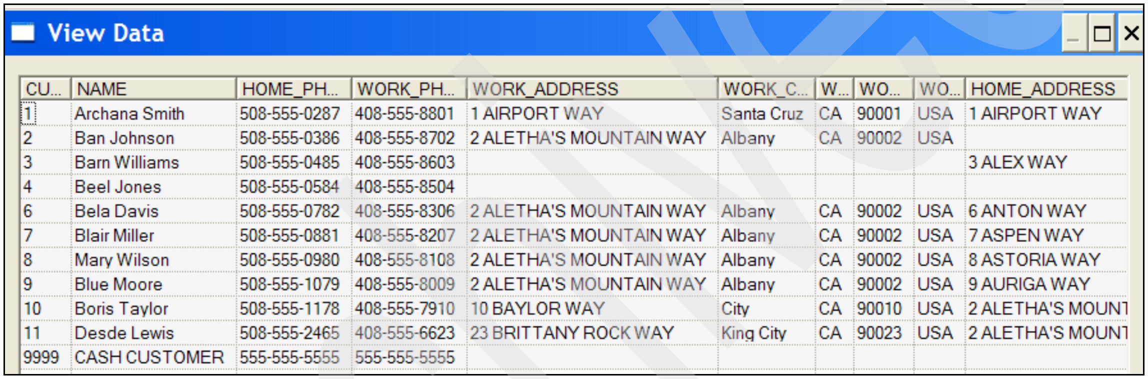
SELLING_PRICE_USD	TOTAL_USD	STORE_DIM_KEY	TOTAL_LOCAL_CURRENCY	COUNTRY_ISO_CODE
25.00	50.00	743	5,726.50	JPN
15.00	60.00	743	109.26	BRA
120.00	360.00	742	14,320.80	IND

Add Row

Delete Row

Figure 3-302 Sales fact table 2/2

- Dimension lookup tables content
  - Customer dimension lookup table (11 rows) is shown in Figure 3-303 and Figure 3-304.



View Data

CU...	NAME	HOME_PH...	WORK_PH...	WORK_ADDRESS	WORK_C...	W...	WO...	WO...	HOME_ADDRESS
1	Archana Smith	508-555-0287	408-555-8801	1 AIRPORT WAY	Santa Cruz	CA	90001	USA	1 AIRPORT WAY
2	Ban Johnson	508-555-0386	408-555-8702	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	
3	Barn Williams	508-555-0485	408-555-8603						3 ALEX WAY
4	Beel Jones	508-555-0584	408-555-8504						
6	Bela Davis	508-555-0782	408-555-8306	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	6 ANTON WAY
7	Blair Miller	508-555-0881	408-555-8207	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	7 ASPEN WAY
8	Mary Wilson	508-555-0980	408-555-8108	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	8 ASTORIA WAY
9	Blue Moore	508-555-1079	408-555-8009	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	9 AURIGA WAY
10	Boris Taylor	508-555-1178	408-555-7910	10 BAYLOR WAY	City	CA	90010	USA	2 ALETHA'S MOUNT
11	Desde Lewis	508-555-2465	408-555-6623	23 BRITTANY ROCK WAY	King City	CA	90023	USA	2 ALETHA'S MOUNT
9999	CASH CUSTOMER	555-555-5555	555-555-5555						

Figure 3-303 Customer dimension lookup table 1/2

View Data

DRESS	HOME_CITY	HO...	H...	HO...	M..	MEMBERSHIP_EXPIRE_DT	M.	TRANSACTION_TS
WAY	Santa Cruz	90001	CA	USA	1	Thursday, February 16, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
					2	Friday, February 17, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
Y	Amador City	90003	CA	USA	3	Saturday, February 18, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
					4	Sunday, February 19, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
VAY	Bradbury	90006	CA	USA	6	Tuesday, February 21, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
/AY	Brawley	90007	CA	USA	7	Wednesday, February 22, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
. WAY	California City	90008	CA	USA	8	Thursday, February 23, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
VAY	Cathedral City	90009	CA	USA	9	Friday, February 24, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
S MOUNTAIN WAY	Albany	90002	CA	USA	10	Saturday, February 25, 2012	S	Monday, November 5, 2007 12:00:00 AM GMT
S MOUNTAIN WAY	Albany	90002	CA	USA	99	Thursday, May 10, 2012	P	Monday, November 5, 2007 12:00:00 AM GMT
					0	Tuesday, December 31, 2999	P	Monday, November 5, 2007 12:00:00 AM GMT

Figure 3-304 Customer dimension lookup table 1/2

– Product dimension lookup table (4 rows) is shown in Figure 3-305.

View Data

PRODUCT_ID	DESCRIPTION	BRAND	CATEGORY	FACTORY	SUPPLIER	SKU	TRANSACTION_TS
5	Neon Genesis Evangelion T-Shirt	JP Design	Accessories	JP Design	F&A Warehouse	JP0819/08	Monday, November 5, 2007 12:00:00 AM GMT
1	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	Monday, November 5, 2007 12:00:00 AM GMT
2	Santos Dummont Watch	Chrono Watches	Accessories	Chrono Watches	SCD	CW2007/07	Monday, November 5, 2007 12:00:00 AM GMT
4	Cowboy Hat	DFW	Accessories	Y'ALL	F&A Warehouse	DW1234/06	Monday, November 5, 2007 12:00:00 AM GMT

Figure 3-305 Product dimension lookup table

– Store dimension lookup table (2 rows) is shown in Figure 3-306 and Figure 3-307.

View Data

STORE_ID	ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP
1	12345 Almaden Expressway	San Jose	929936	CA	33871648	95118
33	8976 Brazil Ave	San Francisco	744041	CA	33871648	94112

Figure 3-306 Store dimension lookup table 1/2

View Data

PULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	MANAGER_NAME	TRANSACTION_TS
	CA	33871648	95118	USA	Aidan Smith	Monday, November 5, 2007 12:00:00 AM GMT
	CA	33871648	94112	USA	Emma Hales	Monday, November 5, 2007 12:00:00 AM GMT

Figure 3-307 Store dimension lookup table 2/2

The three cycles of recurring tasks are described in 3.1.3, “Recurring tasks (Day 1)” on page 348, 3.1.4, “Recurring tasks (Day 2)” on page 507, and 3.1.5, “Recurring tasks (Day 3)” on page 537.

3.1.3 Recurring tasks (Day 1)

In this cycle, we processed the following data on November 6th, 2007:

- Dimension table changes
  - Customer dimension
    - Update (TABLE_CMD value of U) of CUSTOMER_ID 1

      Type 1 changes are the NAME (Arch Smith), WORK_ADDRESS (100 Air Road), and HOME_ADDRESS (2121 Carl St).

      There are no Type 2 changes.
    - Delete (TABLE_CMD value of D) the CUSTOMER_ID (7).

These are shown in Figure 3-308 through Figure 3-310.

J13_Daily_UpdateLookupDim..Ds_Customer.Ds_Customer - Data Browser

CUSTOMER_ID	NAME	HOME_PHONE	WORK_PHONE	WORK_ADDRESS
1	Arch Smith	508-555-0287	408-555-8801	100 AIR ROAD
7	Blair Miller	508-555-0881	408-555-8207	2 ALETHA'S MOUNTAIN WAY

Figure 3-308 Customer dimension attribute changes 1/3

J13_Daily_UpdateLookupDim..Ds_Customer.Ds_Customer - Data Browser

HOME_ADDRESS	HOME_CITY	HOME_ZIP	HI
2121 Carl St	Santa Cruz	90001	Ci
7 ASPEN WAY	Brawley	90023	Ci

Figure 3-309 Customer dimension attribute changes 2/3

J13_Daily_UpdateLookupDim..Ds_Customer.Ds_Customer - Data Browser

MEMBERSHIP_ID	MEMBERSHIP_EXPIRE_DT	MEMBERSHIP_LEVEL	TRANSACTION_TS	TABLE_CMD
1	2012-02-16	S	2007-11-06 12:39:42.445734	U
7	2012-02-22	S	2007-11-06 23:49:42.445734	D

Figure 3-310 Customer dimension attribute changes 3/3

– There are no changes to the Store, Product, and Date dimensions.

- Sales transactions

  Sales transactions are collected from three stores — ST1 (STORE_ID of 1) with 6 transactions as shown in Figure 3-311 here and Figure 3-312 on page 350, ST9 (STORE_ID of 9) with 1 transaction as shown in Figure 3-313 on page 350 and Figure 3-314 on page 350, and ST33 (STORE_ID of 33) with 6 transactions as shown in Figure 3-315 on page 350 and Figure 3-316 on page 351.

Open Table - SALES_ST1

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.SALES_ST1

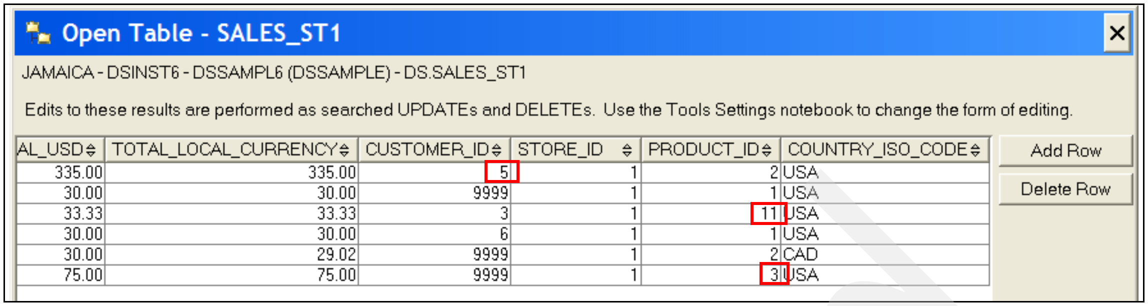
Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

SALES_ID	DATE	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	TOTAL_LOCA
83	Nov 6, 2007 7:5...	1	335.00	335.00	335.00	
94	Nov 6, 2007 10:...	2	17.69	15.00	30.00	
122	Nov 6, 2007 11:...	1	33.33	33.33	33.33	
124	Nov 6, 2007 11:...	2	17.69	15.00	30.00	
126	Nov 6, 2007 11:...	2	17.69	15.00	30.00	
129	Nov 6, 2007 11:...	1	75.00	75.00	75.00	

Add Row

Delete Row

Figure 3-311 STORE_ID 1 sales transactions 1/2



Open Table - SALES_ST1

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.SALES_ST1

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

AL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	COUNTRY_ISO_CODE
335.00	335.00	5	1	2	USA
30.00	30.00	9999	1	1	USA
33.33	33.33	3	1	11	USA
30.00	30.00	6	1	1	USA
30.00	29.02	9999	1	2	CAD
75.00	75.00	9999	1	3	USA

Add Row
Delete Row

Figure 3-312 STORE_ID 1 sales transactions 2/2

Open Table - SALES_ST9

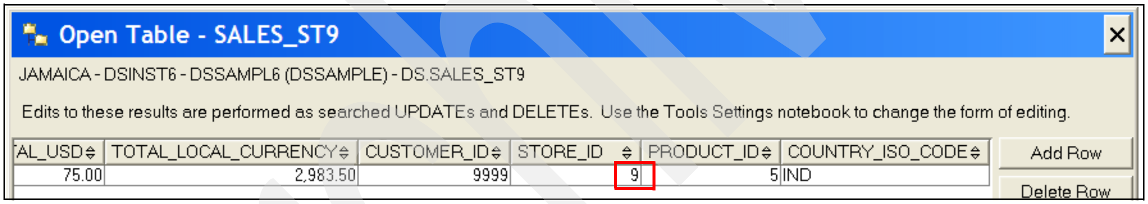
JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.SALES_ST9

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

SALES_ID	DATE	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	TOTAL_LOCA
95	Nov 6, 2007 6:3...	1	75.00	75.00	75.00	

Add Row

Figure 3-313 STORE_ID 9 sales transactions 1/2



Open Table - SALES_ST9

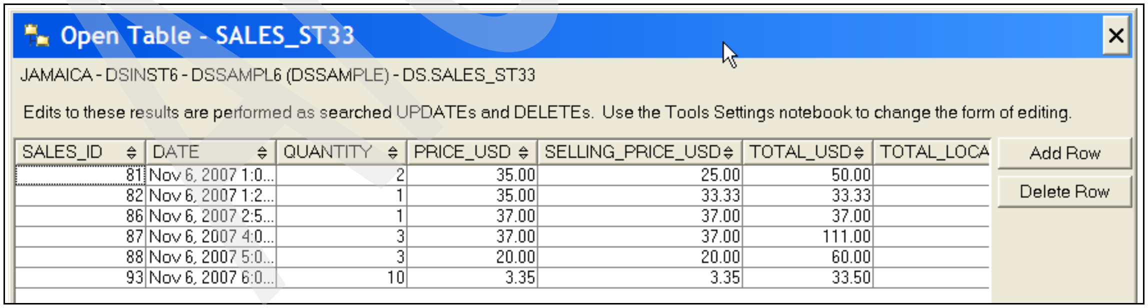
JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.SALES_ST9

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

AL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	COUNTRY_ISO_CODE
75.00	2,983.50	9999	9	5	IND

Add Row
Delete Row

Figure 3-314 STORE_ID 9 sales transactions 2/2



Open Table - SALES_ST33

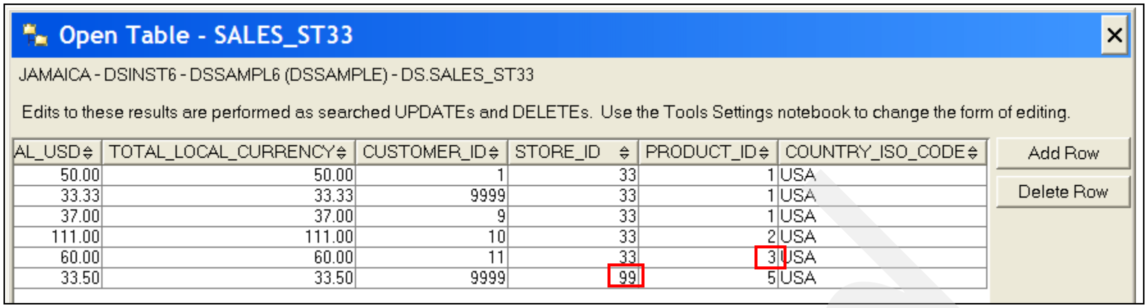
JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.SALES_ST33

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

SALES_ID	DATE	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	TOTAL_LOCA
81	Nov 6, 2007 1:0...	2	35.00	25.00	50.00	
82	Nov 6, 2007 1:2...	1	35.00	33.33	33.33	
86	Nov 6, 2007 2:5...	1	37.00	37.00	37.00	
87	Nov 6, 2007 4:0...	3	37.00	37.00	111.00	
88	Nov 6, 2007 5:0...	3	20.00	20.00	60.00	
93	Nov 6, 2007 6:0...	10	3.35	3.35	33.50	

Add Row
Delete Row

Figure 3-315 STORE_ID 33 sales transactions 1/2



AL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	COUNTRY_ISO_CODE
50.00	50.00	1	33	1	USA
33.33	33.33	9999	33	1	USA
37.00	37.00	9	33	1	USA
111.00	111.00	10	33	2	USA
60.00	60.00	11	33	3	USA
33.50	33.50	9999	99	5	USA

Figure 3-316 STORE_ID 33 sales transactions 2/2

Seven of these sales transactions were deliberately tailored to create the following error conditions, which result in all these transactions being written to a reject file corresponding to late arriving dimensions, since no matching business keys are found (for these records) in the appropriate dimension tables:

- STORE_ID of 9 and 99 do not exist in the Store dimension table.
- CUSTOMER_ID of 5 does not exist in the Customer dimension table.
- PRODUCT_ID of 3 and 11 do not exist in the Product dimension table.

These records are highlighted in Figure 3-311 on page 349 through Figure 3-316.

Note: We also did not have a sales transaction for CUSTOMER_ID of 7, which gets deleted in the operational system.

Table 3-2 on page 342 identifies the jobs executed in the recurring (daily) tasks, and the configuration and execution of these jobs are briefly described in the following sections starting with “J07_IL_Daily_LoadSalesStore” on page 282.

Note: “J06_IL_Daily_CreateCurrencyLookup_Service” on page 227 should be executed every day to pick up the latest exchange rates for each ISO country code. In our case, however, we created all the exchange rates for the different ISO country code countries for our three recurring daily cycles up front (during the initial load phase), and therefore do not repeat it here.

J07_IL_Daily_LoadSalesStore (Day 1)

As described in "J07_IL_Daily_LoadSalesStore" on page 282, this job computes the foreign currency equivalent for a sales transaction involving a non-US credit card and writes it to an interim DB2 table for subsequent processing prior to being loaded into the sales fact table.

Figure 3-317 on page 353 shows the various stages in the job. Since this was described in "J07_IL_Daily_LoadSalesStore" on page 282, it is not repeated here.

This job has to be repeated for sales transactions for each of the three stores (1, 9, and 33).

- Figure 3-318 on page 353 shows the Job Run Options window that identifies the input file (J07_Seq_Sales_20071106_ST1.txt) containing the sales transactions, the name of the schema file (J07_Seq_Sales_schema.osh as shown in Example 3-1 on page 353), and the name of the interim DB2 table (DS.SALES_ST1) to which these sales transactions are written.

  Figure 3-319 on page 354 shows the execution results of this job, indicating 6 sales transactions being processed.

  The contents of the DB2 interim table after the execution are shown in Figure 3-311 on page 349 and Figure 3-312 on page 350.

- Figure 3-320 on page 354 shows the Job Run Options window that identifies the input file (J07_Seq_Sales_20071106_ST9.txt) containing the sales transactions, the name of the schema file (J07_Seq_Sales_schema.osh), and the name of the interim DB2 table (DS.SALES_ST9) to which these sales transactions are written.

  Figure 3-321 on page 355 shows the execution results of this job, indicating 1 sales transaction being processed.

  The contents of the DB2 interim table after the execution are shown in Figure 3-313 on page 350 and Figure 3-314 on page 350.

- Figure 3-322 on page 355 shows the Job Run Options window that identifies the input file (J07_Seq_Sales_20071106_ST33.txt) containing the sales transactions, the name of the schema file (J07_Seq_Sales_schema.osh), and the name of the interim DB2 table (DS.SALES_ST33) to which these sales transactions are written.

  Figure 3-323 on page 356 shows the execution results of this job, indicating 6 sales transactions being processed.

  The contents of the DB2 interim table after the execution are shown in Figure 3-315 on page 350 and Figure 3-316 on page 351.

The next step is to execute the job described in "J13_Daily_UpdateLookupDim (Day 1)" on page 356.

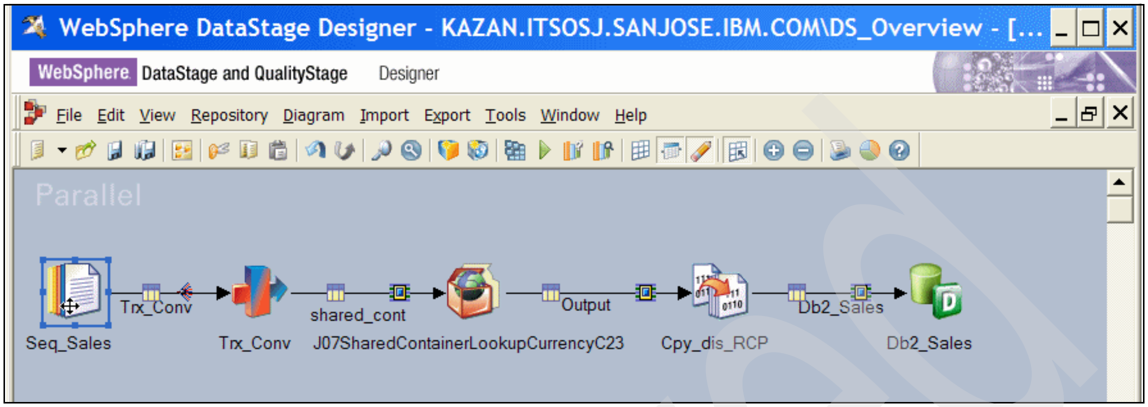


Figure 3-317 J07_IL_Daily_LoadSalesStore (Day 1) execution 1/7

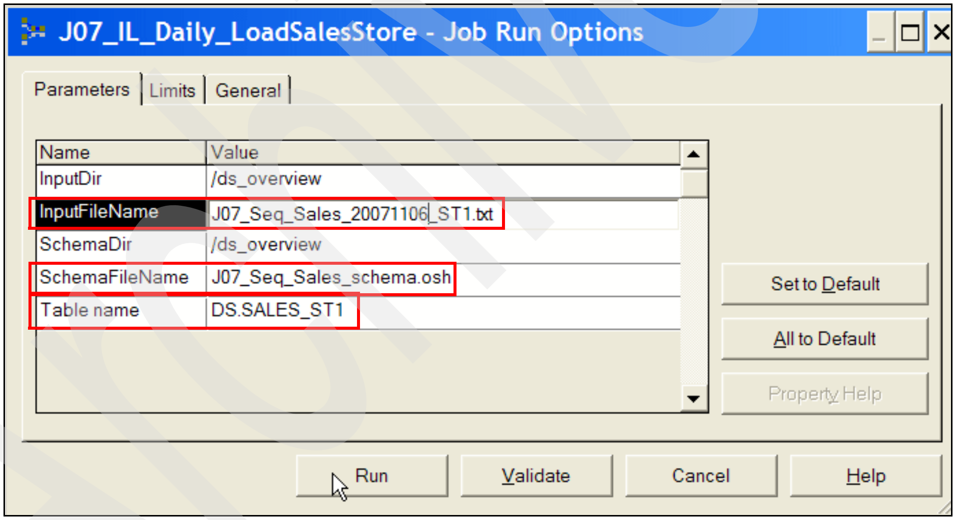


Figure 3-318 J07_IL_Daily_LoadSalesStore (Day 1) execution 2/7

Example 3-1 J07_Seq_Sales_schema.osh schema file

```
record
  {final_delim=end, record_delim='\n', delim=',', quote=double}
(
  SALES_ID:int32 {quote=none};
  DATE:nullable timestamp {null_field=''};
```

```
    QUANTITY:nullable int32 {quote=none, null_field=''};
    PRICE_USD:nullable decimal[10,2] {quote=none, null_field=''};
    SELLING_PRICE_USD:nullable decimal[10,2] {quote=none, null_field=''};
    TOTAL_USD:nullable decimal[10,2] {quote=none, null_field=''};
    TOTAL_LOCAL_CURRENCY:nullable decimal[10,2] {quote=none, null_field=''};
    CUSTOMER_ID:nullable int32 {quote=none, null_field=''};
    STORE_ID:nullable int32 {quote=none, null_field=''};
    PRODUCT_ID:nullable int32 {quote=none, null_field=''};
    COUNTRY_ISO_CODE:nullable string[3] {null_field=''};
)
```

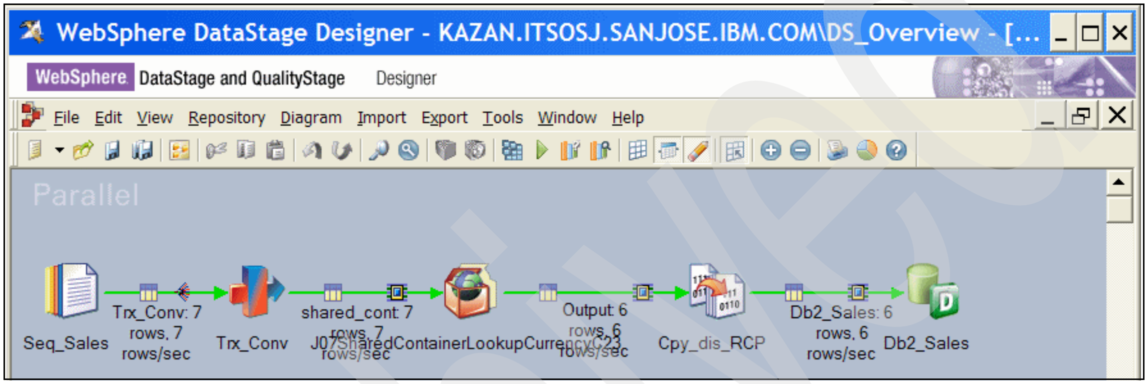


Figure 3-319 J07_IL_Daily_LoadSalesStore (Day 1) execution 3/7

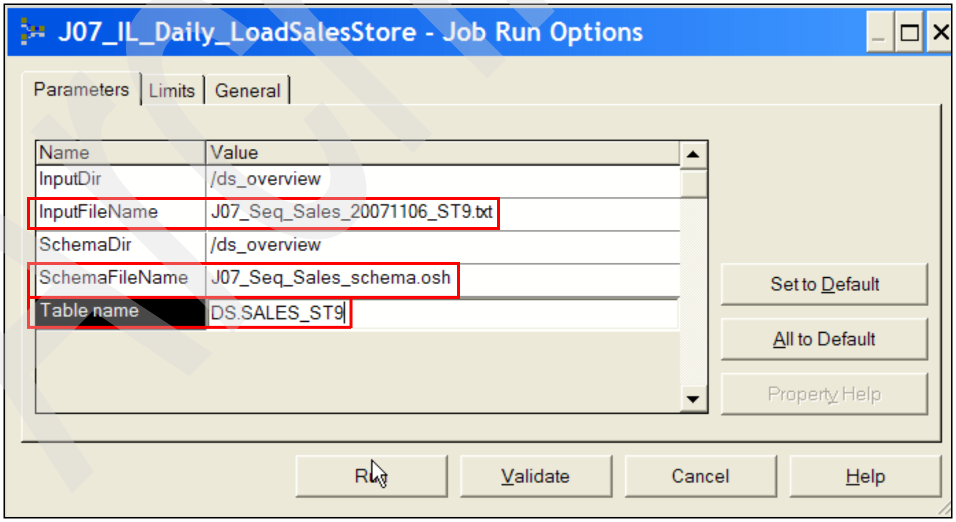


Figure 3-320 J07_IL_Daily_LoadSalesStore (Day 1) execution 4/7

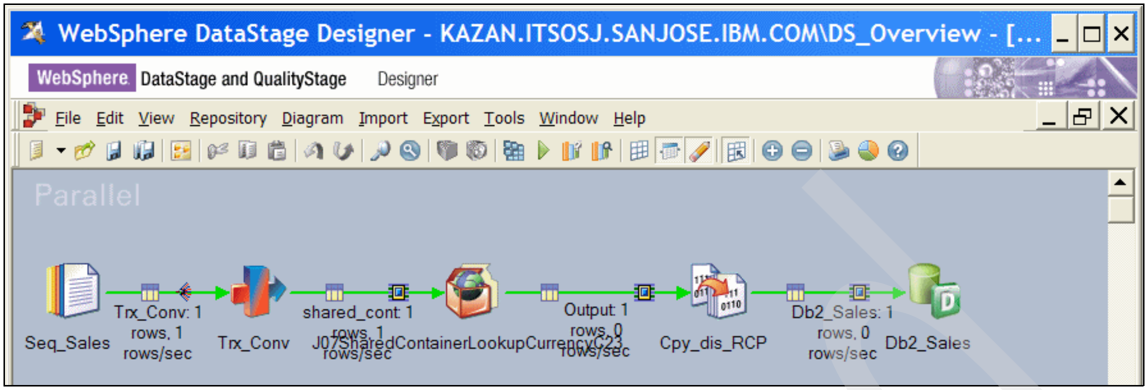


Figure 3-321 J07_IL_Daily_LoadSalesStore (Day 1) execution 5/7

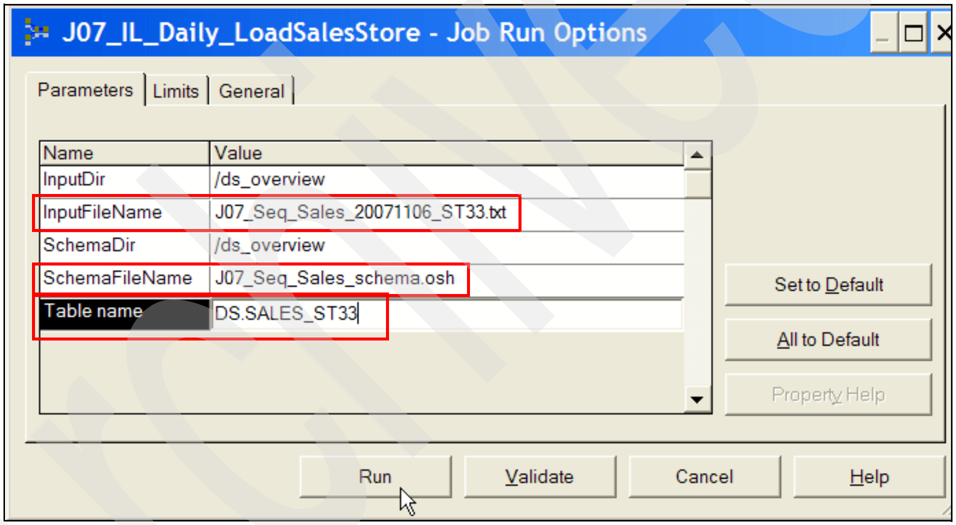


Figure 3-322 J07_IL_Daily_LoadSalesStore (Day 1) execution 6/7

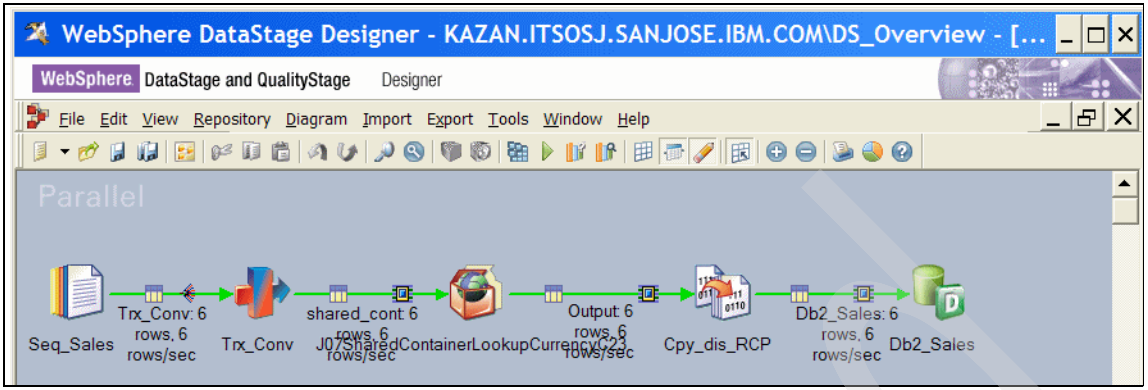


Figure 3-323 J07_IL_Daily_LoadSalesStore (Day 1) execution 7/7

J13_Daily_UpdateLookupDim (Day 1)

This job retrieves changes to customer, product, and store attributes (Type 1 and Type 2) from an IBM WebSphere MQ queue, and then:

1. Updates the dimension lookup tables (created in "J09_IL_LoadLookupCustomerDim" on page 320, "J10_IL_LoadLookupProductDim" on page 327, and "J11_IL_LoadLookupStoreDim" on page 330 jobs)
2. Creates a data set for each dimension table (with nulls in the sales transaction[5] portion of the records) for input to the SCD stage in the "J17_DailyCreateSalesFactDS (Day1)" on page 433 job.

Figure 3-325 on page 362 through Figure 3-351 on page 382 explain the main stages in this job and the configuration of these stages as described in "J13_Daily_UpdateLookupDim configuration" on page 356, while Figure 3-352 on page 383 through Figure 3-358 on page 387 explain the execution of this job with Day 1 input as described in "J13_Daily_UpdateLookupDim execution (Day 1)" on page 382.

J13_Daily_UpdateLookupDim configuration

Figure 3-325 on page 362 shows the various stages in the job — it includes a WebSphereMQConnectorPX stage, a Transformer stage, three sets of Funnel, Column Import, Copy, Data Set, Transformer, and Filter stages, and one DTStagePX stage. The names of the stages were modified as shown.

[5] This record is created to ensure that the dimension tables are updated in the SCD stage in "J17_DailyCreateSalesFactDS (Day1)" on page 433 even if there are no sales transactions associated with those dimension table changes. This is the late arriving (or no existing) sales transactions scenario where the dimension tables must be updated with the Type 1 and Type 2 attribute changes even when there are no incoming sales transactions in that daily cycle.

1. Figure 3-326 on page 363 and Figure 3-327 on page 364 show the configuration of the WebSphereMQConnectorPX stage which is used to access external data sources (message queues) in IBM WebSphere MQ enterprise messaging systems.

   Note: We assume that a process exists on the operational OLTP systems that captures changes occurring to the Customer, Product, and Store entities and writes them out to an IBM WebSphere MQ queue. This is not shown here. The format of the messages written to the queue are as shown in Figure 3-324 on page 357.

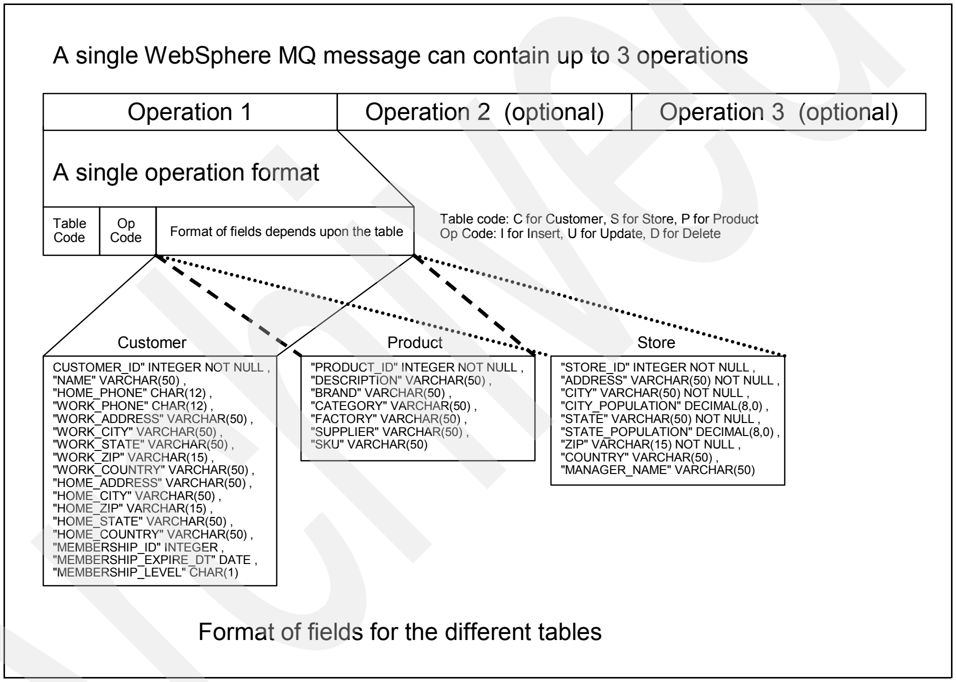


Figure 3-324 General format of IBM WebSphere MQ message

   Figure 3-326 on page 363 shows the **Properties** tab for the output Transform_Parse link that identifies the Connection details (Queue manager, Username and Password), the Queue name (SOURCEQ) and the Access mode[6] (As in queue definition). The Message read mode (Move to work queue) specifies that after the message is read it is removed from the SOURCEQ and moved to the work queue.

[6] This specifies that the queue is opened by using the default access as defined for that queue. This is described in “Create the queues” on page 591.

Figure 3-327 on page 364 shows the **Columns** tab which allows you to define the column metadata for the selected output link. It shows two defined columns — Body (SQL type of Varchar 2000) and DTS_msgID (SQL type of Binary). The Runtime column propagation box is not checked.

2. Figure 3-328 on page 365 shows the Transformer Stage window that processes the input from the WebSphereMQConnectorPX stage and splits the records to nine different outputs depending upon the table and the type of operation involved. There are 3 tables (Customer, Product and Store), and each table can have an insert, update, or delete operation — making up a total of nine output links.

   ‘Example 3-2 on page 365 shows the stage variables defined in the Transformer Stage window in Figure 3-328 on page 365.

   Figure 3-328 on page 365 also shows the mapping of the columns from the incoming message to the Fnl_ParseCustomer_1 output link which has the constraint svCustomerTablePart1 = “‘Y”. Based on the stage variables defined, this Fnl_ParseCustomer_1 output link will contain Customer table records generated from the first transaction in the IBM WebSphere MQ message as long as the TABLE_CMD column has one of the three values ‘I’, ‘U’, or ‘D’ corresponding to an SQL INSERT, UPDATE, or DELETE operation. Fnl_ParseCustomer_2 will contain Customer table records from the second transaction (if any) in the IBM WebSphere MQ message, while Fnl_ParseCustomer_3 will contain Customer table records from the third transaction (if any) in the IBM WebSphere MQ message.

   Figure 3-329 on page 367 shows the Preserves sort order box checked to ensure that the output link has the records written in the same order as the incoming records. This ensures that the sequence of update operations in the source are maintained.

   Figure 3-330 on page 367 shows the constraints associated with each of the nine output links.

3. The three Funnel stages shown in Figure 3-325 on page 362 merge the transactions from the three output links associated with each table into a single output link for each table. This is not shown here since it is similar to other Funnel stage configurations described earlier.

4. Figure 3-331 on page 368 through Figure 3-334 on page 370 show the configuration of the Column Import stage that imports data from a single column and outputs it to one or more columns.

   – Figure 3-331 on page 368 shows the **Properties** tab in the Stage page, which identifies the input column in the Import Input Column property (body_customer) of the Input category.

     The Output category identifies the output columns to which the input column is mapped to.

   - Figure 3-332 on page 368 shows the **Columns** tab in the Input page that defines the metadata of the incoming data.
   - Figure 3-333 on page 369 shows the **Mapping** tab in the Output page, which maps the input columns to the output Cpy_Customer link.
   - Figure 3-334 on page 370 shows the **Columns** tab in the Output page, which defines the metadata of the of the output columns.
5. Figure 3-335 on page 370 and Figure 3-336 on page 371 show the configuration of the Copy stage that essentially copies the same records into two links — one of which is a Data Set stage and the other as input to a Transformer stage.
   - Figure 3-335 on page 370 shows the **Mapping** tab in the Output page, which maps the input columns to the output Trx_Customer link.
   - Figure 3-336 on page 371 shows the **Columns** tab in the Output page, which defines the metadata of the of the output columns in the Trx_Customer link.

   The same mapping and column definitions apply to the Ds_Customer link — this is not shown here.
6. Figure 3-337 on page 371 shows the configuration of the Data Set stage. It shows the **Properties** stage in the Input page that defines the output files name (J13_Customer.ds) and an overwrite update policy.
7. Figure 3-338 on page 372 shows the Transformer Stage window, which adds a column DTS_String_TimeStamp to the output link (Fltr_Customer) that is derived from the timestamp corresponding to when the transaction was executed in the OLTP system. This column is a duplicate of the input TRANSACTION_TS column. This new column is used to sort all the transactions (in the subsequent DTStagePX stage) in the sequence they executed in the OLTP system to ensure that the sequence is faithfully replicated. We have also configured the Transformer stage output to preserve the sort order of the incoming data — this is not shown here.

   Note: We could have chosen to use the existing TRANSACTION_TS column for this sort purpose, but we chose to call attention to the method by creating a separate column.

8. Figure 3-339 on page 372 through Figure 3-343 on page 374 shows the configuration of a Filter stage that directs inserts, updates, and deletes to separate links for each dimension table.

   – Figure 3-339 on page 372 shows the **Properties** tab in the Stage page, which specifies the Where Clause property in the Predicates category. The TABLE_CMD column identifies the SQL operation that is used to direct the records to the appropriate output link.

     The **Options** category has two properties:

     • The Output Rejects = False property indicates that rows that fail all the predicates should not be sent to the reject link (one was not defined by us).

     • The Output Rows Only Once = False specifies that rows are output down the links of all Where clauses that they satisfy.

   – Figure 3-340 on page 373 shows the **Link Ordering** tab in the Stage page that shows how the qualifying rows are directed to the appropriate output link.

   – Figure 3-341 on page 373 shows the **Columns** tab in the Input page that defines the metadata definitions of the incoming data.

   – Figure 3-342 on page 374 shows the **Mapping** tab in the Output page (for the Customer_Insert link) that maps all the columns in the input to the output.

     Figure 3-343 on page 374 shows the **Columns** tab in the Output page that maps all the columns in the input to the output. It confirms all the columns being mapped.

     The same applies to the other two output links Customer_Update and Customer_Delete.

9. Figure 3-344 on page 375 through Figure 3-351 on page 382 show the configuration of the distributed transaction stage DTStagePX that processes all the dimension update rows in the 9 input links in the order in which they were generated in the source OLTP system and updates the corresponding dimension lookup tables Customer, Product and Store.

– Figure 3-344 on page 375 shows the **Properties** tab in the Stage page of the DTStagePX window for the Customer_Insert input link that specifies the Order messages[7] property of Yes which indicates that the messages should be processed in sequence across the various links.

  The rows in the various links are sorted in ascending sequence of DTS_String_TimeStamp as shown in Figure 3-348 on page 379 for the Customer_Insert link.

– Figure 3-345 on page 376 shows the configuration of the **Properties** tab of the Customer_Insert input link which shows the Write mode (Insert) for the SQL statement and Generate SQL No. A portion of the manually generated SQL is shown.

– Figure 3-346 on page 377 shows the **Link Ordering** tab for the Customer_Insert link that identifies and orders all the links.

– Figure 3-347 on page 378 shows the **Columns** tab for the Customer_Insert input link that specifies the metadata of all the columns in the incoming data.

– Figure 3-348 on page 379 shows the **Partitioning** tab for the Customer_Insert input link that specifies a Partition type (Hash) and a sort of the rows in the input in ascending sequence of the DTS_String_TimeStamp.

– Figure 3-349 on page 380 shows the configuration of the **Properties** tab of the Customer_Update input link which shows the Write mode (Update) for the SQL statement and Generate SQL No. Figure 3-350 on page 381 shows the manually generated SQL is shown.

– Figure 3-351 on page 382 shows the configuration of the **Properties** tab of the Customer_Delete input link which shows the Write mode (Delete) for the SQL statement and Generate SQL No. A portion of the manually generated SQL is shown.

The results of the execution of this job on Day 1 are described in "J13_Daily_UpdateLookupDim execution (Day 1)" on page 382.

[7] Also sometimes referred to as cross-link ordering.

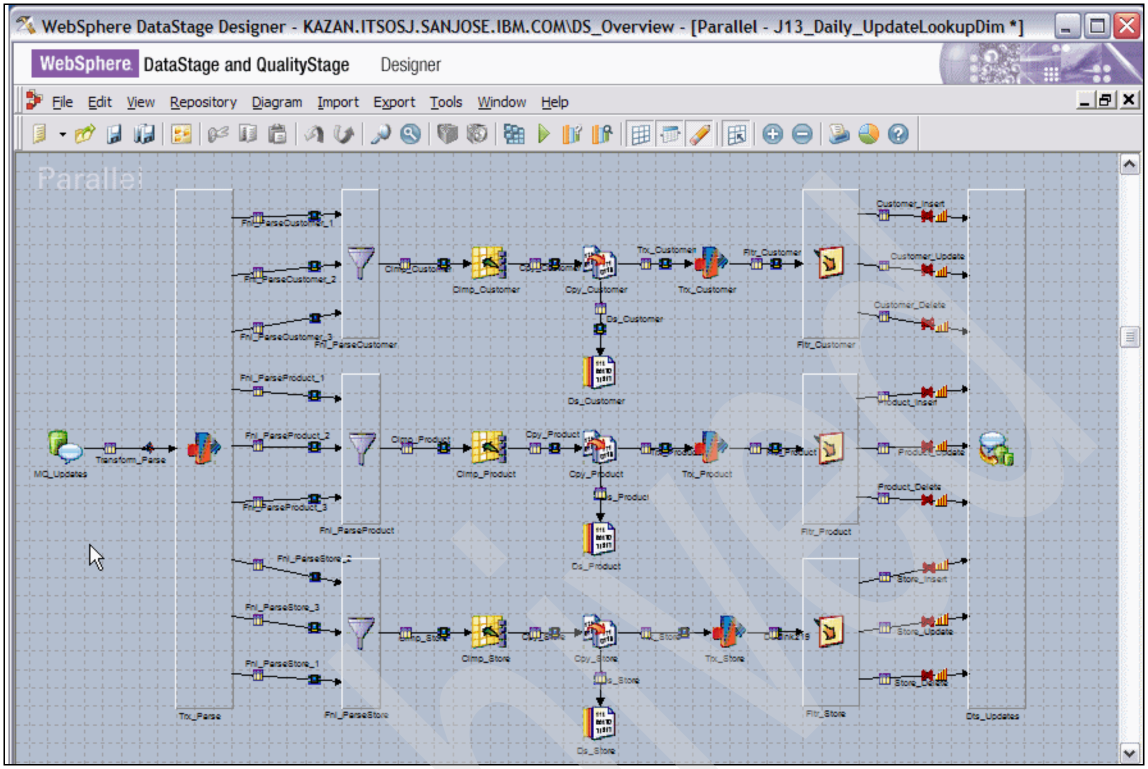


Figure 3-325 Create the J13_Daily_UpdateLookupDim job 1/26

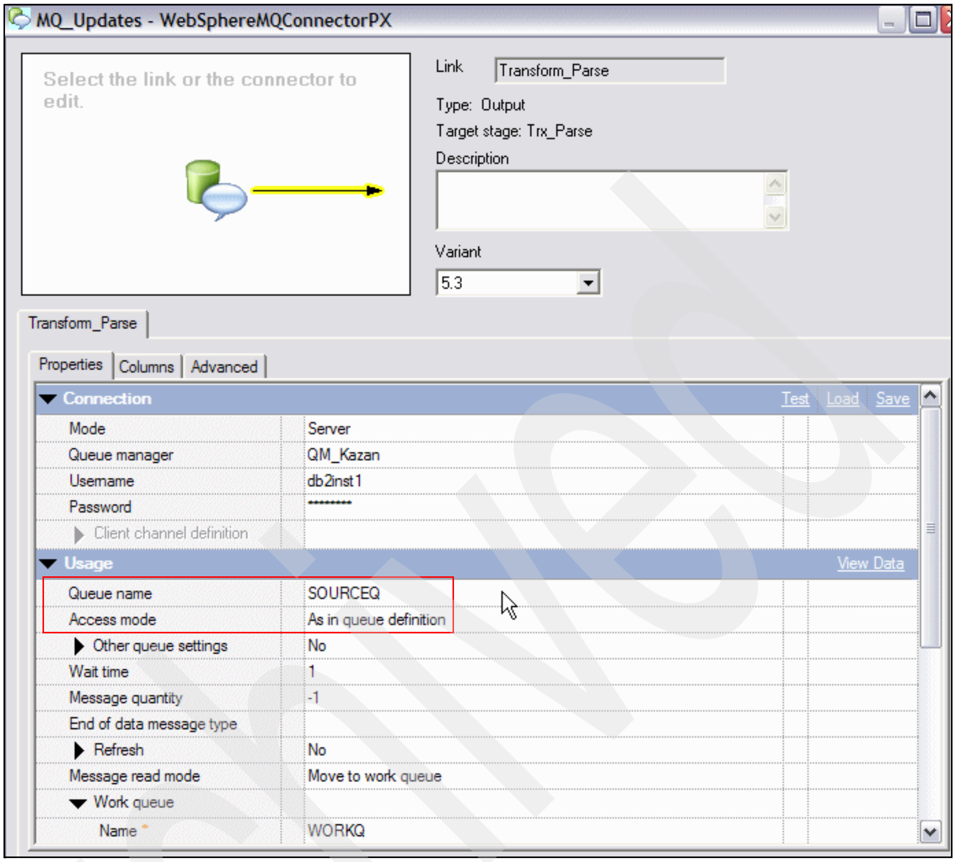


Figure 3-326 Create the J13_Daily_UpdateLookupDim job 2/26

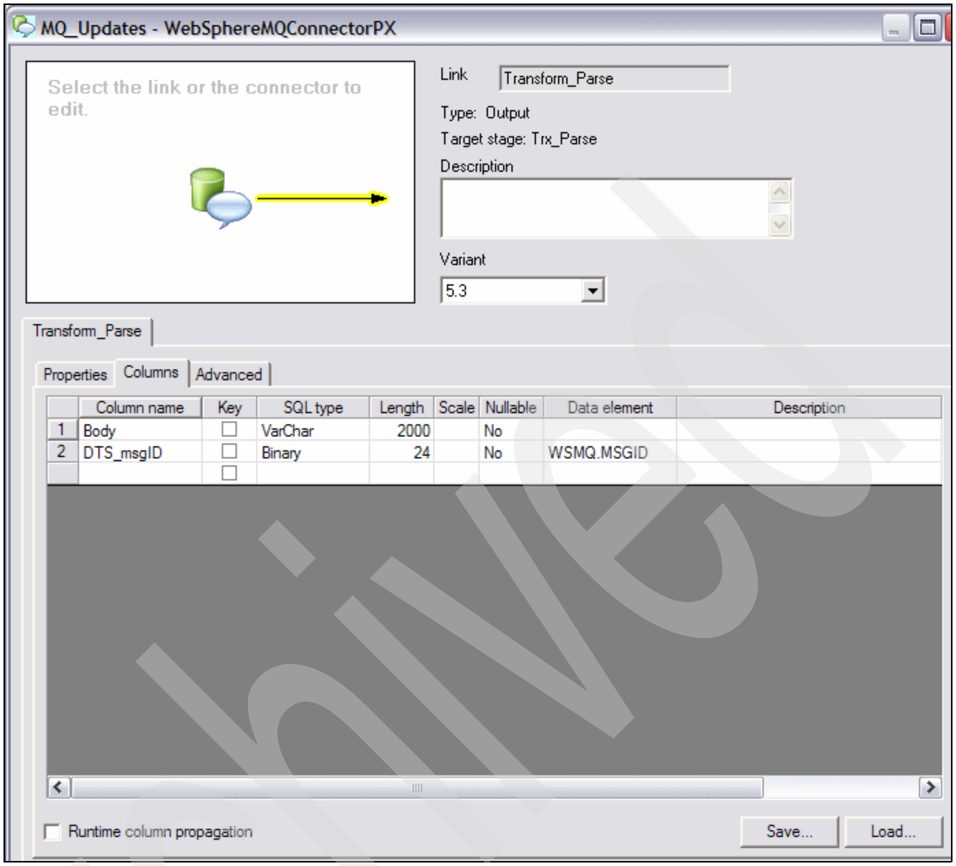


Figure 3-327 Create the J13_Daily_UpdateLookupDim job 3/26

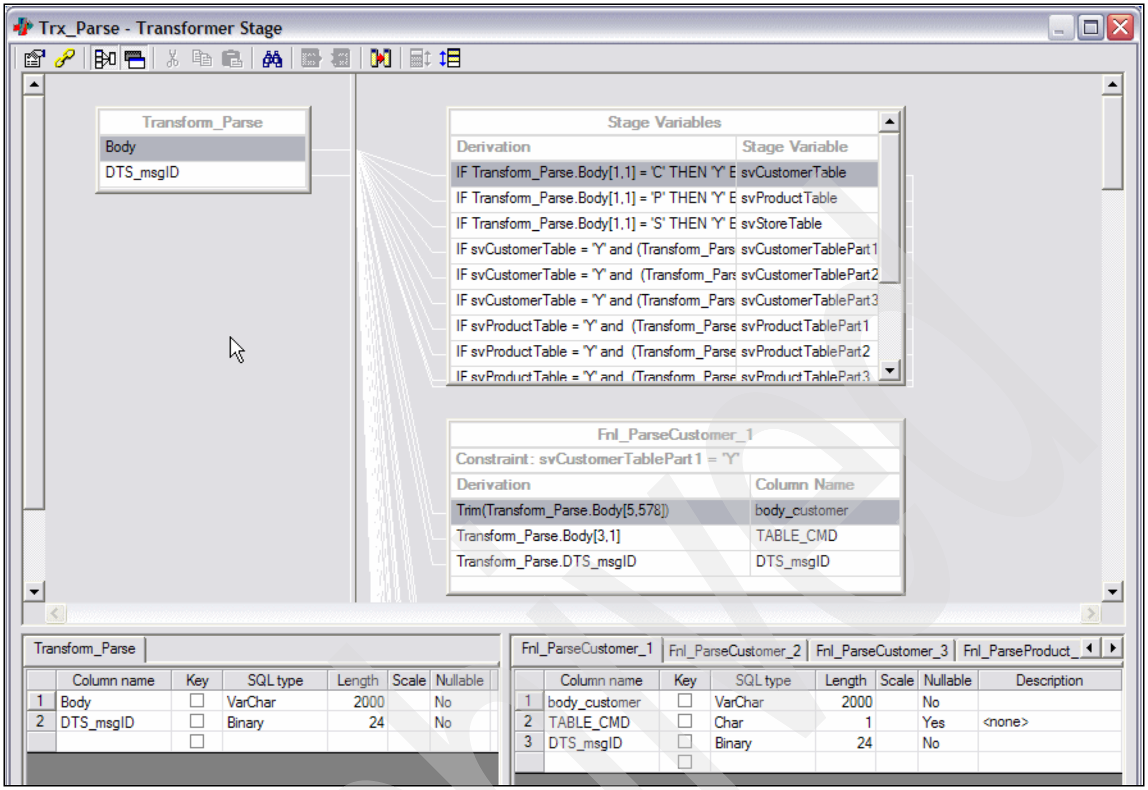


Figure 3-328 Create the J13_Daily_UpdateLookupDim job 4/26

Example 3-2 Derivation of stage variables

```
svCustomerTable:
IF Transform_Parse.Body[1,1] = 'C' THEN 'Y' ELSE 'N'

svProductTable:
IF Transform_Parse.Body[1,1] = 'P' THEN 'Y' ELSE 'N'

svStoreTable:
IF Transform_Parse.Body[1,1] = 'S' THEN 'Y' ELSE 'N'

svCustomerTablePart1:
IF svCustomerTable = 'Y' and (Transform_Parse.Body[3,1] = 'I' or Transform_Parse.Body[3,1] =
'U' or Transform_Parse.Body[3,1] = 'D') then 'Y' else 'N'

svCustomerTablePart2:
```

```
IF svCustomerTable = 'Y' and  (Transform_Parse.Body[584,1] = 'I' or Transform_Parse.Body[584,1]
= 'U' or Transform_Parse.Body[584,1] = 'D') then 'Y' else 'N'

svCustomerTablePart3:
IF svCustomerTable = 'Y' and (Transform_Parse.Body[1165,1] = 'I' or
Transform_Parse.Body[1165,1] = 'U' or Transform_Parse.Body[1165,1] = 'D')  then 'Y' else 'N'

svProductTablePart1:
IF svProductTable = 'Y' and  (Transform_Parse.Body[3,1] = 'I' or Transform_Parse.Body[3,1] =
'U' or Transform_Parse.Body[3,1] = 'D') then 'Y' else 'N'

svProductTablePart2:
IF svProductTable = 'Y' and  (Transform_Parse.Body[349,1] = 'I' or Transform_Parse.Body[349,1]
= 'U' or Transform_Parse.Body[349,1] = 'D') then 'Y' else 'N'

svProductTablePart3:
IF svProductTable = 'Y' and  (Transform_Parse.Body[695,1] = 'I' or Transform_Parse.Body[695,1]
= 'U' or Transform_Parse.Body[695,1] = 'D') then 'Y' else 'N'

svStoreTablePart1:
IF svStoreTable = 'Y' and  (Transform_Parse.Body[3,1] = 'I' or Transform_Parse.Body[3,1] = 'U'
or Transform_Parse.Body[3,1] = 'D')  then 'Y' else 'N'

svStoreTablePart2:
IF svStoreTable = 'Y' and  (Transform_Parse.Body[332,1] = 'I' or Transform_Parse.Body[332,1] =
'U' or Transform_Parse.Body[332,1] = 'D') then 'Y' else 'N'

svStoreTablePart3:
IF svStoreTable = 'Y' and  (Transform_Parse.Body[661,1] = 'I' or Transform_Parse.Body[661,1] =
'U' or Transform_Parse.Body[661,1] = 'D')  then 'Y' else 'N'
```

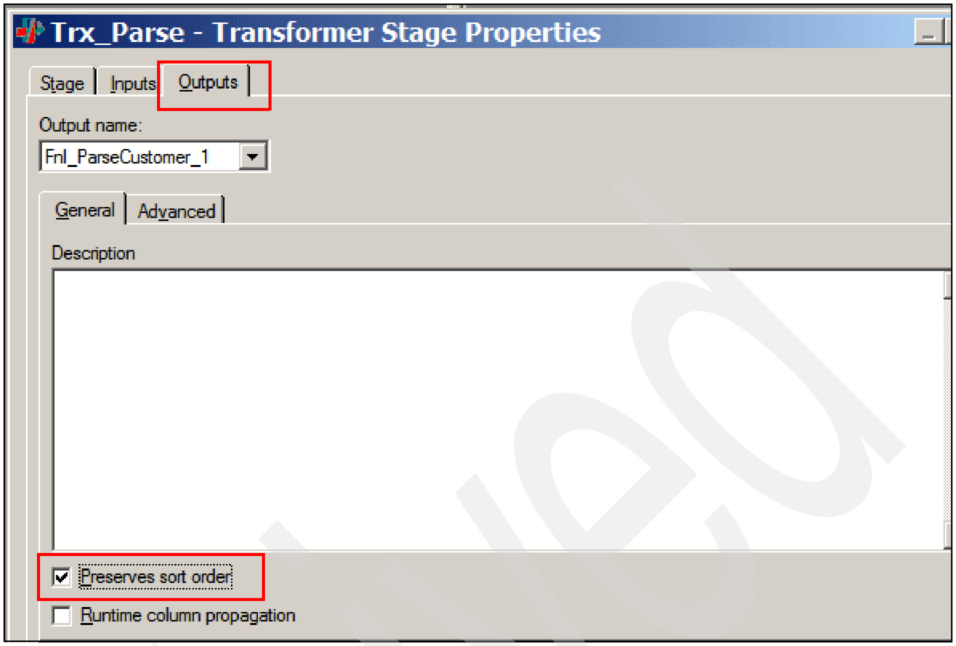


Figure 3-329 Create the J13_Daily_UpdateLookupDim job

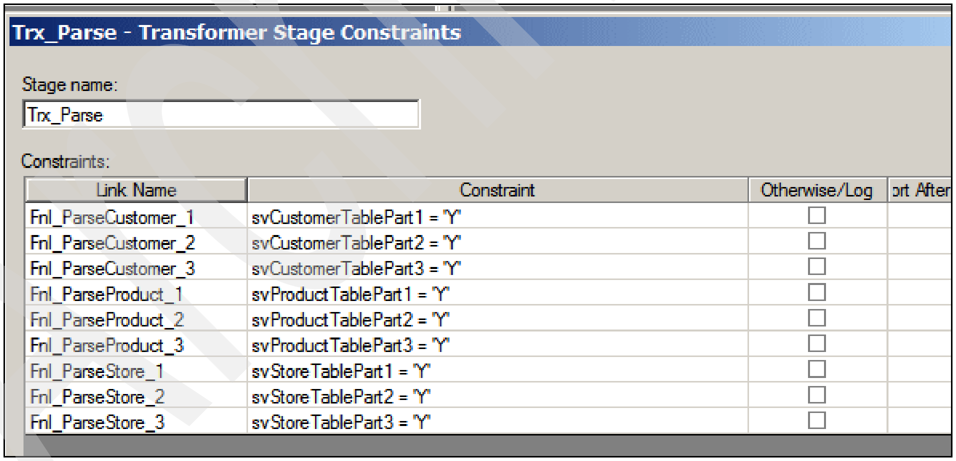


Figure 3-330 Create the J13_Daily_UpdateLookupDim job

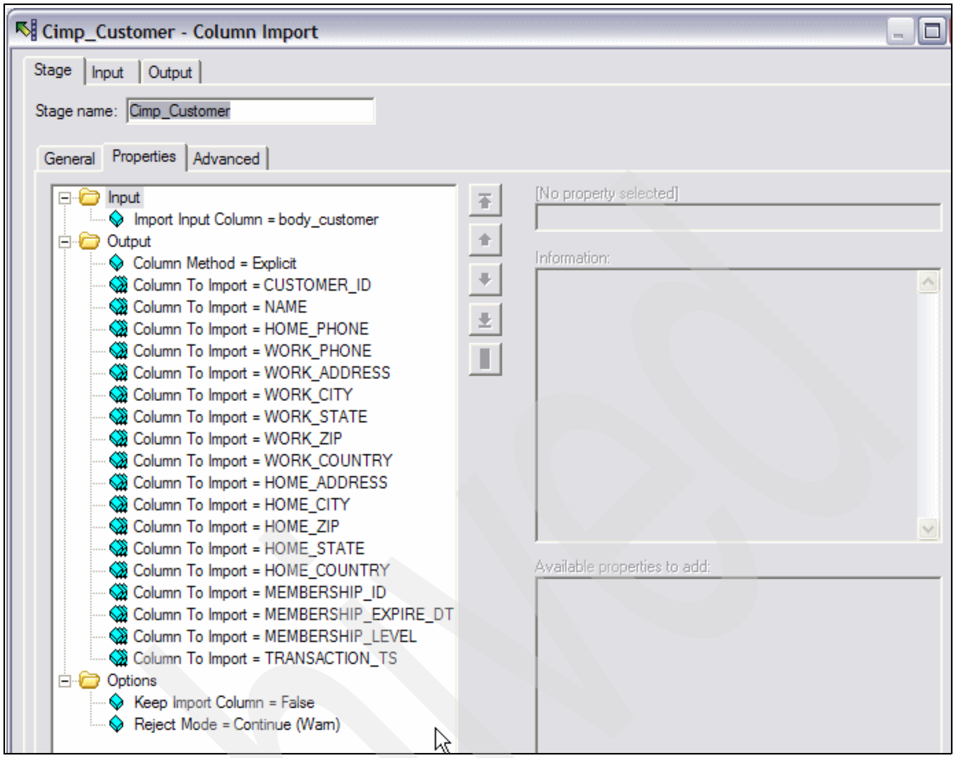


Figure 3-331 Create the J13_Daily_UpdateLookupDim job 5/26

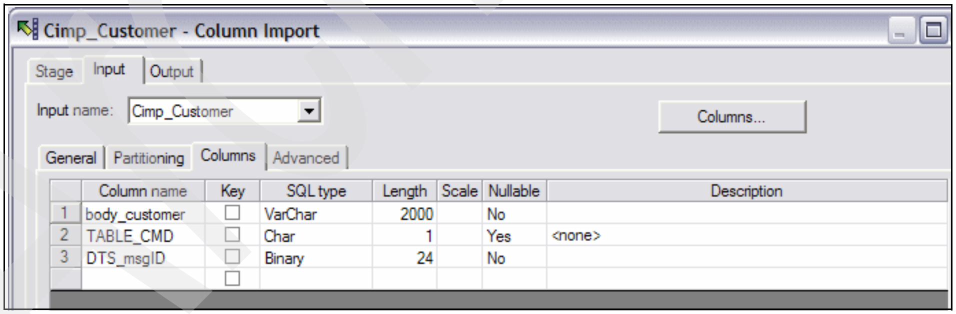


Figure 3-332 Create the J13_Daily_UpdateLookupDim job 6/26

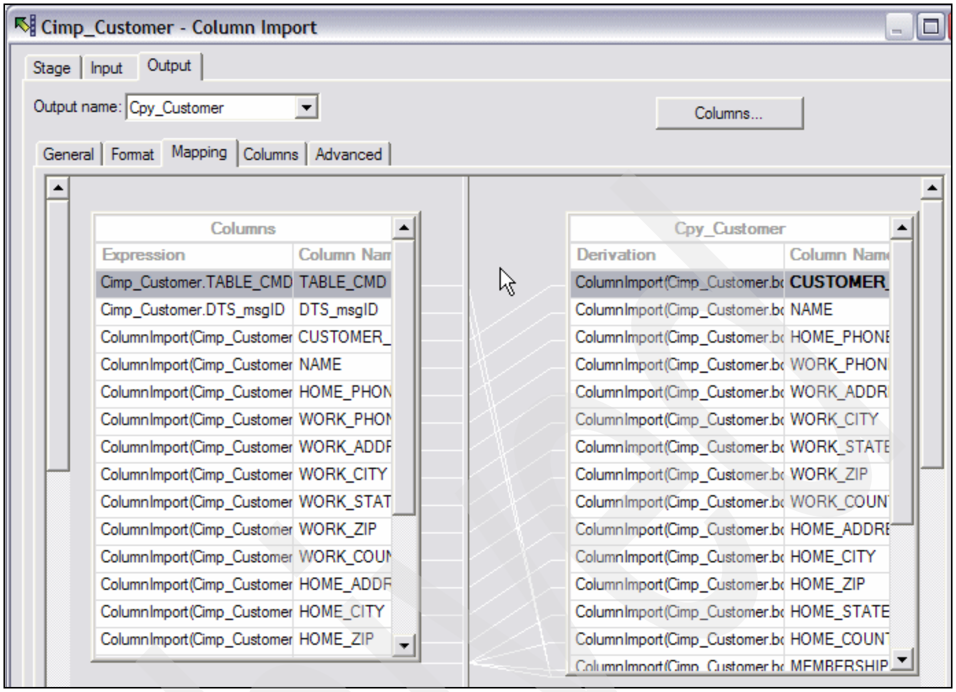


Figure 3-333 Create the J13_Daily_UpdateLookupDim job 7/26

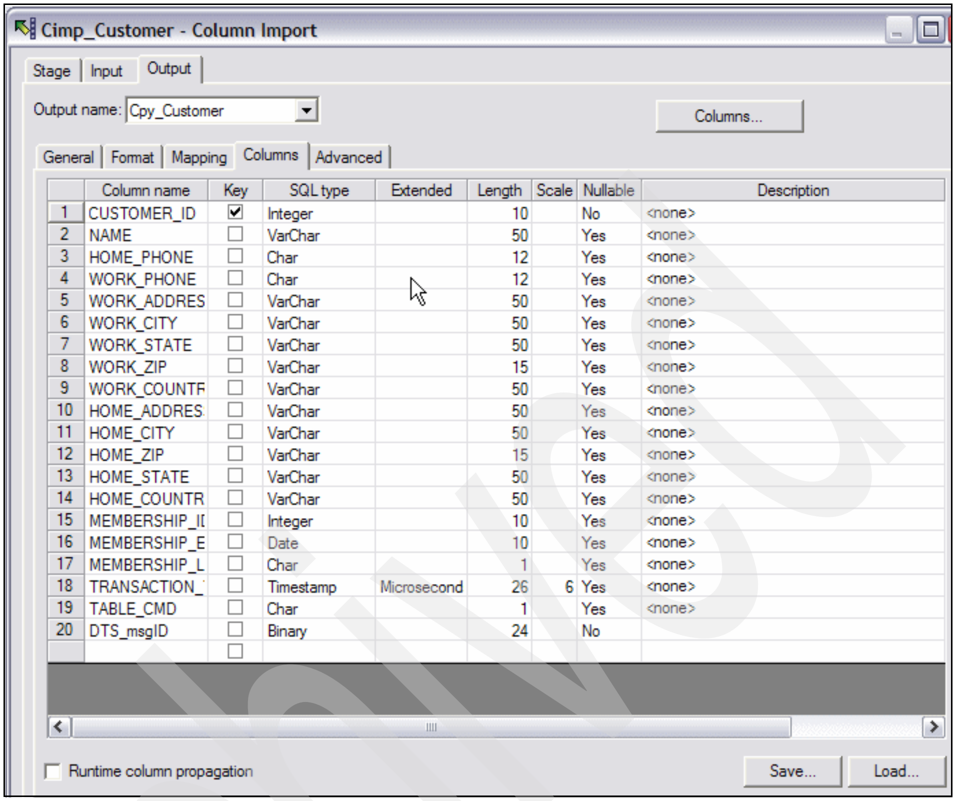


Figure 3-334   Create the J13_Daily_UpdateLookupDim job 8/26

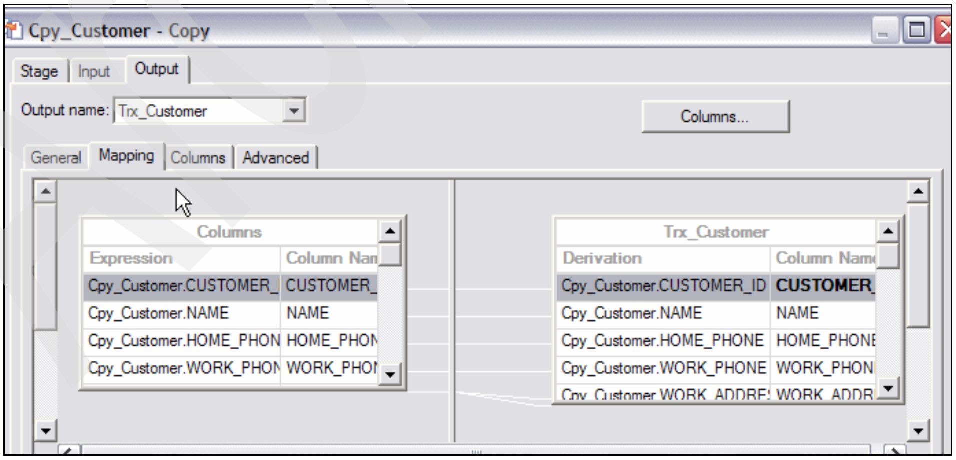


Figure 3-335   Create the J13_Daily_UpdateLookupDim job 9/26

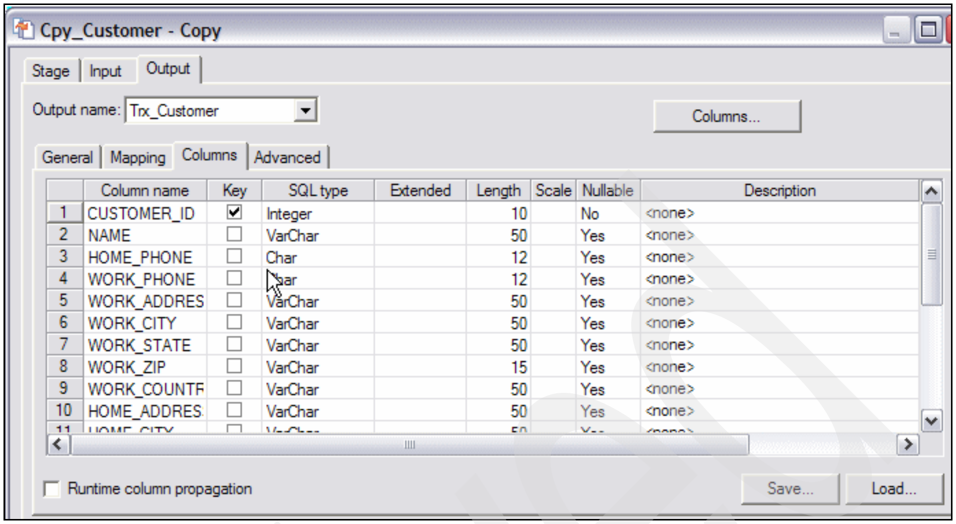


Figure 3-336 Create the J13_Daily_UpdateLookupDim job 10/26

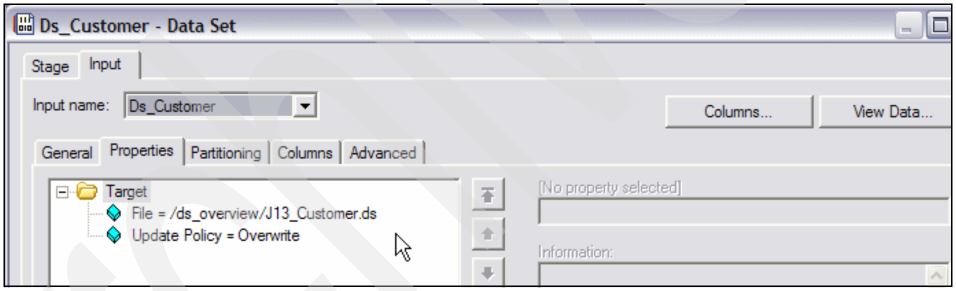


Figure 3-337 Create the J13_Daily_UpdateLookupDim job 11/26

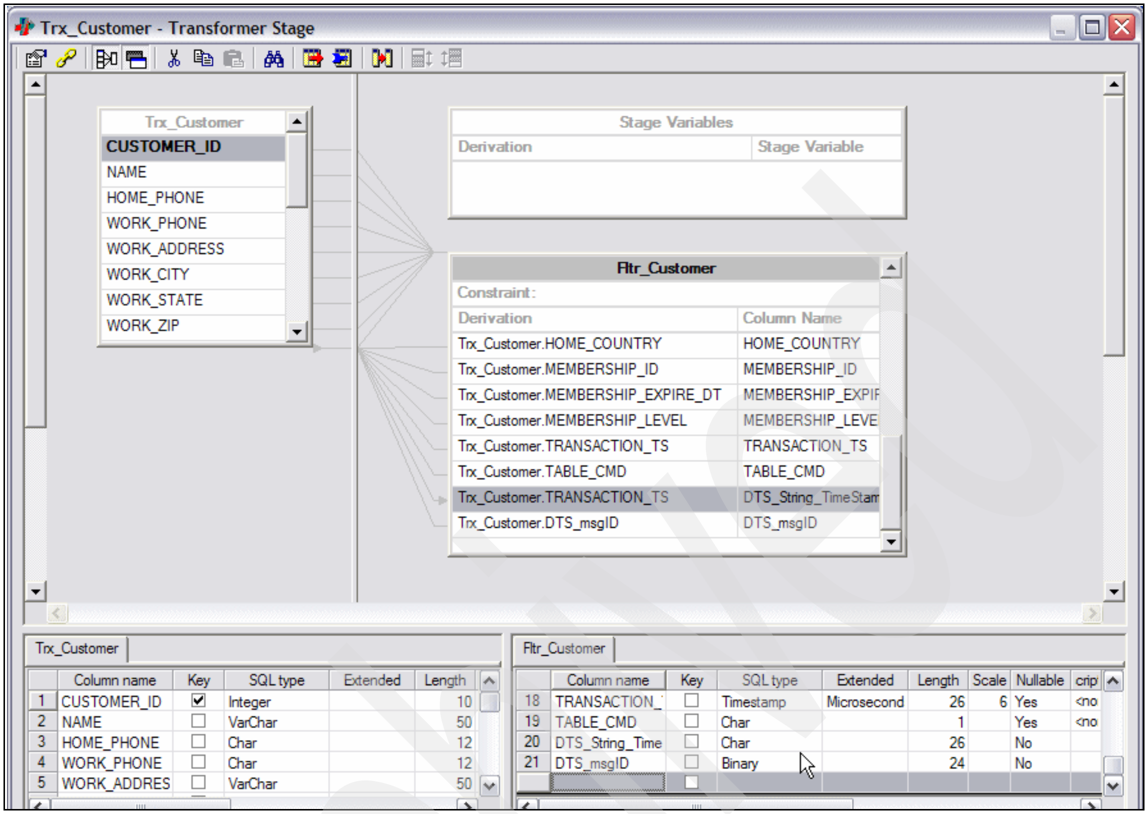


Figure 3-338 Create the J13_Daily_UpdateLookupDim job 12/26

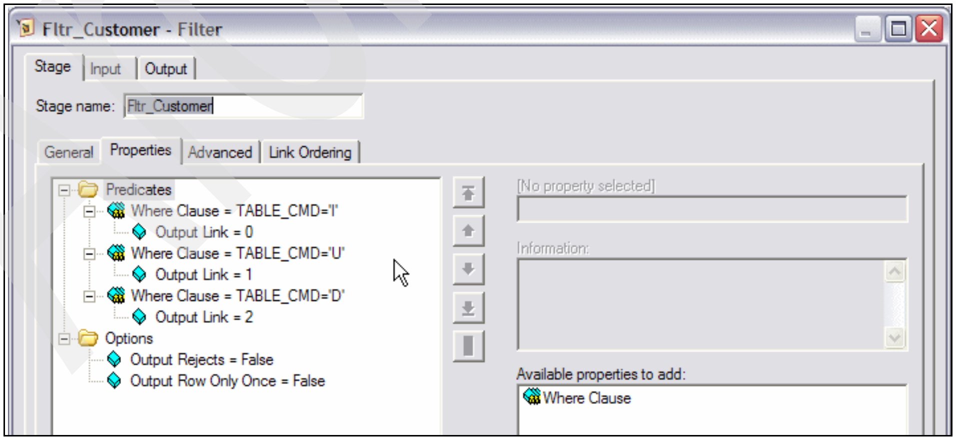


Figure 3-339 Create the J13_Daily_UpdateLookupDim job 13/26

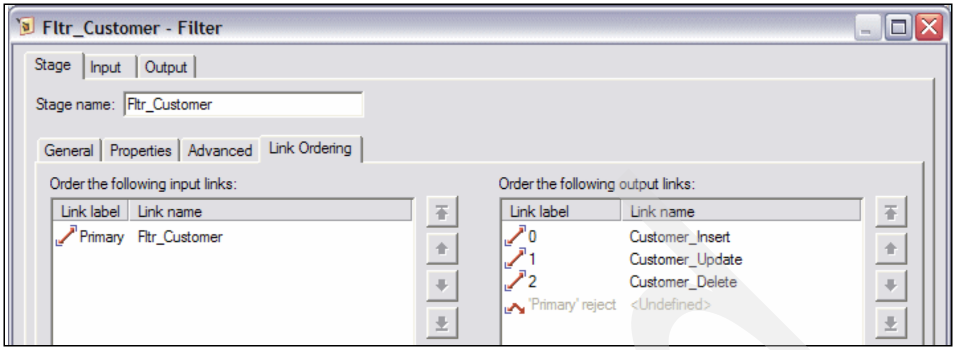


Figure 3-340 Create the J13_Daily_UpdateLookupDim job 14/26

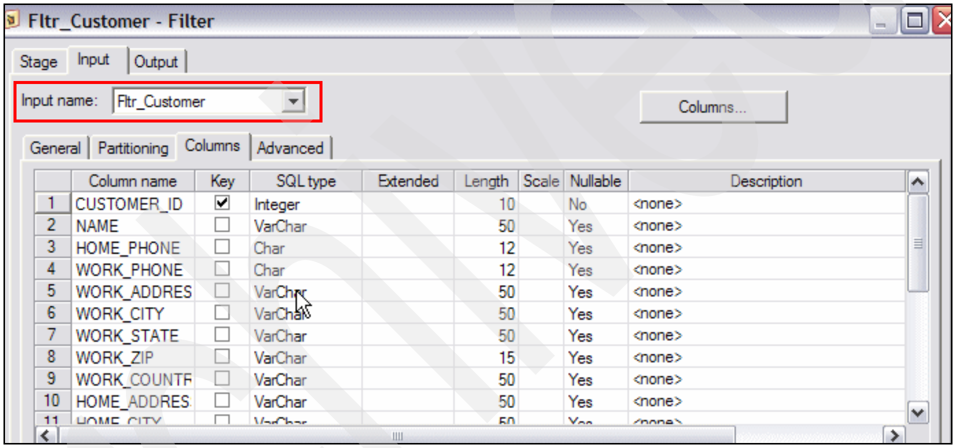


Figure 3-341 Create the J13_Daily_UpdateLookupDim job 15/26

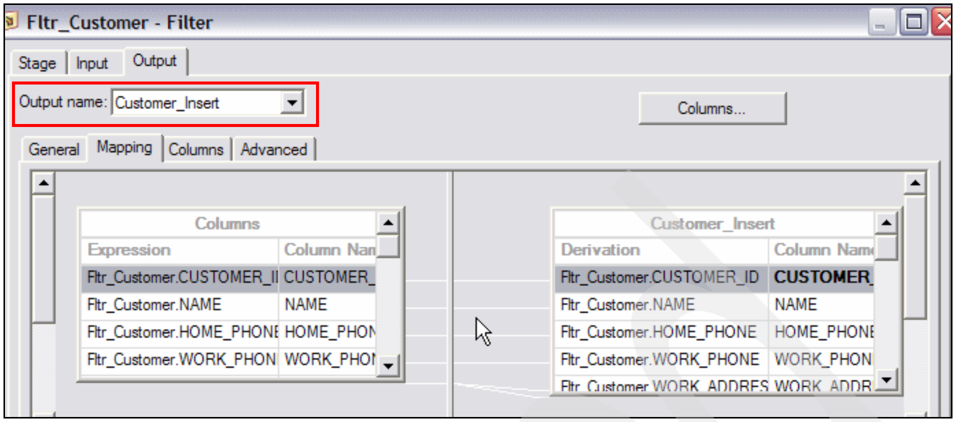


Figure 3-342 Create the J13_Daily_UpdateLookupDim job 16/26

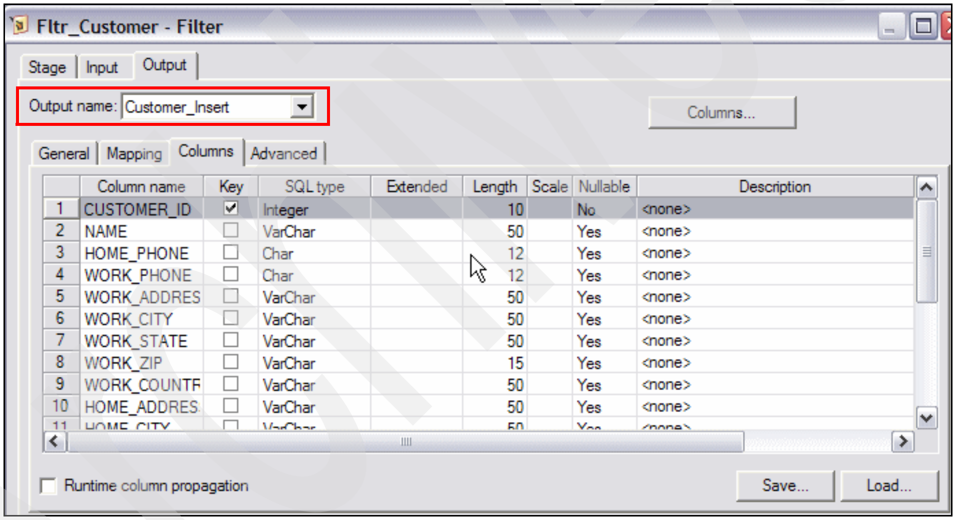


Figure 3-343 Create the J13_Daily_UpdateLookupDim job 18/26

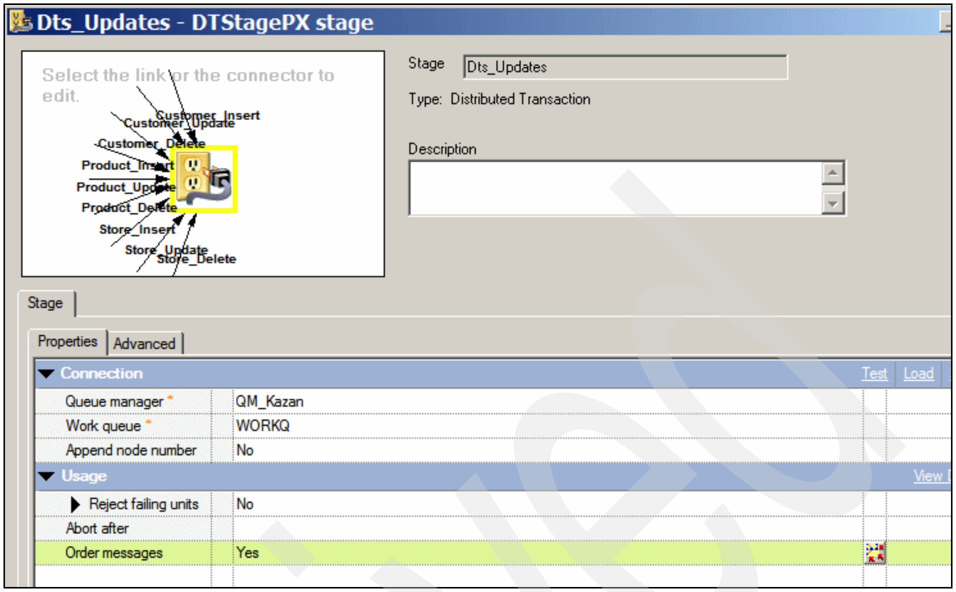


Figure 3-344 Create the J13_Daily_UpdateLookupDim job 19/26

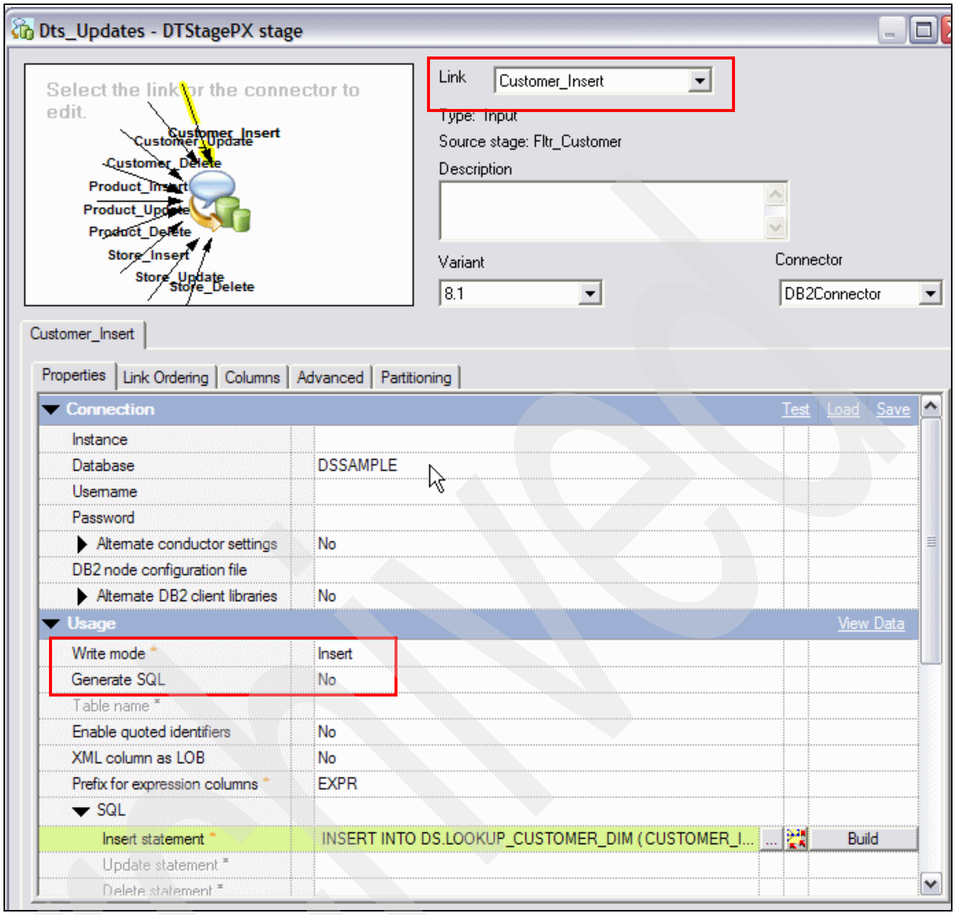


Figure 3-345 Create the J13_Daily_UpdateLookupDim job 20/26

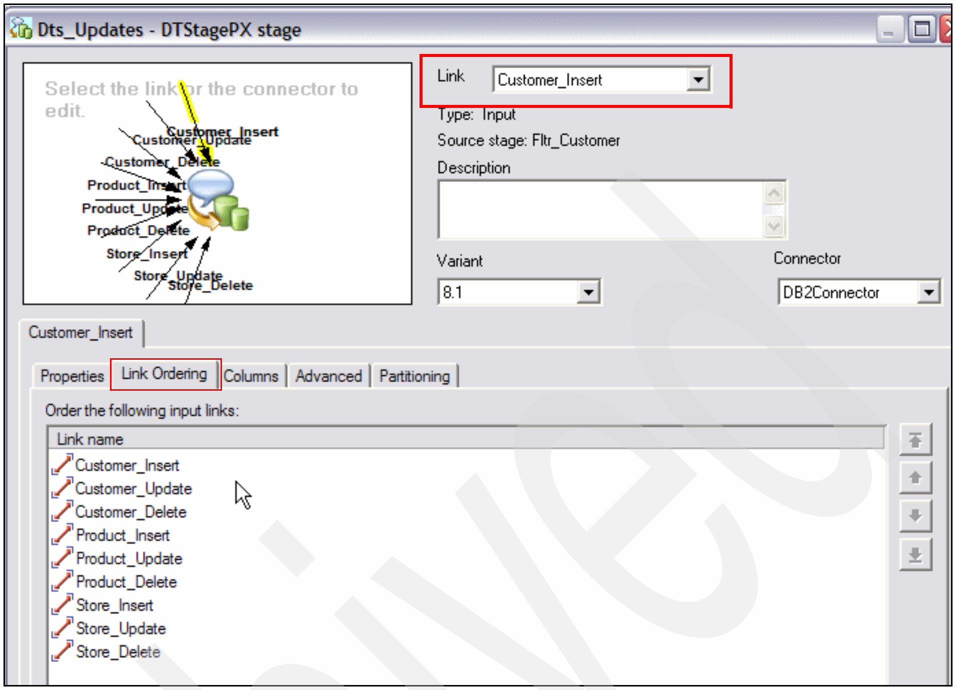


Figure 3-346 Create the J13_Daily_UpdateLookupDim job 21/26

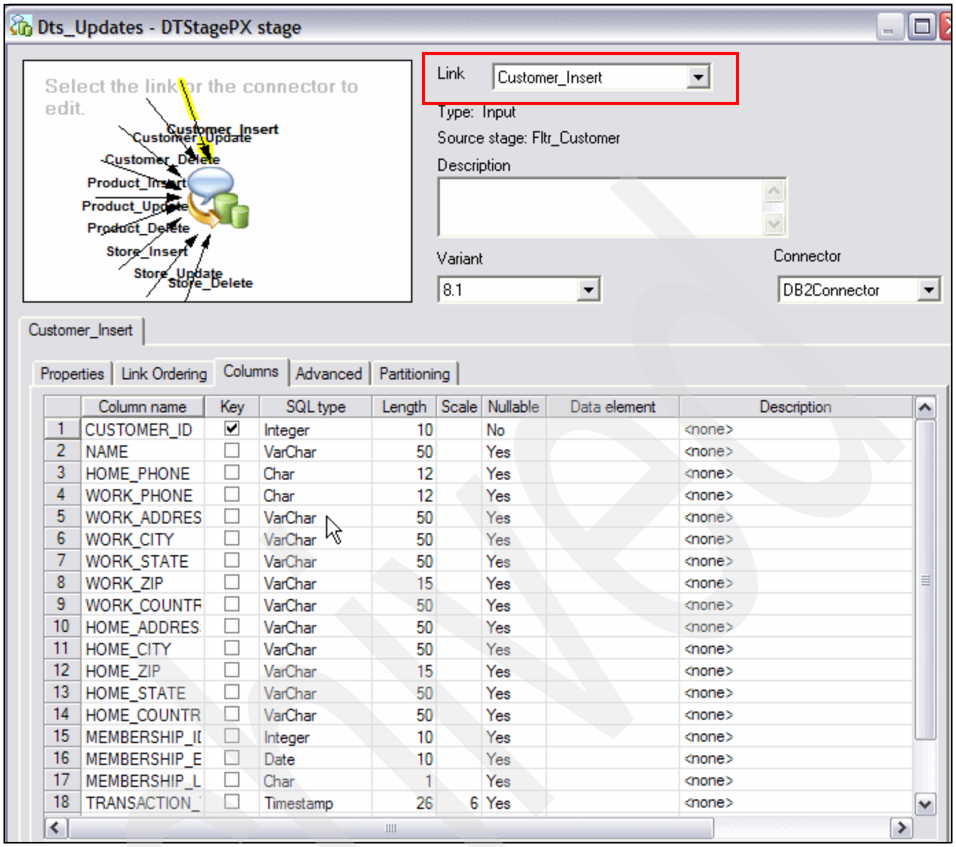


Figure 3-347 Create the J13_Daily_UpdateLookupDim job 22/26

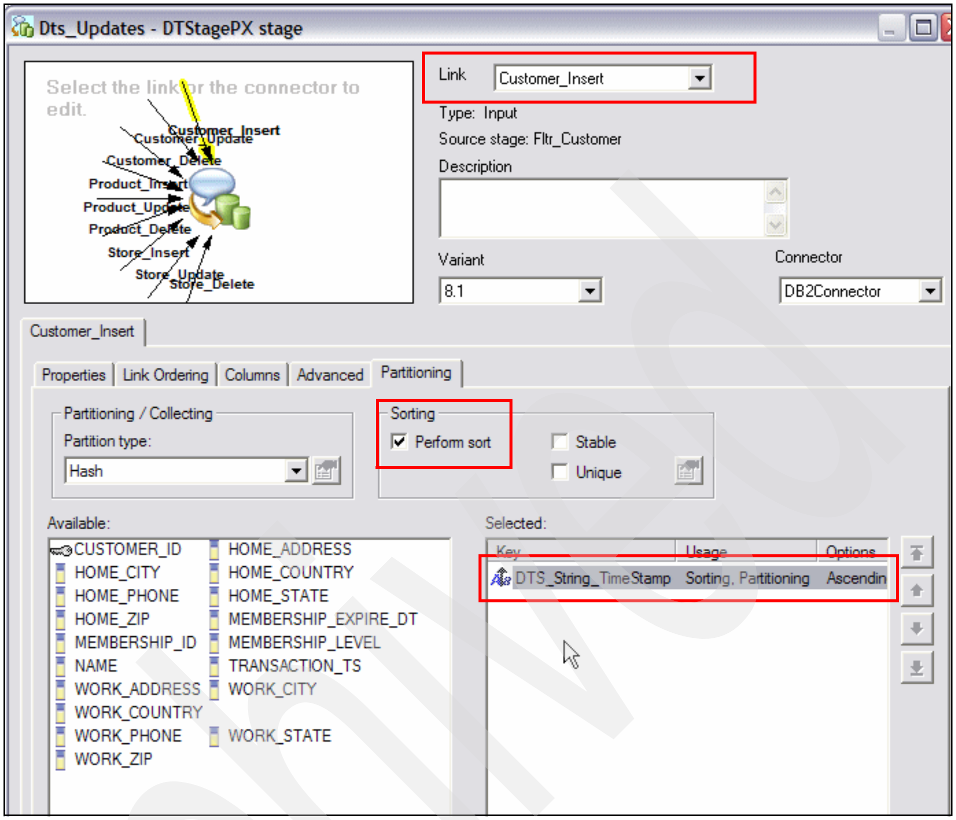


Figure 3-348 Create the J13_Daily_UpdateLookupDim job 23/26

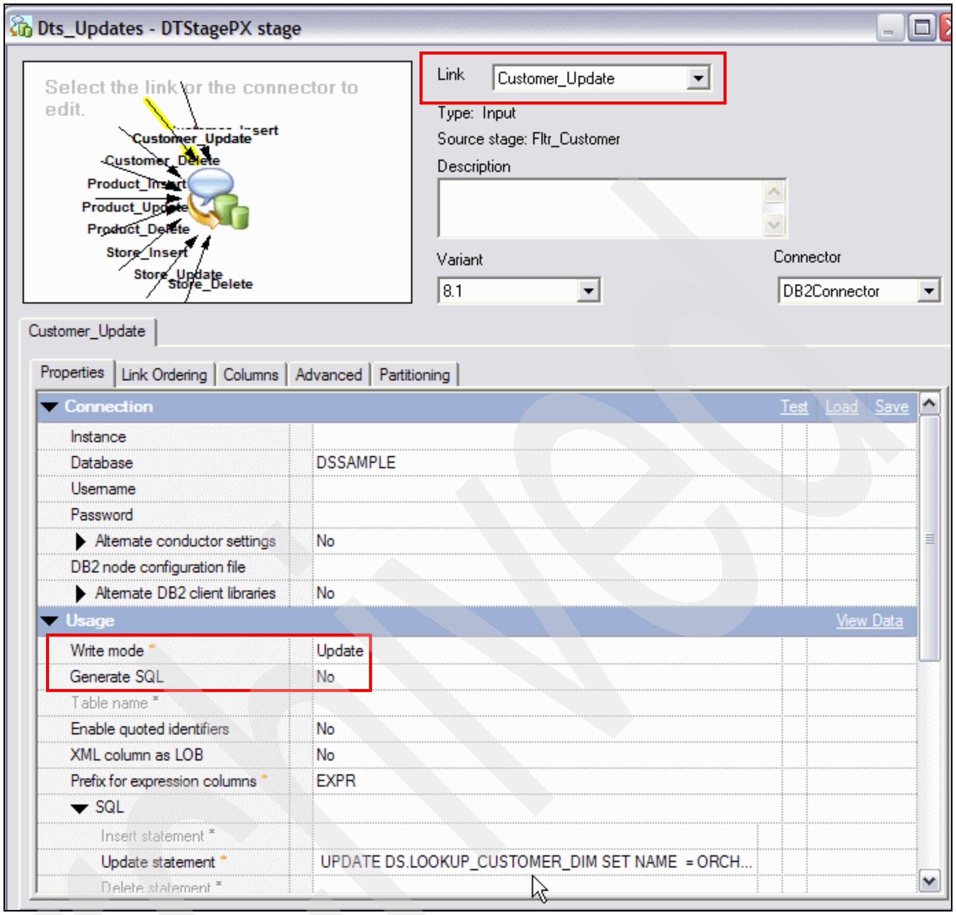


Figure 3-349 Create the J13_Daily_UpdateLookupDim job 24/26

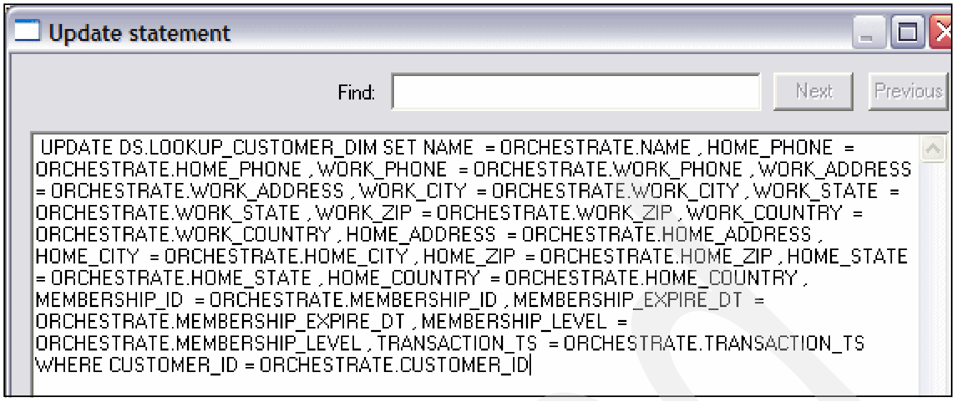


Figure 3-350 Create the J13_Daily_UpdateLookupDim job 25/26

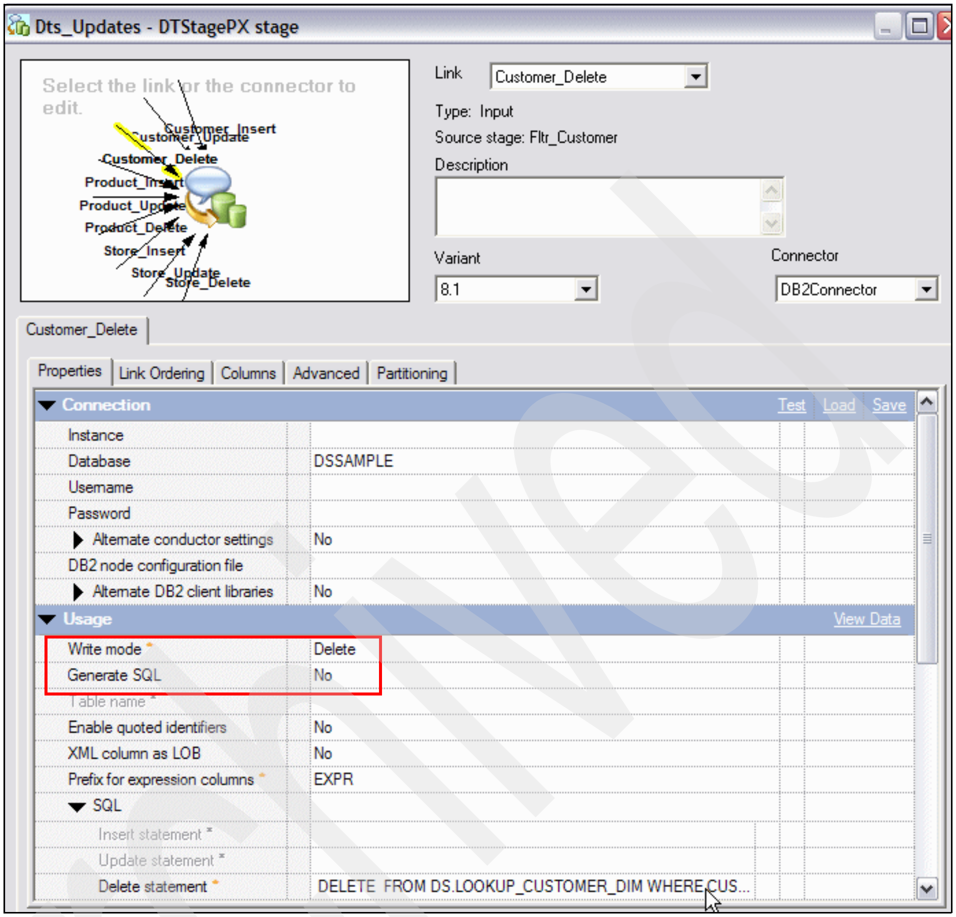


Figure 3-351 Create the J13_Daily_UpdateLookupDim job 26/26

J13_Daily_UpdateLookupDim execution (Day 1)

Figure 3-352 on page 383 through Figure 3-355 on page 385 show the results of the execution of this job with Day 1 data described earlier.

- Figure 3-352 on page 383 shows the results of the execution. It accepts 2 rows as input from the IBM WebSphere MQ message queue, which are both changes (one an update and the other a delete) to the Customer dimension table only. These two changes are written to the Ds_Customer data set as shown in Figure 3-308 on page 349 through Figure 3-310 on page 349.

- Figure 3-353 on page 384 through Figure 3-355 on page 385 show the LOOKUP_CUSTOMER_DIM table that incorporates the changes due to the update and delete. The CUSTOMER_ID 7 is no longer in the table, while the NAME, WORK_ADDRESS, HOME_ADDRESS, and TRANSACTION_TS reflect the incoming changes.

The next step is to execute the job described in “J14_Daily_CreateAllSalesStoreDS (Day 1)” on page 385.

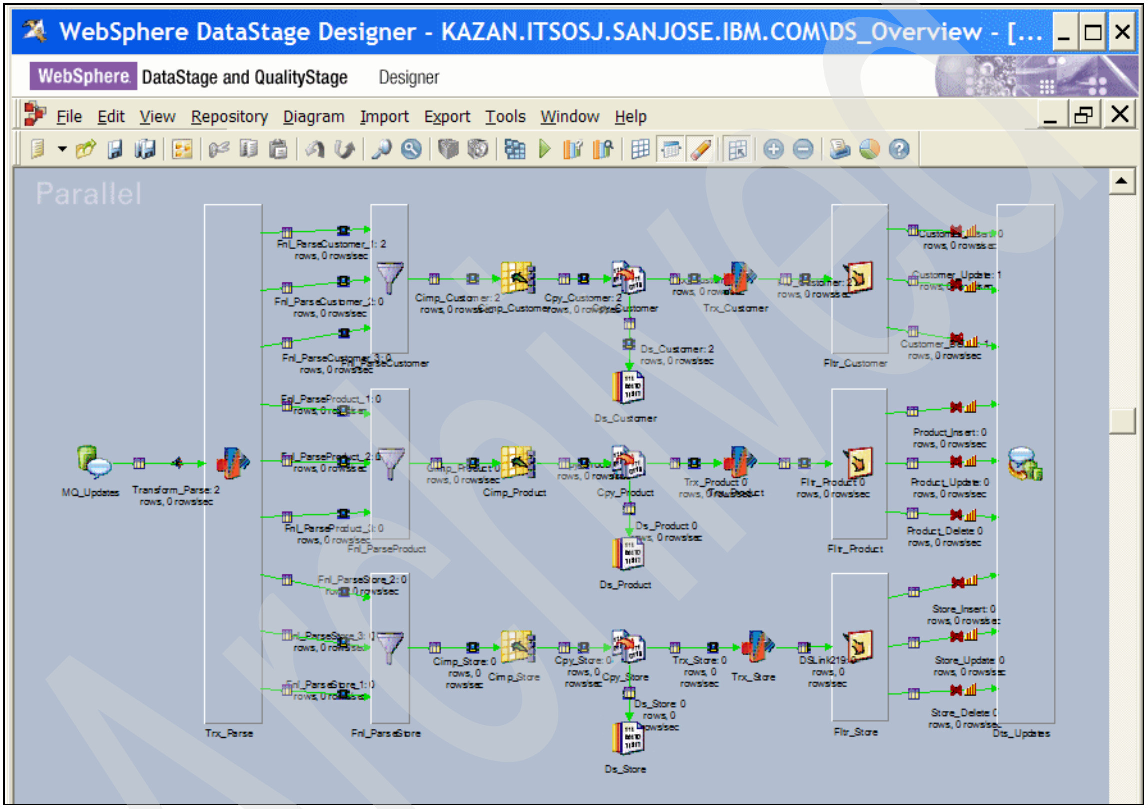
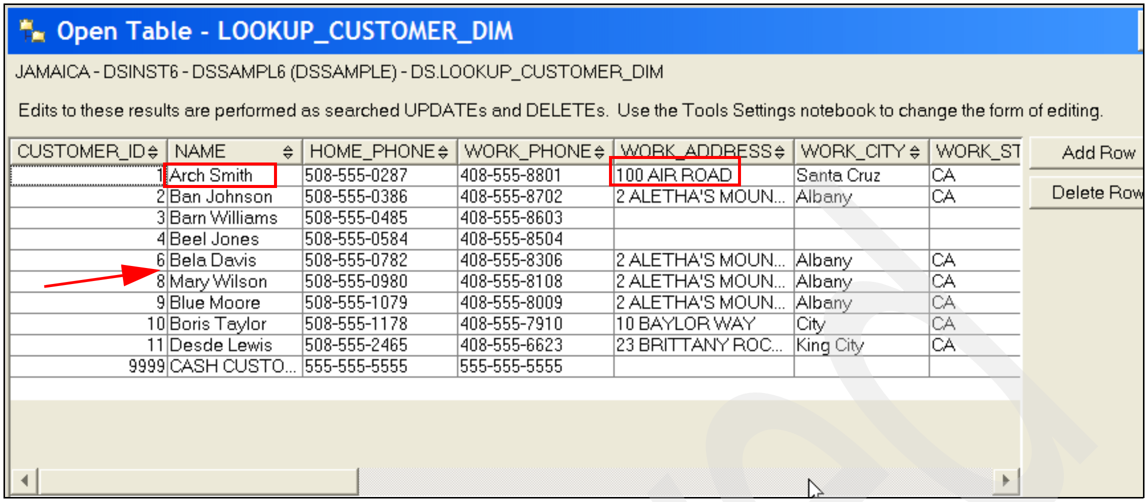


Figure 3-352 Execute the J13_Daily_UpdateLookupDim job (Day 1) 1/4


Open Table - LOOKUP_CUSTOMER_DIM

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.LOOKUP_CUSTOMER_DIM

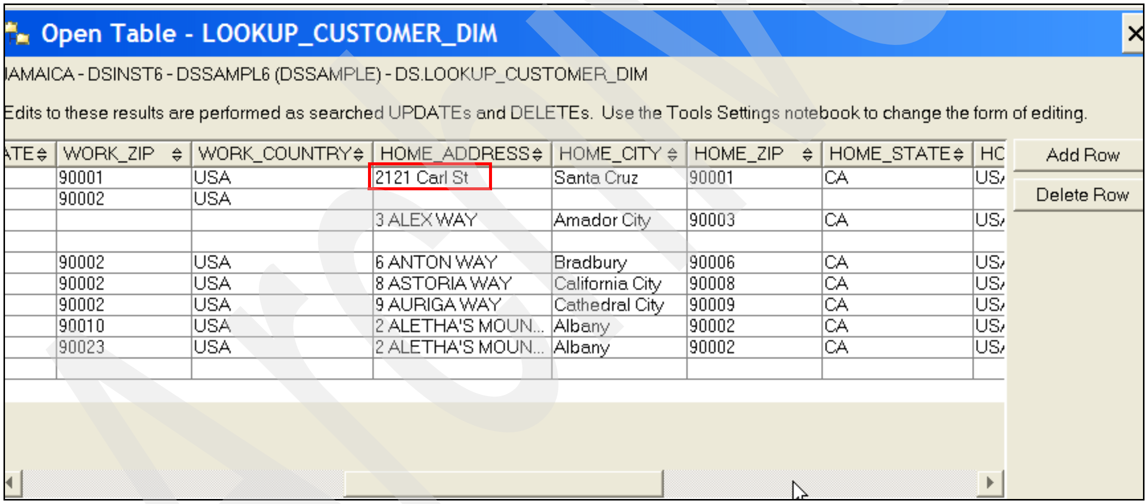
Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

CUSTOMER_ID	NAME	HOME_PHONE	WORK_PHONE	WORK_ADDRESS	WORK_CITY	WORK_ST
1	Arch Smith	508-555-0287	408-555-8801	100 AIR ROAD	Santa Cruz	CA
2	Ban Johnson	508-555-0386	408-555-8702	2 ALETHA'S MOUN...	Albany	CA
3	Barn Williams	508-555-0485	408-555-8603			
4	Beel Jones	508-555-0584	408-555-8504			
6	Bela Davis	508-555-0782	408-555-8306	2 ALETHA'S MOUN...	Albany	CA
8	Mary Wilson	508-555-0980	408-555-8108	2 ALETHA'S MOUN...	Albany	CA
9	Blue Moore	508-555-1079	408-555-8009	2 ALETHA'S MOUN...	Albany	CA
10	Boris Taylor	508-555-1178	408-555-7910	10 BAYLOR WAY	City	CA
11	Desde Lewis	508-555-2465	408-555-6623	23 BRITTANY ROC...	King City	CA
9999	CASH CUSTO...	555-555-5555	555-555-5555			

Add Row

Delete Row

Figure 3-353 Execute the J13_Daily_UpdateLookupDim job (Day 1) 2/4


Open Table - LOOKUP_CUSTOMER_DIM

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.LOOKUP_CUSTOMER_DIM

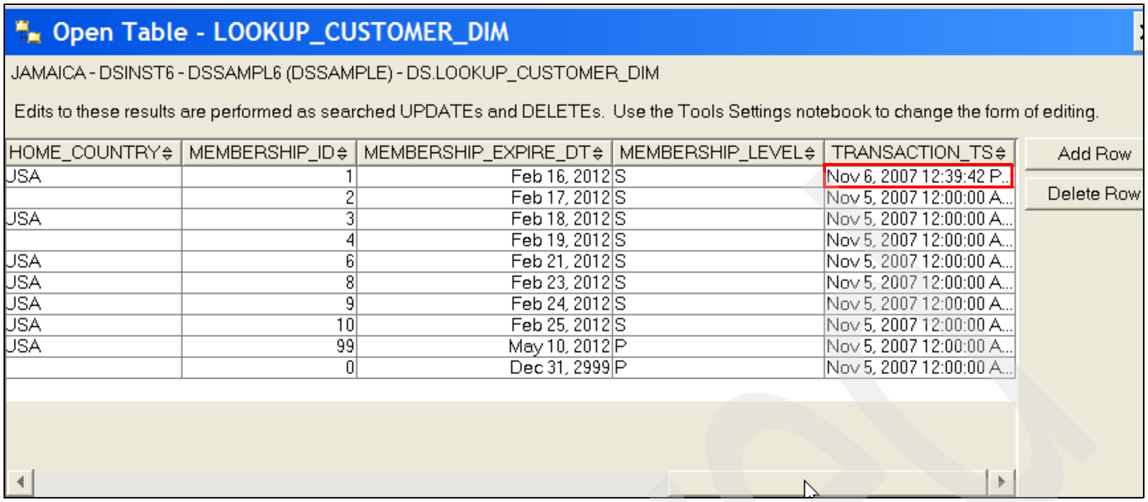
Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

ATE	WORK_ZIP	WORK_COUNTRY	HOME_ADDRESS	HOME_CITY	HOME_ZIP	HOME_STATE	HC
	90001	USA	2121 Carl St	Santa Cruz	90001	CA	US/
	90002	USA					
			3 ALEX WAY	Amador City	90003	CA	US/
	90002	USA	6 ANTON WAY	Bradbury	90006	CA	US/
	90002	USA	8 ASTORIA WAY	California City	90008	CA	US/
	90002	USA	9 AURIGA WAY	Cathedral City	90009	CA	US/
	90010	USA	2 ALETHA'S MOUN...	Albany	90002	CA	US/
	90023	USA	2 ALETHA'S MOUN...	Albany	90002	CA	US/

Add Row

Delete Row

Figure 3-354 Execute the J13_Daily_UpdateLookupDim job (Day 1) 3/4



Open Table - LOOKUP_CUSTOMER_DIM

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.LOOKUP_CUSTOMER_DIM

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

HOME_COUNTRY	MEMBERSHIP_ID	MEMBERSHIP_EXPIRE_DT	MEMBERSHIP_LEVEL	TRANSACTION_TS
USA	1	Feb 16, 2012	S	Nov 6, 2007 12:39:42 P...
	2	Feb 17, 2012	S	Nov 5, 2007 12:00:00 A...
USA	3	Feb 18, 2012	S	Nov 5, 2007 12:00:00 A...
	4	Feb 19, 2012	S	Nov 5, 2007 12:00:00 A...
USA	6	Feb 21, 2012	S	Nov 5, 2007 12:00:00 A...
USA	8	Feb 23, 2012	S	Nov 5, 2007 12:00:00 A...
USA	9	Feb 24, 2012	S	Nov 5, 2007 12:00:00 A...
USA	10	Feb 25, 2012	S	Nov 5, 2007 12:00:00 A...
USA	99	May 10, 2012	P	Nov 5, 2007 12:00:00 A...
	0	Dec 31, 2999	P	Nov 5, 2007 12:00:00 A...

Figure 3-355 Execute the J13_Daily_UpdateLookupDim job (Day 1) 4/4

J14_Daily_CreateAllSalesStoreDS (Day 1)

This job merges the sales transactions from all the stores into a single data set. Since the configuration of a Funnel stage has been described before, it is not repeated here.

Figure 3-356 on page 386 through Figure 3-358 on page 387 show the results of the execution of this job with Day 1 data described earlier.

- Figure 3-356 on page 386 shows the results of the execution. It accepts six rows from store 1, one row from store 9, and six rows from store 33 for a total of 13 rows that are written to the output data set.
- Figure 3-357 on page 386 through Figure 3-358 on page 387 show the contents of the output data set DS_AllSales.

The next step is to execute the job described in “J15_Daily_CreateSalesAggDS (Day 1)” on page 387.

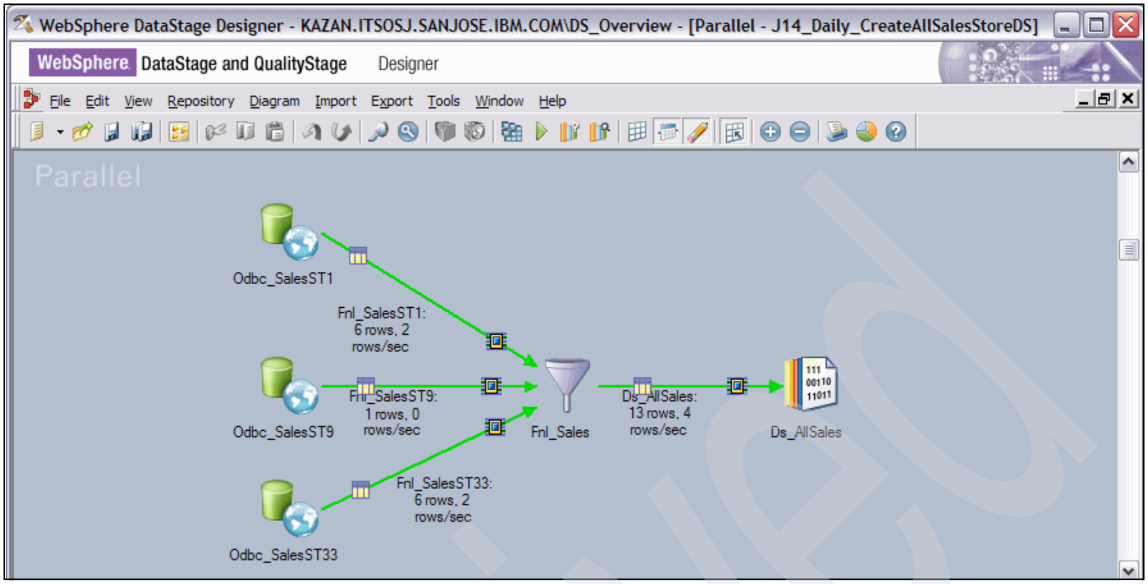
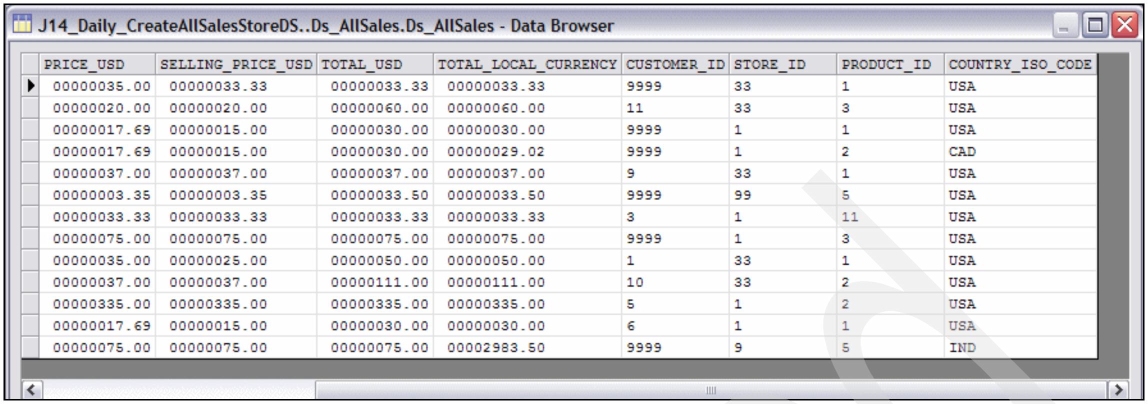


Figure 3-356 Execute the J14_Daily_CreateAllSalesStoreDS job (Day 1) 1/3

J14_Daily_CreateAllSalesStoreDS..Ds_AllSales.Ds_AllSales - Data Browser

SALES_ID	DATE	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID
82	2007-11-06 13:20:43.000000	1	00000035.00	00000033.33	00000033.33	00000033.33	9999
88	2007-11-06 17:03:39.000000	3	00000020.00	00000020.00	00000060.00	00000060.00	11
94	2007-11-06 22:09:22.000000	2	00000017.69	00000015.00	00000030.00	00000030.00	9999
126	2007-11-06 23:23:42.000000	2	00000017.69	00000015.00	00000030.00	00000029.02	9999
86	2007-11-06 14:55:41.000000	1	00000037.00	00000037.00	00000037.00	00000037.00	9
93	2007-11-06 18:03:03.000000	10	00000003.35	00000003.35	00000033.50	00000033.50	9999
122	2007-11-06 23:42:42.000000	1	00000033.33	00000033.33	00000033.33	00000033.33	3
129	2007-11-06 23:49:42.000000	1	00000075.00	00000075.00	00000075.00	00000075.00	9999
81	2007-11-06 13:09:32.000000	2	00000035.00	00000025.00	00000050.00	00000050.00	1
87	2007-11-06 16:00:00.000000	3	00000037.00	00000037.00	00000111.00	00000111.00	10
83	2007-11-06 19:59:42.000000	1	00000335.00	00000335.00	00000335.00	00000335.00	5
124	2007-11-06 23:11:42.000000	2	00000017.69	00000015.00	00000030.00	00000030.00	6
95	2007-11-06 18:39:33.000000	1	00000075.00	00000075.00	00000075.00	00002983.50	9999

Figure 3-357 Execute the J14_Daily_CreateAllSalesStoreDS job (Day 1) 2/3


J14_Daily_CreateAllSalesStoreDS..Ds_AllSales.Ds_AllSales - Data Browser

PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	COUNTRY_ISO_CODE
00000035.00	00000033.33	00000033.33	00000033.33	9999	33	1	USA
00000020.00	00000020.00	00000060.00	00000060.00	11	33	3	USA
00000017.69	00000015.00	00000030.00	00000030.00	9999	1	1	USA
00000017.69	00000015.00	00000030.00	00000029.02	9999	1	2	CAD
00000037.00	00000037.00	00000037.00	00000037.00	9	33	1	USA
00000003.35	00000003.35	00000033.50	00000033.50	9999	99	5	USA
00000033.33	00000033.33	00000033.33	00000033.33	3	1	11	USA
00000075.00	00000075.00	00000075.00	00000075.00	9999	1	3	USA
00000035.00	00000025.00	00000050.00	00000050.00	1	33	1	USA
00000037.00	00000037.00	00000111.00	00000111.00	10	33	2	USA
00000335.00	00000335.00	00000335.00	00000335.00	5	1	2	USA
00000017.69	00000015.00	00000030.00	00000030.00	6	1	1	USA
00000075.00	00000075.00	00000075.00	00002983.50	9999	9	5	IND

Figure 3-358 Execute the J14_Daily_CreateAllSalesStoreDS job (Day 1) 3/3

J15_Daily_CreateSalesAggDS (Day 1)

This job associates dimension attributes from the lookup tables with the sales transactions, and aggregates sales transactions by quantity, foreign currency and US currency using the grouping of customer, product, store, and date. The appending of dimension attributes is required by a subsequent SCD stage, while the aggregation is required for updating the Sales fact table.

Figure 3-359 on page 392 through Figure 3-399 on page 417 explain the main stages in this job and the configuration of these stages as described in "J15_Daily_CreateSalesAggDS (Day 1) configuration" on page 387, while Figure 3-400 on page 418 through Figure 3-412 on page 421 explain the execution of this job with Day 1 input as described in "J15_Daily_CreateSalesAggDS (Day 1) execution" on page 417.

J15_Daily_CreateSalesAggDS (Day 1) configuration

Figure 3-359 on page 392 shows the various stages in the job — it includes a three Data Set stages, three ODBCConnectorPX stages, four Join stages, one Transformer stage, one Aggregator stage, and one Remove Duplicates stage. The names of the stages were modified as shown:

1. Figure 3-360 on page 393 shows the **Properties** tab for the Joi_LookupCustomerDim output link involving an ODBCConnectorPX stage that retrieves dimension attributes from the LOOKUP_CUSTOMER_DIM table using automatically generated SQL.

   Figure 3-361 on page 394 shows the **Columns** tab for the same link that defines the metadata of the columns retrieved from the table.

2. Figure 3-362 on page 394 through Figure 3-364 on page 395 show the configuration of a Join stage that performs a left outer join of the merged sales transactions (from "J14_Daily_CreateAllSalesStoreDS (Day 1)" on page 385) with the Customer dimension lookup table on the CUSTOMER_ID column as the join key. The attributes from the Customer dimension lookup table are appended to those of the sales transactions in the output. The left outer join is specified because we want the sales transaction to appear in the join results, even if a business key in a sales transaction does not match a business key in the dimension lookup table.
   - Figure 3-362 on page 394 shows the **Properties** tab in the Stage page that identifies the Key as CUSTOMER_ID and Join Type as Left Outer.
   - Figure 3-363 on page 395 shows the **Link Ordering** tab in the Stage page that identifies the link Joi_CustomerDim as the Left (table) in the join, while the Joi_LookupCustomerDim link is identified as the Right (table) in the join.
   - Figure 3-364 on page 395 shows the **Mapping** tab in the Output page of the Joi_StoreDim link which maps all the columns from the two sources to the output link.
   - Figure 3-365 on page 396 shows the **Columns** tab in the Output page of the Joi_StoreDim link which defines the metadata of the columns. It includes all the columns from the two input sources.
3. Figure 3-366 on page 397 through Figure 3-372 on page 400 show the configuration of a Join stage that performs a left outer join of the output of the previous stage (Joi_StoreDim link) with the Store dimension lookup table (LOOKUP_STORE_DIM) on the STORE_ID column as the join key. The attributes from the Store dimension lookup table are appended to those of the columns in the output of the previous Join stage. Here again, the left outer join is specified because we want the sales transaction to appear in the join results, even if a business key in a sales transaction does not match a business key in the dimension lookup table.
4. Figure 3-373 on page 401 through Figure 3-377 on page 403 show the configuration of a Join stage that performs a left outer join of the output of the previous stage (Joi_ProductDim link) with the Product dimension lookup table on the PRODUCT_ID column as the join key. The attributes from the Product dimension lookup table are appended to those of the columns in the output of the previous Join stage. Here again, the left outer join is specified because we want the sales transaction to appear in the join results, even if a business key in a sales transaction does not match a business key in the dimension lookup table.

5. The output of the Trx_Dim link contains all the sales transactions appended with all the corresponding attributes (based on the business key).

   This data has to be processed in the Transformer stage as follows:

   a. Because of the left outer join specification, some of the values in the dimension lookup attributes of certain sales transactions will be NULL because of the absence of a business key match. Such a condition corresponds to a late arriving dimension scenario that must be rejected.

   b. The individual sales transactions might have a transaction date with at least one business key that does not correspond to the current version of that business key in the dimension lookup table. Such a condition corresponds to a late arriving data scenario that has to be rejected. Such transactions have to be processed outside the SCD stage flow because they are not handled correctly in the SCD stage.

   c. The remaining individual sales transactions must be aggregated using an Aggregator stage on columns TOTAL_LOCAL_CURRENCY, QUANTITY and TOTAL_USD based on grouping columns CUSTOMER_ID, PRODUCT_ID, STORE_ID, and DATE.

      > **Restriction:** The Aggregator stage has a restriction that all the incoming columns to it must either be Grouping Keys or Aggregations.

      The input data has many columns that are neither grouping keys or aggregations. Therefore, such columns must be separated out from the input to the Aggregator stage and then rejoined with the aggregated columns.

   These actions are performed in the Trx_Dim Transformer stage, Agg_Sales Aggregator stage, Rmd_Dim Remove Duplicates stage, and the Joi_Sales_Dm Join stage. These are described briefly here.

6. Figure 3-378 on page 404 through Figure 3-385 on page 408 shows the configuration of the Transformer stage that uses constraints to reject late arriving dimension and late arriving data sales transactions to a Data Set stage, select grouping keys and aggregation columns for the Aggregator stage, and all the columns in the input Trx_Dim link to the Remove Duplicates stage.

   – Figure 3-378 on page 404 shows the Transformer stage with mappings to the Rmd_Dim and Agg_Sales output links and the constraint that moves the data to these columns. The Agg_Sales output link has only the date component of the timestamp column Trx_Dim.DATE mapped to the output column DATE using the TimestampToDate function.The Ds_LateArrivingDim link is not shown here.

– Figure 3-379 on page 404 through Figure 3-381 on page 405 show the constraints that direct the output to the individual output links. Briefly, the following conditions cause a sales transaction to be directed to the Rmd_Dim and Agg_Sales output links:

  - Any sales transaction with a transaction timestamp (Trx_Dim.DATE column) that is greater than the P_TRANSACTION_TS (Product dimension effective timestamp[8]), S_TRANSACTION_TS (Store dimension effective timestamp) and C_TRANSACTION_TS (Customer dimension effective timestamp) NULL in the P_TRANSACTION_TS, C_TRANSACTION_TS, and S_TRANSACTION_TS columns is directed to the Rmd_Dim and Agg_Sales output links.

> **Note:** The sales transaction timestamp (Trx_Dim.DATE column) that fails this condition is directed to the reject link. A value of NULL in the Trx_Dim.DATE column corresponds to a late arriving dimension and has to be directed to the reject link. It is therefore assigned a timestamp 2099-12-31-00.00.00.000000 (using the NullToValue function) to ensure that the predicate evaluates to false.

– Figure 3-382 on page 406 through Figure 3-384 on page 408 show the partial list of columns associated with the Rmd_Dim, Ag_Sales, and Ds_LateArrivingDim links respectively.

– Figure 3-385 on page 408 shows the **Link Ordering** tab in the Stage page that identifies the ordering of the output links.

7. Figure 3-386 on page 409 through Figure 3-388 on page 410 shows the configuration of the Ds_LateArrivingDim Data Set stage.

   – Figure 3-386 on page 409 shows the **Properties** tab in the Input stage which identifies the target file name (J15_Ds_LateArrivingDim.ds).

   – Figure 3-387 on page 409 and Figure 3-388 on page 410 shows the **Columns** tab in the Input stage which identifies all the columns from the Trx_Dim input link to the Trx_Dim Transformer stage.

8. Figure 3-389 on page 410 and Figure 3-390 on page 411 show the configuration of the Remove Duplicates stage. The incoming data on the Rmd_Dim input link must have any duplicates on the combined columns (CUSTOMER_ID, PRODUCT_ID, STORE_ID, DATE) removed by retaining only the first of such duplicates in the output link Joi_Dim. This is required to ensure that the subsequent Join stage that rebuilds the sales transaction with the aggregations computed in the Agg_Sales Aggregator stage does not produce erroneous results that contain duplicates.

8 An effective timestamp corresponds to the current version of the business key of a dimension.

– Figure 3-389 on page 410 shows the **Properties** tab in the Stage page that identify the columns (CUSTOMER_ID, PRODUCT_ID, STORE_ID, DATE) to be checked for duplicates, and to retain only the first occurrence (Duplicate To Retain = First) in the output.

– Figure 3-390 on page 411 shows the **Mapping** tab in the Output page that shows all the columns being mapped to the output link Joi_Dim.

9. Figure 3-391 on page 412 through Figure 3-394 on page 413 describe the configuration of the Aggregator stage identifying the Grouping Keys (CUSTOMER_ID, PRODUCT_ID, STORE_ID, DATE) and the Aggregations columns TOTAL_LOCAL_CURRENCY, QUANTITY, and TOTAL_USD.

   – Figure 3-391 on page 412 shows the **Properties** tab in the Stage page identifying the four Grouping Keys columns, and the three Aggregations columns.

   The Options category specifies Method = Sort. This is recommended if the the number of groups is large, or if some grouping keys can take on many values. However, sort mode requires the input data set to have been partition sorted with all of the grouping keys specified as hashing and sorting keys (this happens automatically if the auto method is set in the Partitioning tab). Sorting requires a pre-grouping operation — after sorting, all records in a given group in the same partition are consecutive.

   – Figure 3-392 on page 412 shows the **Columns** tab in the Input page, which identifies the metadata of the incoming data. It only includes grouping keys and aggregation columns.

   – Figure 3-393 on page 413 shows the **Mapping** tab in the Output page of the Joi_Sales link. It is a one-to-one mapping of all the columns as seen in the **Columns** tab in the Output page as shown in Figure 3-394 on page 413.

10. Figure 3-395 on page 414 through Figure 3-399 on page 417 show the configuration of the Joi_Sales_Dim Join stage that re-appends the three dimension lookup attributes from the Rmd_Dim Remove Duplicates stage with the aggregated sales transaction output of the Agg_Sales Aggregator stage. An inner join is specified since all the dimension lookup business keys originated from the same sales transactions.

   – Figure 3-395 on page 414 shows the **Properties** tab in the Stage page that identifies the Join Keys (CUSTOMER_ID, PRODUCT_ID, STORE_ID, DATE) and inner join (Join Type = Inner).

   – Figure 3-396 on page 414 shows the **Link Ordering** tab in the Stage page. Since this is an inner join, the choice of left and right do not really matter.

– Figure 3-397 on page 415 shows the **Mapping** tab in the Output page of the Ds_AggSales link. It is a one-to-one mapping of all the columns as seen in the **Columns** tab in the Output page as shown in Figure 3-398 on page 416 and Figure 3-399 on page 417.

The results of the execution of this job on Day 1 are described in “J15_Daily_CreateSalesAggDS (Day 1) execution” on page 417.

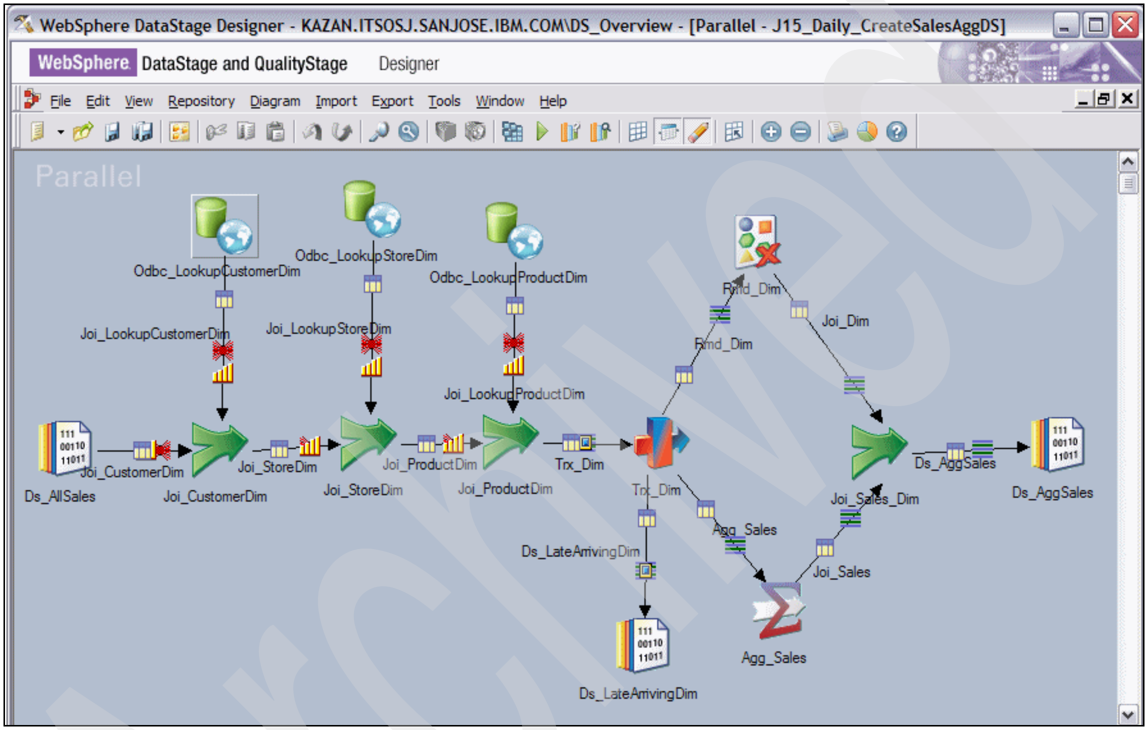


Figure 3-359 Create the J15_Daily_CreateSalesAggDS job 1/41

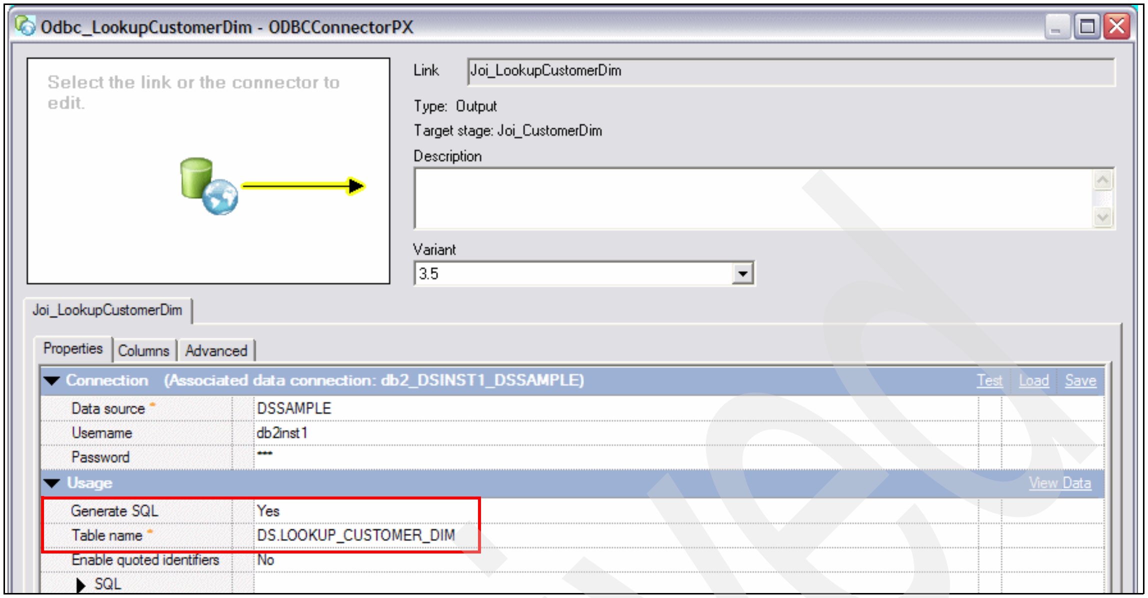


Figure 3-360 Create the J15_Daily_CreateSalesAggDS job 2/41

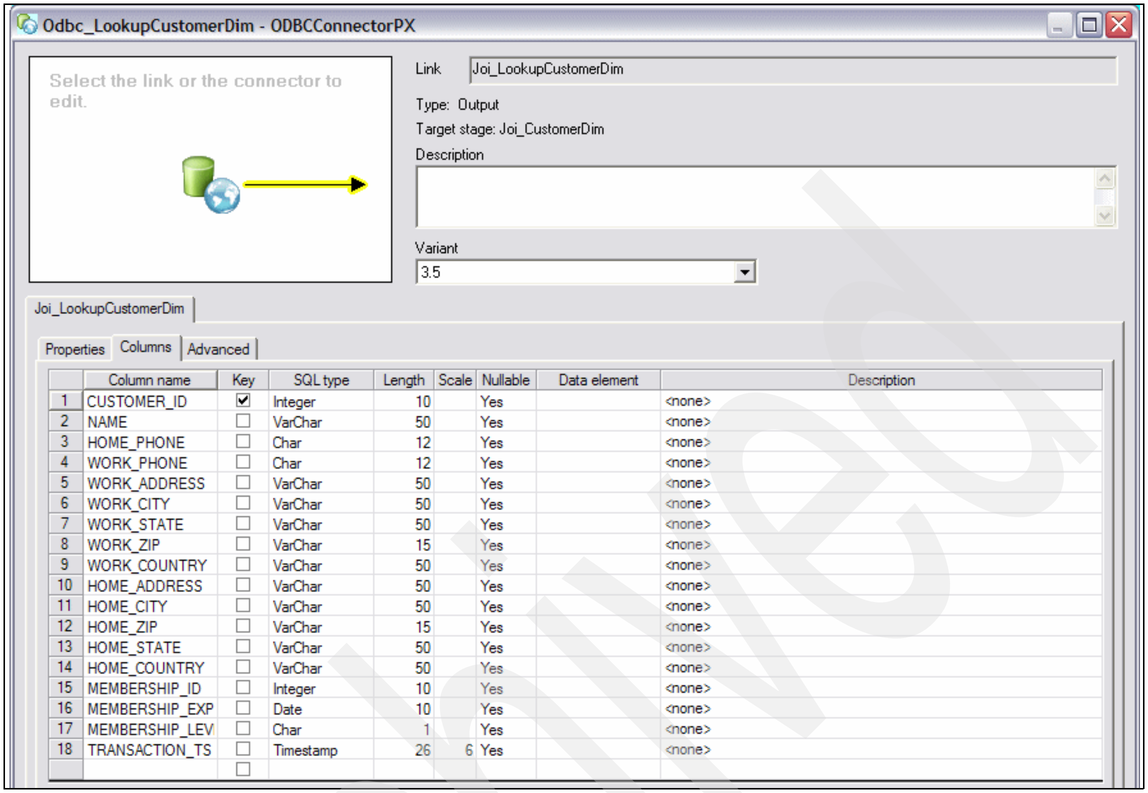


Figure 3-361 Create the J15_Daily_CreateSalesAggDS job 3/41

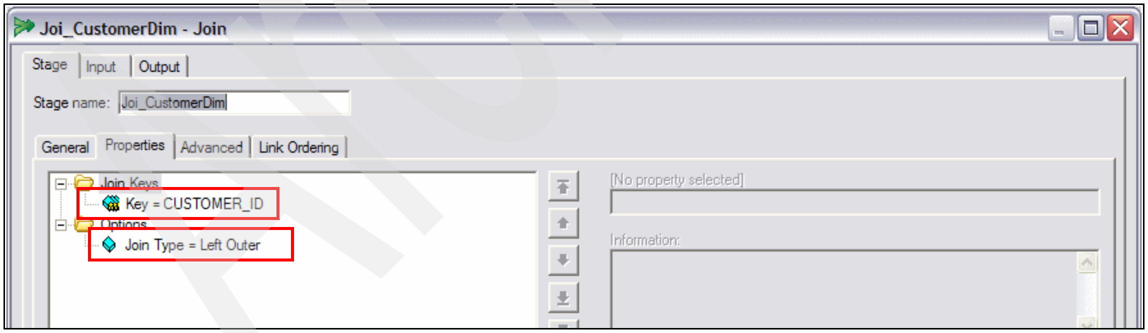


Figure 3-362 Create the J15_Daily_CreateSalesAggDS job 4/41

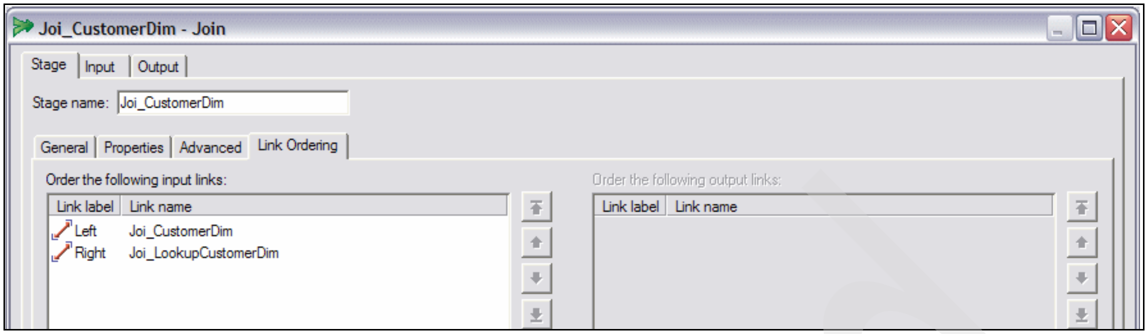


Figure 3-363   Create the J15_Daily_CreateSalesAggDS job 5/41

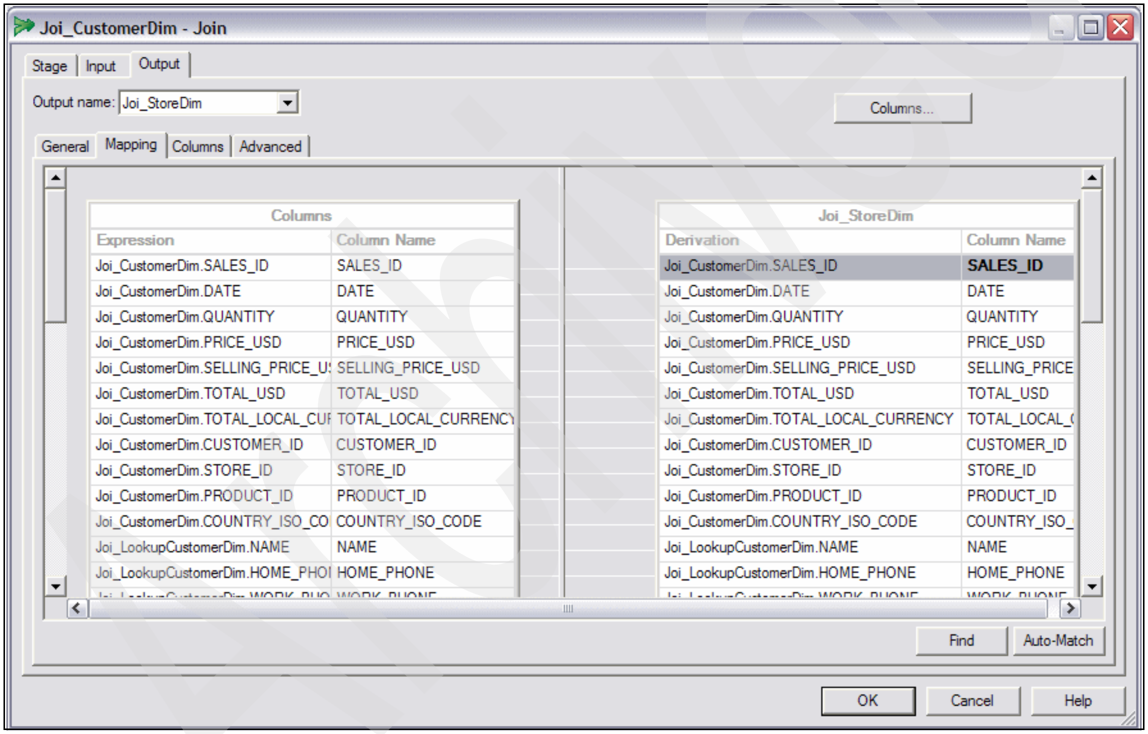


Figure 3-364   Create the J15_Daily_CreateSalesAggDS job 6/41

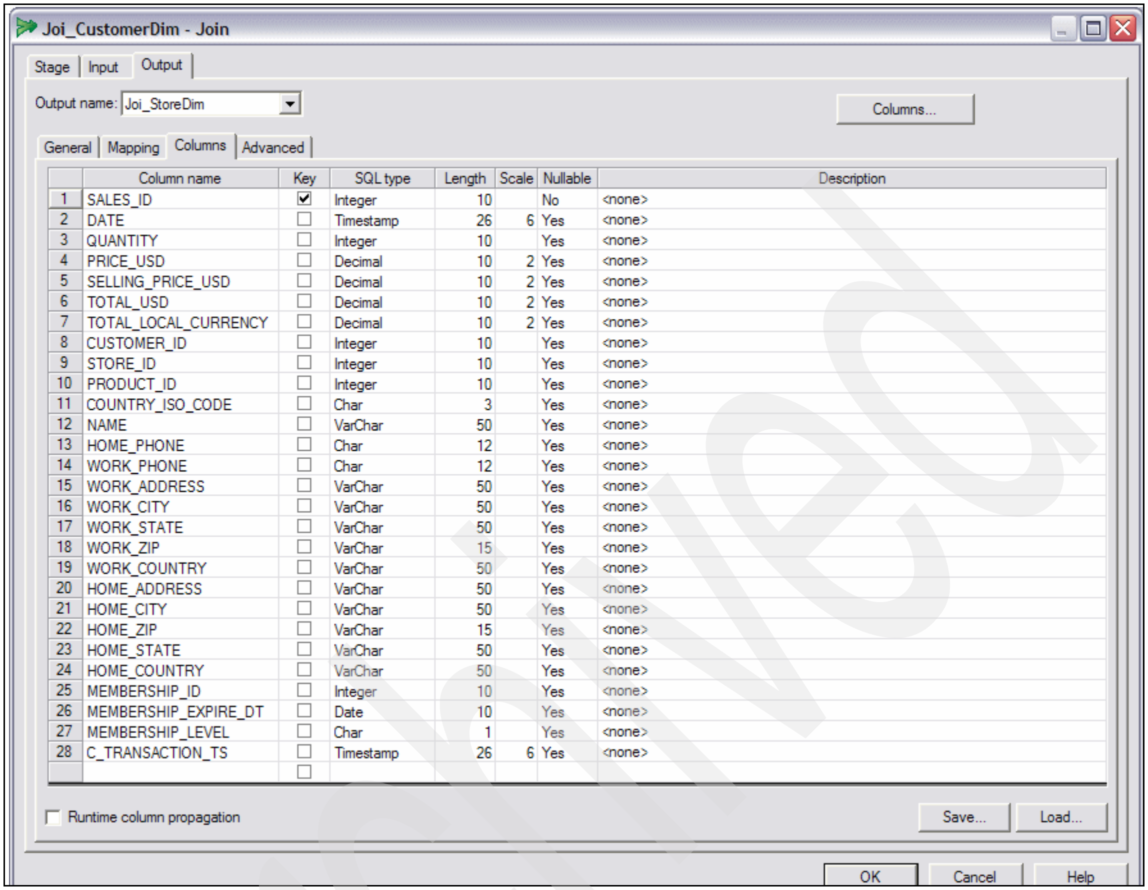


Figure 3-365 Create the J15_Daily_CreateSalesAggDS job 7/41

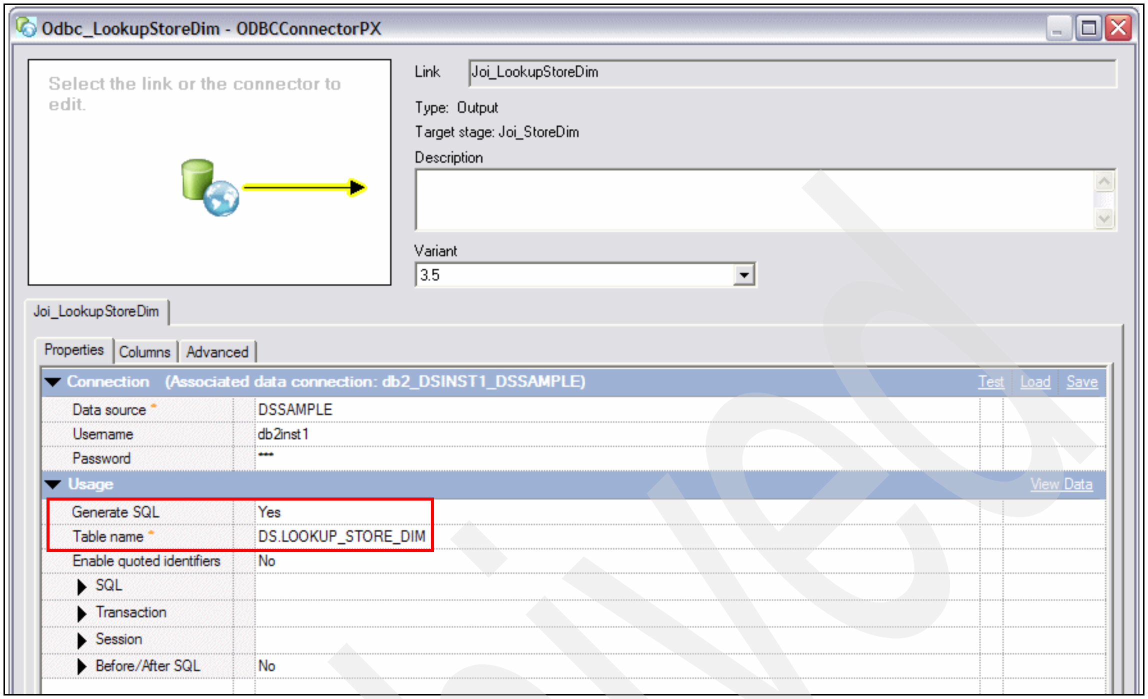


Figure 3-366 Create the J15_Daily_CreateSalesAggDS job 8/41

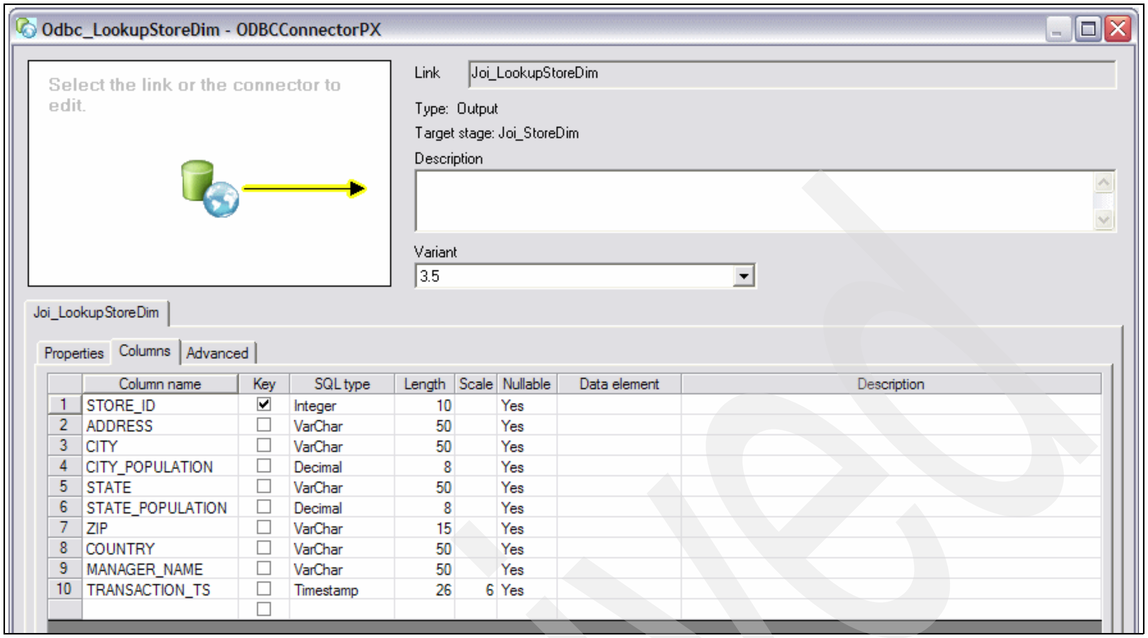


Figure 3-367 Create the J15_Daily_CreateSalesAggDS job 9/41

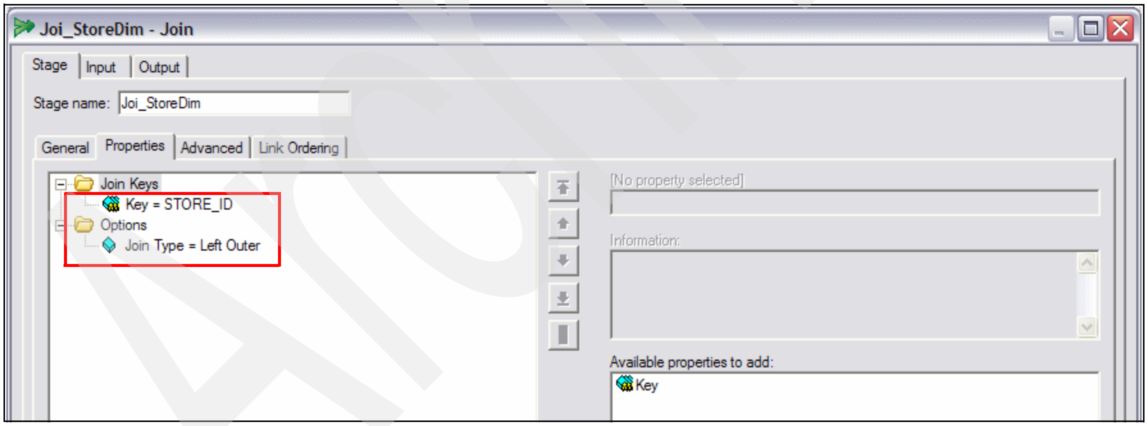


Figure 3-368 Create the J15_Daily_CreateSalesAggDS job 10/41

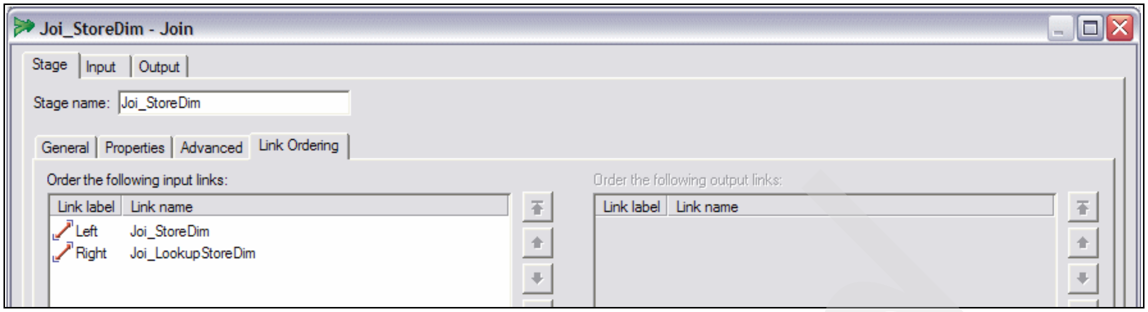


Figure 3-369 Create the J15_Daily_CreateSalesAggDS job 11/41

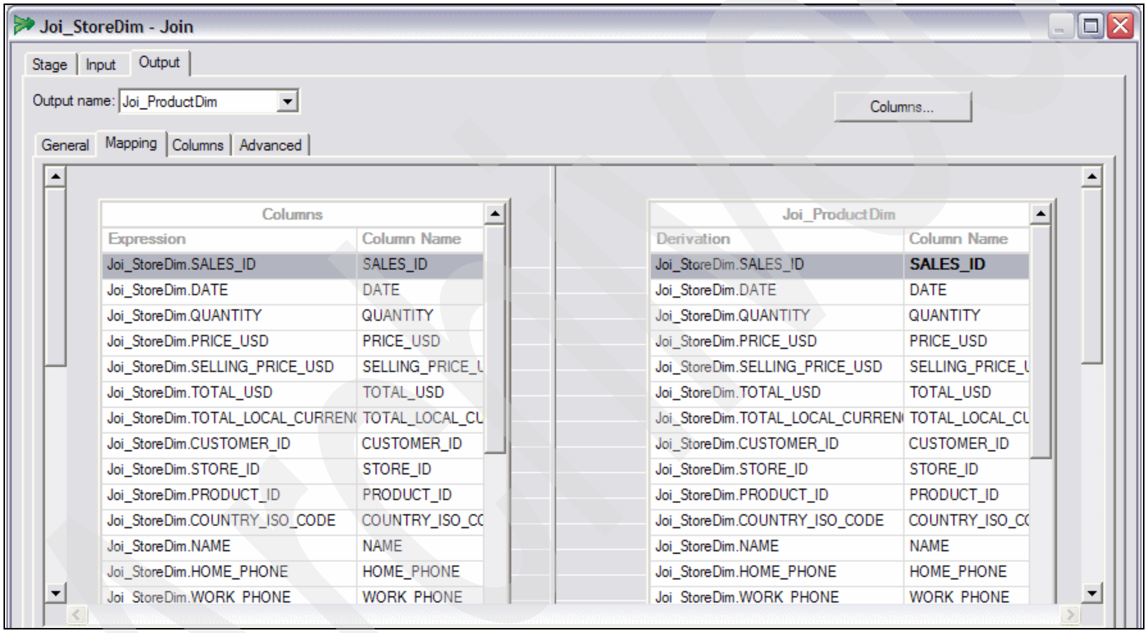
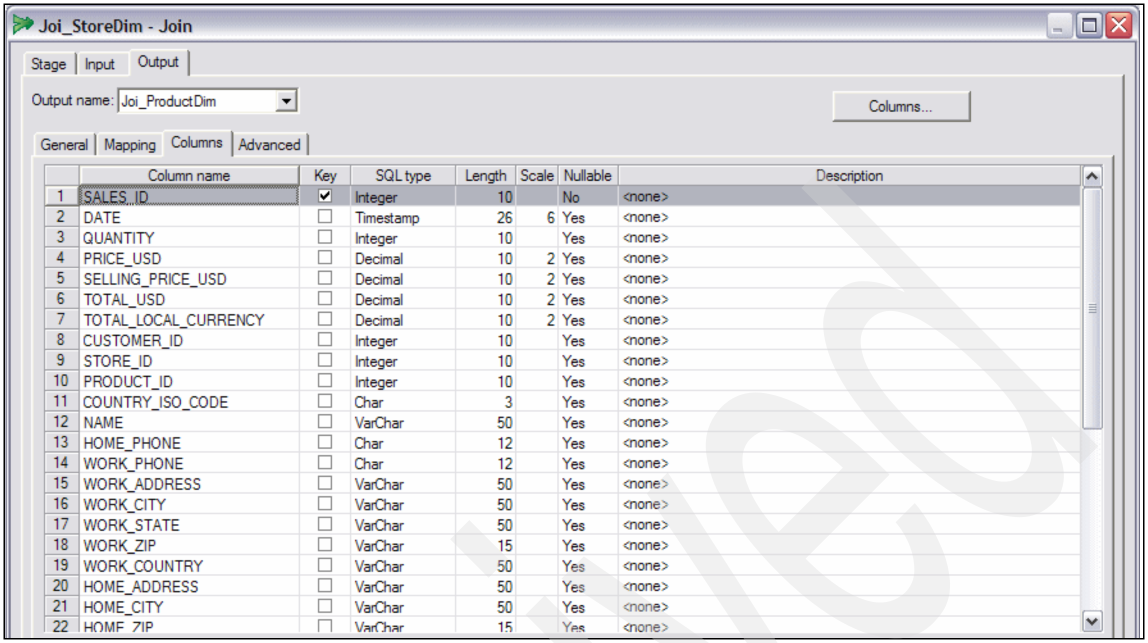
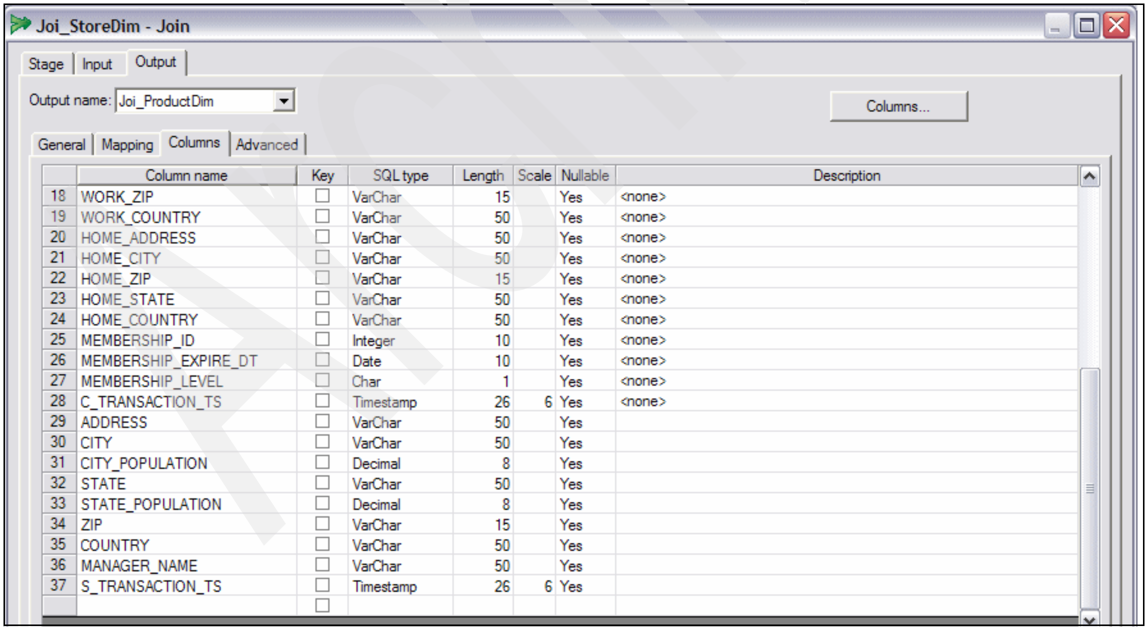


Figure 3-370 Create the J15_Daily_CreateSalesAggDS job 12/41



	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	SALES_ID	☑	Integer	10		No	<none>
2	DATE	☐	Timestamp	26	6	Yes	<none>
3	QUANTITY	☐	Integer	10		Yes	<none>
4	PRICE_USD	☐	Decimal	10	2	Yes	<none>
5	SELLING_PRICE_USD	☐	Decimal	10	2	Yes	<none>
6	TOTAL_USD	☐	Decimal	10	2	Yes	<none>
7	TOTAL_LOCAL_CURRENCY	☐	Decimal	10	2	Yes	<none>
8	CUSTOMER_ID	☐	Integer	10		Yes	<none>
9	STORE_ID	☐	Integer	10		Yes	<none>
10	PRODUCT_ID	☐	Integer	10		Yes	<none>
11	COUNTRY_ISO_CODE	☐	Char	3		Yes	<none>
12	NAME	☐	VarChar	50		Yes	<none>
13	HOME_PHONE	☐	Char	12		Yes	<none>
14	WORK_PHONE	☐	Char	12		Yes	<none>
15	WORK_ADDRESS	☐	VarChar	50		Yes	<none>
16	WORK_CITY	☐	VarChar	50		Yes	<none>
17	WORK_STATE	☐	VarChar	50		Yes	<none>
18	WORK_ZIP	☐	VarChar	15		Yes	<none>
19	WORK_COUNTRY	☐	VarChar	50		Yes	<none>
20	HOME_ADDRESS	☐	VarChar	50		Yes	<none>
21	HOME_CITY	☐	VarChar	50		Yes	<none>
22	HOME_ZIP	☐	VarChar	15		Yes	<none>

Figure 3-371 Create the J15_Daily_CreateSalesAggDS job 13/41



	Column name	Key	SQL type	Length	Scale	Nullable	Description
18	WORK_ZIP	☐	VarChar	15		Yes	<none>
19	WORK_COUNTRY	☐	VarChar	50		Yes	<none>
20	HOME_ADDRESS	☐	VarChar	50		Yes	<none>
21	HOME_CITY	☐	VarChar	50		Yes	<none>
22	HOME_ZIP	☐	VarChar	15		Yes	<none>
23	HOME_STATE	☐	VarChar	50		Yes	<none>
24	HOME_COUNTRY	☐	VarChar	50		Yes	<none>
25	MEMBERSHIP_ID	☐	Integer	10		Yes	<none>
26	MEMBERSHIP_EXPIRE_DT	☐	Date	10		Yes	<none>
27	MEMBERSHIP_LEVEL	☐	Char	1		Yes	<none>
28	C_TRANSACTION_TS	☐	Timestamp	26	6	Yes	<none>
29	ADDRESS	☐	VarChar	50		Yes	
30	CITY	☐	VarChar	50		Yes	
31	CITY_POPULATION	☐	Decimal	8		Yes	
32	STATE	☐	VarChar	50		Yes	
33	STATE_POPULATION	☐	Decimal	8		Yes	
34	ZIP	☐	VarChar	15		Yes	
35	COUNTRY	☐	VarChar	50		Yes	
36	MANAGER_NAME	☐	VarChar	50		Yes	
37	S_TRANSACTION_TS	☐	Timestamp	26	6	Yes	
		☐					

Figure 3-372 Create the J15_Daily_CreateSalesAggDS job 14/41

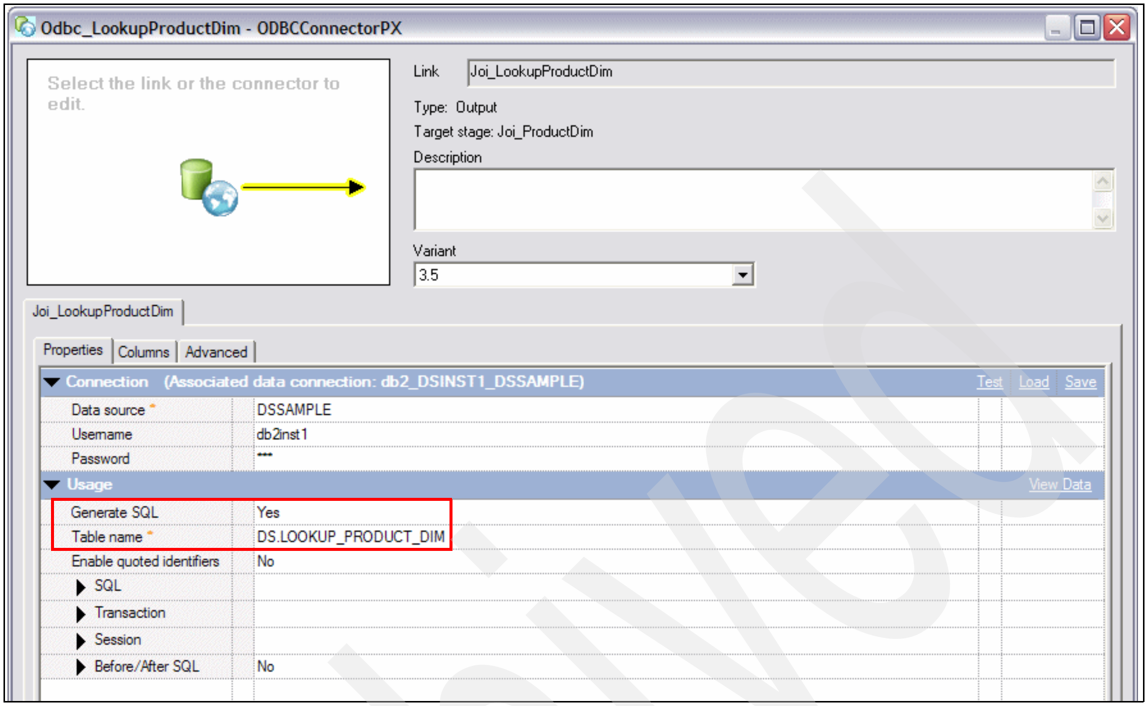


Figure 3-373 Create the J15_Daily_CreateSalesAggDS job 15/41

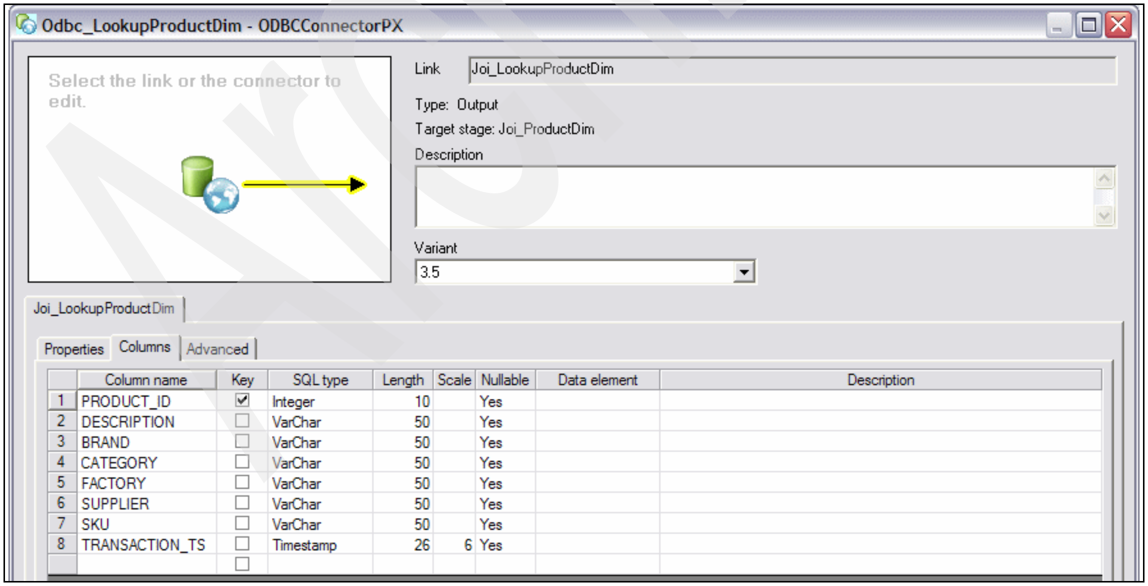


Figure 3-374 Create the J15_Daily_CreateSalesAggDS job 16/41

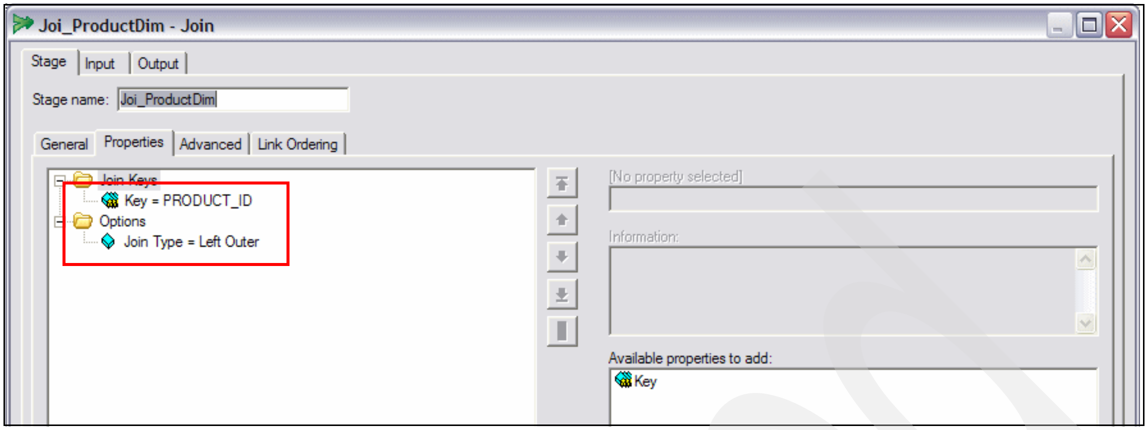


Figure 3-375 Create the J15_Daily_CreateSalesAggDS job 17/41

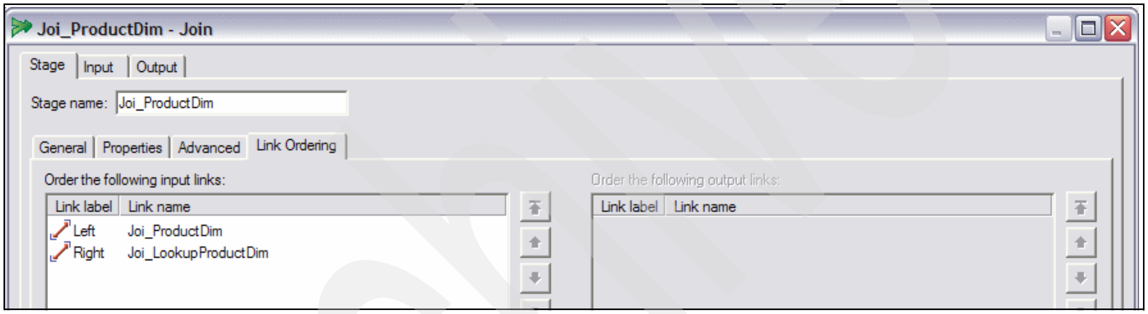


Figure 3-376 Create the J15_Daily_CreateSalesAggDS job 18/41

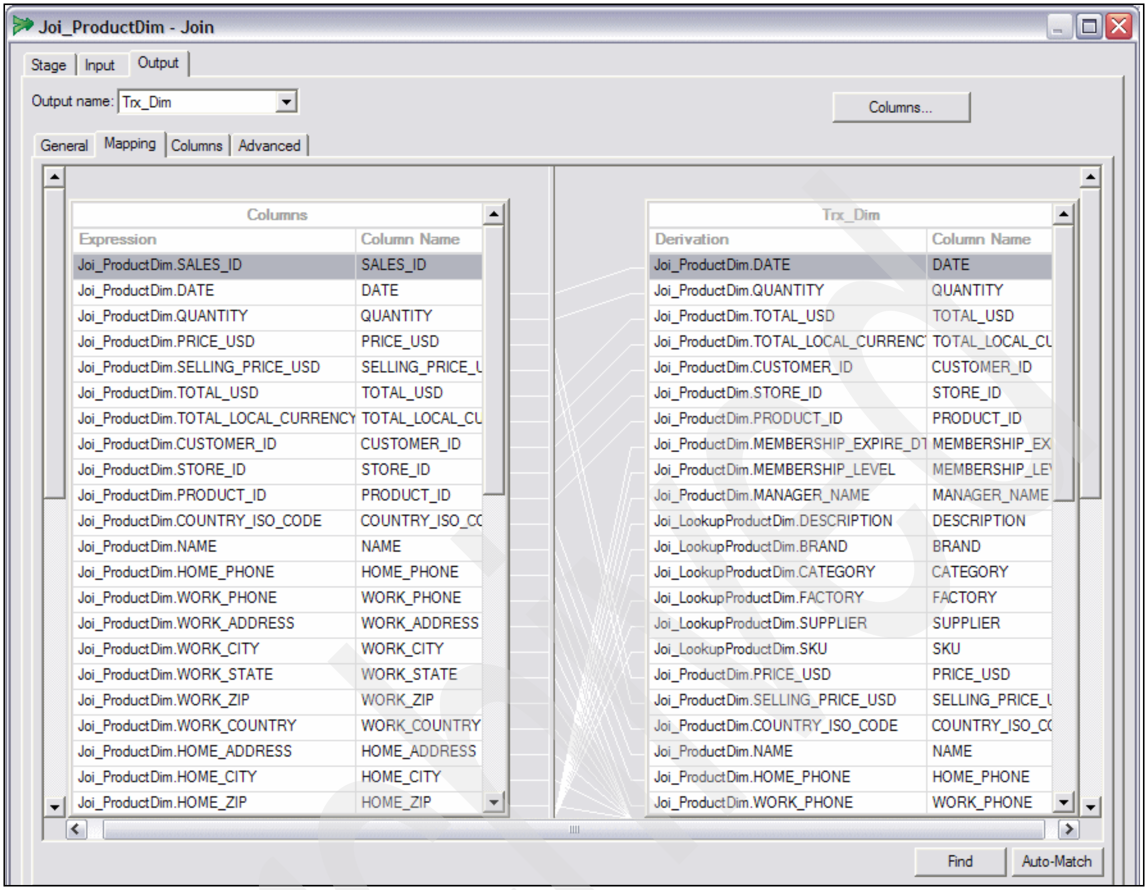


Figure 3-377 Create the J15_Daily_CreateSalesAggDS job 19/41

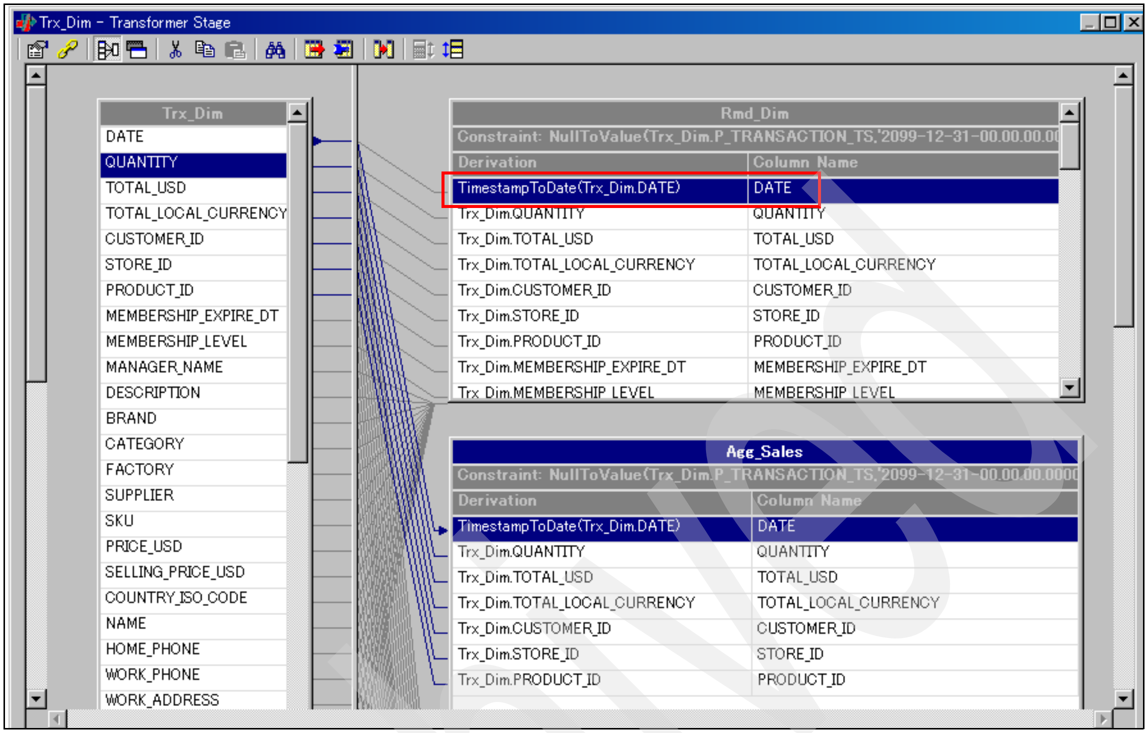


Figure 3-378 Create the J15_Daily_CreateSalesAggDS job 20/41

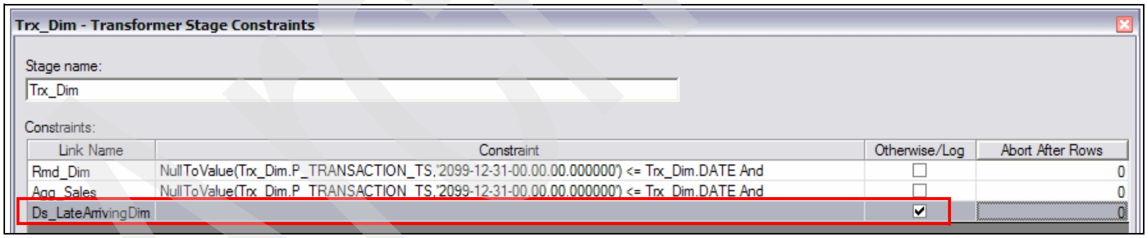


Figure 3-379 Create the J15_Daily_CreateSalesAggDS job 21/41

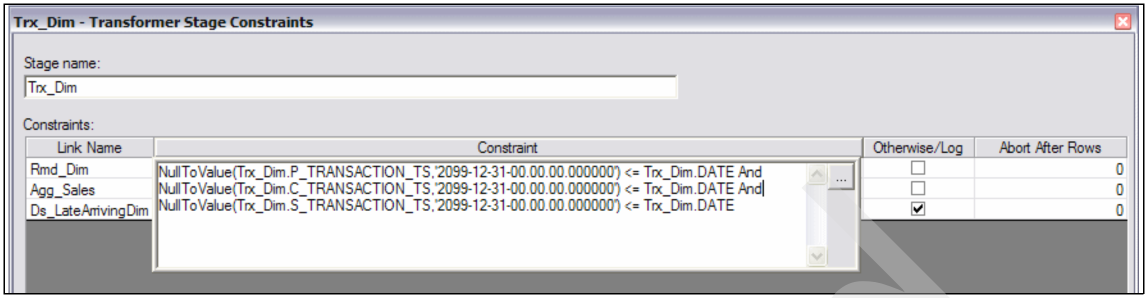


Figure 3-380 Create the J15_Daily_CreateSalesAggDS job 22/41

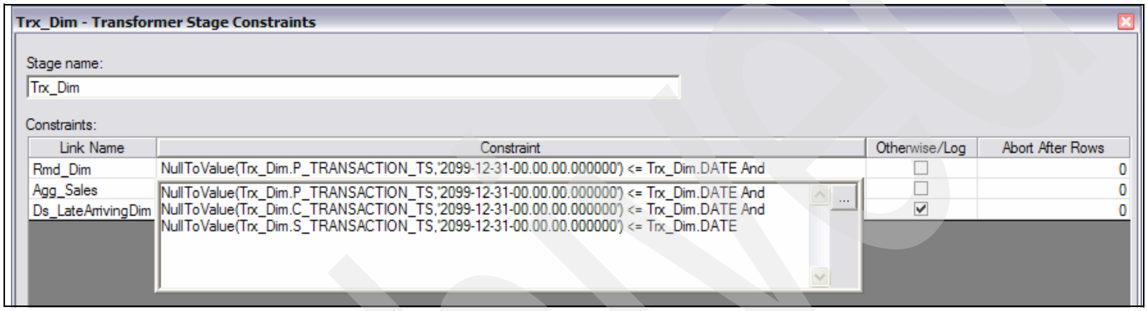


Figure 3-381 Create the J15_Daily_CreateSalesAggDS job 23/41

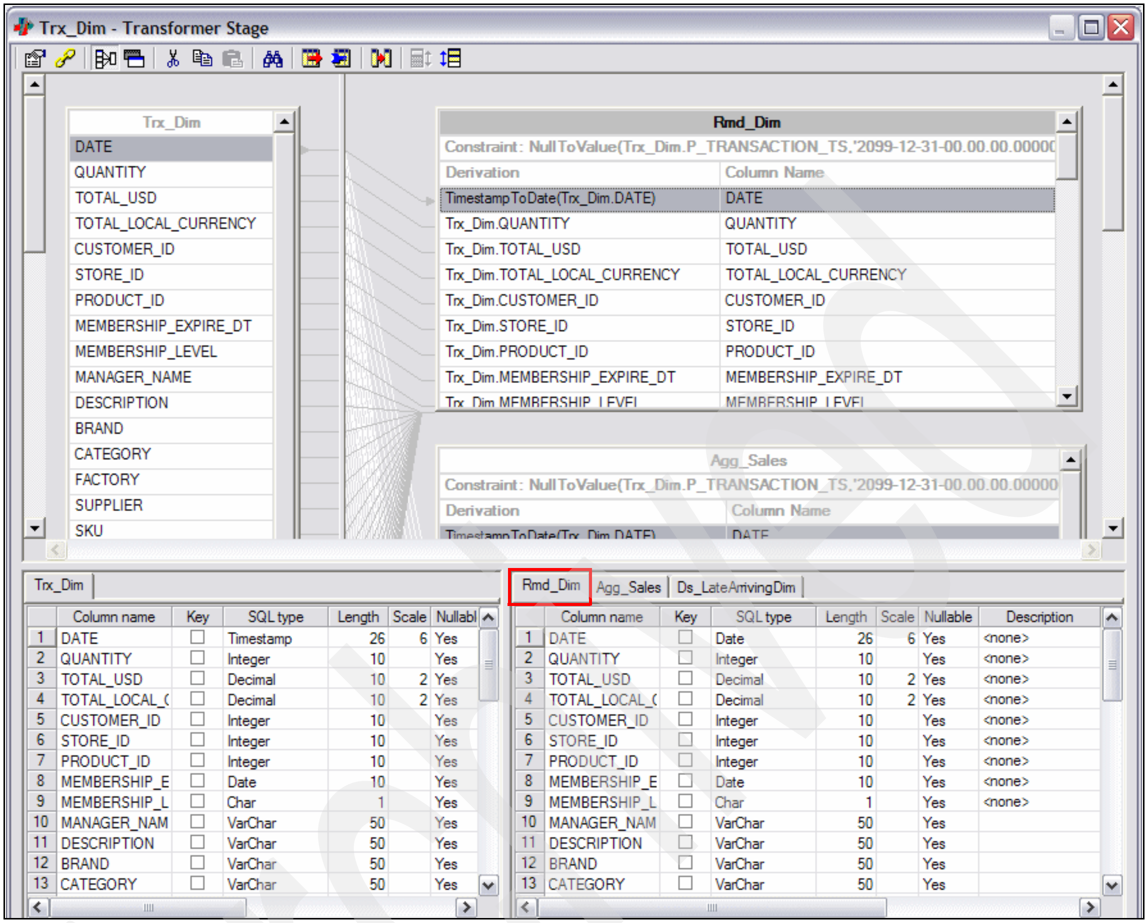


Figure 3-382 Create the J15_Daily_CreateSalesAggDS job 24/41

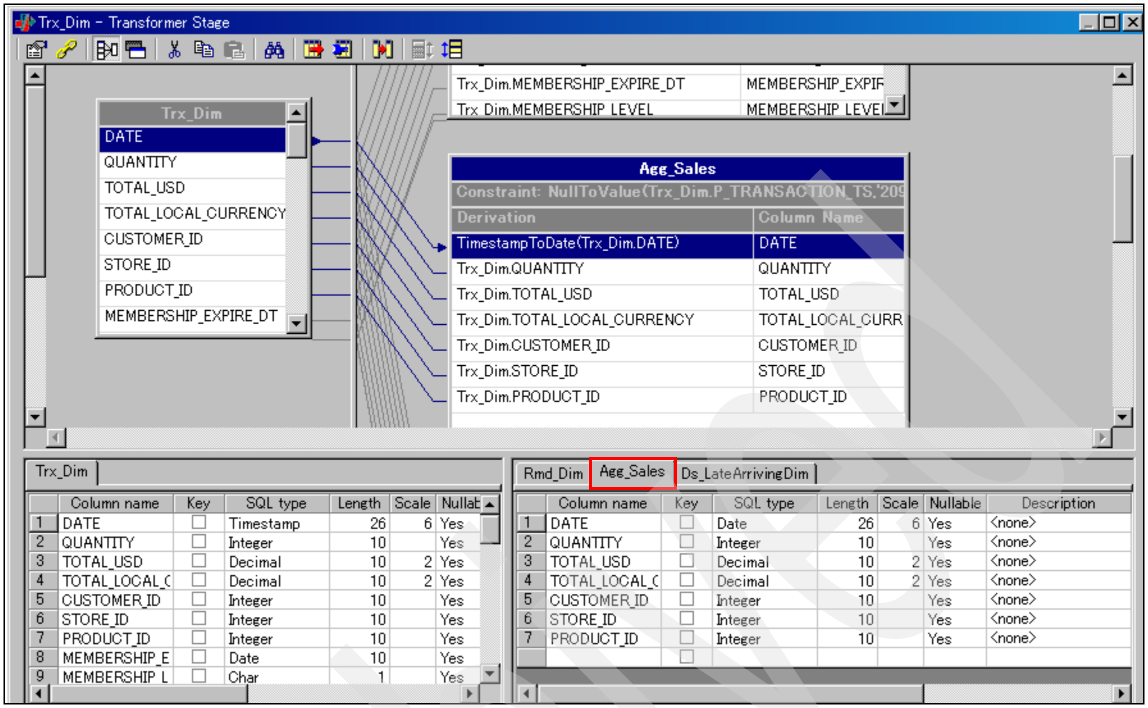


Figure 3-383 Create the J15_Daily_CreateSalesAggDS job 25/41

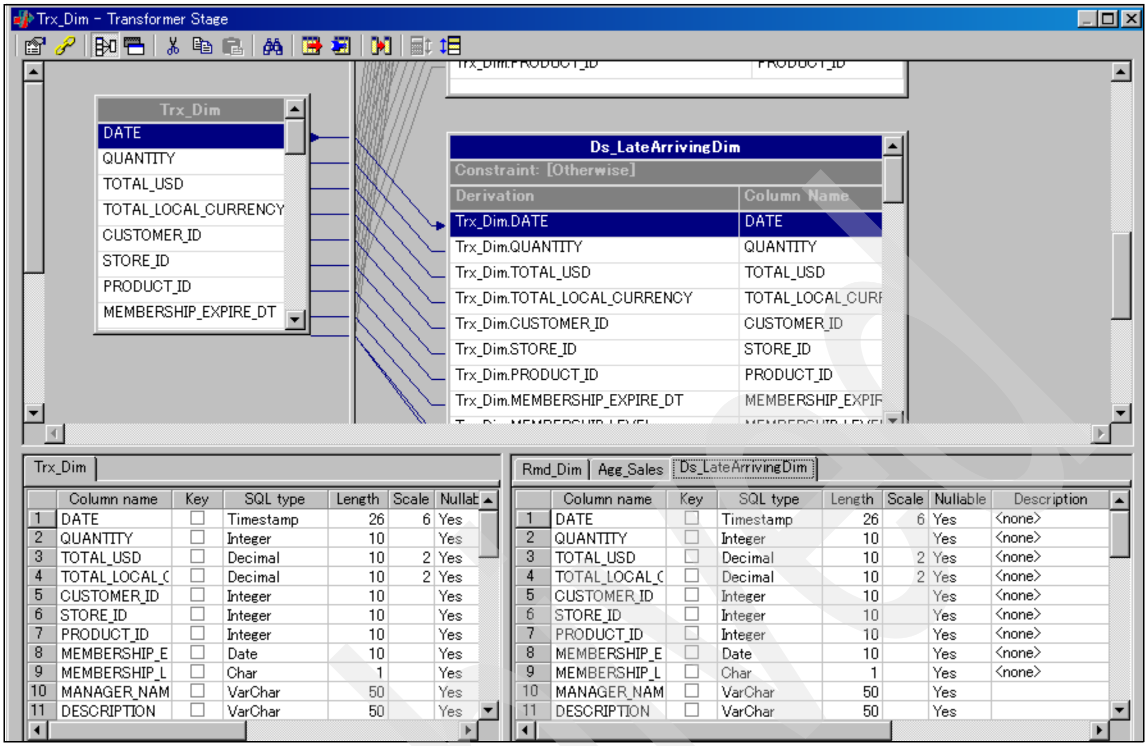


Figure 3-384 Create the J15_Daily_CreateSalesAggDS job 26/41

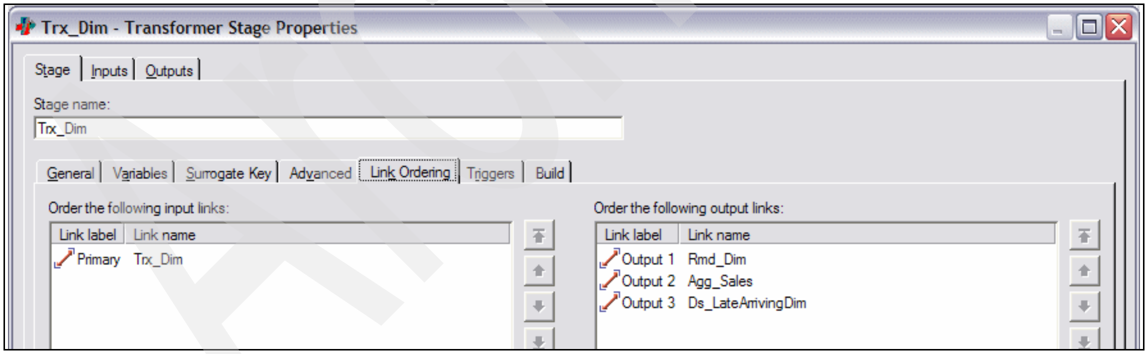


Figure 3-385 Create the J15_Daily_CreateSalesAggDS job 27/41

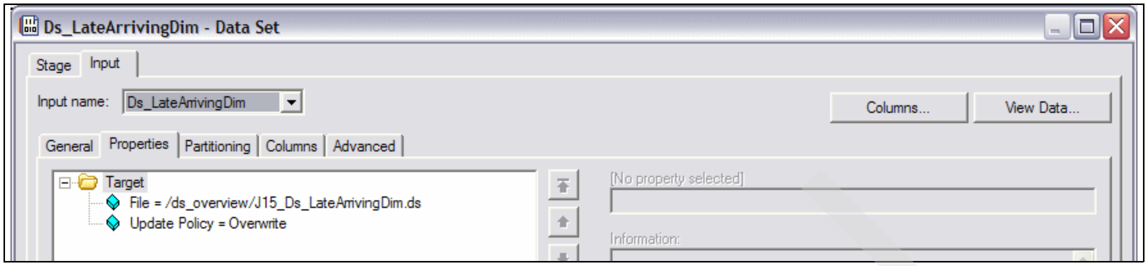


Figure 3-386 Create the J15_Daily_CreateSalesAggDS job 28/41

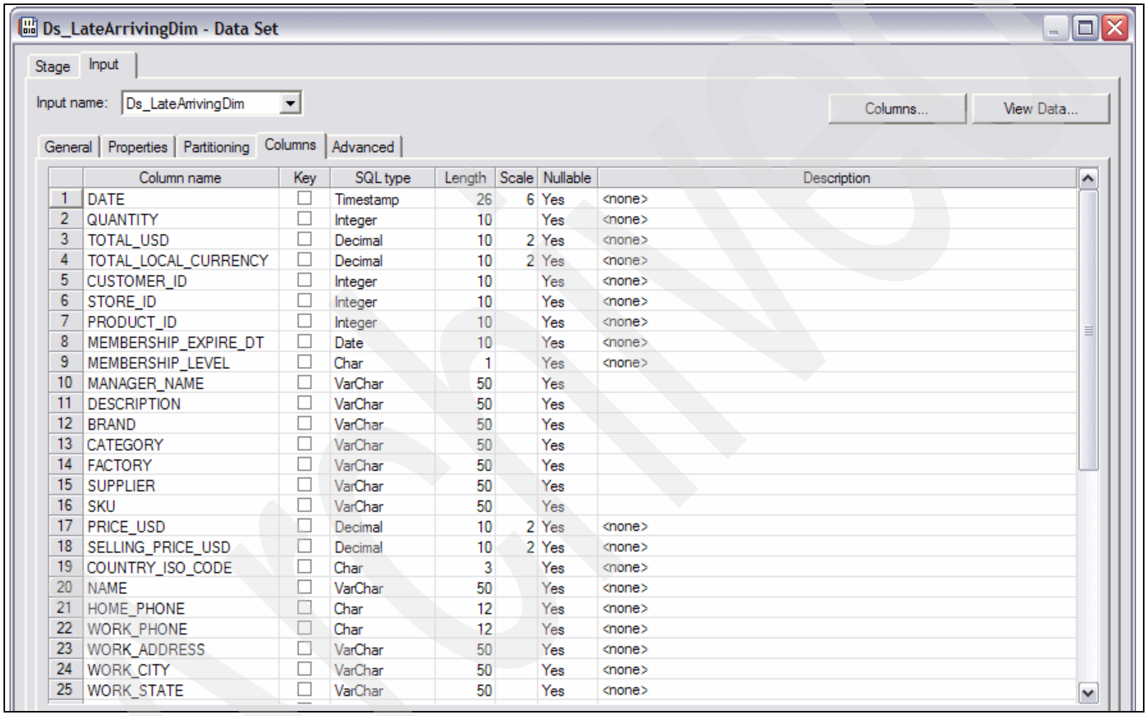


Figure 3-387 Create the J15_Daily_CreateSalesAggDS job 29/41

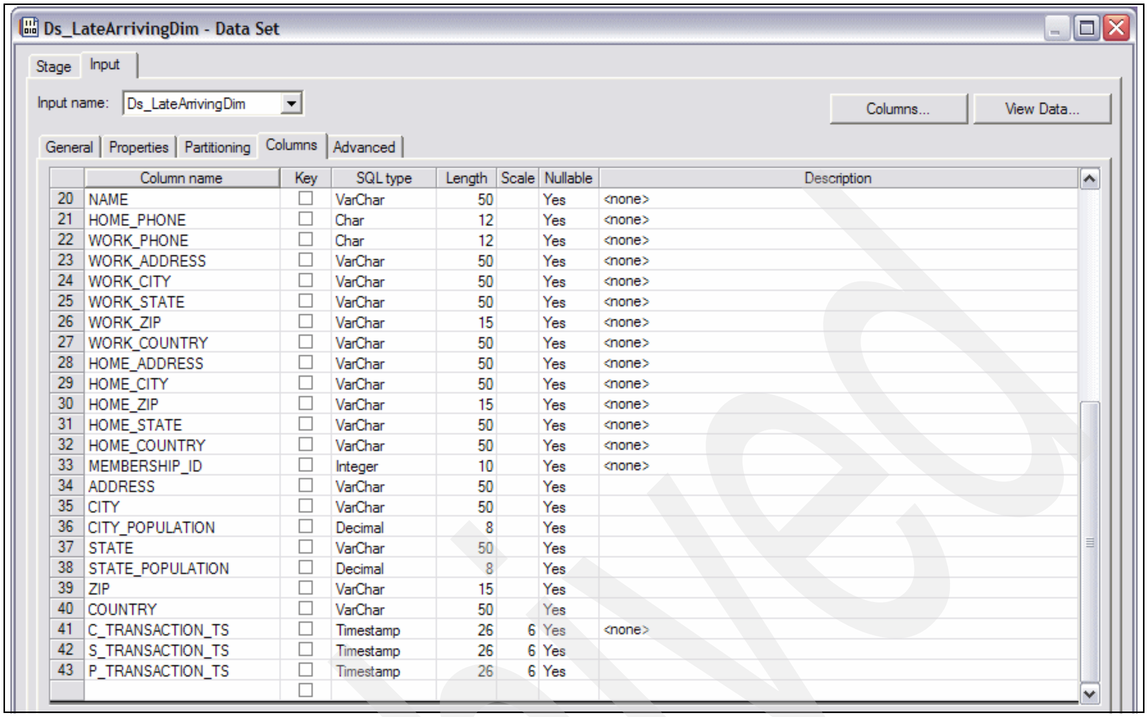


Figure 3-388 Create the J15_Daily_CreateSalesAggDS job 30/41

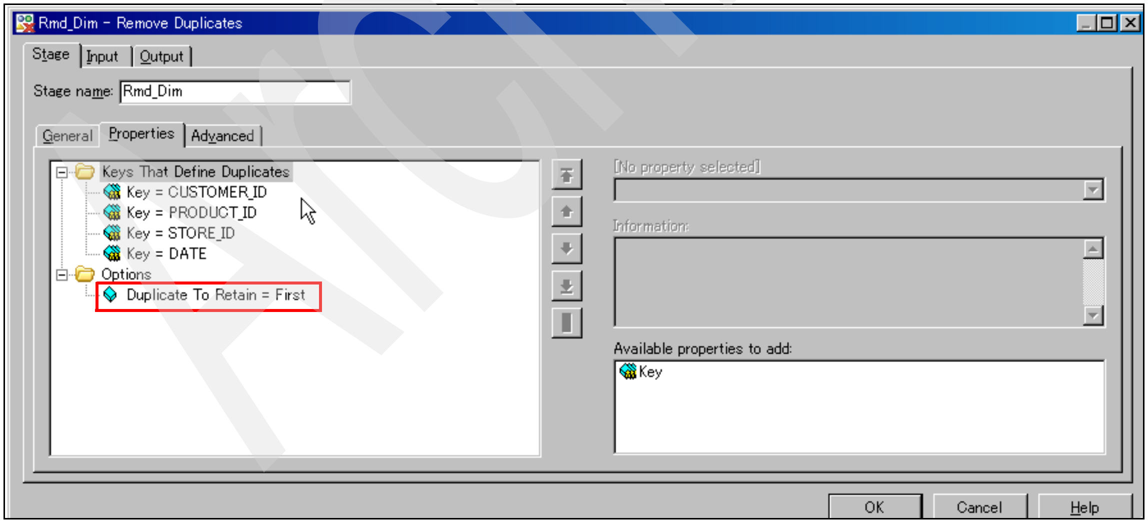


Figure 3-389 Create the J15_Daily_CreateSalesAggDS job 31/41

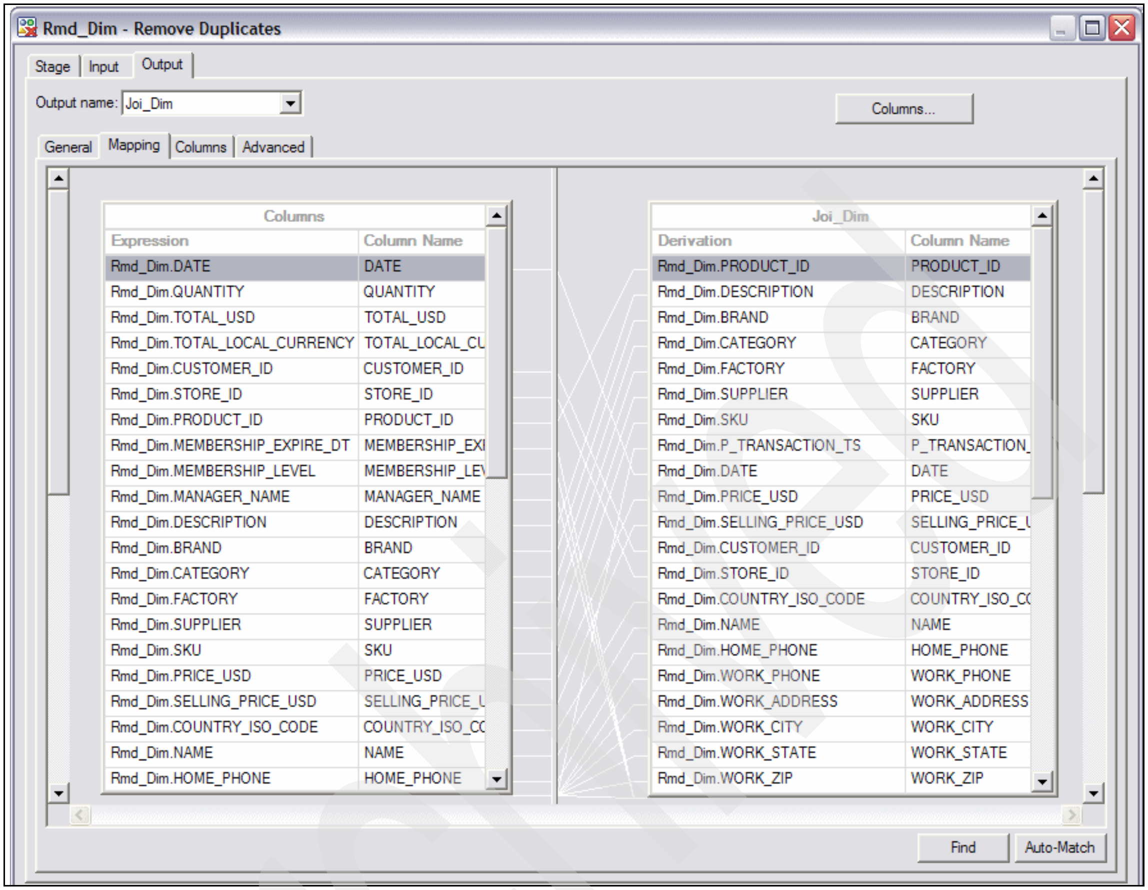


Figure 3-390   Create the J15_Daily_CreateSalesAggDS job 32/41

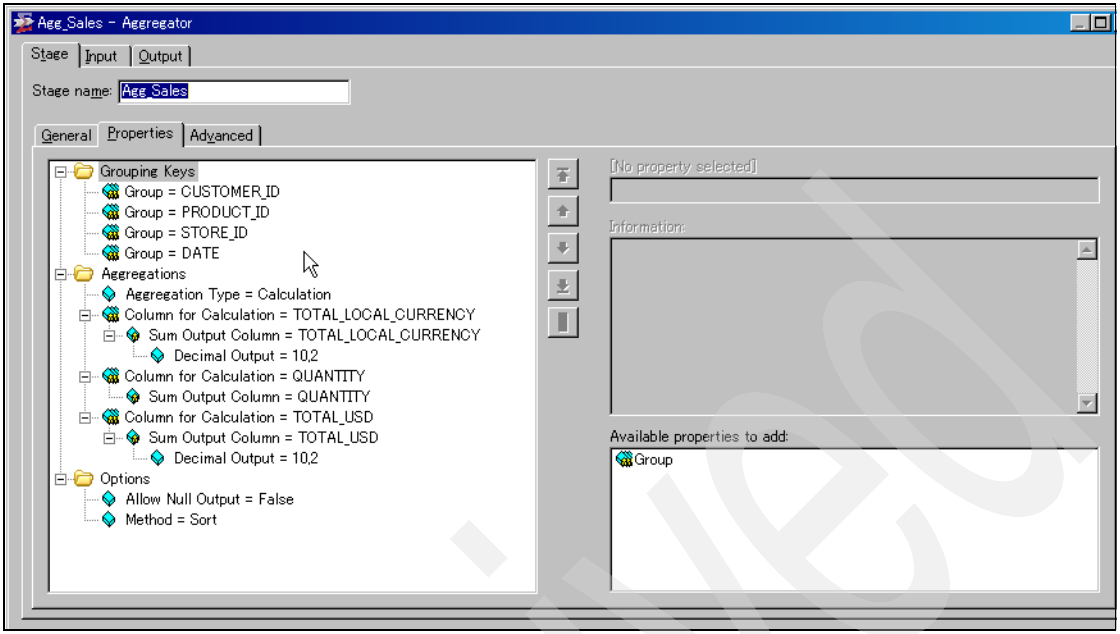


Figure 3-391 Create the J15_Daily_CreateSalesAggDS job 33/41

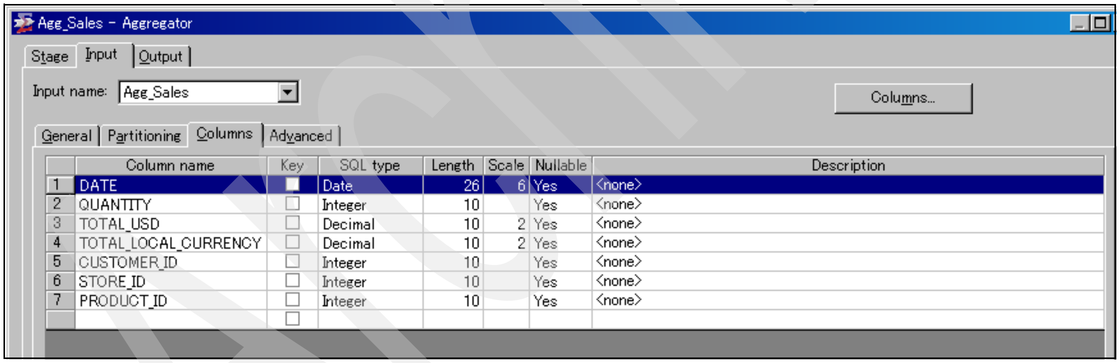


Figure 3-392 Create the J15_Daily_CreateSalesAggDS job 34/41

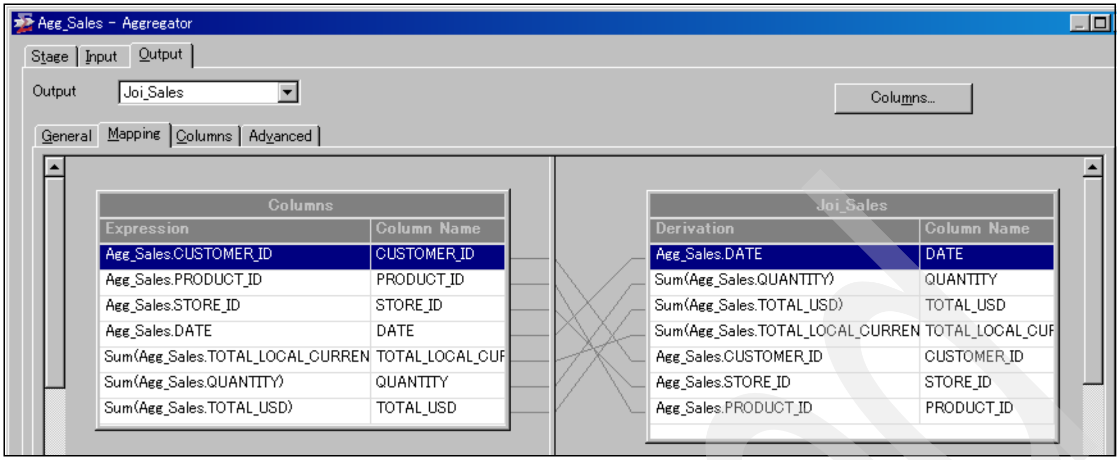


Figure 3-393 Create the J15_Daily_CreateSalesAggDS job 35/41

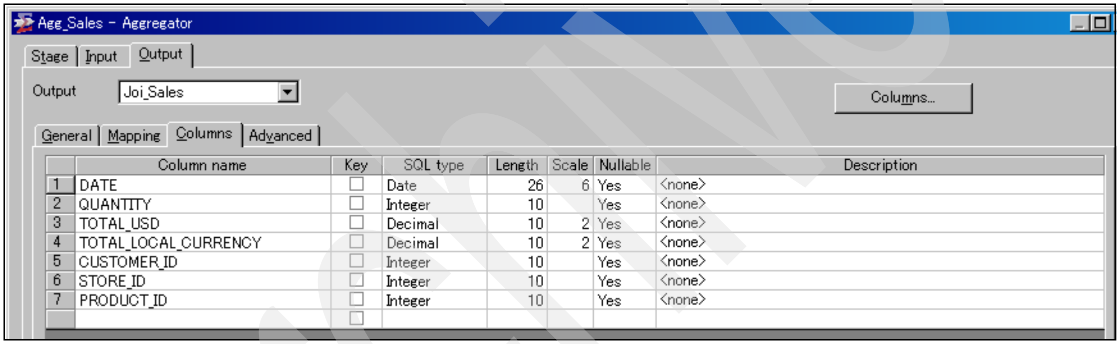


Figure 3-394 Create the J15_Daily_CreateSalesAggDS job 36/41

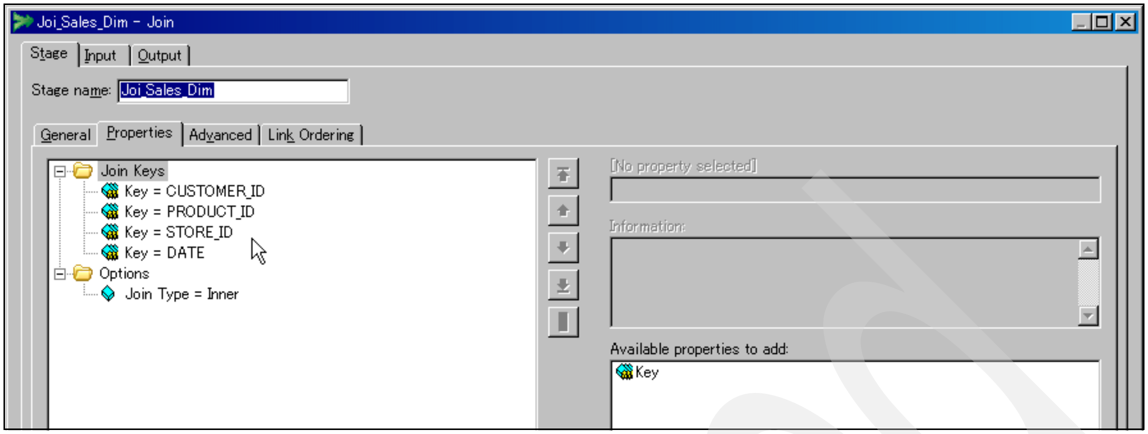


Figure 3-395 Create the J15_Daily_CreateSalesAggDS job 37/41

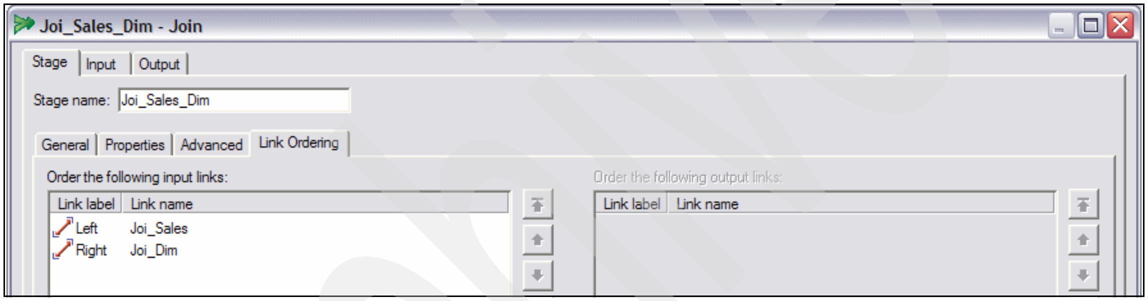


Figure 3-396 Create the J15_Daily_CreateSalesAggDS job 38/41

b

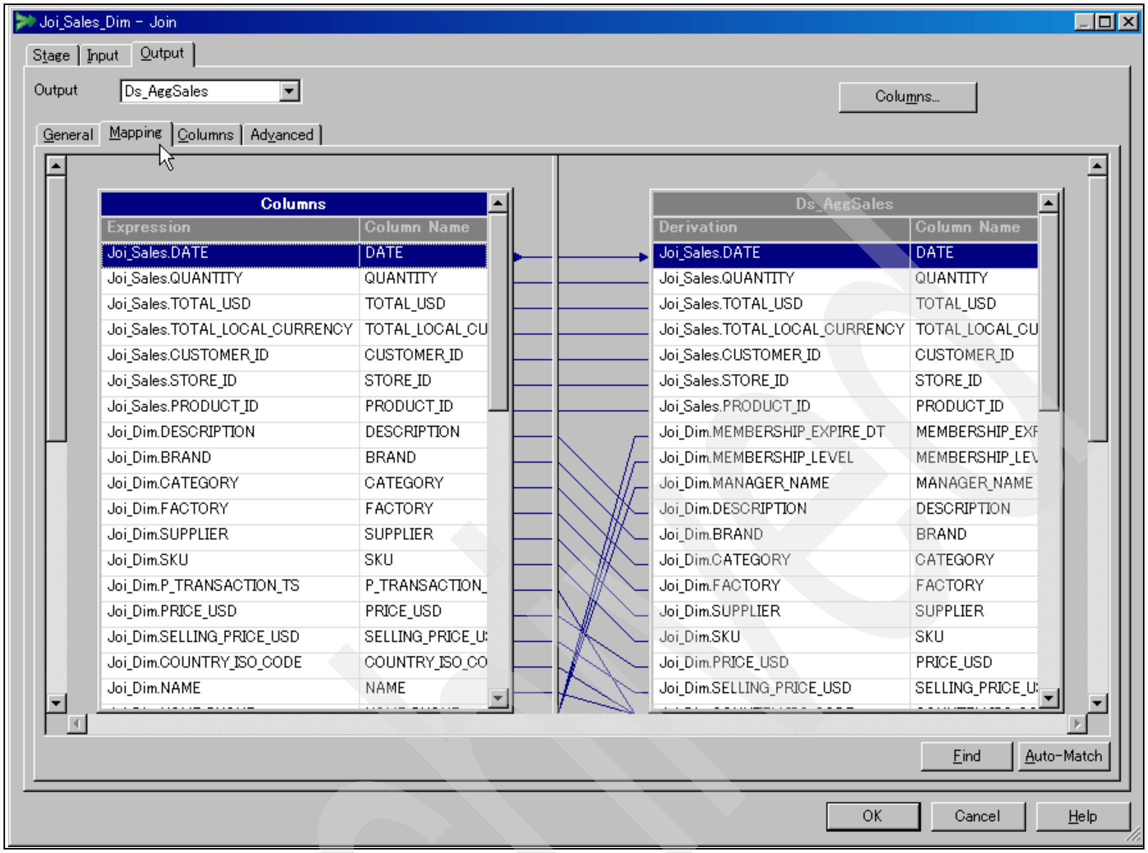
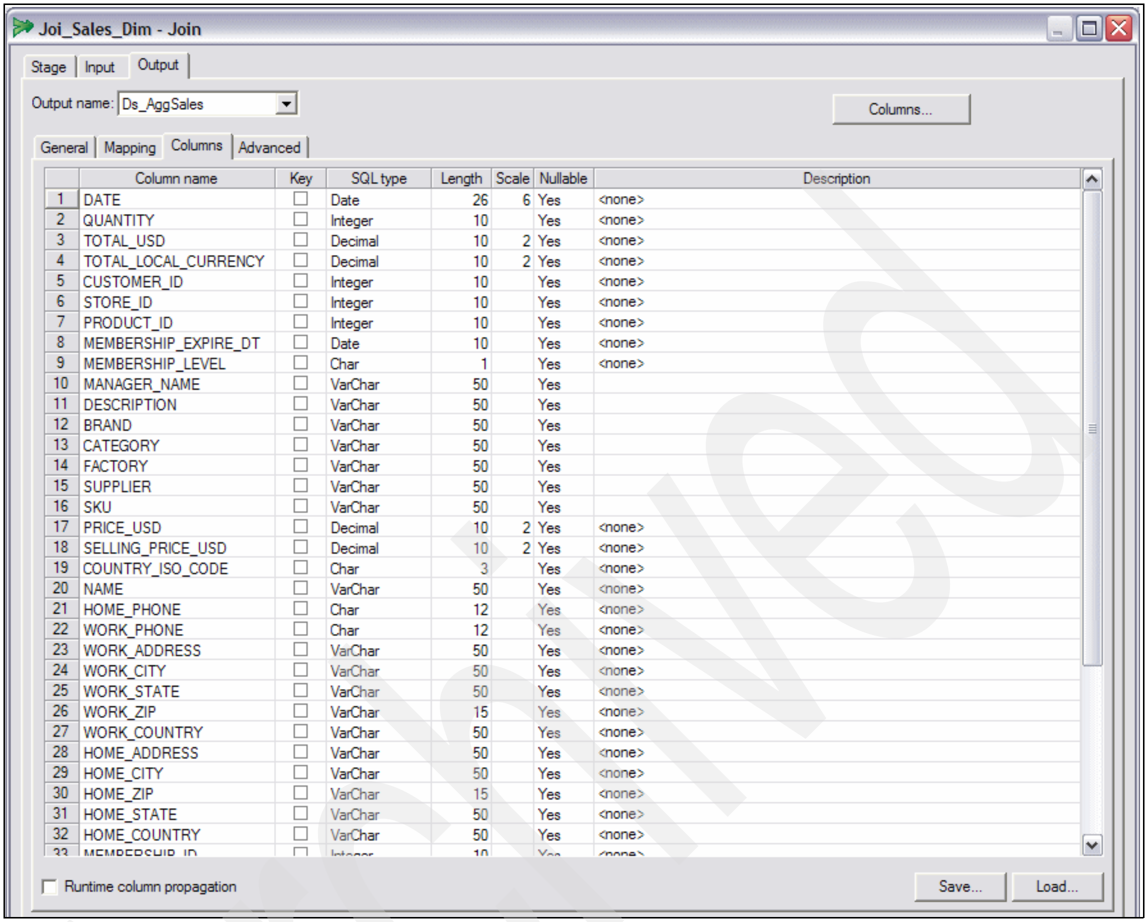
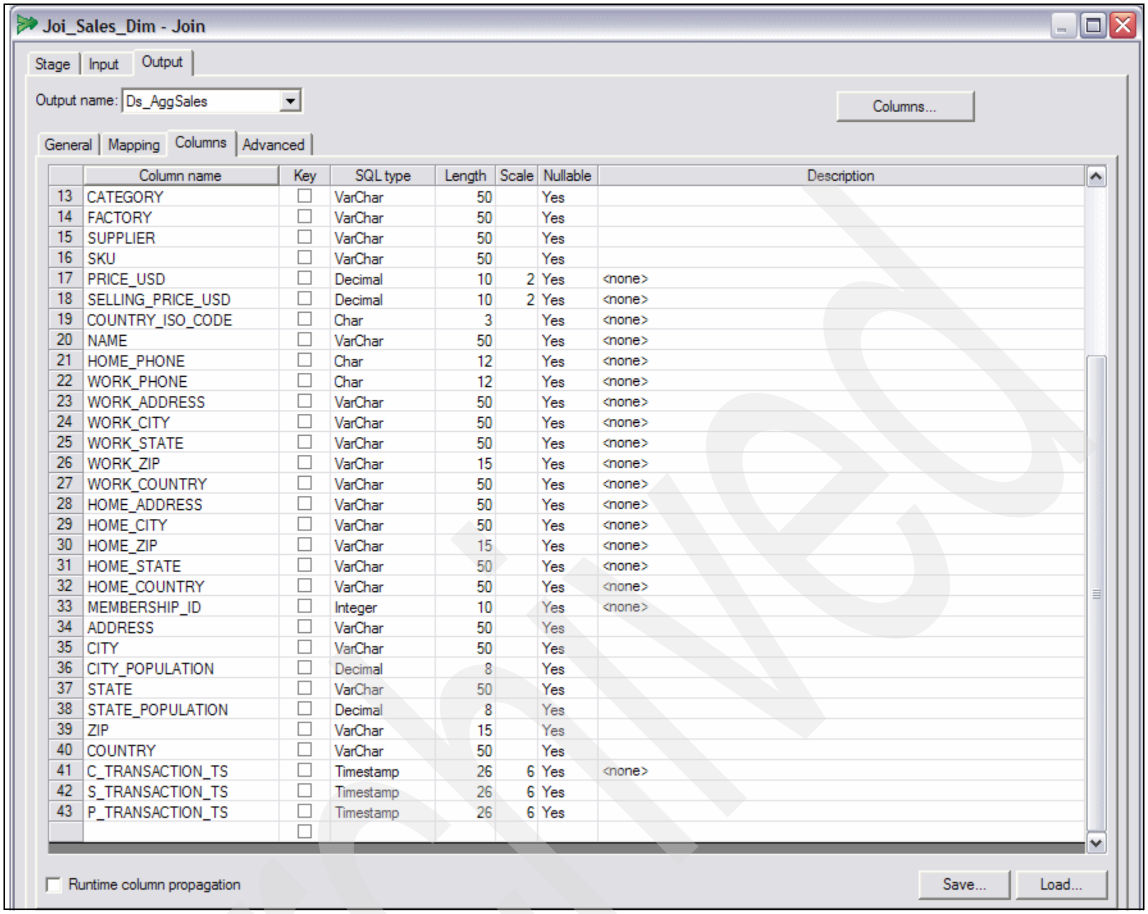


Figure 3-397 Create the J15_Daily_CreateSalesAggDS job 39/41



	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	DATE		Date	26	6	Yes	<none>
2	QUANTITY		Integer	10		Yes	<none>
3	TOTAL_USD		Decimal	10	2	Yes	<none>
4	TOTAL_LOCAL_CURRENCY		Decimal	10	2	Yes	<none>
5	CUSTOMER_ID		Integer	10		Yes	<none>
6	STORE_ID		Integer	10		Yes	<none>
7	PRODUCT_ID		Integer	10		Yes	<none>
8	MEMBERSHIP_EXPIRE_DT		Date	10		Yes	<none>
9	MEMBERSHIP_LEVEL		Char	1		Yes	<none>
10	MANAGER_NAME		VarChar	50		Yes	
11	DESCRIPTION		VarChar	50		Yes	
12	BRAND		VarChar	50		Yes	
13	CATEGORY		VarChar	50		Yes	
14	FACTORY		VarChar	50		Yes	
15	SUPPLIER		VarChar	50		Yes	
16	SKU		VarChar	50		Yes	
17	PRICE_USD		Decimal	10	2	Yes	<none>
18	SELLING_PRICE_USD		Decimal	10	2	Yes	<none>
19	COUNTRY_ISO_CODE		Char	3		Yes	<none>
20	NAME		VarChar	50		Yes	<none>
21	HOME_PHONE		Char	12		Yes	<none>
22	WORK_PHONE		Char	12		Yes	<none>
23	WORK_ADDRESS		VarChar	50		Yes	<none>
24	WORK_CITY		VarChar	50		Yes	<none>
25	WORK_STATE		VarChar	50		Yes	<none>
26	WORK_ZIP		VarChar	15		Yes	<none>
27	WORK_COUNTRY		VarChar	50		Yes	<none>
28	HOME_ADDRESS		VarChar	50		Yes	<none>
29	HOME_CITY		VarChar	50		Yes	<none>
30	HOME_ZIP		VarChar	15		Yes	<none>
31	HOME_STATE		VarChar	50		Yes	<none>
32	HOME_COUNTRY		VarChar	50		Yes	<none>
33	MEMBERSHIP_ID		Integer	10		Yes	<none>

Figure 3-398 Create the J15_Daily_CreateSalesAggDS job 40/41



	Column name	Key	SQL type	Length	Scale	Nullable	Description
13	CATEGORY		VarChar	50		Yes	
14	FACTORY		VarChar	50		Yes	
15	SUPPLIER		VarChar	50		Yes	
16	SKU		VarChar	50		Yes	
17	PRICE_USD		Decimal	10	2	Yes	<none>
18	SELLING_PRICE_USD		Decimal	10	2	Yes	<none>
19	COUNTRY_ISO_CODE		Char	3		Yes	<none>
20	NAME		VarChar	50		Yes	<none>
21	HOME_PHONE		Char	12		Yes	<none>
22	WORK_PHONE		Char	12		Yes	<none>
23	WORK_ADDRESS		VarChar	50		Yes	<none>
24	WORK_CITY		VarChar	50		Yes	<none>
25	WORK_STATE		VarChar	50		Yes	<none>
26	WORK_ZIP		VarChar	15		Yes	<none>
27	WORK_COUNTRY		VarChar	50		Yes	<none>
28	HOME_ADDRESS		VarChar	50		Yes	<none>
29	HOME_CITY		VarChar	50		Yes	<none>
30	HOME_ZIP		VarChar	15		Yes	<none>
31	HOME_STATE		VarChar	50		Yes	<none>
32	HOME_COUNTRY		VarChar	50		Yes	<none>
33	MEMBERSHIP_ID		Integer	10		Yes	<none>
34	ADDRESS		VarChar	50		Yes	
35	CITY		VarChar	50		Yes	
36	CITY_POPULATION		Decimal	8		Yes	
37	STATE		VarChar	50		Yes	
38	STATE_POPULATION		Decimal	8		Yes	
39	ZIP		VarChar	15		Yes	
40	COUNTRY		VarChar	50		Yes	
41	C_TRANSACTION_TS		Timestamp	26	6	Yes	<none>
42	S_TRANSACTION_TS		Timestamp	26	6	Yes	
43	P_TRANSACTION_TS		Timestamp	26	6	Yes	

Figure 3-399 Create the J15_Daily_CreateSalesAggDS job 41/41

J15_Daily_CreateSalesAggDS (Day 1) execution

Figure 3-400 on page 418 through Figure 3-412 on page 421 show the results of the execution of this job with Day 1 data described earlier.

- Figure 3-400 on page 418 shows the results of the execution. It accepts 13 rows as input from the "J14_Daily_CreateAllSalesStoreDS (Day 1)" on page 385 job as seen in Figure 3-357 on page 386 and Figure 3-358 on page 387.
- The two outputs of this job are:
  - The aggregated sales transactions appended with the dimension lookup tables. This is a total of 7 rows as seen in Figure 3-401 on page 418 through Figure 3-406 on page 420.

– The rejected sales transactions (either late arriving dimensions or late arriving data). This is a total of 6 rows as seen in Figure 3-407 on page 420 through Figure 3-412 on page 421. The invalid column values are highlighted.

The next step is to execute the job described in “J16_Daily_CreateScdInputDS (Day 1)” on page 421.

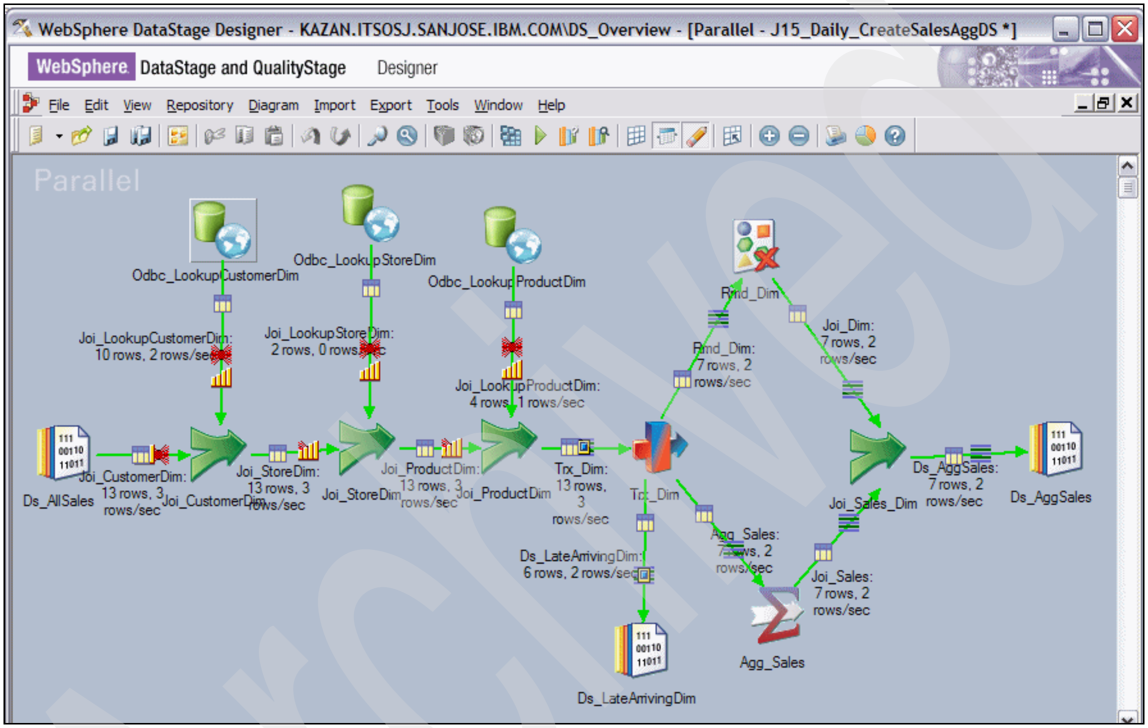
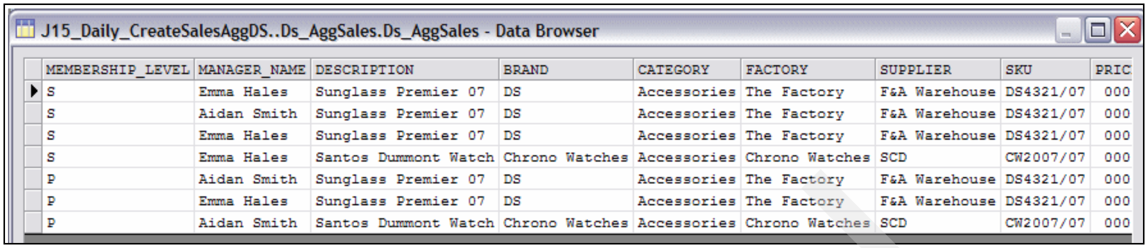


Figure 3-400 Execute the J15_Daily_CreateSalesAggDS job (Day 1) 1/13

J15_Daily_CreateSalesAggDS..Ds_AggSales.Ds_AggSales - Data Browser

DATE	QUANTITY	TOTAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	MEMBERSHIP_EXPIRE_DT	MEMBERSHIP_LE
2007-11-06	2	00000050.00	00000050.00	1	33	1	2012-02-16	S
2007-11-06	2	00000030.00	00000030.00	6	1	1	2012-02-21	S
2007-11-06	1	00000037.00	00000037.00	9	33	1	2012-02-24	S
2007-11-06	3	00000111.00	00000111.00	10	33	2	2012-02-25	S
2007-11-06	2	00000030.00	00000030.00	9999	1	1	2999-12-31	P
2007-11-06	1	00000033.33	00000033.33	9999	33	1	2999-12-31	P
2007-11-06	2	00000030.00	00000029.02	9999	1	2	2999-12-31	P

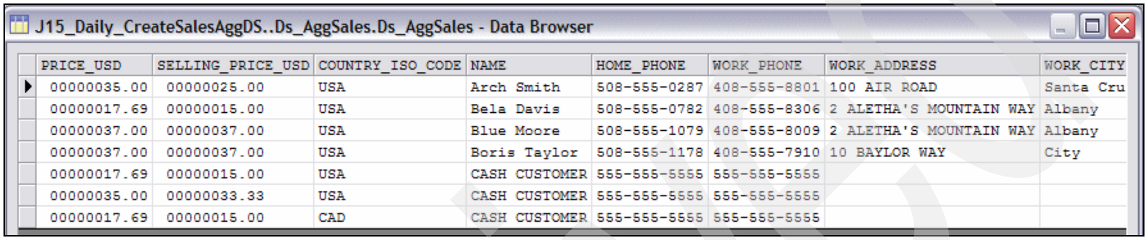
Figure 3-401 Execute the J15_Daily_CreateSalesAggDS job (Day 1) 2/13



J15_Daily_CreateSalesAggDS..Ds_AggSales.Ds_AggSales - Data Browser

MEMBERSHIP_LEVEL	MANAGER_NAME	DESCRIPTION	BRAND	CATEGORY	FACTORY	SUPPLIER	SKU	PRIC
S	Emma Hales	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	000
S	Aidan Smith	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	000
S	Emma Hales	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	000
S	Emma Hales	Santos Dummont Watch	Chrono Watches	Accessories	Chrono Watches	SCD	CW2007/07	000
P	Aidan Smith	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	000
P	Emma Hales	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	000
P	Aidan Smith	Santos Dummont Watch	Chrono Watches	Accessories	Chrono Watches	SCD	CW2007/07	000

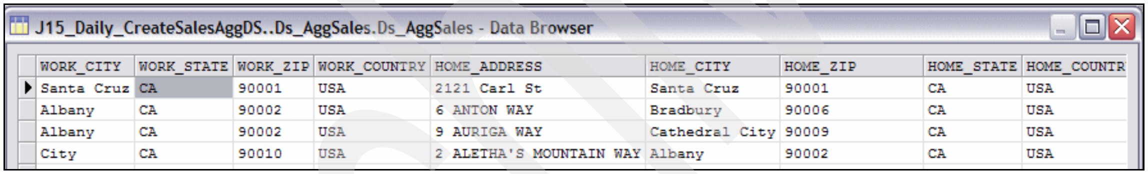
Figure 3-402 Execute the J15_Daily_CreateSalesAggDS job (Day 1) 3/13



J15_Daily_CreateSalesAggDS..Ds_AggSales.Ds_AggSales - Data Browser

PRICE_USD	SELLING_PRICE_USD	COUNTRY_ISO_CODE	NAME	HOME_PHONE	WORK_PHONE	WORK_ADDRESS	WORK_CITY
00000035.00	00000025.00	USA	Arch Smith	508-555-0287	408-555-8801	100 AIR ROAD	Santa Cru
00000017.69	00000015.00	USA	Bela Davis	508-555-0782	408-555-8306	2 ALETHA'S MOUNTAIN WAY	Albany
00000037.00	00000037.00	USA	Blue Moore	508-555-1079	408-555-8009	2 ALETHA'S MOUNTAIN WAY	Albany
00000037.00	00000037.00	USA	Boris Taylor	508-555-1178	408-555-7910	10 BAYLOR WAY	City
00000017.69	00000015.00	USA	CASH CUSTOMER	555-555-5555	555-555-5555		
00000035.00	00000033.33	USA	CASH CUSTOMER	555-555-5555	555-555-5555		
00000017.69	00000015.00	CAD	CASH CUSTOMER	555-555-5555	555-555-5555		

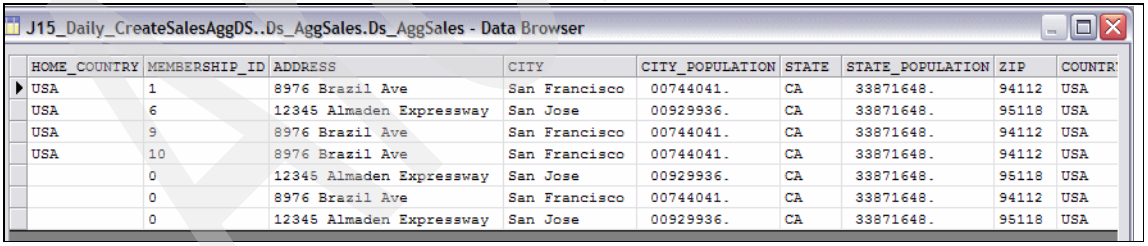
Figure 3-403 Execute the J15_Daily_CreateSalesAggDS job (Day 1) 4/13



J15_Daily_CreateSalesAggDS..Ds_AggSales.Ds_AggSales - Data Browser

WORK_CITY	WORK_STATE	WORK_ZIP	WORK_COUNTRY	HOME_ADDRESS	HOME_CITY	HOME_ZIP	HOME_STATE	HOME_COUNTR
Santa Cruz	CA	90001	USA	2121 Carl St	Santa Cruz	90001	CA	USA
Albany	CA	90002	USA	6 ANTON WAY	Bradbury	90006	CA	USA
Albany	CA	90002	USA	9 AURIGA WAY	Cathedral City	90009	CA	USA
City	CA	90010	USA	2 ALETHA'S MOUNTAIN WAY	Albany	90002	CA	USA

Figure 3-404 Execute the J15_Daily_CreateSalesAggDS job (Day 1) 5/13



J15_Daily_CreateSalesAggDS..Ds_AggSales.Ds_AggSales - Data Browser

HOME_COUNTRY	MEMBERSHIP_ID	ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTR
USA	1	8976 Brazil Ave	San Francisco	00744041.	CA	33871648.	94112	USA
USA	6	12345 Almaden Expressway	San Jose	00929936.	CA	33871648.	95118	USA
USA	9	8976 Brazil Ave	San Francisco	00744041.	CA	33871648.	94112	USA
USA	10	8976 Brazil Ave	San Francisco	00744041.	CA	33871648.	94112	USA
	0	12345 Almaden Expressway	San Jose	00929936.	CA	33871648.	95118	USA
	0	8976 Brazil Ave	San Francisco	00744041.	CA	33871648.	94112	USA
	0	12345 Almaden Expressway	San Jose	00929936.	CA	33871648.	95118	USA

Figure 3-405 Execute the J15_Daily_CreateSalesAggDS job (Day 1) 6/13

J15_Daily_CreateSalesAggDS..Ds_AggSales.Ds_AggSales - Data Browser

STATE	STATE_POPULATION	ZIP	COUNTRY	C_TRANSACTION_TS	S_TRANSACTION_TS	P_TRANSACTION_TS
CA	33871648.	94112	USA	2007-11-06 12:39:42.445734	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000
CA	33871648.	95118	USA	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000
CA	33871648.	94112	USA	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000
CA	33871648.	94112	USA	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000
CA	33871648.	95118	USA	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000
CA	33871648.	94112	USA	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000
CA	33871648.	95118	USA	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000

Figure 3-406 Execute the J15_Daily_CreateSalesAggDS job (Day 1) 7/13

J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

DATE	QUANTITY	TOTAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	MEMBERSHIP_EXPIRE_DT	MEMB
2007-11-06 23:49:42	1	00000075.00	00000075.00	9999	1	3	2999-12-31	P
2007-11-06 17:03:39	3	00000060.00	00000060.00	11	33	3	2012-05-10	P
2007-11-06 19:59:42	1	00000335.00	00000335.00	5	1	2	NULL	NULL
2007-11-06 23:42:42	1	00000033.33	00000033.33	3	1	11	2012-02-18	S
2007-11-06 18:39:33	1	00000075.00	00002983.50	9999	9	5	2999-12-31	P
2007-11-06 18:03:03	10	00000033.50	00000033.50	9999	99	5	2999-12-31	P

Figure 3-407 Execute the J15_Daily_CreateSalesAggDS job (Day 1) 8/13

J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

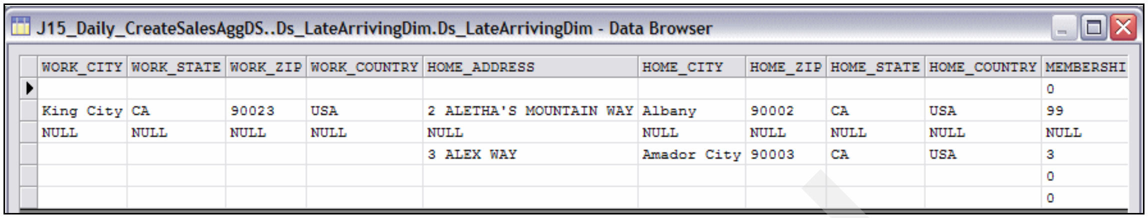
MEMBERSHIP_LEVEL	MANAGER_NAME	DESCRIPTION	BRAND	CATEGORY	FACTORY	SUPPLIER	SKU
P	Aidan Smith	NULL	NULL	NULL	NULL	NULL	NUL
P	Emma Hales	NULL	NULL	NULL	NULL	NULL	NUL
NULL	Aidan Smith	Santos Dummont Watch	Chrono Watches	Accessories	Chrono Watches	SCD	CW2
S	Aidan Smith	NULL	NULL	NULL	NULL	NULL	NUL
P	NULL	Neon Genesis Evangelion T-Shirt	JP Design	Accessories	JP Design	F&A Warehouse	JP0
P	NULL	Neon Genesis Evangelion T-Shirt	JP Design	Accessories	JP Design	F&A Warehouse	JP0

Figure 3-408 Execute the J15_Daily_CreateSalesAggDS job (Day 1) 9/13

J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

SKU	PRICE_USD	SELLING_PRICE_USD	COUNTRY_ISO_CODE	NAME	HOME_PHONE	WORK_PHONE	WORK_ADDRESS	WO
NULL	00000075.00	00000075.00	USA	CASH CUSTOMER	555-555-5555	555-555-5555		
NULL	00000020.00	00000020.00	USA	Desde Lewis	508-555-2465	408-555-6623	23 BRITTANY ROCK WAY	Ki
CW2007/07	00000335.00	00000335.00	USA	NULL	NULL	NULL	NULL	NU
NULL	00000033.33	00000033.33	USA	Barn Williams	508-555-0485	408-555-8603		
JP0819/08	00000075.00	00000075.00	IND	CASH CUSTOMER	555-555-5555	555-555-5555		
JP0819/08	00000003.35	00000003.35	USA	CASH CUSTOMER	555-555-5555	555-555-5555		

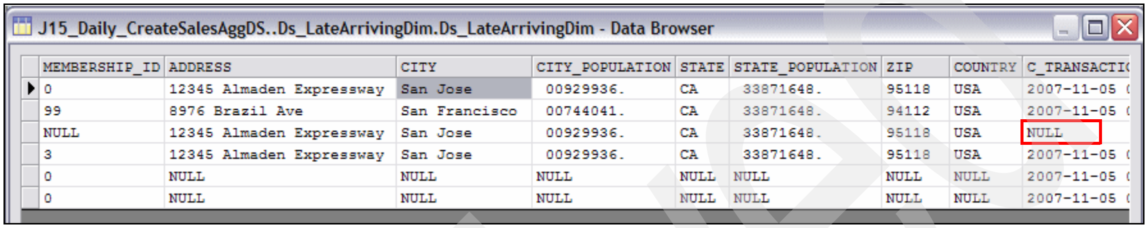
Figure 3-409 Execute the J15_Daily_CreateSalesAggDS job (Day 1) 10/13



J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

WORK_CITY	WORK_STATE	WORK_ZIP	WORK_COUNTRY	HOME_ADDRESS	HOME_CITY	HOME_ZIP	HOME_STATE	HOME_COUNTRY	MEMBERSHI
									0
King City	CA	90023	USA	2 ALETHA'S MOUNTAIN WAY	Albany	90002	CA	USA	99
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
				3 ALEX WAY	Amador City	90003	CA	USA	3
									0
									0

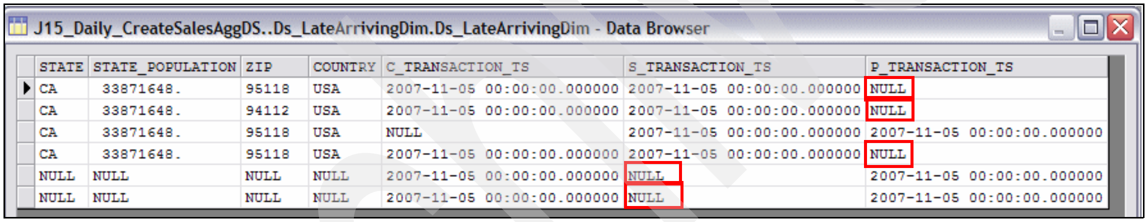
Figure 3-410 Execute the J15_Daily_CreateSalesAggDS job (Day 1) 11/13



J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

MEMBERSHIP_ID	ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	C_TRANSACTI(
0	12345 Almaden Expressway	San Jose	00929936.	CA	33871648.	95118	USA	2007-11-05 (
99	8976 Brazil Ave	San Francisco	00744041.	CA	33871648.	94112	USA	2007-11-05 (
NULL	12345 Almaden Expressway	San Jose	00929936.	CA	33871648.	95118	USA	NULL
3	12345 Almaden Expressway	San Jose	00929936.	CA	33871648.	95118	USA	2007-11-05 (
0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2007-11-05 (
0	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2007-11-05 (

Figure 3-411 Execute the J15_Daily_CreateSalesAggDS job (Day 1) 12/13



J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

STATE	STATE_POPULATION	ZIP	COUNTRY	C_TRANSACTION_TS	S_TRANSACTION_TS	P_TRANSACTION_TS
CA	33871648.	95118	USA	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000	NULL
CA	33871648.	94112	USA	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000	NULL
CA	33871648.	95118	USA	NULL	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000
CA	33871648.	95118	USA	2007-11-05 00:00:00.000000	2007-11-05 00:00:00.000000	NULL
NULL	NULL	NULL	NULL	2007-11-05 00:00:00.000000	NULL	2007-11-05 00:00:00.000000
NULL	NULL	NULL	NULL	2007-11-05 00:00:00.000000	NULL	2007-11-05 00:00:00.000000

Figure 3-412 Execute the J15_Daily_CreateSalesAggDS job (Day 1) 13/13

J16_Daily_CreateScdInputDS (Day 1)

This job merges the aggregated sales transactions created in the “J15_Daily_CreateSalesAggDS (Day 1) execution” on page 417 job with the dimension table updates data sets (with nulls in the sales transaction columns) created in the “J13_Daily_UpdateLookupDim execution (Day 1)” on page 382 job. The result is in the format required as input to the SCD stage.

Note: As mentioned earlier, the dimension table updates must be merged with the actual sales transactions to ensure that dimension changes that do not have corresponding sales transactions (also called the late arriving data scenario) are still reflected in the star-schema’s dimension tables. The sales transaction portion of the merged dimension table changes is set to null in order to conform to the SCD stage input requirements and to enable its union with the actual sales transactions via the Funnel stage.

Figure 3-413 on page 423 through Figure 3-423 on page 430 explain the main stages in this job and the configuration of these stages as described in "J16_Daily_CreateScdInputDS (Day 1) configuration" on page 422, while Figure 3-424 on page 431 through Figure 3-430 on page 433 explain the execution of this job with Day 1 input as described in "J16_Daily_CreateScdInputDS (Day 1) execution" on page 430.

J16_Daily_CreateScdInputDS (Day 1) configuration

Figure 3-413 on page 423 shows the various stages in the job — it includes five Data Set stages, three Transformer stages, and one Funnel stage. The names of the stages were modified as shown:

1. Figure 3-414 on page 423 shows the **Columns** tab in the Output page of the Trx_ProductDimLookup link, which defines the column metadata of the Product dimension lookup table.
2. Figure 3-415 on page 424 shows the Trx_ProductDimLookup Transformer stage that maps all the input columns (except the TABLE_CMD column) from the Trx_ProductDimLookup output link, and adds additional columns (with NULLs in them) present in the Ds_AggSales data such as MEMBERSHIP_EXPIRE_DT, MEMBERSHIP_LEVEL, MANAGER_NAME and PRICE_USD. This is required to be able to union data in a Funnel stage, since there must be a one-to-one match of the columns in the sources input to the Funnel stage.
3. Figure 3-416 on page 425 shows the **Columns** tab in the Output page of the Trx_StoreDimLookup link, which defines the column metadata of the Store dimension lookup table.
4. Figure 3-417 on page 426 shows the Trx_StoreDimLookup Transformer stage that maps all the input columns from the Trx_StoreDimLookup output link, and adds additional columns (with NULLs in them) present in the Ds_AggSales data such as DATE, QUANTITY, TOTAL_USD, and CUSTOMER_ID.
5. Figure 3-418 on page 427 through Figure 3-420 on page 428 show the equivalent transformation for the Customer dimension lookup table.
6. Figure 3-421 on page 428 through Figure 3-423 on page 430 show the configuration of the output Data Set stage Ds_SCDinput that contains the results of the union via the Funnel stage.

The results of the execution of this job on Day 1 are described in "J16_Daily_CreateScdInputDS (Day 1) execution" on page 430.

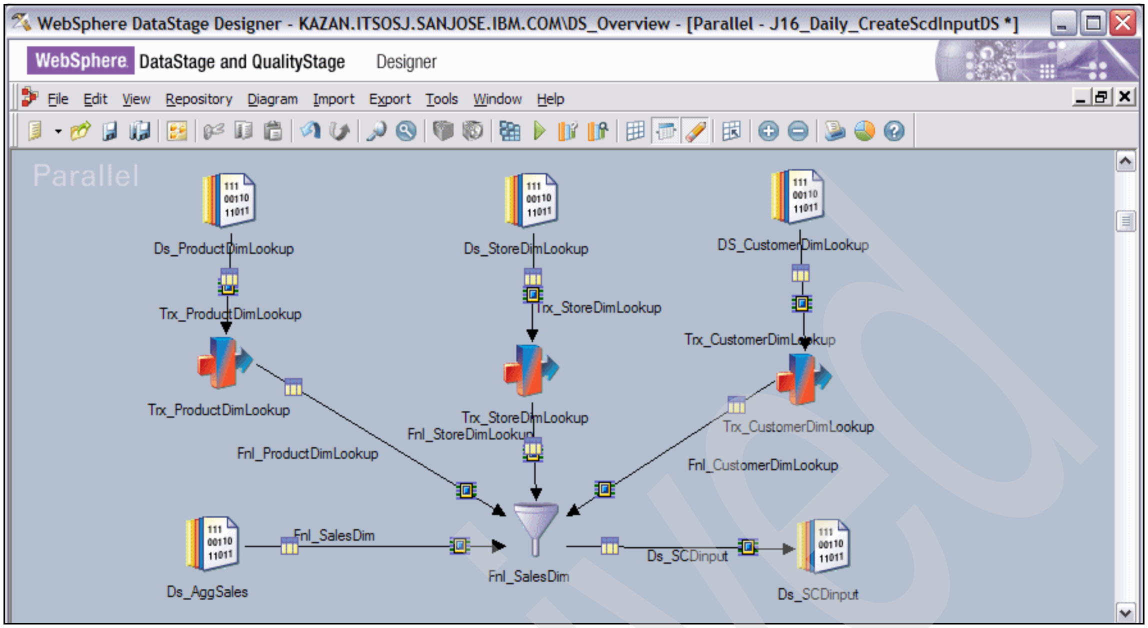


Figure 3-413 Create the J16_Daily_CreateScdInputDS job 1/11

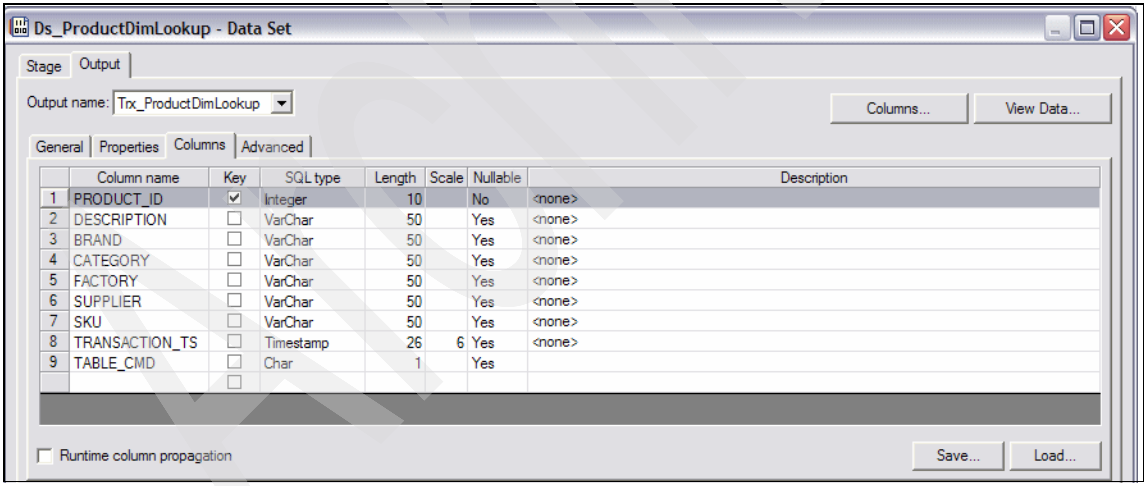


Figure 3-414 Create the J16_Daily_CreateScdInputDS job 2/11

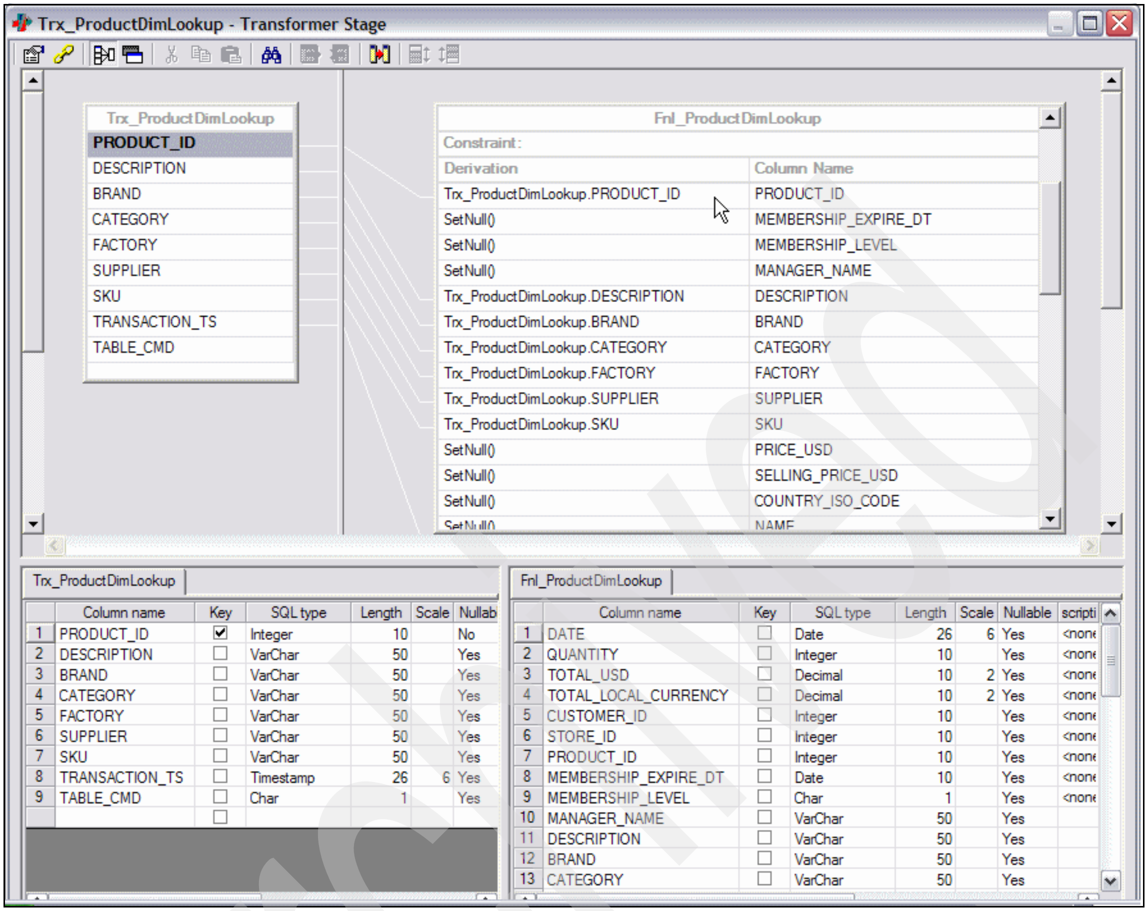


Figure 3-415 Create the J16_Daily_CreateScdInputDS job 3/11

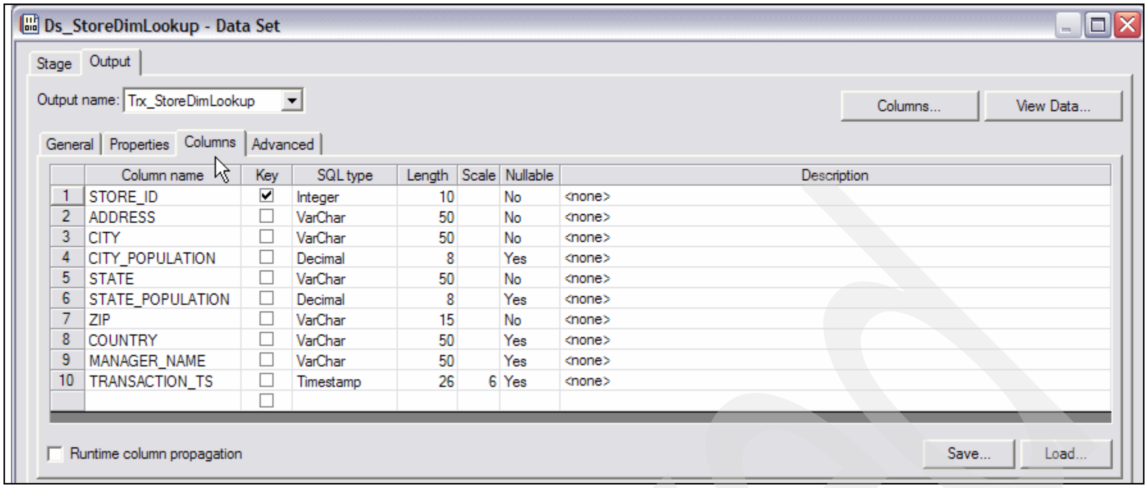


Figure 3-416 Create the J16_Daily_CreateScdInputDS job 4/11

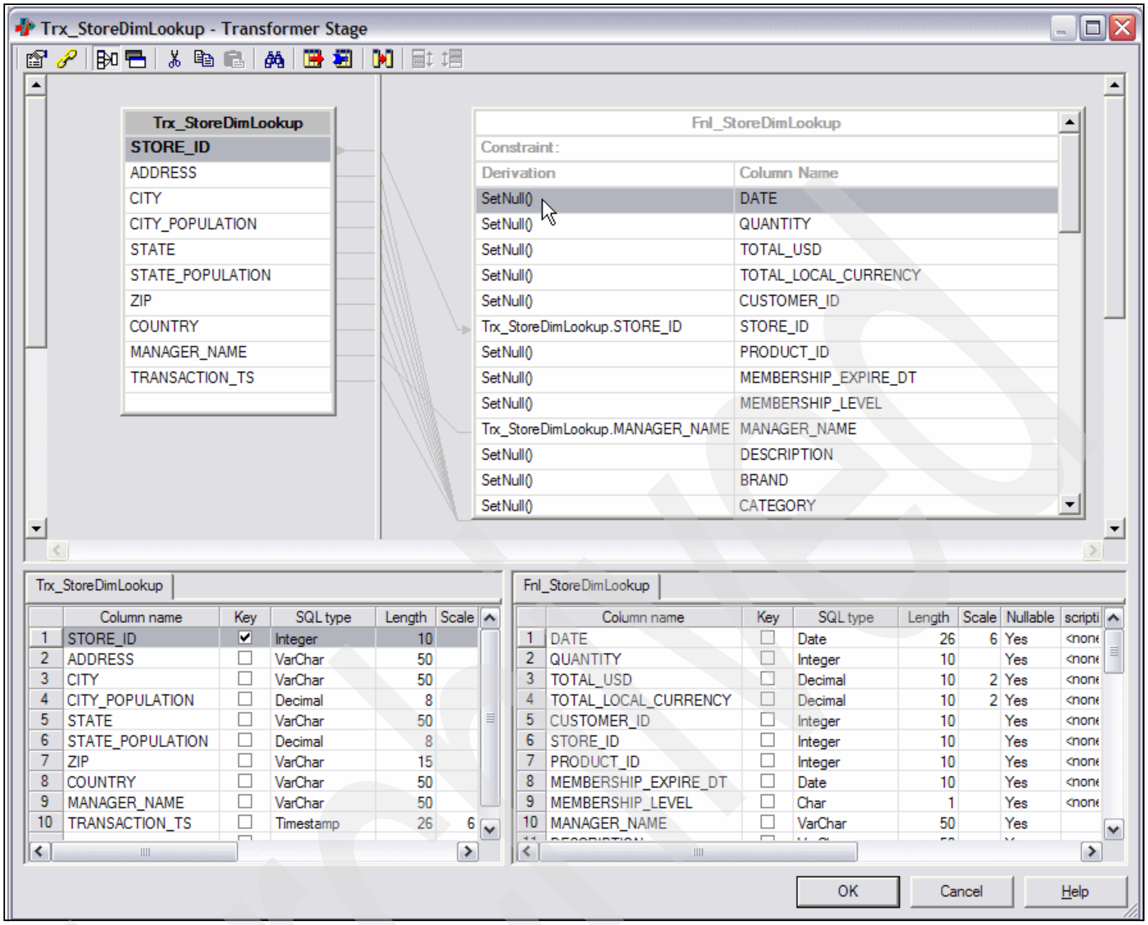
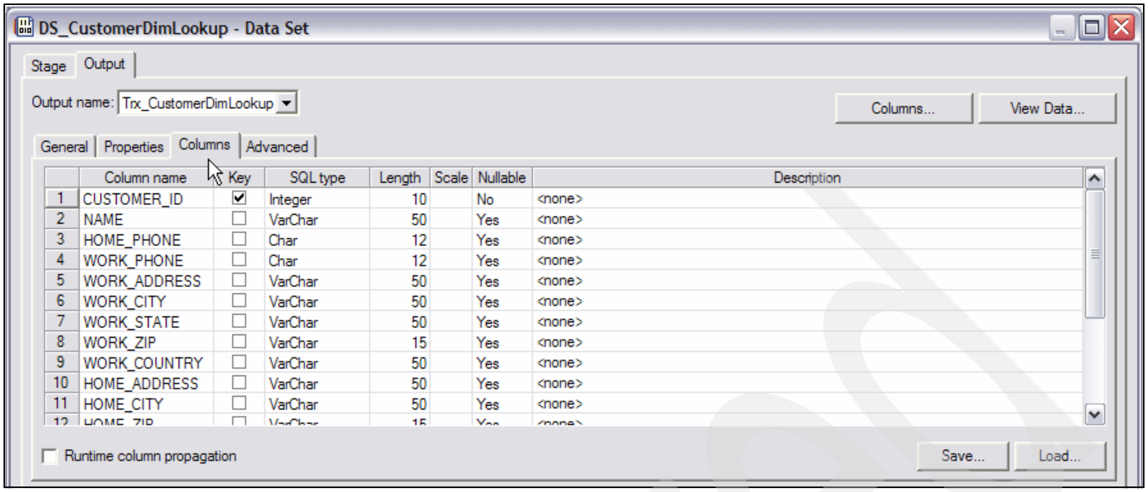
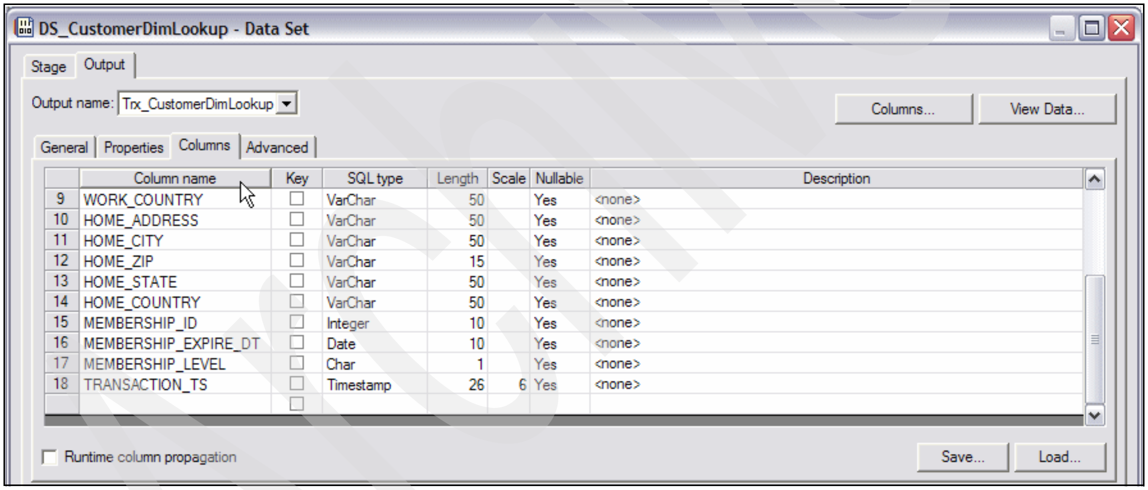


Figure 3-417 Create the J16_Daily_CreateScdInputDS job 5/11



	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	CUSTOMER_ID	☑	Integer	10		No	<none>
2	NAME	☐	VarChar	50		Yes	<none>
3	HOME_PHONE	☐	Char	12		Yes	<none>
4	WORK_PHONE	☐	Char	12		Yes	<none>
5	WORK_ADDRESS	☐	VarChar	50		Yes	<none>
6	WORK_CITY	☐	VarChar	50		Yes	<none>
7	WORK_STATE	☐	VarChar	50		Yes	<none>
8	WORK_ZIP	☐	VarChar	15		Yes	<none>
9	WORK_COUNTRY	☐	VarChar	50		Yes	<none>
10	HOME_ADDRESS	☐	VarChar	50		Yes	<none>
11	HOME_CITY	☐	VarChar	50		Yes	<none>

Figure 3-418 Create the J16_Daily_CreateScdInputDS job 6/11



	Column name	Key	SQL type	Length	Scale	Nullable	Description
9	WORK_COUNTRY	☐	VarChar	50		Yes	<none>
10	HOME_ADDRESS	☐	VarChar	50		Yes	<none>
11	HOME_CITY	☐	VarChar	50		Yes	<none>
12	HOME_ZIP	☐	VarChar	15		Yes	<none>
13	HOME_STATE	☐	VarChar	50		Yes	<none>
14	HOME_COUNTRY	☐	VarChar	50		Yes	<none>
15	MEMBERSHIP_ID	☐	Integer	10		Yes	<none>
16	MEMBERSHIP_EXPIRE_DT	☐	Date	10		Yes	<none>
17	MEMBERSHIP_LEVEL	☐	Char	1		Yes	<none>
18	TRANSACTION_TS	☐	Timestamp	26	6	Yes	<none>

Figure 3-419 Create the J16_Daily_CreateScdInputDS job 7/11

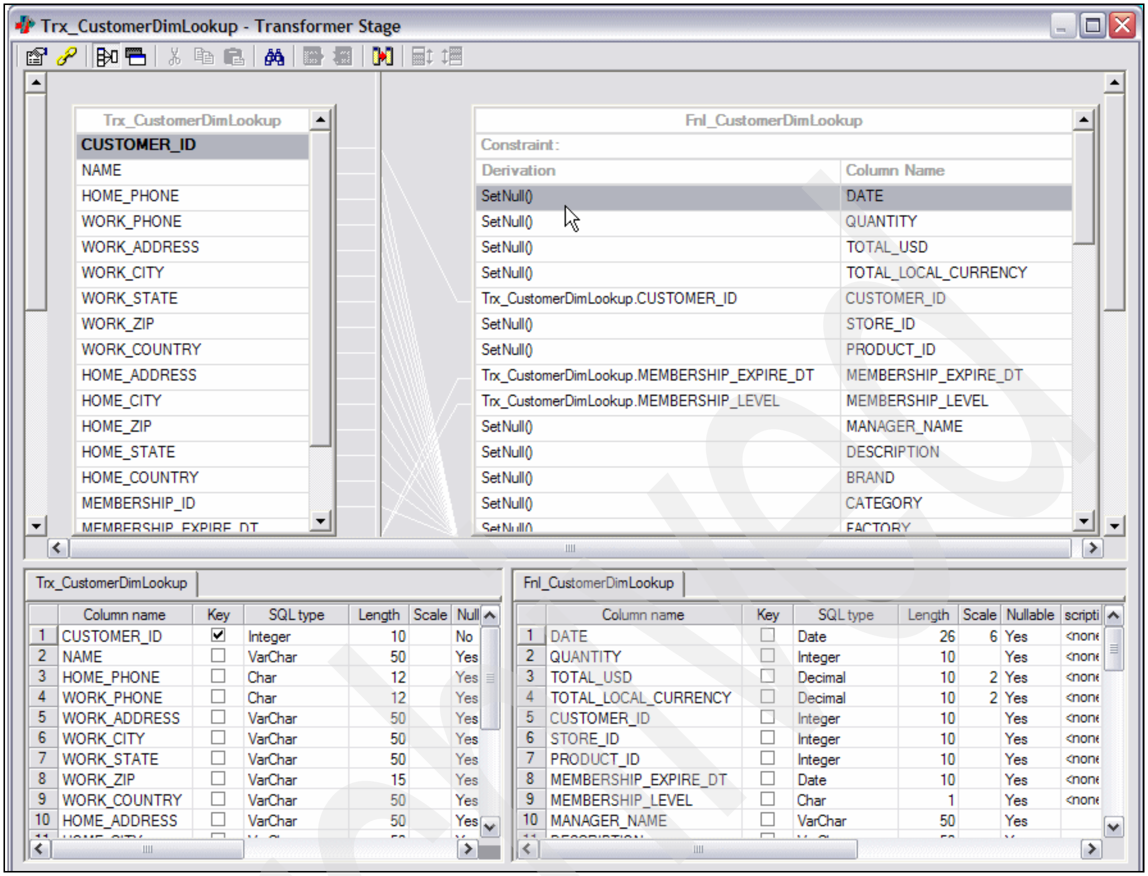


Figure 3-420 Create the J16_Daily_CreateScdInputDS job 8/11

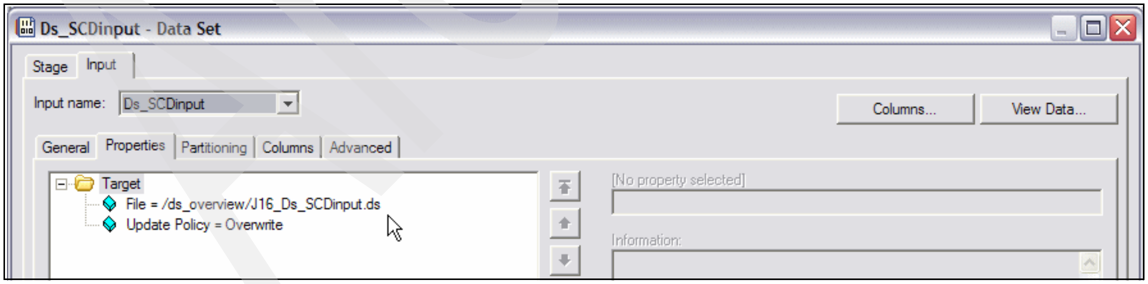
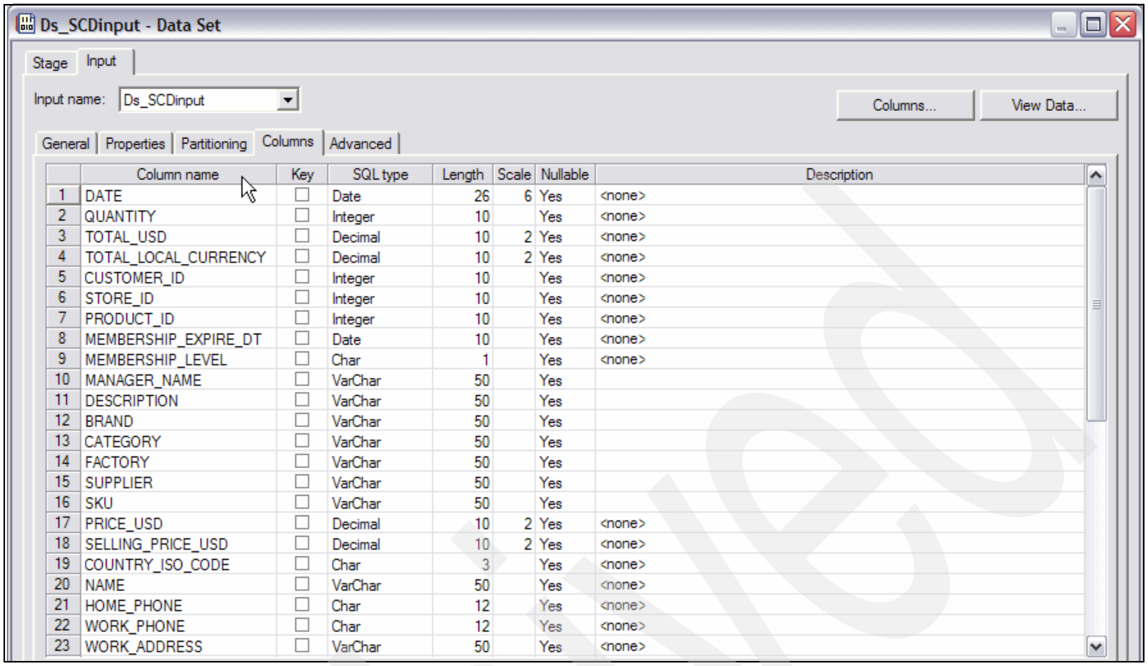
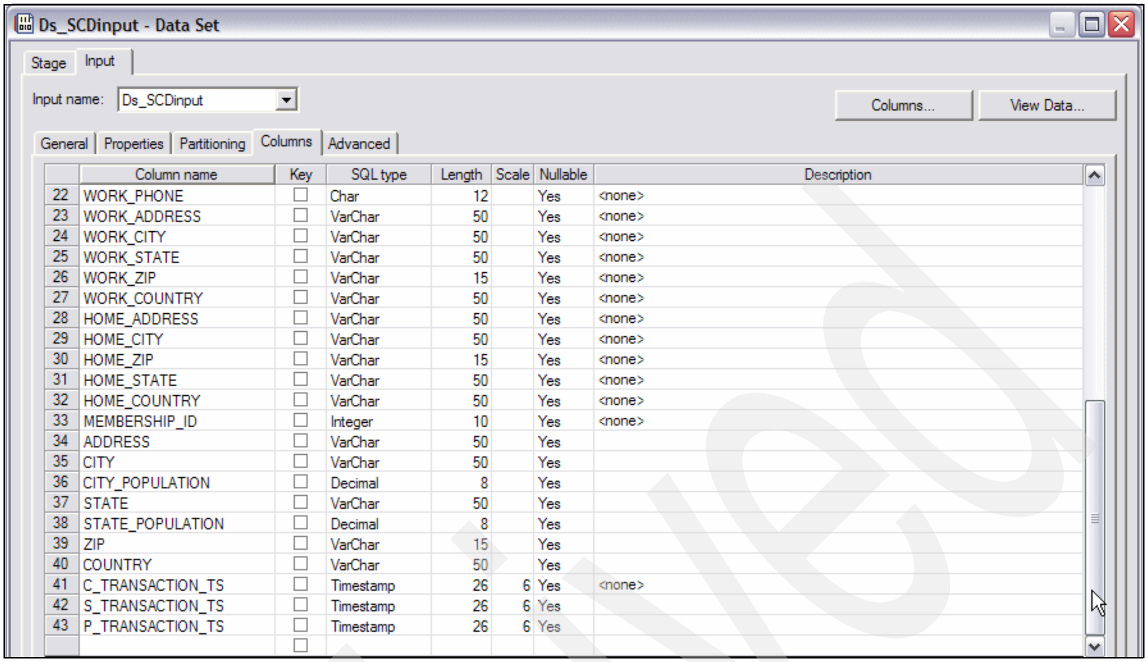


Figure 3-421 Create the J16_Daily_CreateScdInputDS job 9/11



	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	DATE	☐	Date	26	6	Yes	<none>
2	QUANTITY	☐	Integer	10		Yes	<none>
3	TOTAL_USD	☐	Decimal	10	2	Yes	<none>
4	TOTAL_LOCAL_CURRENCY	☐	Decimal	10	2	Yes	<none>
5	CUSTOMER_ID	☐	Integer	10		Yes	<none>
6	STORE_ID	☐	Integer	10		Yes	<none>
7	PRODUCT_ID	☐	Integer	10		Yes	<none>
8	MEMBERSHIP_EXPIRE_DT	☐	Date	10		Yes	<none>
9	MEMBERSHIP_LEVEL	☐	Char	1		Yes	<none>
10	MANAGER_NAME	☐	VarChar	50		Yes	
11	DESCRIPTION	☐	VarChar	50		Yes	
12	BRAND	☐	VarChar	50		Yes	
13	CATEGORY	☐	VarChar	50		Yes	
14	FACTORY	☐	VarChar	50		Yes	
15	SUPPLIER	☐	VarChar	50		Yes	
16	SKU	☐	VarChar	50		Yes	
17	PRICE_USD	☐	Decimal	10	2	Yes	<none>
18	SELLING_PRICE_USD	☐	Decimal	10	2	Yes	<none>
19	COUNTRY_ISO_CODE	☐	Char	3		Yes	<none>
20	NAME	☐	VarChar	50		Yes	<none>
21	HOME_PHONE	☐	Char	12		Yes	<none>
22	WORK_PHONE	☐	Char	12		Yes	<none>
23	WORK_ADDRESS	☐	VarChar	50		Yes	<none>

Figure 3-422 Create the J16_Daily_CreateScdInputDS job 10/11



	Column name	Key	SQL type	Length	Scale	Nullable	Description
22	WORK_PHONE		Char	12		Yes	<none>
23	WORK_ADDRESS		VarChar	50		Yes	<none>
24	WORK_CITY		VarChar	50		Yes	<none>
25	WORK_STATE		VarChar	50		Yes	<none>
26	WORK_ZIP		VarChar	15		Yes	<none>
27	WORK_COUNTRY		VarChar	50		Yes	<none>
28	HOME_ADDRESS		VarChar	50		Yes	<none>
29	HOME_CITY		VarChar	50		Yes	<none>
30	HOME_ZIP		VarChar	15		Yes	<none>
31	HOME_STATE		VarChar	50		Yes	<none>
32	HOME_COUNTRY		VarChar	50		Yes	<none>
33	MEMBERSHIP_ID		Integer	10		Yes	<none>
34	ADDRESS		VarChar	50		Yes	
35	CITY		VarChar	50		Yes	
36	CITY_POPULATION		Decimal	8		Yes	
37	STATE		VarChar	50		Yes	
38	STATE_POPULATION		Decimal	8		Yes	
39	ZIP		VarChar	15		Yes	
40	COUNTRY		VarChar	50		Yes	
41	C_TRANSACTION_TS		Timestamp	26	6	Yes	<none>
42	S_TRANSACTION_TS		Timestamp	26	6	Yes	
43	P_TRANSACTION_TS		Timestamp	26	6	Yes	

Figure 3-423 Create the J16_Daily_CreateScdInputDS job 11/11

J16_Daily_CreateScdInputDS (Day 1) execution

Figure 3-424 on page 431 through Figure 3-430 on page 433 show the results of the execution of this job with Day 1 data described earlier:

- Figure 3-424 on page 431 shows the results of the execution. The inputs to this job are as follows:
  - Accepts 7 rows as input from the “J15_Daily_CreateSalesAggDS (Day 1) execution” on page 417 job as seen in Figure 3-401 on page 418 through Figure 3-406 on page 420.
  - Accepts 2 rows (corresponding to CUSTOMER_ID 1 and 7) as input from the Customer dimension lookup data set generated in “J13_Daily_UpdateLookupDim execution (Day 1)” on page 382.
- The output of this job shows 9 rows corresponding to the union of the two inputs via the Funnel stage. Figure 3-425 on page 431 through Figure 3-430 on page 433 show the nine rows in the output. The NULLs added prior to the Funnel stage are highlighted.

The next step is to execute the job described in “J17_DailyCreateSalesFactDS (Day1)” on page 433.

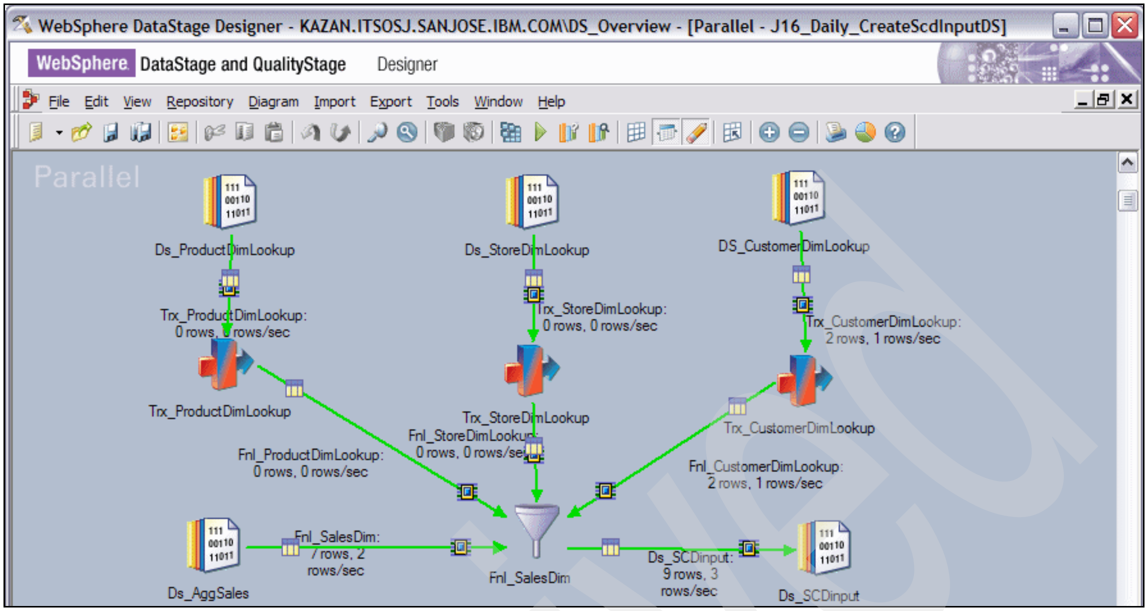
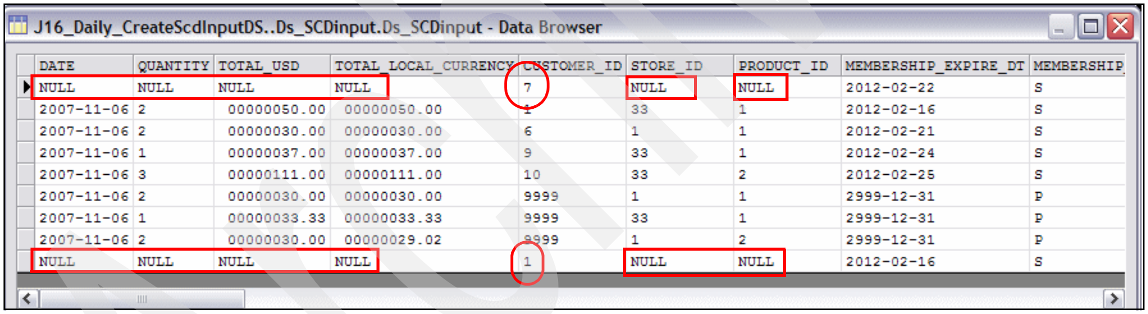


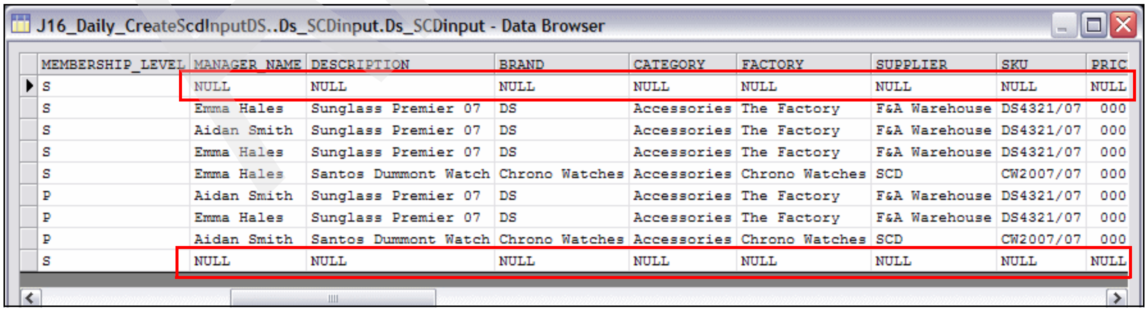
Figure 3-424 Execute the J16_Daily_CreateScdInputDS job (Day 1) 1/7



J16_Daily_CreateScdInputDS..Ds_SCDinput.Ds_SCDinput - Data Browser

DATE	QUANTITY	TOTAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	MEMBERSHIP_EXPIRE_DT	MEMBERSHIP
NULL	NULL	NULL	NULL	7	NULL	NULL	2012-02-22	S
2007-11-06	2	00000050.00	00000050.00	1	33	1	2012-02-16	S
2007-11-06	2	00000030.00	00000030.00	6	1	1	2012-02-21	S
2007-11-06	1	00000037.00	00000037.00	9	33	1	2012-02-24	S
2007-11-06	3	00000111.00	00000111.00	10	33	2	2012-02-25	S
2007-11-06	2	00000030.00	00000030.00	9999	1	1	2999-12-31	P
2007-11-06	1	00000033.33	00000033.33	9999	33	1	2999-12-31	P
2007-11-06	2	00000030.00	00000029.02	9999	1	2	2999-12-31	P
NULL	NULL	NULL	NULL	1	NULL	NULL	2012-02-16	S

Figure 3-425 Execute the J16_Daily_CreateScdInputDS job (Day 1) 2/7



J16_Daily_CreateScdInputDS..Ds_SCDinput.Ds_SCDinput - Data Browser

MEMBERSHIP_LEVEL	MANAGER_NAME	DESCRIPTION	BRAND	CATEGORY	FACTORY	SUPPLIER	SKU	PRIC
S	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
S	Emma Hales	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	000
S	Aidan Smith	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	000
S	Emma Hales	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	000
S	Emma Hales	Santos Dummont Watch	Chrono Watches	Accessories	Chrono Watches	SCD	CW2007/07	000
P	Aidan Smith	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	000
P	Emma Hales	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	000
P	Aidan Smith	Santos Dummont Watch	Chrono Watches	Accessories	Chrono Watches	SCD	CW2007/07	000
S	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 3-426 Execute the J16_Daily_CreateScdInputDS job (Day 1) 3/7

J16_Daily_CreateScdInputDS..Ds_SCDinput.Ds_SCDinput - Data Browser

PRICE_USD	SELLING_PRICE_USD	COUNTRY_ISO_CODE	NAME	HOME_PHONE	WORK_PHONE	WORK_ADDRESS	WORK_CITY
NULL	NULL	NULL	Blair Miller	508-555-0881	408-555-8207	2 ALETHA'S MOUNTAIN WAY	Albany
00000035.00	00000025.00	USA	Arch Smith	508-555-0287	408-555-8801	100 AIR ROAD	Santa Cru
00000017.69	00000015.00	USA	Bela Davis	508-555-0782	408-555-8306	2 ALETHA'S MOUNTAIN WAY	Albany
00000037.00	00000037.00	USA	Blue Moore	508-555-1079	408-555-8009	2 ALETHA'S MOUNTAIN WAY	Albany
00000037.00	00000037.00	USA	Boris Taylor	508-555-1178	408-555-7910	10 BAYLOR WAY	City
00000017.69	00000015.00	USA	CASH CUSTOMER	555-555-5555	555-555-5555		
00000035.00	00000033.33	USA	CASH CUSTOMER	555-555-5555	555-555-5555		
00000017.69	00000015.00	CAD	CASH CUSTOMER	555-555-5555	555-555-5555		
NULL	NULL	NULL	Arch Smith	508-555-0287	408-555-8801	100 AIR ROAD	Santa Cru

Figure 3-427 Execute the J16_Daily_CreateScdInputDS job (Day 1) 4/7

J16_Daily_CreateScdInputDS..Ds_SCDinput.Ds_SCDinput - Data Browser

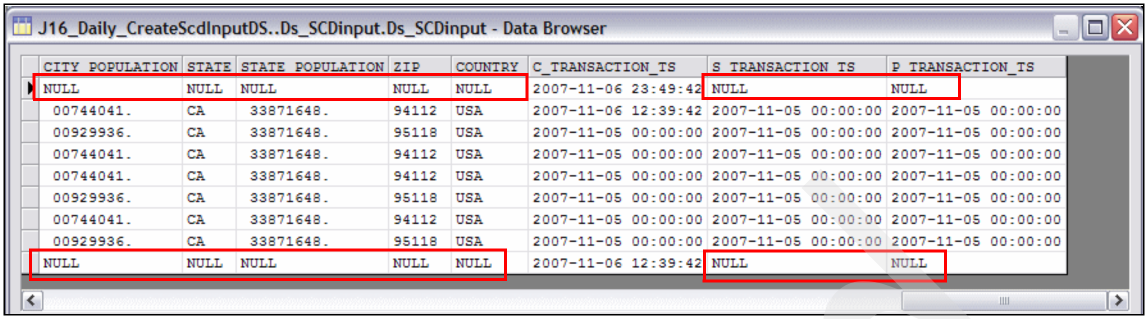
WORK_CITY	WORK_STATE	WORK_ZIP	WORK_COUNTRY	HOME_ADDRESS	HOME_CITY	HOME_ZIP	HOME_STATE	HOME_COUNTRY	MEMBE
Albany	CA	90002	USA	7 ASPEN WAY	Brawley	90023	CA	USA	7
Santa Cruz	CA	90001	USA	2121 Carl St	Santa Cruz	90001	CA	USA	1
Albany	CA	90002	USA	6 ANTON WAY	Bradbury	90006	CA	USA	6
Albany	CA	90002	USA	9 AURIGA WAY	Cathedral City	90009	CA	USA	9
City	CA	90010	USA	2 ALETHA'S MOUNTAIN WAY	Albany	90002	CA	USA	10
									0
									0
									0
Santa Cruz	CA	90001	USA	2121 Carl St	Santa Cruz	90001	CA	USA	1

Figure 3-428 Execute the J16_Daily_CreateScdInputDS job (Day 1) 5/7

J16_Daily_CreateScdInputDS..Ds_SCDinput.Ds_SCDinput - Data Browser

MEMBERSHIP_ID	ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	C_TRANSACTI
7	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2007-11-06
1	8976 Brazil Ave	San Francisco	00744041.	CA	33871648.	94112	USA	2007-11-06
6	12345 Almaden Expressway	San Jose	00929936.	CA	33871648.	95118	USA	2007-11-05
9	8976 Brazil Ave	San Francisco	00744041.	CA	33871648.	94112	USA	2007-11-05
10	8976 Brazil Ave	San Francisco	00744041.	CA	33871648.	94112	USA	2007-11-05
0	12345 Almaden Expressway	San Jose	00929936.	CA	33871648.	95118	USA	2007-11-05
0	8976 Brazil Ave	San Francisco	00744041.	CA	33871648.	94112	USA	2007-11-05
0	12345 Almaden Expressway	San Jose	00929936.	CA	33871648.	95118	USA	2007-11-05
1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2007-11-06

Figure 3-429 Execute the J16_Daily_CreateScdInputDS job (Day 1) 6/7


J16_Daily_CreateScdInputDS..Ds_SCDinput.Ds_SCDinput - Data Browser

CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	C_TRANSACTION_TS	S_TRANSACTION_TS	P_TRANSACTION_TS
NULL	NULL	NULL	NULL	NULL	2007-11-06 23:49:42	NULL	NULL
00744041.	CA	33871648.	94112	USA	2007-11-06 12:39:42	2007-11-05 00:00:00	2007-11-05 00:00:00
00929936.	CA	33871648.	95118	USA	2007-11-05 00:00:00	2007-11-05 00:00:00	2007-11-05 00:00:00
00744041.	CA	33871648.	94112	USA	2007-11-05 00:00:00	2007-11-05 00:00:00	2007-11-05 00:00:00
00744041.	CA	33871648.	94112	USA	2007-11-05 00:00:00	2007-11-05 00:00:00	2007-11-05 00:00:00
00929936.	CA	33871648.	95118	USA	2007-11-05 00:00:00	2007-11-05 00:00:00	2007-11-05 00:00:00
00744041.	CA	33871648.	94112	USA	2007-11-05 00:00:00	2007-11-05 00:00:00	2007-11-05 00:00:00
00929936.	CA	33871648.	95118	USA	2007-11-05 00:00:00	2007-11-05 00:00:00	2007-11-05 00:00:00
NULL	NULL	NULL	NULL	NULL	2007-11-06 12:39:42	NULL	NULL

Figure 3-430 Execute the J16_Daily_CreateScdInputDS job (Day 1) 7/7

J17_DailyCreateSalesFactDS (Day1)

This job creates the files to update dimension tables and the sales fact table in the star-schema using the SCD stage. Late arriving data is identified and written to a reject file. This single job includes updates to all four dimensions (Store, Customer, Product, and Date) and creates separate files for each dimension containing updates to that dimension. It also creates a file that contains updates to the Sales fact table. Late arriving data (updates to dimension tables without corresponding sales transactions) are identified by the fact that sales transaction information (such as QUANTITY and TOTAL_USD are NULL) and written to a reject file.

The actual updates to the four dimension tables and the fact table are done in different jobs — “J18_Daily_UpdateStoreDim (Day 1)” on page 478, “J19_Daily_UpdateCustomerDim (Day 1)” on page 485, “J20_Daily_UpdateProductDim (Day 1)” on page 494, “J21_Daily_UpdateDateDim (Day 1)” on page 499, and “J22_Daily_UpdateSalesFact (Day 1)” on page 502. We deliberately chose to create separate jobs for updating the dimension tables and fact table in order to minimize the use of database facilities. The database connection (that reads the reference database) is active only when it is being accessed and the lookup table in memory is being created.

Figure 3-431 on page 437 through Figure 3-495 on page 473 explain the main stages in this job and the configuration of these stages as described in “J17_DailyCreateSalesFactDS (Day1) configuration” on page 434, while Figure 3-499 on page 476 through Figure 3-506 on page 477 explain the execution of this job with Day 1 input as described in “J17_DailyCreateSalesFactDS (Day1) execution” on page 475.

J17_DailyCreateSalesFactDS (Day1) configuration

Figure 3-431 on page 437 shows the various stages in the job — it includes seven Data Set stages, five Transformer stages, four ODBCConnectorPX stages, four Funnel stages, and four PxSCD stages. The names of the stages were modified as shown:

1. Figure 3-432 on page 438 through Figure 3-434 on page 439 describe the configuration of the Trx_Store Transformer stage that processes the sales transactions data set created in the “J16_Daily_CreateScdInputDS (Day 1) execution” on page 430 job and directs appropriate rows to the Scd_StoreDim PxSCD stage:
   - Figure 3-432 on page 438 shows the Trx_Store Transformer stage with a constraint that directs rows to the Scd_StoreDim or Fnl_StoreIDnull links. All the columns are mapped to each output link — this is not shown here explicitly.
   - Figure 3-433 on page 438 shows the Trx_Store Transformer Stage Constraints window that defines the constraint that directs the rows to the appropriate output link. Briefly, the constraint specifies that records that have the STORE_ID column not null and not equal to zero should be directed to the Scd_StoreDim output link[9], and those records that evaluate the predicate to false are directed to the Fnl_StoreIDnull link.
   - Figure 3-434 on page 439 shows the **Link Ordering** tab in the Trx_Store Stage page that identifies the ordering of the output links as shown.
2. Figure 3-435 on page 439 and Figure 3-436 on page 440 show the configuration of the Odbc_StoreDim ODBCConnectorPX stage retrieves the STORE_DIM table which is the reference link:
   - Figure 3-435 on page 439 identifies the Connection details and the Table name (ds.store_dim) accessed using automatically generated SQL.
   - Figure 3-436 on page 440 shows the **Columns** tab for the Odbc_StoreDim link that identifies column metadata of the reference link.
3. Figure 3-437 on page 441 through Figure 3-439 on page 443 show the configuration of the Scd_StoreDim PxSCD stage that references the STORE_DIM dimension table and writes the following outputs:
   - Dimension updates to a data set on the Ds_StoreDimUpdate link.
   - Sales transactions with the surrogate key for the STORE_ID business key to the Fnl_Store output link.

[9] These records correspond to the late arriving data scenario where there are no sales transactions corresponding to dimension attribute changes that have occurred on that date.

In the event of a Type 2 attribute change, the PxSCD stage expires the earlier version and creates a new version with a new surrogate key. In the case of Type 1 changes only, the attributes are updated in place and no new version is created.

Figure 3-437 on page 441 shows the **Lookup** tab in the Input page (Odbc_StoreDim) that identifies the STORE_ID column as the key of the reference link. The Purpose identifies the various columns and their purpose codes such as Type 1 (CITY_POPULATION and STATE_POPULATION), Type 2 (MANAGER_NAME), Current Indicator (Type 2) [CURRENT_IND], Effective Date (Type 2) [EFFECTIVE_TS], and Expiration Date (Type 2) [EXPIRATION_TS].

Figure 3-438 on page 442 shows the **Dim_Update** tab[10] in the Output page (Ds_StoreDimUpdate) that maps the columns to the Ds_StoreDimUpdate link. The Derivation column specifies how the columns are derived — in particular, the assignment of “Y” to the CURRENT_IND column (with a Type 2 change) and “N” for the Expire record, and “2099-12-31-00.00.00.000000” to the EXPIRATION_TS column (with a Type 2 change) and S_TRANSACTION_TS column for the Expire record.

Figure 3-439 on page 443 shows the **Output Map**[11] tab in the Output page (Fnl_Store) that maps select incoming Scd_StoreDim link columns (including the surrogate key) to the Fnl_Store link that are required to update the Sales fact table. The columns excluded are columns related to the attributes in the STORE_DIM table such as MANAGER_NAME, ADDRESS, CITY, CITY_POPULATION, STATE, STATE_POPULATION, ZIP and COUNTRY since they are not part of the Sales fact table update.

4. Figure 3-440 on page 443 shows the column metadata of the input link Ds_StoreUpdateDim of the Data Set stage Ds_StoreUpdateDim.
5. The Funnel stage Fnl_Store merges the records from late arriving data (Fnl_StoreIDnull link) and the enhanced sales transaction (after the Store dimension table reference) with the surrogate key (Fnl_Store). The Fnl_StoreIDnull link has 43 columns in its metadata as shown in Figure 3-441 on page 444, while the Fnl_Store link has 35 columns in its metadata as shown in Figure 3-442 on page 445. The result of the Funnel stage on the output Trx_Customer link is the 35 columns corresponding to the columns in the input Fnl_Store link as shown in Figure 3-443 on page 446.

[10] This tab is used to create column derivations that specify how to update the dimension table. You must create a derivation for every dimension column. Columns with a purpose code of Type 1 or Type 2 must be derived from a source column. Columns with a purpose code of Current Indicator or Expiration Date must be derived from a literal value, and must also have an Expire derivation.

[11] This tab is used to map data from the input links to the output link. You must create a derivation for every output column.

6. Figure 3-444 on page 447 through Figure 3-456 on page 456 describe the corresponding configurations involving the Customer dimension table reference and update.
7. Figure 3-457 on page 457 through Figure 3-469 on page 463 describe the corresponding configurations involving the Product dimension table reference and update.
8. Figure 3-470 on page 464 through Figure 3-492 on page 472 describe the corresponding configurations involving the Date dimension table reference and update. Figure 3-477 on page 468 through Figure 3-487 on page 469 describe the derivations for the different columns in the Date dimension table.
9. Figure 3-493 on page 472 through Figure 3-498 on page 474 describe the configuration of the Trx_SalesFact Transformer stage, which separates late arriving data onto a separate data set.
   - Figure 3-493 on page 472 shows the Trx_SalesFact Transformer stage with a constraint that directs rows to the Ds_LateArrivingData or Ds_SalesFactUpdate links. All the columns in the input are mapped to the Ds_LateArrivingData output link as shown in Figure 3-497 on page 474, while some columns (C_TRANSACTION_TS, P_TRANSACTION_TS, and S_TRANSACTION_TS) are excluded from the Ds_SalesFactUpdate output link as shown in Figure 3-498 on page 474.
   - Figure 3-494 on page 473 and Figure 3-495 on page 473 show the Trx_SalesFact Transformer Stage Constraints window that defines the constraint that directs the rows to the appropriate output link. Briefly, the constraint specifies that records that have NULLs in the QUANTITY, PRICE_USD, SELLING_PRICE_USD, TOTAL_USD, TOTAL_LOCAL_CURRENCY, or COUNTRY_ISO_CODE columns should be directed to the Ds_LateArrivingData output link, and those records that evaluate the predicate to false are directed to the Ds_SalesFactUpdate link.
   - Figure 3-496 on page 473 shows the **Link Ordering** tab in the Trx_StalesFact Stage page that identifies the ordering of the output links as shown.

The results of the execution of this job on Day 1 are described in "J17_DailyCreateSalesFactDS (Day1) execution" on page 475.

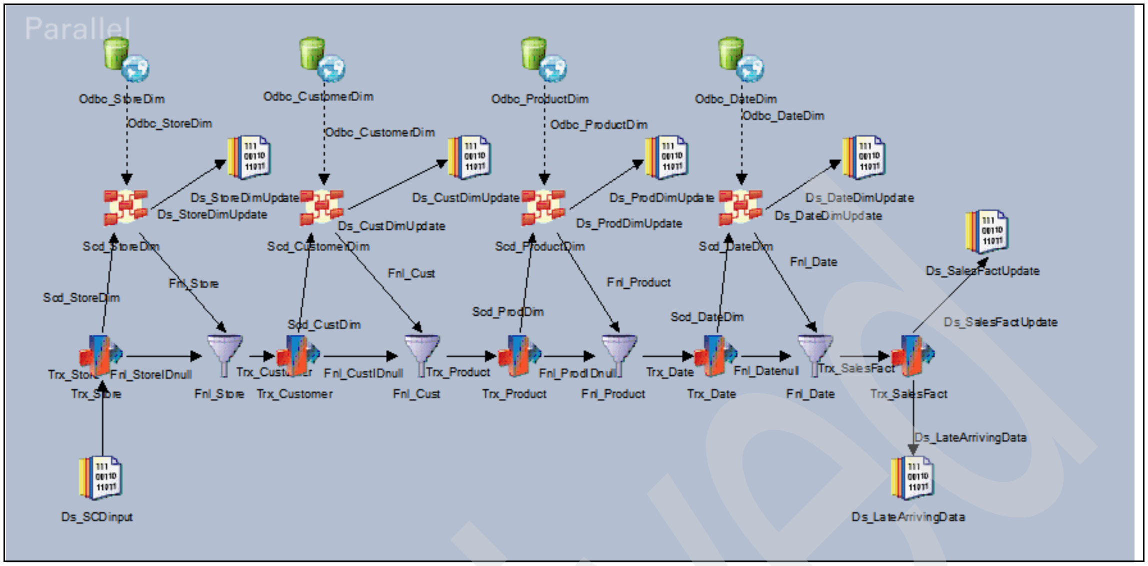


Figure 3-431 Create the J17_DailyCreateSalesFactDS job 1/68

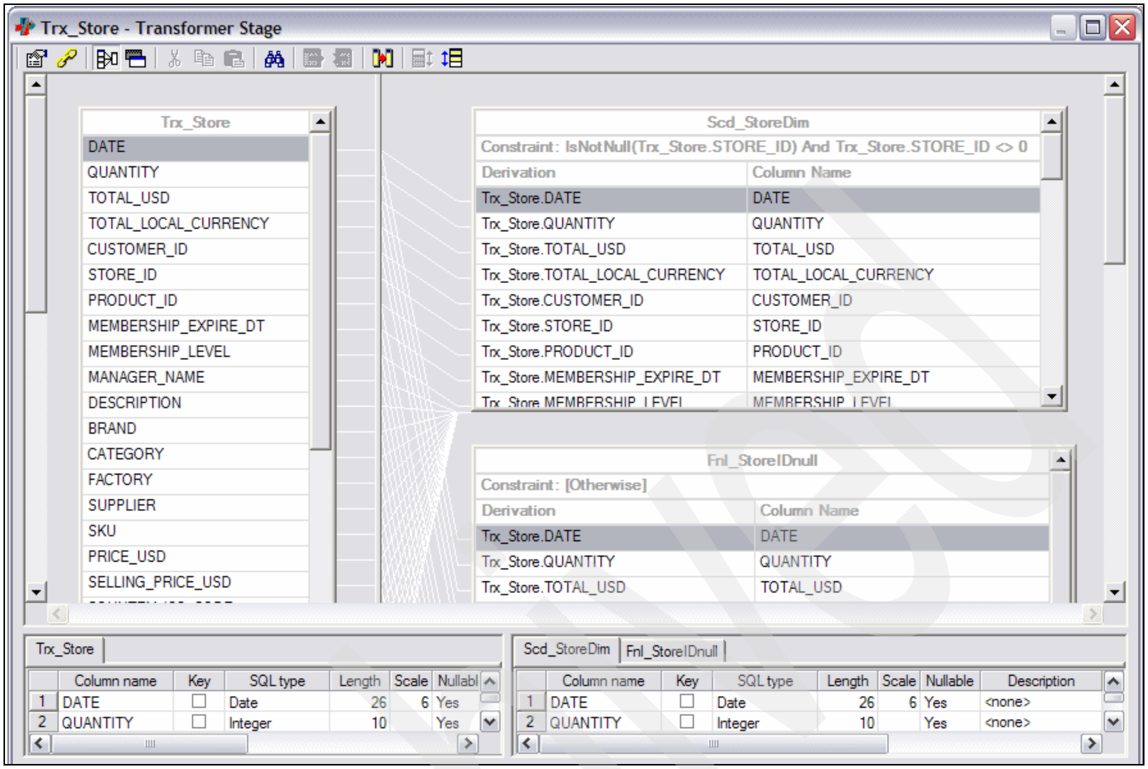


Figure 3-432 Create the J17_DailyCreateSalesFactDS job 2/68

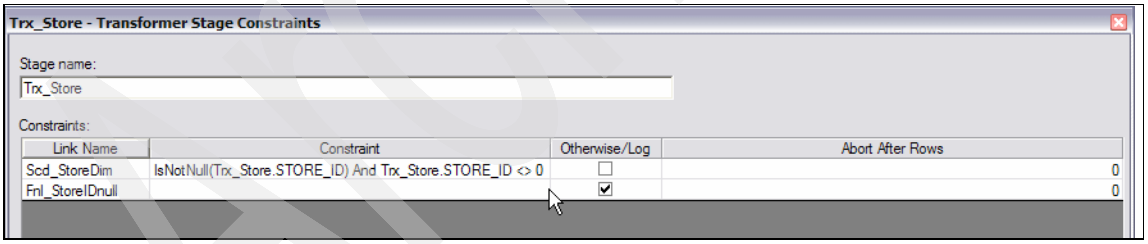


Figure 3-433 Create the J17_DailyCreateSalesFactDS job 3/68

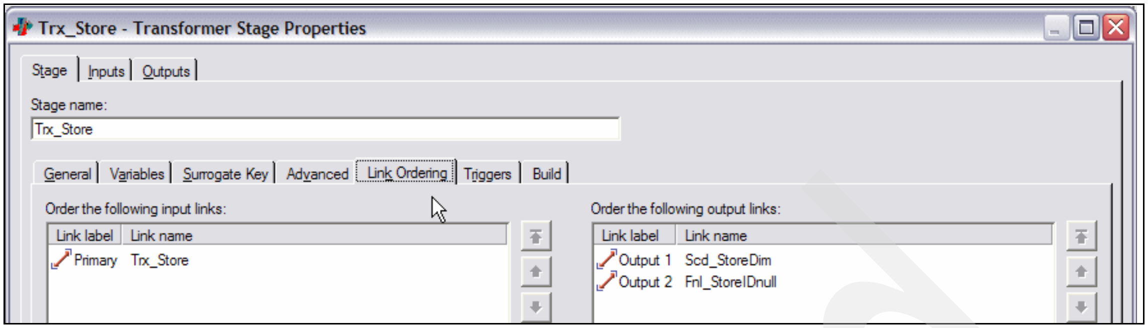


Figure 3-434 Create the J17_DailyCreateSalesFactDS job 4/68

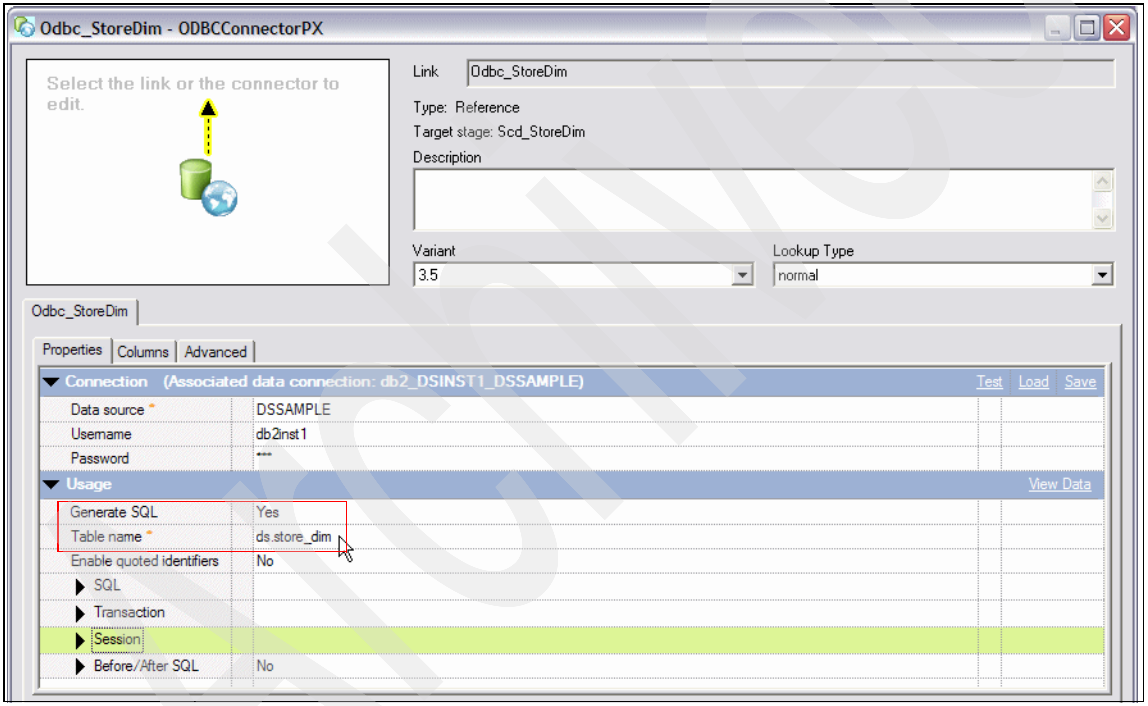


Figure 3-435 Create the J17_DailyCreateSalesFactDS job 5/68

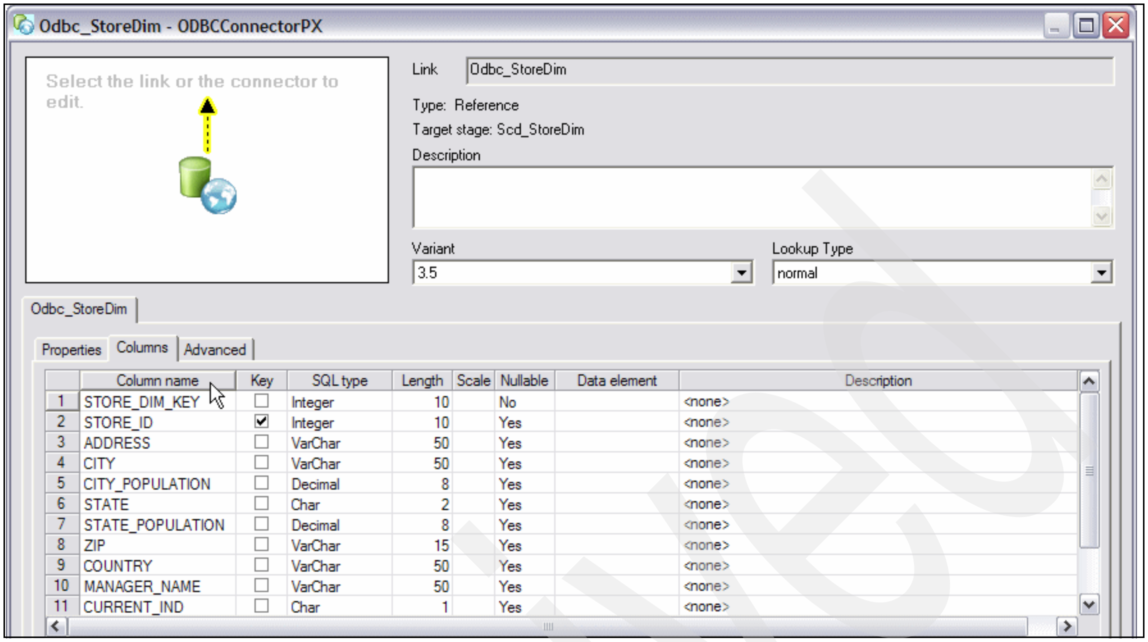


Figure 3-436 Create the J17_DailyCreateSalesFactDS job 6/68

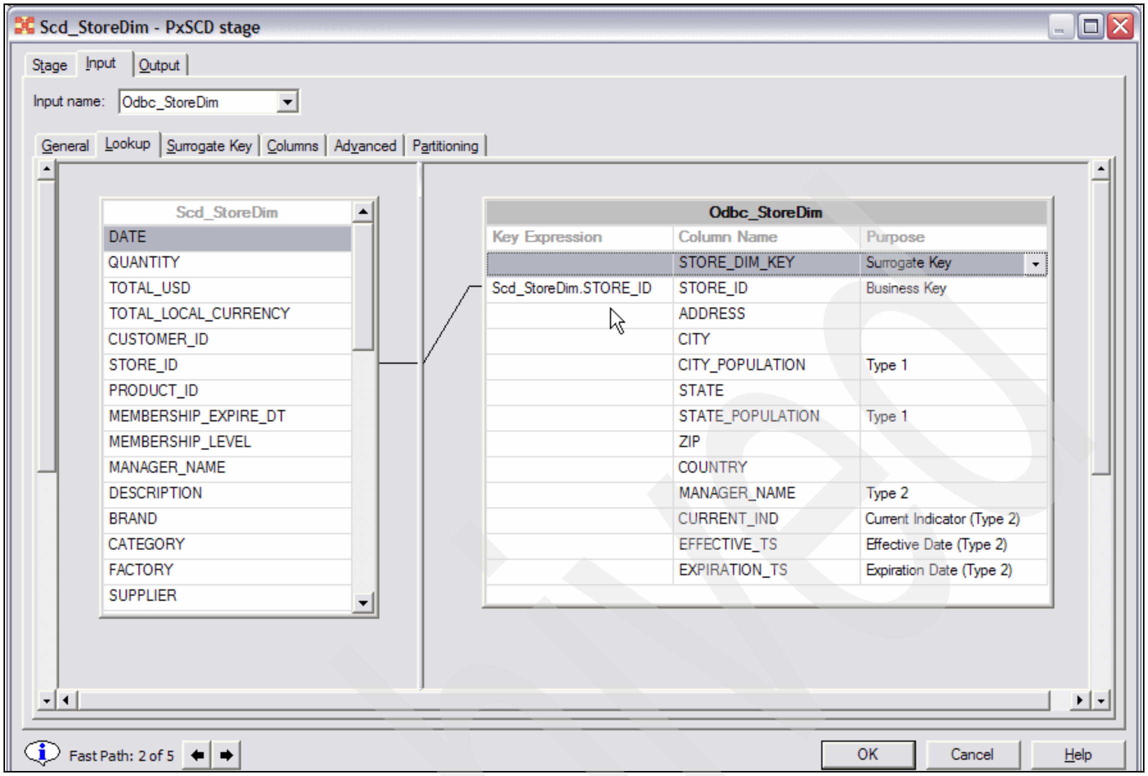


Figure 3-437 Create the J17_DailyCreateSalesFactDS job 7/68

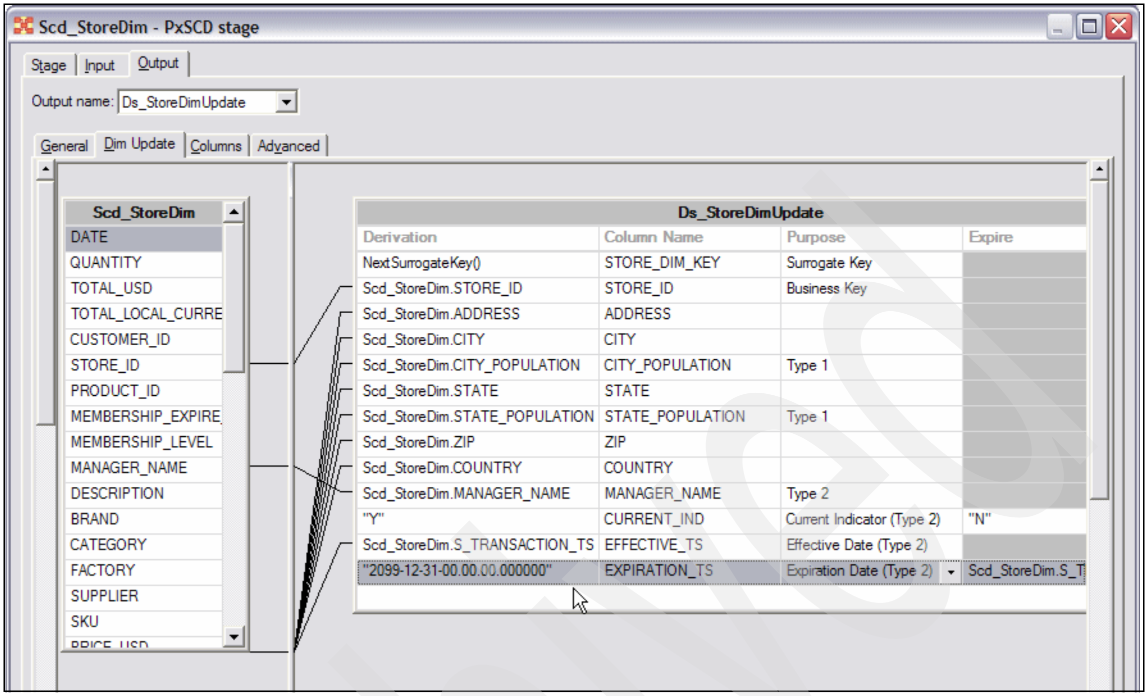


Figure 3-438 Create the J17_DailyCreateSalesFactDS job 8/68

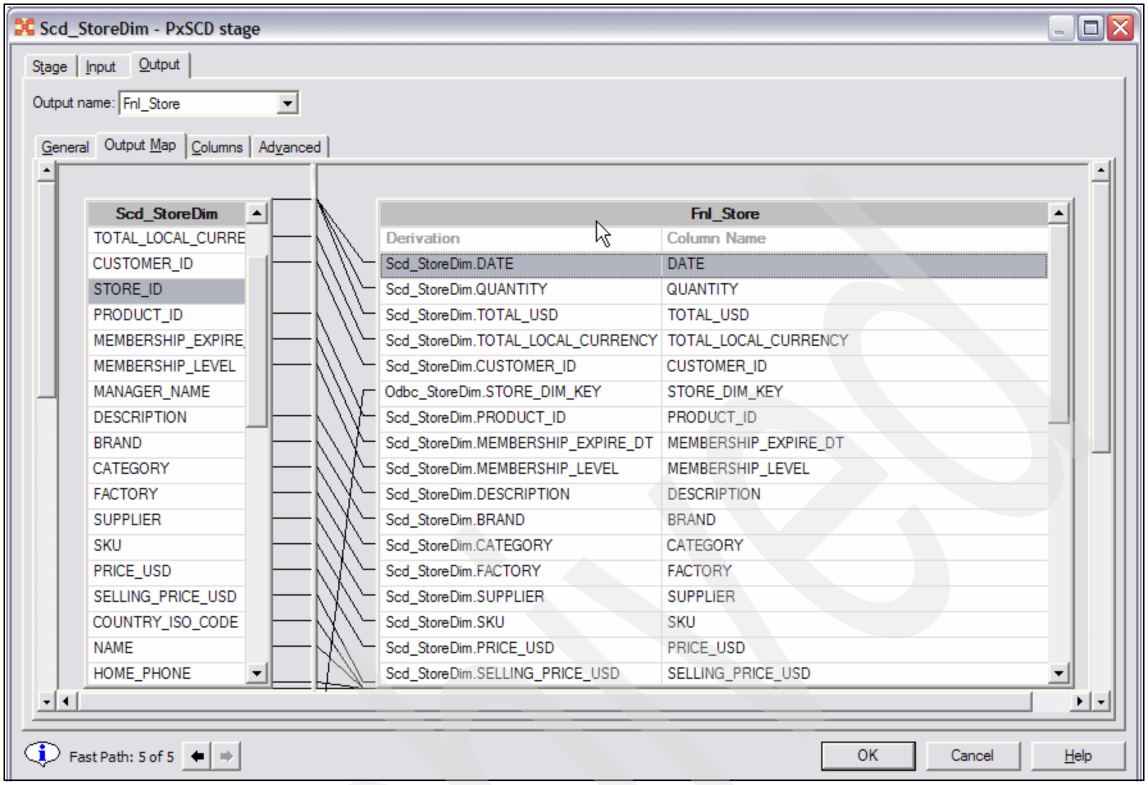


Figure 3-439 Create the J17_DailyCreateSalesFactDS job 9/68

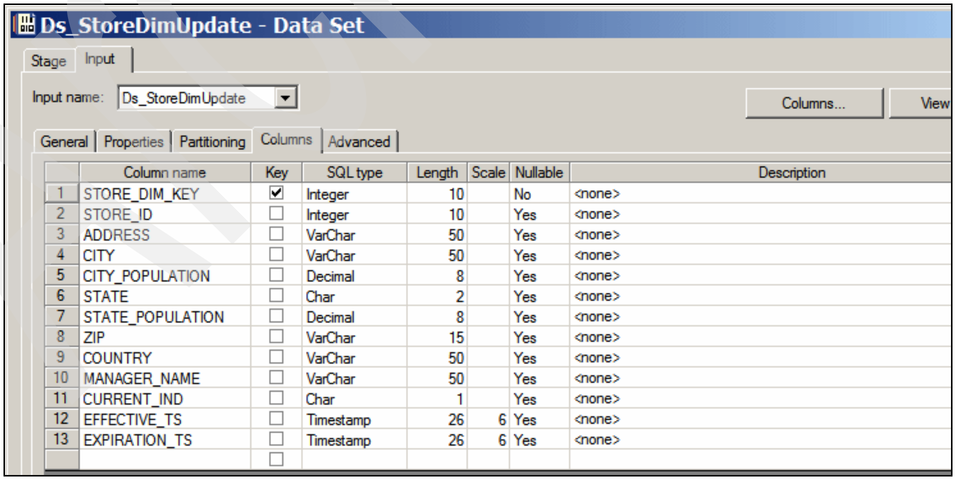
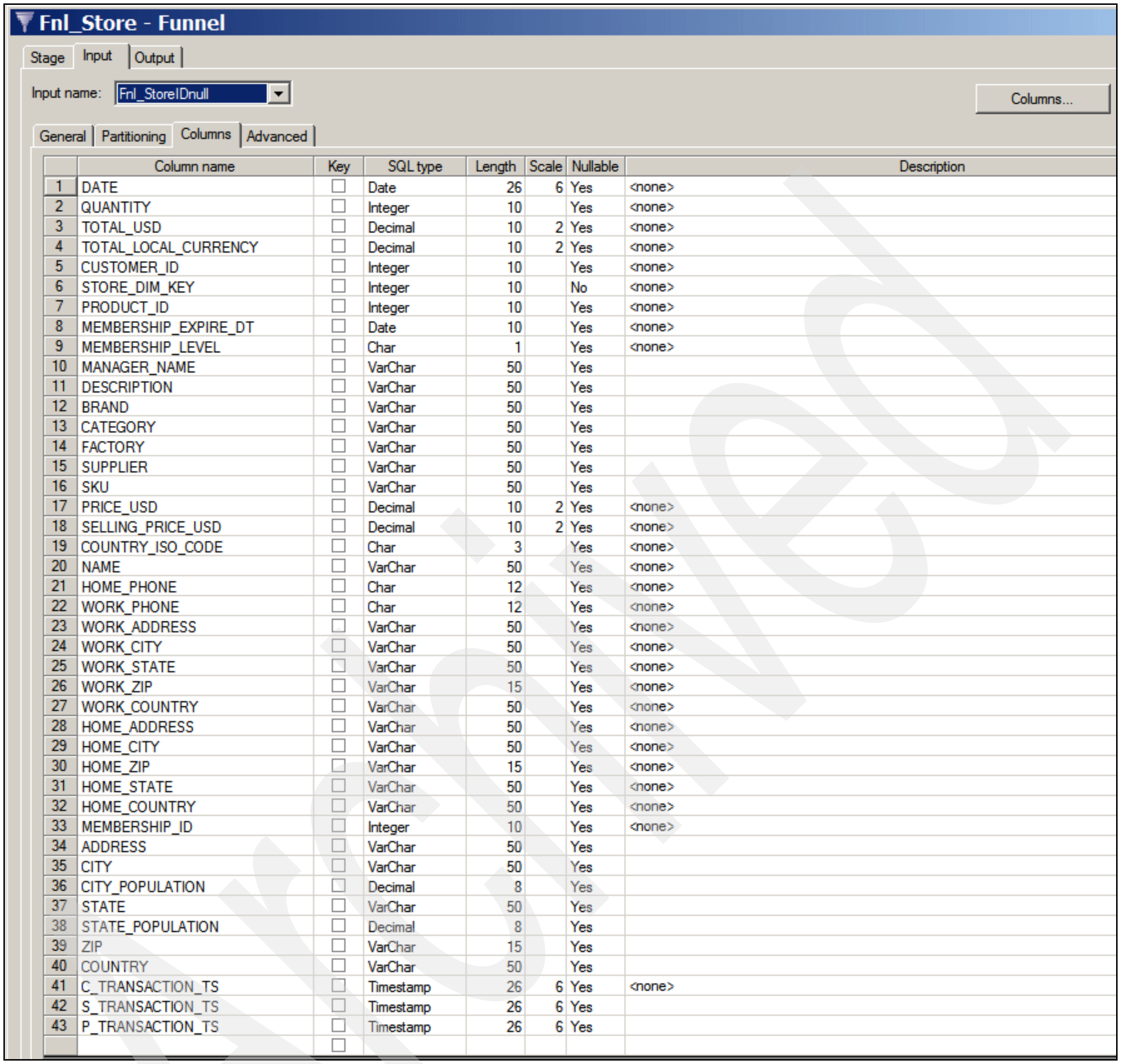


Figure 3-440 Create the J17_DailyCreateSalesFactDS job 10/68



Fnl_Store - Funnel

Stage | Input | Output

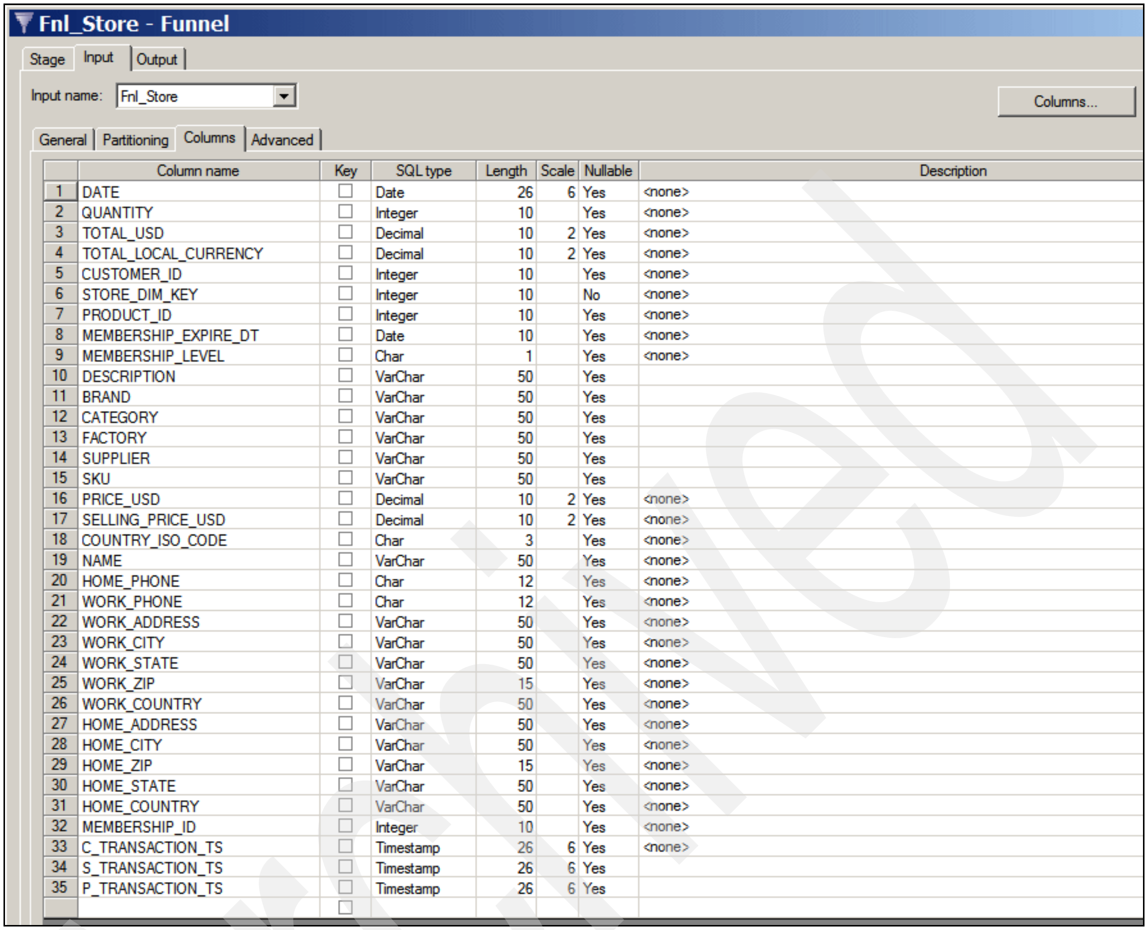
Input name: Fnl_StoreIDnull

Columns...

General | Partitioning | Columns | Advanced

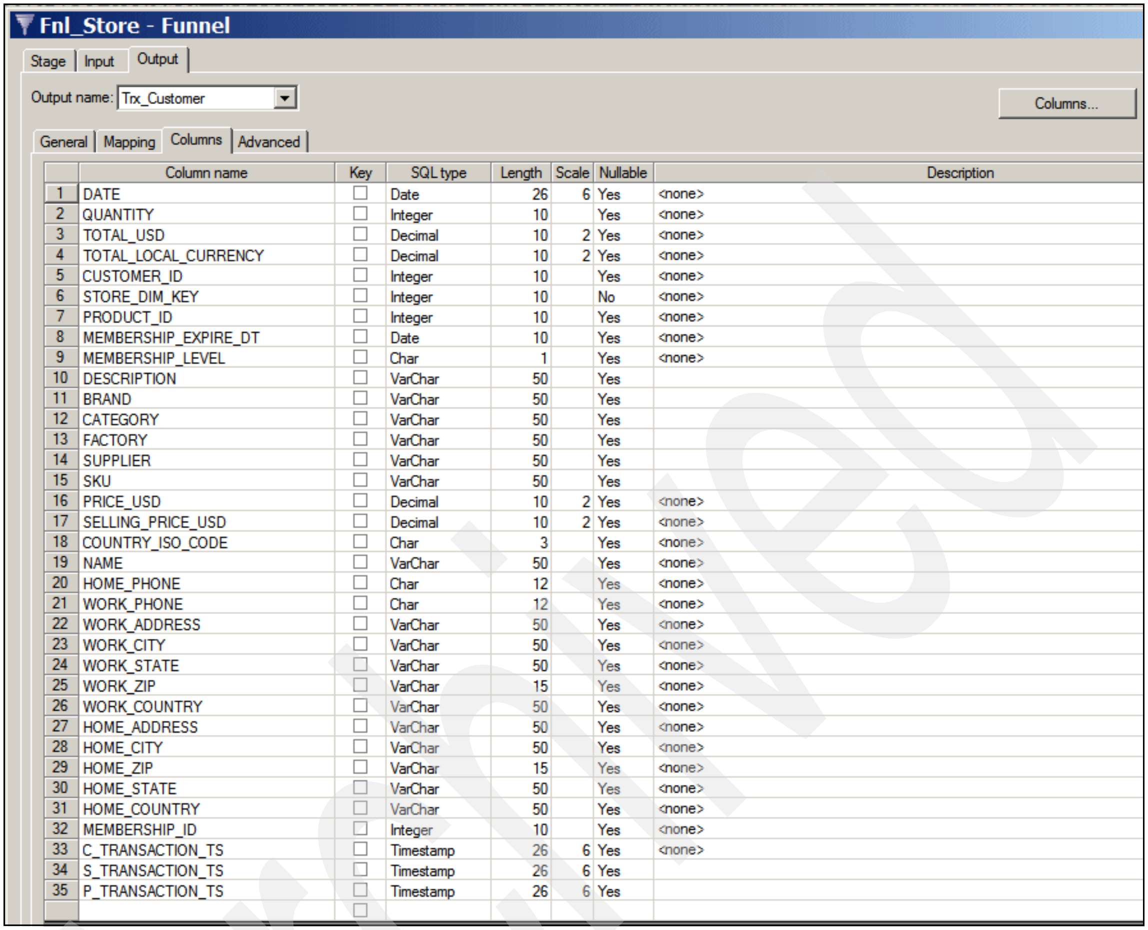
	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	DATE	☐	Date	26	6	Yes	<none>
2	QUANTITY	☐	Integer	10		Yes	<none>
3	TOTAL_USD	☐	Decimal	10	2	Yes	<none>
4	TOTAL_LOCAL_CURRENCY	☐	Decimal	10	2	Yes	<none>
5	CUSTOMER_ID	☐	Integer	10		Yes	<none>
6	STORE_DIM_KEY	☐	Integer	10		No	<none>
7	PRODUCT_ID	☐	Integer	10		Yes	<none>
8	MEMBERSHIP_EXPIRE_DT	☐	Date	10		Yes	<none>
9	MEMBERSHIP_LEVEL	☐	Char	1		Yes	<none>
10	MANAGER_NAME	☐	VarChar	50		Yes	
11	DESCRIPTION	☐	VarChar	50		Yes	
12	BRAND	☐	VarChar	50		Yes	
13	CATEGORY	☐	VarChar	50		Yes	
14	FACTORY	☐	VarChar	50		Yes	
15	SUPPLIER	☐	VarChar	50		Yes	
16	SKU	☐	VarChar	50		Yes	
17	PRICE_USD	☐	Decimal	10	2	Yes	<none>
18	SELLING_PRICE_USD	☐	Decimal	10	2	Yes	<none>
19	COUNTRY_ISO_CODE	☐	Char	3		Yes	<none>
20	NAME	☐	VarChar	50		Yes	<none>
21	HOME_PHONE	☐	Char	12		Yes	<none>
22	WORK_PHONE	☐	Char	12		Yes	<none>
23	WORK_ADDRESS	☐	VarChar	50		Yes	<none>
24	WORK_CITY	☐	VarChar	50		Yes	<none>
25	WORK_STATE	☐	VarChar	50		Yes	<none>
26	WORK_ZIP	☐	VarChar	15		Yes	<none>
27	WORK_COUNTRY	☐	VarChar	50		Yes	<none>
28	HOME_ADDRESS	☐	VarChar	50		Yes	<none>
29	HOME_CITY	☐	VarChar	50		Yes	<none>
30	HOME_ZIP	☐	VarChar	15		Yes	<none>
31	HOME_STATE	☐	VarChar	50		Yes	<none>
32	HOME_COUNTRY	☐	VarChar	50		Yes	<none>
33	MEMBERSHIP_ID	☐	Integer	10		Yes	<none>
34	ADDRESS	☐	VarChar	50		Yes	
35	CITY	☐	VarChar	50		Yes	
36	CITY_POPULATION	☐	Decimal	8		Yes	
37	STATE	☐	VarChar	50		Yes	
38	STATE_POPULATION	☐	Decimal	8		Yes	
39	ZIP	☐	VarChar	15		Yes	
40	COUNTRY	☐	VarChar	50		Yes	
41	C_TRANSACTION_TS	☐	Timestamp	26	6	Yes	<none>
42	S_TRANSACTION_TS	☐	Timestamp	26	6	Yes	
43	P_TRANSACTION_TS	☐	Timestamp	26	6	Yes	

Figure 3-441 Create the J17_DailyCreateSalesFactDS job 11/68



	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	DATE	☐	Date	26	6	Yes	<none>
2	QUANTITY	☐	Integer	10		Yes	<none>
3	TOTAL_USD	☐	Decimal	10	2	Yes	<none>
4	TOTAL_LOCAL_CURRENCY	☐	Decimal	10	2	Yes	<none>
5	CUSTOMER_ID	☐	Integer	10		Yes	<none>
6	STORE_DIM_KEY	☐	Integer	10		No	<none>
7	PRODUCT_ID	☐	Integer	10		Yes	<none>
8	MEMBERSHIP_EXPIRE_DT	☐	Date	10		Yes	<none>
9	MEMBERSHIP_LEVEL	☐	Char	1		Yes	<none>
10	DESCRIPTION	☐	VarChar	50		Yes	
11	BRAND	☐	VarChar	50		Yes	
12	CATEGORY	☐	VarChar	50		Yes	
13	FACTORY	☐	VarChar	50		Yes	
14	SUPPLIER	☐	VarChar	50		Yes	
15	SKU	☐	VarChar	50		Yes	
16	PRICE_USD	☐	Decimal	10	2	Yes	<none>
17	SELLING_PRICE_USD	☐	Decimal	10	2	Yes	<none>
18	COUNTRY_ISO_CODE	☐	Char	3		Yes	<none>
19	NAME	☐	VarChar	50		Yes	<none>
20	HOME_PHONE	☐	Char	12		Yes	<none>
21	WORK_PHONE	☐	Char	12		Yes	<none>
22	WORK_ADDRESS	☐	VarChar	50		Yes	<none>
23	WORK_CITY	☐	VarChar	50		Yes	<none>
24	WORK_STATE	☐	VarChar	50		Yes	<none>
25	WORK_ZIP	☐	VarChar	15		Yes	<none>
26	WORK_COUNTRY	☐	VarChar	50		Yes	<none>
27	HOME_ADDRESS	☐	VarChar	50		Yes	<none>
28	HOME_CITY	☐	VarChar	50		Yes	<none>
29	HOME_ZIP	☐	VarChar	15		Yes	<none>
30	HOME_STATE	☐	VarChar	50		Yes	<none>
31	HOME_COUNTRY	☐	VarChar	50		Yes	<none>
32	MEMBERSHIP_ID	☐	Integer	10		Yes	<none>
33	C_TRANSACTION_TS	☐	Timestamp	26	6	Yes	<none>
34	S_TRANSACTION_TS	☐	Timestamp	26	6	Yes	
35	P_TRANSACTION_TS	☐	Timestamp	26	6	Yes	

Figure 3-442 Create the J17_DailyCreateSalesFactDS job 12/68



	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	DATE	☐	Date	26	6	Yes	<none>
2	QUANTITY	☐	Integer	10		Yes	<none>
3	TOTAL_USD	☐	Decimal	10	2	Yes	<none>
4	TOTAL_LOCAL_CURRENCY	☐	Decimal	10	2	Yes	<none>
5	CUSTOMER_ID	☐	Integer	10		Yes	<none>
6	STORE_DIM_KEY	☐	Integer	10		No	<none>
7	PRODUCT_ID	☐	Integer	10		Yes	<none>
8	MEMBERSHIP_EXPIRE_DT	☐	Date	10		Yes	<none>
9	MEMBERSHIP_LEVEL	☐	Char	1		Yes	<none>
10	DESCRIPTION	☐	VarChar	50		Yes	
11	BRAND	☐	VarChar	50		Yes	
12	CATEGORY	☐	VarChar	50		Yes	
13	FACTORY	☐	VarChar	50		Yes	
14	SUPPLIER	☐	VarChar	50		Yes	
15	SKU	☐	VarChar	50		Yes	
16	PRICE_USD	☐	Decimal	10	2	Yes	<none>
17	SELLING_PRICE_USD	☐	Decimal	10	2	Yes	<none>
18	COUNTRY_ISO_CODE	☐	Char	3		Yes	<none>
19	NAME	☐	VarChar	50		Yes	<none>
20	HOME_PHONE	☐	Char	12		Yes	<none>
21	WORK_PHONE	☐	Char	12		Yes	<none>
22	WORK_ADDRESS	☐	VarChar	50		Yes	<none>
23	WORK_CITY	☐	VarChar	50		Yes	<none>
24	WORK_STATE	☐	VarChar	50		Yes	<none>
25	WORK_ZIP	☐	VarChar	15		Yes	<none>
26	WORK_COUNTRY	☐	VarChar	50		Yes	<none>
27	HOME_ADDRESS	☐	VarChar	50		Yes	<none>
28	HOME_CITY	☐	VarChar	50		Yes	<none>
29	HOME_ZIP	☐	VarChar	15		Yes	<none>
30	HOME_STATE	☐	VarChar	50		Yes	<none>
31	HOME_COUNTRY	☐	VarChar	50		Yes	<none>
32	MEMBERSHIP_ID	☐	Integer	10		Yes	<none>
33	C_TRANSACTION_TS	☐	Timestamp	26	6	Yes	<none>
34	S_TRANSACTION_TS	☐	Timestamp	26	6	Yes	
35	P_TRANSACTION_TS	☐	Timestamp	26	6	Yes	
		☐					

Figure 3-443 Create the J17_DailyCreateSalesFactDS job 13/68

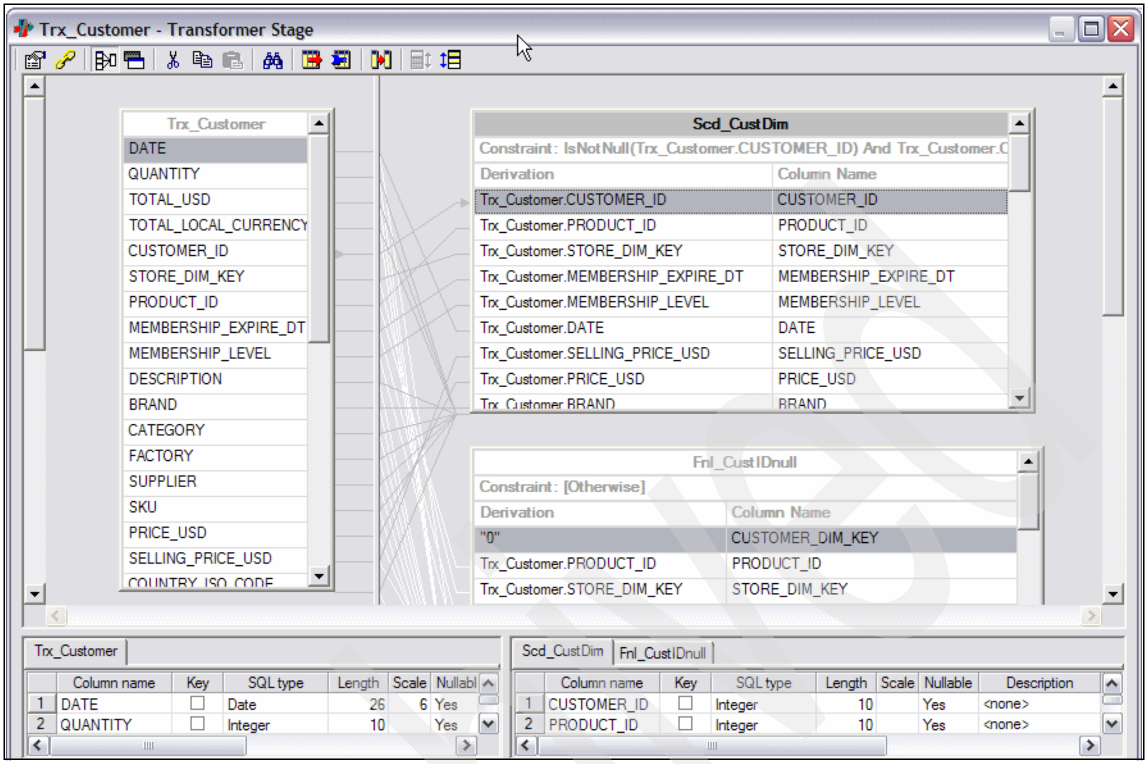


Figure 3-444 Create the J17_DailyCreateSalesFactDS job 14/68

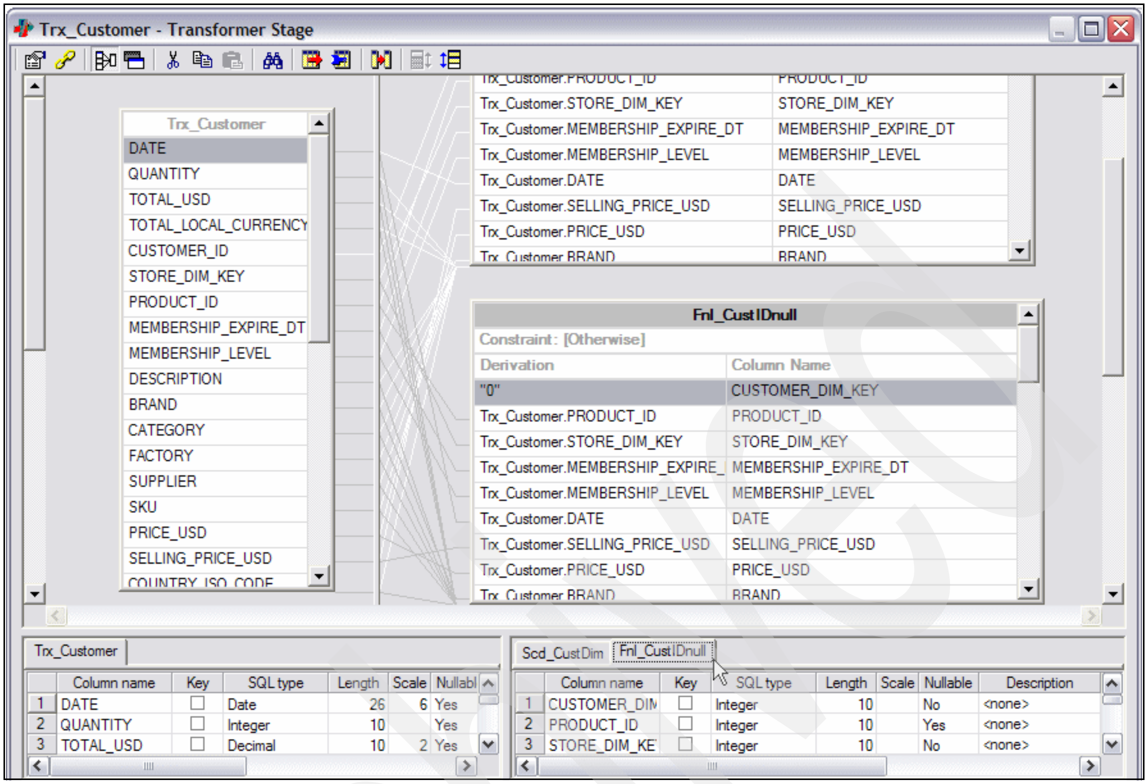


Figure 3-445   Create the J17_DailyCreateSalesFactDS job 15/68

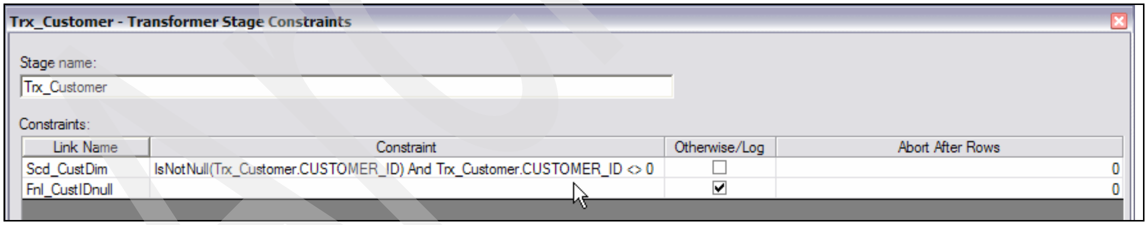


Figure 3-446   Create the J17_DailyCreateSalesFactDS job 16/68

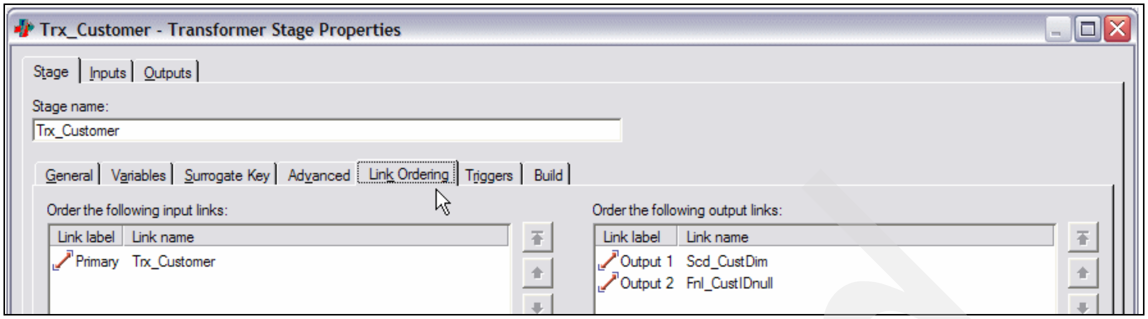


Figure 3-447   Create the J17_DailyCreateSalesFactDS job 17/68

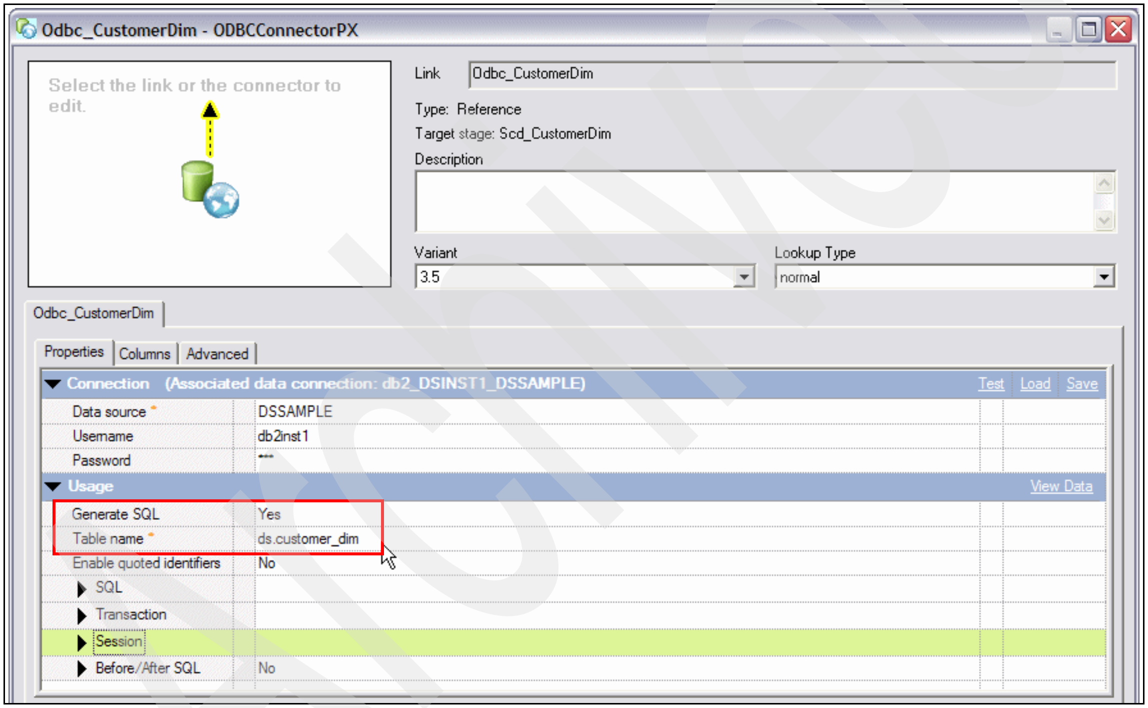


Figure 3-448   Create the J17_DailyCreateSalesFactDS job 18/68

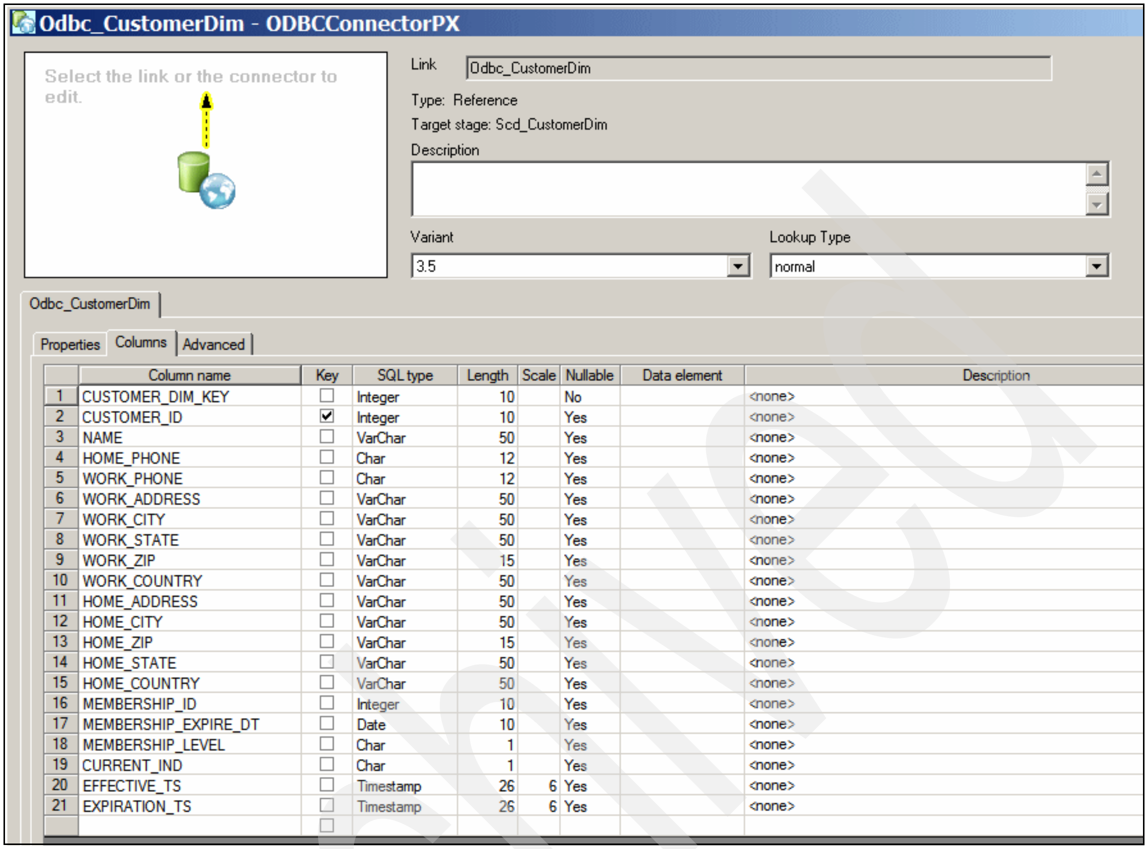


Figure 3-449 Create the J17_DailyCreateSalesFactDS job 19/68

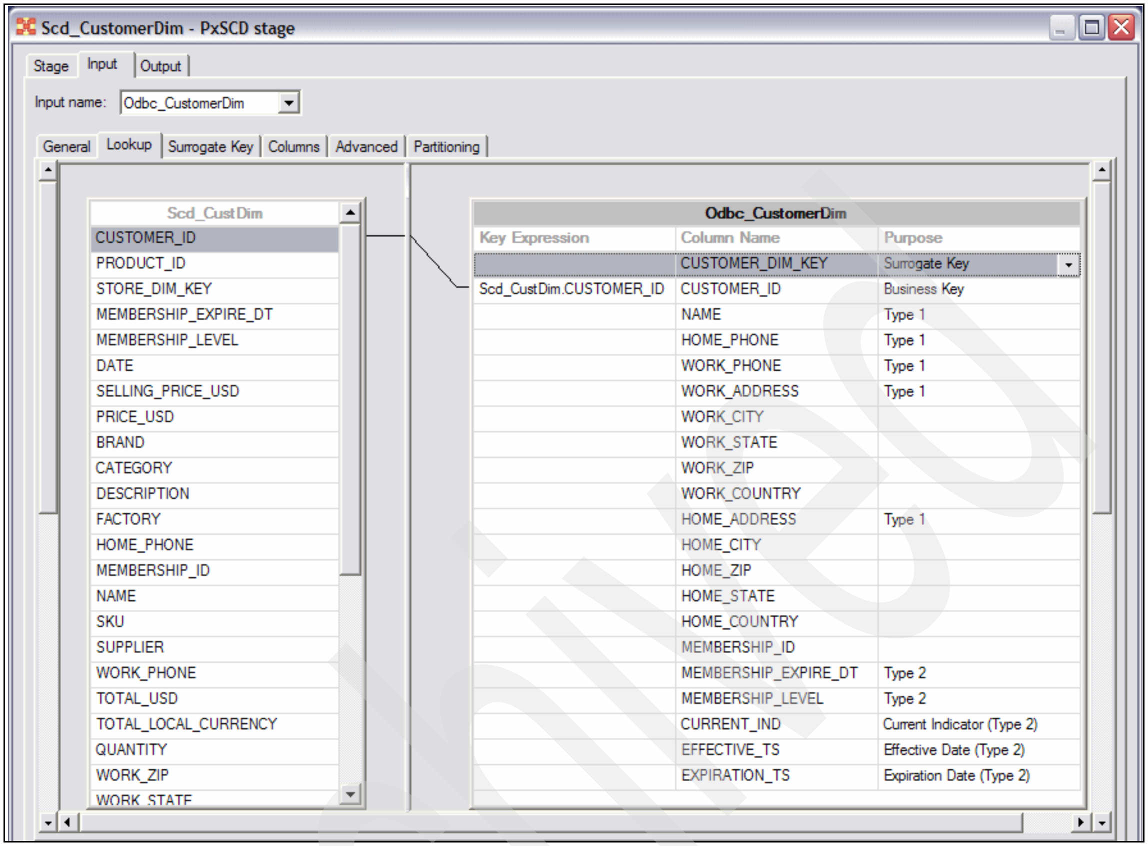


Figure 3-450 Create the J17_DailyCreateSalesFactDS job 20/68

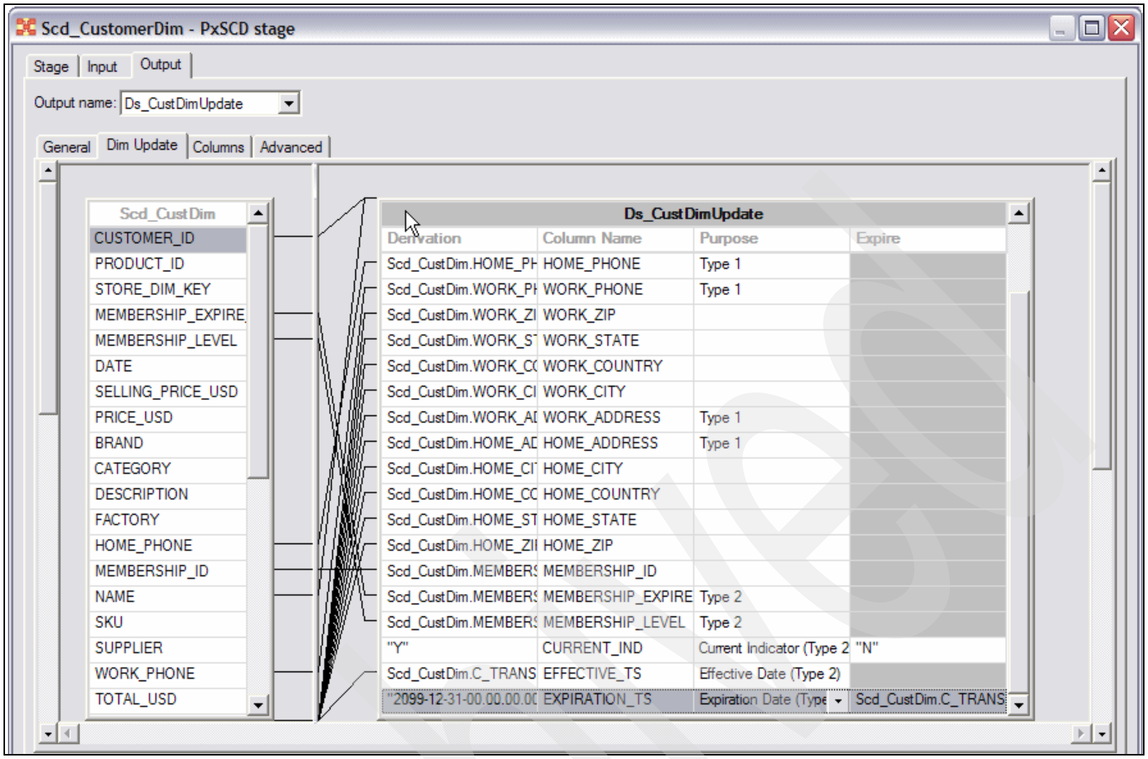


Figure 3-451 Create the J17_DailyCreateSalesFactDS job 21/68

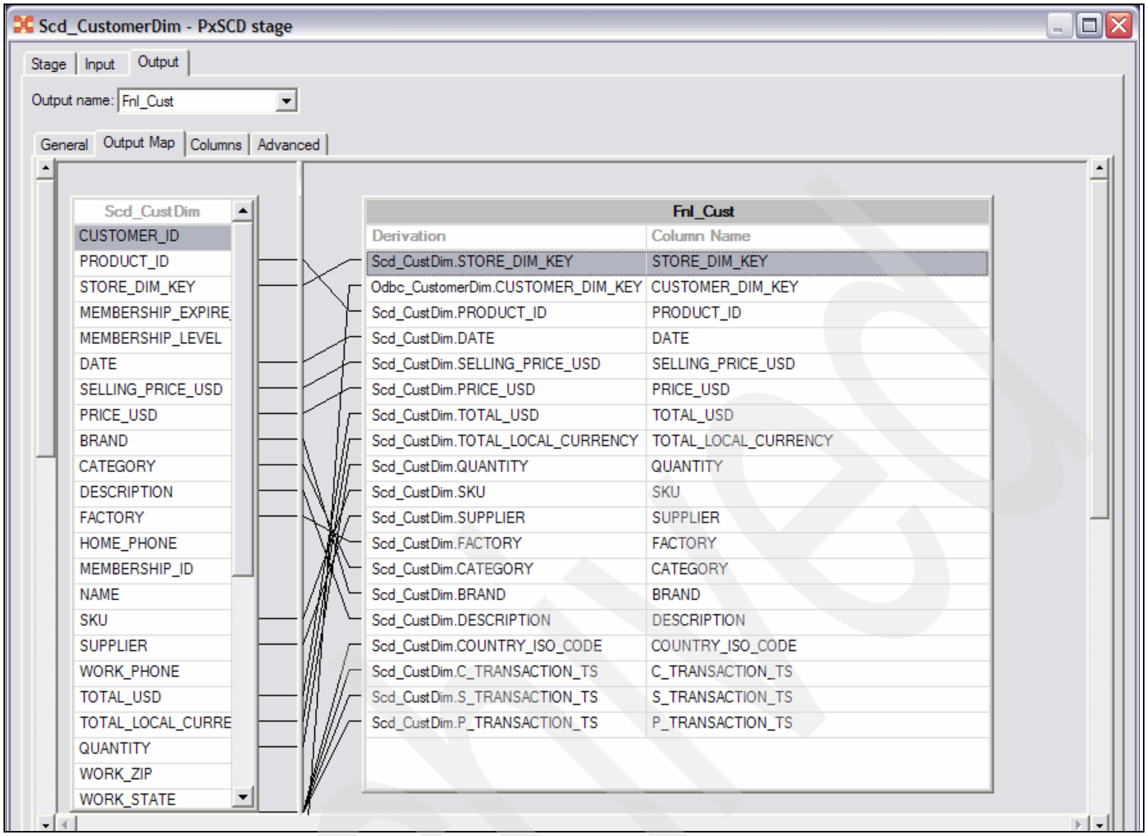
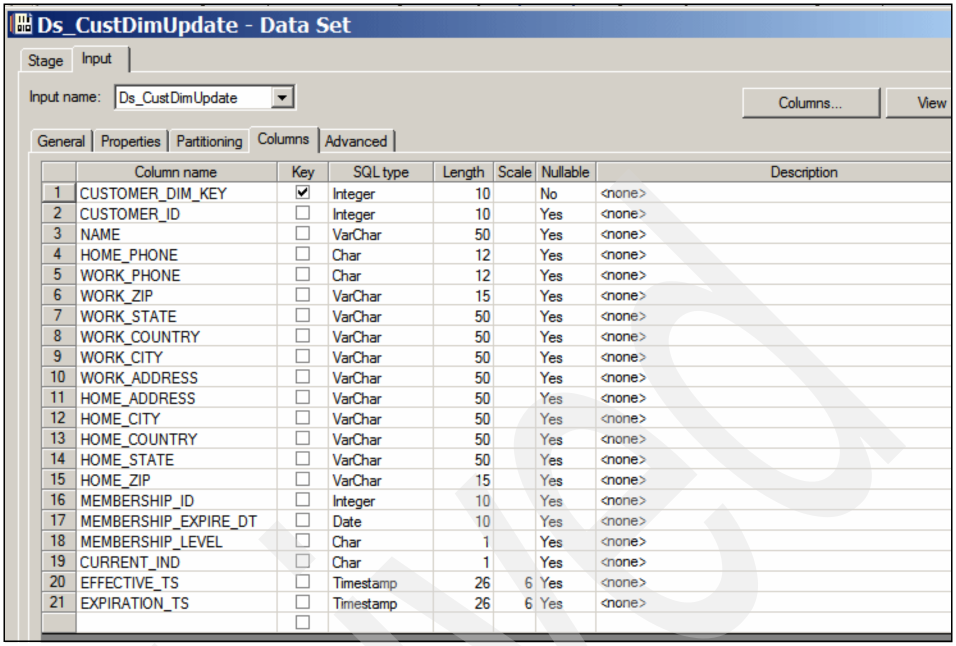
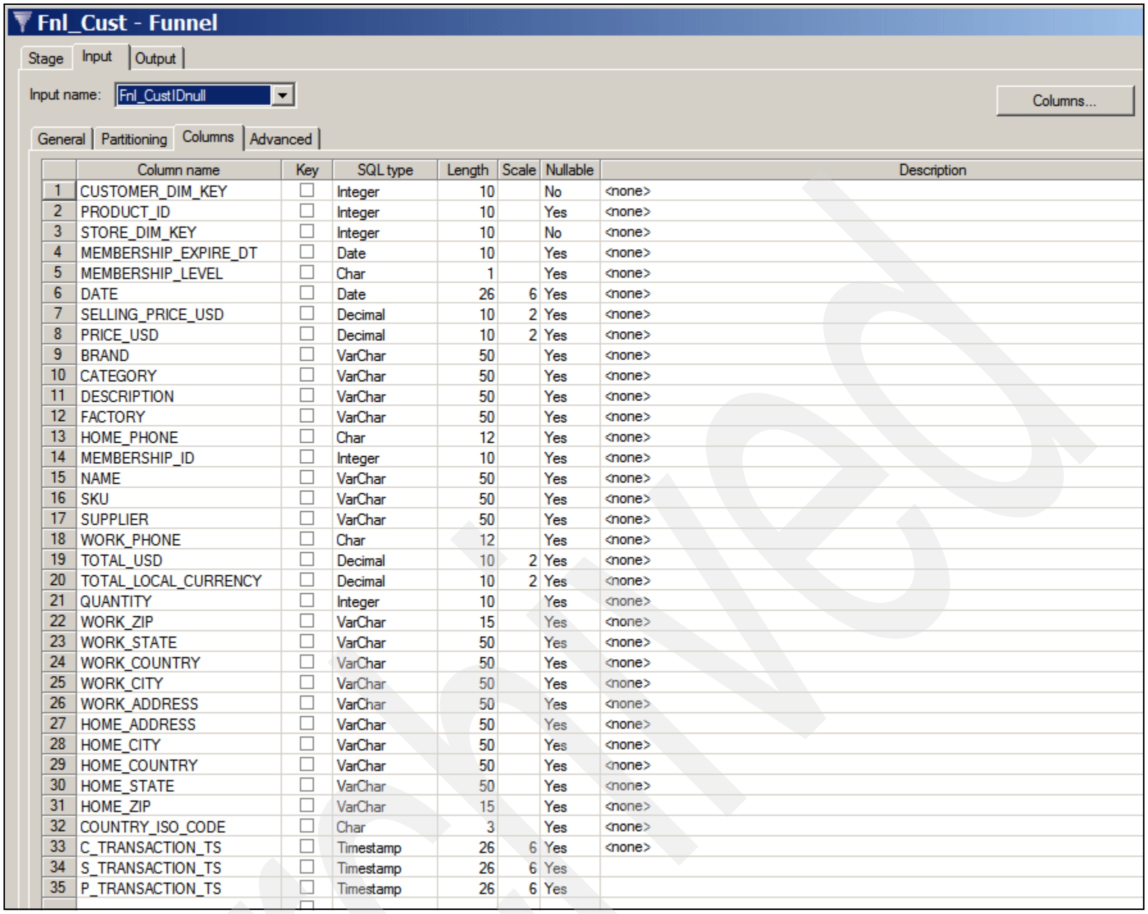


Figure 3-452 Create the J17_DailyCreateSalesFactDS job 22/68



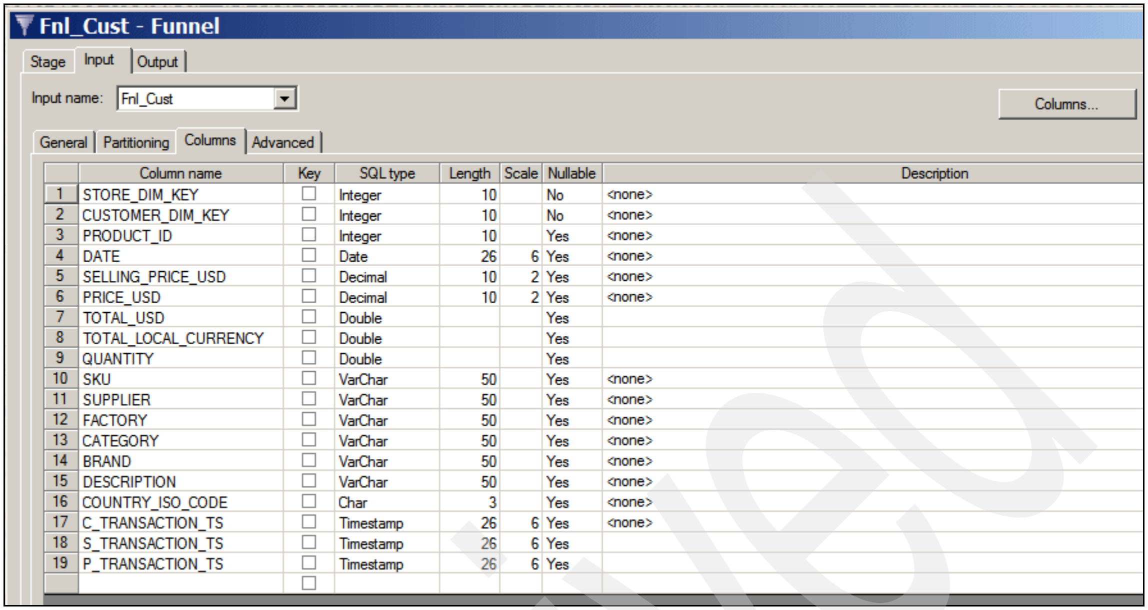
	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	CUSTOMER_DIM_KEY	☑	Integer	10		No	<none>
2	CUSTOMER_ID	☐	Integer	10		Yes	<none>
3	NAME	☐	VarChar	50		Yes	<none>
4	HOME_PHONE	☐	Char	12		Yes	<none>
5	WORK_PHONE	☐	Char	12		Yes	<none>
6	WORK_ZIP	☐	VarChar	15		Yes	<none>
7	WORK_STATE	☐	VarChar	50		Yes	<none>
8	WORK_COUNTRY	☐	VarChar	50		Yes	<none>
9	WORK_CITY	☐	VarChar	50		Yes	<none>
10	WORK_ADDRESS	☐	VarChar	50		Yes	<none>
11	HOME_ADDRESS	☐	VarChar	50		Yes	<none>
12	HOME_CITY	☐	VarChar	50		Yes	<none>
13	HOME_COUNTRY	☐	VarChar	50		Yes	<none>
14	HOME_STATE	☐	VarChar	50		Yes	<none>
15	HOME_ZIP	☐	VarChar	15		Yes	<none>
16	MEMBERSHIP_ID	☐	Integer	10		Yes	<none>
17	MEMBERSHIP_EXPIRE_DT	☐	Date	10		Yes	<none>
18	MEMBERSHIP_LEVEL	☐	Char	1		Yes	<none>
19	CURRENT_IND	☐	Char	1		Yes	<none>
20	EFFECTIVE_TS	☐	Timestamp	26	6	Yes	<none>
21	EXPIRATION_TS	☐	Timestamp	26	6	Yes	<none>
		☐					

Figure 3-453 Create the J17_DailyCreateSalesFactDS job 23/68



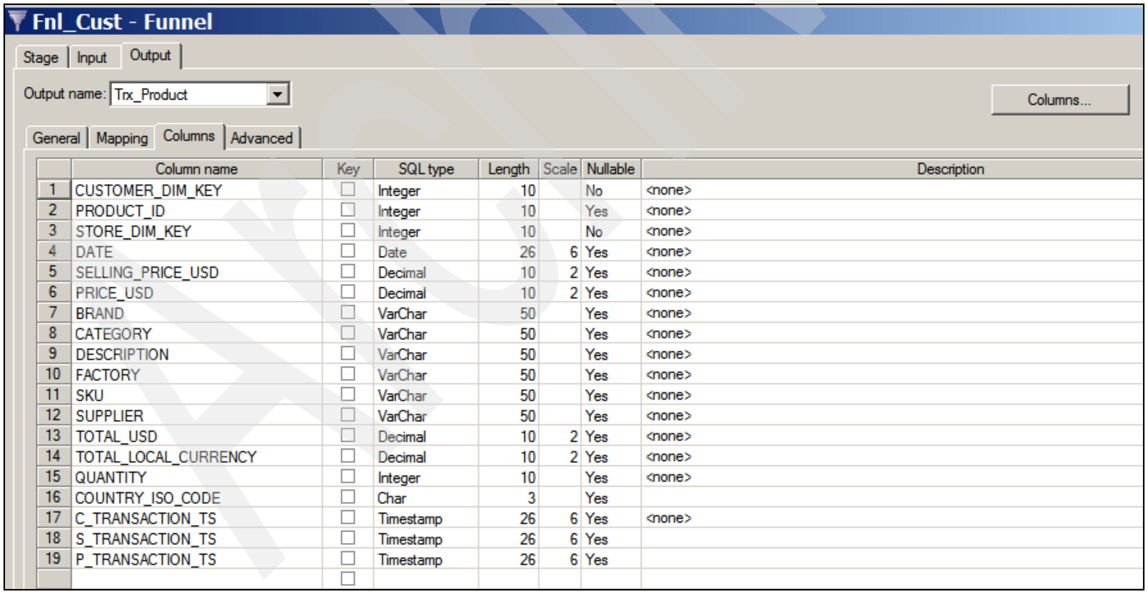
	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	CUSTOMER_DIM_KEY	☐	Integer	10		No	<none>
2	PRODUCT_ID	☐	Integer	10		Yes	<none>
3	STORE_DIM_KEY	☐	Integer	10		No	<none>
4	MEMBERSHIP_EXPIRE_DT	☐	Date	10		Yes	<none>
5	MEMBERSHIP_LEVEL	☐	Char	1		Yes	<none>
6	DATE	☐	Date	26	6	Yes	<none>
7	SELLING_PRICE_USD	☐	Decimal	10	2	Yes	<none>
8	PRICE_USD	☐	Decimal	10	2	Yes	<none>
9	BRAND	☐	VarChar	50		Yes	<none>
10	CATEGORY	☐	VarChar	50		Yes	<none>
11	DESCRIPTION	☐	VarChar	50		Yes	<none>
12	FACTORY	☐	VarChar	50		Yes	<none>
13	HOME_PHONE	☐	Char	12		Yes	<none>
14	MEMBERSHIP_ID	☐	Integer	10		Yes	<none>
15	NAME	☐	VarChar	50		Yes	<none>
16	SKU	☐	VarChar	50		Yes	<none>
17	SUPPLIER	☐	VarChar	50		Yes	<none>
18	WORK_PHONE	☐	Char	12		Yes	<none>
19	TOTAL_USD	☐	Decimal	10	2	Yes	<none>
20	TOTAL_LOCAL_CURRENCY	☐	Decimal	10	2	Yes	<none>
21	QUANTITY	☐	Integer	10		Yes	<none>
22	WORK_ZIP	☐	VarChar	15		Yes	<none>
23	WORK_STATE	☐	VarChar	50		Yes	<none>
24	WORK_COUNTRY	☐	VarChar	50		Yes	<none>
25	WORK_CITY	☐	VarChar	50		Yes	<none>
26	WORK_ADDRESS	☐	VarChar	50		Yes	<none>
27	HOME_ADDRESS	☐	VarChar	50		Yes	<none>
28	HOME_CITY	☐	VarChar	50		Yes	<none>
29	HOME_COUNTRY	☐	VarChar	50		Yes	<none>
30	HOME_STATE	☐	VarChar	50		Yes	<none>
31	HOME_ZIP	☐	VarChar	15		Yes	<none>
32	COUNTRY_ISO_CODE	☐	Char	3		Yes	<none>
33	C_TRANSACTION_TS	☐	Timestamp	26	6	Yes	<none>
34	S_TRANSACTION_TS	☐	Timestamp	26	6	Yes	
35	P_TRANSACTION_TS	☐	Timestamp	26	6	Yes	

Figure 3-454 Create the J17_DailyCreateSalesFactDS job 24/68



	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	STORE_DIM_KEY		Integer	10		No	<none>
2	CUSTOMER_DIM_KEY		Integer	10		No	<none>
3	PRODUCT_ID		Integer	10		Yes	<none>
4	DATE		Date	26	6	Yes	<none>
5	SELLING_PRICE_USD		Decimal	10	2	Yes	<none>
6	PRICE_USD		Decimal	10	2	Yes	<none>
7	TOTAL_USD		Double			Yes	
8	TOTAL_LOCAL_CURRENCY		Double			Yes	
9	QUANTITY		Double			Yes	
10	SKU		VarChar	50		Yes	<none>
11	SUPPLIER		VarChar	50		Yes	<none>
12	FACTORY		VarChar	50		Yes	<none>
13	CATEGORY		VarChar	50		Yes	<none>
14	BRAND		VarChar	50		Yes	<none>
15	DESCRIPTION		VarChar	50		Yes	<none>
16	COUNTRY_ISO_CODE		Char	3		Yes	<none>
17	C_TRANSACTION_TS		Timestamp	26	6	Yes	<none>
18	S_TRANSACTION_TS		Timestamp	26	6	Yes	
19	P_TRANSACTION_TS		Timestamp	26	6	Yes	

Figure 3-455 Create the J17_DailyCreateSalesFactDS job 25/68



	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	CUSTOMER_DIM_KEY		Integer	10		No	<none>
2	PRODUCT_ID		Integer	10		Yes	<none>
3	STORE_DIM_KEY		Integer	10		No	<none>
4	DATE		Date	26	6	Yes	<none>
5	SELLING_PRICE_USD		Decimal	10	2	Yes	<none>
6	PRICE_USD		Decimal	10	2	Yes	<none>
7	BRAND		VarChar	50		Yes	<none>
8	CATEGORY		VarChar	50		Yes	<none>
9	DESCRIPTION		VarChar	50		Yes	<none>
10	FACTORY		VarChar	50		Yes	<none>
11	SKU		VarChar	50		Yes	<none>
12	SUPPLIER		VarChar	50		Yes	<none>
13	TOTAL_USD		Decimal	10	2	Yes	<none>
14	TOTAL_LOCAL_CURRENCY		Decimal	10	2	Yes	<none>
15	QUANTITY		Integer	10		Yes	<none>
16	COUNTRY_ISO_CODE		Char	3		Yes	
17	C_TRANSACTION_TS		Timestamp	26	6	Yes	<none>
18	S_TRANSACTION_TS		Timestamp	26	6	Yes	
19	P_TRANSACTION_TS		Timestamp	26	6	Yes	

Figure 3-456 Create the J17_DailyCreateSalesFactDS job 26/68

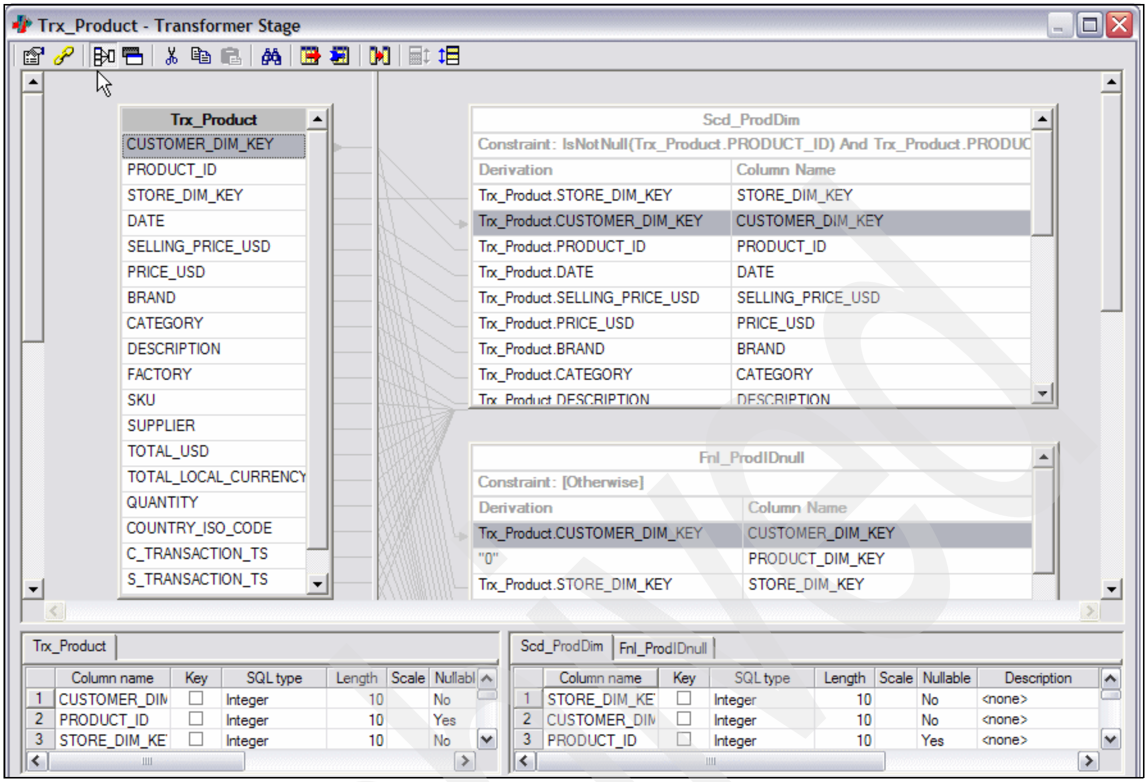


Figure 3-457 Create the J17_DailyCreateSalesFactDS job 27/68

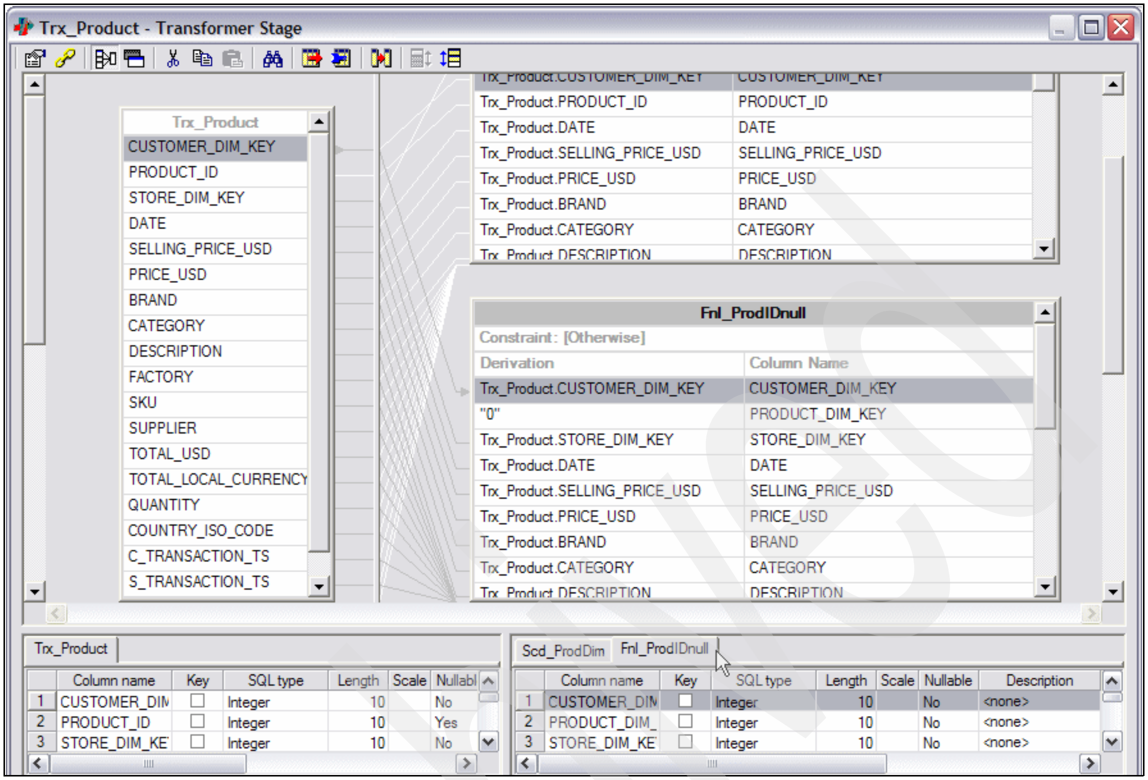


Figure 3-458 Create the J17_DailyCreateSalesFactDS job 28/68

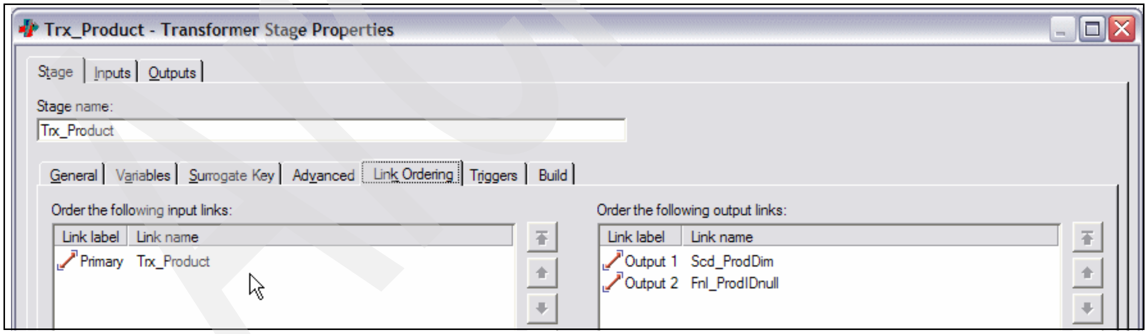


Figure 3-459 Create the J17_DailyCreateSalesFactDS job 29/68

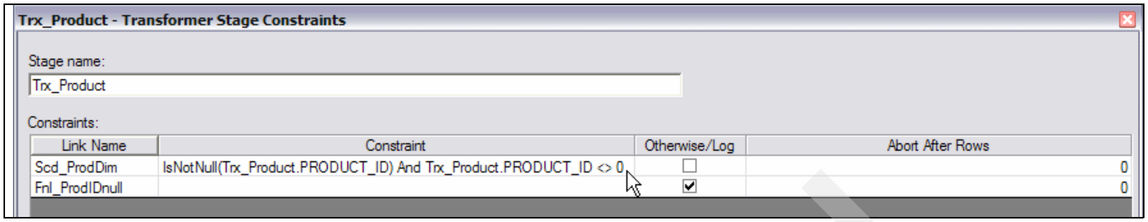


Figure 3-460 Create the J17_DailyCreateSalesFactDS job 30/68

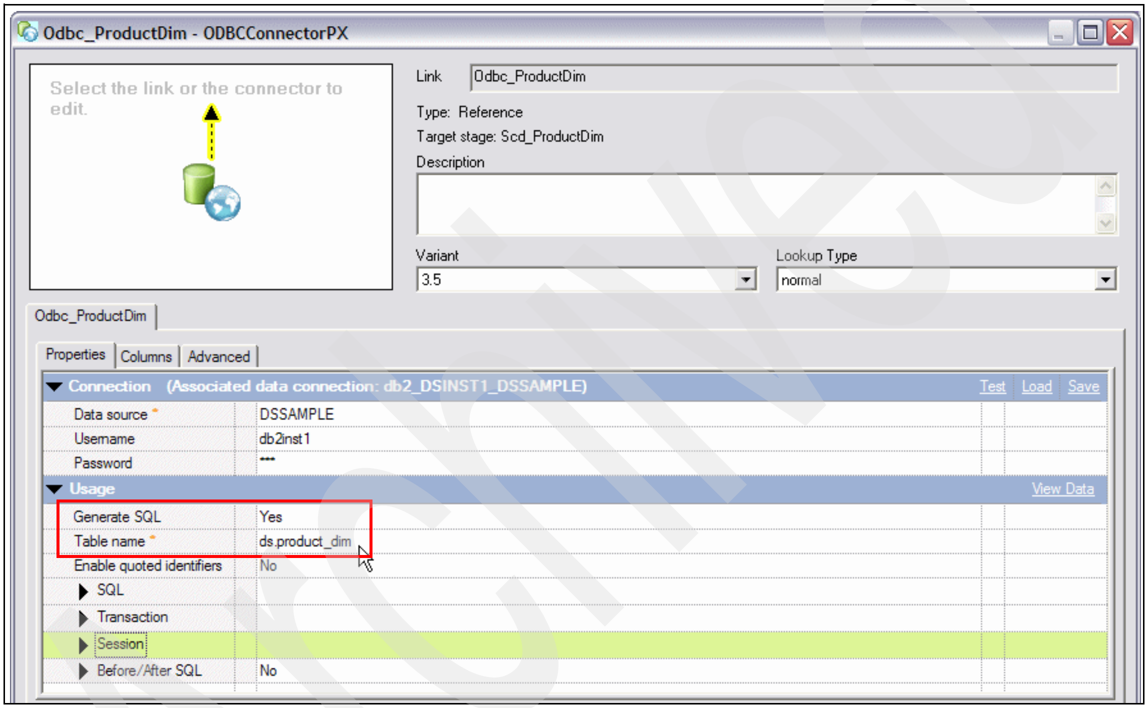


Figure 3-461 Create the J17_DailyCreateSalesFactDS job 31/68

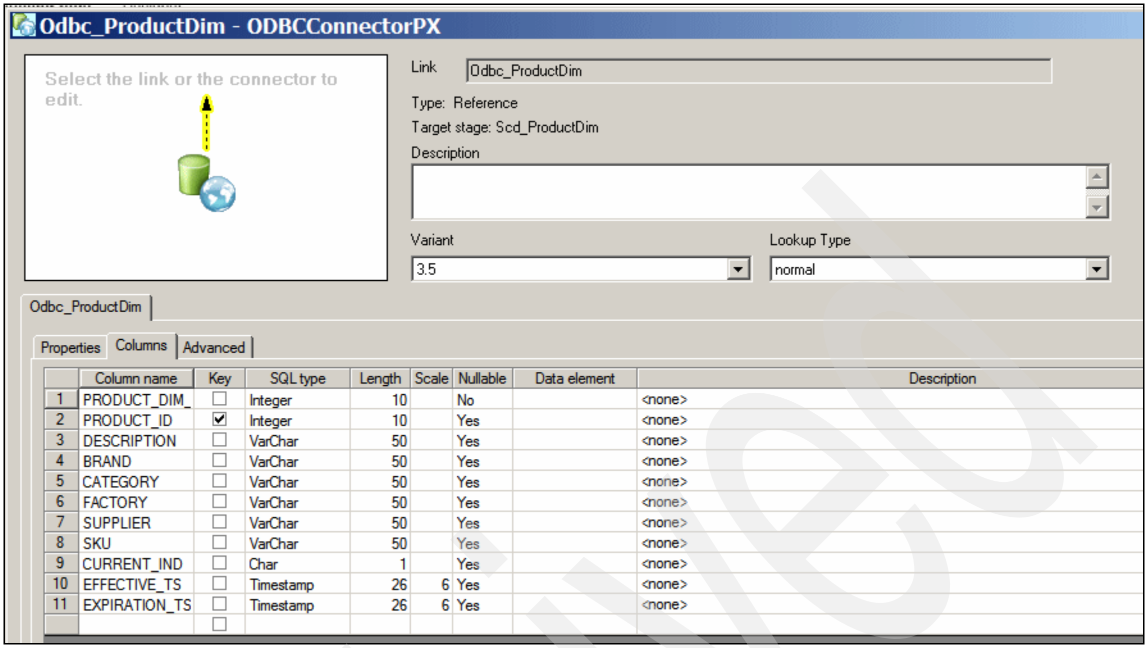


Figure 3-462 Create the J17_DailyCreateSalesFactDS job 32/68

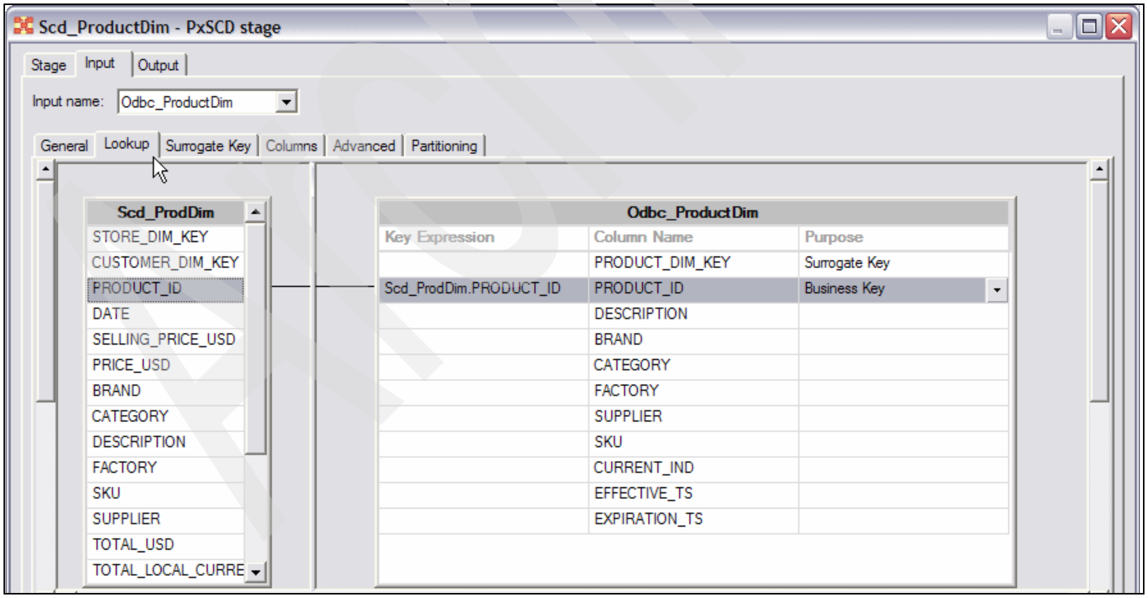


Figure 3-463 Create the J17_DailyCreateSalesFactDS job 33/68

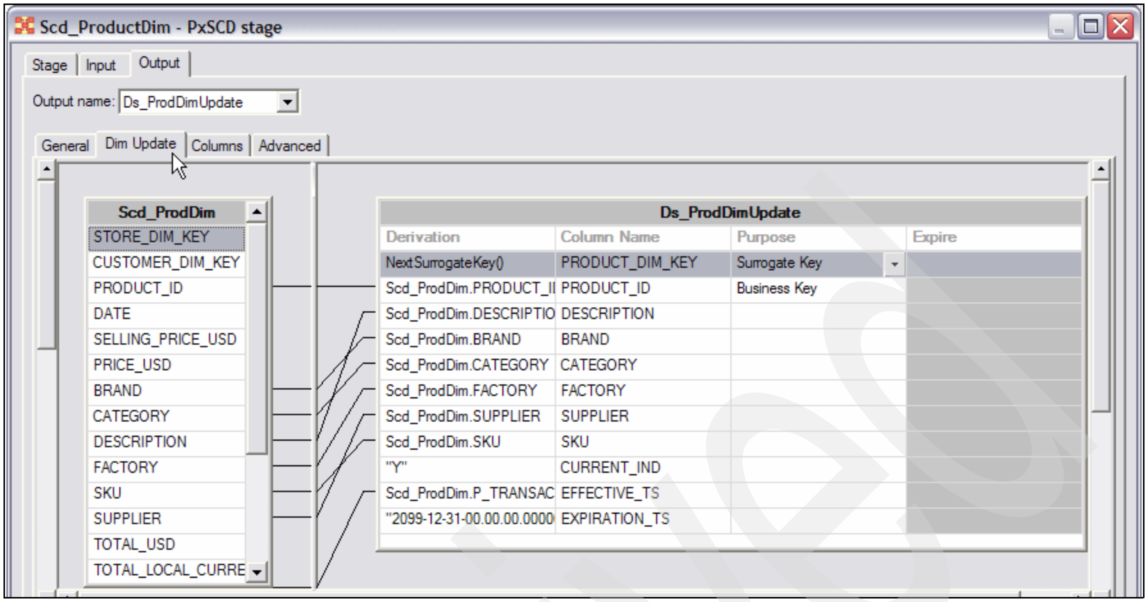


Figure 3-464 Create the J17_DailyCreateSalesFactDS job 34/68

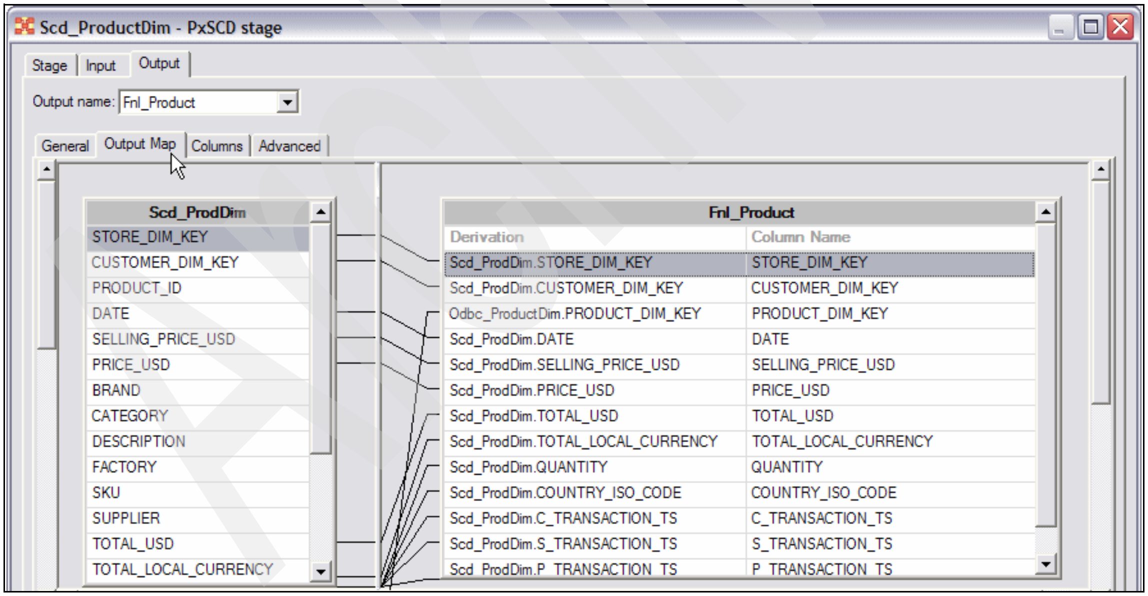
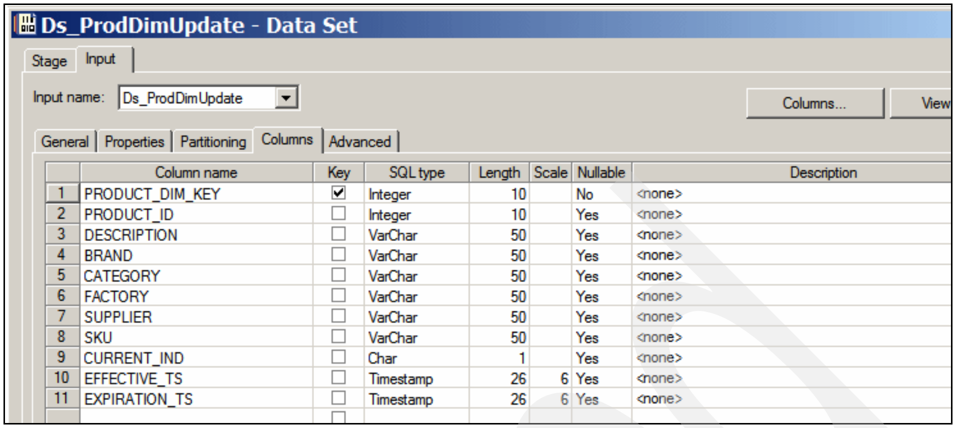
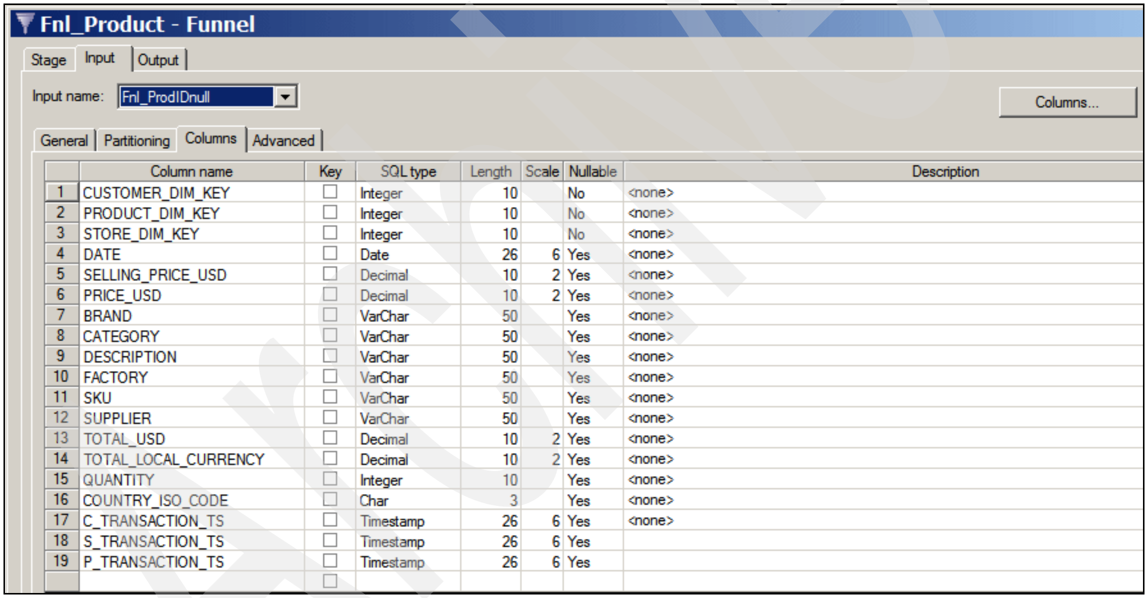


Figure 3-465 Create the J17_DailyCreateSalesFactDS job 35/68



	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	PRODUCT_DIM_KEY	☑	Integer	10		No	<none>
2	PRODUCT_ID	☐	Integer	10		Yes	<none>
3	DESCRIPTION	☐	VarChar	50		Yes	<none>
4	BRAND	☐	VarChar	50		Yes	<none>
5	CATEGORY	☐	VarChar	50		Yes	<none>
6	FACTORY	☐	VarChar	50		Yes	<none>
7	SUPPLIER	☐	VarChar	50		Yes	<none>
8	SKU	☐	VarChar	50		Yes	<none>
9	CURRENT_IND	☐	Char	1		Yes	<none>
10	EFFECTIVE_TS	☐	Timestamp	26	6	Yes	<none>
11	EXPIRATION_TS	☐	Timestamp	26	6	Yes	<none>

Figure 3-466 Create the J17_DailyCreateSalesFactDS job 36/68



	Column name	Key	SQL type	Length	Scale	Nullable	Description
1	CUSTOMER_DIM_KEY	☐	Integer	10		No	<none>
2	PRODUCT_DIM_KEY	☐	Integer	10		No	<none>
3	STORE_DIM_KEY	☐	Integer	10		No	<none>
4	DATE	☐	Date	26	6	Yes	<none>
5	SELLING_PRICE_USD	☐	Decimal	10	2	Yes	<none>
6	PRICE_USD	☐	Decimal	10	2	Yes	<none>
7	BRAND	☐	VarChar	50		Yes	<none>
8	CATEGORY	☐	VarChar	50		Yes	<none>
9	DESCRIPTION	☐	VarChar	50		Yes	<none>
10	FACTORY	☐	VarChar	50		Yes	<none>
11	SKU	☐	VarChar	50		Yes	<none>
12	SUPPLIER	☐	VarChar	50		Yes	<none>
13	TOTAL_USD	☐	Decimal	10	2	Yes	<none>
14	TOTAL_LOCAL_CURRENCY	☐	Decimal	10	2	Yes	<none>
15	QUANTITY	☐	Integer	10		Yes	<none>
16	COUNTRY_ISO_CODE	☐	Char	3		Yes	<none>
17	C_TRANSACTION_TS	☐	Timestamp	26	6	Yes	<none>
18	S_TRANSACTION_TS	☐	Timestamp	26	6	Yes	
19	P_TRANSACTION_TS	☐	Timestamp	26	6	Yes	

Figure 3-467 Create the J17_DailyCreateSalesFactDS job 37/68

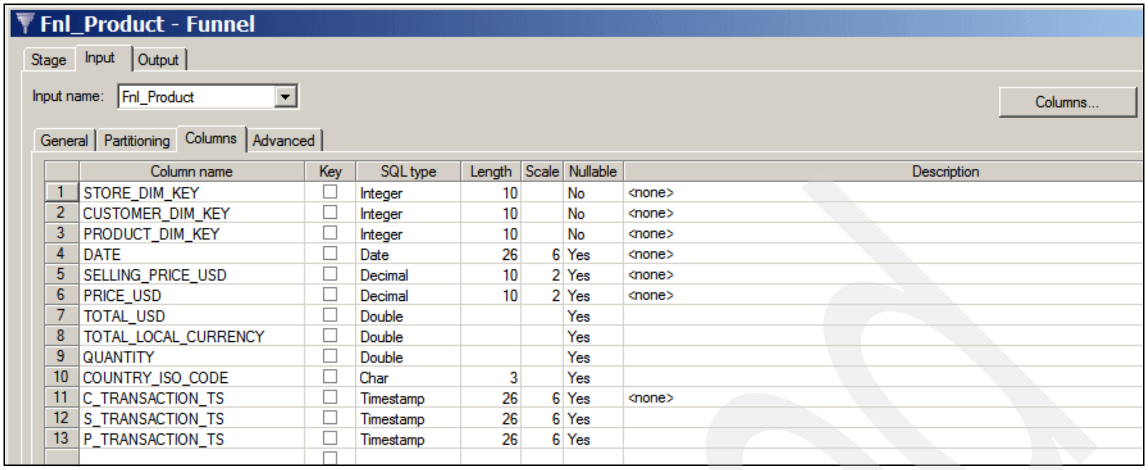


Figure 3-468 Create the J17_DailyCreateSalesFactDS job 38/68

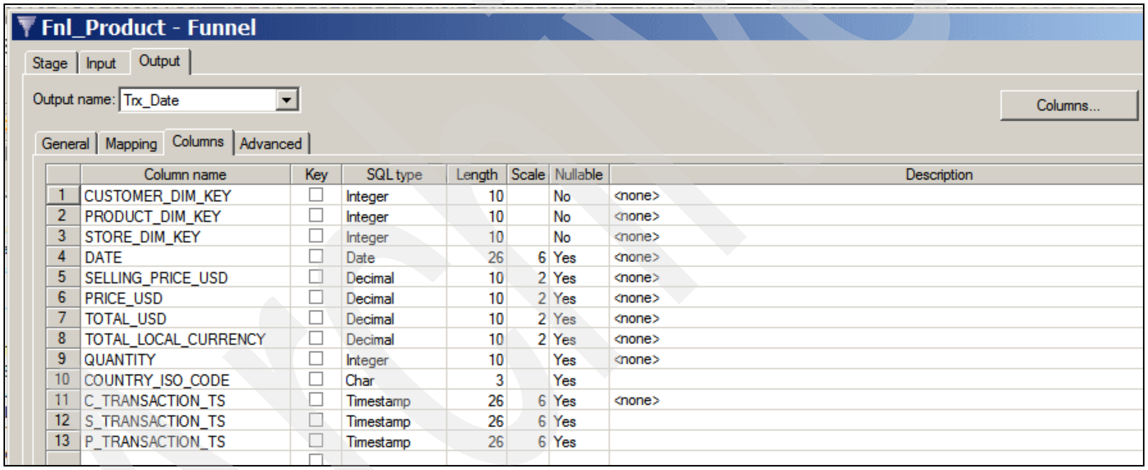


Figure 3-469 Create the J17_DailyCreateSalesFactDS job 39/68

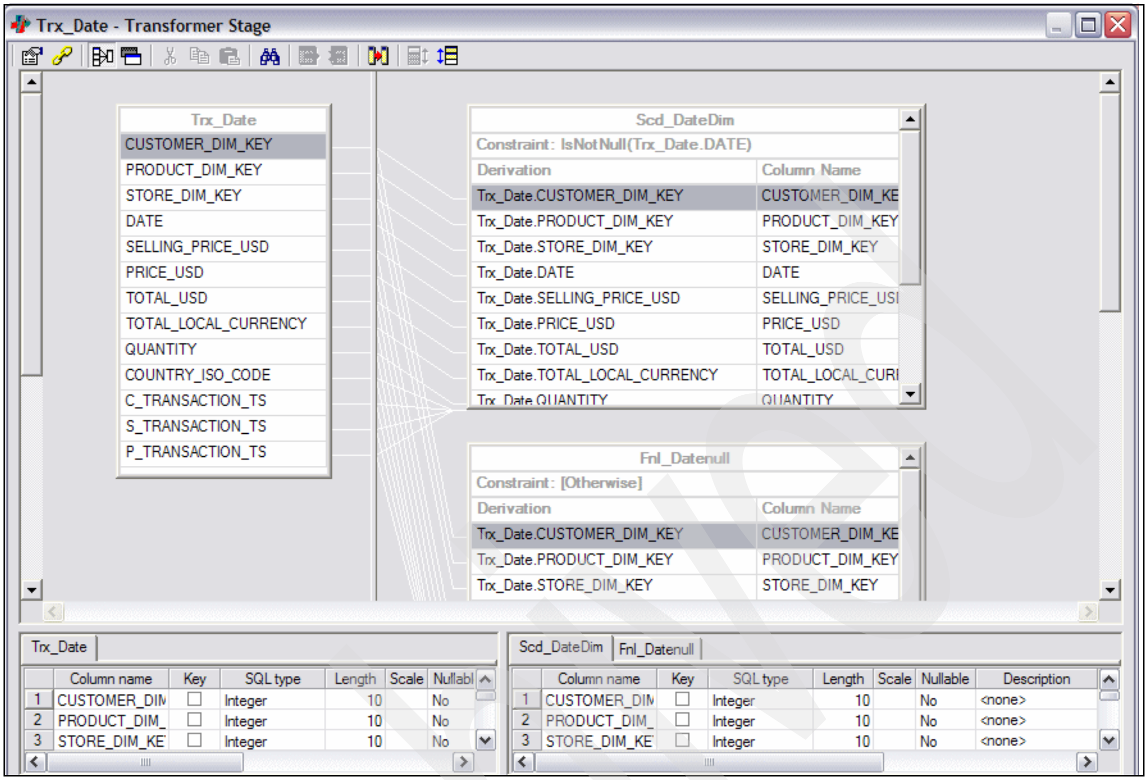


Figure 3-470 Create the J17_DailyCreateSalesFactDS job 40/68

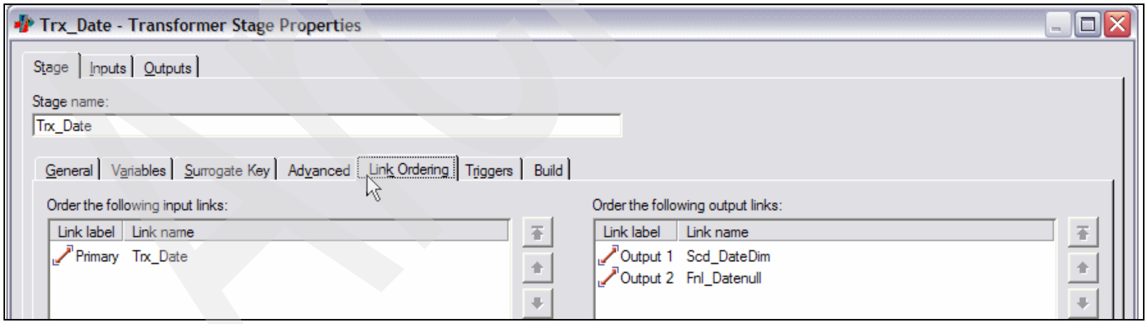


Figure 3-471 Create the J17_DailyCreateSalesFactDS job 41/68

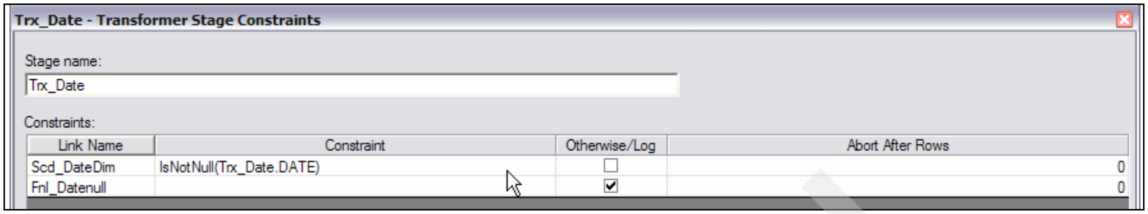


Figure 3-472 Create the J17_DailyCreateSalesFactDS job 42/68

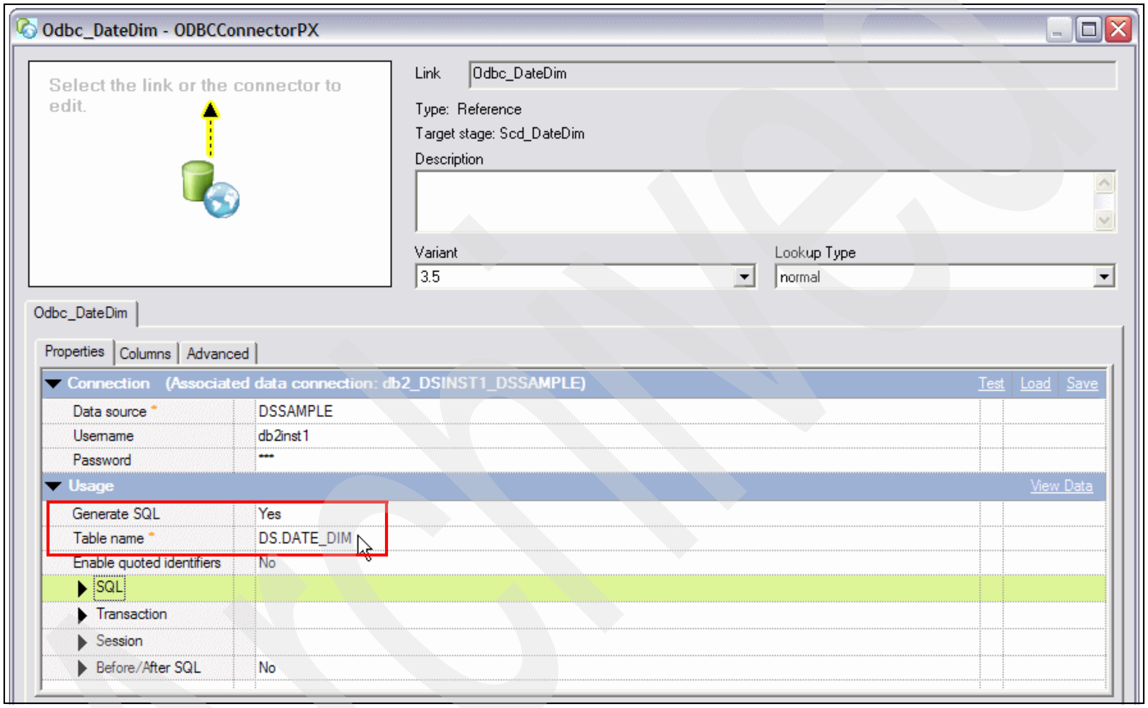


Figure 3-473 Create the J17_DailyCreateSalesFactDS job 43/68

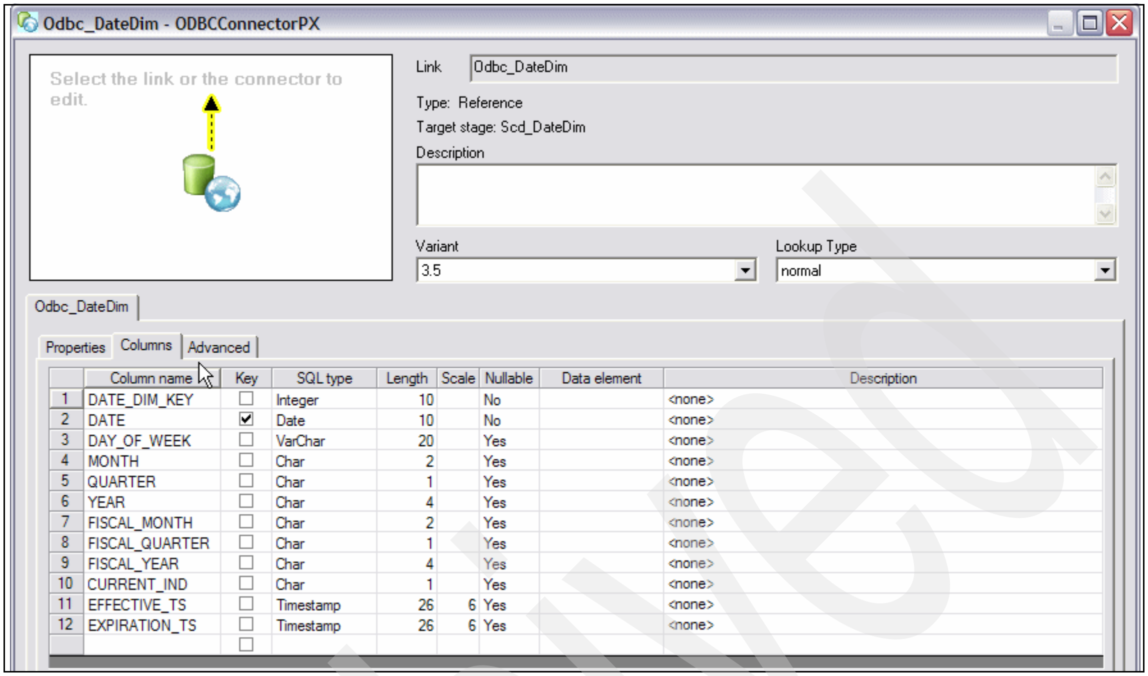


Figure 3-474 Create the J17_DailyCreateSalesFactDS job 44/68

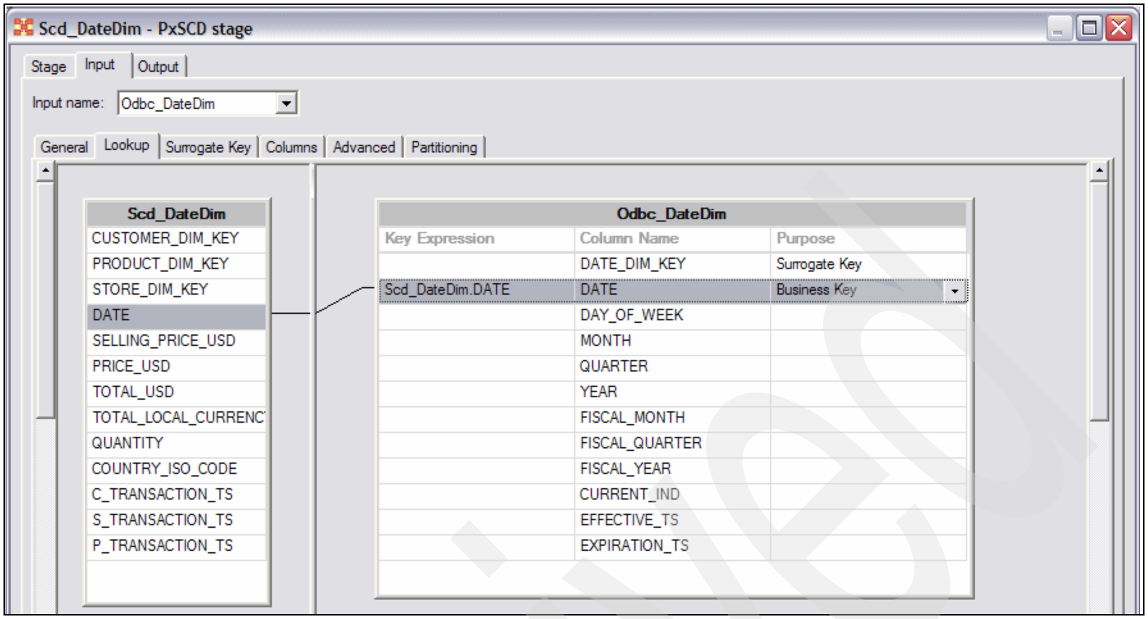


Figure 3-475 Create the J17_DailyCreateSalesFactDS job 45/68

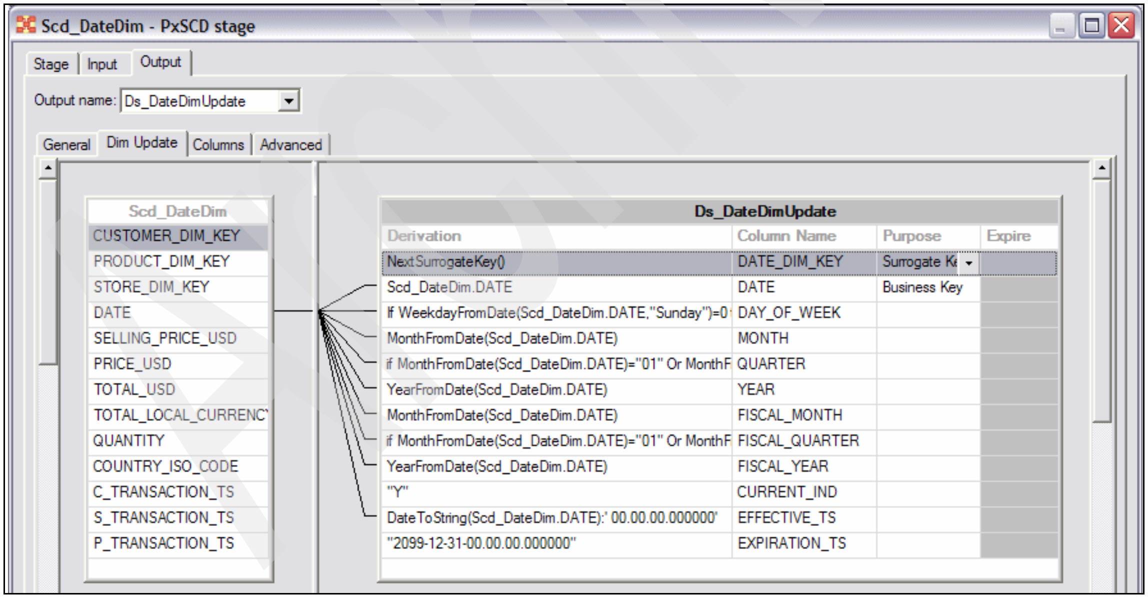


Figure 3-476 Create the J17_DailyCreateSalesFactDS job 46/68

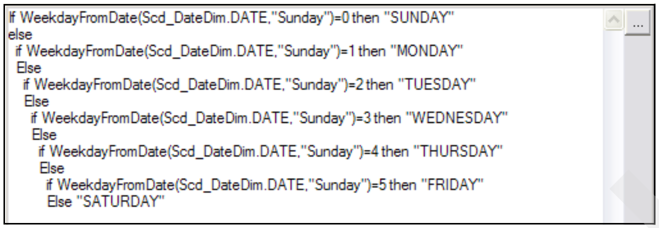


Figure 3-477 Create the J17_DailyCreateSalesFactDS job 47/68

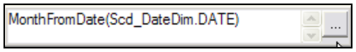


Figure 3-478 Create the J17_DailyCreateSalesFactDS job 48/68

Figure 3-479 Create the J17_DailyCreateSalesFactDS job 49/68

Figure 3-480 Create the J17_DailyCreateSalesFactDS job 50/68

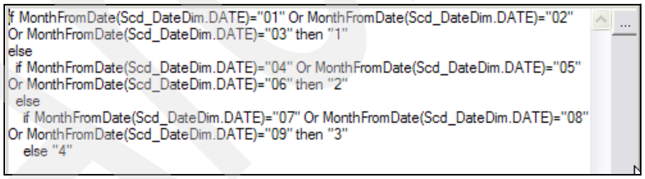


Figure 3-481 Create the J17_DailyCreateSalesFactDS job 51/68

Figure 3-482 Create the J17_DailyCreateSalesFactDS job 52/68

Figure 3-483 Create the J17_DailyCreateSalesFactDS job 53/68

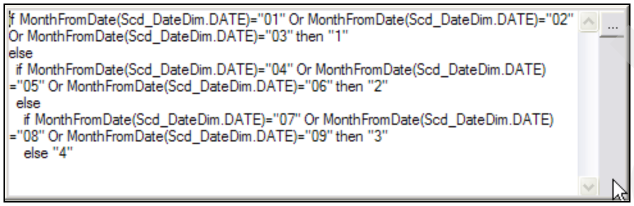


Figure 3-484 Create the J17_DailyCreateSalesFactDS job 54/68

Figure 3-485 Create the J17_DailyCreateSalesFactDS job 55/68

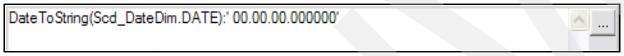


Figure 3-486 Create the J17_DailyCreateSalesFactDS job 56/68

Figure 3-487 Create the J17_DailyCreateSalesFactDS job 57/68

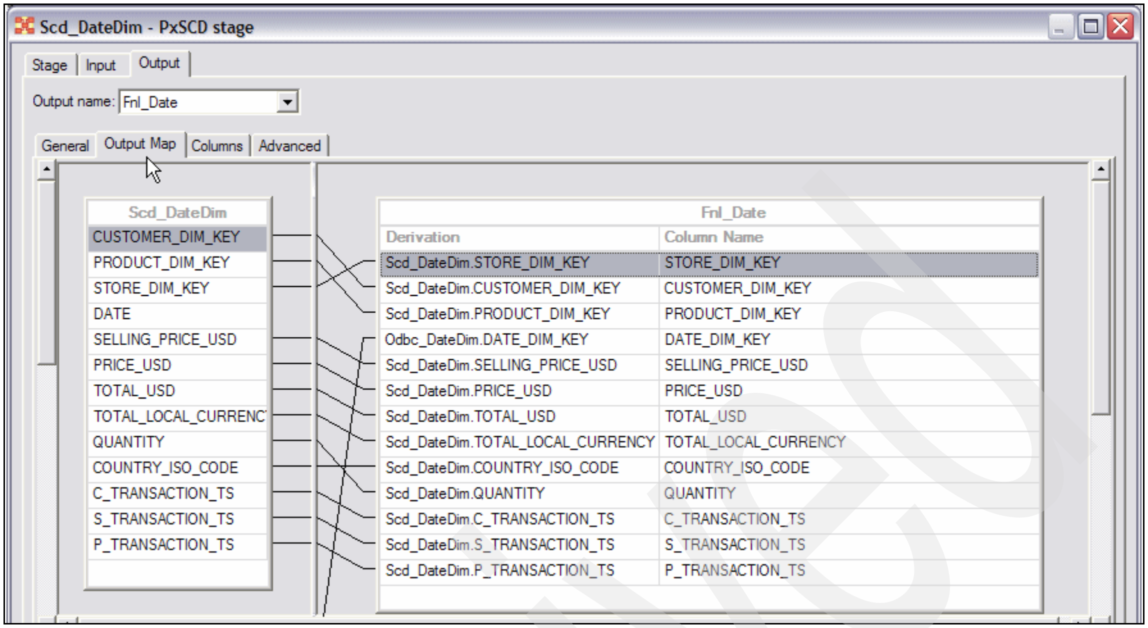


Figure 3-488 Create the J17_DailyCreateSalesFactDS job 58/68

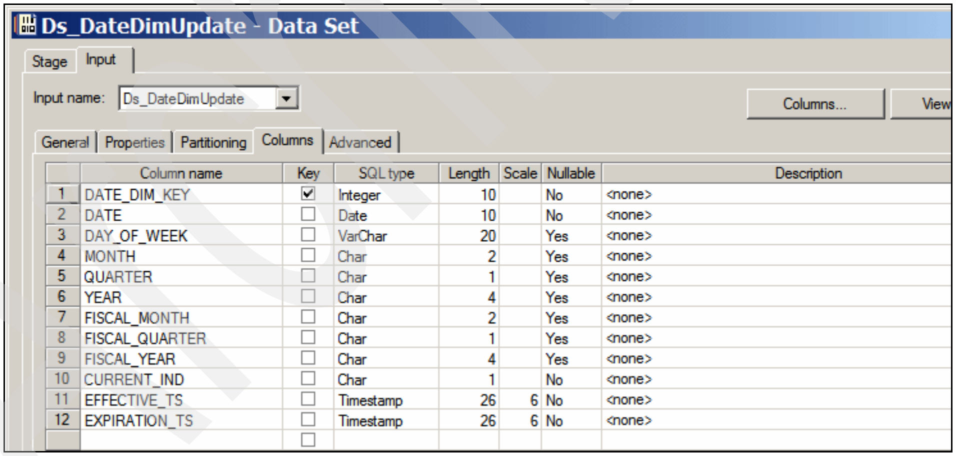


Figure 3-489 Create the J17_DailyCreateSalesFactDS job 59/68

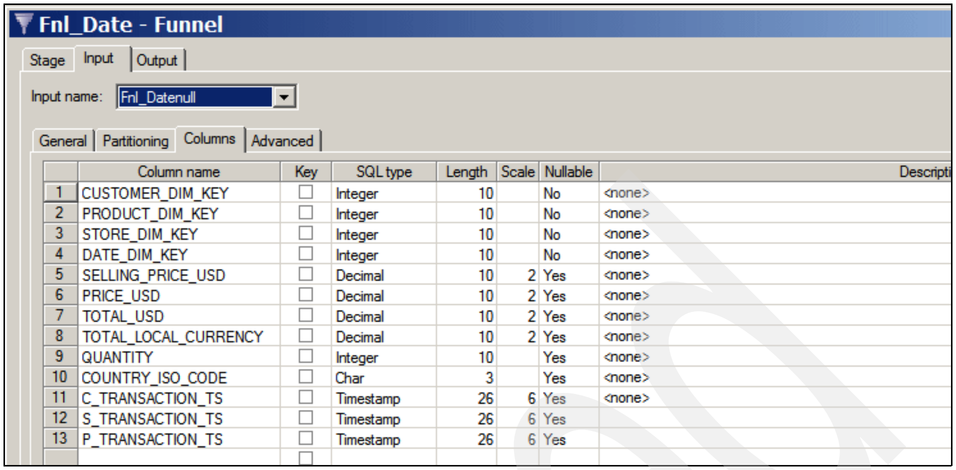


Figure 3-490 Create the J17_DailyCreateSalesFactDS job 60/68

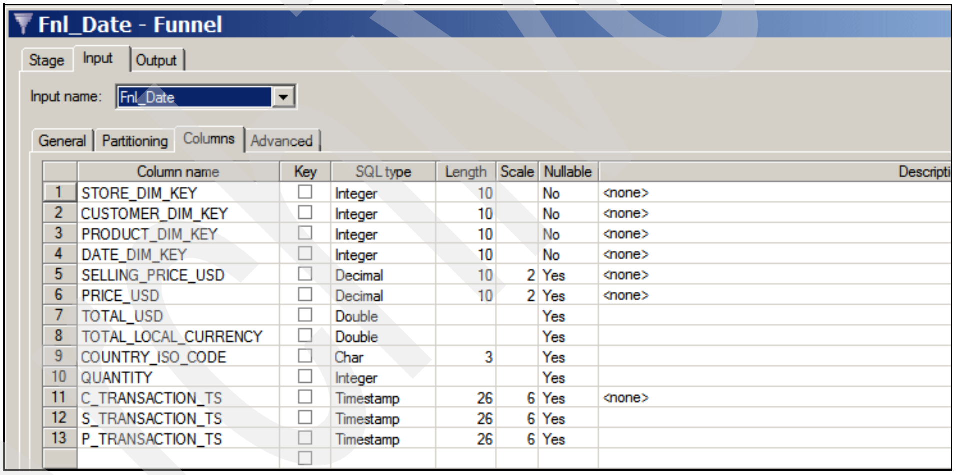


Figure 3-491 Create the J17_DailyCreateSalesFactDS job 61/68

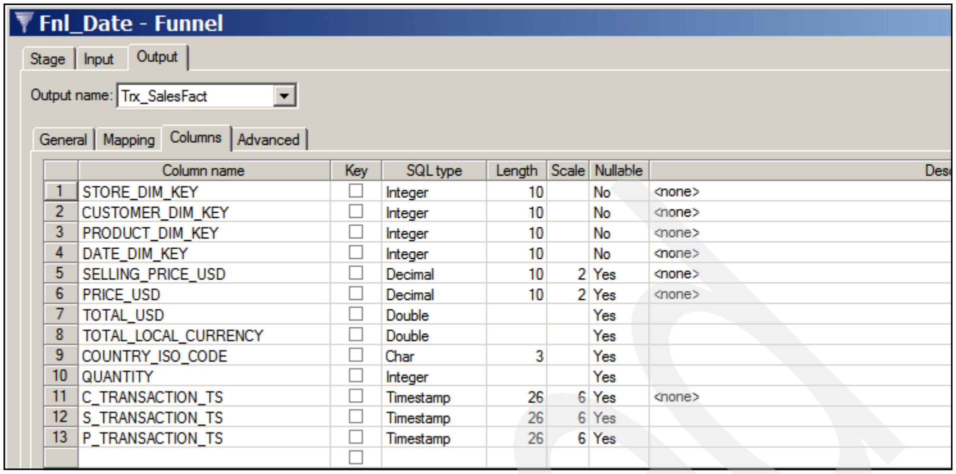


Figure 3-492 Create the J17_DailyCreateSalesFactDS job 62/68

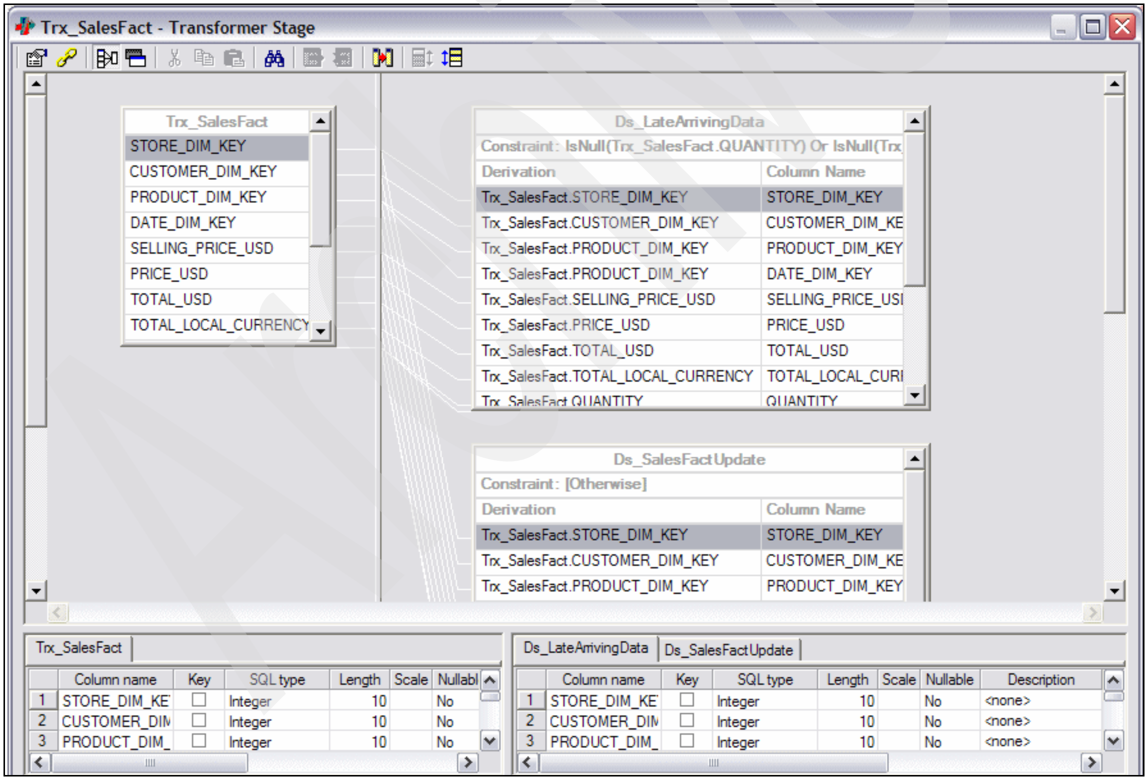


Figure 3-493 Create the J17_DailyCreateSalesFactDS job 63/68

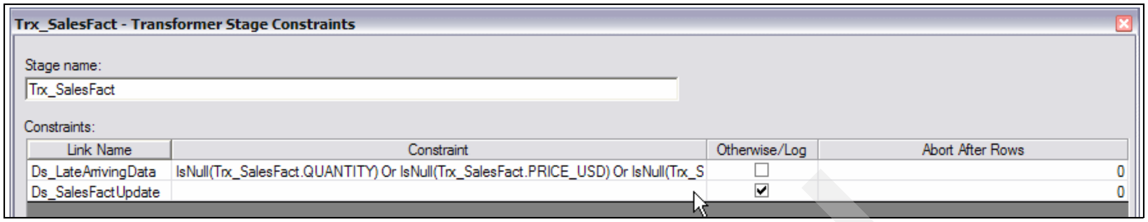


Figure 3-494 Create the J17_DailyCreateSalesFactDS job 64/68

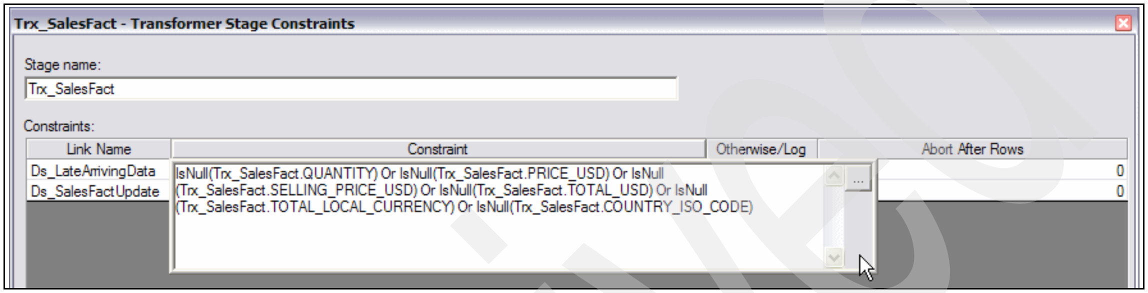


Figure 3-495 Create the J17_DailyCreateSalesFactDS job 65/68

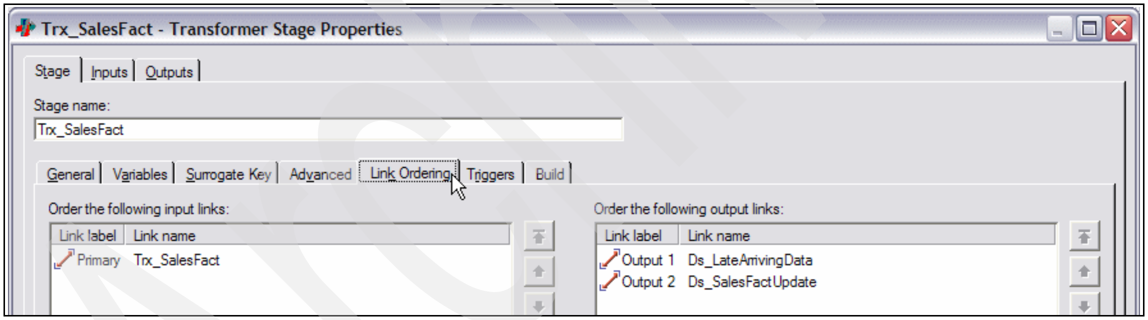


Figure 3-496 Create the J17_DailyCreateSalesFactDS job 66/68

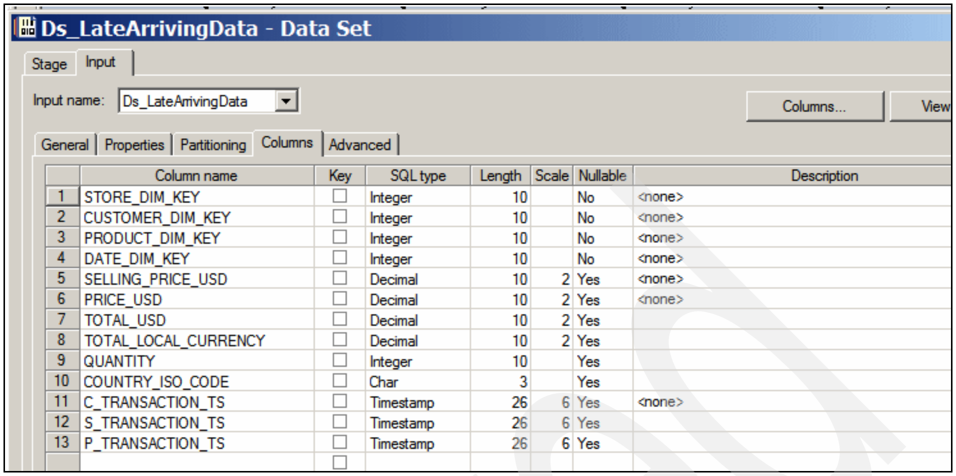


Figure 3-497 Create the J17_DailyCreateSalesFactDS job 67/68

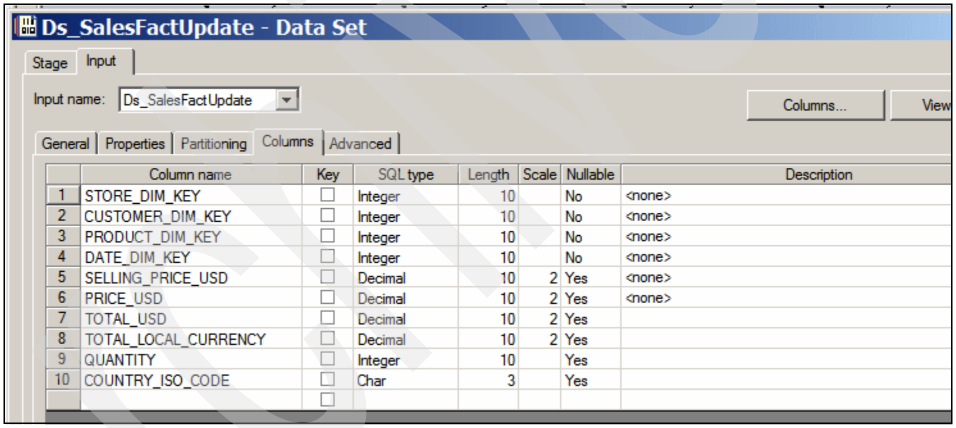


Figure 3-498 Create the J17_DailyCreateSalesFactDS job 68/68

J17_DailyCreateSalesFactDS (Day1) execution

Figure 3-499 on page 476 through Figure 3-506 on page 477 show the results of the execution of this job with Day 1 data described earlier.

- Figure 3-499 on page 476 shows the results of the execution. It accepts 9 rows as input from the “J16_Daily_CreateScdInputDS (Day 1) execution” on page 430 job as seen in Figure 3-425 on page 431 through Figure 3-430 on page 433.
- The outputs of this job are as follows:
  - The input had two Customer dimension attribute changes. One was an update of CUSTOMER_ID 1, while the other was a delete of CUSTOMER_ID 7. However, the input in this case does not have the operation code, that is, update or delete.

    However, the Ds_CustDimUpdate data set has only 1 row in the output corresponding to the Type 1 update of CUSTOMER_ID 1 as shown in Figure 3-500 on page 476 through Figure 3-502 on page 476. There is no corresponding record for CUSTOMER_ID 7 because all the values in the Type 1 and Type 2 attributes of this record are identical to those attributes for CUSTOMER_ID 7 in the CUSTOMER_DIM table, and the SCD stage therefore considers that it is not necessary to update the dimension table for this record.
  - Seven rows (as expected from the input) are written to the Ds_SalesFactUpdate data set with the appropriate surrogate key assigned to each sales transaction as shown in Figure 3-503 on page 477 through Figure 3-505 on page 477.
  - The two rows corresponding to late arriving dimensions in the input are rejected and written to the Ds_LateArrivingData data set as shown in Figure 3-505 on page 477 and Figure 3-506 on page 477.

The next step is to execute the job described in “J18_Daily_UpdateStoreDim (Day 1)” on page 478.

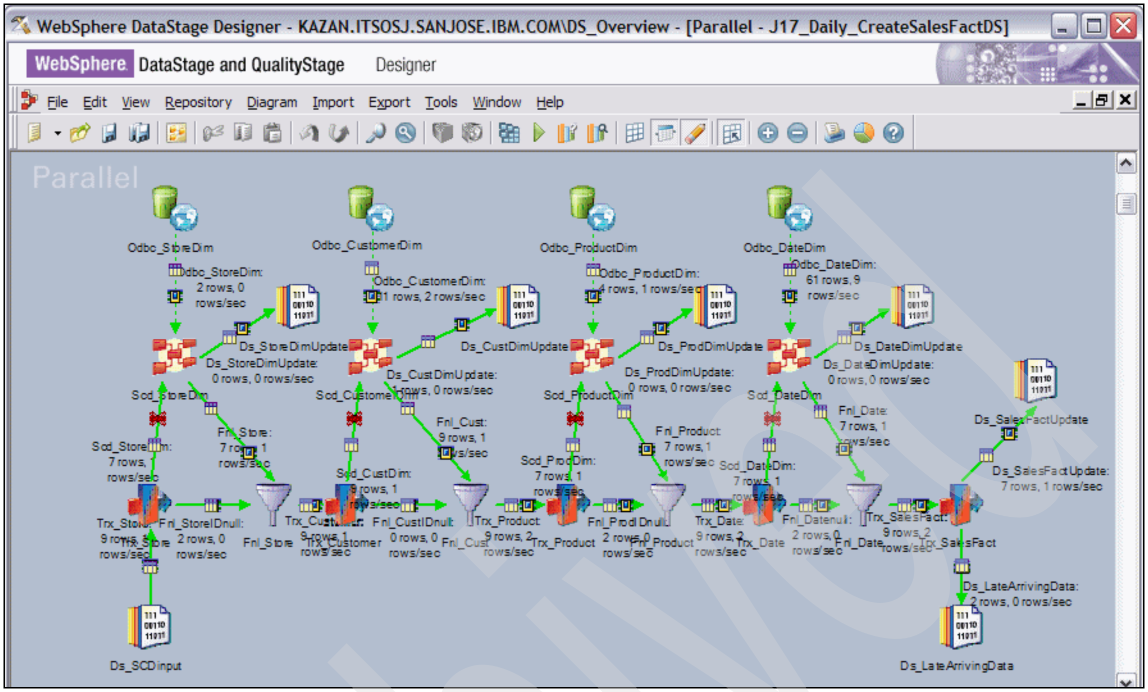
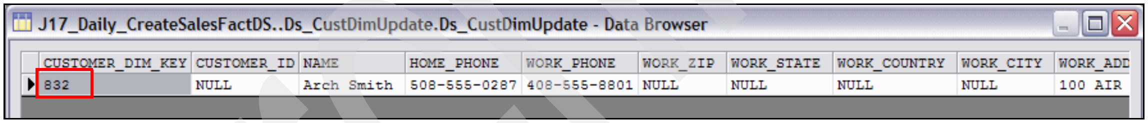


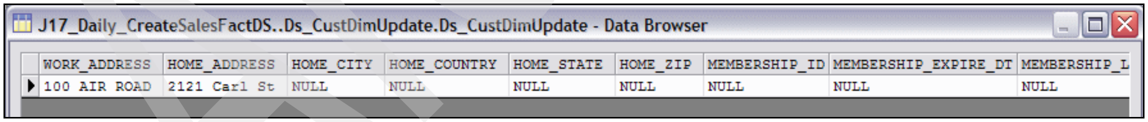
Figure 3-499 Execute the J17_DailyCreateSalesFactDS job (Day 1) 1/8



J17_Daily_CreateSalesFactDS..Ds_CustDimUpdate.Ds_CustDimUpdate - Data Browser

CUSTOMER_DIM_KEY	CUSTOMER_ID	NAME	HOME_PHONE	WORK_PHONE	WORK_ZIP	WORK_STATE	WORK_COUNTRY	WORK_CITY	WORK_ADD
832	NULL	Arch Smith	508-555-0287	408-555-8801	NULL	NULL	NULL	NULL	100 AIR

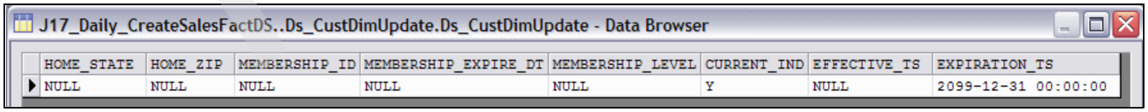
Figure 3-500 Execute the J17_DailyCreateSalesFactDS job (Day 1) 2/8



J17_Daily_CreateSalesFactDS..Ds_CustDimUpdate.Ds_CustDimUpdate - Data Browser

WORK_ADDRESS	HOME_ADDRESS	HOME_CITY	HOME_COUNTRY	HOME_STATE	HOME_ZIP	MEMBERSHIP_ID	MEMBERSHIP_EXPIRE_DT	MEMBERSHIP_L
100 AIR ROAD	2121 Carl St	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 3-501 Execute the J17_DailyCreateSalesFactDS job (Day 1) 3/8



J17_Daily_CreateSalesFactDS..Ds_CustDimUpdate.Ds_CustDimUpdate - Data Browser

HOME_STATE	HOME_ZIP	MEMBERSHIP_ID	MEMBERSHIP_EXPIRE_DT	MEMBERSHIP_LEVEL	CURRENT_IND	EFFECTIVE_TS	EXPIRATION_TS
NULL	NULL	NULL	NULL	NULL	Y	NULL	2099-12-31 00:00:00

Figure 3-502 Execute the J17_DailyCreateSalesFactDS job (Day 1) 4/8

J17_Daily_CreateSalesFactDS..Ds_SalesFactUpdate.Ds_SalesFactUpdate - Data Browser

STORE_DIM_KEY	CUSTOMER_DIM_KEY	PRODUCT_DIM_KEY	DATE_DIM_KEY	SELLING_PRICE_USD	PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURREN
743	836	777	37	00000015.00	00000017.69	00000030.0	00000030.00
742	840	776	37	00000037.00	00000037.00	00000111.0	00000111.00
742	832	777	37	00000025.00	00000035.00	00000050.0	00000050.00
743	842	777	37	00000015.00	00000017.69	00000030.0	00000030.00
742	842	777	37	00000033.33	00000035.00	00000033.3	00000033.33
743	842	776	37	00000015.00	00000017.69	00000030.0	00000029.02
742	839	777	37	00000037.00	00000037.00	00000037.0	00000037.00

Figure 3-503 Execute the J17_DailyCreateSalesFactDS job (Day 1) 5/8

J17_Daily_CreateSalesFactDS..Ds_SalesFactUpdate.Ds_SalesFactUpdate - Data Browser

PRODUCT_DIM_KEY	DATE_DIM_KEY	SELLING_PRICE_USD	PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURRENCY	QUANTITY	COUNTRY_ISO_CODE
777	37	00000015.00	00000017.69	00000030.0	00000030.00	2	USA
776	37	00000037.00	00000037.00	00000111.0	00000111.00	3	USA
777	37	00000025.00	00000035.00	00000050.0	00000050.00	2	USA
777	37	00000015.00	00000017.69	00000030.0	00000030.00	2	USA
777	37	00000033.33	00000035.00	00000033.3	00000033.33	1	USA
776	37	00000015.00	00000017.69	00000030.0	00000029.02	2	CAD
777	37	00000037.00	00000037.00	00000037.0	00000037.00	1	USA

Figure 3-504 Execute the J17_DailyCreateSalesFactDS job (Day 1) 6/8

J17_Daily_CreateSalesFactDS..Ds_LateArrivingData.Ds_LateArrivingData - Data Browser

STORE_DIM_KEY	CUSTOMER_DIM_KEY	PRODUCT_DIM_KEY	DATE_DIM_KEY	SELLING_PRICE_USD	PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURRENCY	QU
0	832	0	0	NULL	NULL	NULL	NULL	NU
0	837	0	0	NULL	NULL	NULL	NULL	NU

Figure 3-505 Execute the J17_DailyCreateSalesFactDS job (Day 1) 7/8

J17_Daily_CreateSalesFactDS..Ds_LateArrivingData.Ds_LateArrivingData - Data Browser

TOTAL_USD	TOTAL_LOCAL_CURRENCY	QUANTITY	COUNTRY_ISO_CODE	C_TRANSACTION_TS	S_TRANSACTION_TS	P_TRANSACTION_TS
NULL	NULL	NULL	NULL	2007-11-06 12:39:42	NULL	NULL
NULL	NULL	NULL	NULL	2007-11-06 23:49:42	NULL	NULL

Figure 3-506 Execute the J17_DailyCreateSalesFactDS job (Day 1) 8/8

J18_Daily_UpdateStoreDim (Day 1)

This job updates the Store dimension table using the file created in the "J17_DailyCreateSalesFactDS (Day1)" on page 433 job. The input record does not contain an operation code (insert, update, or delete). The update of the dimension table is therefore performed with an SQL UPDATE operation followed by an SQL INSERT operation using the surrogate key of the business key.

- If the record exists in the dimension table, then the SQL UPDATE operation will update the appropriate Type 1 columns and the SQL INSERT operation will fail.
- If the record does not exist in the dimension table, then the SQL UPDATE operation will fail and the SQL INSERT operation will succeed.

Any records that have nulls in the Type 1 columns CITY_POPULATION or STATE_POPULATION must be rejected, because this would otherwise set the corresponding Type 1 columns in the dimension table to NULL, which is not our desired semantics.

> **Note:** If you want to set the Type 1 columns to NULL in the dimension table, then you must take such action independently using the records in the reject file.

Figure 3-507 on page 479 through Figure 3-514 on page 483 explain the main stages in this job and the configuration of these stages as described in "J18_Daily_UpdateStoreDim (Day 1) configuration" on page 478, while Figure 3-515 on page 484 explains the execution of this job with Day 1 input as described in "J18_Daily_UpdateStoreDim (Day 1) execution" on page 484.

J18_Daily_UpdateStoreDim (Day 1) configuration

Figure 3-507 on page 479 shows the various stages in the job — it includes a Data Set stage, a Sequential File stage, a Transformer stage, and a ODBCConnectorPX stage. The names of the stages were modified as shown:

1. Figure 3-508 on page 480 through Figure 3-514 on page 483 describe the configuration of the Trx_StoreDim Transformer stage that processes the dimension update records data set created in the "J17_DailyCreateSalesFactDS (Day1) execution" on page 475 job and directs appropriate rows to two output links.
   - Figure 3-508 on page 480 shows the Trx_StoreDim Transformer stage with a constraint that directs rows to the Odbc_StoreDim or Rej_StoreDim links. All the columns are mapped to each output link as shown in Figure 3-510 on page 481 and Figure 3-511 on page 481 respectively.

- Figure 3-509 on page 481 shows the Trx_StoreDim Transformer Stage Constraints window that defines the constraint that directs the rows to the appropriate output link. Briefly, the constraint specifies that records that have a NULL in the CITY_POPULATION or STATE_POPULATION columns should be directed to the Rej_StoreDim output link, and those records that evaluate the predicate to false are directed to the Odbc_StoreDim link.

2. Figure 3-512 on page 482 through Figure 3-514 on page 483 show the configuration of the Odbc_StoreDim ODBCConnectorPX stage that updates the STORE_DIM table which is the reference link.

   - Figure 3-512 on page 482 identifies the Connection details, the Write mode (Update then Insert), and manually generated SQL.

   - Figure 3-513 on page 483 shows the manually generated SQL UPDATE statement, while Figure 3-514 on page 483 shows the manually generated SQL INSERT statement.

The results of the execution of this job on Day 1 are described in "J18_Daily_UpdateStoreDim (Day 1) execution" on page 484.

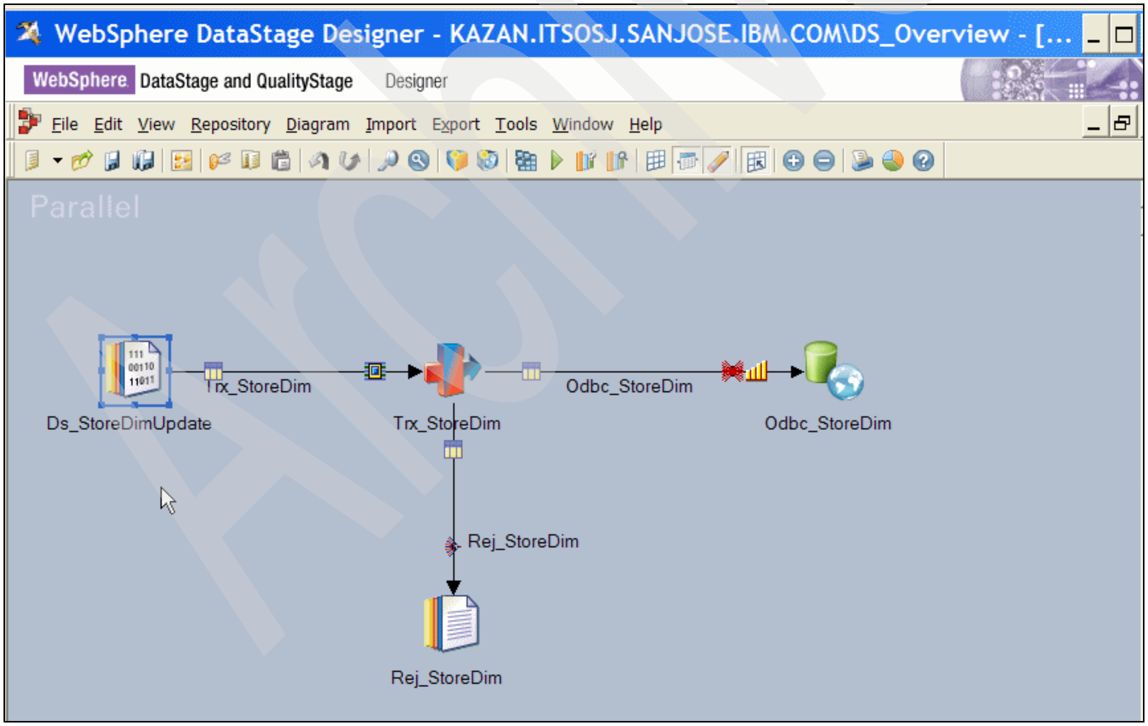


Figure 3-507 Create the J18_Daily_UpdateStoreDim job 1/8

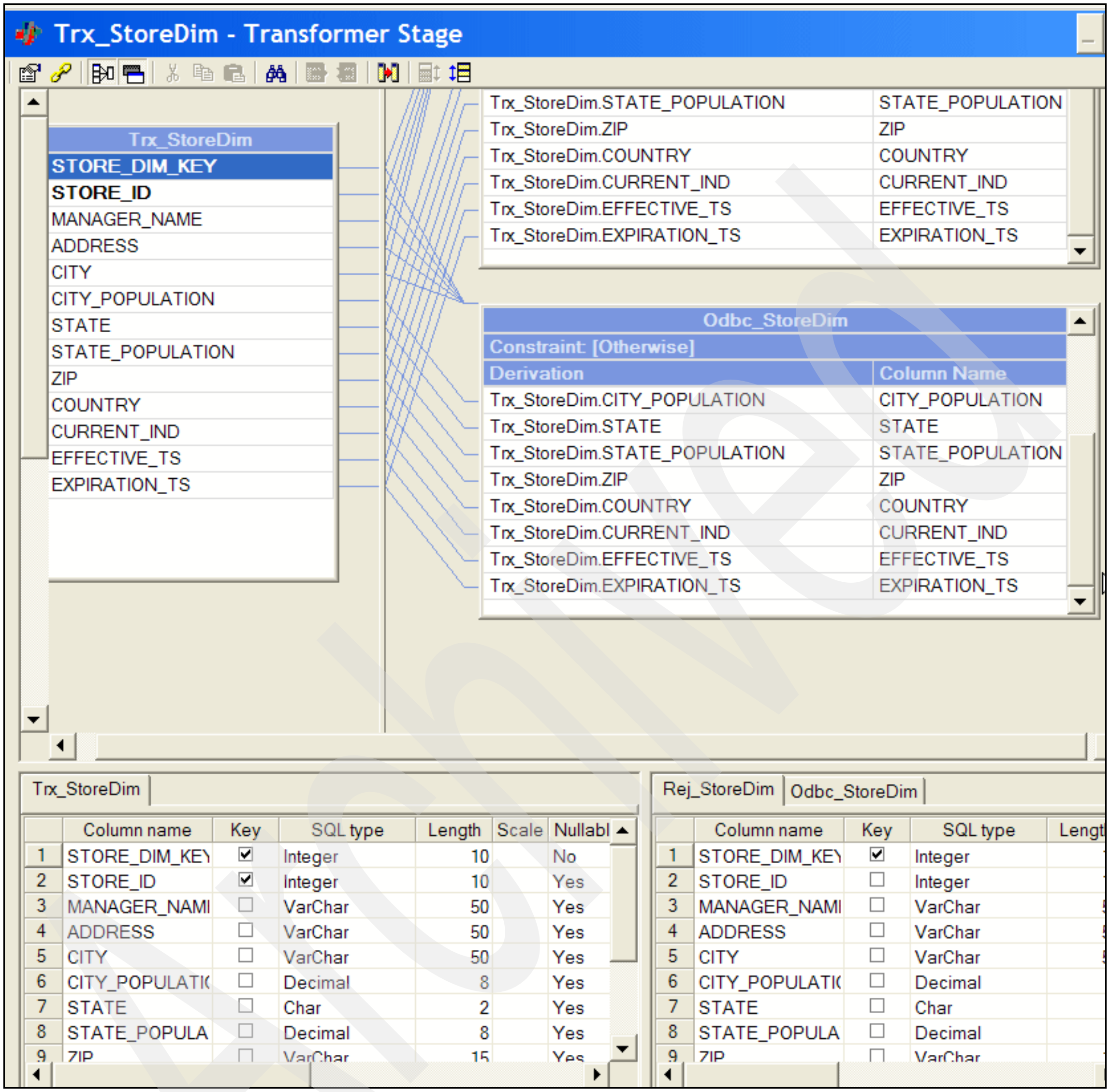


Figure 3-508 Create the J18_Daily_UpdateStoreDim job 2/8

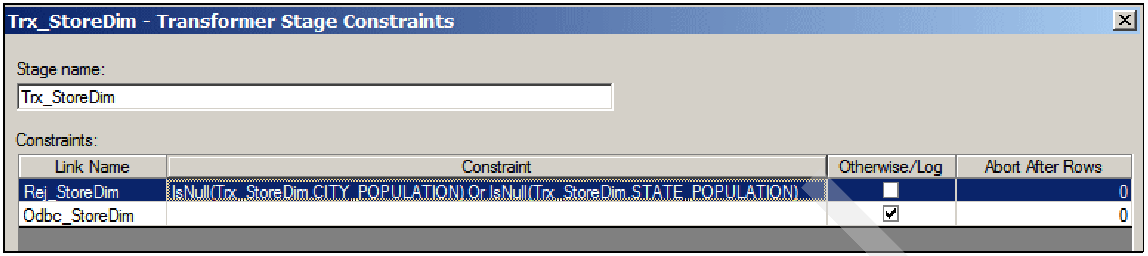


Figure 3-509 Create the J18_Daily_UpdateStoreDim job 3/8

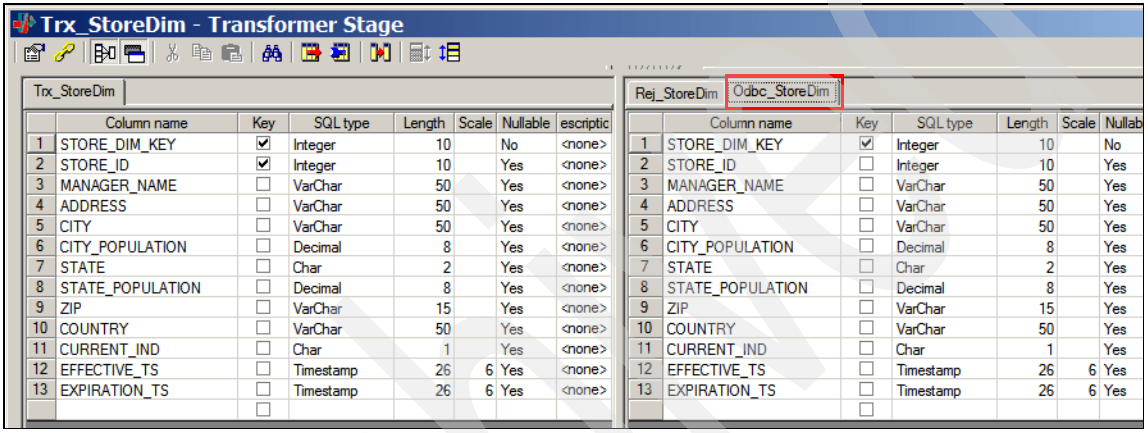


Figure 3-510 Create the J18_Daily_UpdateStoreDim job 4/8

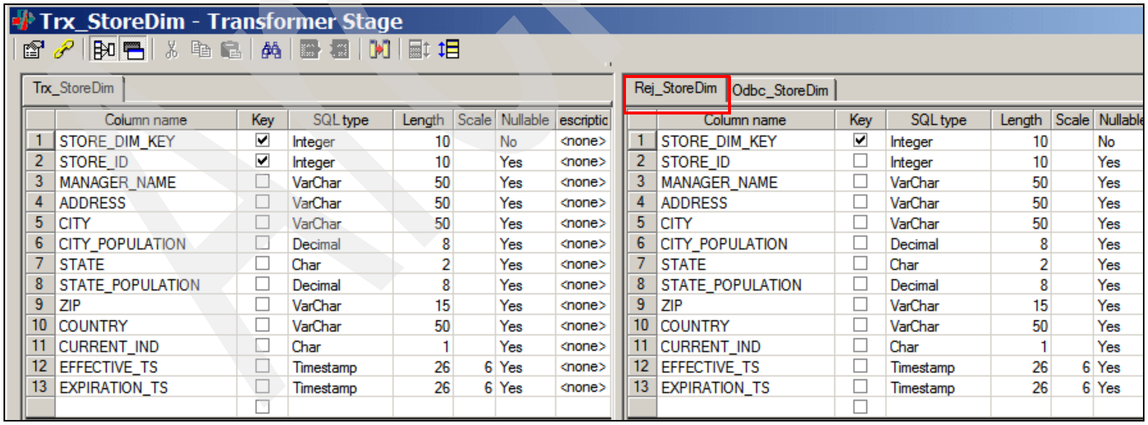


Figure 3-511 Create the J18_Daily_UpdateStoreDim job 5/8

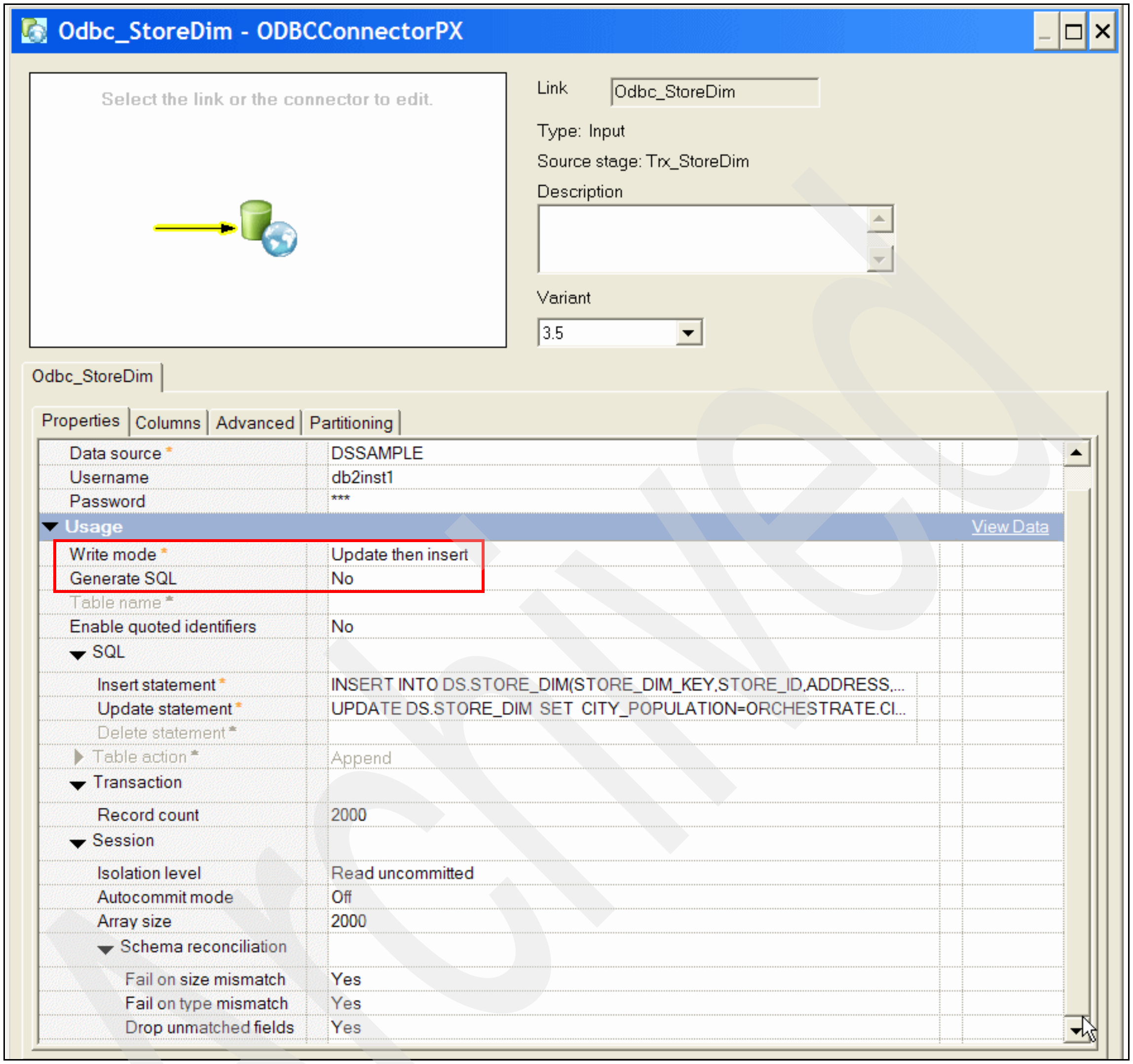


Figure 3-512 Create the J18_Daily_UpdateStoreDim job 6/8

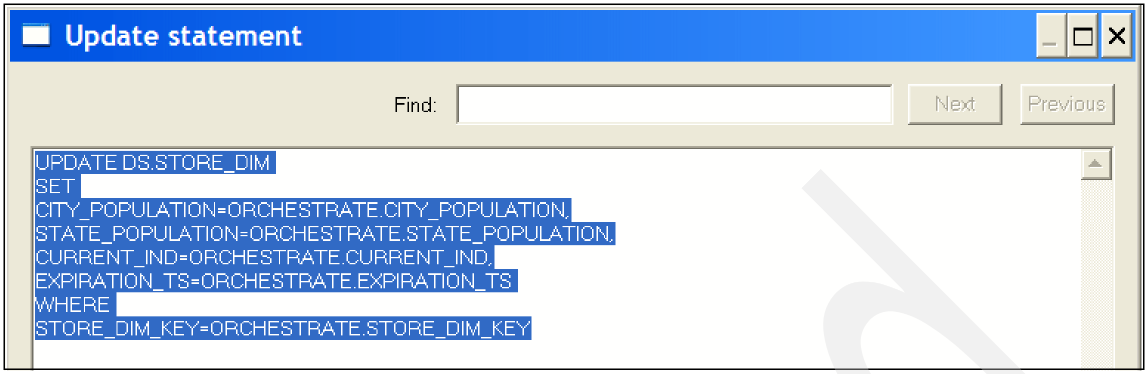


Figure 3-513 Create the J18_Daily_UpdateStoreDim job 7/8

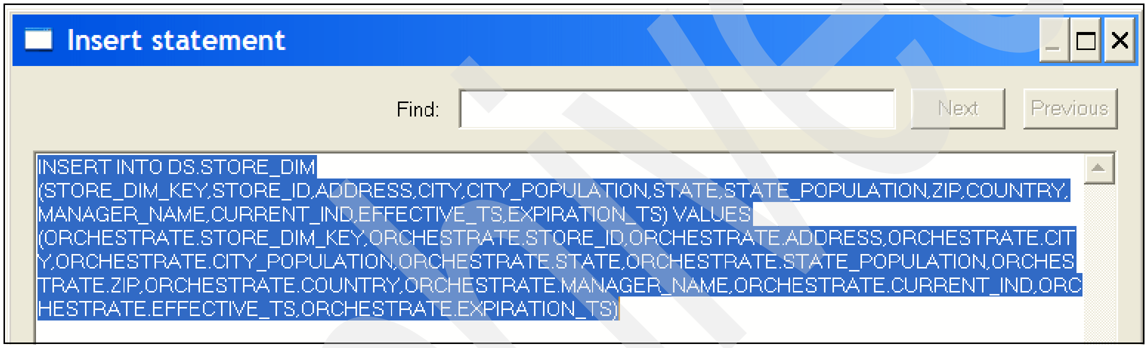


Figure 3-514 Create the J18_Daily_UpdateStoreDim job 8/8

J18_Daily_UpdateStoreDim (Day 1) execution

Figure 3-515 shows the results of the execution of this job with Day 1 data described earlier.

It shows no input records to update the Store dimension tables.

The next step is to execute the job described in “J19_Daily_UpdateCustomerDim (Day 1)” on page 485 job.

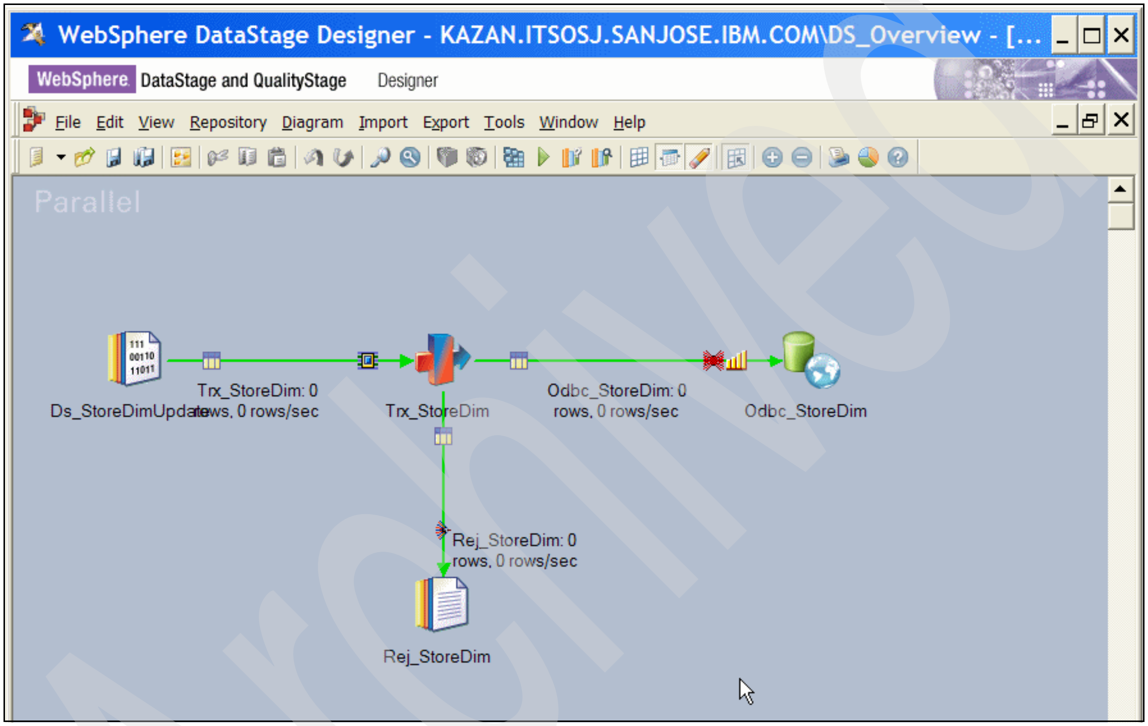


Figure 3-515 Execute the J18_Daily_UpdateStoreDim job (Day 1)

J19_Daily_UpdateCustomerDim (Day 1)

This job updates the Customer dimension table using the file created in the “J17_DailyCreateSalesFactDS (Day1)” on page 433 job similar to the process described in “J18_Daily_UpdateStoreDim (Day 1)” on page 478.

Figure 3-516 on page 485 through Figure 3-524 on page 491 explain the main stages in this job and the configuration of these stages as described in “J19_Daily_UpdateCustomerDim (Day 1) configuration” on page 485, while Figure 3-525 on page 493 through Figure 3-528 on page 494 explain the execution of this job with Day 1 input as described in “J19_Daily_UpdateCustomerDim (Day 1) execution” on page 492.

J19_Daily_UpdateCustomerDim (Day 1) configuration

Since this configuration is very similar to that described in “J18_Daily_UpdateStoreDim (Day 1) configuration” on page 478, it is not repeated here.

The results of the execution of this job on Day 1 are described in “J19_Daily_UpdateCustomerDim (Day 1) execution” on page 492.

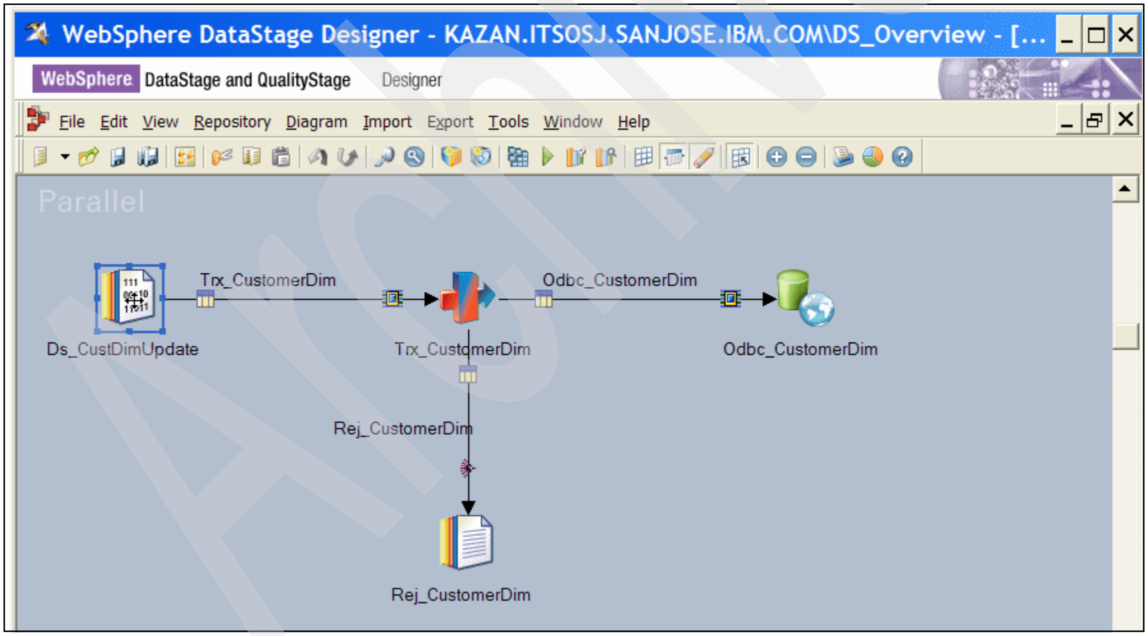


Figure 3-516 Create the J19_Daily_UpdateCustomerDim job 1/9

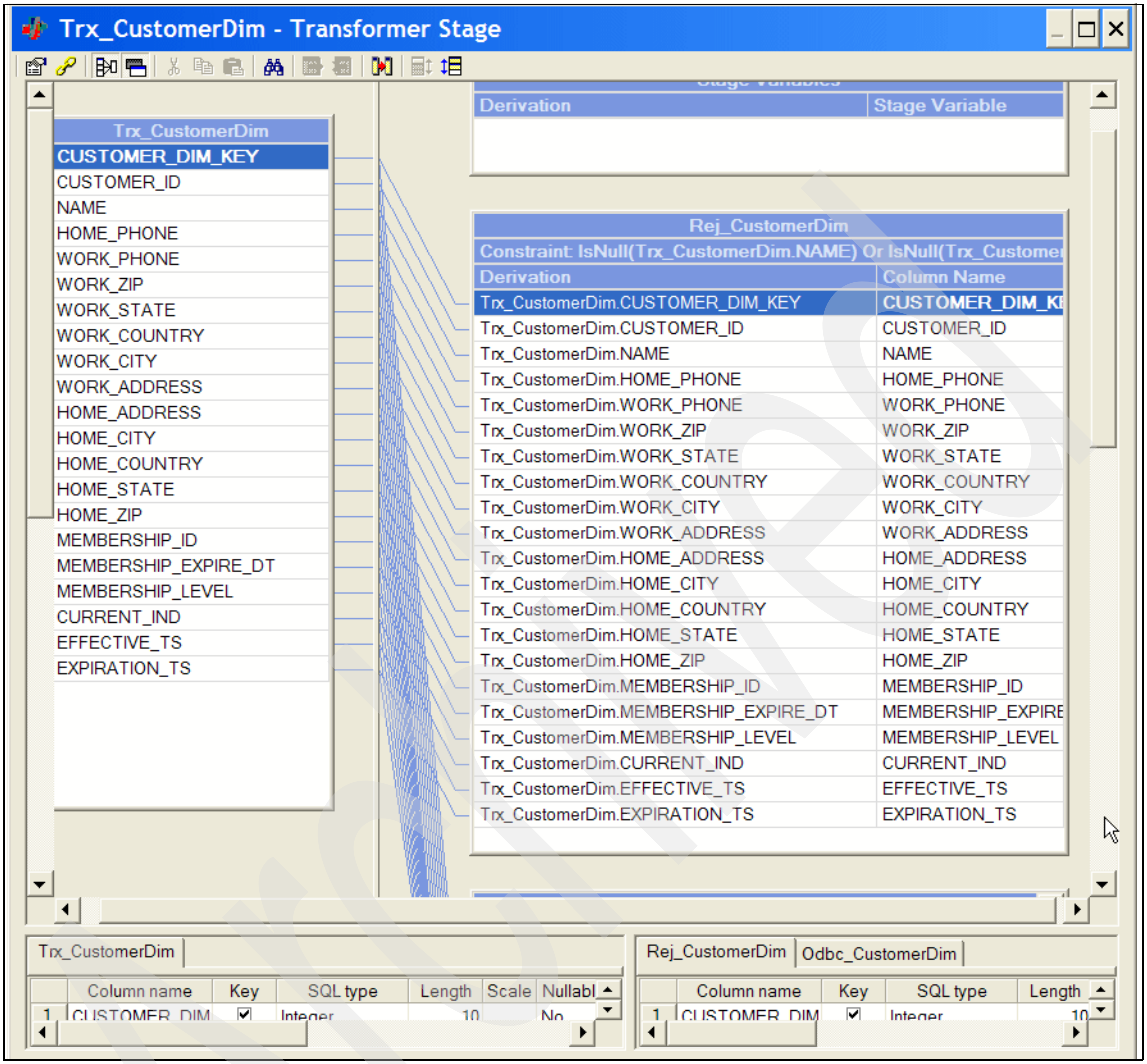


Figure 3-517 Create the J19_Daily_UpdateCustomerDim job 2/9

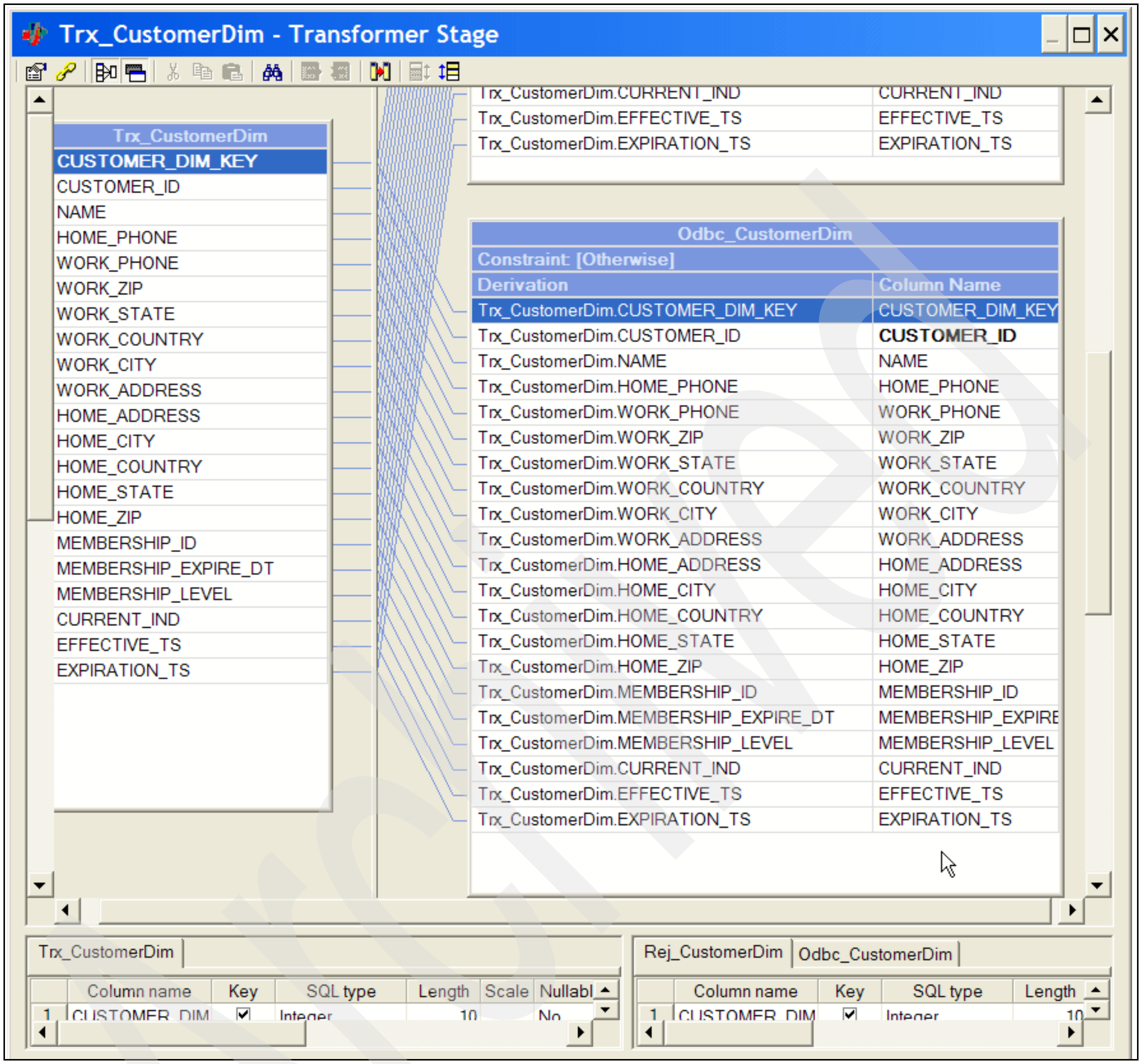
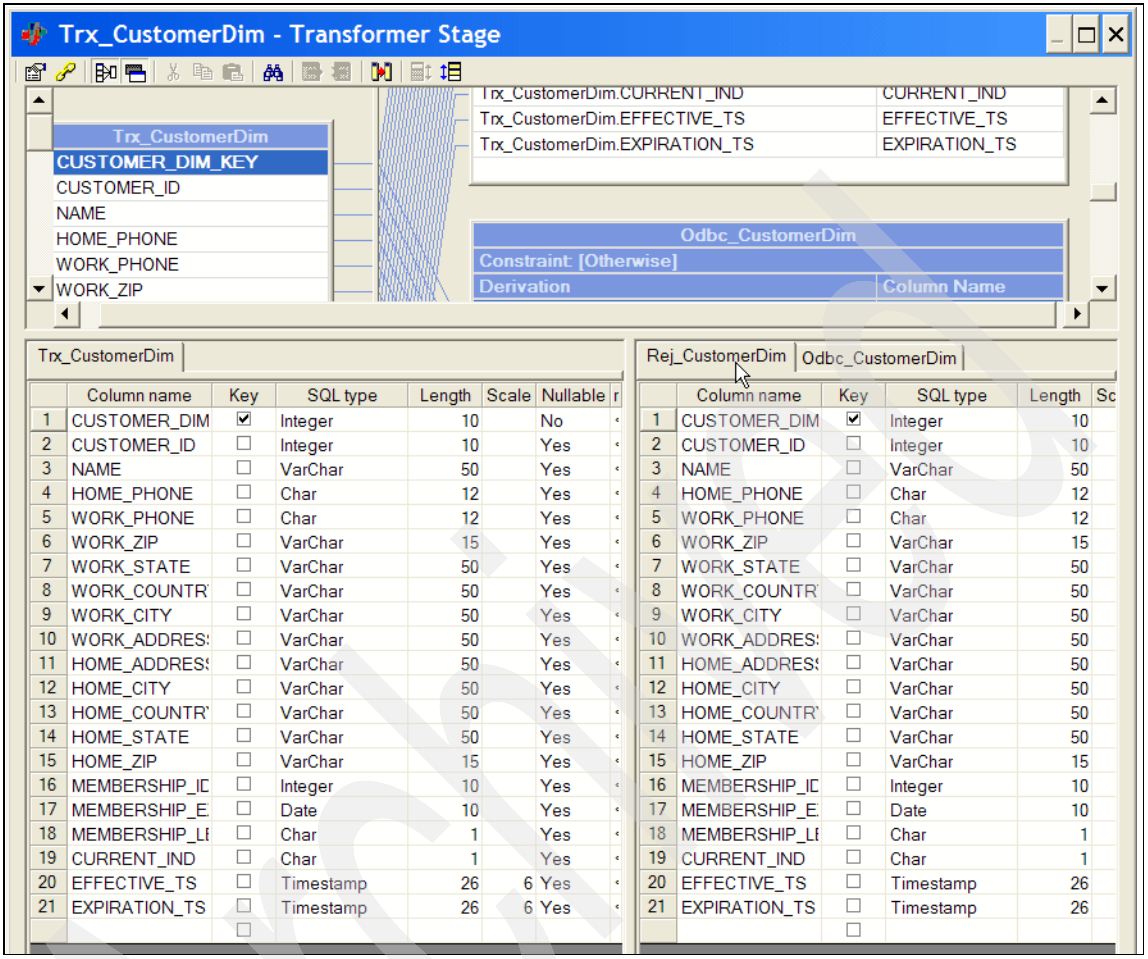


Figure 3-518 Create the J19_Daily_UpdateCustomerDim job 3/9



	Column name	Key	SQL type	Length	Scale	Nullable
1	CUSTOMER_DIM	☑	Integer	10		No
2	CUSTOMER_ID	☐	Integer	10		Yes
3	NAME	☐	VarChar	50		Yes
4	HOME_PHONE	☐	Char	12		Yes
5	WORK_PHONE	☐	Char	12		Yes
6	WORK_ZIP	☐	VarChar	15		Yes
7	WORK_STATE	☐	VarChar	50		Yes
8	WORK_COUNTR	☐	VarChar	50		Yes
9	WORK_CITY	☐	VarChar	50		Yes
10	WORK_ADDRES	☐	VarChar	50		Yes
11	HOME_ADDRES	☐	VarChar	50		Yes
12	HOME_CITY	☐	VarChar	50		Yes
13	HOME_COUNTR	☐	VarChar	50		Yes
14	HOME_STATE	☐	VarChar	50		Yes
15	HOME_ZIP	☐	VarChar	15		Yes
16	MEMBERSHIP_IC	☐	Integer	10		Yes
17	MEMBERSHIP_E	☐	Date	10		Yes
18	MEMBERSHIP_LI	☐	Char	1		Yes
19	CURRENT_IND	☐	Char	1		Yes
20	EFFECTIVE_TS	☐	Timestamp	26	6	Yes
21	EXPIRATION_TS	☐	Timestamp	26	6	Yes

	Column name	Key	SQL type	Length
1	CUSTOMER_DIM	☑	Integer	10
2	CUSTOMER_ID	☐	Integer	10
3	NAME	☐	VarChar	50
4	HOME_PHONE	☐	Char	12
5	WORK_PHONE	☐	Char	12
6	WORK_ZIP	☐	VarChar	15
7	WORK_STATE	☐	VarChar	50
8	WORK_COUNTR	☐	VarChar	50
9	WORK_CITY	☐	VarChar	50
10	WORK_ADDRES	☐	VarChar	50
11	HOME_ADDRES	☐	VarChar	50
12	HOME_CITY	☐	VarChar	50
13	HOME_COUNTR	☐	VarChar	50
14	HOME_STATE	☐	VarChar	50
15	HOME_ZIP	☐	VarChar	15
16	MEMBERSHIP_IC	☐	Integer	10
17	MEMBERSHIP_E	☐	Date	10
18	MEMBERSHIP_LI	☐	Char	1
19	CURRENT_IND	☐	Char	1
20	EFFECTIVE_TS	☐	Timestamp	26
21	EXPIRATION_TS	☐	Timestamp	26

Figure 3-519 Create the J19_Daily_UpdateCustomerDim job 4/9

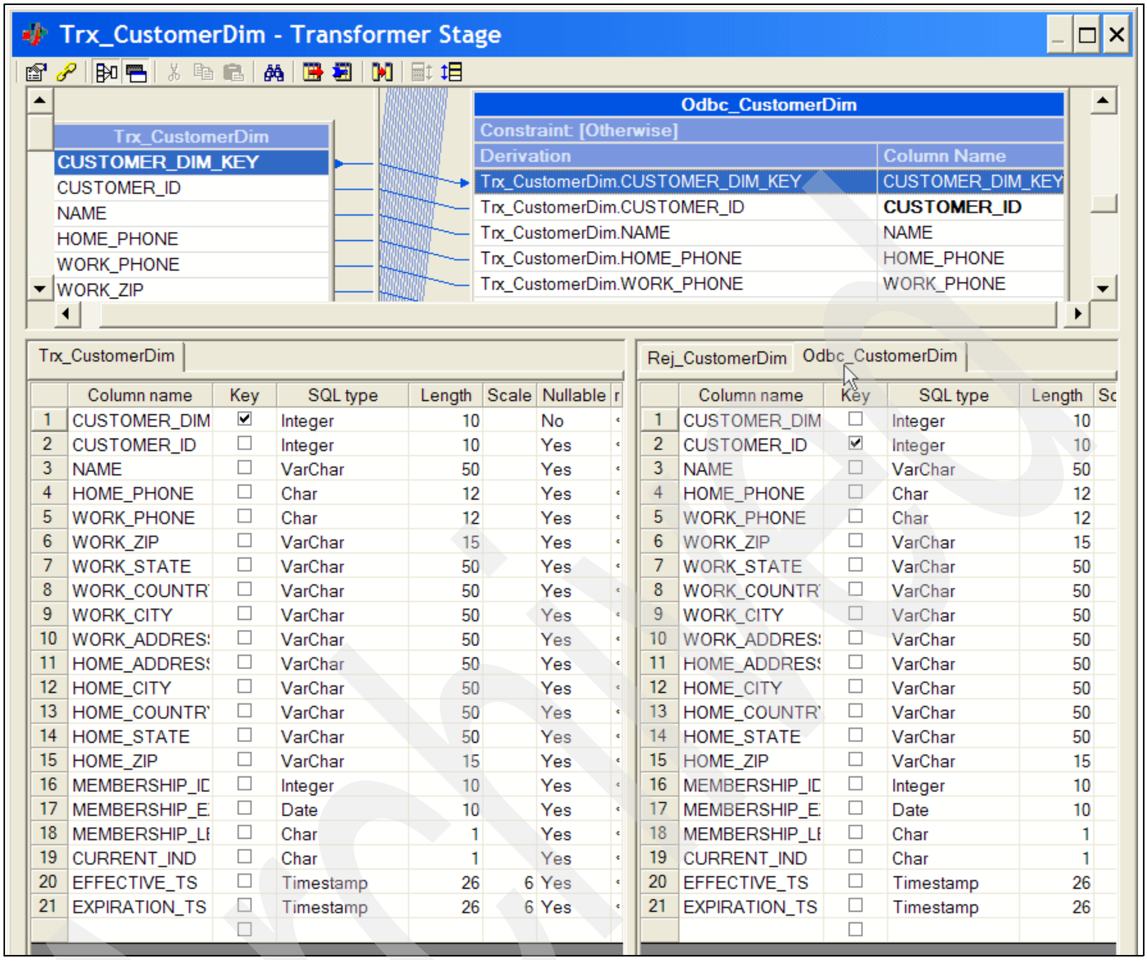


Figure 3-520 Create the J19_Daily_UpdateCustomerDim job 5/9

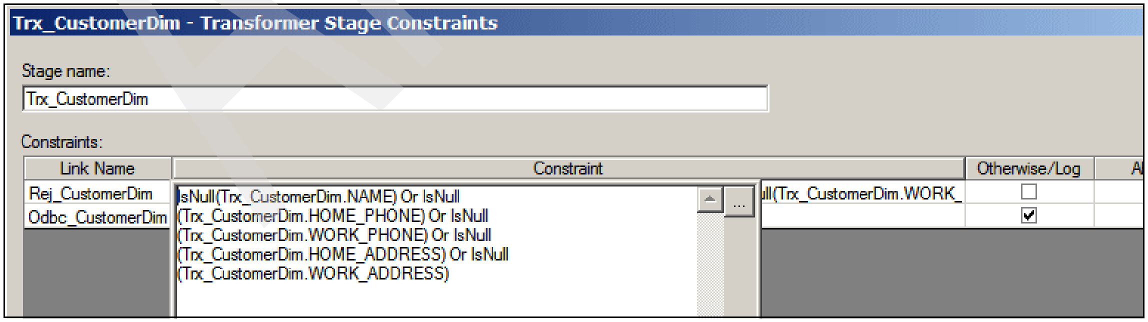


Figure 3-521 Create the J19_Daily_UpdateCustomerDim job 6/9

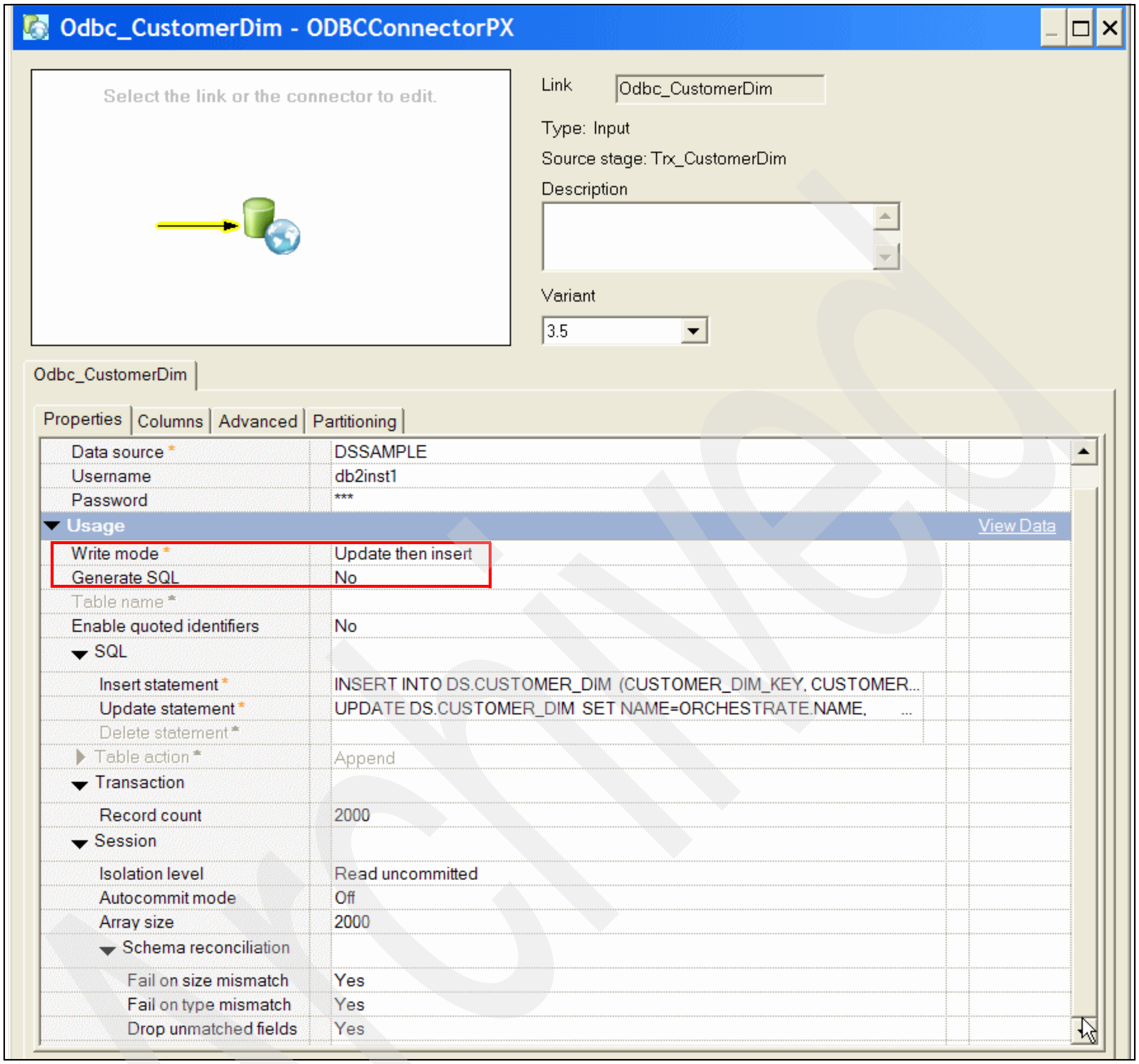


Figure 3-522 Create the J19_Daily_UpdateCustomerDim job 7/9

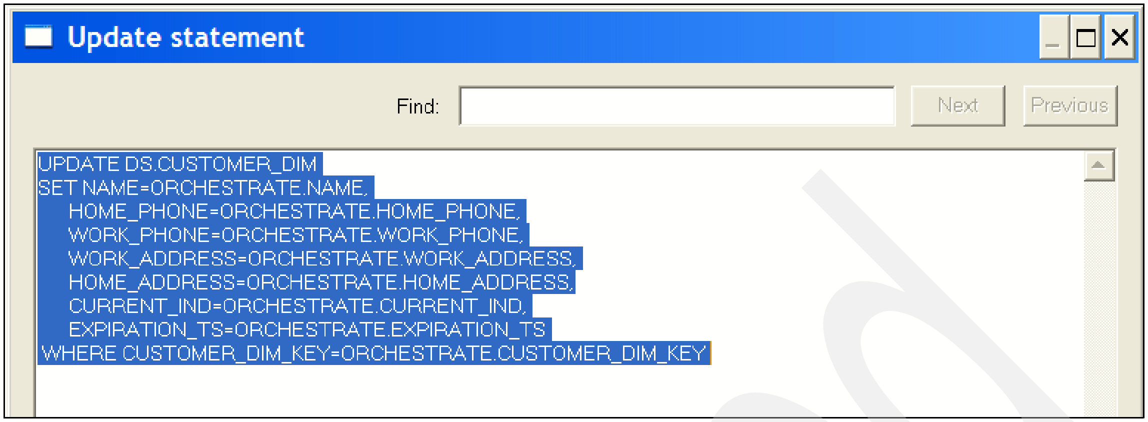


Figure 3-523 Create the J19_Daily_UpdateCustomerDim job 8/9

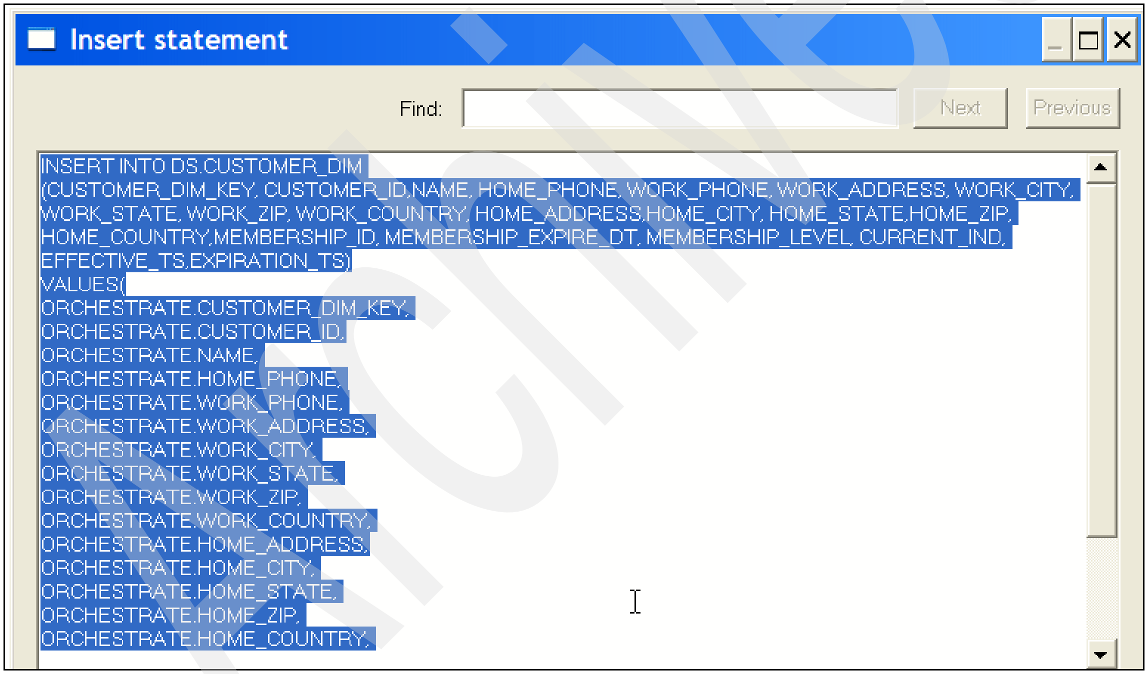


Figure 3-524 Create the J19_Daily_UpdateCustomerDim job 9/9

J19_Daily_UpdateCustomerDim (Day 1) execution

Figure 3-525 on page 493 through Figure 3-528 on page 494 show the results of the execution of this job with Day 1 data described earlier.

- Figure 3-525 on page 493 shows the results of the execution. It accepts 1 row as input from the "J17_DailyCreateSalesFactDS (Day1) execution" on page 475 job as seen in Figure 3-500 on page 476 through Figure 3-502 on page 476.
- The outputs are as follows:
  - There are no rows written to the Rej_CustomerDim link.
  - The 1 row written to the Odbc_CustomerDim link updates the CUSTOMER_DIM dimension table with these changes (as highlighted) as seen in Figure 3-526 on page 493 through Figure 3-528 on page 494.

> **Note:** CUSTOMER_ID 7 still exists in the Customer dimension table because the SCD stage does not support a delete operation. The general concept here is that there will usually be some records in the fact table for every business key in the dimension tables. Therefore, deleting a business key in the dimension table will affect queries interested in looking at reports in an earlier time interval, ignoring for the moment, potential referential integrity violations that would occur with such a delete operation. If you still want to go ahead and delete a business key in a dimension table, you should first delete all the entries referencing this business key in the fact table and then delete the business key in the dimension table.

The next step is to execute the job described in "J20_Daily_UpdateProductDim (Day 1)" on page 494.

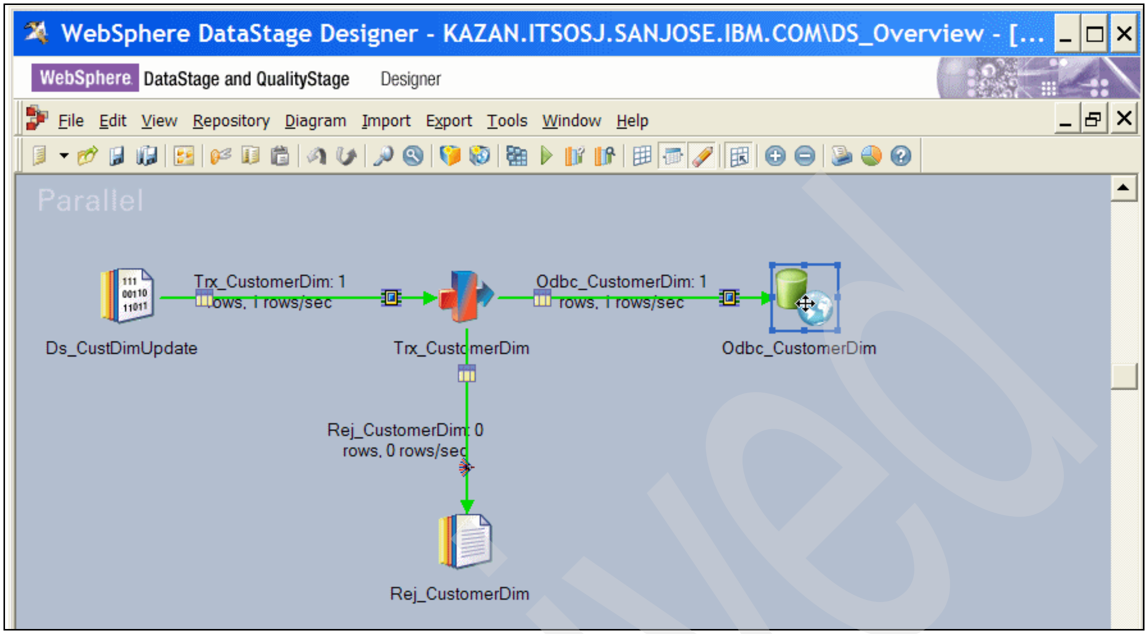
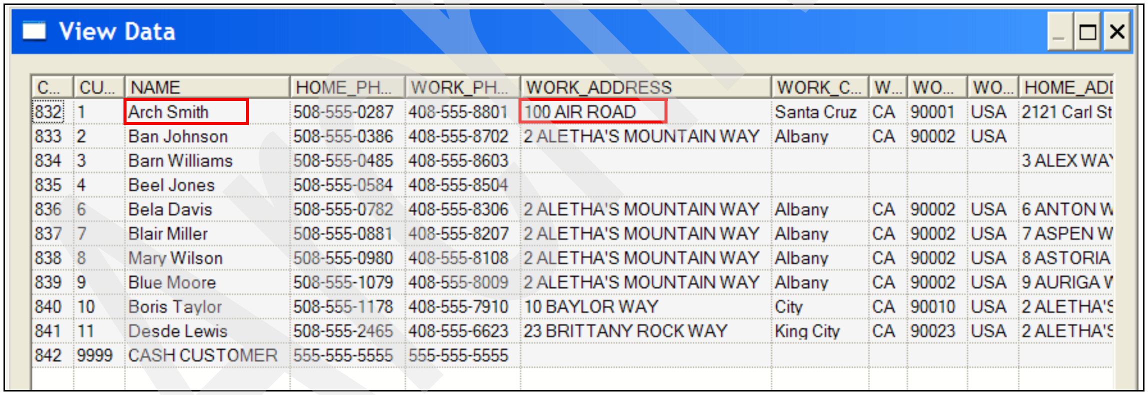
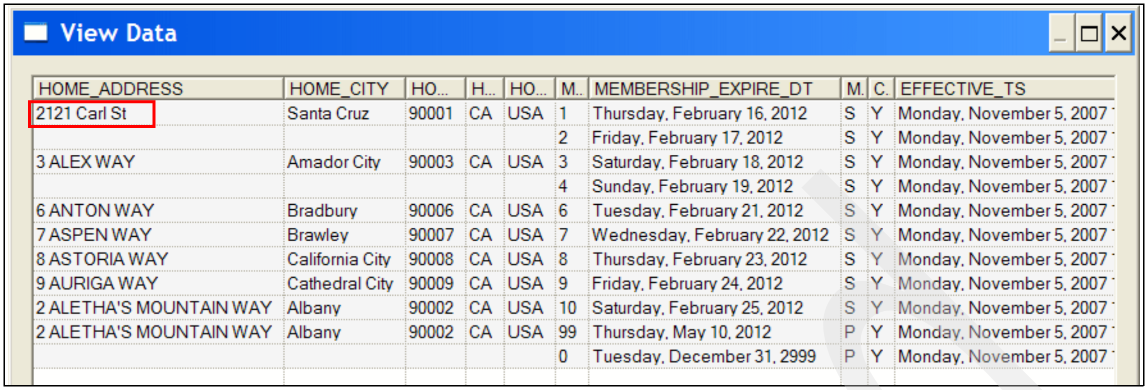


Figure 3-525 Execute the J19_Daily_UpdateCustomerDim job (Day 1) 1/4


View Data

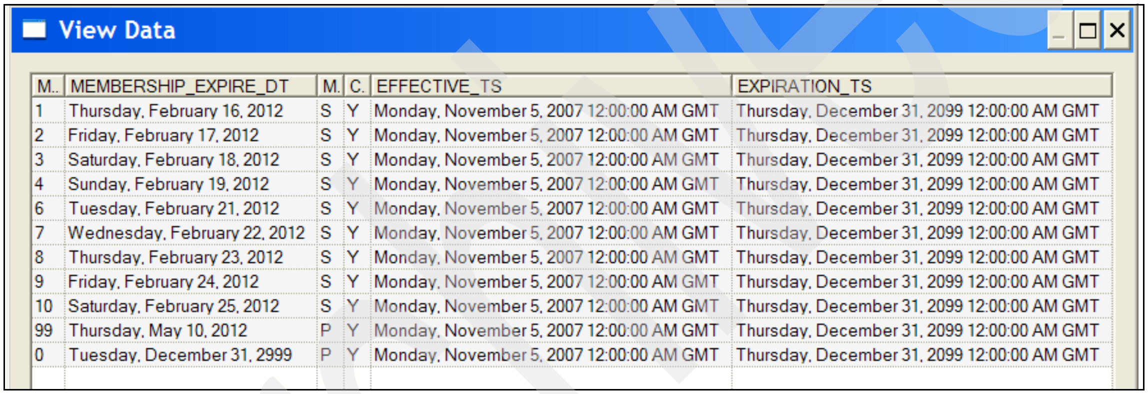
C...	CU...	NAME	HOME_PH...	WORK_PH...	WORK_ADDRESS	WORK_C...	W...	WO...	WO...	HOME_ADI
832	1	Arch Smith	508-555-0287	408-555-8801	100 AIR ROAD	Santa Cruz	CA	90001	USA	2121 Carl St
833	2	Ban Johnson	508-555-0386	408-555-8702	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	
834	3	Barn Williams	508-555-0485	408-555-8603						3 ALEX WA'
835	4	Beel Jones	508-555-0584	408-555-8504						
836	6	Bela Davis	508-555-0782	408-555-8306	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	6 ANTON W
837	7	Blair Miller	508-555-0881	408-555-8207	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	7 ASPEN W
838	8	Mary Wilson	508-555-0980	408-555-8108	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	8 ASTORIA
839	9	Blue Moore	508-555-1079	408-555-8009	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	9 AURIGA V
840	10	Boris Taylor	508-555-1178	408-555-7910	10 BAYLOR WAY	City	CA	90010	USA	2 ALETHA'S
841	11	Desde Lewis	508-555-2465	408-555-6623	23 BRITTANY ROCK WAY	King City	CA	90023	USA	2 ALETHA'S
842	9999	CASH CUSTOMER	555-555-5555	555-555-5555						

Figure 3-526 Execute the J19_Daily_UpdateCustomerDim job (Day 1) 2/4



HOME_ADDRESS	HOME_CITY	HO...	H...	HO...	M..	MEMBERSHIP_EXPIRE_DT	M.	C.	EFFECTIVE_TS
2121 Carl St	Santa Cruz	90001	CA	USA	1	Thursday, February 16, 2012	S	Y	Monday, November 5, 2007
					2	Friday, February 17, 2012	S	Y	Monday, November 5, 2007
3 ALEX WAY	Amador City	90003	CA	USA	3	Saturday, February 18, 2012	S	Y	Monday, November 5, 2007
					4	Sunday, February 19, 2012	S	Y	Monday, November 5, 2007
6 ANTON WAY	Bradbury	90006	CA	USA	6	Tuesday, February 21, 2012	S	Y	Monday, November 5, 2007
7 ASPEN WAY	Brawley	90007	CA	USA	7	Wednesday, February 22, 2012	S	Y	Monday, November 5, 2007
8 ASTORIA WAY	California City	90008	CA	USA	8	Thursday, February 23, 2012	S	Y	Monday, November 5, 2007
9 AURIGA WAY	Cathedral City	90009	CA	USA	9	Friday, February 24, 2012	S	Y	Monday, November 5, 2007
2 ALETHA'S MOUNTAIN WAY	Albany	90002	CA	USA	10	Saturday, February 25, 2012	S	Y	Monday, November 5, 2007
2 ALETHA'S MOUNTAIN WAY	Albany	90002	CA	USA	99	Thursday, May 10, 2012	P	Y	Monday, November 5, 2007
					0	Tuesday, December 31, 2999	P	Y	Monday, November 5, 2007

Figure 3-527 Execute the J19_Daily_UpdateCustomerDim job (Day 1) 3/4



M..	MEMBERSHIP_EXPIRE_DT	M.	C.	EFFECTIVE_TS	EXPIRATION_TS
1	Thursday, February 16, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
2	Friday, February 17, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
3	Saturday, February 18, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
4	Sunday, February 19, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
6	Tuesday, February 21, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
7	Wednesday, February 22, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
8	Thursday, February 23, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
9	Friday, February 24, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
10	Saturday, February 25, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
99	Thursday, May 10, 2012	P	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
0	Tuesday, December 31, 2999	P	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT

Figure 3-528 Execute the J19_Daily_UpdateCustomerDim job (Day 1) 4/4

J20_Daily_UpdateProductDim (Day 1)

This job updates the Product dimension table using the data set created in the "J17_DailyCreateSalesFactDS (Day1)" on page 433 job. However, there are no Type 1 attribute changes for the Product dimension table, and therefore no requirement to introduce a Transformer stage as in the case of the process described in "J18_Daily_UpdateStoreDim (Day 1)" on page 478.

Figure 3-529 on page 495 through Figure 3-531 on page 497 explain the main stages in this job and the configuration of these stages as described in "J20_Daily_UpdateProductDim (Day 1) configuration" on page 495, while Figure 3-532 on page 498 explains the execution of this job with Day 1 input, as described in "J20_Daily_UpdateProductDim (Day 1) execution" on page 498.

J20_Daily_UpdateProductDim (Day 1) configuration

Figure 3-529 shows the various stages in the job — it includes a Data Set stage and a ODBCConnectorPX stage. The names of the stages were modified as shown.

Figure 3-530 on page 496 and Figure 3-531 on page 497 show the configuration of the Odbc_ProductDim ODBCConnectorPX stage that inserts a row into PRODUCT_DIM table which is the reference link. There is no update requirement since this table has no Type 1 attributes defined.

- Figure 3-530 on page 496 identifies the Connection details, the Write mode (Insert), and manually generated SQL.
- Figure 3-531 on page 497 shows the manually generated SQL INSERT statement.

The results of the execution of this job on Day 1 are described in "J20_Daily_UpdateProductDim (Day 1) execution" on page 498.

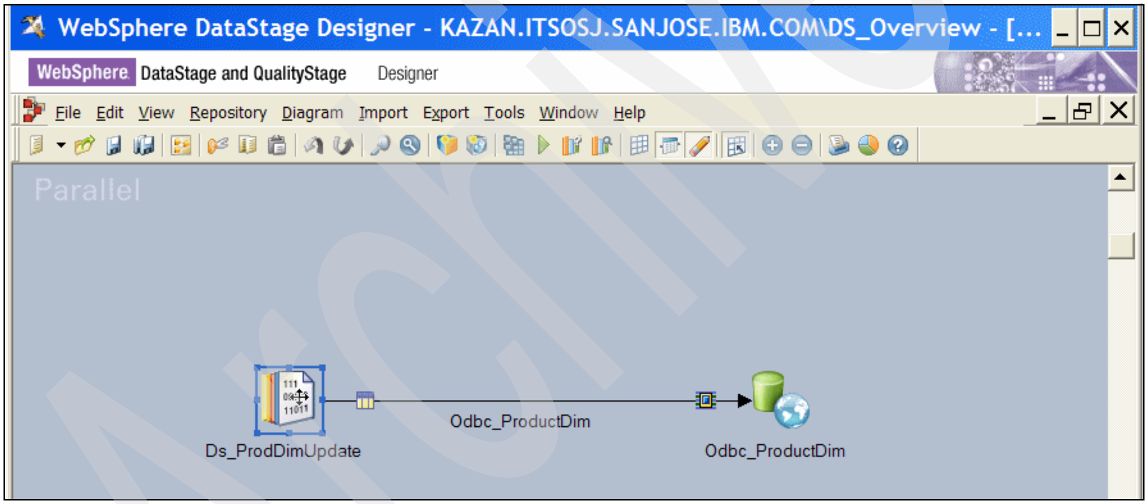


Figure 3-529 Create the J20_Daily_UpdateProductDim job 1/3

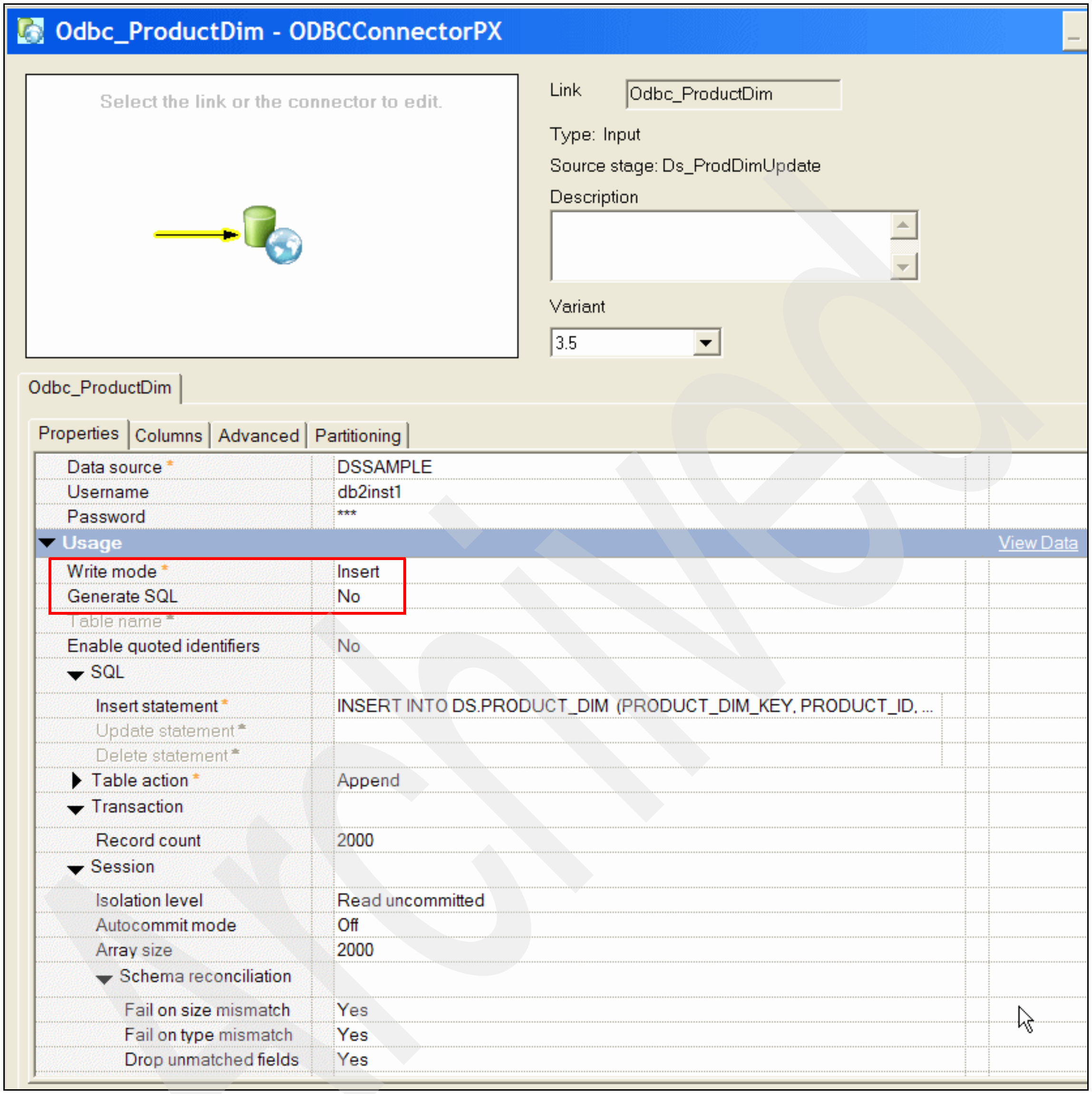


Figure 3-530 Create the J20_Daily_UpdateProductDim job 2/3

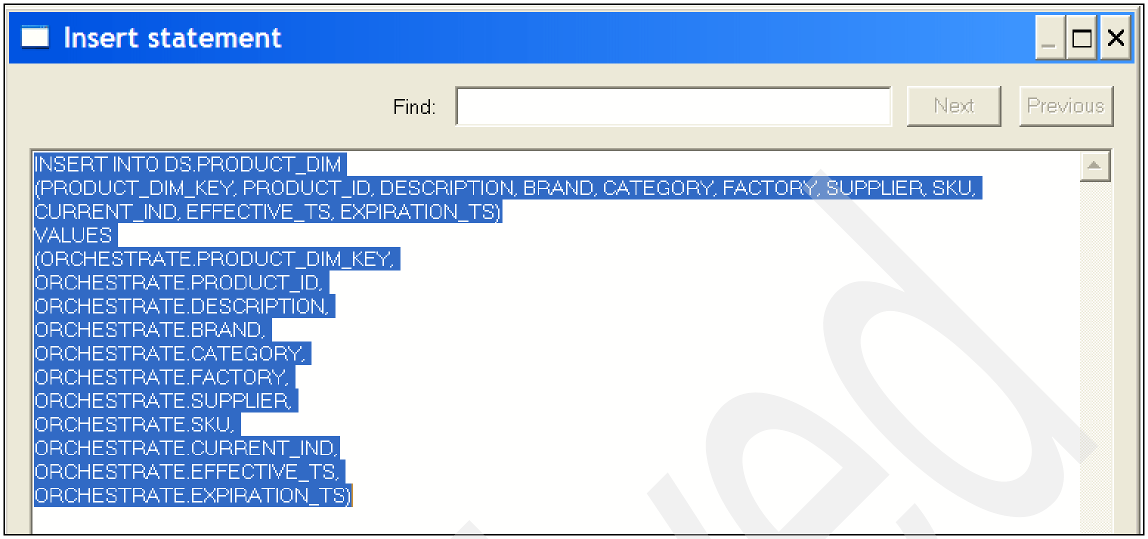


Figure 3-531 Create the J20_Daily_UpdateProductDim job 3/3

J20_Daily_UpdateProductDim (Day 1) execution

Figure 3-532 on page 498 shows the results of the execution of this job with the Day 1 data described earlier.

It shows no input records to update the Product dimension tables.

The next step is to execute the job described in “J21_Daily_UpdateDateDim (Day 1)” on page 499.

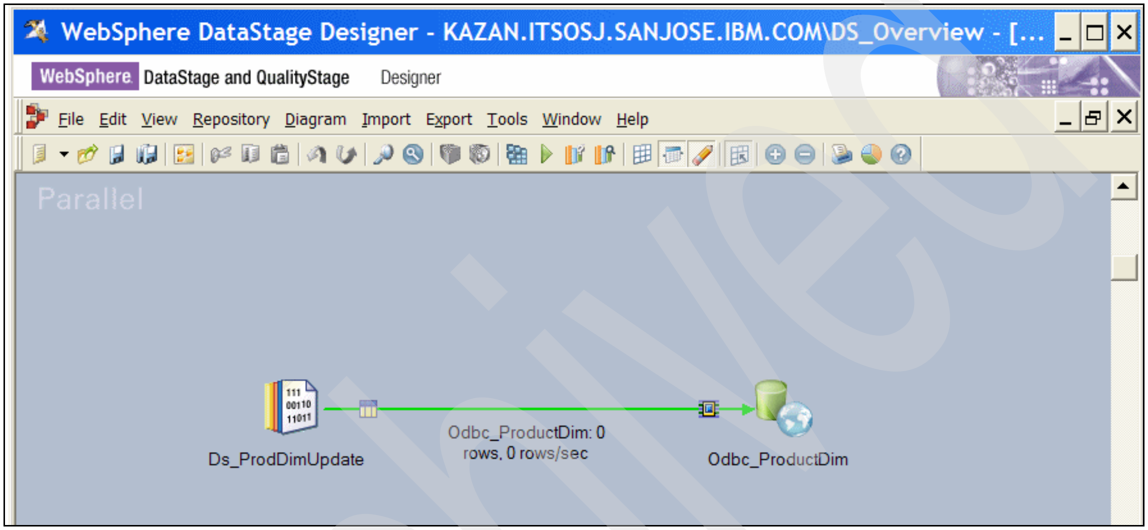


Figure 3-532 Execute the J20_Daily_UpdateProductDim job (Day 1)

J21_Daily_UpdateDateDim (Day 1)

This job updates the Date dimension table using the file created in the "J17_DailyCreateSalesFactDS (Day1)" on page 433 job similar to the process described in "J20_Daily_UpdateProductDim (Day 1)" on page 494.

Figure 3-533 on page 499 through Figure 3-535 on page 501 explain the main stages in this job and the configuration of these stages as described in "J21_Daily_UpdateDateDim (Day 1) configuration" on page 499, while Figure 3-536 on page 502 explains the execution of this job with Day 1 input as described in "J19_Daily_UpdateCustomerDim (Day 1) execution" on page 492.

J21_Daily_UpdateDateDim (Day 1) configuration

Since this configuration is very similar to that described in "J20_Daily_UpdateProductDim (Day 1) configuration" on page 495, it is not repeated here.

The results of the execution of this job on Day 1 are described in "J21_Daily_UpdateDateDim (Day 1) execution" on page 502.

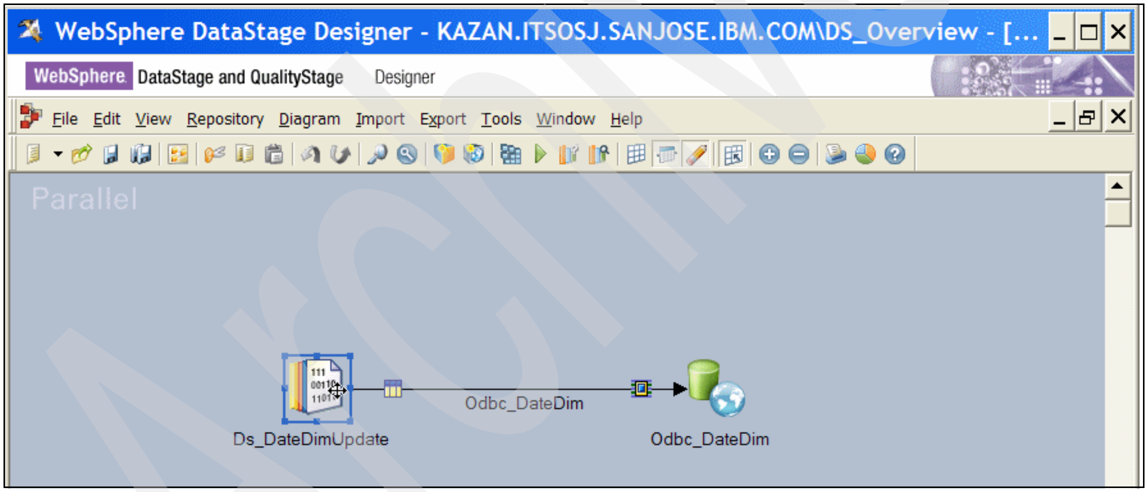


Figure 3-533 Create the J21_Daily_UpdateDateDim job 1/3

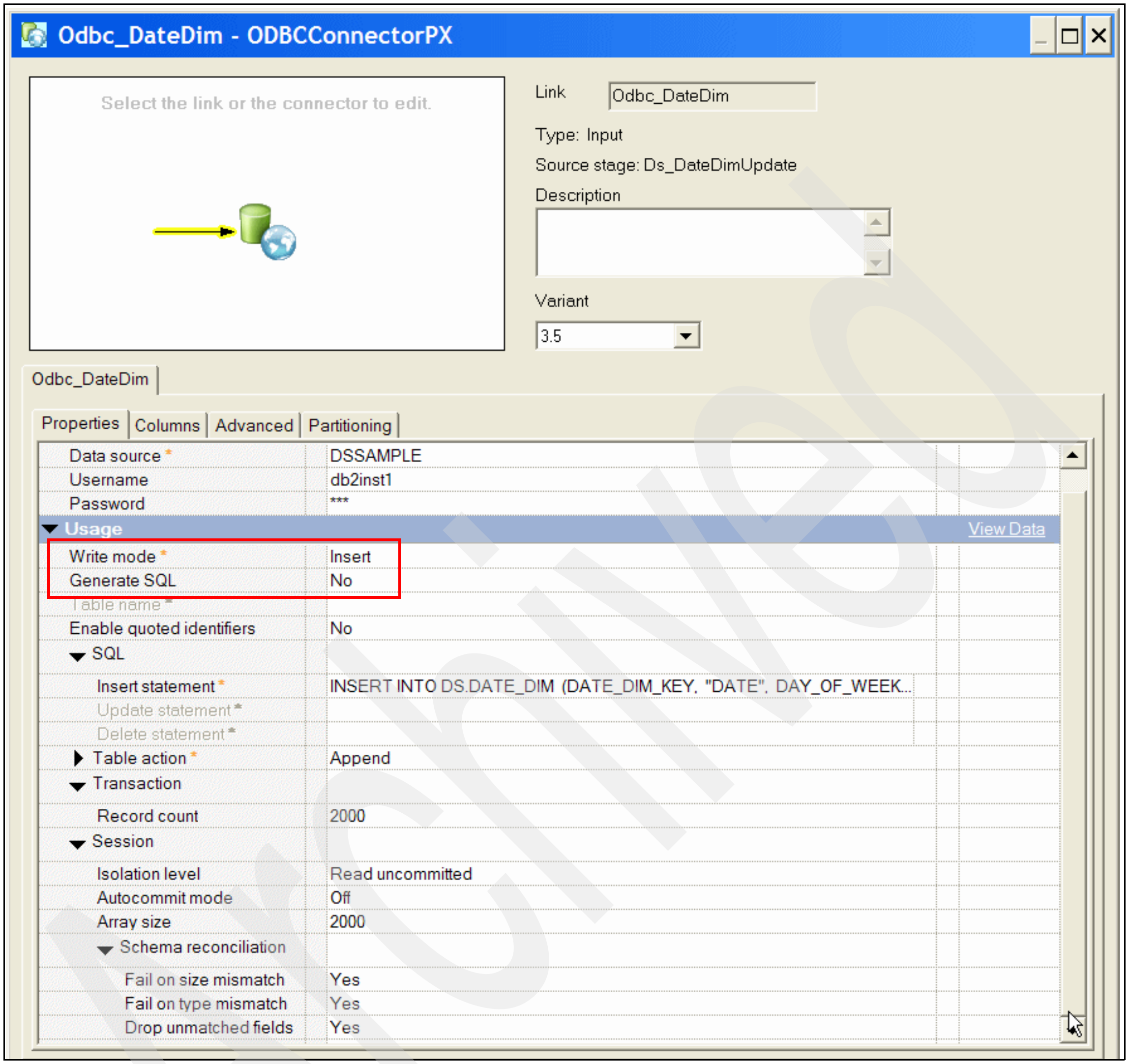


Figure 3-534 Create the J21_Daily_UpdateDateDim job 2/3

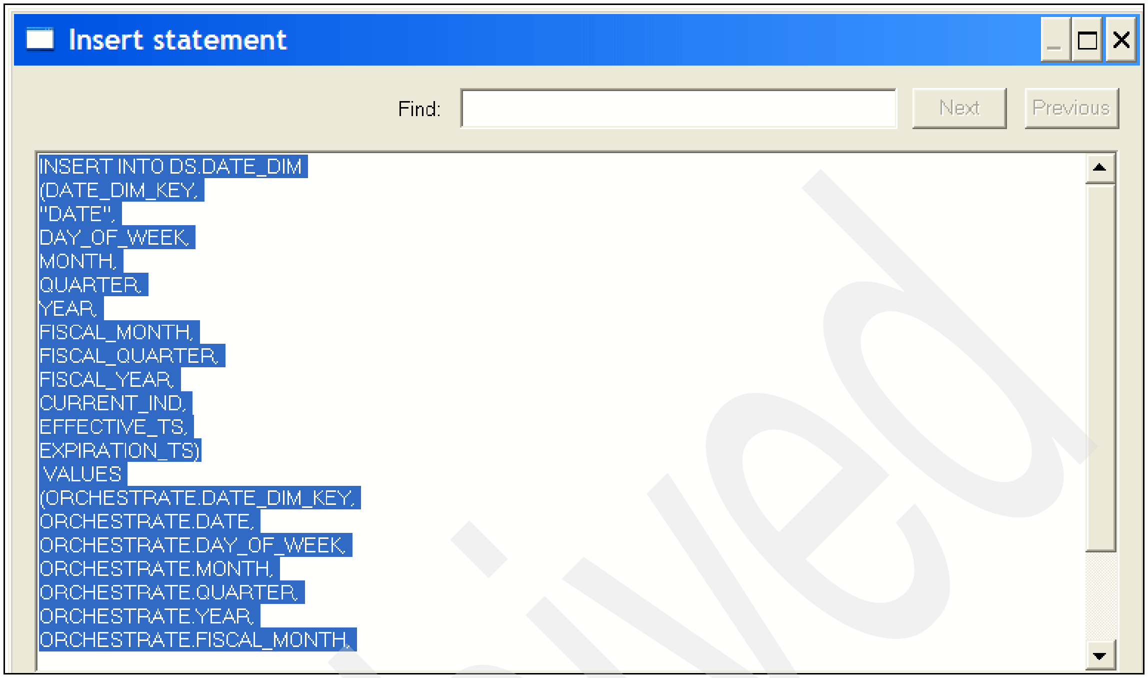


Figure 3-535 Create the J21_Daily_UpdateDateDim job 3/3

J21_Daily_UpdateDateDim (Day 1) execution

Figure 3-536 on page 502 shows the results of the execution of this job with Day 1 data described earlier.

It shows no input records to update the Date dimension tables.

The next step is to execute the job described in “J22_Daily_UpdateSalesFact (Day 1)” on page 502.

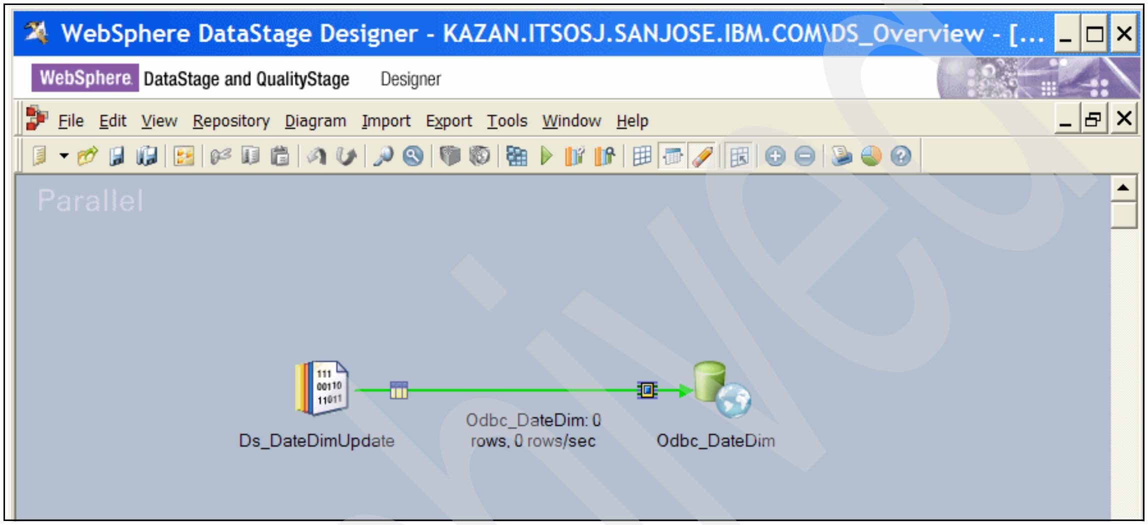


Figure 3-536 Execute the J21_Daily_UpdateDateDim job (Day 1)

J22_Daily_UpdateSalesFact (Day 1)

This job updates the Product dimension table using the data set created in the “J17_DailyCreateSalesFactDS (Day1)” on page 433 job. However, there are no Type 1 attribute changes for the Product dimension table, and therefore no requirement to introduce a Transformer stage as in the case of the process described in “J18_Daily_UpdateStoreDim (Day 1)” on page 478.

Figure 3-529 on page 495 through Figure 3-531 on page 497 explain the main stages in this job and the configuration of these stages as described in “J20_Daily_UpdateProductDim (Day 1) configuration” on page 495, while Figure 3-532 on page 498 explains the execution of this job with Day 1 input as described in “J20_Daily_UpdateProductDim (Day 1) execution” on page 498.

J22_Daily_UpdateSalesFact (Day 1) configuration

Figure 3-529 on page 495 shows the various stages in the job — it includes a Data Set stage and a ODBCConnectorPX stage. The names of the stages were modified as shown.

Figure 3-530 on page 496 and Figure 3-531 on page 497 show the configuration of the Odbc_ProductDim ODBCConnectorPX stage that inserts a row into PRODUCT_DIM table which is the reference link. There is no update requirement since this table has no Type 1 attributes defined.

- Figure 3-530 on page 496 identifies the Connection details, the Write mode (Insert), and manually generated SQL.
- Figure 3-531 on page 497 shows the manually generated SQL INSERT statement.

The results of the execution of this job on Day 1 are described in "J20_Daily_UpdateProductDim (Day 1) execution" on page 498.

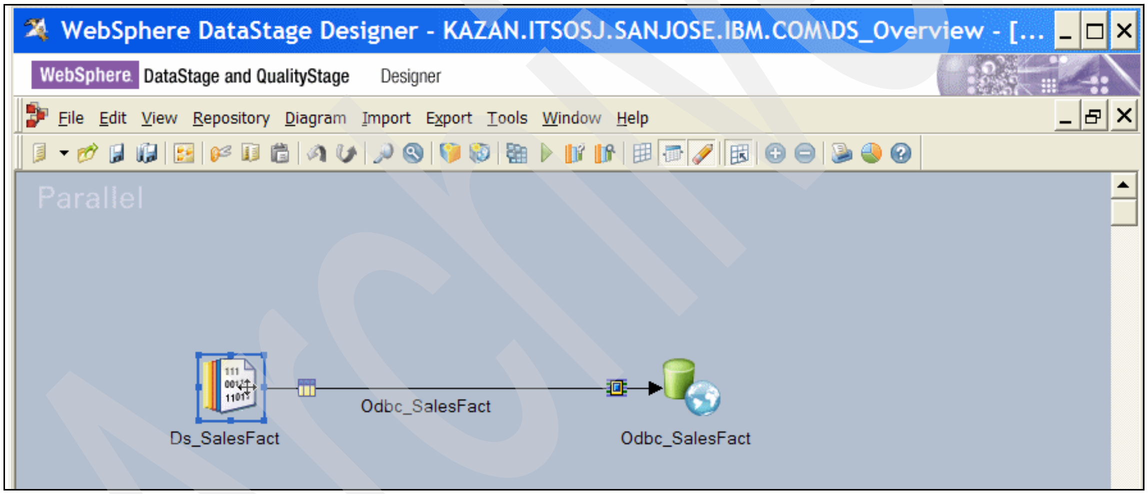


Figure 3-537 Create the J22_Daily_UpdateSalesFact job 1/3

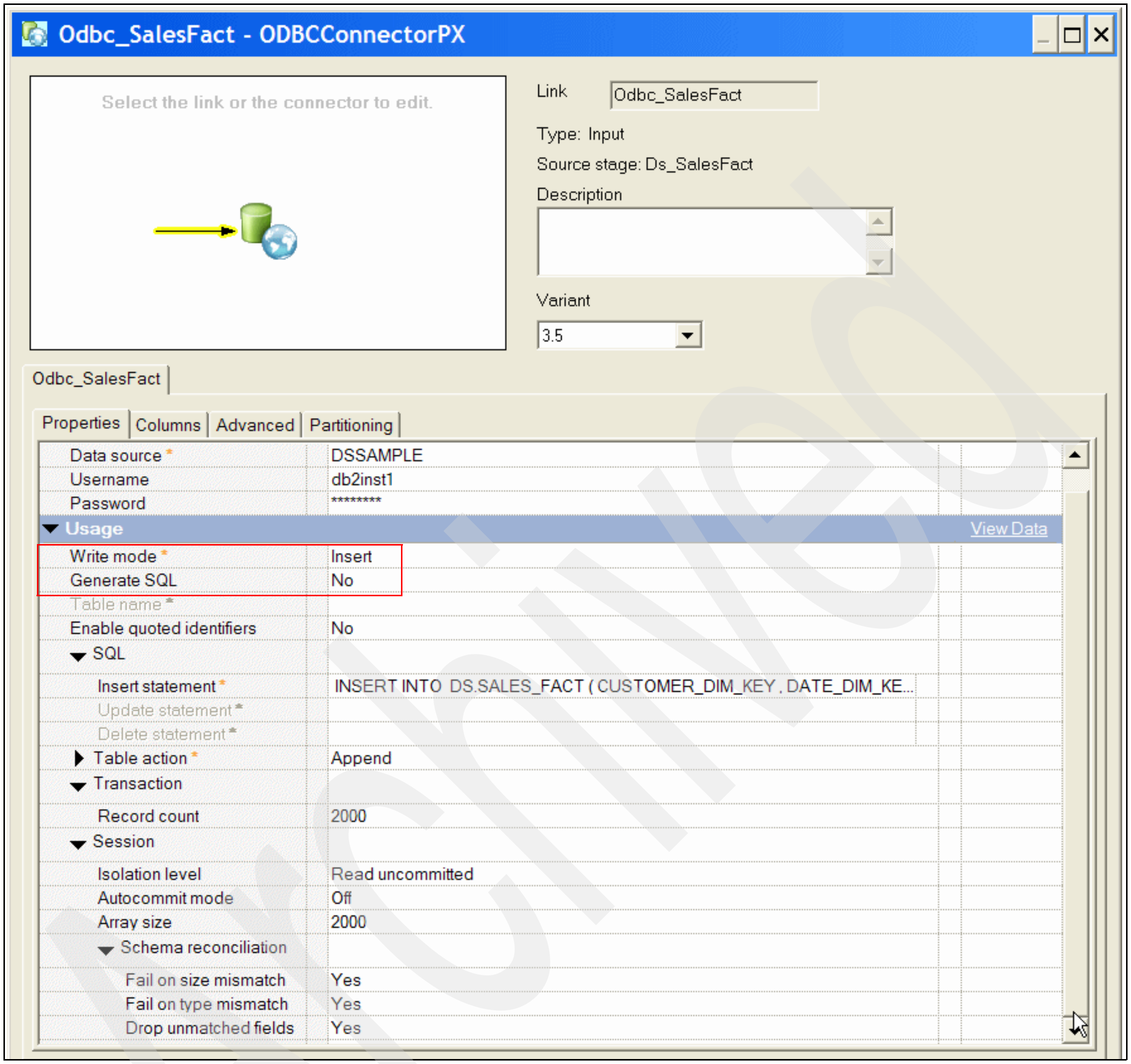


Figure 3-538 Create the J22_Daily_UpdateSalesFact job 2/3

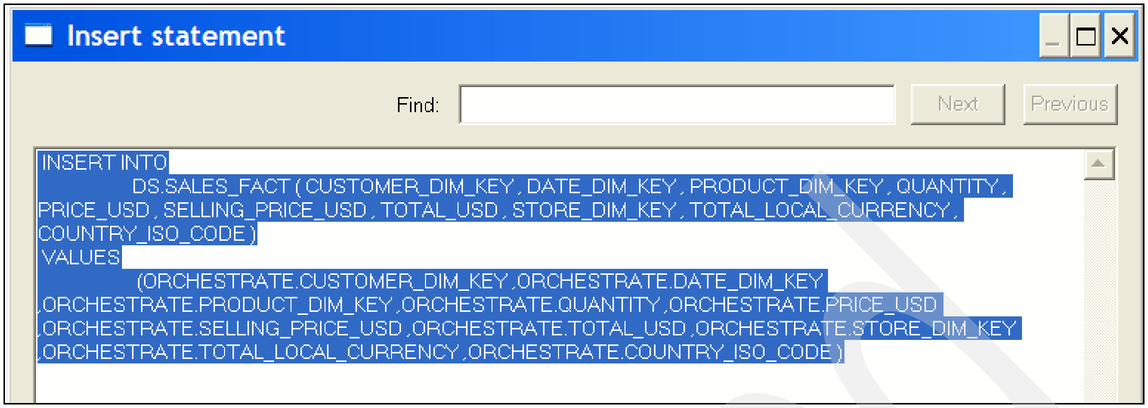


Figure 3-539 Create the J22_Daily_UpdateSalesFact job 3/3

J22_Daily_UpdateSalesFact (Day 1) execution

Figure 3-540 on page 506 through Figure 3-542 on page 506 show the results of the execution of this job with Day 1 data described earlier.

- Figure 3-540 on page 506 shows the results of the execution. It accepts 1 row as input from the “J17_DailyCreateSalesFactDS (Day1) execution” on page 475 job as seen in Figure 3-503 on page 477 and Figure 3-504 on page 477.
- The output shows 7 rows being written to the Odbc_SalesFact link which is used to update the SALES_FACT table. Figure 3-541 on page 506 and Figure 3-542 on page 506 show the updated contents of the SALES_FACT table as highlighted.

This concludes Day 1 processing.

You can proceed to Day 2 processing as described in 3.1.4, “Recurring tasks (Day 2)” on page 507.

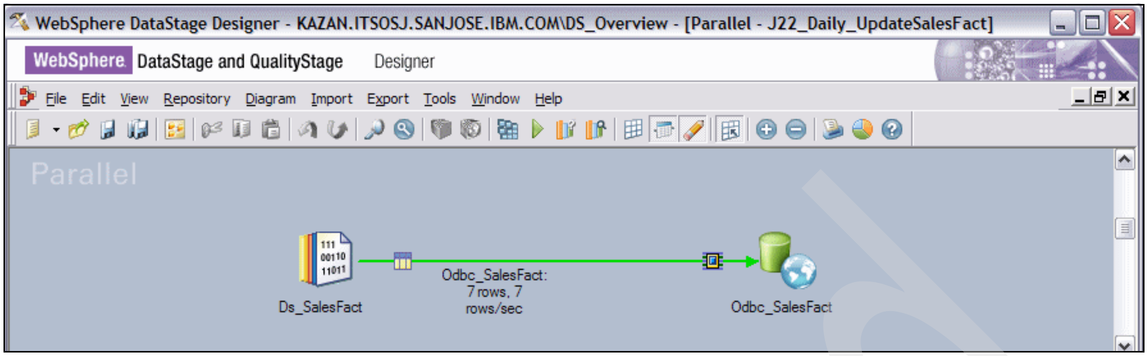


Figure 3-540 Execute the J22_Daily_UpdateSalesFact job (Day 1) 1/3

View Data

CUSTOMER_DIM_KEY	DATE_DIM_KEY	PRODUCT_DIM_KEY	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	STORE_DIM_KEY	TOTAL_LOC
832	36	777	2	35.00	25.00	50.00	743	5726.50
832	37	777	2	35.00	25.00	50.00	742	50.00
836	36	777	4	17.69	15.00	60.00	743	109.26
836	37	777	2	17.69	15.00	30.00	743	30.00
838	36	779	3	120.00	120.00	360.00	742	14320.80
839	37	777	1	37.00	37.00	37.00	742	37.00
840	37	776	3	37.00	37.00	111.00	742	111.00
842	37	776	2	17.69	15.00	30.00	743	29.02
842	37	777	1	35.00	33.33	33.33	742	33.33
842	37	777	2	17.69	15.00	30.00	743	30.00

Figure 3-541 Execute the J22_Daily_UpdateSalesFact job (Day 1) 2/3

View Data

PRODUCT_DIM_KEY	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	STORE_DIM_KEY	TOTAL_LOCAL_CURRENCY	COUNTRY_ISO_CODE
777	2	35.00	25.00	50.00	743	5726.50	JPN
777	2	35.00	25.00	50.00	742	50.00	USA
777	4	17.69	15.00	60.00	743	109.26	BRA
777	2	17.69	15.00	30.00	743	30.00	USA
779	3	120.00	120.00	360.00	742	14320.80	IND
777	1	37.00	37.00	37.00	742	37.00	USA
776	3	37.00	37.00	111.00	742	111.00	USA
776	2	17.69	15.00	30.00	743	29.02	CAD
777	1	35.00	33.33	33.33	742	33.33	USA
777	2	17.69	15.00	30.00	743	30.00	USA

Figure 3-542 Execute the J22_Daily_UpdateSalesFact job (Day 1) 3/3

3.1.4 Recurring tasks (Day 2)

In this cycle, we processed the following data on November 7th, 2007:

- Dimension table changes:
  - Customer dimension:
    - Update (TABLE_CMD value of U) of CUSTOMER_ID 6

      The Type 1 changes are NAME (Belad Davis), WORK_PHONE (408-555-8333), and WORK_ADDRESS (2 N First Street).

      The Type 2 changes are MEMBERSHIP_EXPIRE_DT (2020-02-13) and MEMBERSHIP_LEVEL (G).

    These are shown in Figure 3-543 through Figure 3-544.

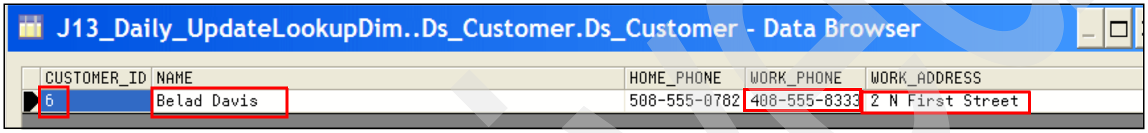


Figure 3-543 Customer dimension attribute changes 1/2

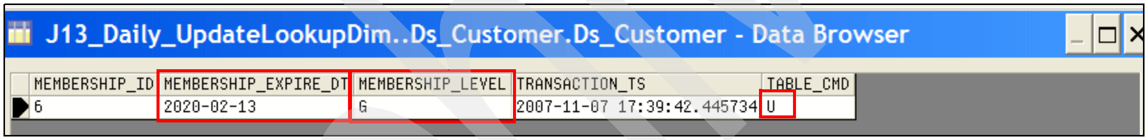


Figure 3-544 Customer dimension attribute changes 2/2

  - Product dimension:
    - Insert (TABLE_CMD value of I) of PRODUCT_ID 7
    - Insert (TABLE_CMD value of I) of PRODUCT_ID 11

    These are shown in Figure 3-545 through Figure 3-548.

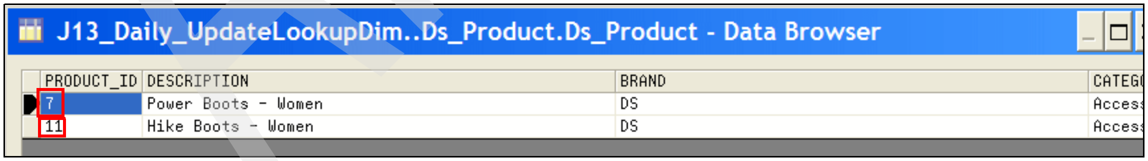


Figure 3-545 Product dimension attribute changes 1/4

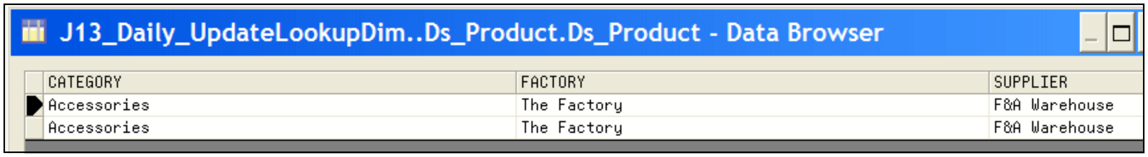


Figure 3-546 Product dimension attribute changes 2/4

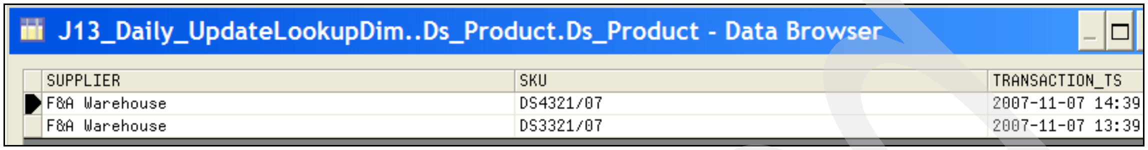


Figure 3-547 Product dimension attribute changes 3/4

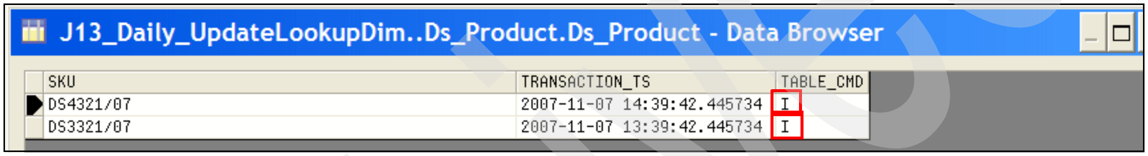


Figure 3-548 Product dimension attribute changes 4/4

– Store dimension:
  - Insert (TABLE_CMD value of I) of STORE_ID 9
  - Update (TABLE_CMD value of U) of STORE_ID 33

    The Type 1 change is STATE_POPULATION (37700000).

    The Type 2 change is MANAGER_NAME (Abigail Wilson).
  - Update (TABLE_CMD value of U) of STORE_ID 1

    The Type 1 change is STATE_POPULATION (37700000).

    There are no Type 2 changes.

These are shown in Figure 3-549 through Figure 3-552.

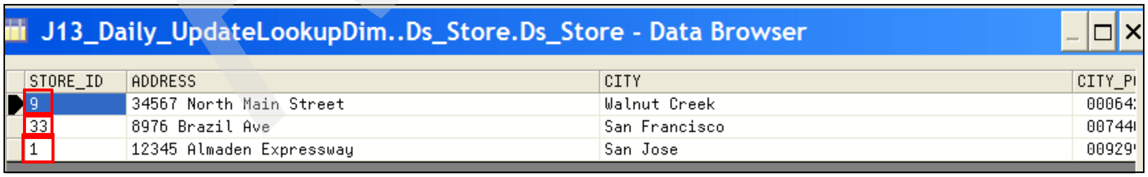
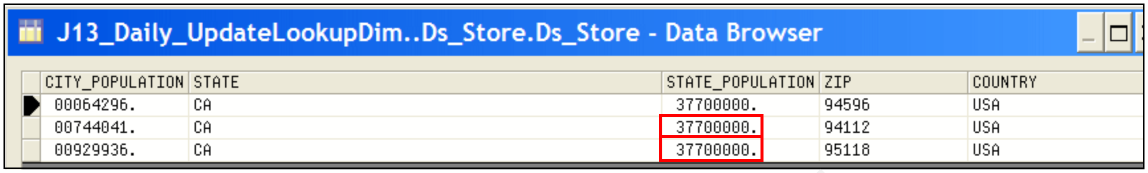


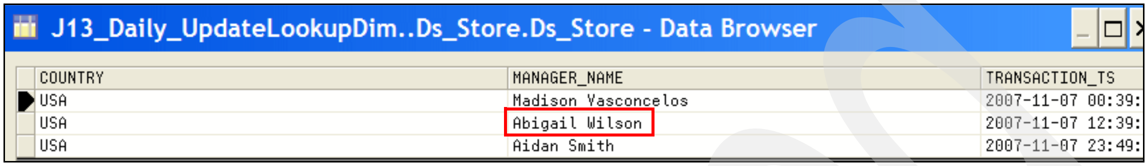
Figure 3-549 Store dimension attribute changes 1/4



J13_Daily_UpdateLookupDim..Ds_Store.Ds_Store - Data Browser

CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY
00064296.	CA	37700000.	94596	USA
00744041.	CA	37700000.	94112	USA
00929936.	CA	37700000.	95118	USA

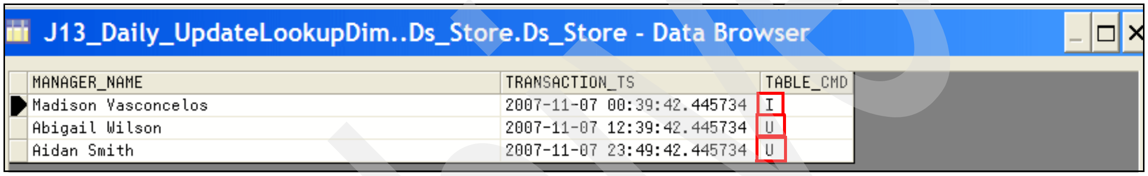
Figure 3-550 Store dimension attribute changes 2/4



J13_Daily_UpdateLookupDim..Ds_Store.Ds_Store - Data Browser

COUNTRY	MANAGER_NAME	TRANSACTION_TS
USA	Madison Vasconcelos	2007-11-07 00:39:
USA	Abigail Wilson	2007-11-07 12:39:
USA	Aidan Smith	2007-11-07 23:49:

Figure 3-551 Store dimension attribute changes 3/4



J13_Daily_UpdateLookupDim..Ds_Store.Ds_Store - Data Browser

MANAGER_NAME	TRANSACTION_TS	TABLE_CMD
Madison Vasconcelos	2007-11-07 00:39:42.445734	I
Abigail Wilson	2007-11-07 12:39:42.445734	U
Aidan Smith	2007-11-07 23:49:42.445734	U

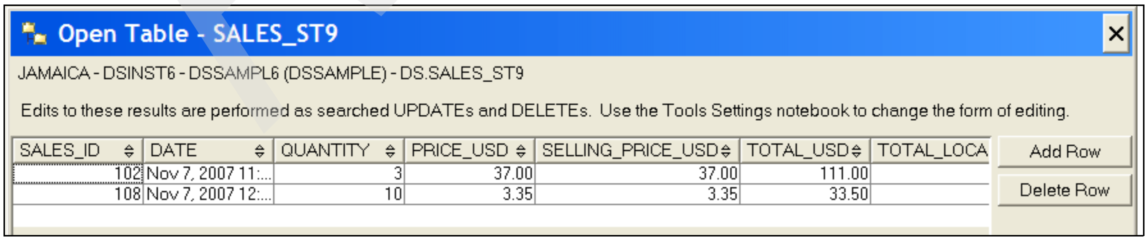
Figure 3-552 Store dimension attribute changes 4/4

- Sales transactions:

  Sales transactions are collected from three stores — ST1 (STORE_ID of 1) with 1 transaction as shown in Example 3-3, ST9 (STORE_ID of 9) with 2 transactions, as shown in Figure 3-553 and Figure 3-554, and ST33 (STORE_ID of 33) with 3 transactions as shown in Figure 3-555 and Figure 3-556.

Example 3-3 STORE_ID 1 sales transactions

```
SALES_ID, DATE, QUANTITY, PRICE_USD, SELLING_PRICE_USD, COUNTRY_ISO_CODE, TOTAL_USD, CUSTOMER_ID,
STORE_ID, PRODUCT_ID
101,2007-11-07 10:09:42,1,37,37,CHN,37,0009,1,5,
```



Open Table - SALES_ST9

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.SALES_ST9

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

SALES_ID	DATE	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	TOTAL_LOCA
102	Nov 7, 2007 11:...	3	37.00	37.00	111.00	
108	Nov 7, 2007 12:...	10	3.35	3.35	33.50	

Add Row

Delete Row

Figure 3-553 STORE_ID 9 sales transactions 1/2

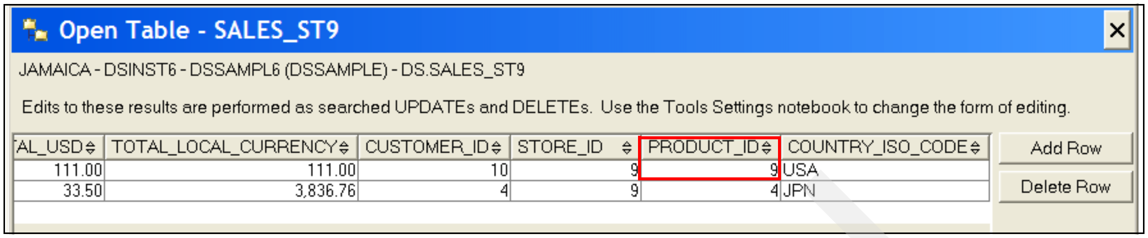


Figure 3-554 STORE_ID 9 sales transactions 2/2

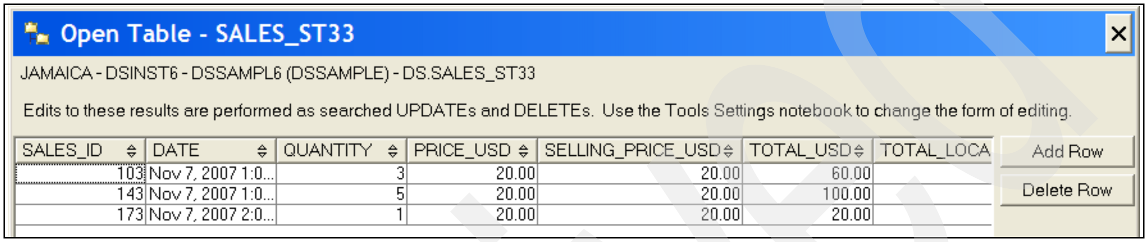


Figure 3-555 STORE_ID 33 sales transactions 1/2

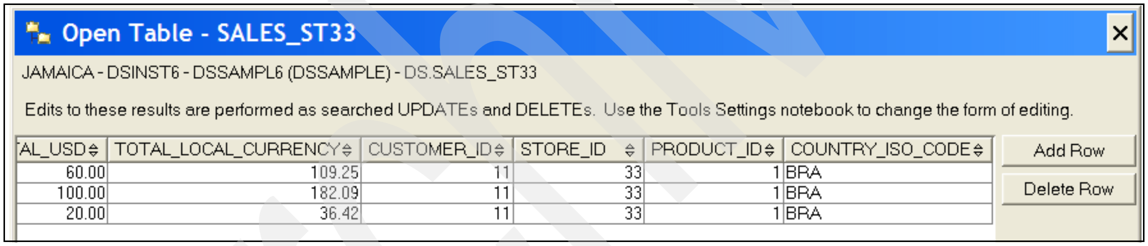


Figure 3-556 STORE_ID 33 sales transactions 2/2

Two of these sales transactions were deliberately tailored to create the following error conditions, which result in these transactions being rejected at some point.

- The one sales transaction from Store ST1 has an invalid COUNTRY_ISO_CODE of ‘CHN’.
- PRODUCT_ID of 9 does not exist in the Product dimension table, which invalidates a sales transaction in Store 9.

These fields are highlighted in Example 3-3 on page 509 through Figure 3-554 here.

In addition, no sales transactions were created with PRODUCT_ID of 7 and 11 (which are inserted as new business keys), which results in these dimension changes corresponding to a late arriving data scenario.

Table 3-2 on page 342 identifies the jobs executed in the recurring (daily) tasks.

- The configuration of these tasks is briefly described in “Recurring tasks (Day 1)” on page 348.
- The execution of these jobs and the corresponding recurring tasks (Day 2) are briefly described in the following sections starting with “J07_IL_Daily_LoadSalesStore (Day 2) execution” on page 511.

> **Note:** “J06_IL_Daily_CreateCurrencyLookup_Service” on page 227 should be executed every day to pick up the latest exchange rates for each ISO country code. In our case however, we created all the exchange rates for the different ISO country code countries for our three recurring daily cycles up front (during the initial load phase), and therefore do not repeat it here.

J07_IL_Daily_LoadSalesStore (Day 2) execution

This job has to be repeated for sales transactions for each of the three stores (1, 9, and 33) for Day 2.

- Figure 3-557 on page 512 shows the Job Run Options window that identifies the input file (J07_Seq_Sales_20071107_ST1.txt) containing the sales transactions, the name of the schema file (J07_Seq_Sales_schema.osh), and the name of the interim DB2 table (DS.SALES_ST1) to which these sales transactions are written.

  Figure 3-558 on page 512 shows the execution results of this job, indicating one sales transaction being processed but zero sales transaction in the output, since the input transaction was rejected because of an invalid COUNTRY_ISO_CODE of ‘CHN’.

  The contents of the DB2 interim table after the execution are shown in Figure 3-558 on page 512.

- Figure 3-559 on page 513 shows the Job Run Options window that identifies the input file (J07_Seq_Sales_20071107_ST9.txt) containing the sales transactions, the name of the schema file (J07_Seq_Sales_schema.osh), and the name of the interim DB2 table (DS.SALES_ST9) to which these sales transactions are written.

  Figure 3-560 on page 513 shows the execution results of this job, indicating 2 sales transactions being processed.

  The contents of the DB2 interim table after the execution are shown in Figure 3-553 on page 509 and Figure 3-554 on page 510.

- Figure 3-561 on page 513 shows the Job Run Options window that identifies the input file (J07_Seq_Sales_20071107_ST33.txt) containing the sales transactions, the name of the schema file (J07_Seq_Sales_schema.osh), and the name of the interim DB2 table (DS.SALES_ST33) to which these sales transactions are written.

Figure 3-562 on page 514 shows the execution results of this job, indicating 6 sales transactions being processed.

The contents of the DB2 interim table after the execution are shown in Figure 3-555 on page 510 and Figure 3-556 on page 510.

The next step is to execute the job described in "J14_Daily_CreateAllSalesStoreDS (Day 2) execution" on page 518.

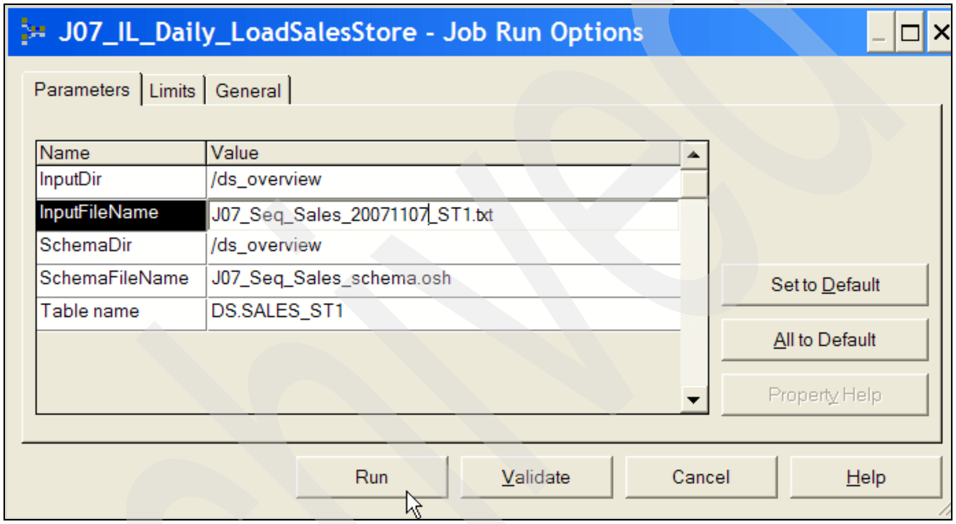


Figure 3-557 Execute the J07_IL_Daily_LoadSalesStore job (Day 2) 1/7

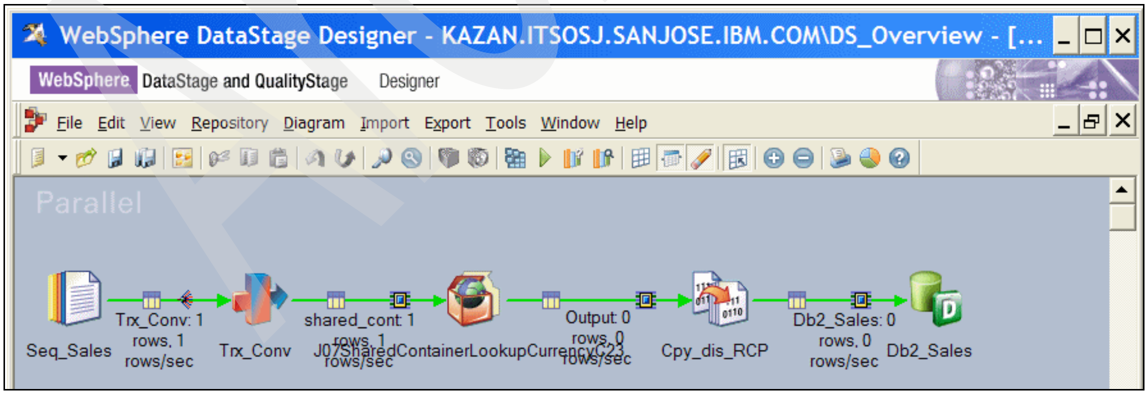


Figure 3-558 Execute the J07_IL_Daily_LoadSalesStore job (Day 2) 2/7

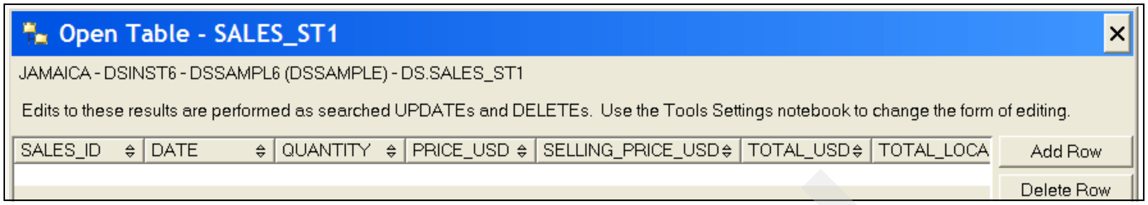


Figure 3-559 Execute the J07_IL_Daily_LoadSalesStore job (Day 2) 3/7

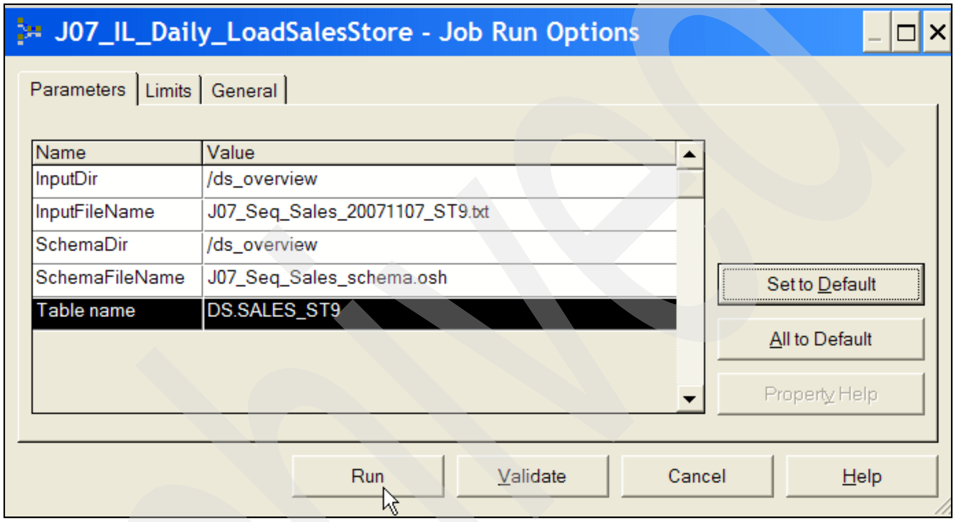


Figure 3-560 Execute the J07_IL_Daily_LoadSalesStore job (Day 2) 4/7

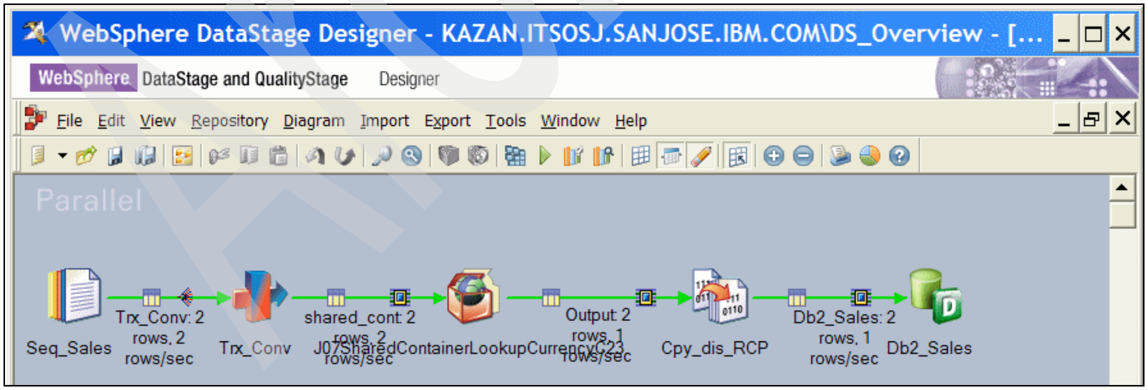


Figure 3-561 Execute the J07_IL_Daily_LoadSalesStore job (Day 2) 5/7

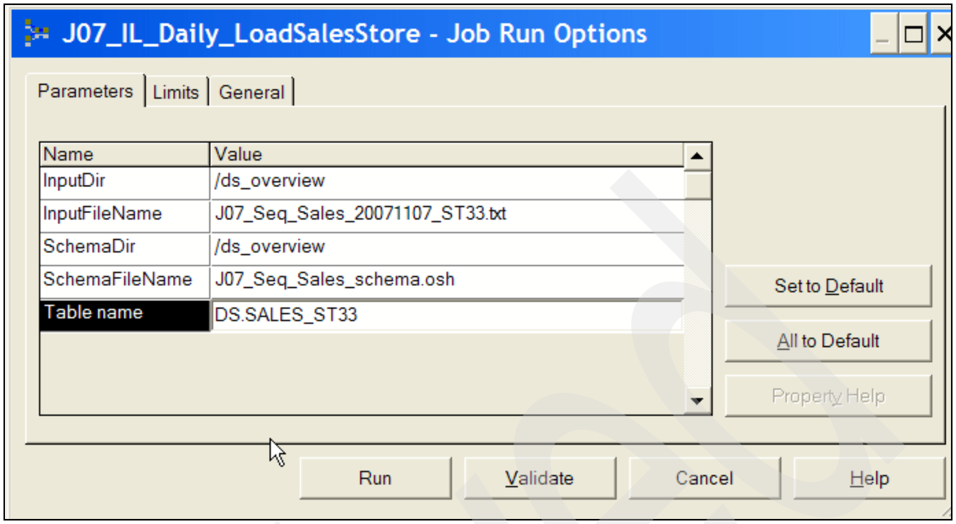


Figure 3-562   Execute the J07_IL_Daily_LoadSalesStore job (Day 2) 6/7

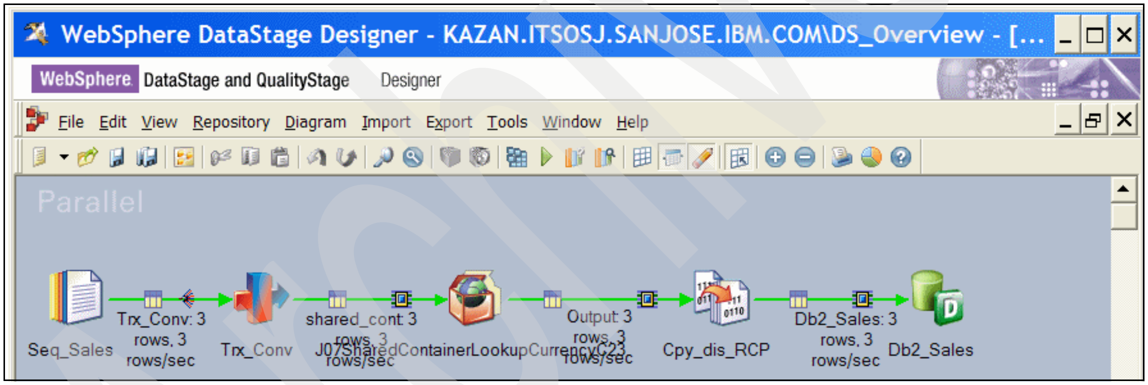


Figure 3-563   Execute the J07_IL_Daily_LoadSalesStore job (Day 2) 7/7

J13_Daily_UpdateLookupDim (Day 2) execution

Figure 3-564 on page 515 through Figure 3-571 on page 518 show the results of the execution of this job with Day 2 data described earlier.

- Figure 3-564 on page 515 shows the results of the execution. It accepts 6 rows as input from the IBM WebSphere MQ message queue, which are changes (three inserts and four updates) to the Customer, Product and Store dimension tables. These changes are written to the Ds_Customer data set (as shown in Figure 3-543 on page 507 and Figure 3-544 on page 507), Ds_Product data set (as shown in Figure 3-545 on page 507 through Figure 3-548 on page 508), and Ds_Store data set (as shown in Figure 3-549 on page 508 through Figure 3-552 on page 509).

- Figure 3-565 on page 516 through Figure 3-567 on page 516 show the LOOKUP_CUSTOMER_DIM table that incorporates the changes (highlighted) due to the update to CUSTOMER_ID 6.
- Figure 3-568 on page 517 and Figure 3-569 on page 517 show the LOOKUP_PRODUCT_DIM table that incorporates the changes (highlighted) due to the inserts of PRODUCT_ID 7 and 11.
- Figure 3-570 on page 517 and Figure 3-571 on page 518 show the LOOKUP_STORE_DIM table that incorporates the changes (highlighted) due to the insert of STORE_ID 9, and updates to STORE_ID 33 and 1.

The next step is to execute the job described in "J14_Daily_CreateAllSalesStoreDS (Day 2) execution" on page 518.

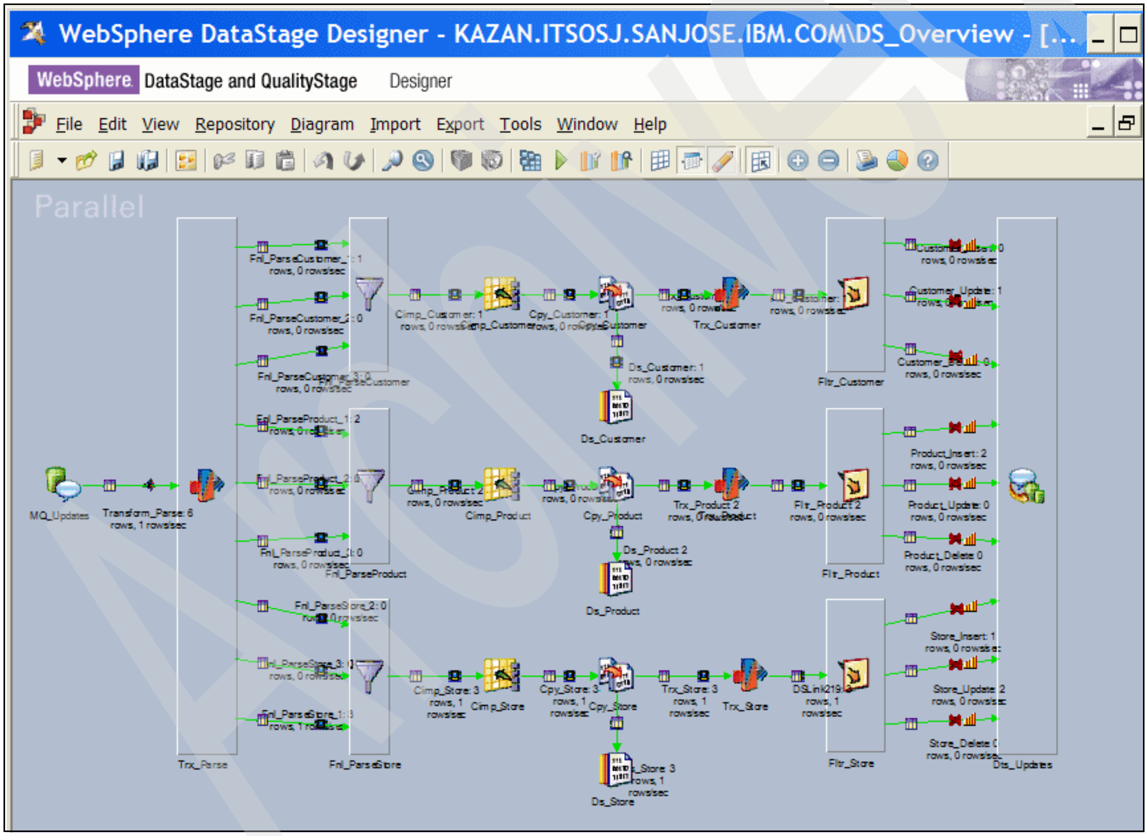


Figure 3-564 Execute the J13_Daily_UpdateLookupDim job (Day 2) 1/8

Open Table - LOOKUP_CUSTOMER_DIM

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.LOOKUP_CUSTOMER_DIM

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

CUSTOMER_ID	NAME	HOME_PHONE	WORK_PHONE	WORK_ADDRESS	WORK_CITY	WORK_S
1	Arch Smith	508-555-0287	408-555-8801	100 AIR ROAD	Santa Cruz	CA
2	Ban Johnson	508-555-0386	408-555-8702	2 ALETHA'S MOUN...	Albany	CA
3	Barn Williams	508-555-0485	408-555-8603			
4	Beel Jones	508-555-0584	408-555-8504			
6	Belad Davis	508-555-0782	408-555-8333	2 N First Street	Albany	CA
8	Mary Wilson	508-555-0980	408-555-8108	2 ALETHA'S MOUN...	Albany	CA
9	Blue Moore	508-555-1079	408-555-8009	2 ALETHA'S MOUN...	Albany	CA
10	Boris Taylor	508-555-1178	408-555-7910	10 BAYLOR WAY	City	CA
11	Desde Lewis	508-555-2465	408-555-6623	23 BRITTANY ROC...	King City	CA
9999	CASH CUSTO...	555-555-5555	555-555-5555			

Add Row

Delete Row

Figure 3-565 Execute the J13_Daily_UpdateLookupDim job (Day 2) 2/8

Open Table - LOOKUP_CUSTOMER_DIM

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.LOOKUP_CUSTOMER_DIM

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

TATE	WORK_ZIP	WORK_COUNTRY	HOME_ADDRESS	HOME_CITY	HOME_ZIP	HOME_STATE	H
	90001	USA	2121 Carl St	Santa Cruz	90001	CA	US
	90002	USA					
			3 ALEX WAY	Amador City	90003	CA	US
	90002	USA	6 ANTON WAY	Bradbury	90006	CA	US
	90002	USA	8 ASTORIA WAY	California City	90008	CA	US
	90002	USA	9 AURIGA WAY	Cathedral City	90009	CA	US
	90010	USA	2 ALETHA'S MOUN...	Albany	90002	CA	US
	90023	USA	2 ALETHA'S MOUN...	Albany	90002	CA	US

Add Row

Delete Row

Figure 3-566 Execute the J13_Daily_UpdateLookupDim job (Day 2) 3/8

Open Table - LOOKUP_CUSTOMER_DIM

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.LOOKUP_CUSTOMER_DIM

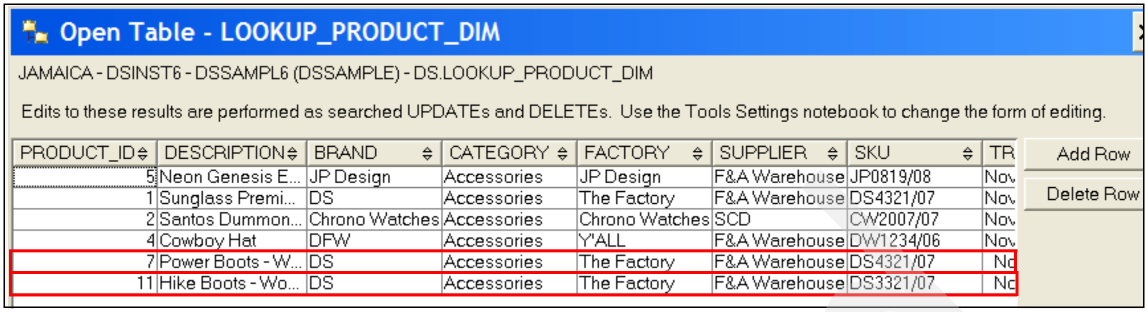
Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

HOME_COUNTRY	MEMBERSHIP_ID	MEMBERSHIP_EXPIRE_DT	MEMBERSHIP_LEVEL	TRANSACTION_TS
USA	1	Feb 16, 2012	S	Nov 6, 2007 12:39:42 P...
	2	Feb 17, 2012	S	Nov 5, 2007 12:00:00 A...
USA	3	Feb 18, 2012	S	Nov 5, 2007 12:00:00 A...
	4	Feb 19, 2012	S	Nov 5, 2007 12:00:00 A...
USA	6	Feb 13, 2020	G	Nov 7, 2007 5:39:42 P...
USA	8	Feb 23, 2012	S	Nov 5, 2007 12:00:00 A...
USA	9	Feb 24, 2012	S	Nov 5, 2007 12:00:00 A...
USA	10	Feb 25, 2012	S	Nov 5, 2007 12:00:00 A...
USA	99	May 10, 2012	P	Nov 5, 2007 12:00:00 A...
USA	0	Dec 31, 2999	P	Nov 5, 2007 12:00:00 A...

Add Row

Delete Row

Figure 3-567 Execute the J13_Daily_UpdateLookupDim job (Day 2) 4/8



Open Table - LOOKUP_PRODUCT_DIM

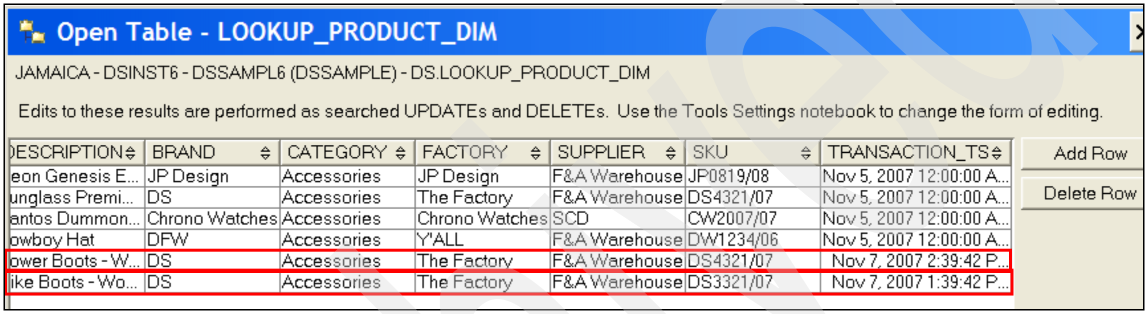
JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.LOOKUP_PRODUCT_DIM

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

PRODUCT_ID	DESCRIPTION	BRAND	CATEGORY	FACTORY	SUPPLIER	SKU	TR
5	Neon Genesis E...	JP Design	Accessories	JP Design	F&A Warehouse	JP0819/08	Nov
1	Sunglass Premi...	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	Nov
2	Santos Dummon...	Chrono Watches	Accessories	Chrono Watches	SCD	CW2007/07	Nov
4	Cowboy Hat	DFW	Accessories	Y'ALL	F&A Warehouse	DW1234/06	Nov
7	Power Boots - W...	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	No
11	Hike Boots - Wo...	DS	Accessories	The Factory	F&A Warehouse	DS3321/07	No

Add Row
Delete Row

Figure 3-568 Execute the J13_Daily_UpdateLookupDim job (Day 2) 5/8



Open Table - LOOKUP_PRODUCT_DIM

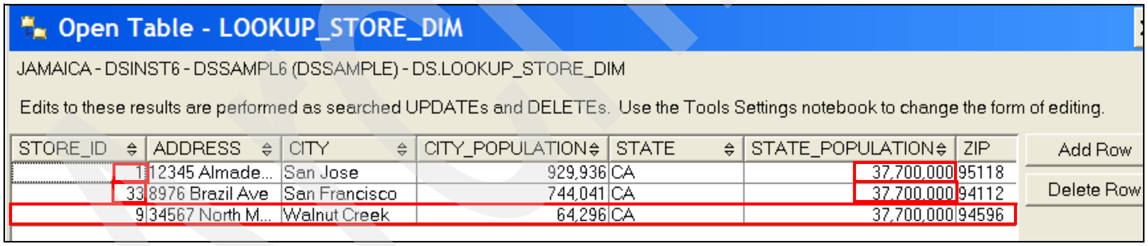
JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.LOOKUP_PRODUCT_DIM

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

DESCRIPTION	BRAND	CATEGORY	FACTORY	SUPPLIER	SKU	TRANSACTION_TS
eon Genesis E...	JP Design	Accessories	JP Design	F&A Warehouse	JP0819/08	Nov 5, 2007 12:00:00 A...
unglass Premi...	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	Nov 5, 2007 12:00:00 A...
antos Dummon...	Chrono Watches	Accessories	Chrono Watches	SCD	CW2007/07	Nov 5, 2007 12:00:00 A...
owboy Hat	DFW	Accessories	Y'ALL	F&A Warehouse	DW1234/06	Nov 5, 2007 12:00:00 A...
ower Boots - W...	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	Nov 7, 2007 2:39:42 P...
ike Boots - Wo...	DS	Accessories	The Factory	F&A Warehouse	DS3321/07	Nov 7, 2007 1:39:42 P...

Add Row
Delete Row

Figure 3-569 Execute the J13_Daily_UpdateLookupDim job (Day 2) 6/8



Open Table - LOOKUP_STORE_DIM

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.LOOKUP_STORE_DIM

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

STORE_ID	ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP
1	12345 Almade...	San Jose	929,936	CA	37,700,000	95118
33	8976 Brazil Ave	San Francisco	744,041	CA	37,700,000	94112
9	34567 North M...	Walnut Creek	64,296	CA	37,700,000	94596

Add Row
Delete Row

Figure 3-570 Execute the J13_Daily_UpdateLookupDim job (Day 2) 7/8

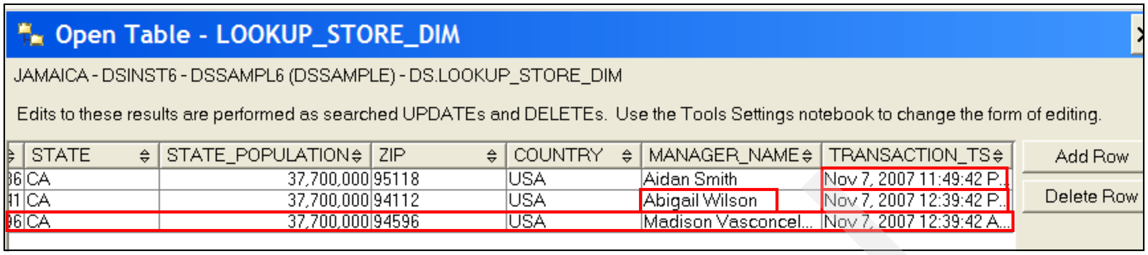


Figure 3-571 Execute the J13_Daily_UpdateLookupDim job (Day 2) 8/8

J14_Daily_CreateAllSalesStoreDS (Day 2) execution

Figure 3-572 through Figure 3-574 on page 519 show the results of the execution of this job with Day 2 data described earlier.

- Figure 3-572 shows the results of the execution. It accepts zero rows from store 1, three row from store 9, and two rows from store 33 for a total of 5 rows that are written to the output data set.
- Figure 3-573 on page 519 and Figure 3-574 on page 519 show the contents of the output data set DS_AllSales.

The next step is to execute the job described in "J15_Daily_CreateSalesAggDS (Day 2) execution" on page 519.

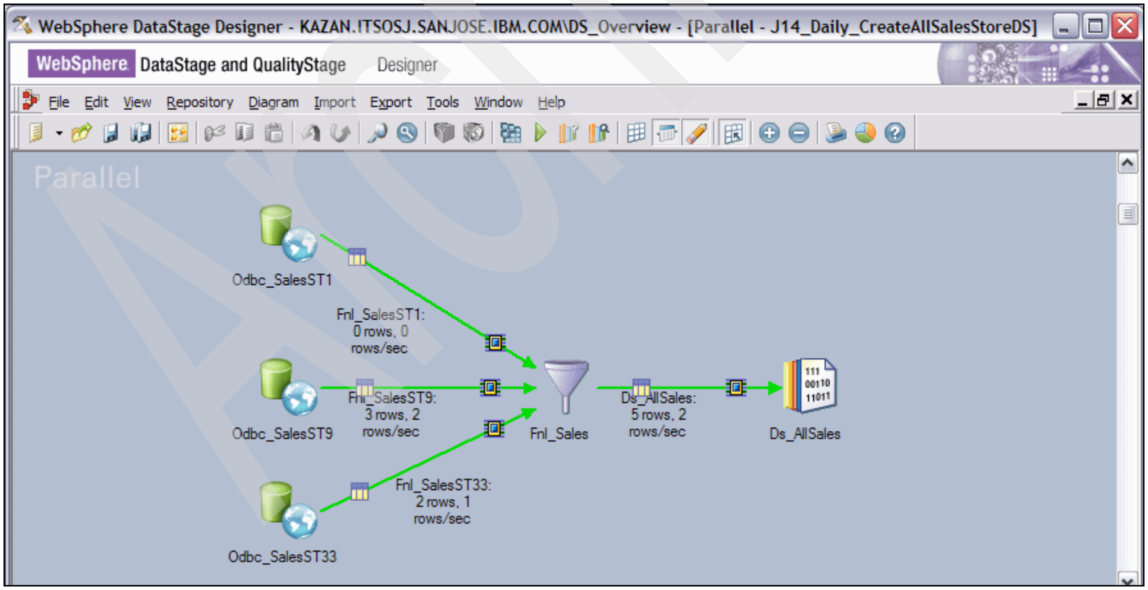


Figure 3-572 Execute the J14_Daily_CreateAllSalesStoreDS job (Day 2) 1/3

J14_Daily_CreateAllSalesStoreDS..Ds_AllSales.Ds_AllSales - Data Browser

SALES_ID	DATE	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID
103	2007-11-07 13:00:02.000000	3	00000020.00	00000020.00	00000060.00	00000109.25	11
102	2007-11-07 11:33:30.000000	3	00000037.00	00000037.00	00000111.00	00000111.00	10
143	2007-11-07 13:07:02.000000	5	00000020.00	00000020.00	00000100.00	00000182.09	11
108	2007-11-07 12:39:41.000000	10	00000003.35	00000003.35	00000033.50	00003836.76	4
173	2007-11-07 14:07:02.000000	1	00000020.00	00000020.00	00000020.00	00000036.42	11

Figure 3-573 Execute the J14_Daily_CreateAllSalesStoreDS job (Day 2) 2/3

J14_Daily_CreateAllSalesStoreDS..Ds_AllSales.Ds_AllSales - Data Browser

PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	COUNTRY_ISO_CODE
00000020.00	00000020.00	00000060.00	00000109.25	11	33	1	BRA
00000037.00	00000037.00	00000111.00	00000111.00	10	9	9	USA
00000020.00	00000020.00	00000100.00	00000182.09	11	33	1	BRA
00000003.35	00000003.35	00000033.50	00003836.76	4	9	4	JPN
00000020.00	00000020.00	00000020.00	00000036.42	11	33	1	BRA

Figure 3-574 Execute the J14_Daily_CreateAllSalesStoreDS job (Day 2) 3/3

J15_Daily_CreateSalesAggDS (Day 2) execution

Figure 3-575 on page 520 through Figure 3-587 on page 522 show the results of the execution of this job with Day 2 data described earlier.

- Figure 3-575 on page 520 shows the results of the execution. It accepts 5 rows as input from the “J14_Daily_CreateAllSalesStoreDS (Day 2) execution” on page 518 job as seen in Figure 3-573 and Figure 3-574.
- The two outputs of this job are:
  – The aggregated sales transactions appended with the dimension lookup tables. This is a total of 2 rows as seen in Figure 3-576 on page 520 through Figure 3-581 on page 521.
  – The rejected sales transactions (either late arriving dimensions or late arriving data). This is a total of 1 row as seen in Figure 3-582 on page 521 through Figure 3-587 on page 522. The invalid column value (PRODUCT_ID of 9) is highlighted.

The next step is to execute the job described in “J16_Daily_CreateScdInputDS (Day 2) execution” on page 522.

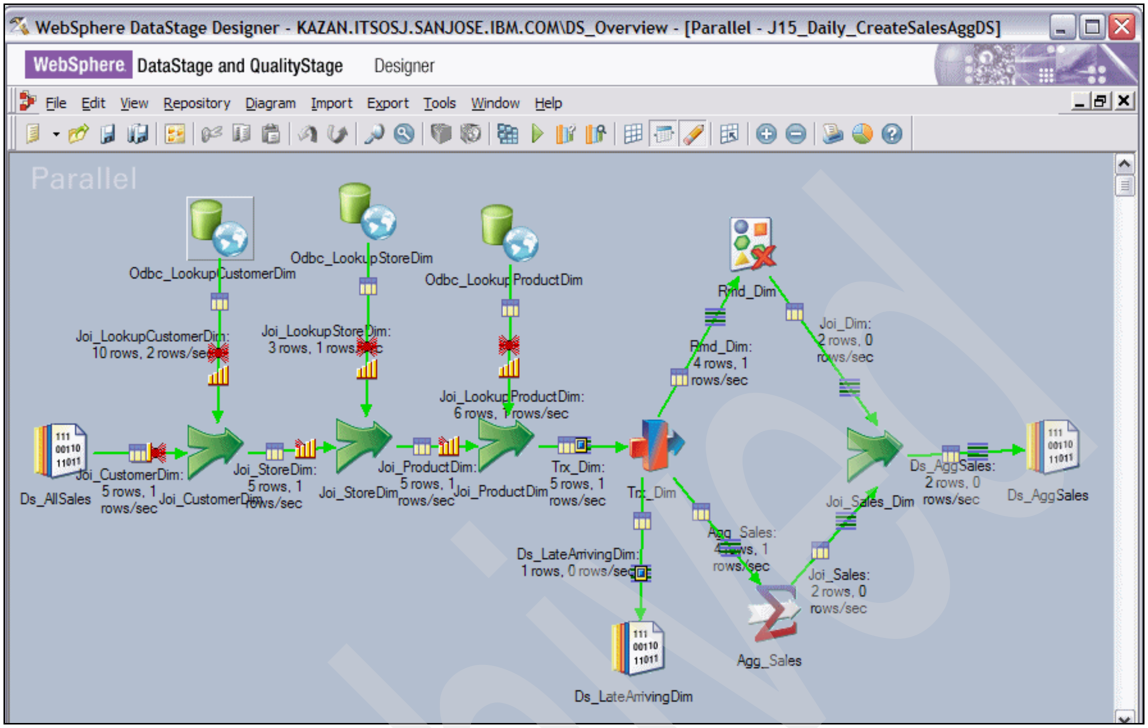


Figure 3-575 Execute the J15_Daily_CreateSalesAggDS job (Day 2) 1/13

J15_Daily_CreateSalesAggDS..Ds_AggSales.Ds_AggSales - Data Browser

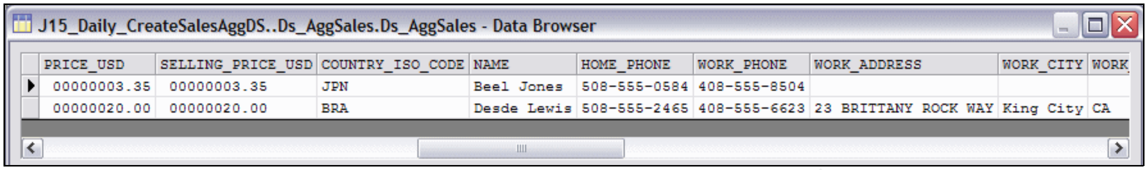
DATE	QUANTITY	TOTAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	MEMBERSHIP_EXPIRE_DT	MEMBERSHIP_LE
2007-11-07	10	00000033.50	00003836.76	4	9	4	2012-02-19	S
2007-11-07	9	00000180.00	00000327.76	11	33	1	2012-05-10	P

Figure 3-576 Execute the J15_Daily_CreateSalesAggDS job (Day 2) 2/13

J15_Daily_CreateSalesAggDS..Ds_AggSales.Ds_AggSales - Data Browser

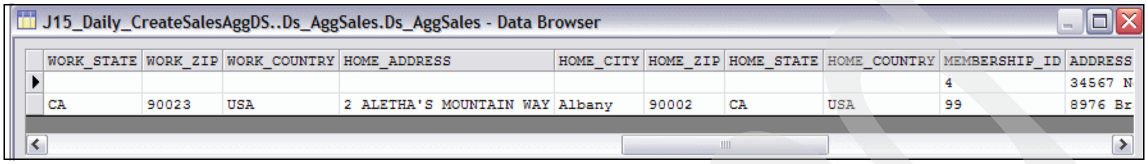
MEMBERSHIP_LEVEL	MANAGER_NAME	DESCRIPTION	BRAND	CATEGORY	FACTORY	SUPPLIER	SKU	PRICE_USD
S	Madison Vasconcelos	Cowboy Hat	DFW	Accessories	Y'ALL	F&A Warehouse	DW1234/06	00000003.
P	Abigail Wilson	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	00000020.

Figure 3-577 Execute the J15_Daily_CreateSalesAggDS job (Day 2) 3/13



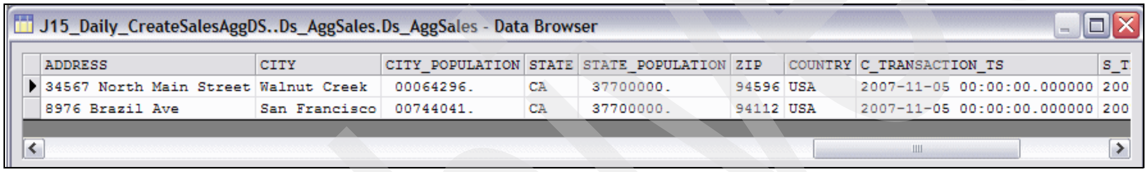
PRICE_USD	SELLING_PRICE_USD	COUNTRY_ISO_CODE	NAME	HOME_PHONE	WORK_PHONE	WORK_ADDRESS	WORK_CITY	WORK
00000003.35	00000003.35	JPN	Beel Jones	508-555-0584	408-555-8504			
00000020.00	00000020.00	BRA	Desde Lewis	508-555-2465	408-555-6623	23 BRITTANY ROCK WAY	King City	CA

Figure 3-578 Execute the J15_Daily_CreateSalesAggDS job (Day 2) 4/13



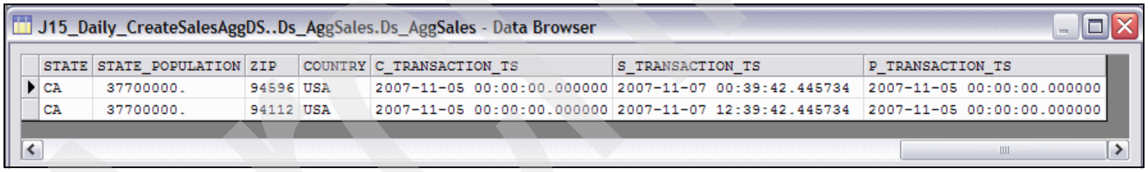
WORK_STATE	WORK_ZIP	WORK_COUNTRY	HOME_ADDRESS	HOME_CITY	HOME_ZIP	HOME_STATE	HOME_COUNTRY	MEMBERSHIP_ID	ADDRESS
								4	34567 N
CA	90023	USA	2 ALETHA'S MOUNTAIN WAY	Albany	90002	CA	USA	99	8976 Br

Figure 3-579 Execute the J15_Daily_CreateSalesAggDS job (Day 2) 5/13



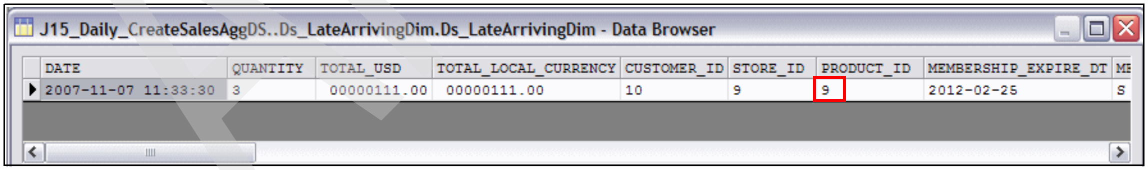
ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	C_TRANSACTION_TS	S_T
34567 North Main Street	Walnut Creek	00064296.	CA	37700000.	94596	USA	2007-11-05 00:00:00.000000	200
8976 Brazil Ave	San Francisco	00744041.	CA	37700000.	94112	USA	2007-11-05 00:00:00.000000	200

Figure 3-580 Execute the J15_Daily_CreateSalesAggDS job (Day 2) 6/13



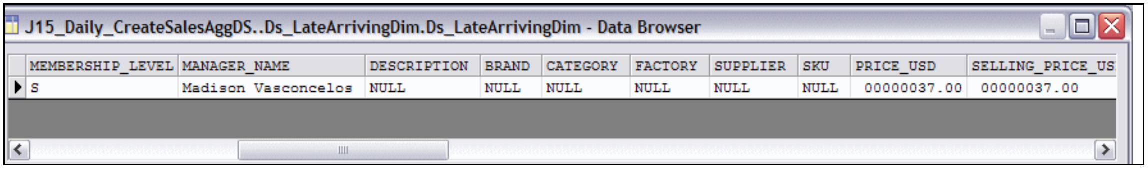
STATE	STATE_POPULATION	ZIP	COUNTRY	C_TRANSACTION_TS	S_TRANSACTION_TS	P_TRANSACTION_TS
CA	37700000.	94596	USA	2007-11-05 00:00:00.000000	2007-11-07 00:39:42.445734	2007-11-05 00:00:00.000000
CA	37700000.	94112	USA	2007-11-05 00:00:00.000000	2007-11-07 12:39:42.445734	2007-11-05 00:00:00.000000

Figure 3-581 Execute the J15_Daily_CreateSalesAggDS job (Day 2) 7/13



DATE	QUANTITY	TOTAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	MEMBERSHIP_EXPIRE_DT	ME
2007-11-07 11:33:30	3	00000111.00	00000111.00	10	9	9	2012-02-25	S

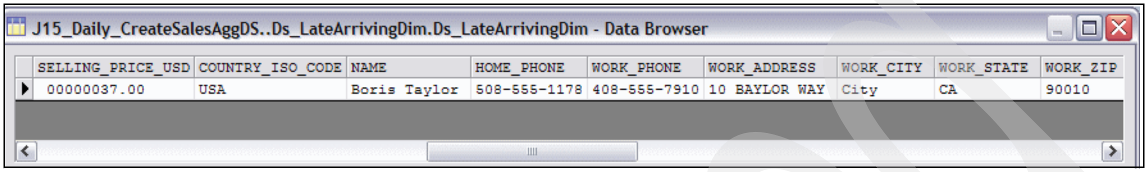
Figure 3-582 Execute the J15_Daily_CreateSalesAggDS job (Day 2) 8/13



J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

MEMBERSHIP_LEVEL	MANAGER_NAME	DESCRIPTION	BRAND	CATEGORY	FACTORY	SUPPLIER	SKU	PRICE_USD	SELLING_PRICE_US
S	Madison Vasconcelos	NULL	NULL	NULL	NULL	NULL	NULL	00000037.00	00000037.00

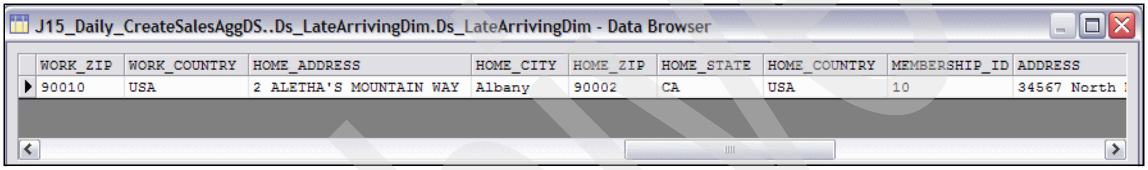
Figure 3-583 Execute the J15_Daily_CreateSalesAggDS job (Day 2) 9/13



J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

SELLING_PRICE_USD	COUNTRY_ISO_CODE	NAME	HOME_PHONE	WORK_PHONE	WORK_ADDRESS	WORK_CITY	WORK_STATE	WORK_ZIP
00000037.00	USA	Boris Taylor	508-555-1178	408-555-7910	10 BAYLOR WAY	City	CA	90010

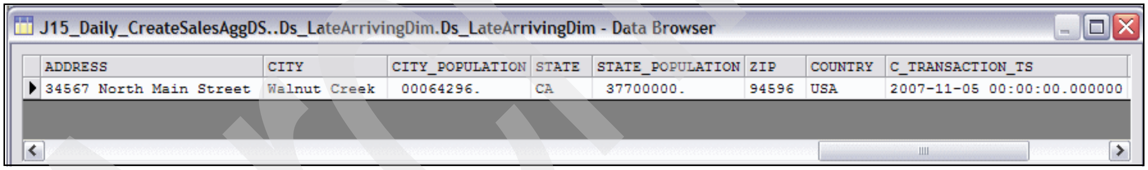
Figure 3-584 Execute the J15_Daily_CreateSalesAggDS job (Day 2) 10/13



J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

WORK_ZIP	WORK_COUNTRY	HOME_ADDRESS	HOME_CITY	HOME_ZIP	HOME_STATE	HOME_COUNTRY	MEMBERSHIP_ID	ADDRESS
90010	USA	2 ALETHA'S MOUNTAIN WAY	Albany	90002	CA	USA	10	34567 North

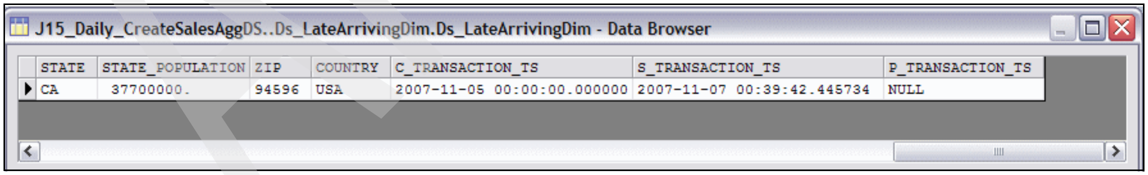
Figure 3-585 Execute the J15_Daily_CreateSalesAggDS job (Day 2) 11/13



J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	C_TRANSACTION_TS
34567 North Main Street	Walnut Creek	00064296.	CA	37700000.	94596	USA	2007-11-05 00:00:00.000000

Figure 3-586 Execute the J15_Daily_CreateSalesAggDS job (Day 2) 12/13



J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

STATE	STATE_POPULATION	ZIP	COUNTRY	C_TRANSACTION_TS	S_TRANSACTION_TS	P_TRANSACTION_TS
CA	37700000.	94596	USA	2007-11-05 00:00:00.000000	2007-11-07 00:39:42.445734	NULL

Figure 3-587 Execute the J15_Daily_CreateSalesAggDS job (Day 2) 13/13

J16_Daily_CreateScdInputDS (Day 2) execution

Figure 3-588 on page 523 through Figure 3-594 on page 525 show the results of the execution of this job with Day 2 data described earlier.

- Figure 3-588 shows the results of the execution. The inputs to this job are as follows:
  - Accepts 2 rows as input from the “J15_Daily_CreateSalesAggDS (Day 2) execution” on page 519 job as seen in Figure 3-576 on page 520 through Figure 3-581 on page 521.
  - Accepts 2 rows (corresponding to PRODUCT_ID 7 and 11) as input from the Product dimension lookup data set generated in “J13_Daily_UpdateLookupDim (Day 2) execution” on page 514.
  - Accepts 3 rows (corresponding to STORE_ID 9, 33, and 1) as input from the Store dimension lookup data set generated in “J13_Daily_UpdateLookupDim (Day 2) execution” on page 514.
  - Accepts 1 row (corresponding to CUSTOMER_ID 6) as input from the Customer dimension lookup data set generated in “J13_Daily_UpdateLookupDim (Day 2) execution” on page 514.
- The output of this job shows 8 rows corresponding to the union of the two inputs via the Funnel stage. Figure 3-589 on page 524 through Figure 3-594 on page 525 show the 8 rows in the output.

The next step is to execute the job described in “J17_DailyCreateSalesFactDS (Day 2) execution” on page 526.

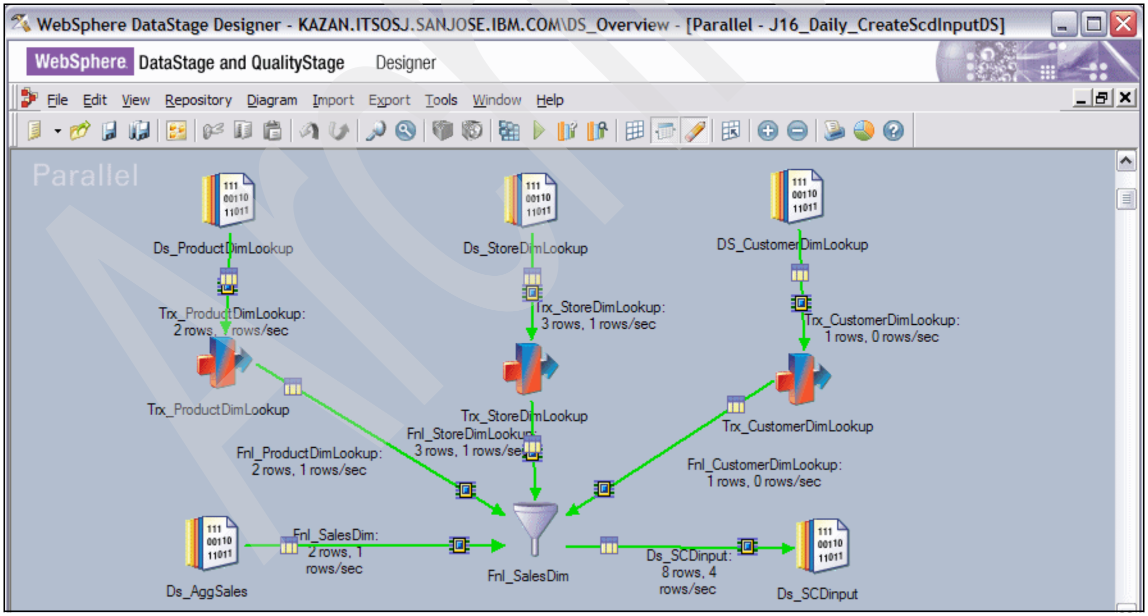
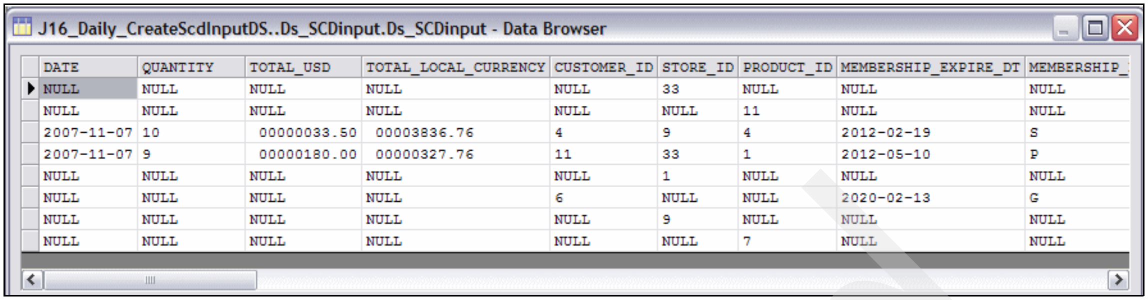
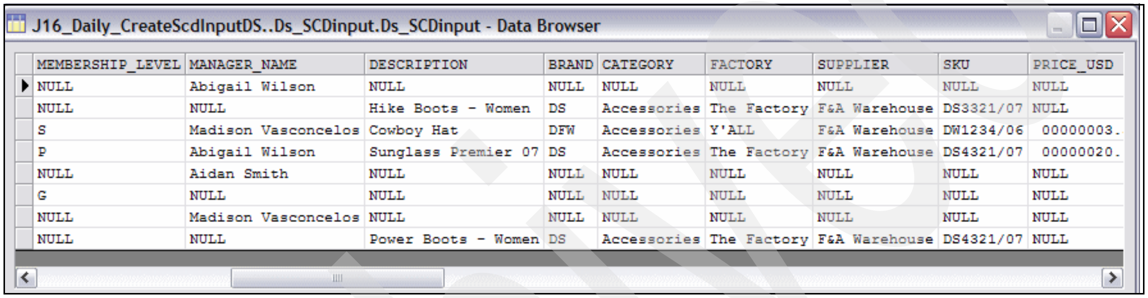


Figure 3-588 Execute the J16_Daily_CreateScdInputDS job (Day 2) 1/7



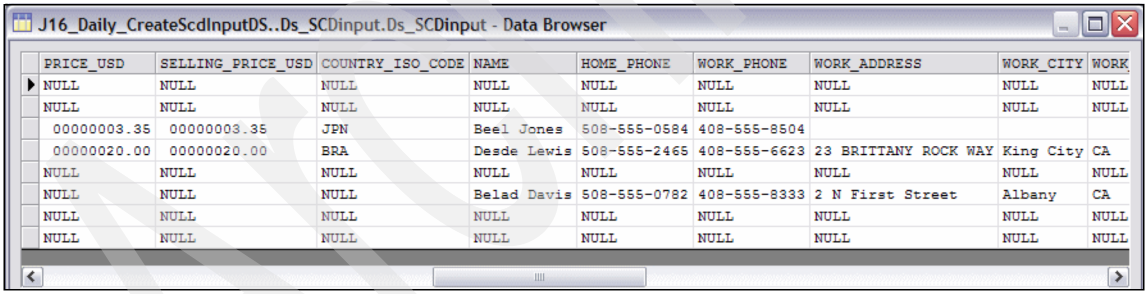
DATE	QUANTITY	TOTAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	MEMBERSHIP_EXPIRE_DT	MEMBERSHIP_
NULL	NULL	NULL	NULL	NULL	33	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	11	NULL	NULL
2007-11-07	10	00000033.50	00003836.76	4	9	4	2012-02-19	S
2007-11-07	9	00000180.00	00000327.76	11	33	1	2012-05-10	P
NULL	NULL	NULL	NULL	NULL	1	NULL	NULL	NULL
NULL	NULL	NULL	NULL	6	NULL	NULL	2020-02-13	G
NULL	NULL	NULL	NULL	NULL	9	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	7	NULL	NULL

Figure 3-589 Execute the J16_Daily_CreateScdInputDS job (Day 2) 2/7



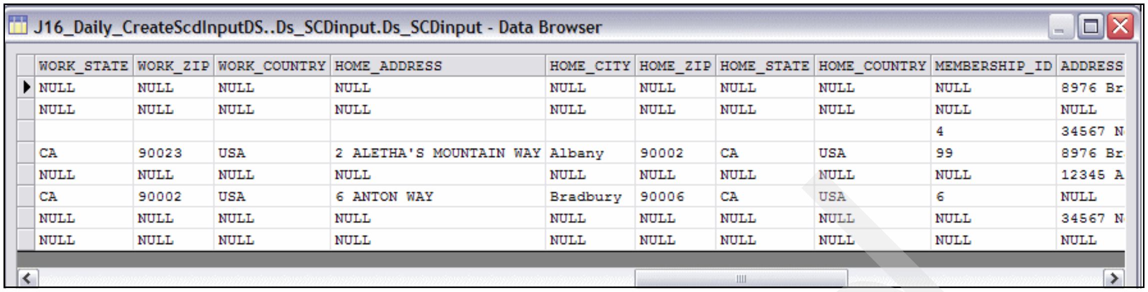
MEMBERSHIP_LEVEL	MANAGER_NAME	DESCRIPTION	BRAND	CATEGORY	FACTORY	SUPPLIER	SKU	PRICE_USD
NULL	Abigail Wilson	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	Hike Boots - Women	DS	Accessories	The Factory	F&A Warehouse	DS3321/07	NULL
S	Madison Vasconcelos	Cowboy Hat	DFW	Accessories	Y'ALL	F&A Warehouse	DW1234/06	00000003.
P	Abigail Wilson	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	00000020.
NULL	Aidan Smith	NULL	NULL	NULL	NULL	NULL	NULL	NULL
G	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	Madison Vasconcelos	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	Power Boots - Women	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	NULL

Figure 3-590 Execute the J16_Daily_CreateScdInputDS job (Day 2) 3/7



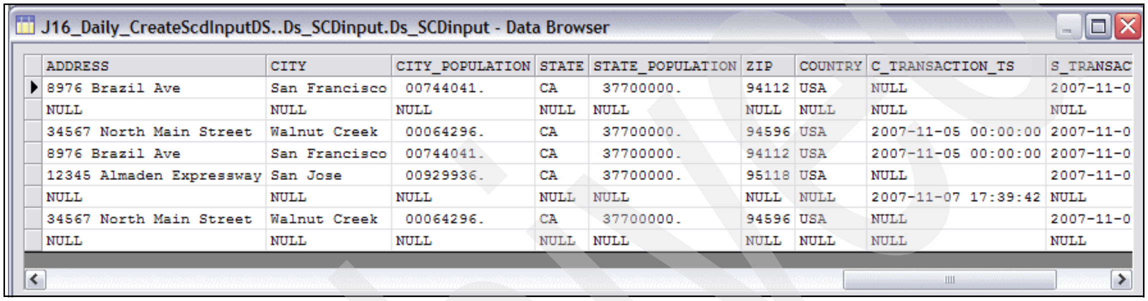
PRICE_USD	SELLING_PRICE_USD	COUNTRY_ISO_CODE	NAME	HOME_PHONE	WORK_PHONE	WORK_ADDRESS	WORK_CITY	WORK
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
00000003.35	00000003.35	JPN	Beel Jones	508-555-0584	408-555-8504			
00000020.00	00000020.00	BRA	Desde Lewis	508-555-2465	408-555-6623	23 BRITTANY ROCK WAY	King City	CA
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	Belad Davis	508-555-0782	408-555-8333	2 N First Street	Albany	CA
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 3-591 Execute the J16_Daily_CreateScdInputDS job (Day 2) 4/7


J16_Daily_CreateScdInputDS..Ds_SCDinput.Ds_SCDinput - Data Browser

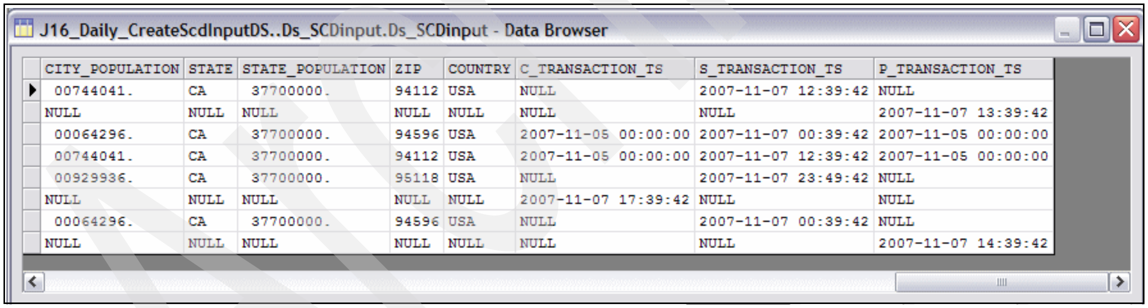
WORK_STATE	WORK_ZIP	WORK_COUNTRY	HOME_ADDRESS	HOME_CITY	HOME_ZIP	HOME_STATE	HOME_COUNTRY	MEMBERSHIP_ID	ADDRESS
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	8976 Br
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
								4	34567 N
CA	90023	USA	2 ALETHA'S MOUNTAIN WAY	Albany	90002	CA	USA	99	8976 Br
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	12345 A
CA	90002	USA	6 ANTON WAY	Bradbury	90006	CA	USA	6	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	34567 N
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 3-592 Execute the J16_Daily_CreateScdInputDS job (Day 2) 5/7


J16_Daily_CreateScdInputDS..Ds_SCDinput.Ds_SCDinput - Data Browser

ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	C_TRANSACTION_TS	S_TRANSAC
8976 Brazil Ave	San Francisco	00744041.	CA	37700000.	94112	USA	NULL	2007-11-0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
34567 North Main Street	Walnut Creek	00064296.	CA	37700000.	94596	USA	2007-11-05 00:00:00	2007-11-0
8976 Brazil Ave	San Francisco	00744041.	CA	37700000.	94112	USA	2007-11-05 00:00:00	2007-11-0
12345 Almaden Expressway	San Jose	00929936.	CA	37700000.	95118	USA	NULL	2007-11-0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	2007-11-07 17:39:42	NULL
34567 North Main Street	Walnut Creek	00064296.	CA	37700000.	94596	USA	NULL	2007-11-0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 3-593 Execute the J16_Daily_CreateScdInputDS job (Day 2) 6/7


J16_Daily_CreateScdInputDS..Ds_SCDinput.Ds_SCDinput - Data Browser

CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	C_TRANSACTION_TS	S_TRANSACTION_TS	P_TRANSACTION_TS
00744041.	CA	37700000.	94112	USA	NULL	2007-11-07 12:39:42	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	2007-11-07 13:39:42
00064296.	CA	37700000.	94596	USA	2007-11-05 00:00:00	2007-11-07 00:39:42	2007-11-05 00:00:00
00744041.	CA	37700000.	94112	USA	2007-11-05 00:00:00	2007-11-07 12:39:42	2007-11-05 00:00:00
00929936.	CA	37700000.	95118	USA	NULL	2007-11-07 23:49:42	NULL
NULL	NULL	NULL	NULL	NULL	2007-11-07 17:39:42	NULL	NULL
00064296.	CA	37700000.	94596	USA	NULL	2007-11-07 00:39:42	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	2007-11-07 14:39:42

Figure 3-594 Execute the J16_Daily_CreateScdInputDS job (Day 2) 7/7

J17_DailyCreateSalesFactDS (Day 2) execution

Figure 3-595 on page 527 through Figure 3-606 on page 529 show the results of the execution of this job with Day 2 data described earlier.

- Figure 3-595 on page 527 shows the results of the execution. It accepts 8 rows as input from the "J16_Daily_CreateScdInputDS (Day 2) execution" on page 522 job as seen in Figure 3-589 on page 524 through Figure 3-594.
- The outputs of this job are as follows:
  - Four rows to the Ds_StoreDimUpdate data set (shown in Figure 3-596 on page 527 and Figure 3-597 on page 527).
    - There is one row for the insert of STORE_ID 9.
    - There are two rows for the update of STORE_ID 33 because it has both Type 1 and Type 2 (MANAGER_NAME) changes. The Type 2 change requires the expiry of the existing row in the Store dimension table (CURRENT_IND to 'N' and EXPIRATION_TS to Current Timestamp[12]), and the addition of a new current row (CURRENT_IND of 'Y', EFFECTIVE_TS and EXPIRATION-TS).
    - There is only 1 row for the update of STORE_ID 1 because it only has Type 1 changes which requires an update in place.
  - Two rows to the Ds_CustomerDimUpdate data set (shown in Figure 3-598 on page 528 through Figure 3-600 on page 528) for the update of CUSTOMER_ID 6 because it has both Type 1 and Type 2 changes requiring expiry of the existing record in the dimension table.
  - Two rows to the Ds_ProductDimUpdate data set (shown in Figure 3-601 on page 528 and Figure 3-602 on page 528) corresponding to the 2 inserts to the Product dimension table.
  - No rows to the Ds_DateDimUpdate data set, since there were no changes to the Date dimension table.
  - Two rows (as expected from the input) are written to the Ds_SalesFactUpdate data set with the appropriate surrogate key assigned to each sales transaction as shown in Figure 3-603 on page 529 through Figure 3-604 on page 529.
  - The six rows corresponding to late arriving data in the input are rejected and written to the Ds_LateArrivingData data set as shown in Figure 3-605 on page 529 and Figure 3-606 on page 529.

The next step is to execute the job described in "J18_Daily_UpdateStoreDim (Day 2) execution" on page 529.

[12] This should actually have been the C_TRANSACTION_TS value of November 7th, 2007, but was wrongly configured.

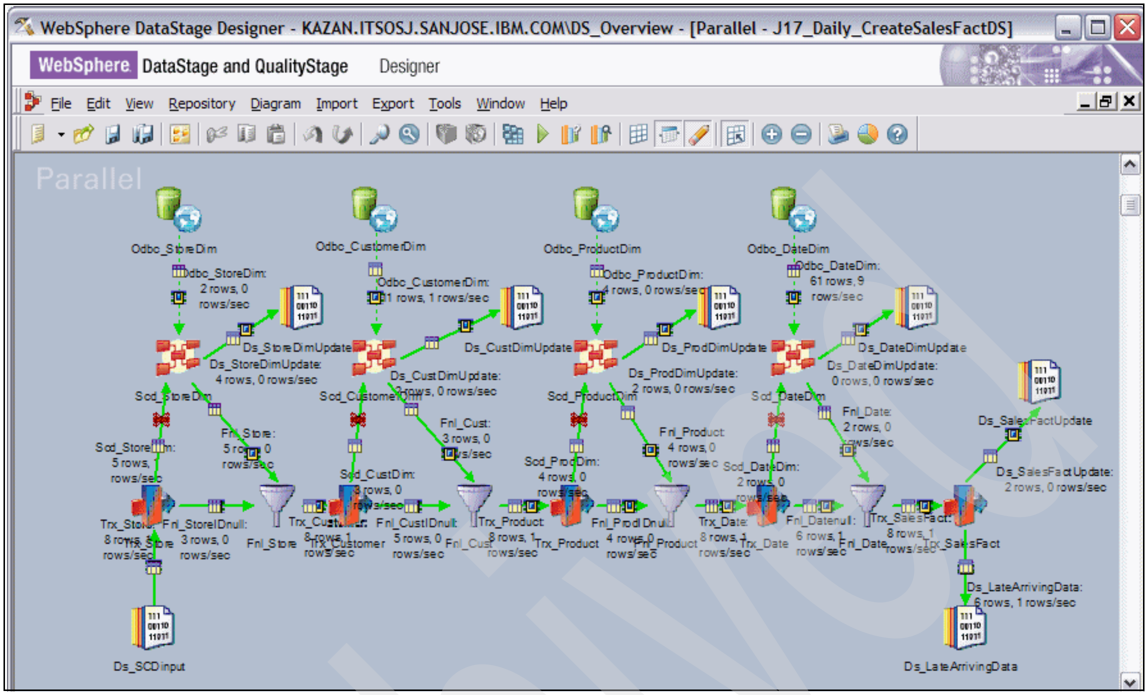


Figure 3-595 Execute the J17_DailyCreateSalesFactDS (Day 2) job (Day 2) 1/12

J17_Daily_CreateSalesFactDS..Ds_StoreDimUpdate.Ds_StoreDimUpdate - Data Browser

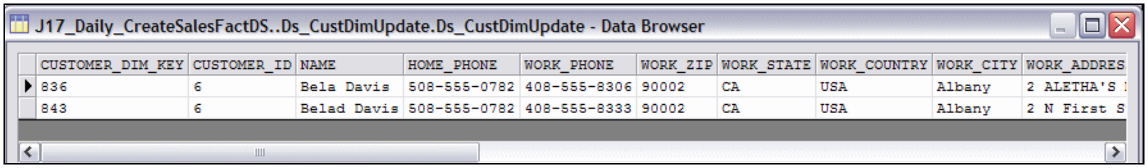
STORE_DIM_KEY	STORE_ID	ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	M
742	33	8976 Brazil Ave	San Francisco	00744041.	CA	33871648.	94112	USA	E
745	33	8976 Brazil Ave	San Francisco	00744041.	CA	37700000.	94112	USA	A
743	NULL	NULL	NULL	00929936.	NULL	37700000.	NULL	NULL	N
744	9	34567 North Main Street	Walnut Creek	00064296.	CA	37700000.	94596	USA	M

Figure 3-596 Execute the J17_DailyCreateSalesFactDS (Day 2) job (Day 2) 2/12

J17_Daily_CreateSalesFactDS..Ds_StoreDimUpdate.Ds_StoreDimUpdate - Data Browser

STATE	STATE_POPULATION	ZIP	COUNTRY	MANAGER_NAME	CURRENT_IND	EFFECTIVE_TS	EXPIRATION_TS
CA	33871648.	94112	USA	Emma Hales	N	2007-11-05 00:00:00	2007-11-07 12:39:42
CA	37700000.	94112	USA	Abigail Wilson	Y	2007-11-07 12:39:42	2099-12-31 00:00:00
NULL	37700000.	NULL	NULL	NULL	Y	NULL	2099-12-31 00:00:00
CA	37700000.	94596	USA	Madison Vasconcelos	Y	2007-11-07 00:39:42	2099-12-31 00:00:00

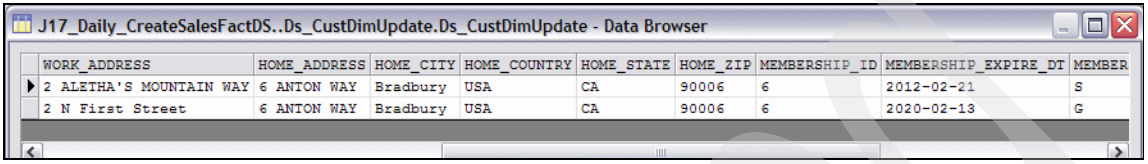
Figure 3-597 Execute the J17_DailyCreateSalesFactDS (Day 2) job (Day 2) 3/12



J17_Daily_CreateSalesFactDS..Ds_CustDimUpdate.Ds_CustDimUpdate - Data Browser

CUSTOMER_DIM_KEY	CUSTOMER_ID	NAME	HOME_PHONE	WORK_PHONE	WORK_ZIP	WORK_STATE	WORK_COUNTRY	WORK_CITY	WORK_ADDRES
836	6	Bela Davis	508-555-0782	408-555-8306	90002	CA	USA	Albany	2 ALETHA'S
843	6	Belad Davis	508-555-0782	408-555-8333	90002	CA	USA	Albany	2 N First S

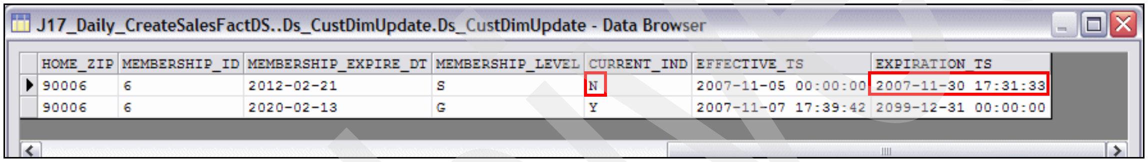
Figure 3-598 Execute the J17_DailyCreateSalesFactDS (Day 2) job (Day 2) 4/12



J17_Daily_CreateSalesFactDS..Ds_CustDimUpdate.Ds_CustDimUpdate - Data Browser

WORK_ADDRESS	HOME_ADDRESS	HOME_CITY	HOME_COUNTRY	HOME_STATE	HOME_ZIP	MEMBERSHIP_ID	MEMBERSHIP_EXPIRE_DT	MEMBER
2 ALETHA'S MOUNTAIN WAY	6 ANTON WAY	Bradbury	USA	CA	90006	6	2012-02-21	S
2 N First Street	6 ANTON WAY	Bradbury	USA	CA	90006	6	2020-02-13	G

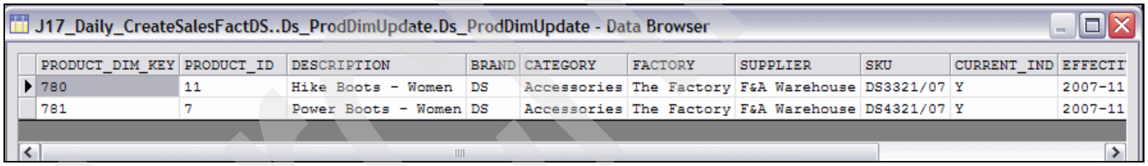
Figure 3-599 Execute the J17_DailyCreateSalesFactDS (Day 2) job (Day 2) 5/12



J17_Daily_CreateSalesFactDS..Ds_CustDimUpdate.Ds_CustDimUpdate - Data Browser

HOME_ZIP	MEMBERSHIP_ID	MEMBERSHIP_EXPIRE_DT	MEMBERSHIP_LEVEL	CURRENT_IND	EFFECTIVE_TS	EXPIRATION_TS
90006	6	2012-02-21	S	N	2007-11-05 00:00:00	2007-11-30 17:31:33
90006	6	2020-02-13	G	Y	2007-11-07 17:39:42	2099-12-31 00:00:00

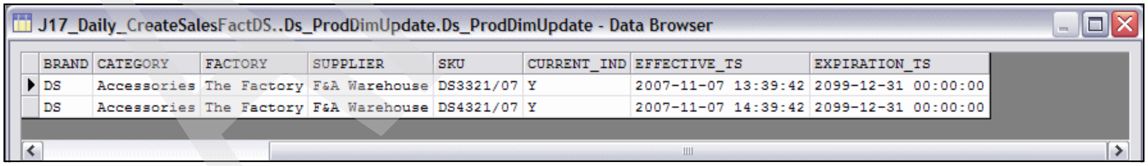
Figure 3-600 Execute the J17_DailyCreateSalesFactDS (Day 2) job (Day 2) 6/12



J17_Daily_CreateSalesFactDS..Ds_ProdDimUpdate.Ds_ProdDimUpdate - Data Browser

PRODUCT_DIM_KEY	PRODUCT_ID	DESCRIPTION	BRAND	CATEGORY	FACTORY	SUPPLIER	SKU	CURRENT_IND	EFFECTI
780	11	Hike Boots - Women	DS	Accessories	The Factory	F&A Warehouse	DS3321/07	Y	2007-11
781	7	Power Boots - Women	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	Y	2007-11

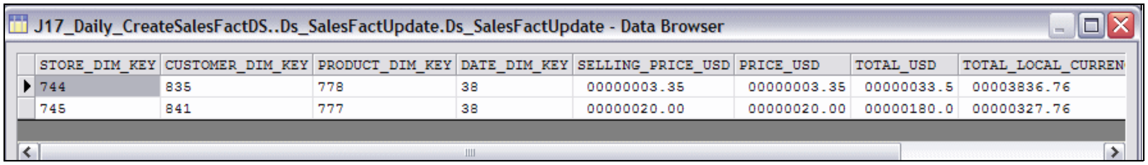
Figure 3-601 Execute the J17_DailyCreateSalesFactDS (Day 2) job (Day 2) 7/12



J17_Daily_CreateSalesFactDS..Ds_ProdDimUpdate.Ds_ProdDimUpdate - Data Browser

BRAND	CATEGORY	FACTORY	SUPPLIER	SKU	CURRENT_IND	EFFECTIVE_TS	EXPIRATION_TS
DS	Accessories	The Factory	F&A Warehouse	DS3321/07	Y	2007-11-07 13:39:42	2099-12-31 00:00:00
DS	Accessories	The Factory	F&A Warehouse	DS4321/07	Y	2007-11-07 14:39:42	2099-12-31 00:00:00

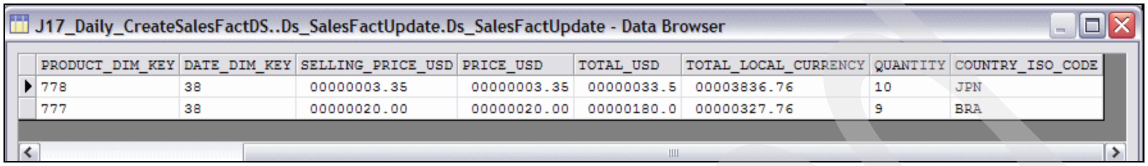
Figure 3-602 Execute the J17_DailyCreateSalesFactDS (Day 2) job (Day 2) 8/12



J17_Daily_CreateSalesFactDS..Ds_SalesFactUpdate.Ds_SalesFactUpdate - Data Browser

STORE_DIM_KEY	CUSTOMER_DIM_KEY	PRODUCT_DIM_KEY	DATE_DIM_KEY	SELLING_PRICE_USD	PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURREN
744	835	778	38	00000003.35	00000003.35	00000033.5	00003836.76
745	841	777	38	00000020.00	00000020.00	00000180.0	00000327.76

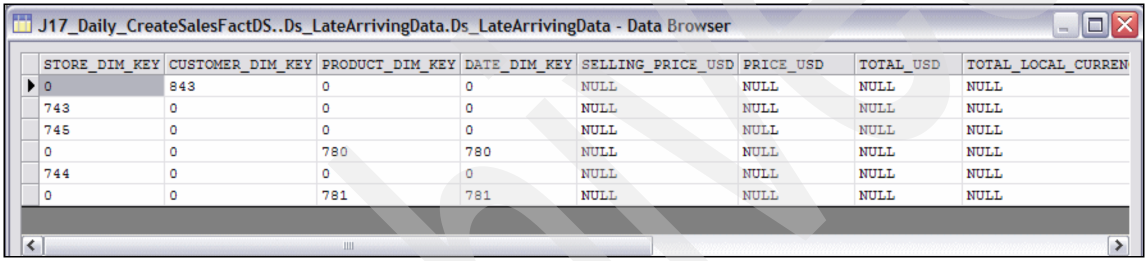
Figure 3-603 Execute the J17_DailyCreateSalesFactDS (Day 2) job (Day 2) 9/12



J17_Daily_CreateSalesFactDS..Ds_SalesFactUpdate.Ds_SalesFactUpdate - Data Browser

PRODUCT_DIM_KEY	DATE_DIM_KEY	SELLING_PRICE_USD	PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURRENCY	QUANTITY	COUNTRY_ISO_CODE
778	38	00000003.35	00000003.35	00000033.5	00003836.76	10	JPN
777	38	00000020.00	00000020.00	00000180.0	00000327.76	9	BRA

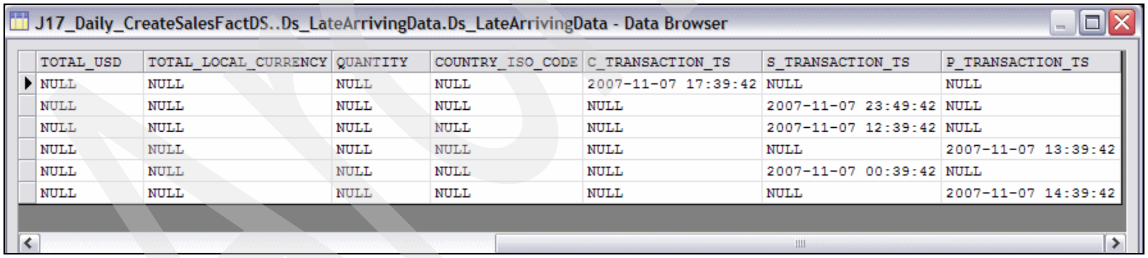
Figure 3-604 Execute the J17_DailyCreateSalesFactDS (Day 2) job (Day 2) 10/12



J17_Daily_CreateSalesFactDS..Ds_LateArrivingData.Ds_LateArrivingData - Data Browser

STORE_DIM_KEY	CUSTOMER_DIM_KEY	PRODUCT_DIM_KEY	DATE_DIM_KEY	SELLING_PRICE_USD	PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURREN
0	843	0	0	NULL	NULL	NULL	NULL
743	0	0	0	NULL	NULL	NULL	NULL
745	0	0	0	NULL	NULL	NULL	NULL
0	0	780	780	NULL	NULL	NULL	NULL
744	0	0	0	NULL	NULL	NULL	NULL
0	0	781	781	NULL	NULL	NULL	NULL

Figure 3-605 Execute the J17_DailyCreateSalesFactDS (Day 2) job (Day 2) 11/12



J17_Daily_CreateSalesFactDS..Ds_LateArrivingData.Ds_LateArrivingData - Data Browser

TOTAL_USD	TOTAL_LOCAL_CURRENCY	QUANTITY	COUNTRY_ISO_CODE	C_TRANSACTION_TS	S_TRANSACTION_TS	P_TRANSACTION_TS
NULL	NULL	NULL	NULL	2007-11-07 17:39:42	NULL	NULL
NULL	NULL	NULL	NULL	NULL	2007-11-07 23:49:42	NULL
NULL	NULL	NULL	NULL	NULL	2007-11-07 12:39:42	NULL
NULL	NULL	NULL	NULL	NULL	NULL	2007-11-07 13:39:42
NULL	NULL	NULL	NULL	NULL	2007-11-07 00:39:42	NULL
NULL	NULL	NULL	NULL	NULL	NULL	2007-11-07 14:39:42

Figure 3-606 Execute the J17_DailyCreateSalesFactDS (Day 2) job (Day 2) 12/12

J18_Daily_UpdateStoreDim (Day 2) execution

Figure 3-607 on page 530 through Figure 3-609 on page 531 show the results of the execution of this job with Day 2 data described earlier.

- Figure 3-607 on page 530 shows the results of the execution. It accepts 4 rows as input from the “J17_DailyCreateSalesFactDS (Day 2) execution” on page 526 job as seen in Figure 3-596 on page 527 and Figure 3-597 on page 527.

- The outputs are as follows:
  - There are no rows written to the Rej_StoreDim link.
  - The 4 rows written to the Odbc_StoreDim link updates the STORE_DIM dimension table with these changes (as highlighted) as seen in Figure 3-608 and Figure 3-609 on page 531.

The next step is to execute the job described in “J19_Daily_UpdateCustomerDim (Day 2) execution” on page 531.

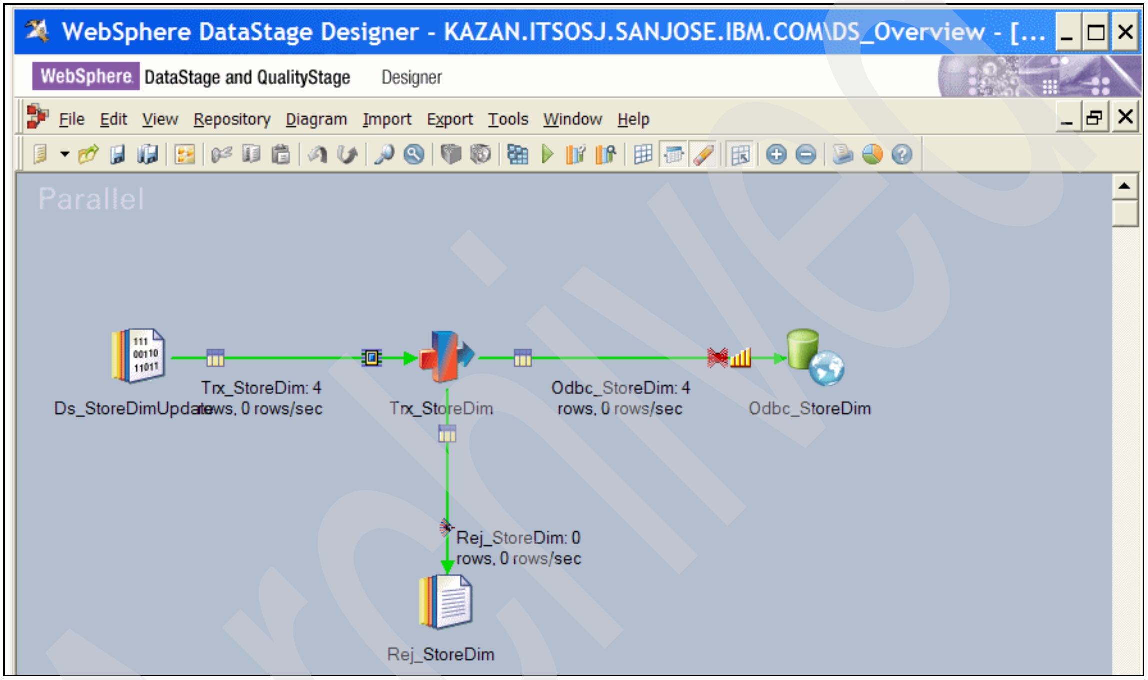
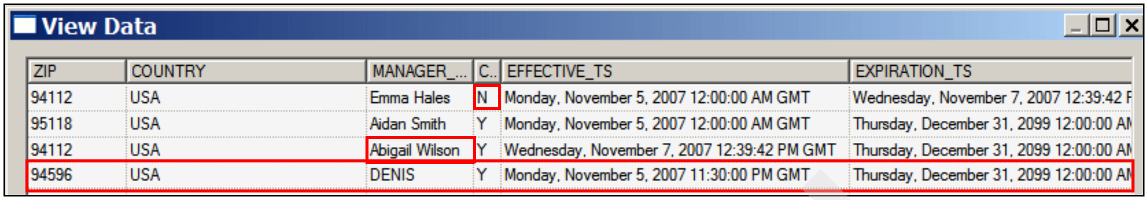


Figure 3-607 Execute the J18_Daily_UpdateStoreDim job (Day 2) 1/3

View Data

S...	S...	ADDRESS	CITY	CITY_P...	S...	STATE_...	ZIP	COUNTRY
742	33	8976 Brazil Ave	San Francisco	744041	CA	33871648	94112	USA
743	1	12345 Almaden Expressway	San Jose	929936	CA	37700000	95118	USA
745	33	8976 Brazil Ave	San Francisco	744041	CA	37700000	94112	USA
746	9	34567 North Main Street	Walnut Creek	66111	CA	37700000	94596	USA

Figure 3-608 Execute the J18_Daily_UpdateStoreDim job (Day 2) 2/3


View Data

ZIP	COUNTRY	MANAGER_...	C..	EFFECTIVE_TS	EXPIRATION_TS
94112	USA	Emma Hales	N	Monday, November 5, 2007 12:00:00 AM GMT	Wednesday, November 7, 2007 12:39:42 F
95118	USA	Aidan Smith	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM
94112	USA	Abigail Wilson	Y	Wednesday, November 7, 2007 12:39:42 PM GMT	Thursday, December 31, 2099 12:00:00 AM
94596	USA	DENIS	Y	Monday, November 5, 2007 11:30:00 PM GMT	Thursday, December 31, 2099 12:00:00 AM

Figure 3-609 Execute the J18_Daily_UpdateStoreDim job (Day 2) 3/3

J19_Daily_UpdateCustomerDim (Day 2) execution

Figure 3-607 on page 530 shows the results of the execution of this job with Day 2 data described earlier.

- Figure 3-607 on page 530 shows the results of the execution. It accepts 4 rows as input from the “J17_DailyCreateSalesFactDS (Day 2) execution” on page 526 job as seen in Figure 3-598 on page 528 through Figure 3-600 on page 528.
- The outputs are as follows:
  - There are no rows written to the Rej_StoreDim link.
  - The 4 rows written to the Odbc_StoreDim link updates the STORE_DIM dimension table with these changes (as highlighted) as seen in Figure 3-608 on page 530 and Figure 3-609.

The next step is to execute the job described in “J20_Daily_UpdateProductDim (Day 2) execution” on page 533.

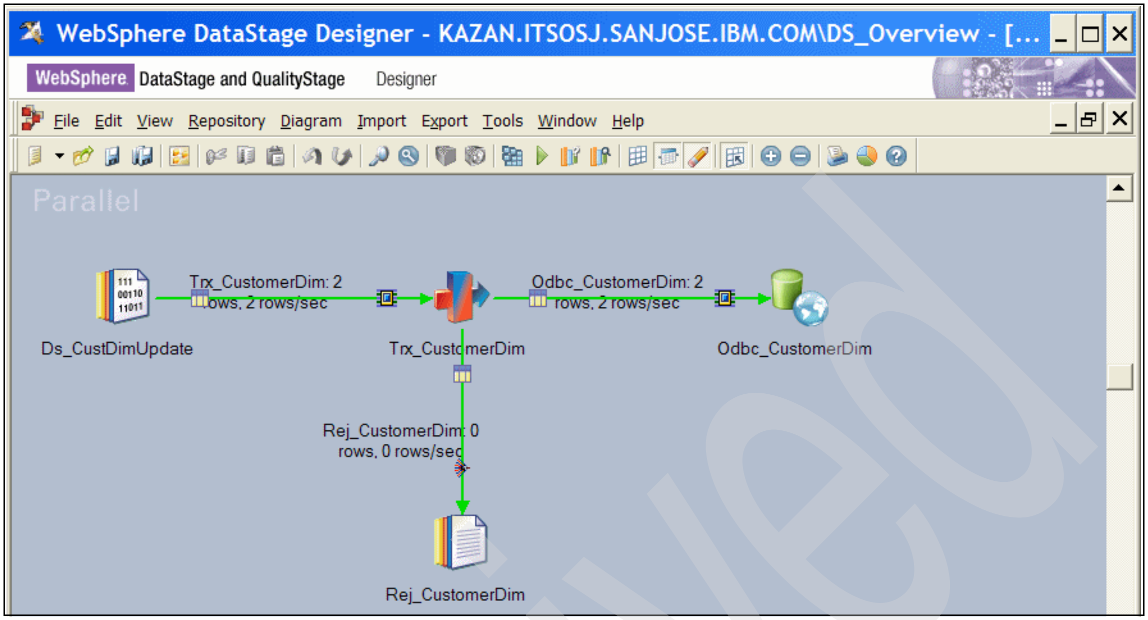


Figure 3-610 Execute the J19_Daily_UpdateCustomerDim job (Day 2) 1/4

View Data

C...	CU...	NAME	HOME_PH...	WORK_PH...	WORK_ADDRESS	WORK_C...	W...	WO...	WO...	HOME_ADI
832	1	Arch Smith	508-555-0287	408-555-8801	100 AIR ROAD	Santa Cruz	CA	90001	USA	2121 Carl St
833	2	Ban Johnson	508-555-0386	408-555-8702	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	
834	3	Barn Williams	508-555-0485	408-555-8603						3 ALEX WAY
835	4	Beel Jones	508-555-0584	408-555-8504						
836	6	Bela Davis	508-555-0782	408-555-8306	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	6 ANTON W
837	7	Blair Miller	508-555-0881	408-555-8207	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	7 ASPEN W.
838	8	Mary Wilson	508-555-0980	408-555-8108	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	8 ASTORIA
839	9	Blue Moore	508-555-1079	408-555-8009	2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	9 AURIGA W
840	10	Boris Taylor	508-555-1178	408-555-7910	10 BAYLOR WAY	City	CA	90010	USA	2 ALETHA'S
841	11	Desde Lewis	508-555-2465	408-555-6623	23 BRITTANY ROCK WAY	King City	CA	90023	USA	2 ALETHA'S
842	9999	CASH CUSTOMER	555-555-5555	555-555-5555						
843	6	Belad Davis	508-555-0782	408-555-8333	2 N First Street	Albany	CA	90002	USA	6 ANTON W

Figure 3-611 Execute the J19_Daily_UpdateCustomerDim job (Day 2) 2/4

View Data

HOME_ADDRESS	HOME_CITY	HO...	H...	HO...	M..	MEMBERSHIP_EXPIRE_DT	M.	C..	EFFECTIVE_TS
2121 Carl St	Santa Cruz	90001	CA	USA	1	Thursday, February 16, 2012	S	Y	Monday, November 5, 2007
					2	Friday, February 17, 2012	S	Y	Monday, November 5, 2007
3 ALEX WAY	Amador City	90003	CA	USA	3	Saturday, February 18, 2012	S	Y	Monday, November 5, 2007
					4	Sunday, February 19, 2012	S	Y	Monday, November 5, 2007
6 ANTON WAY	Bradbury	90006	CA	USA	6	Tuesday, February 21, 2012	S	N	Monday, November 5, 2007
7 ASPEN WAY	Brawley	90007	CA	USA	7	Wednesday, February 22, 2012	S	Y	Monday, November 5, 2007
8 ASTORIA WAY	California City	90008	CA	USA	8	Thursday, February 23, 2012	S	Y	Monday, November 5, 2007
9 AURIGA WAY	Cathedral City	90009	CA	USA	9	Friday, February 24, 2012	S	Y	Monday, November 5, 2007
2 ALETHA'S MOUNTAIN WAY	Albany	90002	CA	USA	10	Saturday, February 25, 2012	S	Y	Monday, November 5, 2007
2 ALETHA'S MOUNTAIN WAY	Albany	90002	CA	USA	99	Thursday, May 10, 2012	P	Y	Monday, November 5, 2007
					0	Tuesday, December 31, 2999	P	Y	Monday, November 5, 2007
6 ANTON WAY	Bradbury	90006	CA	USA	6	Thursday, February 13, 2020	G	Y	Wednesday, November 7, 2

Figure 3-612 Execute the J19_Daily_UpdateCustomerDim job (Day 2) 3/4

View Data

MEMBERSHIP_EXPIRE_DT	M.	C..	EFFECTIVE_TS	EXPIRATION_TS
Thursday, February 16, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
Friday, February 17, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
Saturday, February 18, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
Sunday, February 19, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
Tuesday, February 21, 2012	S	N	Monday, November 5, 2007 12:00:00 AM GMT	Wednesday, November 7, 2007 5:39:42 PM GMT
Wednesday, February 22, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
Thursday, February 23, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
Friday, February 24, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
Saturday, February 25, 2012	S	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
Thursday, May 10, 2012	P	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
Tuesday, December 31, 2999	P	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
Thursday, February 13, 2020	G	Y	Wednesday, November 7, 2007 5:39:42 PM GMT	Thursday, December 31, 2099 12:00:00 AM GMT

Figure 3-613 Execute the J19_Daily_UpdateCustomerDim job (Day 2) 4/4

J20_Daily_UpdateProductDim (Day 2) execution

Figure 3-614 on page 534 shows the results of the execution of this job with Day 2 data described earlier.

- Figure 3-614 on page 534 shows the results of the execution. It accepts 2 rows as input from the "J17_DailyCreateSalesFactDS (Day 2) execution" on page 526 job as seen in Figure 3-601 on page 528 and Figure 3-602 on page 528.

- The outputs are as follows:
  – There are no rows written to the Rej_StoreDim link.
  – The 2 rows written to the Odbc_ProductDim link updates the PRODUCT_DIM dimension table with these changes (as highlighted) as seen in Figure 3-615 and Figure 3-616 on page 535.

The next step is to execute the job described in “J21_Daily_UpdateDateDim (Day 2) execution” on page 535.

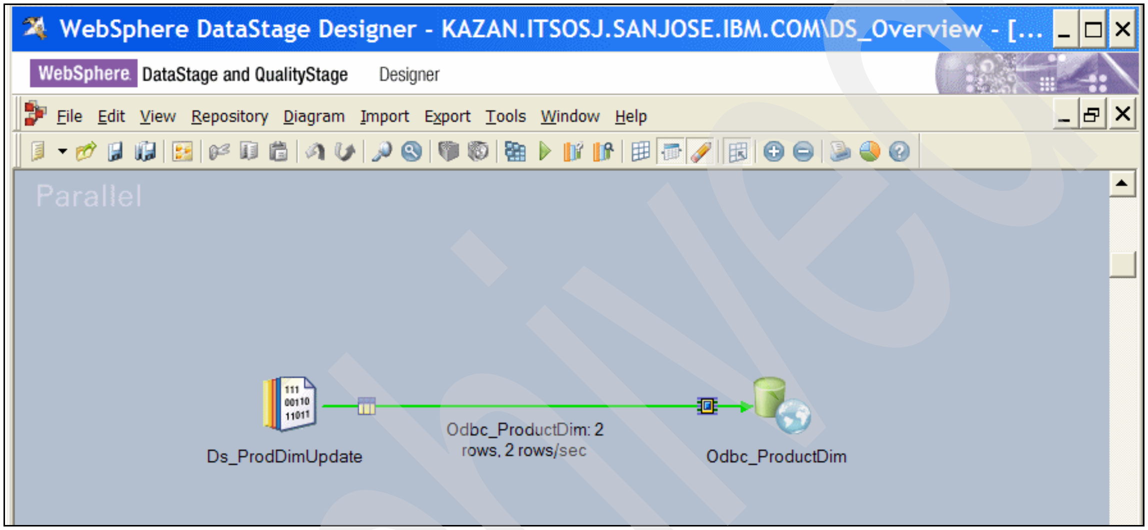
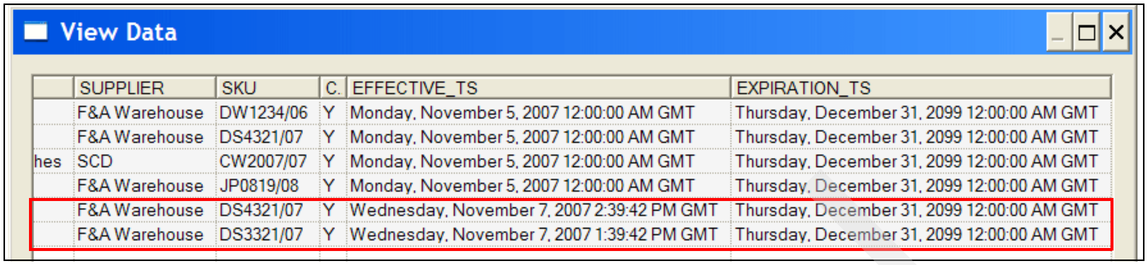


Figure 3-614 Execute the J20_Daily_UpdateProductDim job (Day 2) 1/3

View Data

P...	P...	DESCRIPTION	BRAND	CATEGORY	FACTORY	SUPPLIER	SKU	C.	EFFEC
778	4	Cowboy Hat	DFW	Accessories	Y'ALL	F&A Warehouse	DW1234/06	Y	Monday
777	1	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	Y	Monday
776	2	Santos Dummont Watch	Chrono Watches	Accessories	Chrono Watches	SCD	CW2007/07	Y	Monday
779	5	Neon Genesis Evangelion T-Shirt	JP Design	Accessories	JP Design	F&A Warehouse	JP0819/08	Y	Monday
781	7	Power Boots - Women	DS	Accessories	The Factory	F&A Warehouse	DS4321/07	Y	Wednes
780	11	Hike Boots - Women	DS	Accessories	The Factory	F&A Warehouse	DS3321/07	Y	Wednes

Figure 3-615 Execute the J20_Daily_UpdateProductDim job (Day 2) 2/3



View Data

	SUPPLIER	SKU	C.	EFFECTIVE_TS	EXPIRATION_TS
	F&A Warehouse	DW1234/06	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
	F&A Warehouse	DS4321/07	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
hes	SCD	CW2007/07	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
	F&A Warehouse	JP0819/08	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
	F&A Warehouse	DS4321/07	Y	Wednesday, November 7, 2007 2:39:42 PM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
	F&A Warehouse	DS3321/07	Y	Wednesday, November 7, 2007 1:39:42 PM GMT	Thursday, December 31, 2099 12:00:00 AM GMT

Figure 3-616 Execute the J20_Daily_UpdateProductDim job (Day 2) 3/3

J21_Daily_UpdateDateDim (Day 2) execution

Figure 3-617 shows the results of the execution of this job with Day 2 data described earlier.

It shows no input records to update the Date dimension table.

The next step is to execute the job described in "J22_Daily_UpdateSalesFact (Day 2) execution" on page 535.

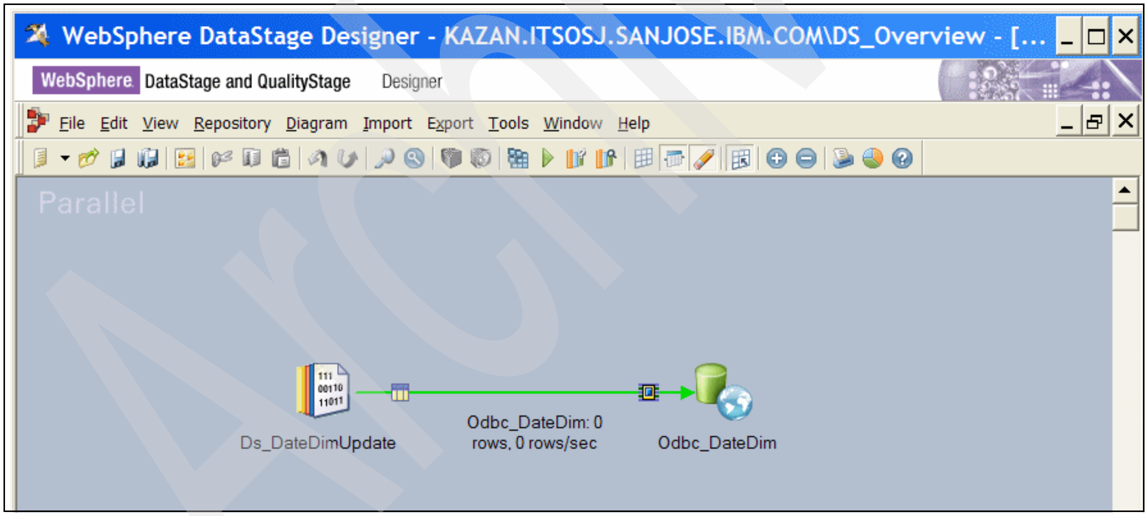


Figure 3-617 Execute the J21_Daily_UpdateDateDim job (Day 2) 1/?

J22_Daily_UpdateSalesFact (Day 2) execution

Figure 3-618 on page 536 through Figure 3-621 on page 537 show the results of the execution of this job with Day 2 data described earlier.

- Figure 3-618 shows the results of the execution. It accepts 2 rows as input from the “J17_DailyCreateSalesFactDS (Day 2) execution” on page 526 job as seen in Figure 3-603 on page 529 and Figure 3-604 on page 529.
- The output shows 2 rows being written to the Odbc_SalesFact link which is used to update the SALES_FACT table. Figure 3-619 through Figure 3-621 on page 537 show the updated contents of the SALES_FACT table as highlighted.

This concludes Day 2 processing.

You can proceed to Day 3 processing as described in 3.1.5, “Recurring tasks (Day 3)” on page 537.

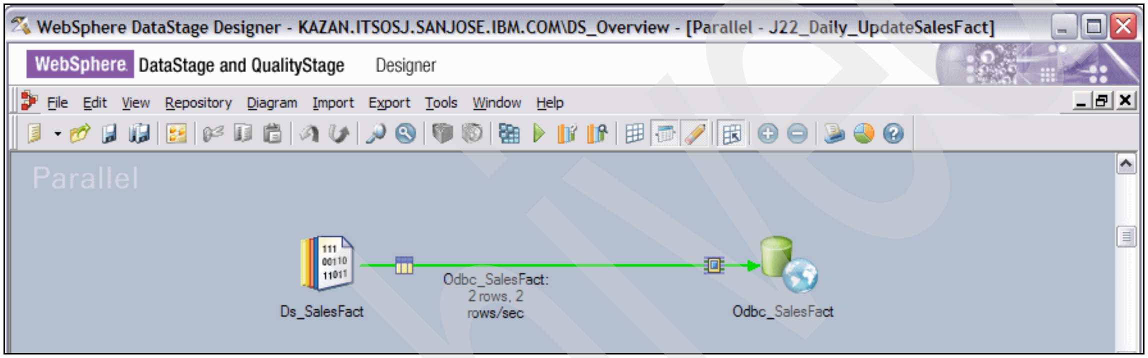


Figure 3-618 Execute the J22_Daily_UpdateSalesFact job (Day 2) 1/4

View Data

CUSTOMER_DIM_KEY	DATE_DIM_KEY	PRODUCT_DIM_KEY	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	STORE_DIM_KEY	TOTAL_LO
832	36	777	2	35.00	25.00	50.00	743	5726.50
832	37	777	2	35.00	25.00	50.00	742	50.00
835	38	778	10	3.35	3.35	33.50	744	3836.76
836	36	777	4	17.69	15.00	60.00	743	109.26
836	37	777	2	17.69	15.00	30.00	743	30.00
838	36	779	3	120.00	120.00	360.00	742	14320.80
839	37	777	1	37.00	37.00	37.00	742	37.00
840	37	776	3	37.00	37.00	111.00	742	111.00
841	38	777	9	20.00	20.00	180.00	745	327.76
842	37	776	2	17.69	15.00	30.00	743	29.02
842	37	777	1	35.00	33.33	33.33	742	33.33
842	37	777	2	17.69	15.00	30.00	743	30.00

Figure 3-619 Execute the J22_Daily_UpdateSalesFact job (Day 2) 2/4

PRODUCT_DIM_KEY	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	STORE_DIM_KEY	TOTAL_LOCAL_CURRENCY	COUNTRY_ISO_CODE
777	2	35.00	25.00	50.00	743	5726.50	JPN
777	2	35.00	25.00	50.00	742	50.00	USA
778	10	3.35	3.35	33.50	744	3836.76	JPN
777	4	17.69	15.00	60.00	743	109.26	BRA
777	2	17.69	15.00	30.00	743	30.00	USA
779	3	120.00	120.00	360.00	742	14320.80	IND
777	1	37.00	37.00	37.00	742	37.00	USA
776	3	37.00	37.00	111.00	742	111.00	USA
777	9	20.00	20.00	180.00	745	327.76	BRA
776	2	17.69	15.00	30.00	743	29.02	CAD
777	1	35.00	33.33	33.33	742	33.33	USA
777	2	17.69	15.00	30.00	743	30.00	USA

Figure 3-620 Execute the J22_Daily_UpdateSalesFact job (Day 2) 3/4

C...	D...	P...	Q...	PRICE_...	SELLIN...	TOTAL...	S...	TOTAL_...	CO...
832	36	777	2	35.00	25.00	50.00	743	5726.50	JPN
832	37	777	2	35.00	25.00	50.00	742	50.00	USA
835	38	778	10	3.35	3.35	33.50	744	3836.76	JPN
836	36	777	4	17.69	15.00	60.00	743	109.26	BRA
836	37	777	2	17.69	15.00	30.00	743	30.00	USA
838	36	779	3	120.00	120.00	360.00	742	14320.80	IND
839	37	777	1	37.00	37.00	37.00	742	37.00	USA
840	37	776	3	37.00	37.00	111.00	742	111.00	USA
841	38	777	9	20.00	20.00	180.00	745	327.76	BRA
842	37	776	2	17.69	15.00	30.00	743	29.02	CAD
842	37	777	1	35.00	33.33	33.33	742	33.33	USA
842	37	777	2	17.69	15.00	30.00	743	30.00	USA

Figure 3-621 Execute the J22_Daily_UpdateSalesFact job (Day 2) 4/4

3.1.5 Recurring tasks (Day 3)

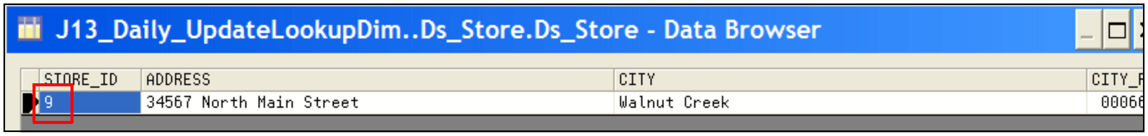
In this cycle, we processed the following data on November 8th, 2007:

- Dimension table changes:
  - Store dimension:
    - Update (TABLE_CMD value of U) of STORE_ID 9.

      The Type 2 change is MANAGER_NAME (Isabella Paris).

      There are no Type 1 changes.

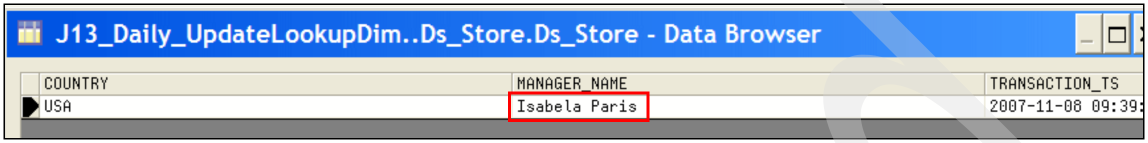
  These are shown in Figure 3-622 on page 538 through Figure 3-624 on page 538.



J13_Daily_UpdateLookupDim..Ds_Store.Ds_Store - Data Browser

STORE_ID	ADDRESS	CITY	CITY_F
9	34567 North Main Street	Walnut Creek	0006

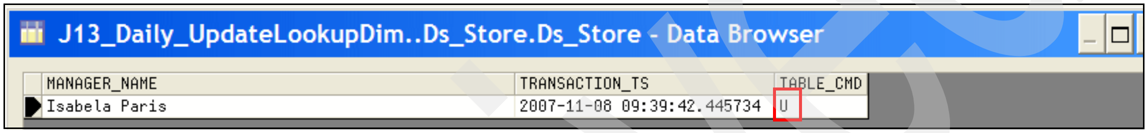
Figure 3-622 Store dimension attribute changes 1/3



J13_Daily_UpdateLookupDim..Ds_Store.Ds_Store - Data Browser

COUNTRY	MANAGER_NAME	TRANSACTION_TS
USA	Isabela Paris	2007-11-08 09:39:

Figure 3-623 Execute the J13_Daily_UpdateLookupDim job (Day 3) 2/3



J13_Daily_UpdateLookupDim..Ds_Store.Ds_Store - Data Browser

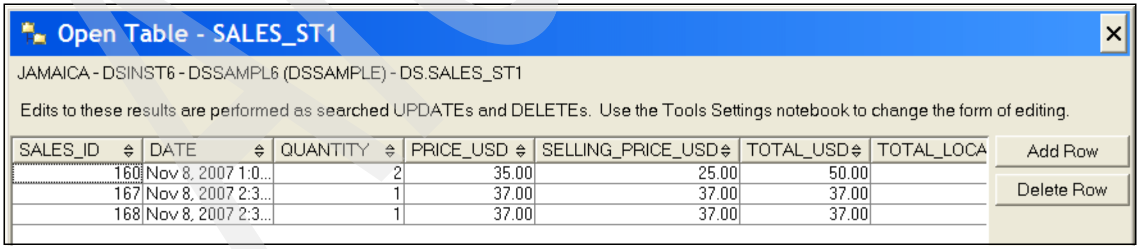
MANAGER_NAME	TRANSACTION_TS	TABLE_CMD
Isabela Paris	2007-11-08 09:39:42.445734	U

Figure 3-624 Execute the J13_Daily_UpdateLookupDim job (Day 3) 3/3

   – There are no Customer, Product, and Date dimension changes.

- Sales transactions:

  Sales transactions are collected from three stores — ST1 (STORE_ID of 1) with 3 transactions as shown in Figure 3-625 and Figure 3-626 on page 539, ST9 (STORE_ID of 9) with 1 transaction as shown in Figure 3-627 on page 539 and Figure 3-628 on page 539, and ST33 (STORE_ID of 33) with 5 transactions as shown in Figure 3-629 on page 539 and Figure 3-630 on page 540.



Open Table - SALES_ST1

JAMAICA - DSINST6 - DSSAMPL6 (DSSAMPLE) - DS.SALES_ST1

Edits to these results are performed as searched UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

SALES_ID	DATE	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	TOTAL_LOCA
160	Nov 8, 2007 1:0...	2	35.00	25.00	50.00	
167	Nov 8, 2007 2:3...	1	37.00	37.00	37.00	
168	Nov 8, 2007 2:3...	1	37.00	37.00	37.00	

Figure 3-625 STORE_ID 1 sales transactions 2/2

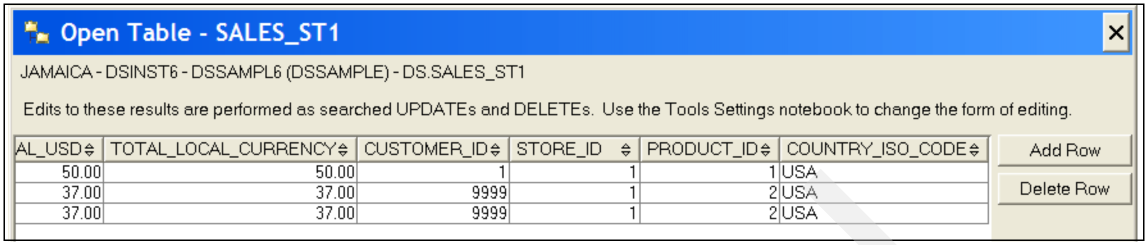


Figure 3-626 STORE_ID 1 sales transactions 2/2

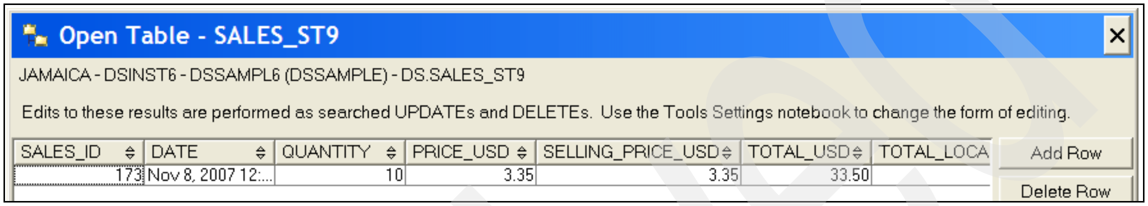


Figure 3-627 STORE_ID 9 sales transactions 1/2

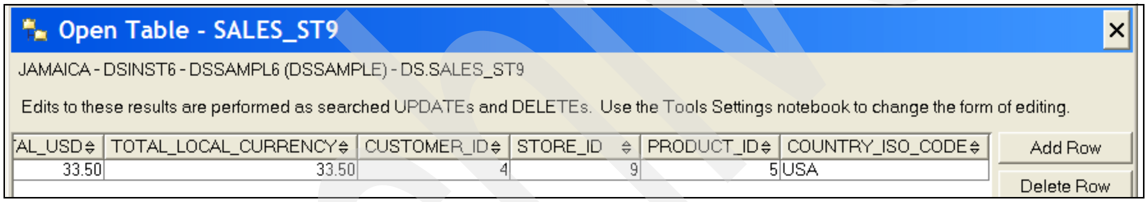


Figure 3-628 STORE_ID 9 sales transactions 2/2

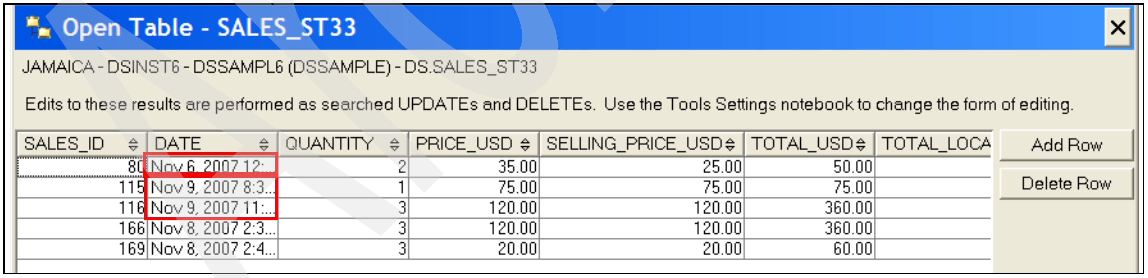
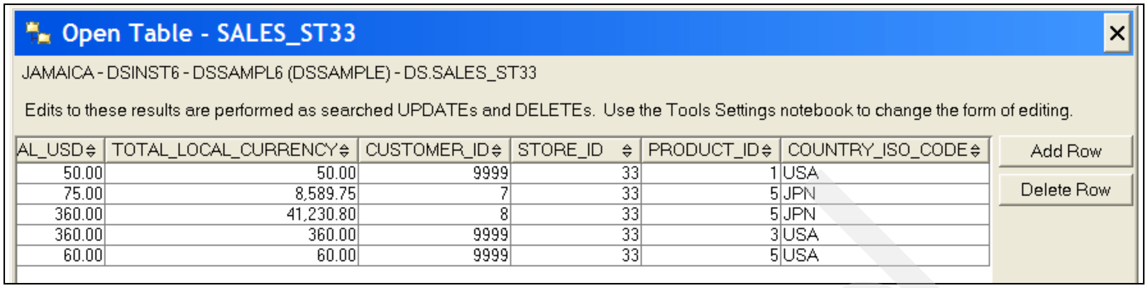


Figure 3-629 STORE_ID 33 sales transactions 1/2



AL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	COUNTRY_ISO_CODE
50.00	50.00	9999	33	1	USA
75.00	8,589.75	7	33	5	JPN
360.00	41,230.80	8	33	5	JPN
360.00	360.00	9999	33	3	USA
60.00	60.00	9999	33	5	USA

Figure 3-630 STORE_ID 33 sales transactions 2/2

Three of these sales transactions were deliberately tailored to create the following error condition, which resulted in this transactions being rejected as late arriving data.

- One sales transaction is from Store ST9 which has a date of November 6th, 2007.
- Two sales transactions are from Store ST9 which has a date of November 9th, 2007.

These fields are highlighted in Figure 3-629 on page 539.

Table 3-2 on page 342 identifies the jobs executed in the recurring (daily) tasks.

- The configuration of these tasks is briefly described in “Recurring tasks (Day 1)” on page 348.
- The execution of these jobs and the corresponding recurring tasks (Day 3) are briefly described in the following sections starting with “J07_IL_Daily_LoadSalesStore (Day 3) execution” on page 541.

Note: “J06_IL_Daily_CreateCurrencyLookup_Service” on page 227 should be executed every day to pick up the latest exchange rates for each ISO country code. In our case, however, we created all the exchange rates for the different ISO country code countries for our three recurring daily cycles up front (during the initial load phase), and therefore do not repeat it here.

J07_IL_Daily_LoadSalesStore (Day 3) execution

This job has to be repeated for sales transactions for each of the three stores (1, 9, and 33) for Day 2.

- Figure 3-631 on page 542 shows the Job Run Options window that identifies the input file (J07_Seq_Sales_20071108_ST1.txt) containing the sales transactions, the name of the schema file (J07_Seq_Sales_schema.osh), and the name of the interim DB2 table (DS.SALES_ST1) to which these sales transactions are written.

  Figure 3-632 on page 542 shows the execution results of this job, indicating 3 sales transactions being processed.

  The contents of the DB2 interim table after the execution are shown in Figure 3-625 on page 538 and Figure 3-626 on page 539.

- Figure 3-633 on page 543 shows the Job Run Options window that identifies the input file (J07_Seq_Sales_20071108_ST9.txt) containing the sales transactions, the name of the schema file (J07_Seq_Sales_schema.osh), and the name of the interim DB2 table (DS.SALES_ST9) to which these sales transactions are written.

  Figure 3-634 on page 543 shows the execution results of this job, indicating 1 sales transaction being processed.

  The contents of the DB2 interim table after the execution are shown in Figure 3-627 on page 539 and Figure 3-628 on page 539.

- Figure 3-635 on page 544 shows the Job Run Options window that identifies the input file (J07_Seq_Sales_20071108_ST33.txt) containing the sales transactions, the name of the schema file (J07_Seq_Sales_schema.osh), and the name of the interim DB2 table (DS.SALES_ST33) to which these sales transactions are written.

  Figure 3-636 on page 544 shows the execution results of this job, indicating 5 sales transactions being processed.

  The contents of the DB2 interim table after the execution are shown in Figure 3-629 on page 539 and Figure 3-630 on page 540.

The next step is to execute the job described in "J14_Daily_CreateAllSalesStoreDS (Day 3) execution" on page 546.

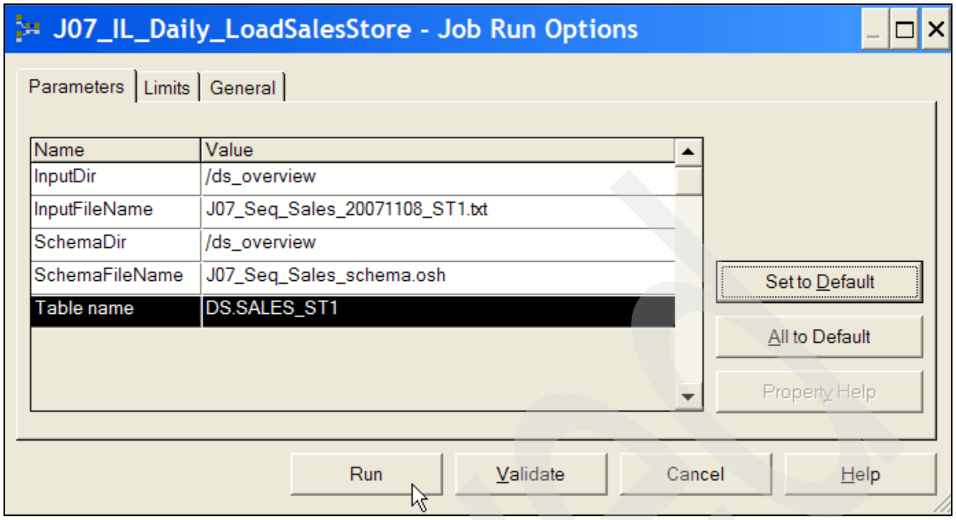


Figure 3-631 Execute the J07_IL_Daily_LoadSalesStore job (Day 3) 1/6

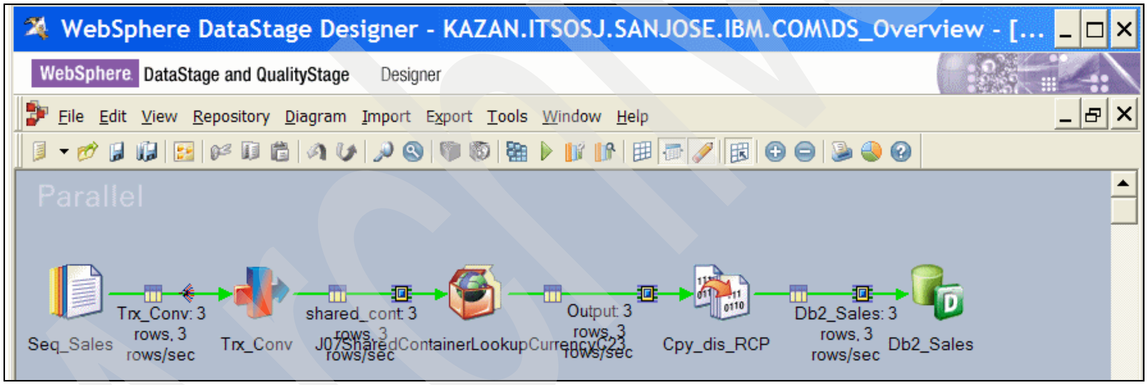


Figure 3-632 Execute the J07_IL_Daily_LoadSalesStore job (Day 3) 2/6

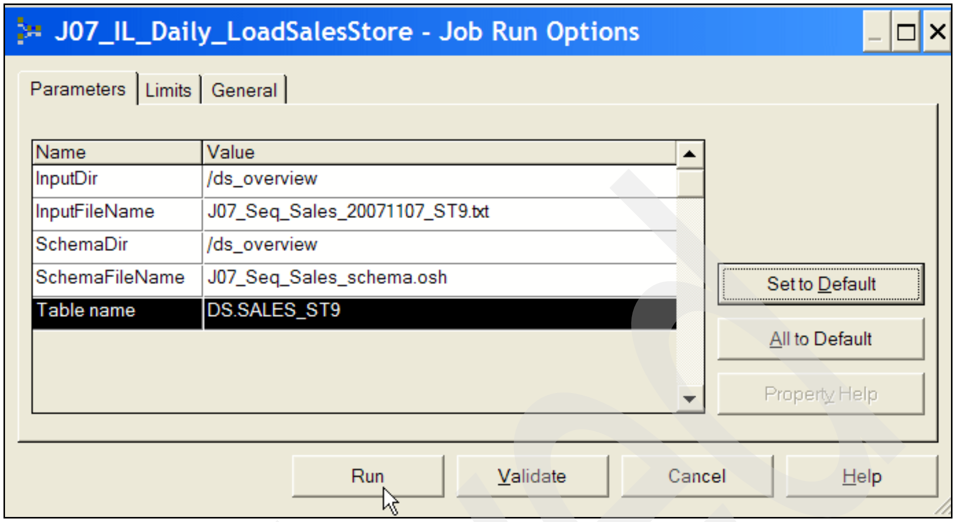


Figure 3-633 Execute the J07_IL_Daily_LoadSalesStore job (Day 3) 3/6

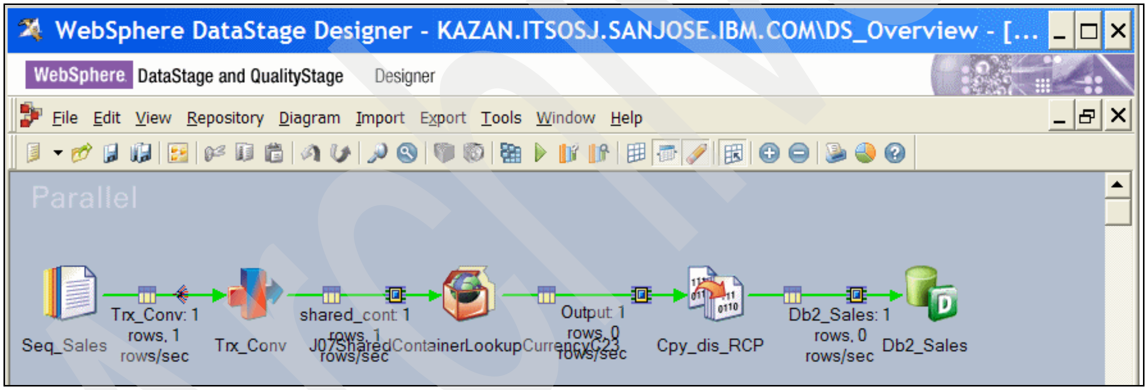


Figure 3-634 Execute the J07_IL_Daily_LoadSalesStore job (Day 3) 4/6

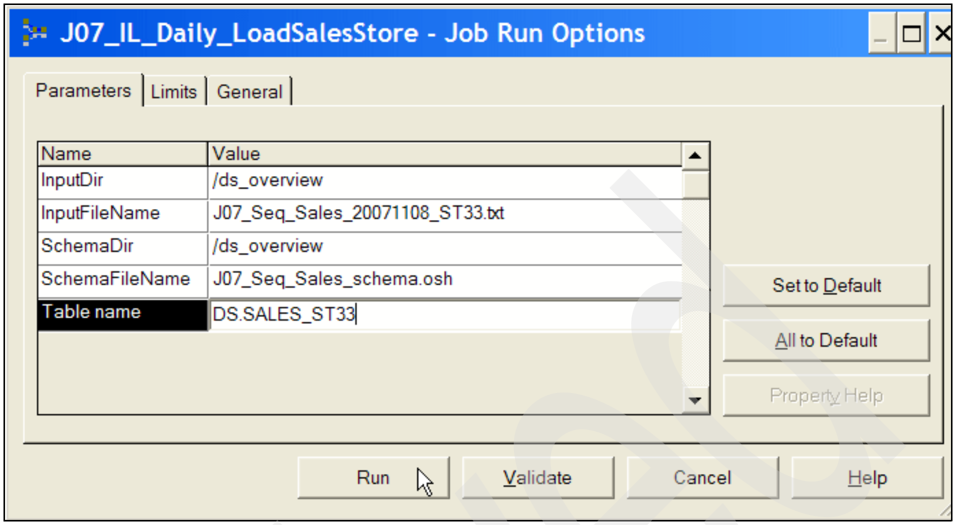


Figure 3-635 Execute the J07_IL_Daily_LoadSalesStore job (Day 3) 5/6

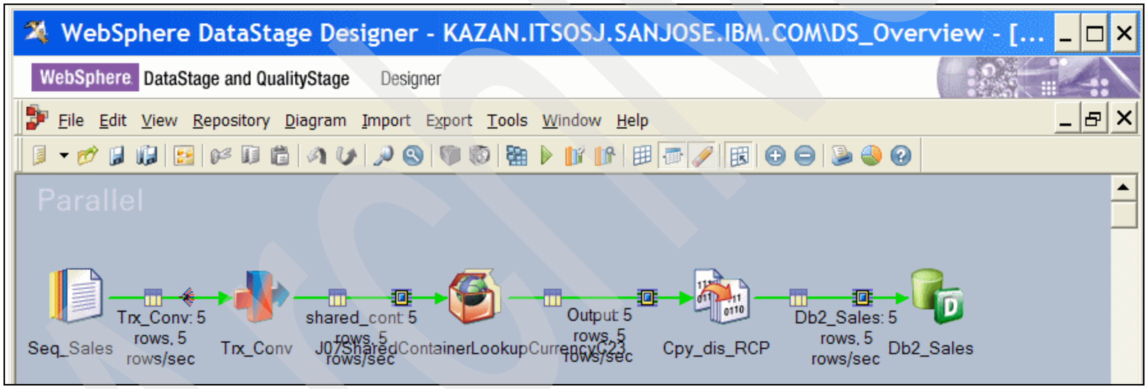


Figure 3-636 Execute the J07_IL_Daily_LoadSalesStore job (Day 3) 6/6

J13_Daily_UpdateLookupDim (Day 3) execution

Figure 3-637 on page 545 through Figure 3-640 on page 546 show the results of the execution of this job with Day 3 data described earlier.

- Figure 3-637 on page 545 shows the results of the execution. It accepts 1 row as input from the IBM WebSphere MQ message queue which is a change to the Store dimension table. This change is written to the Store_IDs data set as shown in Figure 3-622 on page 538 through Figure 3-624 on page 538.

- Figure 3-638 through Figure 3-640 on page 546 show the LOOKUP_STORE_DIM table that incorporates the changes (highlighted) due to the update of STORE_ID 9.

The next step is to execute the job described in "J14_Daily_CreateAllSalesStoreDS (Day 3) execution" on page 546.

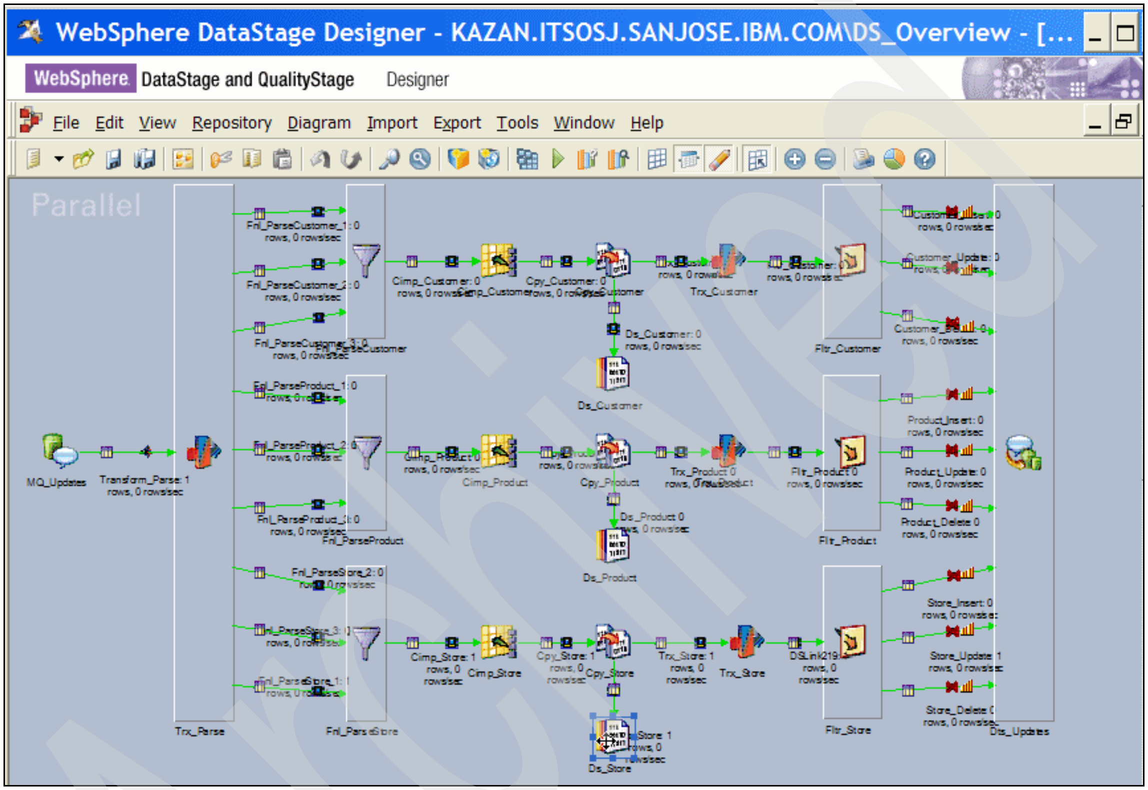
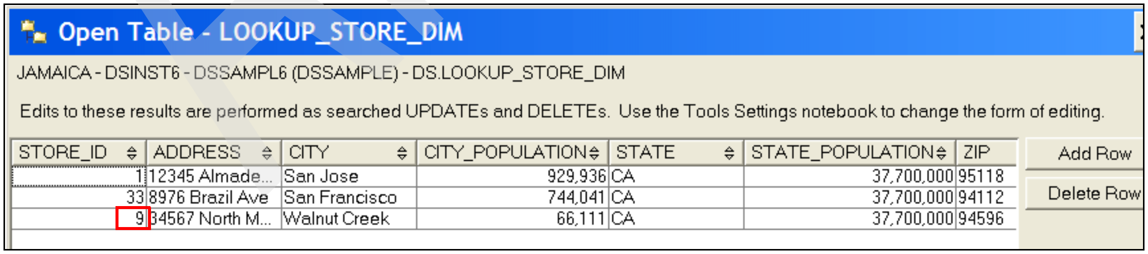


Figure 3-637 Execute the J13_Daily_UpdateLookupDim job (Day 3) 1/4



STORE_ID	ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP
1	12345 Almade...	San Jose	929,936	CA	37,700,000	95118
33	8976 Brazil Ave	San Francisco	744,041	CA	37,700,000	94112
9	34567 North M...	Walnut Creek	66,111	CA	37,700,000	94596

Figure 3-638 Execute the J13_Daily_UpdateLookupDim job (Day 3) 2/4

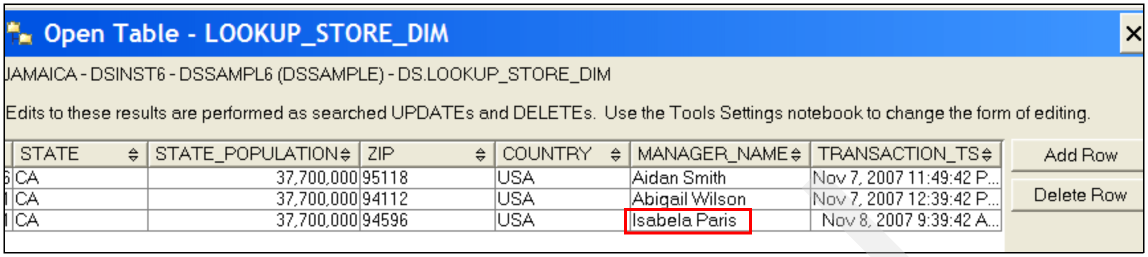


Figure 3-639 Execute the J13_Daily_UpdateLookupDim job (Day 3) 3/4

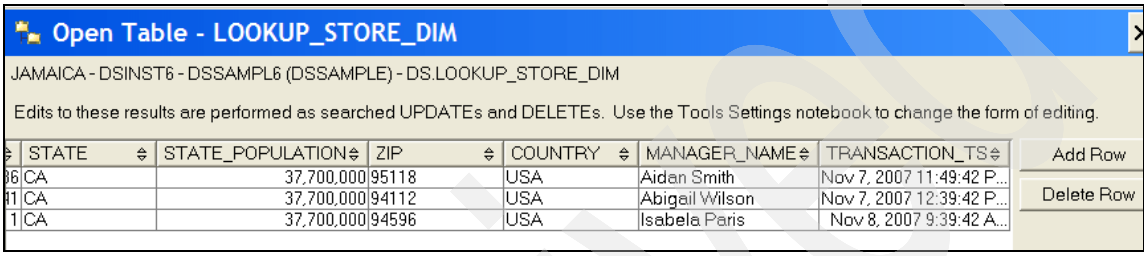


Figure 3-640 Execute the J13_Daily_UpdateLookupDim job (Day 3) 4/4

J14_Daily_CreateAllSalesStoreDS (Day 3) execution

Figure 3-641 on page 547 through Figure 3-643 on page 547 show the results of the execution of this job with Day 3 data described earlier.

- Figure 3-641 on page 547 shows the results of the execution. It accepts 3 rows from store 1, five rows from store 9, and five rows from store 33 for a total of 9 rows that are written to the output data set.
- Figure 3-642 on page 547 and Figure 3-643 on page 547 show the contents of the output data set DS_AllSales.

The next step is to execute the job described in “J15_Daily_CreateSalesAggDS (Day 3) execution” on page 548.

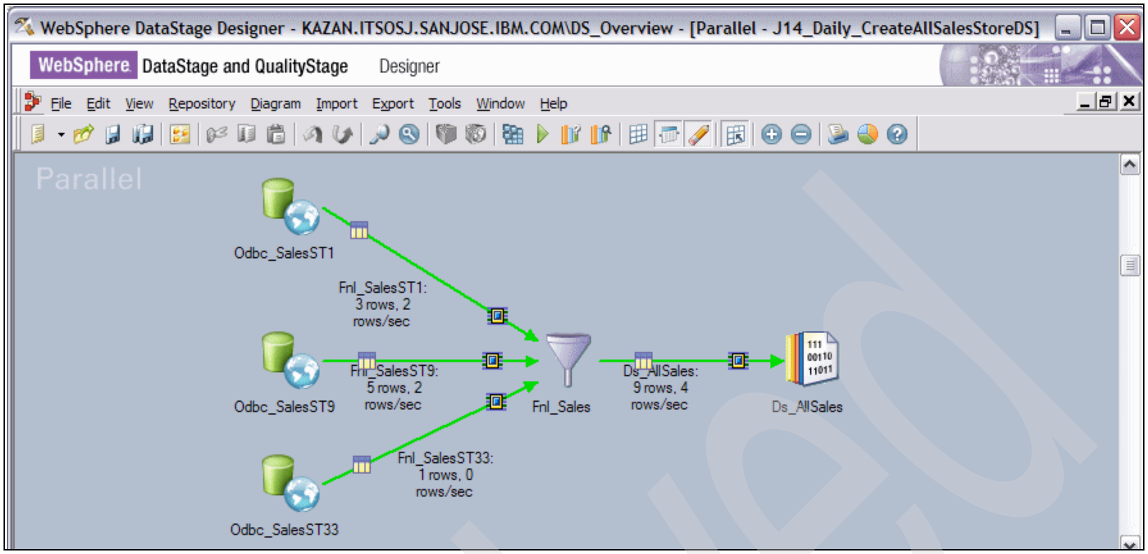


Figure 3-641 Execute the J14_Daily_CreateAllSalesStoreDS job (Day 3) 1/3

J14_Daily_CreateAllSalesStoreDS..Ds_AllSales.Ds_AllSales - Data Browser

SALES_ID	DATE	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID
115	2007-11-09 08:30:00.000000	1	00000075.00	00000075.00	00000075.00	00008589.75	7
169	2007-11-08 14:45:45.000000	3	00000020.00	00000020.00	00000060.00	00000060.00	9999
167	2007-11-08 14:39:22.000000	1	00000037.00	00000037.00	00000037.00	00000037.00	9999
116	2007-11-09 11:02:11.000000	3	00000120.00	00000120.00	00000360.00	00041230.80	8
168	2007-11-08 14:39:22.000000	1	00000037.00	00000037.00	00000037.00	00000037.00	9999
173	2007-11-08 12:00:42.000000	10	00000003.35	00000003.35	00000033.50	00000033.50	4
80	2007-11-06 12:39:11.000000	2	00000035.00	00000025.00	00000050.00	00000050.00	9999
166	2007-11-08 14:30:12.000000	3	00000120.00	00000120.00	00000360.00	00000360.00	9999
160	2007-11-08 13:00:02.000000	2	00000035.00	00000025.00	00000050.00	00000050.00	1

Figure 3-642 Execute the J14_Daily_CreateAllSalesStoreDS job (Day 3) 2/3

J14_Daily_CreateAllSalesStoreDS..Ds_AllSales.Ds_AllSales - Data Browser

PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	COUNTRY_ISO_CODE
00000075.00	00000075.00	00000075.00	00008589.75	7	33	5	JPN
00000020.00	00000020.00	00000060.00	00000060.00	9999	33	5	USA
00000037.00	00000037.00	00000037.00	00000037.00	9999	1	2	USA
00000120.00	00000120.00	00000360.00	00041230.80	8	33	5	JPN
00000037.00	00000037.00	00000037.00	00000037.00	9999	1	2	USA
00000003.35	00000003.35	00000033.50	00000033.50	4	9	5	USA
00000035.00	00000025.00	00000050.00	00000050.00	9999	33	1	USA
00000120.00	00000120.00	00000360.00	00000360.00	9999	33	3	USA
00000035.00	00000025.00	00000050.00	00000050.00	1	1	1	USA

Figure 3-643 Execute the J14_Daily_CreateAllSalesStoreDS job (Day 3) 3/3

J15_Daily_CreateSalesAggDS (Day 3) execution

Figure 3-644 through Figure 3-656 on page 551 show the results of the execution of this job with Day 3 data described earlier.

- Figure 3-644 shows the results of the execution. It accepts 9 rows as input from the "J14_Daily_CreateAllSalesStoreDS (Day 3) execution" on page 546 job as seen in Figure 3-642 on page 547 and Figure 3-643 on page 547.
- The two outputs of this job are:
  - The aggregated sales transactions appended with the dimension lookup tables. This is a total of 5 rows as seen in Figure 3-645 on page 549 through Figure 3-650 on page 550.
  - The rejected sales transactions (either late arriving dimensions or late arriving data). This is a total of 3 rows as seen in Figure 3-651 on page 550 through Figure 3-656 on page 551. The invalid dates are highlighted.

The next step is to execute the job described in "J16_Daily_CreateScdInputDS (Day 3) execution" on page 552.

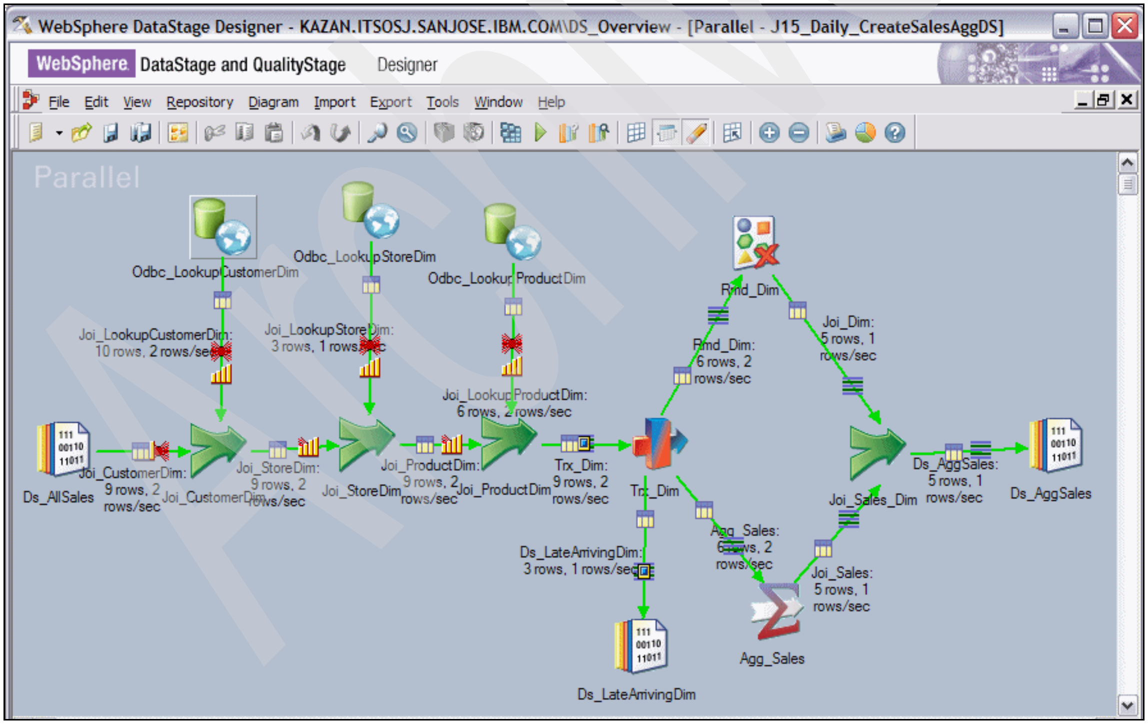


Figure 3-644 Execute the J15_Daily_CreateSalesAggDS job (Day 3) 1/13

J15_Daily_CreateSalesAggDS..Ds_AggSales.Ds_AggSales - Data Browser

DATE	QUANTITY	TOTAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	MEMBERSHIP_EXPIRE_DT	MEMBERSHIP_LE
2007-11-08	2	00000050.00	00000050.00	1	1	1	2012-02-16	S
2007-11-08	2	00000074.00	00000074.00	9999	1	2	2999-12-31	P
2007-11-08	10	00000033.50	00000033.50	4	9	5	2012-02-19	S
2007-11-09	3	00000360.00	00041230.80	8	33	5	2012-02-23	S
2007-11-08	3	00000060.00	00000060.00	9999	33	5	2999-12-31	P

Figure 3-645 Execute the J15_Daily_CreateSalesAggDS job (Day 3) 2/13

J15_Daily_CreateSalesAggDS..Ds_AggSales.Ds_AggSales - Data Browser

MEMBERSHIP_LEVEL	MANAGER_NAME	DESCRIPTION	BRAND	CATEGORY	FACTORY	SUPPLIER	S
S	Aidan Smith	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	D
P	Aidan Smith	Santos Dummont Watch	Chrono Watches	Accessories	Chrono Watches	SCD	C
S	Isabela Paris	Neon Genesis Evangelion T-Shirt	JP Design	Accessories	JP Design	F&A Warehouse	J
S	Abigail Wilson	Neon Genesis Evangelion T-Shirt	JP Design	Accessories	JP Design	F&A Warehouse	J
P	Abigail Wilson	Neon Genesis Evangelion T-Shirt	JP Design	Accessories	JP Design	F&A Warehouse	J

Figure 3-646 Execute the J15_Daily_CreateSalesAggDS job (Day 3) 3/13

J15_Daily_CreateSalesAggDS..Ds_AggSales.Ds_AggSales - Data Browser

SKU	PRICE_USD	SELLING_PRICE_USD	COUNTRY_ISO_CODE	NAME	HOME_PHONE	WORK_PHONE	WORK_ADDRESS
DS4321/07	00000035.00	00000025.00	USA	Arch Smith	508-555-0287	408-555-8801	100 AIR ROAD
CW2007/07	00000037.00	00000037.00	USA	CASH CUSTOMER	555-555-5555	555-555-5555	
JP0819/08	00000003.35	00000003.35	USA	Beel Jones	508-555-0584	408-555-8504	
JP0819/08	00000120.00	00000120.00	JPN	Mary Wilson	508-555-0980	408-555-8108	2 ALETHA'S MOUNTAIN WAY
JP0819/08	00000020.00	00000020.00	USA	CASH CUSTOMER	555-555-5555	555-555-5555	

Figure 3-647 Execute the J15_Daily_CreateSalesAggDS job (Day 3) 4/13

J15_Daily_CreateSalesAggDS..Ds_AggSales.Ds_AggSales - Data Browser

WORK_ADDRESS	WORK_CITY	WORK_STATE	WORK_ZIP	WORK_COUNTRY	HOME_ADDRESS	HOME_CITY	HOME_ZIP	HOME_STATE	HOM
100 AIR ROAD	Santa Cruz	CA	90001	USA	2121 Carl St	Santa Cruz	90001	CA	USA
2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	8 ASTORIA WAY	California City	90008	CA	USA

Figure 3-648 Execute the J15_Daily_CreateSalesAggDS job (Day 3) 5/13

J15_Daily_CreateSalesAggDS..Ds_AggSales.Ds_AggSales - Data Browser

HOME_COUNTRY	MEMBERSHIP_ID	ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	C_
USA	1	12345 Almaden Expressway	San Jose	00929936.	CA	37700000.	95118	USA	20
	0	12345 Almaden Expressway	San Jose	00929936.	CA	37700000.	95118	USA	20
	4	34567 North Main Street	Walnut Creek	00066111.	CA	37700000.	94596	USA	20
USA	8	8976 Brazil Ave	San Francisco	00744041.	CA	37700000.	94112	USA	20
	0	8976 Brazil Ave	San Francisco	00744041.	CA	37700000.	94112	USA	20

Figure 3-649 Execute the J15_Daily_CreateSalesAggDS job (Day 3) 6/13

J15_Daily_CreateSalesAggDS..Ds_AggSales.Ds_AggSales - Data Browser

STATE	STATE_POPULATION	ZIP	COUNTRY	C_TRANSACTION_TS	S_TRANSACTION_TS	P_TRANSACTION_TS
CA	37700000.	95118	USA	2007-11-06 12:39:42.445734	2007-11-07 23:49:42.445734	2007-11-05 00:00:00.000000
CA	37700000.	95118	USA	2007-11-05 00:00:00.000000	2007-11-07 23:49:42.445734	2007-11-05 00:00:00.000000
CA	37700000.	94596	USA	2007-11-05 00:00:00.000000	2007-11-08 09:39:42.445734	2007-11-05 00:00:00.000000
CA	37700000.	94112	USA	2007-11-05 00:00:00.000000	2007-11-07 12:39:42.445734	2007-11-05 00:00:00.000000
CA	37700000.	94112	USA	2007-11-05 00:00:00.000000	2007-11-07 12:39:42.445734	2007-11-05 00:00:00.000000

Figure 3-650 Execute the J15_Daily_CreateSalesAggDS job (Day 3) 7/13

J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

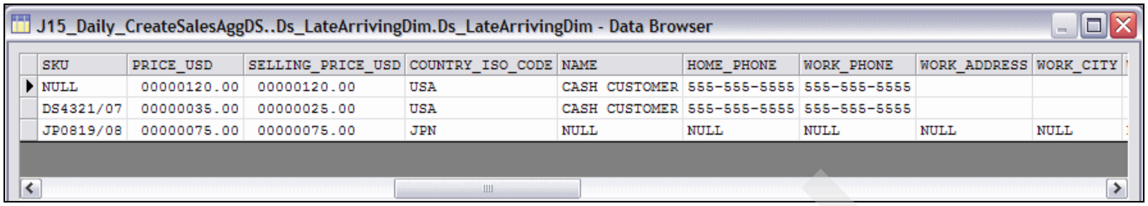
DATE	QUANTITY	TOTAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	MEMBERSHIP_EXPIRE_DT	MEMB
2007-11-08 14:30:12	3	00000360.00	00000360.00	9999	33	3	2999-12-31	P
2007-11-06 12:39:11	2	00000050.00	00000050.00	9999	33	1	2999-12-31	P
2007-11-09 08:30:00	1	00000075.00	00008589.75	7	33	5	NULL	NULL

Figure 3-651 Execute the J15_Daily_CreateSalesAggDS job (Day 3) 8/13

J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

MEMBERSHIP_LEVEL	MANAGER_NAME	DESCRIPTION	BRAND	CATEGORY	FACTORY	SUPPLIER	SKU
P	Abigail Wilson	NULL	NULL	NULL	NULL	NULL	NULL
P	Abigail Wilson	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	DS4321/07
NULL	Abigail Wilson	Neon Genesis Evangelion T-Shirt	JP Design	Accessories	JP Design	F&A Warehouse	JP0819/08

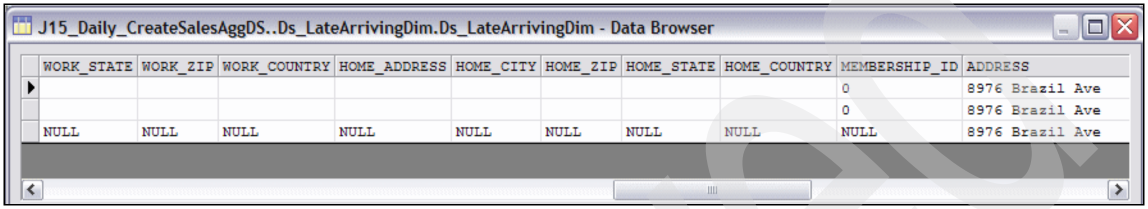
Figure 3-652 Execute the J15_Daily_CreateSalesAggDS job (Day 3) 9/13



J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

SKU	PRICE_USD	SELLING_PRICE_USD	COUNTRY_ISO_CODE	NAME	HOME_PHONE	WORK_PHONE	WORK_ADDRESS	WORK_CITY
NULL	00000120.00	00000120.00	USA	CASH CUSTOMER	555-555-5555	555-555-5555		
DS4321/07	00000035.00	00000025.00	USA	CASH CUSTOMER	555-555-5555	555-555-5555		
JP0819/08	00000075.00	00000075.00	JPN	NULL	NULL	NULL	NULL	NULL

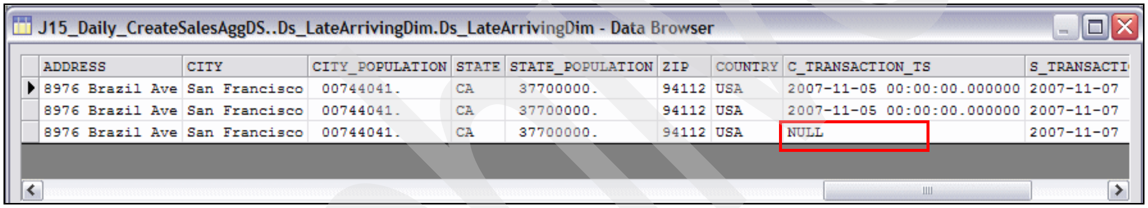
Figure 3-653 Execute the J15_Daily_CreateSalesAggDS job (Day 3) 10/13



J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

WORK_STATE	WORK_ZIP	WORK_COUNTRY	HOME_ADDRESS	HOME_CITY	HOME_ZIP	HOME_STATE	HOME_COUNTRY	MEMBERSHIP_ID	ADDRESS
								0	8976 Brazil Ave
								0	8976 Brazil Ave
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	8976 Brazil Ave

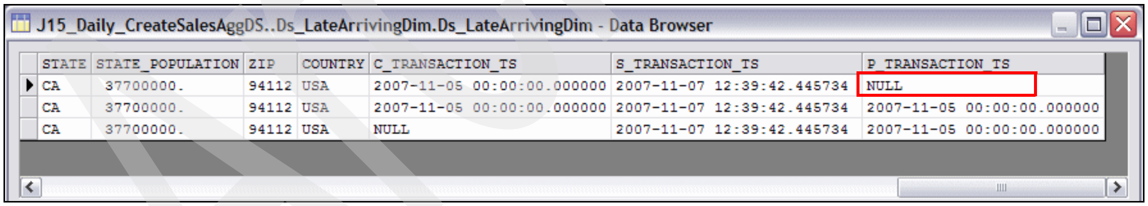
Figure 3-654 Execute the J15_Daily_CreateSalesAggDS job (Day 3) 11/13



J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	C_TRANSACTION_TS	S_TRANSACTI
8976 Brazil Ave	San Francisco	00744041.	CA	37700000.	94112	USA	2007-11-05 00:00:00.000000	2007-11-07
8976 Brazil Ave	San Francisco	00744041.	CA	37700000.	94112	USA	2007-11-05 00:00:00.000000	2007-11-07
8976 Brazil Ave	San Francisco	00744041.	CA	37700000.	94112	USA	NULL	2007-11-07

Figure 3-655 Execute the J15_Daily_CreateSalesAggDS job (Day 3) 12/13



J15_Daily_CreateSalesAggDS..Ds_LateArrivingDim.Ds_LateArrivingDim - Data Browser

STATE	STATE_POPULATION	ZIP	COUNTRY	C_TRANSACTION_TS	S_TRANSACTION_TS	P_TRANSACTION_TS
CA	37700000.	94112	USA	2007-11-05 00:00:00.000000	2007-11-07 12:39:42.445734	NULL
CA	37700000.	94112	USA	2007-11-05 00:00:00.000000	2007-11-07 12:39:42.445734	2007-11-05 00:00:00.000000
CA	37700000.	94112	USA	NULL	2007-11-07 12:39:42.445734	2007-11-05 00:00:00.000000

Figure 3-656 Execute the J15_Daily_CreateSalesAggDS job (Day 3) 13/13

J16_Daily_CreateScdInputDS (Day 3) execution

Figure 3-657 on page 552 through Figure 3-663 on page 554 show the results of the execution of this job with Day 3 data described earlier.

- Figure 3-657 on page 552 shows the results of the execution. The inputs to this job are as follows:
  - Accepts 5 rows as input from the "J15_Daily_CreateSalesAggDS (Day 3) execution" on page 548 job as seen in Figure 3-645 on page 549 through Figure 3-650 on page 550.
  - Accepts 1 row (corresponding to STORE_ID 9) as input from the Store dimension lookup data set generated in "J13_Daily_UpdateLookupDim (Day 3) execution" on page 544.
- The output of this job shows 6 rows corresponding to the union of the two inputs via the Funnel stage. Figure 3-658 on page 553 through Figure 3-663 on page 554 show the 6 rows in the output.

The next step is to execute the job described in "J17_DailyCreateSalesFactDS (Day 3) execution" on page 554.

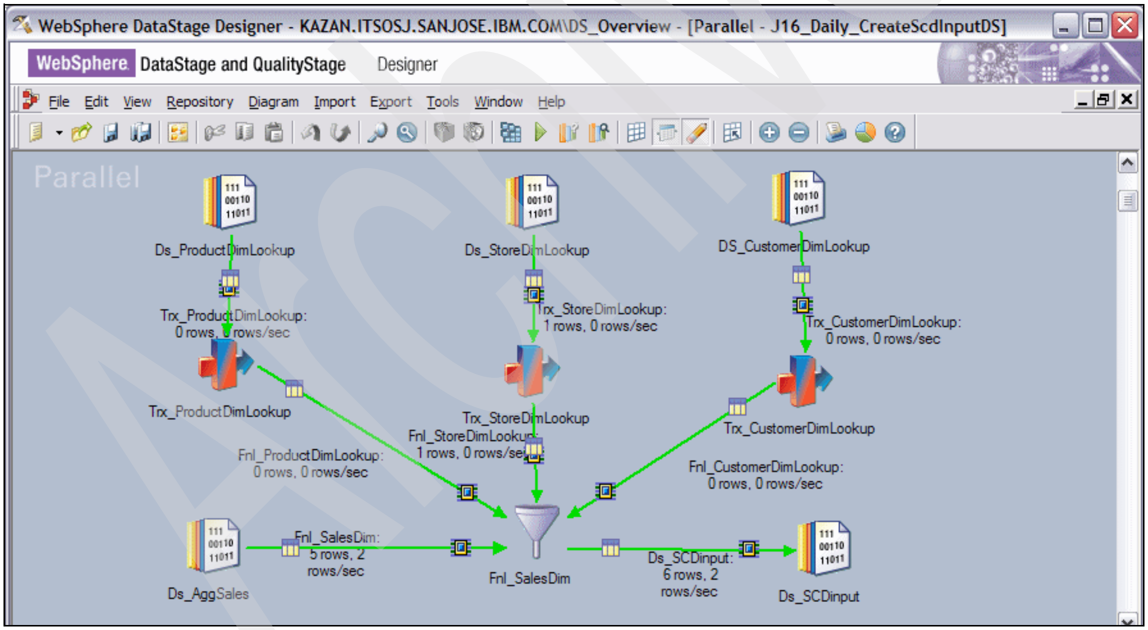
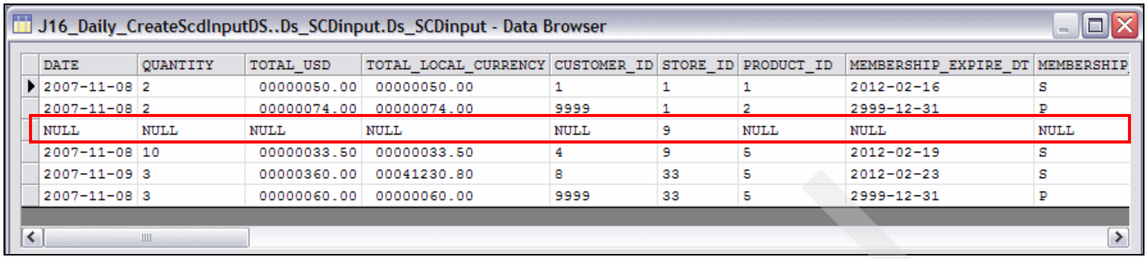


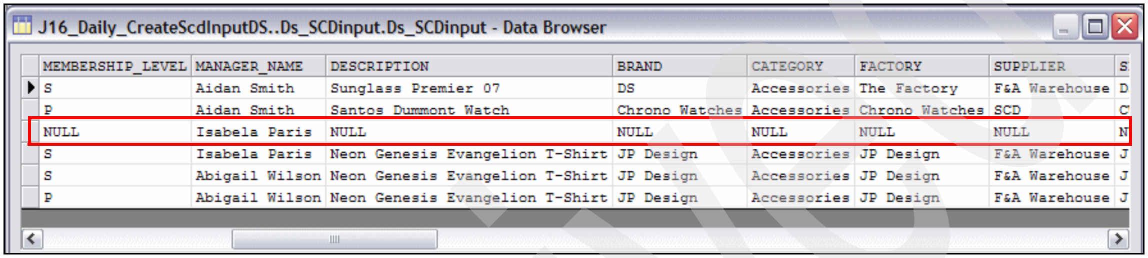
Figure 3-657 Execute the J16_Daily_CreateScdInputDS job (Day 3) 1/7



J16_Daily_CreateScdInputDS..Ds_SCDinput.Ds_SCDinput - Data Browser

DATE	QUANTITY	TOTAL_USD	TOTAL_LOCAL_CURRENCY	CUSTOMER_ID	STORE_ID	PRODUCT_ID	MEMBERSHIP_EXPIRE_DT	MEMBERSHIP
2007-11-08	2	00000050.00	00000050.00	1	1	1	2012-02-16	S
2007-11-08	2	00000074.00	00000074.00	9999	1	2	2999-12-31	P
NULL	NULL	NULL	NULL	NULL	9	NULL	NULL	NULL
2007-11-08	10	00000033.50	00000033.50	4	9	5	2012-02-19	S
2007-11-09	3	00000360.00	00041230.80	8	33	5	2012-02-23	S
2007-11-08	3	00000060.00	00000060.00	9999	33	5	2999-12-31	P

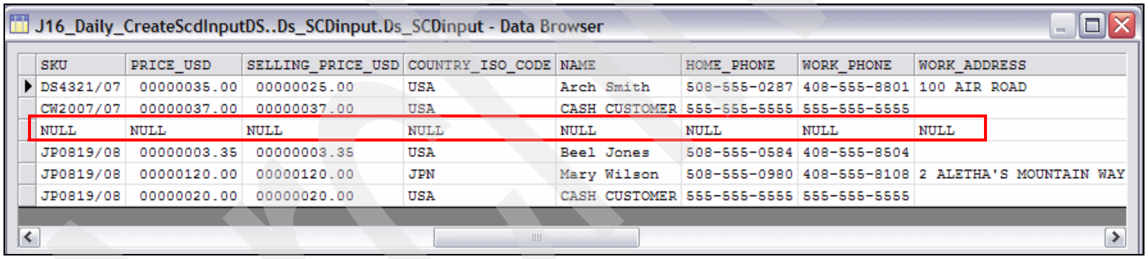
Figure 3-658 Execute the J16_Daily_CreateScdInputDS job (Day 3) 2/7



J16_Daily_CreateScdInputDS..Ds_SCDinput.Ds_SCDinput - Data Browser

MEMBERSHIP_LEVEL	MANAGER_NAME	DESCRIPTION	BRAND	CATEGORY	FACTORY	SUPPLIER	S
S	Aidan Smith	Sunglass Premier 07	DS	Accessories	The Factory	F&A Warehouse	D
P	Aidan Smith	Santos Dummont Watch	Chrono Watches	Accessories	Chrono Watches	SCD	C
NULL	Isabela Paris	NULL	NULL	NULL	NULL	NULL	N
S	Isabela Paris	Neon Genesis Evangelion T-Shirt	JP Design	Accessories	JP Design	F&A Warehouse	J
S	Abigail Wilson	Neon Genesis Evangelion T-Shirt	JP Design	Accessories	JP Design	F&A Warehouse	J
P	Abigail Wilson	Neon Genesis Evangelion T-Shirt	JP Design	Accessories	JP Design	F&A Warehouse	J

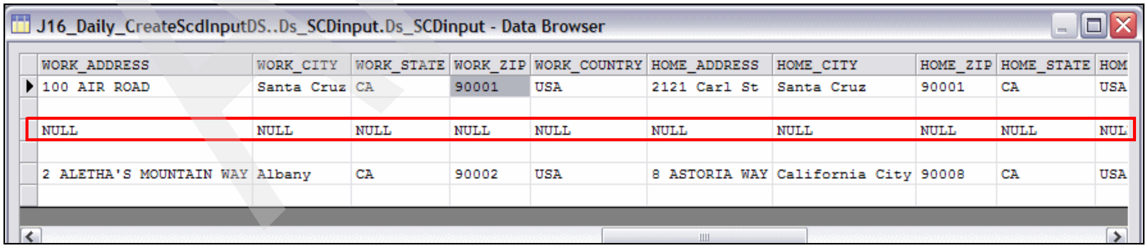
Figure 3-659 Execute the J16_Daily_CreateScdInputDS job (Day 3) 3/7



J16_Daily_CreateScdInputDS..Ds_SCDinput.Ds_SCDinput - Data Browser

SKU	PRICE_USD	SELLING_PRICE_USD	COUNTRY_ISO_CODE	NAME	HOME_PHONE	WORK_PHONE	WORK_ADDRESS
DS4321/07	00000035.00	00000025.00	USA	Arch Smith	508-555-0287	408-555-8801	100 AIR ROAD
CW2007/07	00000037.00	00000037.00	USA	CASH CUSTOMER	555-555-5555	555-555-5555	
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
JP0819/08	00000003.35	00000003.35	USA	Beel Jones	508-555-0584	408-555-8504	
JP0819/08	00000120.00	00000120.00	JPN	Mary Wilson	508-555-0980	408-555-8108	2 ALETHA'S MOUNTAIN WAY
JP0819/08	00000020.00	00000020.00	USA	CASH CUSTOMER	555-555-5555	555-555-5555	

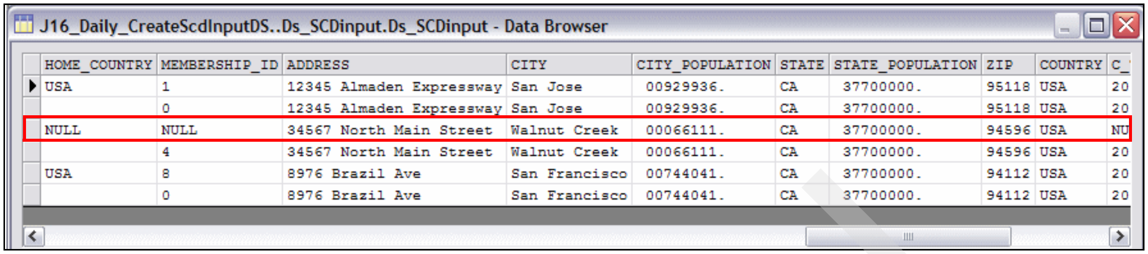
Figure 3-660 Execute the J16_Daily_CreateScdInputDS job (Day 3) 4/7



J16_Daily_CreateScdInputDS..Ds_SCDinput.Ds_SCDinput - Data Browser

WORK_ADDRESS	WORK_CITY	WORK_STATE	WORK_ZIP	WORK_COUNTRY	HOME_ADDRESS	HOME_CITY	HOME_ZIP	HOME_STATE	HOM
100 AIR ROAD	Santa Cruz	CA	90001	USA	2121 Carl St	Santa Cruz	90001	CA	USA
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NUL
2 ALETHA'S MOUNTAIN WAY	Albany	CA	90002	USA	8 ASTORIA WAY	California City	90008	CA	USA

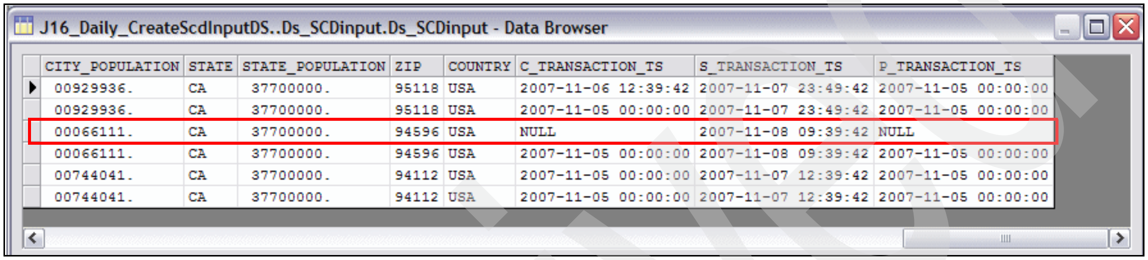
Figure 3-661 Execute the J16_Daily_CreateScdInputDS job (Day 3) 5/7



J16_Daily_CreateScdInputDS..Ds_SCDinput.Ds_SCDinput - Data Browser

HOME_COUNTRY	MEMBERSHIP_ID	ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	C_
USA	1	12345 Almaden Expressway	San Jose	00929936.	CA	37700000.	95118	USA	20
	0	12345 Almaden Expressway	San Jose	00929936.	CA	37700000.	95118	USA	20
NULL	NULL	34567 North Main Street	Walnut Creek	00066111.	CA	37700000.	94596	USA	NU
	4	34567 North Main Street	Walnut Creek	00066111.	CA	37700000.	94596	USA	20
USA	8	8976 Brazil Ave	San Francisco	00744041.	CA	37700000.	94112	USA	20
	0	8976 Brazil Ave	San Francisco	00744041.	CA	37700000.	94112	USA	20

Figure 3-662 Execute the J16_Daily_CreateScdInputDS job (Day 3) 6/7



J16_Daily_CreateScdInputDS..Ds_SCDinput.Ds_SCDinput - Data Browser

CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	C_TRANSACTION_TS	S_TRANSACTION_TS	P_TRANSACTION_TS
00929936.	CA	37700000.	95118	USA	2007-11-06 12:39:42	2007-11-07 23:49:42	2007-11-05 00:00:00
00929936.	CA	37700000.	95118	USA	2007-11-05 00:00:00	2007-11-07 23:49:42	2007-11-05 00:00:00
00066111.	CA	37700000.	94596	USA	NULL	2007-11-08 09:39:42	NULL
00066111.	CA	37700000.	94596	USA	2007-11-05 00:00:00	2007-11-08 09:39:42	2007-11-05 00:00:00
00744041.	CA	37700000.	94112	USA	2007-11-05 00:00:00	2007-11-07 12:39:42	2007-11-05 00:00:00
00744041.	CA	37700000.	94112	USA	2007-11-05 00:00:00	2007-11-07 12:39:42	2007-11-05 00:00:00

Figure 3-663 Execute the J16_Daily_CreateScdInputDS job (Day 3) 7/7

J17_DailyCreateSalesFactDS (Day 3) execution

Figure 3-664 on page 555 through Figure 3-670 on page 556 show the results of the execution of this job with Day 3 data described earlier.

- Figure 3-664 on page 555 shows the results of the execution. It accepts 6 rows as input from the “J16_Daily_CreateScdInputDS (Day 3) execution” on page 552 job as seen in Figure 3-658 on page 553 through Figure 3-663.
- The outputs of this job are as follows:
  - Two rows to the Ds_StoreDimUpdate data set (shown in Figure 3-665 on page 555 and Figure 3-666 on page 556).
    - There are two rows for the update of STORE_ID 9 because it has a Type 2 (MANAGER_NAME) change. The Type 2 change requires the expiry of the existing row in the Store dimension table (CURRENT_IND to ‘N’ and EXPIRATION_TS to Current Timestamp), and the addition of a new current row (CURRENT_IND of ‘Y’, EFFECTIVE_TS and EXPIRATION-TS).
  - No rows to the Ds_CustDimUpdate, Ds_ProdDimUpdate, and Ds_DateDimUpdate data set since there were no changes to the Customer, Product, and Date dimension tables.

– Five rows (as expected from the input) are written to the Ds_SalesFactUpdate data set with the appropriate surrogate key assigned to each sales transaction as shown in Figure 3-667 on page 556 through Figure 3-668 on page 556.

– The one row corresponding to late arriving data in the input is rejected and written to the Ds_LateArrivingData data set as shown in Figure 3-669 on page 556 and Figure 3-670 on page 556.

The next step is to execute the job described in “J18_Daily_UpdateStoreDim (Day 3) execution” on page 557.

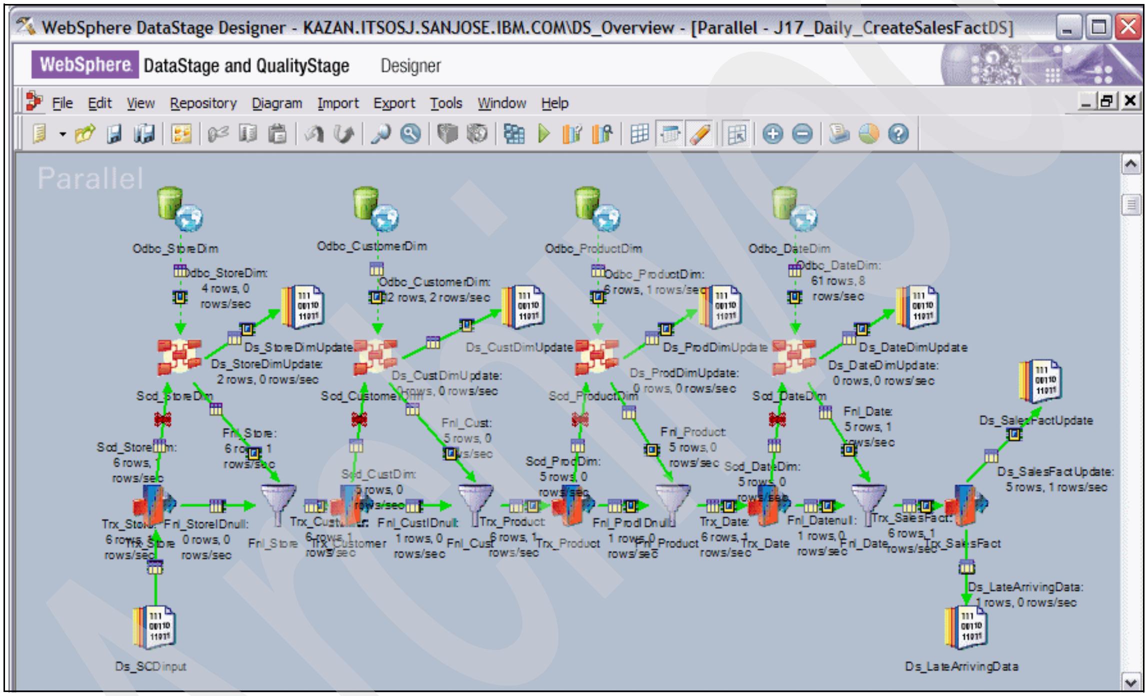
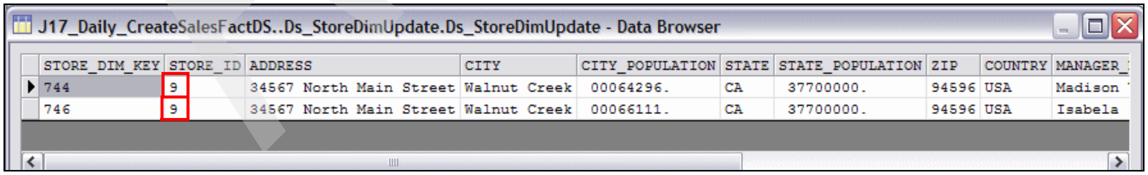
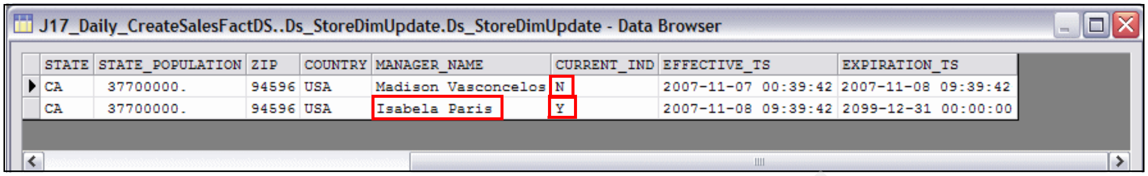


Figure 3-664 Execute the J17_DailyCreateSalesFactDS (Day 3) job (Day 3) 1/7


J17_Daily_CreateSalesFactDS..Ds_StoreDimUpdate.Ds_StoreDimUpdate - Data Browser

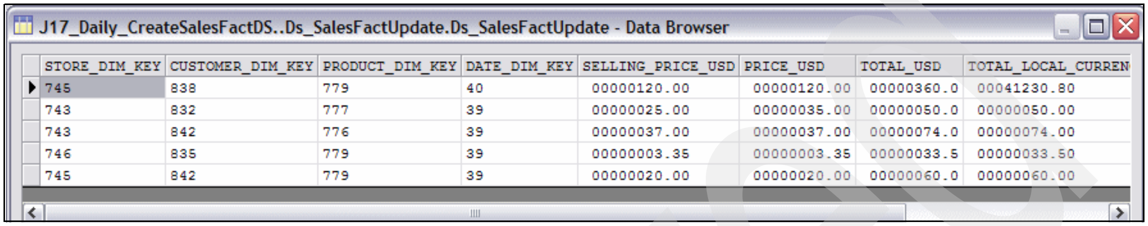
STORE_DIM_KEY	STORE_ID	ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY	MANAGER_
744	9	34567 North Main Street	Walnut Creek	00064296.	CA	37700000.	94596	USA	Madison
746	9	34567 North Main Street	Walnut Creek	00066111.	CA	37700000.	94596	USA	Isabela

Figure 3-665 Execute the J17_DailyCreateSalesFactDS (Day 3) job (Day 3) 2/7



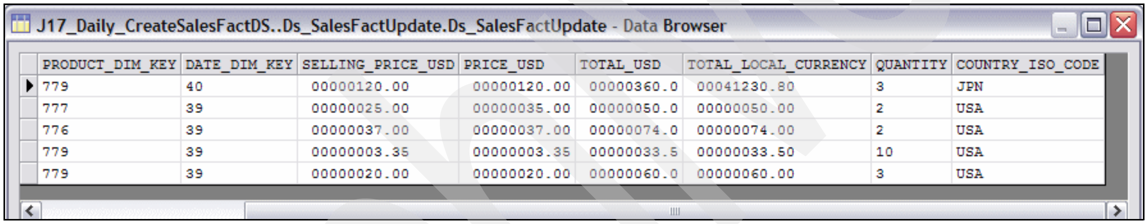
STATE	STATE_POPULATION	ZIP	COUNTRY	MANAGER_NAME	CURRENT_IND	EFFECTIVE_TS	EXPIRATION_TS
CA	37700000.	94596	USA	Madison Vasconcelos	N	2007-11-07 00:39:42	2007-11-08 09:39:42
CA	37700000.	94596	USA	Isabela Paris	Y	2007-11-08 09:39:42	2099-12-31 00:00:00

Figure 3-666 Execute the J17_DailyCreateSalesFactDS (Day 3) job (Day 3) 3/7



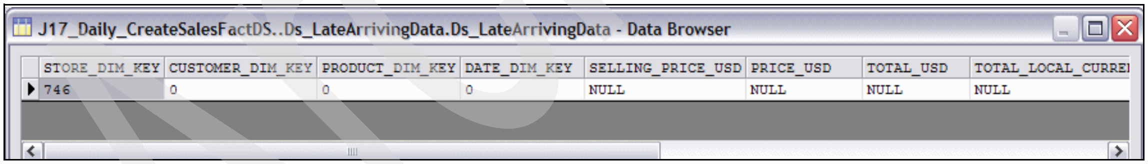
STORE_DIM_KEY	CUSTOMER_DIM_KEY	PRODUCT_DIM_KEY	DATE_DIM_KEY	SELLING_PRICE_USD	PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURREN
745	838	779	40	00000120.00	00000120.00	00000360.0	00041230.80
743	832	777	39	00000025.00	00000035.00	00000050.0	00000050.00
743	842	776	39	00000037.00	00000037.00	00000074.0	00000074.00
746	835	779	39	00000003.35	00000003.35	00000033.5	00000033.50
745	842	779	39	00000020.00	00000020.00	00000060.0	00000060.00

Figure 3-667 Execute the J17_DailyCreateSalesFactDS (Day 3) job (Day 3) 4/7



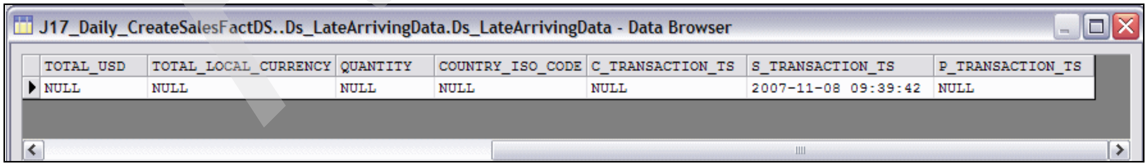
PRODUCT_DIM_KEY	DATE_DIM_KEY	SELLING_PRICE_USD	PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURRENCY	QUANTITY	COUNTRY_ISO_CODE
779	40	00000120.00	00000120.00	00000360.0	00041230.80	3	JPN
777	39	00000025.00	00000035.00	00000050.0	00000050.00	2	USA
776	39	00000037.00	00000037.00	00000074.0	00000074.00	2	USA
779	39	00000003.35	00000003.35	00000033.5	00000033.50	10	USA
779	39	00000020.00	00000020.00	00000060.0	00000060.00	3	USA

Figure 3-668 Execute the J17_DailyCreateSalesFactDS (Day 3) job (Day 3) 5/7



STORE_DIM_KEY	CUSTOMER_DIM_KEY	PRODUCT_DIM_KEY	DATE_DIM_KEY	SELLING_PRICE_USD	PRICE_USD	TOTAL_USD	TOTAL_LOCAL_CURRE
746	0	0	0	NULL	NULL	NULL	NULL

Figure 3-669 Execute the J17_DailyCreateSalesFactDS (Day 3) job (Day 3) 6/7



TOTAL_USD	TOTAL_LOCAL_CURRENCY	QUANTITY	COUNTRY_ISO_CODE	C_TRANSACTION_TS	S_TRANSACTION_TS	P_TRANSACTION_TS
NULL	NULL	NULL	NULL	NULL	2007-11-08 09:39:42	NULL

Figure 3-670 Execute the J17_DailyCreateSalesFactDS (Day 3) job (Day 3) 7/7

J18_Daily_UpdateStoreDim (Day 3) execution

Figure 3-671 here through Figure 3-673 on page 558 show the results of the execution of this job with Day 3 data described earlier.

- Figure 3-671 shows the results of the execution. It accepts 2 rows as input from the “J17_DailyCreateSalesFactDS (Day 3) execution” on page 554 job as seen in Figure 3-665 on page 555 and Figure 3-666 on page 556.
- The outputs are as follows:
  - There are no rows written to the Rej_StoreDim link.
  - The 2 rows written to the Odbc_StoreDim link updates the STORE_DIM dimension table with these changes (as highlighted) as seen in Figure 3-672 on page 558 and Figure 3-673 on page 558.

The next step is to execute the job described in “J19_Daily_UpdateCustomerDim (Day 3) execution” on page 558.

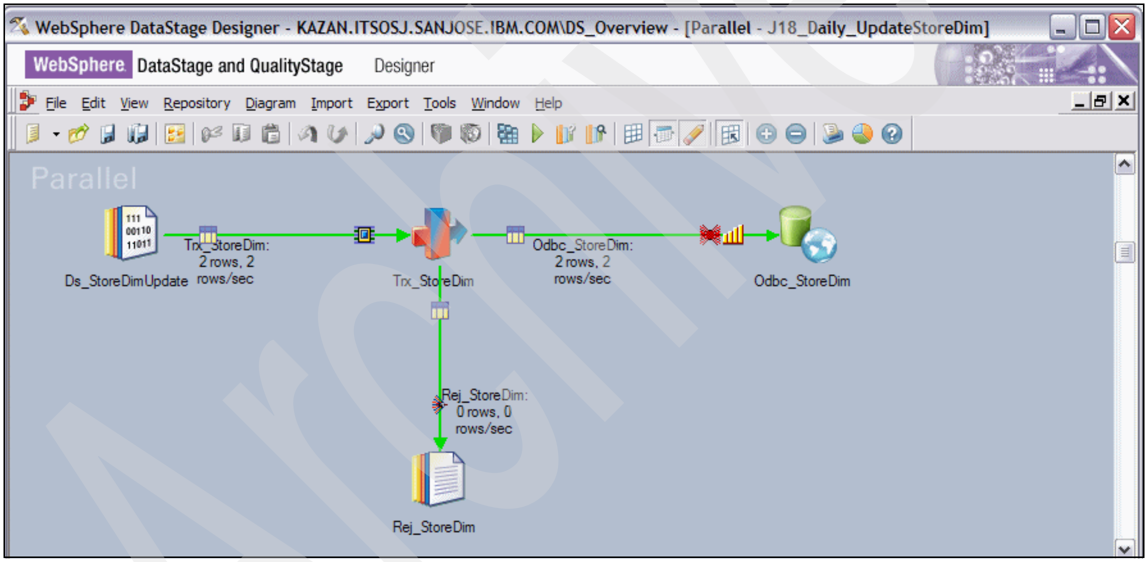
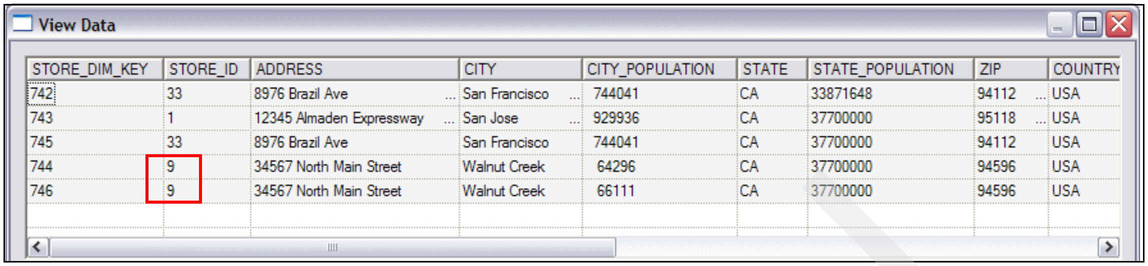
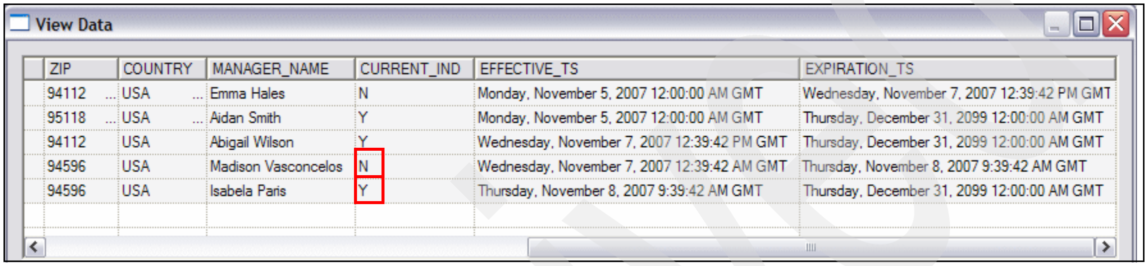


Figure 3-671 Execute the J18_Daily_UpdateStoreDim job (Day 3) 1/3



STORE_DIM_KEY	STORE_ID	ADDRESS	CITY	CITY_POPULATION	STATE	STATE_POPULATION	ZIP	COUNTRY
742	33	8976 Brazil Ave ...	San Francisco ...	744041	CA	33871648	94112 ...	USA
743	1	12345 Almaden Expressway ...	San Jose ...	929936	CA	37700000	95118 ...	USA
745	33	8976 Brazil Ave	San Francisco	744041	CA	37700000	94112	USA
744	9	34567 North Main Street	Walnut Creek	64296	CA	37700000	94596	USA
746	9	34567 North Main Street	Walnut Creek	66111	CA	37700000	94596	USA

Figure 3-672 Execute the J18_Daily_UpdateStoreDim job (Day 3) 2/3



ZIP	COUNTRY	MANAGER_NAME	CURRENT_IND	EFFECTIVE_TS	EXPIRATION_TS
94112 ...	USA ...	Emma Hales	N	Monday, November 5, 2007 12:00:00 AM GMT	Wednesday, November 7, 2007 12:39:42 PM GMT
95118 ...	USA ...	Aidan Smith	Y	Monday, November 5, 2007 12:00:00 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
94112	USA	Abigail Wilson	Y	Wednesday, November 7, 2007 12:39:42 PM GMT	Thursday, December 31, 2099 12:00:00 AM GMT
94596	USA	Madison Vasconcelos	N	Wednesday, November 7, 2007 12:39:42 AM GMT	Thursday, November 8, 2007 9:39:42 AM GMT
94596	USA	Isabela Paris	Y	Thursday, November 8, 2007 9:39:42 AM GMT	Thursday, December 31, 2099 12:00:00 AM GMT

Figure 3-673 Execute the J18_Daily_UpdateStoreDim job (Day 3) 3/3

J19_Daily_UpdateCustomerDim (Day 3) execution

Figure 3-674 on page 559 shows the results of the execution of this job with Day 3 data described earlier.

It shows no input records to update the Customer dimension table.

The next step is to execute the job described in “J20_Daily_UpdateProductDim (Day 3) execution” on page 559.

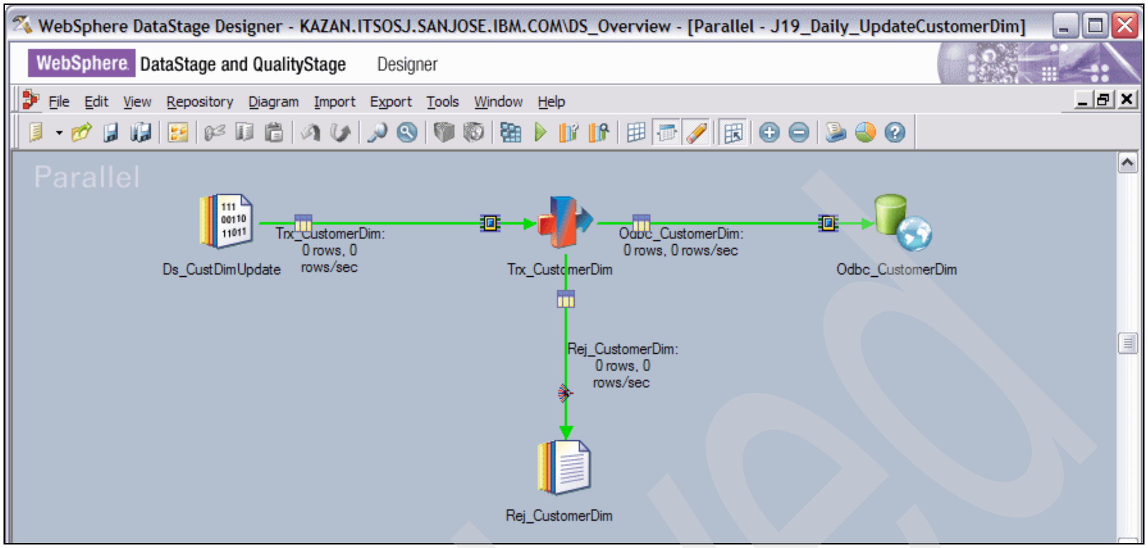


Figure 3-674 Execute the J19_Daily_UpdateCustomerDim job (Day 3)

J20_Daily_UpdateProductDim (Day 3) execution

Figure 3-675 shows the results of the execution of this job with Day 3 data described earlier.

It shows no input records to update the Product dimension table.

The next step is to execute the job described in “J21_Daily_UpdateDateDim (Day 3) execution” on page 560.

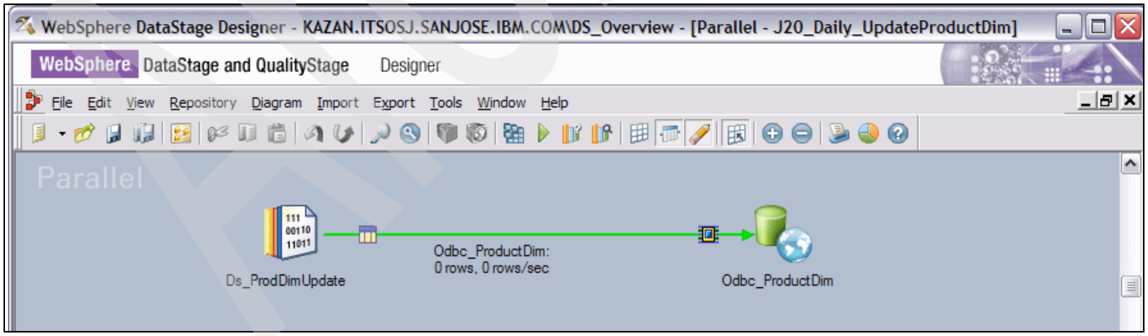


Figure 3-675 Execute the J20_Daily_UpdateProductDim job (Day 3) 1/?

J21_Daily_UpdateDateDim (Day 3) execution

Figure 3-676 shows the results of the execution of this job with Day 3 data described earlier.

It shows no input records to update the Date dimension table.

The next step is to execute the job described in “J22_Daily_UpdateSalesFact (Day 3) execution” on page 560.

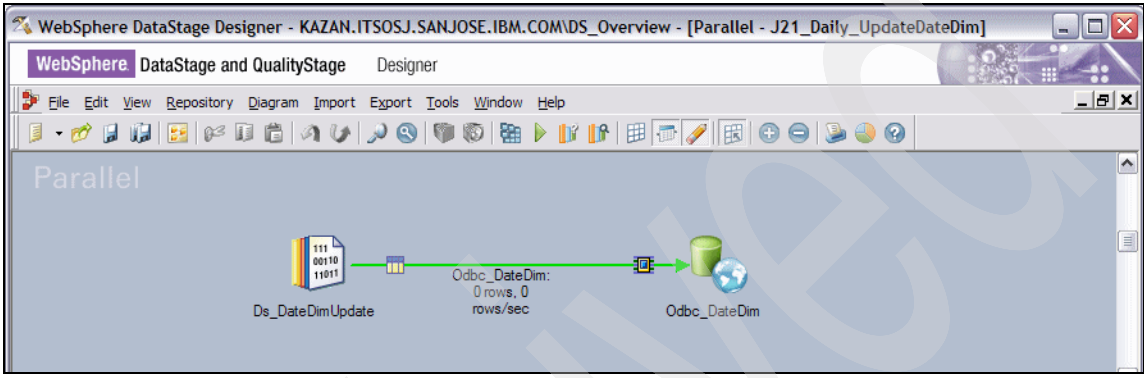


Figure 3-676 Execute the J21_Daily_UpdateDateDim job (Day 3) 1/?

J22_Daily_UpdateSalesFact (Day 3) execution

Figure 3-677 on page 561 through Figure 3-679 on page 562 show the results of the execution of this job with Day 3 data described earlier.

- Figure 3-677 on page 561 shows the results of the execution. It accepts 5 rows as input from the “J17_DailyCreateSalesFactDS (Day 3) execution” on page 554 job as seen in Figure 3-667 on page 556 and Figure 3-668 on page 556.
- The output shows 5 rows being written to the Odbc_SalesFact link which is used to update the SALES_FACT table. Figure 3-678 on page 561 and Figure 3-679 on page 562 show the updated contents of the SALES_FACT table as highlighted.

This concludes Day 3 processing and our retail industry scenario.

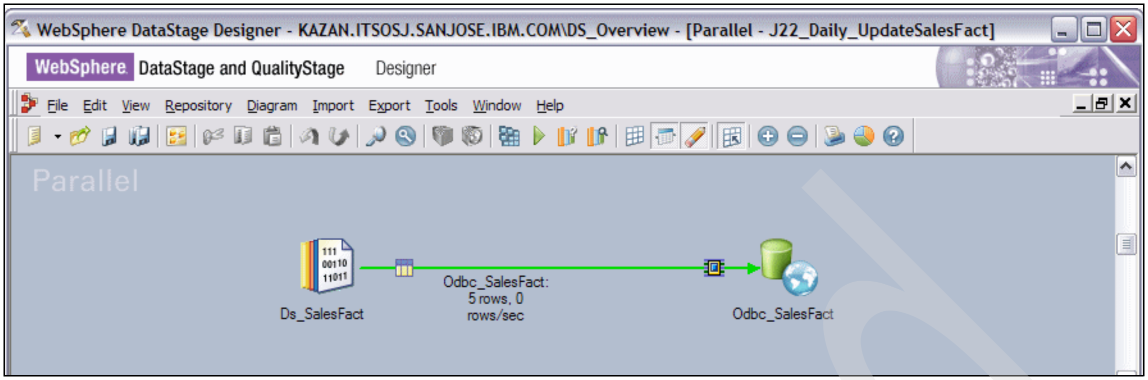
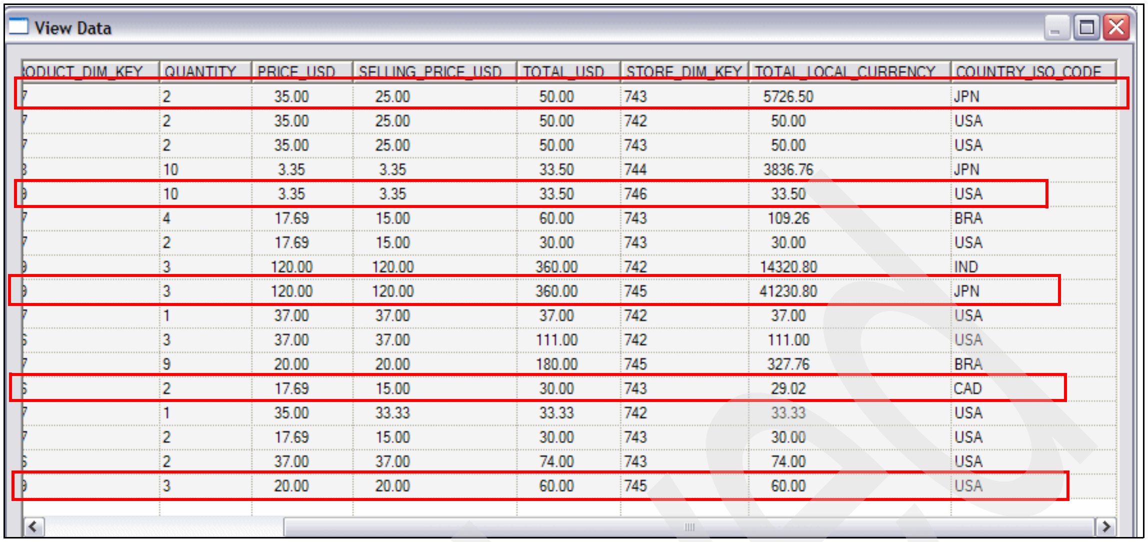


Figure 3-677 Execute the J22_Daily_UpdateSalesFact job (Day 3) 1/3

View Data

CUSTOMER_DIM_KEY	DATE_DIM_KEY	PRODUCT_DIM_KEY	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	STORE_DIM_KEY	TOTAL
832	36	777	2	35.00	25.00	50.00	743	5726.
832	37	777	2	35.00	25.00	50.00	742	50.0
832	39	777	2	35.00	25.00	50.00	743	50.0
835	38	778	10	3.35	3.35	33.50	744	3836.
835	39	779	10	3.35	3.35	33.50	746	33.5
836	36	777	4	17.69	15.00	60.00	743	109.
836	37	777	2	17.69	15.00	30.00	743	30.0
838	36	779	3	120.00	120.00	360.00	742	14320
838	40	779	3	120.00	120.00	360.00	745	41230
839	37	777	1	37.00	37.00	37.00	742	37.0
840	37	776	3	37.00	37.00	111.00	742	111.
841	38	777	9	20.00	20.00	180.00	745	327.
842	37	776	2	17.69	15.00	30.00	743	29.0
842	37	777	1	35.00	33.33	33.33	742	33.3
842	37	777	2	17.69	15.00	30.00	743	30.0
842	39	776	2	37.00	37.00	74.00	743	74.0
842	39	779	3	20.00	20.00	60.00	745	60.0

Figure 3-678 Execute the J22_Daily_UpdateSalesFact job (Day 3) 2/3


View Data

ODUCT_DIM_KEY	QUANTITY	PRICE_USD	SELLING_PRICE_USD	TOTAL_USD	STORE_DIM_KEY	TOTAL_LOCAL_CURRENCY	COUNTRY_ISO_CODE
7	2	35.00	25.00	50.00	743	5726.50	JPN
7	2	35.00	25.00	50.00	742	50.00	USA
7	2	35.00	25.00	50.00	743	50.00	USA
8	10	3.35	3.35	33.50	744	3836.76	JPN
9	10	3.35	3.35	33.50	746	33.50	USA
7	4	17.69	15.00	60.00	743	109.26	BRA
7	2	17.69	15.00	30.00	743	30.00	USA
9	3	120.00	120.00	360.00	742	14320.80	IND
9	3	120.00	120.00	360.00	745	41230.80	JPN
7	1	37.00	37.00	37.00	742	37.00	USA
6	3	37.00	37.00	111.00	742	111.00	USA
7	9	20.00	20.00	180.00	745	327.76	BRA
6	2	17.69	15.00	30.00	743	29.02	CAD
7	1	35.00	33.33	33.33	742	33.33	USA
7	2	17.69	15.00	30.00	743	30.00	USA
6	2	37.00	37.00	74.00	743	74.00	USA
9	3	20.00	20.00	60.00	745	60.00	USA

Figure 3-679 Execute the J22_Daily_UpdateSalesFact job (Day 3) 3/3

A

IBM Information Server setups

In this appendix we describe the setup of various products used in the retail industry scenario, such as IBM Information Integrator Classic Federation server for z/OS, creating a Queue Manager, setting up the XA parameters on Queue Manager, and creating the queues.

The topics covered include:

- Configuring IBM InfoSphere Classic Federation Server for z/OS
- Creating the Queue Manager
- Setting up the XA parameters on Queue Manager
- Creating the queues

A.1 Introduction

WantThatStuff's operational systems are provided on a z/OS platform. While most of the data sources are on DB2 for z/OS, two of the data sources (Product and Store) are VSAM files, while three data sources (Customer, Employee, and SalesTrans) are sequential files.

The sequential files (Customer, Employee, and SalesTrans) are processed on the IBM Information Server (kazan.itsosj.sanjose.ibm.com) using the IBM InfoSphere DataStage FTP Enterprise and CFF stages similar to that described in "J01_IL_FTPCustomerFile" on page 159 and "J02_IL_LoadCustomerDim" on page 184.

The VSAM files (Product and Store) are accessed as relational tables on the IBM Information Server (kazan.itsosj.sanjose.ibm.com) platform using IBM InfoSphere Classic Federation Server for z/OS similar to that described in "J03_IL_LoadProductDim" on page 202.

Classic Data Architect is used to create relational tables and views that map to data sources in supported non-relational database management systems. With IBM InfoSphere Classic Federation Server for z/OS, client applications can issue SQL queries against these tables to access data in the non-relational databases. Client applications can also issue INSERT, DELETE, and UPDATE requests against the tables to modify the data in the non-relational databases. Before you begin, you must perform the following tasks on the data server where the query processor will run:

1. Create and initialize a metadata catalog as described in A.2.2, "Configuration of IBM InfoSphere Classic Federation for z/OS system catalog" on page 567.
2. Set up the configuration file (contents are shown in Example A-1 on page 568 — the highlighted portion shows the changes made for our scenario).
3. Start the data server (not shown here).

The configuration of IBM InfoSphere Classic Federation Server for z/OS for the Product and Store VSAM files is described in "Configure IBM InfoSphere Classic Federation Server for z/OS" on page 565.

The "J13_Daily_UpdateLookupDim (Day 1)" on page 356 retrieves changes to customer, product, and store attributes (Type 1 and Type 2) from an IBM WebSphere MQ queue, updates the dimension lookup tables, and creates a data set for each dimension table (with nulls in the sales transaction[1] portion of the records) for input to the SCD stage in the job, "J17_DailyCreateSalesFactDS (Day1)" on page 424.

The configuration of the IBM WebSphere MQ queue manager, setting up the XA parameters for the queue manager, and creating the queues, are described in “Create the Queue Manager” on page 580, “Set up the XA parameters on Queue Manager” on page 587, and “Create the queues” on page 591.

A.2 Configure IBM InfoSphere Classic Federation Server for z/OS

IBM InfoSphere Classic Federation Server for z/OS is a complete, high-powered solution that provides SQL access to mainframe databases and files without mainframe programming.

Using the key product features, you can:

- Read from and write to mainframe data sources using SQL.
- Map logical relational table structures to existing physical mainframe databases and files.
- Use the Classic Data Architect graphical user interface (GUI) to issue standard SQL commands to the logical tables.
- Use standards-based access with ODBC, JDBC, or CLI interfaces.
- Take advantage of multi-threading with native drivers for scalable performance.

The architecture of InfoSphere Classic Federation Server for z/OS consists of the following major components:

- **Data server**

  Data servers perform all data access. The architecture of the data server is service-based. The data server consists of several components, or services. A major service embedded in the data server is the query processor that acts as the relational engine for Classic federation.

- **Data connectors**

  The query processor dynamically loads one or more data connectors to access the target database or file system that is referenced in an SQL request.

[1] This record is created to ensure that the dimension tables are updated in the SCD stage in “J17_DailyCreateSalesFactDS (Day1)” on page 433 even if there are no sales transactions associated with those dimension table changes. This is the late arriving (or no existing) sales transactions scenario where the dimension tables must be updated with the Type 1 and Type 2 attribute changes even when there are no incoming sales transactions in that daily cycle.

- **Classic Data Architect**

  To process SQL data access requests, data definitions must be mapped to logical tables. Classic Data Architect[2] is the administrative tool that you should use to perform this mapping.

  Classic Data Architect is the enhanced interface introduced in Version 9 that replaces the Classic Data Mapper. The purpose of the Classic Data Architect is to administer the logical table definitions, views, and SQL security information that are stored in the metadata catalog.

  The key benefits that the Classic Data Architect tool provides make it easier for you to perform the following tasks:

  - Define tables, columns, primary keys, indexes, stored procedures, and views.
  - Specify user authorization for all objects.
  - Import existing physical definitions from copybooks, CA-IDMS schemas, and IMS database descriptors (DBDs).
  - Generate DDL for the objects that you create that can be run directly on a server or saved to a script file.
  - Generate DDL script from objects already defined in the catalog and export DDL scripts to a data set on the server for use with the metadata utility.
  - Connect directly to a Classic data source and view the objects in the system catalog.

- **Metadata catalog**

  The information that you generate from the Classic Data Architect is stored in metadata catalogs. A metadata catalog is a set of relational tables that contain information about how to convert data from non-relational to relational formats. The data server accesses the information stored in these catalogs.

- **Clients (ODBC, JDBC, and CLI)**

  InfoSphere Classic Federation Server for z/OS provides the ODBC, JDBC, and CLI clients. The clients enable client applications or tools to submit SQL queries to the data server.

The following sections briefly describe the installation of Classic Data Architect and IBM InfoSphere Classic Federation Server for z/OS, configuration of the IBM InfoSphere Classic Federation Server for z/OS system catalog, and the configuration of Classic Data Architect.

[2] Classic Data Architect is a new Eclipse-based GUI tool that assists you in configuring access to mainframe data sources and InfoSphere Classic components.

A.2.1 Installation

The installation of Classic Data Architect and IBM InfoSphere Classic Federation Server for z/OS is briefly described here:

1. Install Classic Data Architect with the typical setup option on the Linux platform where IBM InfoSphere DataStage is installed — kazan.itsosj.sanjose.ibm.com in our case.

   For details on installing Classic Data Architect, refer to:
   `http://publib.boulder.ibm.com/infocenter/iisclzos/v9r1/index.jsp?topic=/com.ibm.websphere.ii.product.install.clas.doc/topics/iiypicac-instcda.html`

2. Install IBM InfoSphere Classic Federation Server for z/OS on the z/OS platform where WantThatStuff's VSAM data sources are located.

   For details on installing IBM InfoSphere Classic Federation Server for z/OS, refer to *Program Directory for IBM WebSphere Classic Federation Server for z/OS V09.01.00, Program Number 5655-R52*, GI10-8750-00.

> **Attention:** It is essential that you install Classic Data Architect *before* you install IBM InfoSphere Classic Federation Server for z/OS. Failure to do so will result in the ODBC drivers for z/OS not being installed if you happen to use them in your scenario. For examples of configuring ODBC data sources on the z/OS platform, refer to the Redbooks publication, *IBM WebSphere Information Analyzer & Data Quality Assessment*, SG24-7508.

A.2.2 Configuration of IBM InfoSphere Classic Federation for z/OS system catalog

In this section we allocate the system (metadata) catalog and update it with metadata about the Product and Store VSAM files (Example A-1 on page 568).

Example A-2 on page 570 shows the CACPOST job that allocates and populates the appropriate data sets such as the error message catalog and the metadata catalog. The catalog initialization and maintenance utility (CACCATUT) is a z/OS batch job that creates or performs operations on an offline metadata catalog — the INIT operation of the CACCATUT initializes data sets for a version 9.1 sequential metadata catalog and creates the SYSIBM and SYSCAC system tables that make up the metadata catalog. The ENGCAT DD statement references the message catalog[3].

[3] The message catalog is accessed by the CLI component and the metadata utility to retrieve the text for error messages reported by the data server and error conditions detected by CLI or by the metadata utility.

Example A-3 on page 571 shows the CACMETAU[4] job that connects to the data server (Example A-4 on page 573 shows the connect configuration details), then reads the DDL statements from SYSIN * and sends the statements to the server to update the system catalog. Example A-5 on page 573 shows the DDL statements for the PRODUCT data source, while Example A-6 on page 573 shows the DDL statements for the STORE data source.

Example: A-1 Configuration file contents on the data server

```
**********************************************************************
*                                                                    *
*     DATA SERVER CONFIGURATION                                      *
*                                                                    *
*     THIS FILE CONTAINS CONFIGURATION DATA REQUIRED                 *
*     FOR THE OPERATION OF THE DATA SERVER.                          *
*                                                                    *
*     THE FILE IS ORGANIZED AS A SERIES OF ENTRIES EACH              *
*     OF WHICH CONSISTS OF A KEYWORD AND VALUE PAIR                  *
*     SEPARATED BY A REQUIRED "=" SIGN.  ORDER OF THE                *
*     ENTRIES IS NOT IMPORTANT, EXCEPT WHERE NOTED.                  *
*     MAXIMUM LENGTH OF AN ENTRY IS 80 CHARACTERS PER                *
*     LINE, WITH A MAXIMUM PARAMETER LENGTH OF 255                   *
*     CHARACTERS, SPANNED BY THE BACKSLASH CONTINUATION              *
*     CHARACTER - \.                                                 *
*                                                                    *
*     NOTE: WHEN EDITING CONFIGURATION MEMBERS ENSURE THAT           *
*           "NUM OFF" IS SPECIFIED. IF THE CONFIGURATION             *
*           CONTAINS SEQUENCE NUMBERS, UNKNOWN CONFIGURATION         *
*           PARAMETERS, OR INVALID SUB-PARAMETER VALUES THE          *
*           DATA SERVER WILL NOT RUN.                                *
*                                                                    *
**********************************************************************
*
* THE FOLLOWING SERVICE INFO ENTRIES ARE REQUIRED.
SERVICE INFO ENTRY = CACCNTL CNTL 0 1 1 100 4 5M 5M NO_DATA
SERVICE INFO ENTRY = CACLOG LOG 1 1 1 100 1 5M 5M DISPLAY
SERVICE INFO ENTRY = CACOPER OPER 2 1 1 100 4 5M 5M NO_DATA
*
**************************************************************
*
* LANGUAGE ENVIRONMENT
*     UNCOMMENT THE FOLLOWING SERVICE INFO ENTRIES IF YOU WILL BE
*     USING RECORD EXITS OR STORED PROCEDURES USING IBM'S
*     LANGUAGE ENVIRONMENT OR COBOL II. THE FIRST ENTRY IS FOR LE
*     AND THE SECOND FOR COBOL II.
```

[4] The catalog initialization and maintenance utility (CACCATUT) is a z/OS batch job that creates or performs operations on an offline metadata catalog. Offline means that no services can reference the metadata catalog while CACCATUT is running.

```
*     FOR COBOL II, IF YOU WILL HAVE MORE THAN ONE CONCURRENT USER,
*     DO NOT ACTIVATE THIS INTERFACE.
*SERVICE INFO ENTRY = CACLE LANGENV 2 1 1 50 4 5M 5M CEEPIPI
*SERVICE INFO ENTRY = CACLE LANGENV 2 1 1 50 4 5M 5M IGZERRE
*
***************************************************************
*
* WLM USER EXIT INTERFACE INITIALIZATION
*SERVICE INFO ENTRY = CACWLM WLM01 2 1 1 1 4 5M 5M \
*       CACSX06,SUBSYS=JES,SUBSYSNM=CAC01
*
***************************************************************
*
* QUERY PROCESSOR SERVICE INFO ENTRY
*   THE LAST SUBPARAMETER POINTS TO A QP SERVICE CONFIGURATION FILE
SERVICE INFO ENTRY = CACQP CACSAMP 2 5 10 20 4 5M 5M CACQPCF
*
***************************************************************
*
* CA-DATACOM/DB INTERFACE
*SERVICE INFO ENTRY = CACDCI DCOM 2 1 1 50 4 5M 5M 4
*
* DB2 INTERFACE
* CHANGE THE DSN FIELD TO THE SUBSYSTEM IDENTIFIER FOR
*    YOUR SITE'S DB2 SUBSYSTEM.
*SERVICE INFO ENTRY = CACCAF DB8A 2 1 5 1 4 5M 5M CAC91PLN
SERVICE INFO ENTRY = CACCAF DB8A 2 1 5 1 1 5M 5M CAC91PLN
*
* IMS DBB/BMP INTERFACE
*SERVICE INFO ENTRY = CACIMSIF IMS  2 1 1 50 4 5M 5M NO_DATA
*
* IMS DRA INTERFACE
*SERVICE INFO ENTRY = CACDRA IMS  2 1 1 50 4 5M 5M 00,DRAUSER,DEFPSB
*
* IMS ODBA INTERFACE
*SERVICE INFO ENTRY = CACRRSI IMS  2 1 1 50 4 5M 5M SSID,DEFPSB
*
* VSAM INTERFACE
SERVICE INFO ENTRY = CACVSMS VSAMSRV 2 1 1 50 4 5M 5M CLOSE_ON_IDLE
*
***************************************************************
*
* TCP/IP CONNECTION HANDLER
*  REFER TO DOCUMENTATION FOR DETAILED INFORMATION ON LAST SUBPARAMETER
SERVICE INFO ENTRY = CACINIT TCPIP 2 1 1 100 4 5M 5M \
TCP/0.0.0.0/5525
*TCP/WTSC59.ITSO.IBM.COM/5001
*
*  TCP/IP SYSTEM FILE HIGH LEVEL QUALIFIER, SUBSYSTEM NAME
```

```
*  AND TIMEZONE SETTING
*TASK PARAMETERS = =TCPIP_PREFIX=HLQUAL =TCPIP_MACH=TCPIP =TZ=PST9PDT
*
* XM CONNECTION HANDLER
*  REFER TO DOCUMENTATION FOR DETAILED INFORMATION ON LAST SUBPARAMETER
*SERVICE INFO ENTRY = CACINIT XMNT 2 1 1 50 4 5M 5M \
* XM1/CAC/CAC
*
* MQ-SERIES CONNECTION HANDLER
*  REFER TO DOCUMENTATION FOR DETAILED INFORMATION ON LAST SUBPARAMETER
*SERVICE INFO ENTRY = CACINIT MQI 2 1 1 50 4 5M 5M \
* MQI/SCQ1/CAC.SERVER
*
**************************************************************
*
* SAF (SECURITY) SYSTEM EXIT
*SAF EXIT = CACSX04 IMS CLASS=PIMS
*
* SMF (REPORTING) SYSTEM EXIT
*SMF EXIT = CACSX02 RECTYPE=255,SYSID=JES2
*
*************************************************************
*
* MISC REQUIRED PARAMETERS
*
MESSAGE POOL SIZE = 16777216
*
NL = US ENGLISH
NL CAT = DD:ENGCAT
*
* IF YOU ARE NOT ALLOWING UPDATES TO THE CATALOG FILES WHILE
* ANY DATA SERVERS ARE ACCESSING THE CATALOG FILES, CHANGE THE
* VALUE TO A ONE.  THE CATALOG FILES WILL ONLY BE OPENED DURING
* QP INITIALIZATION RATHER THAN DURING EACH QUERY OPEN CURSOR.
*
STATIC CATALOGS = 0
```

Example: A-2   Allocate data sets

```
//CACPOST JOB (999,POK),'POST SMPE TASKS',CLASS=A,
//     MSGCLASS=X,NOTIFY=&SYSUID
//*******************************************************************
//CACCLN   EXEC PGM=IEFBR14
//SCACMENU  DD  DISP=(MOD,DELETE,DELETE),VOL=SER=OP1TSD,
//          UNIT=SYSALLDA,RECFM=FBS,LRECL=80,BLKSIZE=27920,
//          SPACE=(CYL,(1,1)),
//          DSN=NALUR1.CAC.PRODUCT.SCACMENU
```

```
//*
//CACENG1 EXEC PGM=IEBGENER
//SYSPRINT  DD  SYSOUT=*
//SYSUT1    DD  DISP=SHR,
//          DSN=CAC.SCACSAMP(CACENGCT)
//SYSUT2    DD  DISP=(NEW,CATLG,DELETE),VOL=SER=OP1TSD,
//          UNIT=SYSALLDA,RECFM=FBS,LRECL=80,BLKSIZE=27920,
//          SPACE=(CYL,(1,1)),
//          DSN=NALUR1.CAC.PRODUCT.SCACMENU
//SYSIN     DD  DUMMY
//*
//CACCATAL EXEC PGM=IEFBR14
//CACCAT   DD  UNIT=SYSALLDA,VOL=SER=OP1TSD,
//             DSN=NALUR1.CAC.PRODUCT.CATALOG,
//             SPACE=(CYL,(10,10)),
//             DCB=(RECFM=FBS,LRECL=1,BLKSIZE=5120),
//             DISP=(NEW,CATLG,DELETE)
//CACINDX  DD  UNIT=SYSALLDA,VOL=SER=OP1TSD,
//             DSN=NALUR1.CAC.PRODUCT.CATINDX,
//             SPACE=(CYL,(2,1)),
//             DCB=(RECFM=FBS,LRECL=1,BLKSIZE=5120),
//             DISP=(NEW,CATLG,DELETE)
//*
//CACCATIN EXEC PGM=CACCATUT,PARM='INIT'
//STEPLIB  DD  DISP=SHR,DSN=CAC.SCACLOAD
//ENGCAT   DD  DISP=SHR,DSN=NALUR1.CAC.PRODUCT.SCACMENU
//CTRANS   DD  DISP=SHR,DSN=CAC.SCACSASC
//CACCAT   DD  DISP=SHR,DSN=NALUR1.CAC.PRODUCT.CATALOG
//CACINDX  DD  DISP=SHR,DSN=NALUR1.CAC.PRODUCT.CATINDX
//SYSTERM  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//*
```

Example: A-3 Update IBM InfoSphere Classic Federation Server system catalog

```
//CACMETAU JOB (POK,999),'METADATA UTILITY',CLASS=A,MSGCLASS=X,
//     NOTIFY=&SYSUID
//*********************************************************************
//*                                                                   *
//*   CACMETAU - JCL TO UPDATE THE SYSTEM CATALOG                     *
//*                                                                   *
//*   THIS JOB INVOKES THE META DATA UTILITY.                         *
//*   THE METADATA UTILITY CONNECTS TO DATA THE SERVER IDENTIFIED BY *
//*   THE CONNECT TO SERVER STATEMENT. A SAMPLE CONNECT TO STATEMENT *
//*   IS PROVIDED IN THE SCACCONF CACMETAU MEMBER.                    *
```

```
//*                                                                     *
//*   THE METADATA UTILITY THEN READS THE DDL STATEMENTS FROM SYSIN  *
//*   AND SENDS THE STATEMENTS TO THE SERVER IDENTIFIED IN THE       *
//*   CONNECT TO SERVER STATEMENT TO UPDATE THE SYSTEM CATALOG.      *
//*   THE METADATA UTILITY ALSO ACCEPTS CONNECT TO DB2 AND DB2       *
//*   IMPORT STATEMENTS THAT CAUSE THE METADATA UTILITY TO ACCESS A  *
//*   LOCAL DB2 SUBSYSTEM TO EXTRACT THE REQUIRED INFORMATION TO     *
//*   GENERATE CREATE TABLE AND INDEX STATEMENTS FOR DB2 OBJECTS.    *
//*                                                                     *
//*   1) PROVIDE A JOB CARD THAT IS VALID FOR YOUR SITE              *
//*   2) CHANGE CAC PARM TO INSTALLED HIGH LEVEL QUALIFIER           *
//*   3) UNCOMMENT THE DB2 PARM AND THEN CHANGE TO THE APPROPRIATE   *
//*      SYSTEM HLQ IF YOU ARE IMPORTING DB2 DEFINITIONS             *
//*   4) TAILOR CONNECT MEMBER (CACMETAU) AND PROVIDE SERVER         *
//*      CONNECTION AND IDENTIFICATION INFORMATION                   *
//*   5) CHANGE THE DDLIN PARM TO THE MEMBER THAT CONTAINS THE       *
//*      DDL STATEMENTS TO BE PROCESSED                              *
//*   6) UPDATE THE RGN PARAMETER IF YOU NEED TO PROCESS LARGE       *
//*      DDL STATEMENTS. IF 'OUT-OF-MEMORY' ERRORS ARE REPORTED BY   *
//*      THE METADATA UTILITY THEN THE REGION SIZE NEEDS TO BE       *
//*      INCREASED. INCRESE THE REGION SIZE IN TWO MEGA-BYTE         *
//*      INCREMENTS.                                                 *
//*                                                                     *
//*********************************************************************
//*
//METAUTL PROC CAC='CAC',          INSTALLED HIGH LEVEL QUALIFIER
//             CONNECT=CACMUCON,  SAMPLE CONFIGURATION MEMBER
//             DB2='DB8A8',         DB2 HIGH LEVEL QUALIFIER
//             DDLIN=CACDB2P,    INPUT DDL STATEMENT MEMBER NAME
//             RGN=8M,             REGION SIZE
//             SOUT='*'            SYSOUT CLASS
//*
//*********************************************************************
//METAU   EXEC PGM=CACMETA,REGION=&RGN
//STEPLIB  DD  DISP=SHR,DSN=&CAC..SCACLOAD
//         DD  DISP=SHR,DSN=&DB2..SDSNLOAD
//*
//CTRANS   DD  DISP=SHR,DSN=&CAC..SCACSASC
//*
//CACCAT   DD DISP=SHR,DSN=NALUR1.CAC.PRODUCT.CATALOG
//CACINDX  DD DISP=SHR,DSN=NALUR1.CAC.PRODUCT.CATINDX
//*
//ENGCAT   DD  DISP=SHR,DSN=NALUR1.CAC.PRODUCT.SCACMENU
//SYSTERM  DD  SYSOUT=&SOUT
//SYSPRINT DD  SYSOUT=&SOUT
```

```
//SYSIN    DD  DISP=SHR,DSN=&CAC..SCACCONF(&CONNECT)
//         DD  DISP=SHR,DSN=&CAC..SCACSAMP(&DDLIN)
//        PEND
//METAUTL EXEC METAUTL
```

Example: A-4   Contents of CACMUCON file

```
CONNECT TO SERVER CACSAMP "TCP/0.0.0.0/5525";
```

Example: A-5   Product VSAM file DDL definition

```
DROP TABLE CAC.PRODUCT;
USE TABLE CAC.PRODUCT DBTYPE VSAM
  DS 'NALUR1.CAC.VSAM.PRODUCT' (
 PRODUCT_ID SOURCE DEFINITION DATAMAP
  OFFSET 0 LENGTH 4 DATATYPE C USE AS DECIMAL(6),
 DESCRIPTION SOURCE DEFINITION DATAMAP
  OFFSET 5 LENGTH 50 DATATYPE C USE AS CHAR(50),
 BRAND SOURCE DEFINITION DATAMAP
  OFFSET 55 LENGTH 50 DATATYPE C USE AS CHAR(50),
 CATEGORY SOURCE DEFINITION DATAMAP
  OFFSET 105 LENGTH 50 DATATYPE C USE AS CHAR(50),
 FACTORY SOURCE DEFINITION DATAMAP
  OFFSET 155 LENGTH 50 DATATYPE C USE AS CHAR(50),
 SUPPLIER SOURCE DEFINITION DATAMAP
  OFFSET 205 LENGTH 50 DATATYPE C USE AS CHAR(50),
 SKU SOURCE DEFINITION DATAMAP
  OFFSET 255 LENGTH 50 DATATYPE C USE AS CHAR(50));
```

Example: A-6   Store VSAM file DDL definition

```
DROP TABLE CAC.STORE;
USE TABLE CAC.STORE DBTYPE VSAM
  DS 'NALUR1.CAC.VSAM.STORE' (
 STORE_ID SOURCE DEFINITION DATAMAP
  OFFSET 0 LENGTH 4 DATATYPE C USE AS DECIMAL(6),
 ADDRESS SOURCE DEFINITION DATAMAP
  OFFSET 5 LENGTH 50 DATATYPE C USE AS CHAR(50),
 CITY SOURCE DEFINITION DATAMAP
  OFFSET 55 LENGTH 50 DATATYPE C USE AS CHAR(50),
 CITY_POPULATION SOURCE DEFINITION DATAMAP
  OFFSET 105 LENGTH 8 DATATYPE C USE AS DECIMAL(10),
 STATE SOURCE DEFINITION DATAMAP
```

```
 OFFSET 114 LENGTH 50 DATATYPE C USE AS CHAR(50),
STATE_POPULATION SOURCE DEFINITION DATAMAP
 OFFSET 164 LENGTH 8 DATATYPE C USE AS DECIMAL(10),
ZIP SOURCE DEFINITION DATAMAP
 OFFSET 173 LENGTH 15 DATATYPE C USE AS CHAR(15),
COUNTRY SOURCE DEFINITION DATAMAP
 OFFSET 188 LENGTH 50 DATATYPE C USE AS CHAR(50),
MANAGER_ID SOURCE DEFINITION DATAMAP
 OFFSET 238 LENGTH 4 DATATYPE C USE AS DECIMAL(6));
```

A.2.3 Configuration of Classic Data Architect

In this section, Figure A-1 on page 575 through Figure A-8 on page 580 show how Classic Data Architect (CDA) is used to access the Product (VSAM) file as a logical relational table:

1. Launch CDA from your desktop (not shown here), choose the workspace for your session, and click **OK** as shown in Figure A-1 on page 575.
2. Right-click **Connections** in Database Explorer and select **New Connection** as shown in Figure A-2 on page 575.
3. Provide details of the database manager, JDBC driver and required connection parameters including User ID and Password, and click **Test Connection.**
4. A successful connection is shown in Figure A-4 on page 576. Click **OK**.
5. Click **Next** in Figure A-5 on page 577 to specify any filter for the objects to view in the CACSAMP database.
6. Check the Disable filter box and click **Finish** in Figure A-6 on page 578 to proceed to view all the objects in the CACSAMP database.
7. Expand the navigation tree in the CACSAMP database, and right-click **PRODUCT (VSAM)** → **Data** → **Sample Contents** as shown in Figure A-7 on page 579 to view the contents of the PRODUCT table (4 rows is shown under the **Data Output** tab in Figure A-8 on page 580).

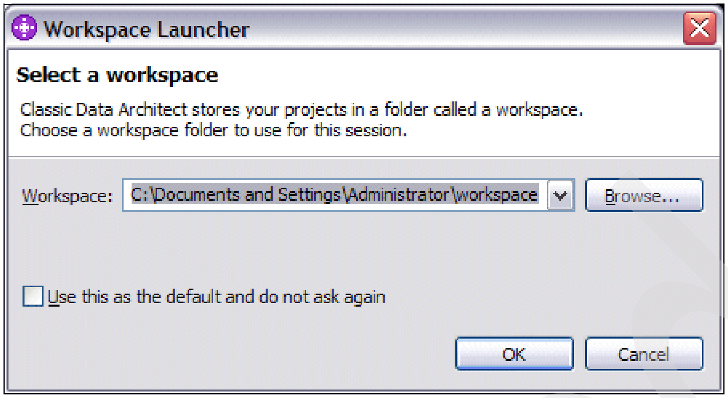


Figure A-1 Configure access to PRODUCT VSAM file 1/8

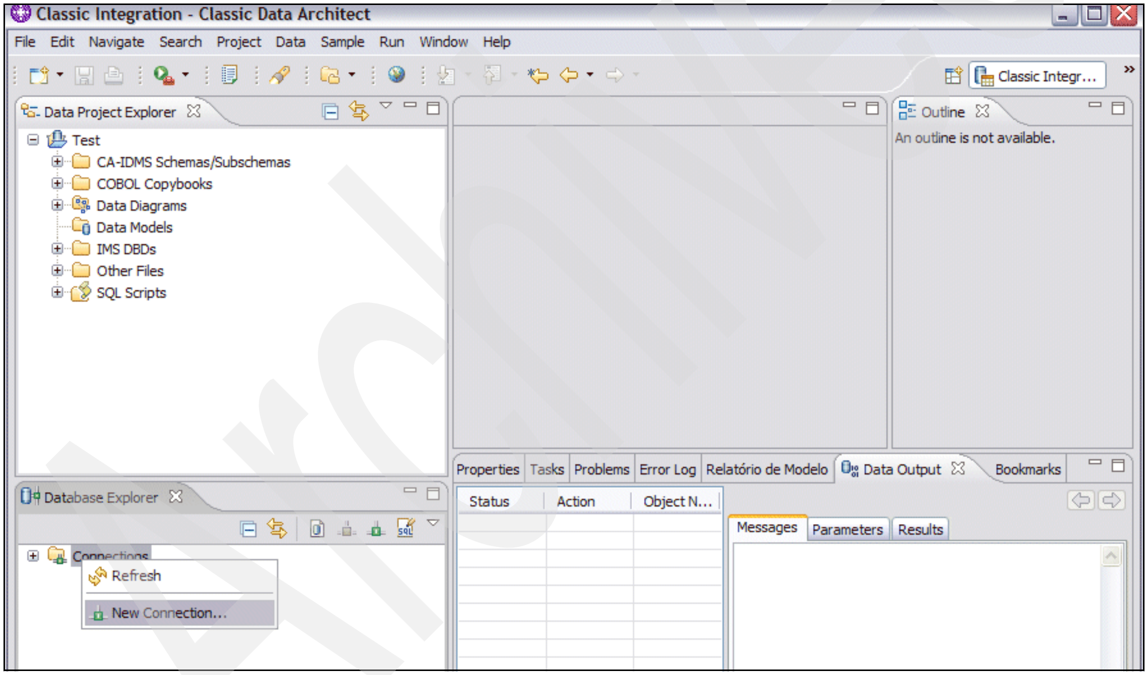


Figure A-2 Configure access to PRODUCT VSAM file 2/8

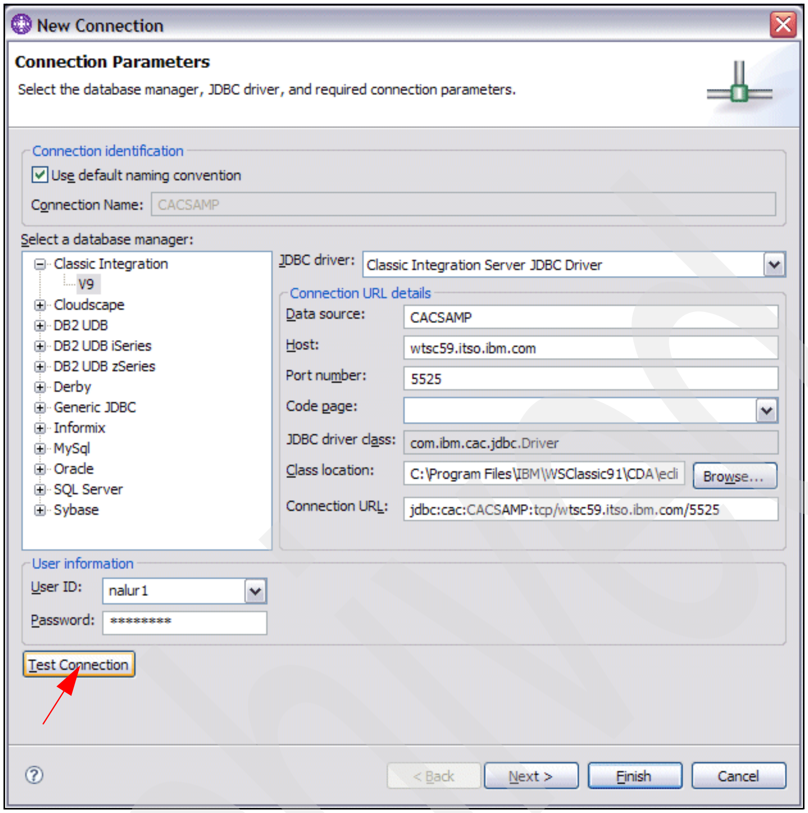


Figure A-3 Configure access to PRODUCT VSAM file 3/8

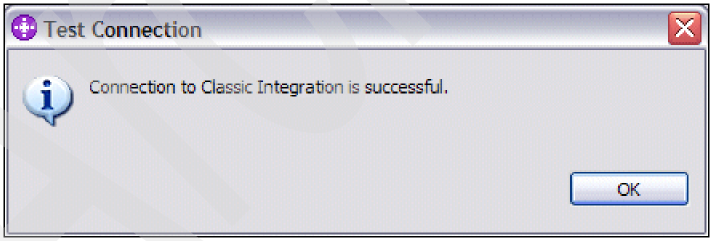


Figure A-4 Configure access to PRODUCT VSAM file 4/8

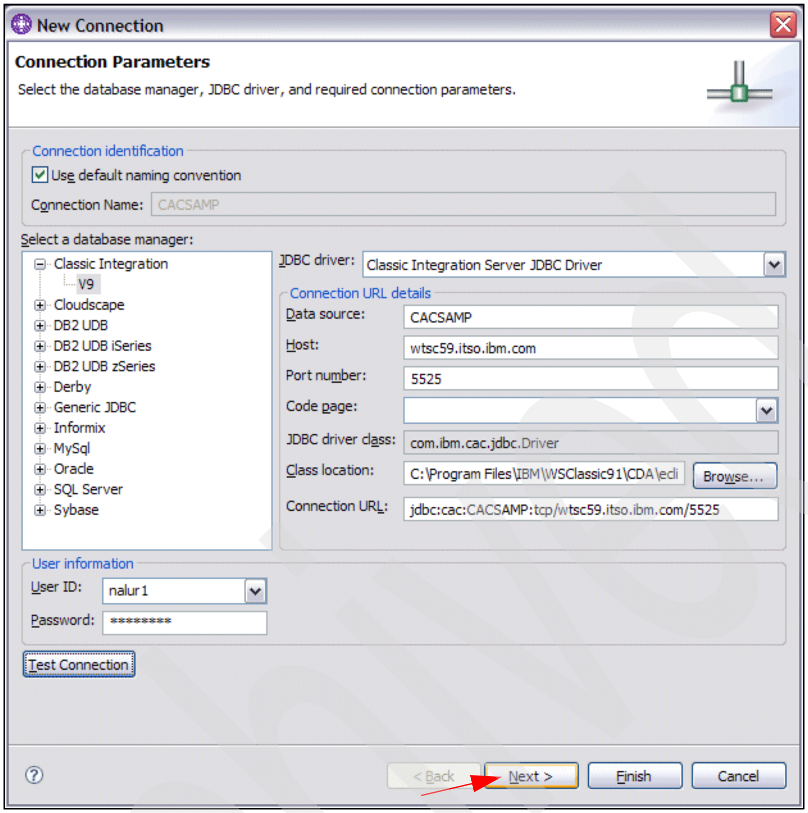


Figure A-5 Configure access to PRODUCT VSAM file 5/8

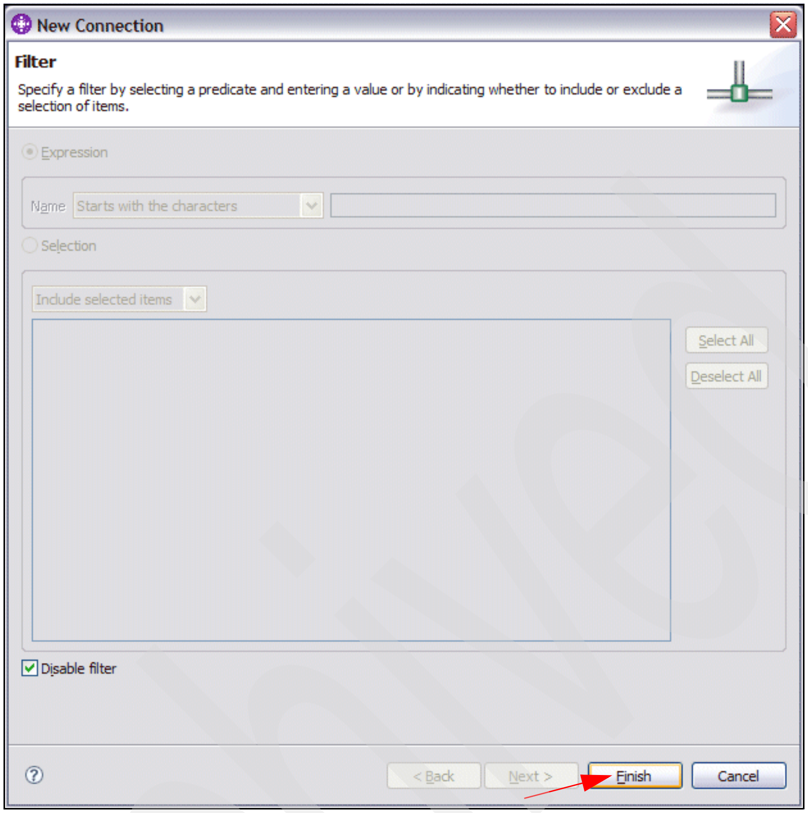


Figure A-6 Configure access to PRODUCT VSAM file 6/8

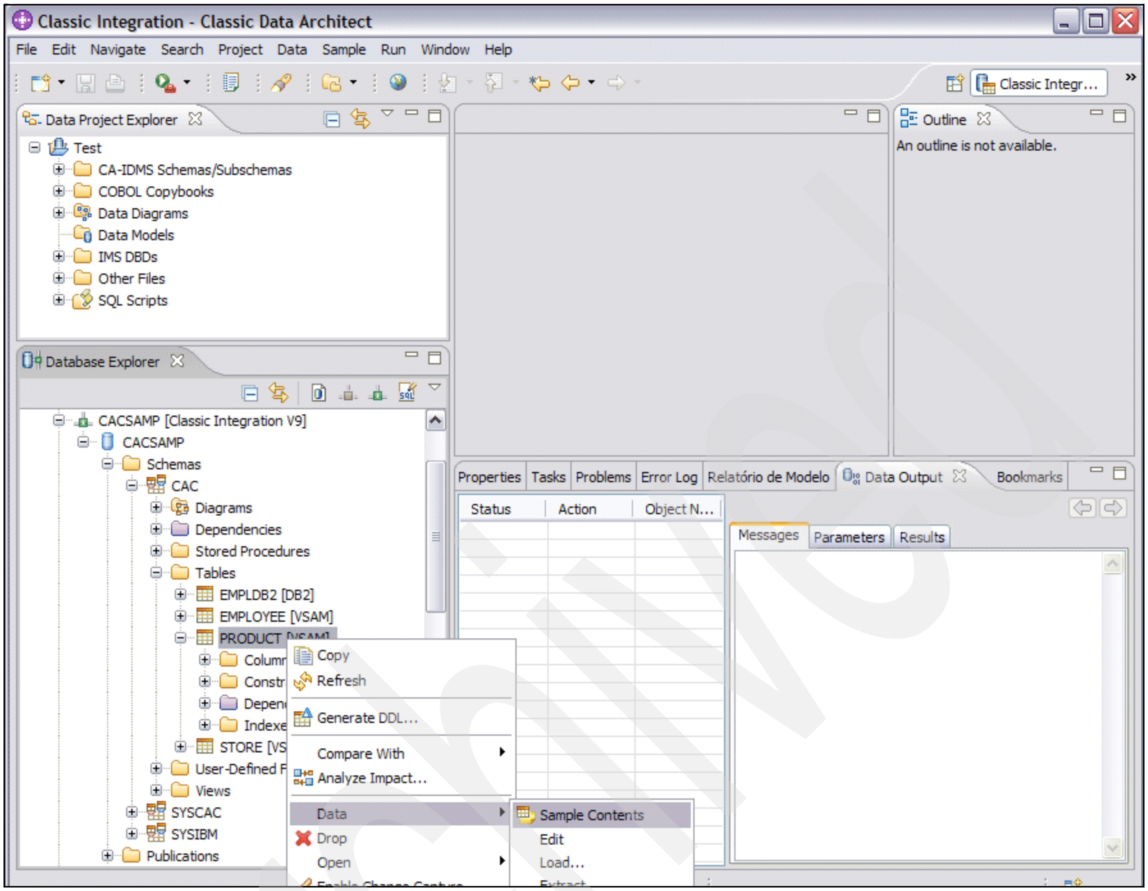


Figure A-7 Configure access to PRODUCT VSAM file 7/8

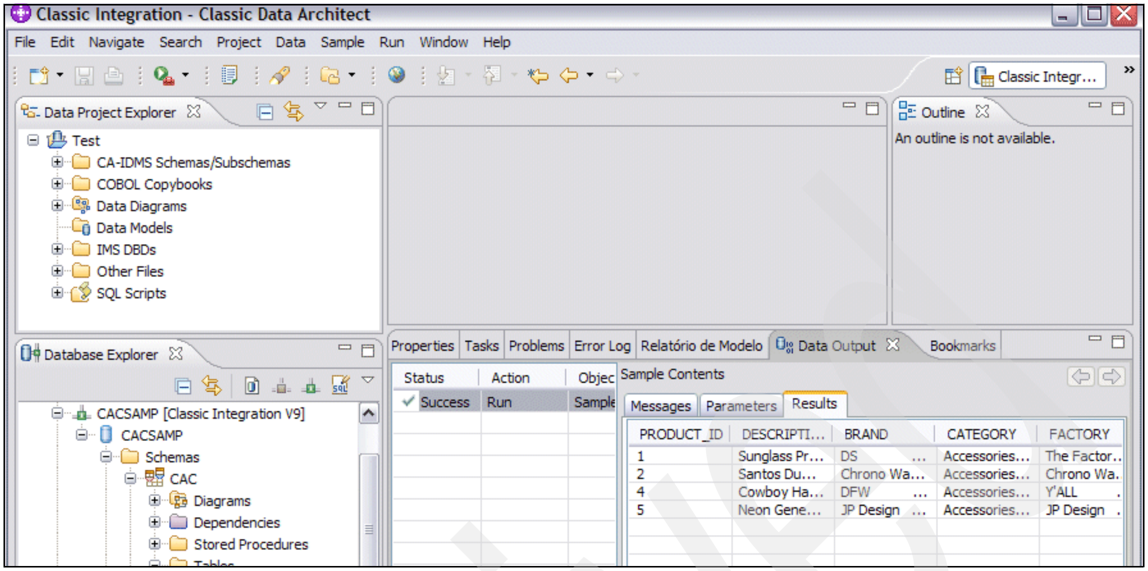


Figure A-8 Configure access to PRODUCT VSAM file 8/8

A.3 Create the Queue Manager

In this section, Figure A-9 on page 581 through Figure A-16 on page 587 show the creation of a queue manager[5] QM_Kazan using WebSphere MQ Explorer, as follows:

1. From the WebSphere MQ Explorer window, expand the IBM WebSphere MQ label, then right-click **Queue Managers** and select **New** → **Queue Manager** from the pop-up menu as shown in Figure A-9 on page 581.
2. In Figure A-10 on page 582, provide the name of the queue manager (QM_Kazan) and other details, and check the box to make this your default queue manager. Click **Next**.
3. Specify the type of logging that the queue manager will perform, and the maximum number of log files that can be produced in Figure A-11 on page 583, and click **Next**.
4. Check the Start queue manager box and click **Next** in Figure A-11 on page 583.

[5] Before you use the WebSphere MQ applications, you must create a queue manager. The queue manager is a system program that is responsible for maintaining the queues and ensuring that the messages in the queues reach their destination. It also performs other functions associated with message queuing.

5. In Figure A-12 on page 584, specify the information that enables the WebSphere MQ applications that are running on your machine to communicate with other machines. Check the Create listener configured for TCP/IP box, and enter the port number for WebSphere MQ (default is 1414) as shown in Figure A-13 on page 585. Click **Next** to continue.
6. Check the Autoreconnect and Automatically refresh information shown for this queue manager boxes in Figure A-14 on page 586, and click **Finish** to create your queue manager. It might take a minute to create and start the queue manager as shown in Figure A-15 on page 586.
7. On successful creation and startup, the status of this queue manager QM_Kazan is shown in Figure A-16 on page 587.

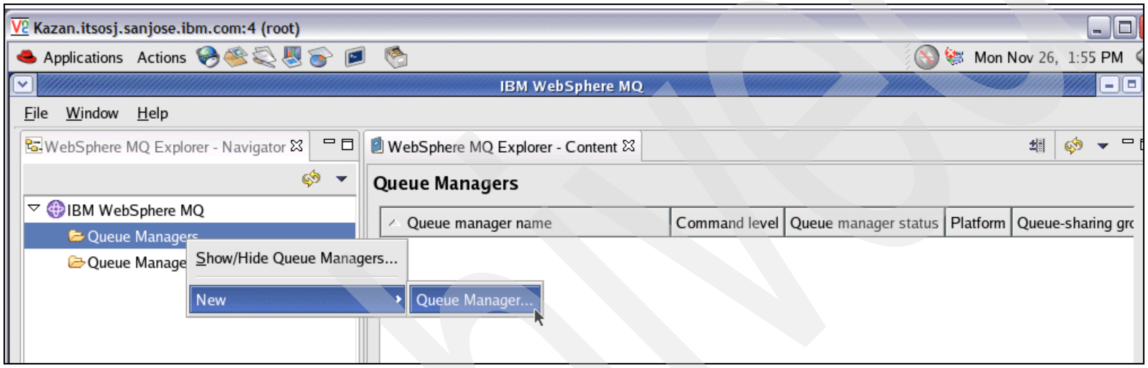


Figure A-9 Create the Queue Manager 1/8

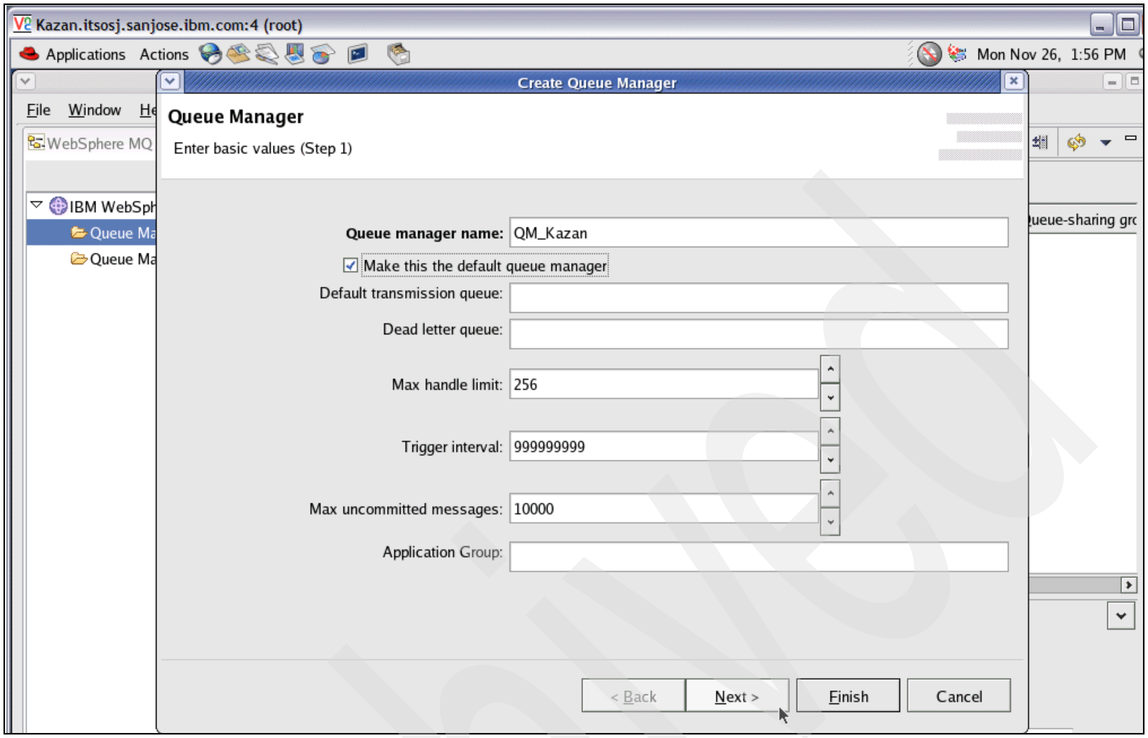


Figure A-10   Create the Queue Manager 2/8

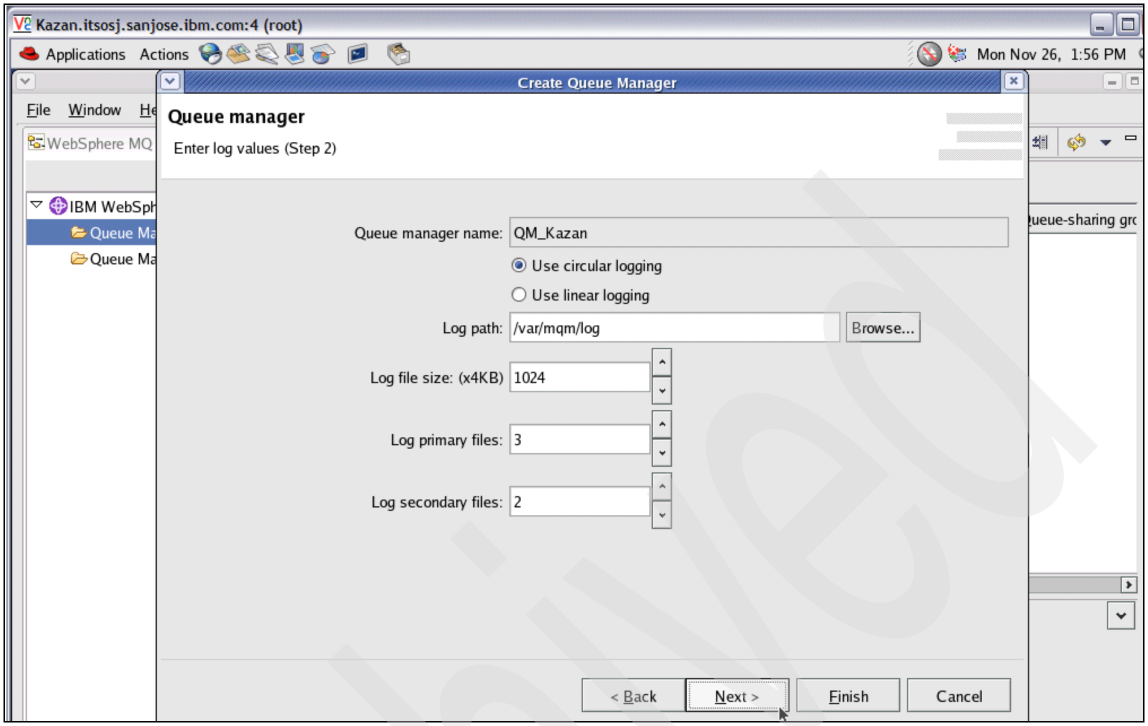


Figure A-11 Create the Queue Manager 3/8

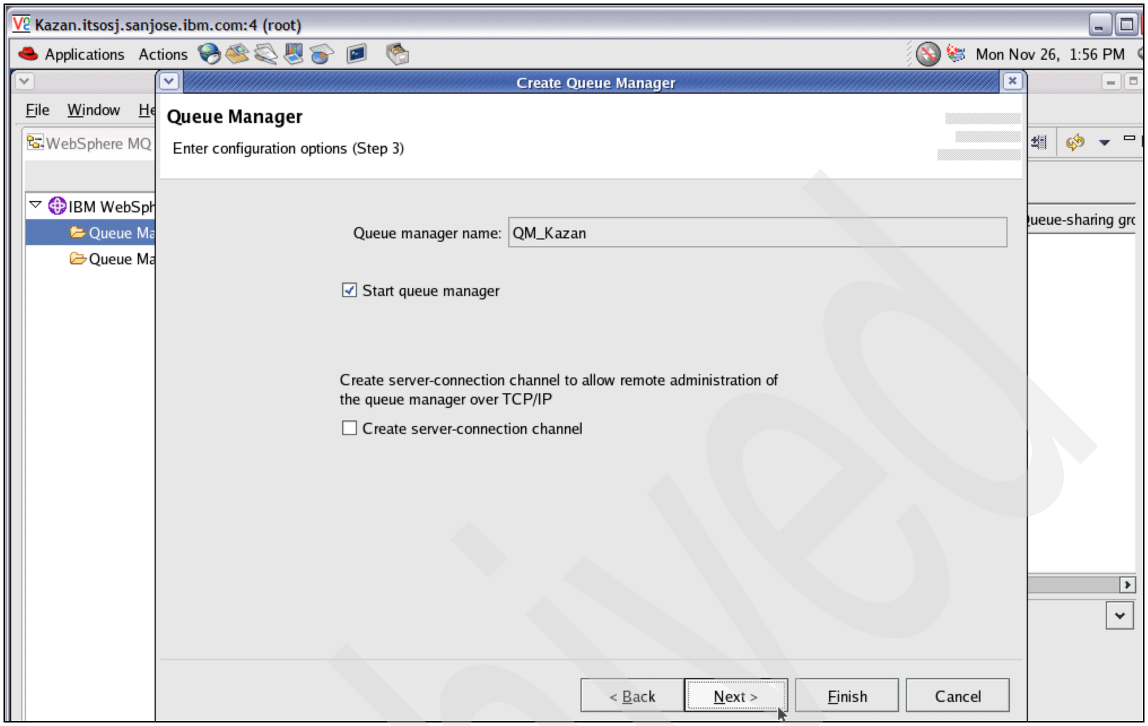


Figure A-12 Create the Queue Manager 4/8

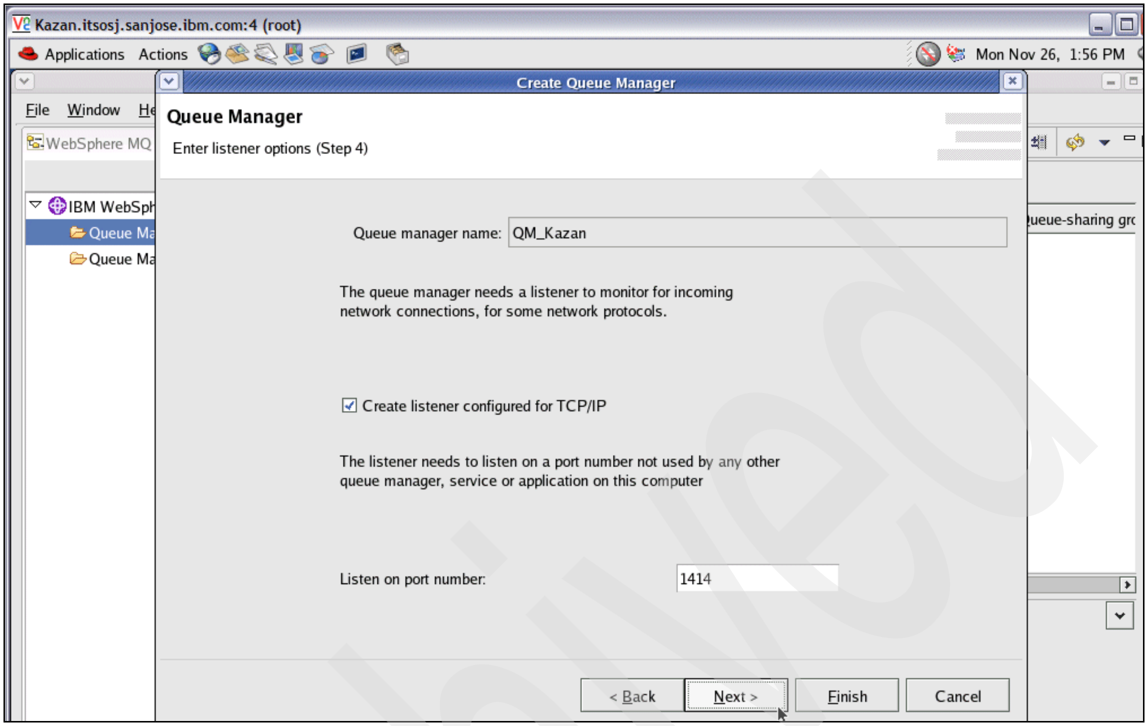


Figure A-13 Create the Queue Manager 5/8

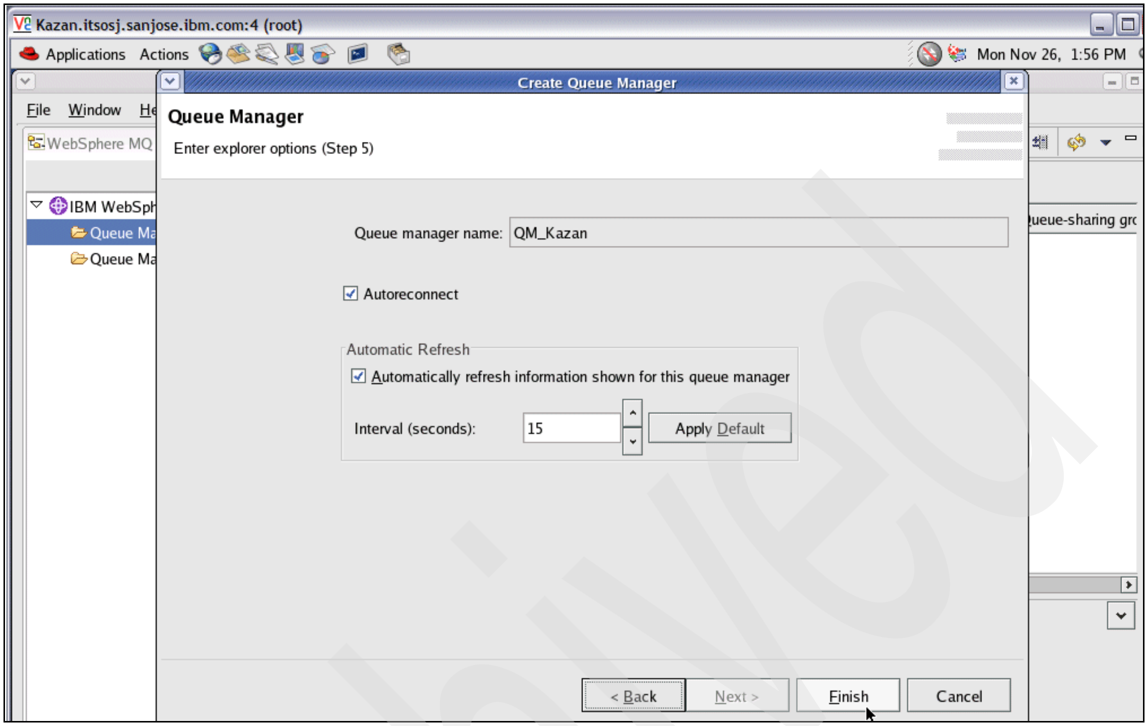


Figure A-14 Create the Queue Manager 6/8

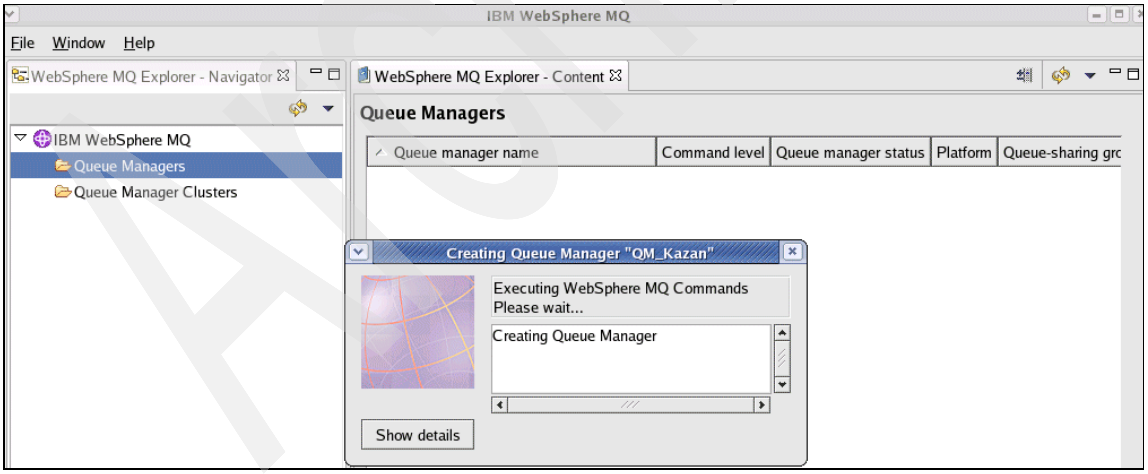


Figure A-15 Create the Queue Manager 7/8

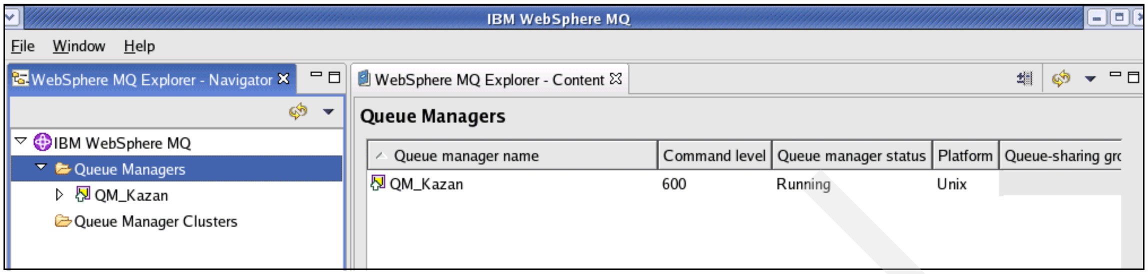


Figure A-16 Create the Queue Manager 8/8

A.4 Set up the XA parameters on Queue Manager

IBM WebSphere MQ has to be configured for the distributed transaction support provided by the Distributed Transaction stage described in 2.7, “Distributed Transaction (new in Version 8.1)” on page 63 as follows:

1. Ensure that your DB2 environment variables are set for queue manager processes as well as in your application processes. In particular, you must always set the DB2INSTANCE environment variable *before you start* the queue manager. The DB2INSTANCE environment variable identifies the DB2 instance containing the DB2 databases that are being updated. For example:

   ```
   set DB2INSTANCE=DB2
   ```

2. Copy db2swit.dll[6] to the appropriate directory (default location is C:\Program Files\IBM\WebSphere MQ\exits on the Microsoft Windows platform) of IBM WebSphere MQ.
3. Launch IBM WebSphere MQ Explorer (not shown here), right-click the queue manager (QM_Kazan) to configure, and select **Properties...** as shown in Figure A-17 on page 588.
4. Click **XA resource managers** from the Properties dialog and click the **Add...** button to add an XA resource as shown in Figure A-18 on page 589.

[6] The switch load file is a shared library (a DLL on Windows systems) that is loaded by the code in your WebSphere MQ application and the queue manager. Its purpose is to simplify the loading of the database's client shared library, and to return the pointers to the XA functions. The details of the switch load file must be specified before the queue manager is started.

5. In the dialog shown in Figure A-19 on page 590, enter the values for Name (db2) and SwitchFile (which matches the name of the DLL you copied above). The XAOpenString is composed of these components:-

   – databaseName,username,password,toc=c

   – toc=p means 'thread of control is thread'. Include this in the XAOpenString and make sure you set ThreadOfControl to Thread.

   Click **OK** in Figure A-19 on page 590 to complete the successful application of the changes is shown in Figure A-20 on page 590.

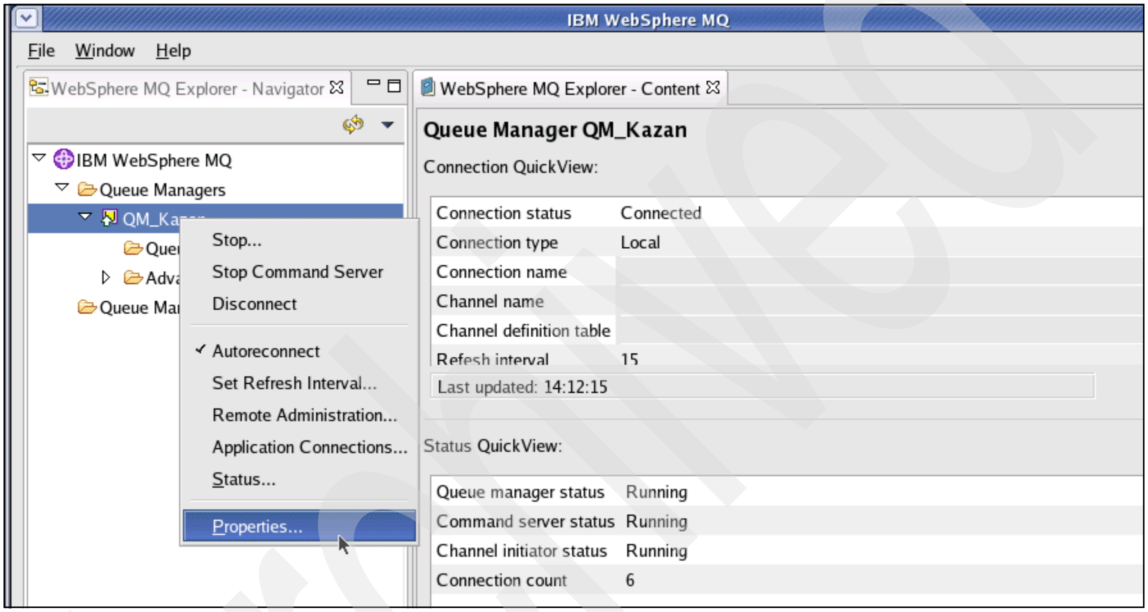


Figure A-17 Set up the XA parameters on Queue Manager 1/4

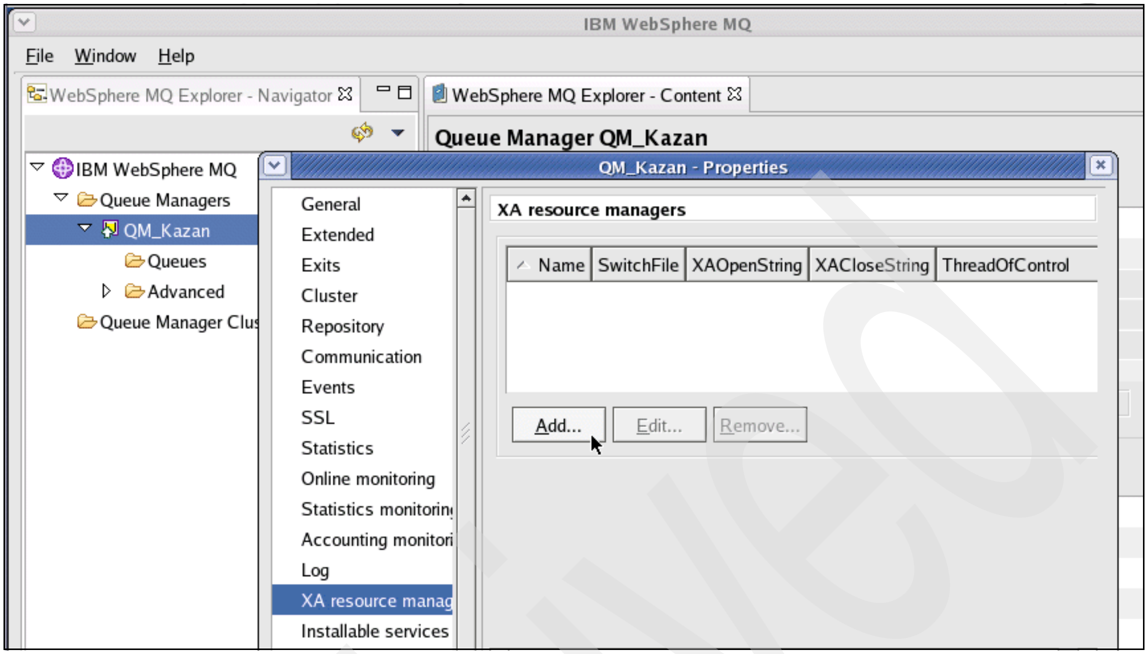


Figure A-18 Set up the XA parameters on Queue Manager 2/4

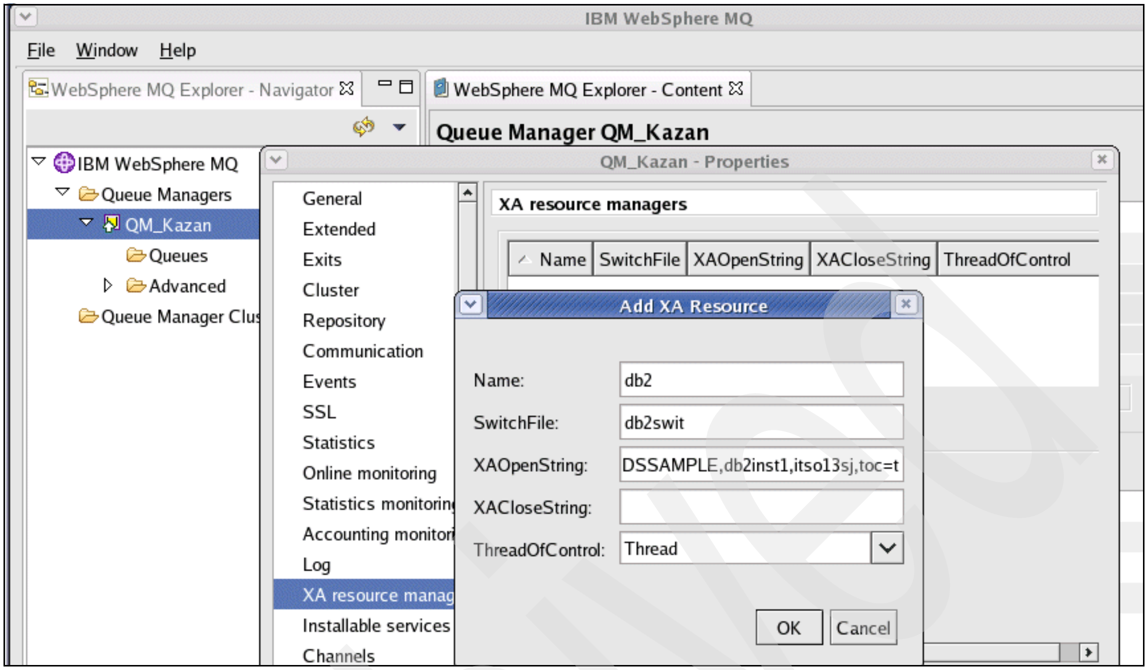


Figure A-19 Set up the XA parameters on Queue Manager 3/4

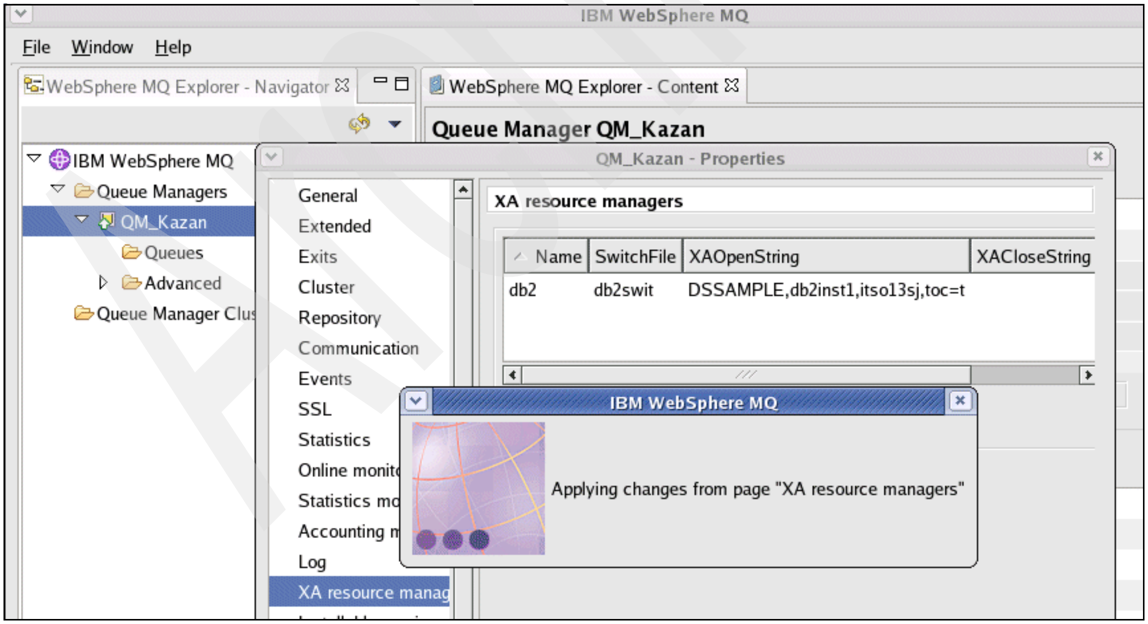


Figure A-20 Set up the XA parameters on Queue Manager 4/4

A.5 Create the queues

The Distributed Transaction stage uses the following IBM WebSphere MQ queues:

- SOURCEQ is the source queue for the Distributed Transaction stage jobs.
- WORKQ is the work queue used by the Distributed Transaction stage jobs.
- REJECTQ is the reject queue used by job RejectTransaction.

Figure A-21 on page 592 through Figure A-26 on page 595 show the definition of the SOURCEQ using IBM WebSphere MQ Explorer:

1. Expand the navigation tree and right-click **Queues** (under the QM_Kazan queue manager), and then select **New** → **Local Queue** from the pop-up menu as shown in Figure A-21.
2. Provide details of the local queue to be create such as Name (SOURCEQ) and model it with the attributes of the SYSTEM.DEFAULT.LOCAL.QUEUE. Click **Next** in Figure A-22 on page 592 to view and change the properties of the queue.
3. Change the properties as required and click **Finish** as shown in Figure A-23 on page 593.
4. The successful creation of this queue is shown in Figure A-24 on page 594 Figure A-25 on page 594.
5. Figure A-26 on page 595 shows the three local queues (SOURCEQ, REJECTQ, and WORKQ) created for the Distributed Transaction stage.

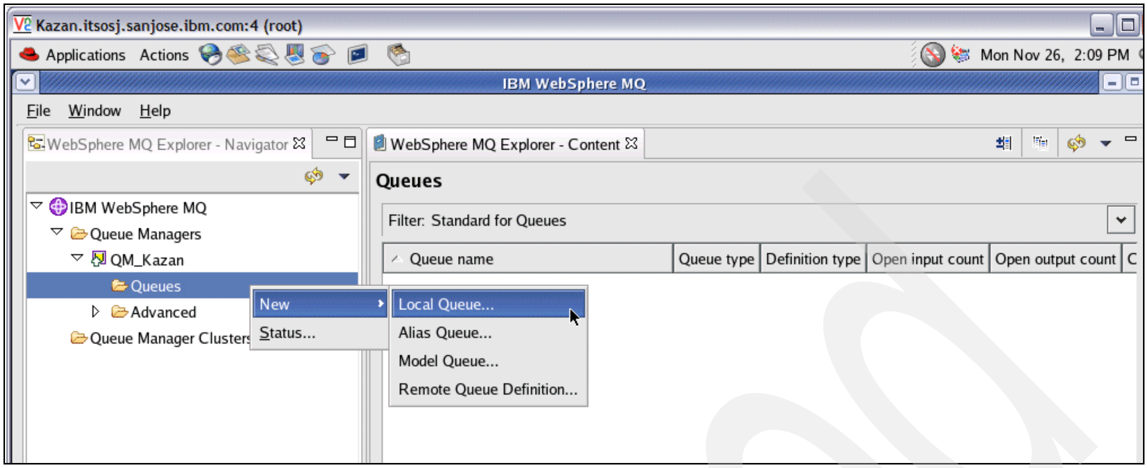


Figure A-21 Create the queues 1/6

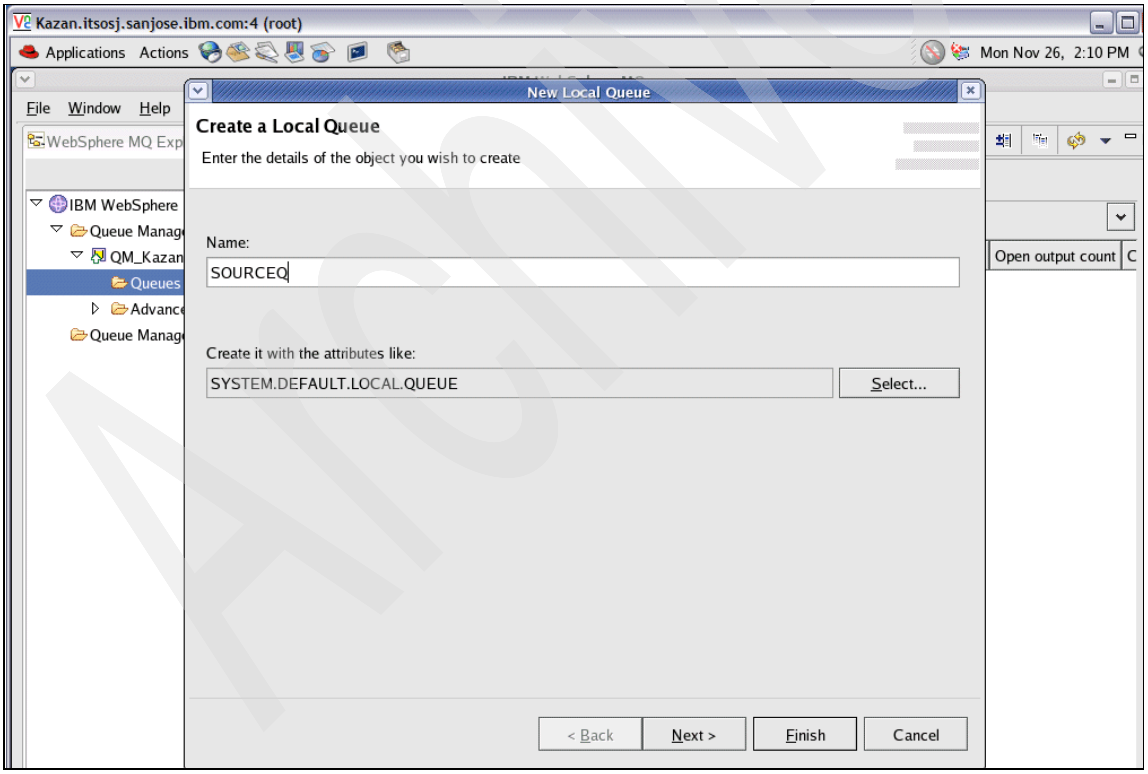


Figure A-22 Create the queues 2/6

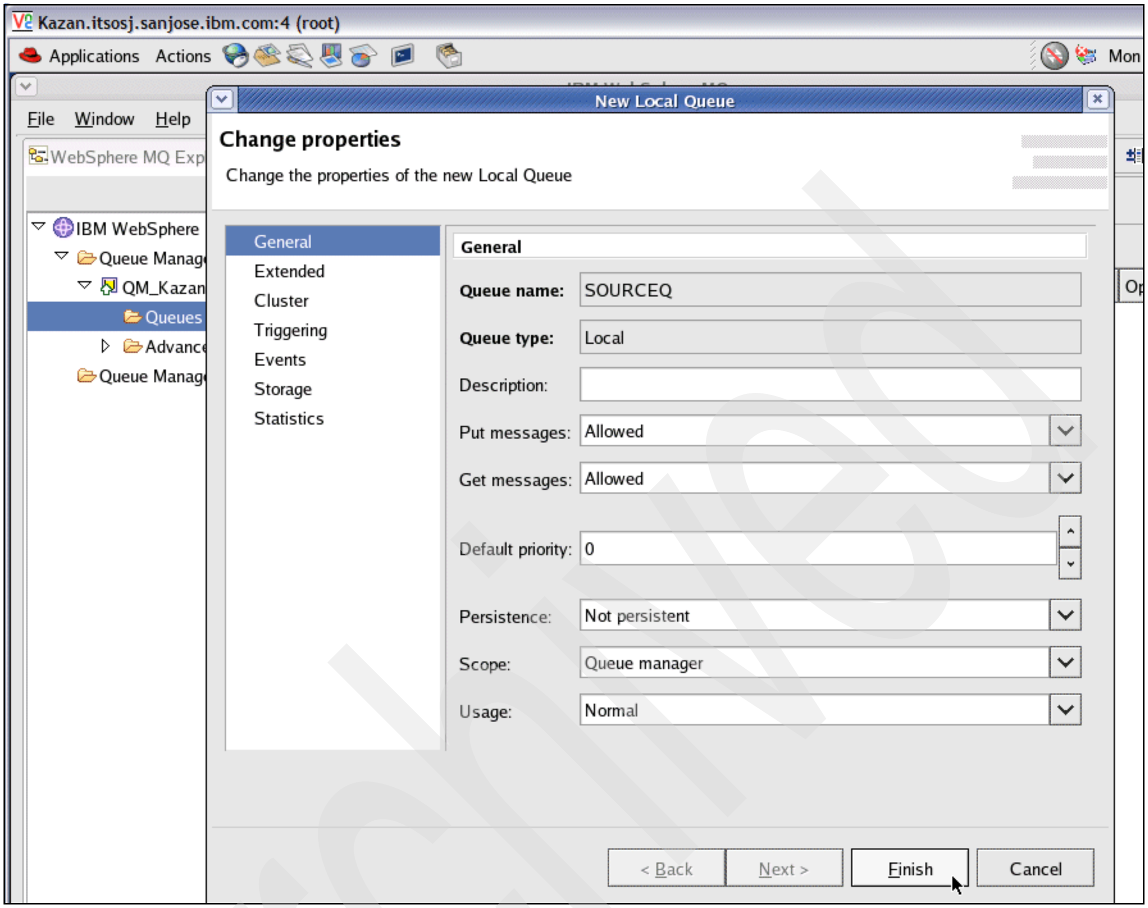


Figure A-23 Create the queues 3/6

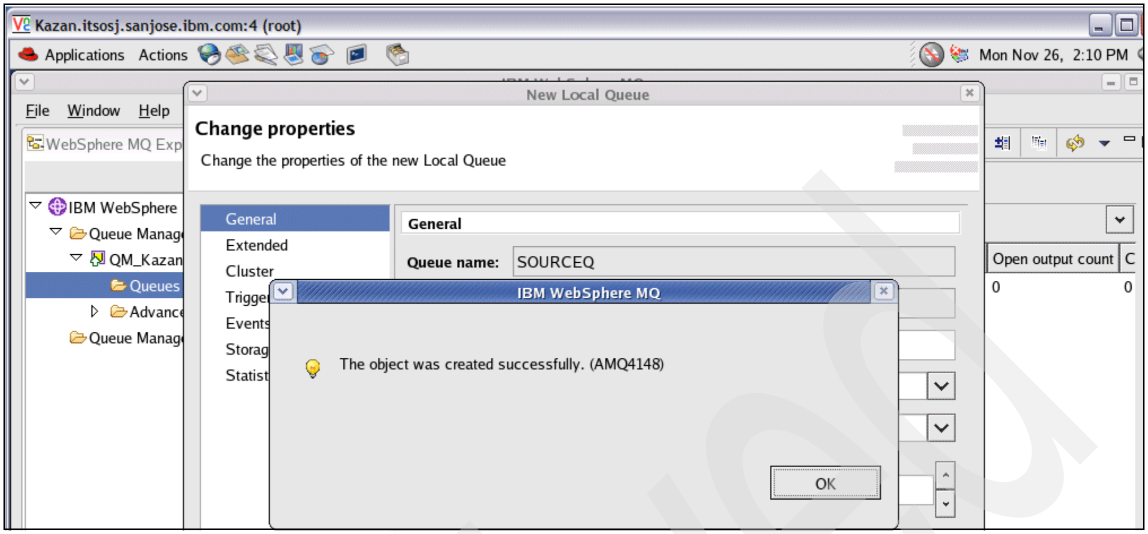


Figure A-24 Create the queues 4/6

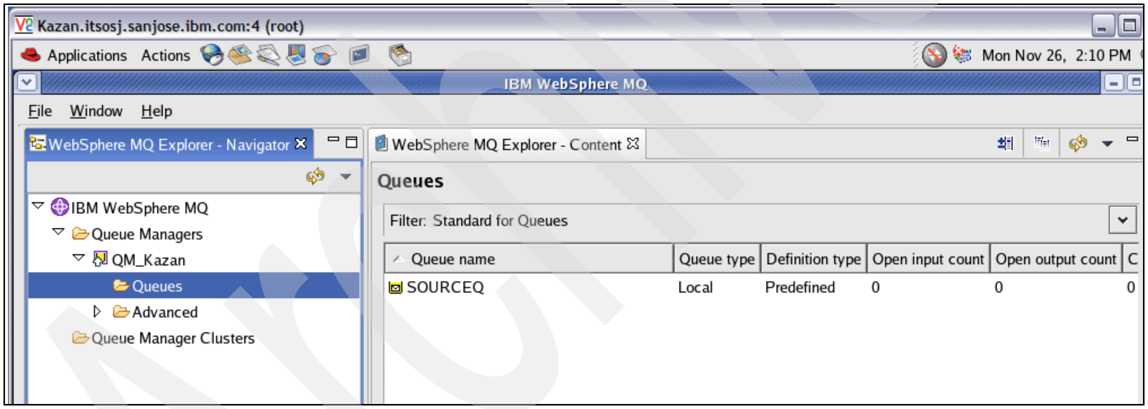


Figure A-25 Create the queues 5/6

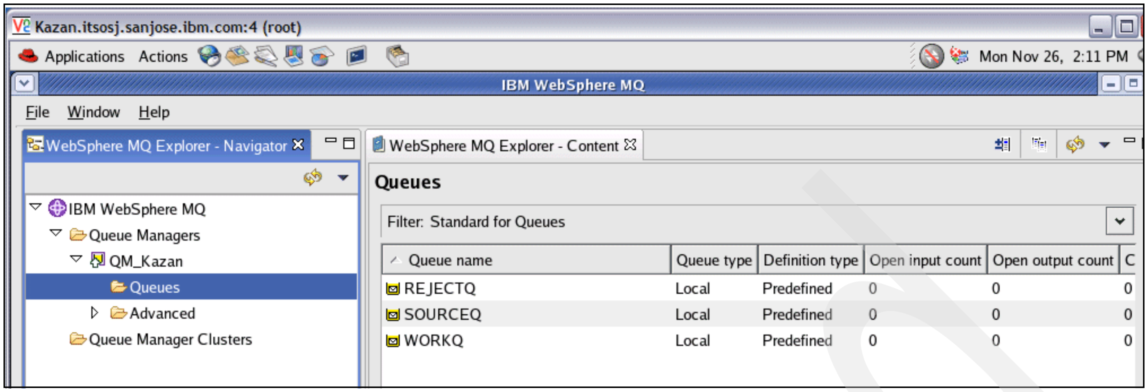


Figure A-26 Create the queues 6/6

B

Code and scripts used in the retail industry scenario

In this appendix we document some of the code and scripts used in the retail industry scenario.

B.1 Introduction

This appendix documents some of the code and scripts used in the retail industry scenario, as follows:

- Figure B-1 here shows the entities and fields in WantThatStuff's OLTP systems. Product and Store are VSAM files, while the others are sequential files.
- Example B-1 on page 599 shows the DDL for creating the tables in WantThatStuff's star-schema data warehouse.
- Example B-2 on page 603 shows the DDL for creating the interim sales transaction tables used in WantThatStuff's recurring tasks.

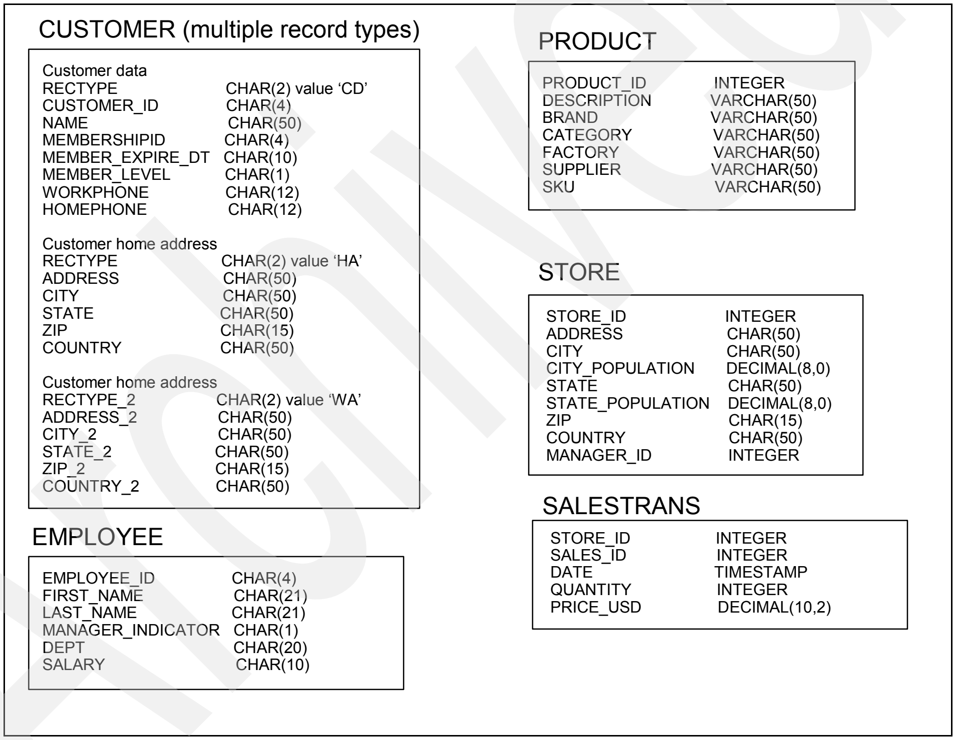


Figure B-1 Entities and fields in WantThatStuff's OLTP systems

Example: B-1   DDL statements in the WantThatStuff star-schema data warehouse

```
-- This CLP file was created using DB2LOOK Version 9.1
-- Timestamp: Mon Mar  3 14:55:15 CST 2008
-- Database Name: DSSAMPLE
-- Database Manager Version: DB2/AIX64 Version 9.1.3
-- Database Codepage: 819
-- Database Collating Sequence is: UNIQUE
CONNECT TO DSSAMPLE;
------------------------------------------------
-- DDL Statements for table "DS      "."CUSTOMER_DIM"
------------------------------------------------
CREATE TABLE "DS      "."CUSTOMER_DIM"  (
         "CUSTOMER_DIM_KEY" INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY (
           START WITH +700
           INCREMENT BY +1
           MINVALUE +700
           MAXVALUE +2147483647
           NO CYCLE
           CACHE 20
           NO ORDER ) ,
         "CUSTOMER_ID" INTEGER ,
         "NAME" VARCHAR(50) ,
         "HOME_PHONE" CHAR(12) ,
         "WORK_PHONE" CHAR(12) ,
         "WORK_ADDRESS" VARCHAR(50) ,
         "WORK_CITY" VARCHAR(50) ,
         "WORK_STATE" VARCHAR(50) ,
         "WORK_ZIP" VARCHAR(15) ,
         "WORK_COUNTRY" VARCHAR(50) ,
         "HOME_ADDRESS" VARCHAR(50) ,
         "HOME_CITY" VARCHAR(50) ,
         "HOME_ZIP" VARCHAR(15) ,
         "HOME_STATE" VARCHAR(50) ,
         "HOME_COUNTRY" VARCHAR(50) ,
         "MEMBERSHIP_ID" INTEGER ,
         "MEMBERSHIP_EXPIRE_DT" DATE ,
         "MEMBERSHIP_LEVEL" CHAR(1) ,
         "CURRENT_IND" CHAR(1) WITH DEFAULT 'Y' ,
         "EFFECTIVE_TS" TIMESTAMP WITH DEFAULT CURRENT TIMESTAMP ,
         "EXPIRATION_TS" TIMESTAMP WITH DEFAULT '2099-12-31-00.00.00.000000' )
        IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "DS      "."CUSTOMER_DIM"

ALTER TABLE "DS      "."CUSTOMER_DIM"
    ADD PRIMARY KEY
        ("CUSTOMER_DIM_KEY");

ALTER TABLE "DS      "."CUSTOMER_DIM" ALTER COLUMN "CUSTOMER_DIM_KEY" RESTART WITH 879;

------------------------------------------------
-- DDL Statements for table "DS      "."DATE_DIM"
------------------------------------------------
```

```
CREATE TABLE "DS      "."DATE_DIM"  (
         "DATE_DIM_KEY" INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY (
           START WITH +700
           INCREMENT BY +1
           MINVALUE +700
           MAXVALUE +2147483647
           NO CYCLE
           CACHE 20
           NO ORDER ) ,
         "DATE" DATE NOT NULL ,
         "DAY_OF_WEEK" VARCHAR(20) ,
         "MONTH" CHAR(2) ,
         "QUARTER" CHAR(1) ,
         "YEAR" CHAR(4) ,
         "FISCAL_MONTH" CHAR(2) ,
         "FISCAL_QUARTER" CHAR(1) ,
         "FISCAL_YEAR" CHAR(4) ,
         "CURRENT_IND" CHAR(1) WITH DEFAULT 'Y' ,
         "EFFECTIVE_TS" TIMESTAMP WITH DEFAULT CURRENT TIMESTAMP ,
         "EXPIRATION_TS" TIMESTAMP WITH DEFAULT '2099-12-31-00.00.00.000000' )
        IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "DS      "."DATE_DIM"

ALTER TABLE "DS      "."DATE_DIM"
    ADD PRIMARY KEY
        ("DATE_DIM_KEY");

------------------------------------------------
-- DDL Statements for table "DS      "."PRODUCT_DIM"
------------------------------------------------
CREATE TABLE "DS      "."PRODUCT_DIM"  (
         "PRODUCT_DIM_KEY" INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY (
           START WITH +700
           INCREMENT BY +1
           MINVALUE +700
           MAXVALUE +2147483647
           NO CYCLE
           CACHE 20
           NO ORDER ) ,
         "PRODUCT_ID" INTEGER ,
         "DESCRIPTION" VARCHAR(50) ,
         "BRAND" VARCHAR(50) ,
         "CATEGORY" VARCHAR(50) ,
         "FACTORY" VARCHAR(50) ,
         "SUPPLIER" VARCHAR(50) ,
         "SKU" VARCHAR(50) ,
         "CURRENT_IND" CHAR(1) WITH DEFAULT 'Y' ,
         "EFFECTIVE_TS" TIMESTAMP WITH DEFAULT CURRENT TIMESTAMP ,
         "EXPIRATION_TS" TIMESTAMP WITH DEFAULT '2099-12-31-00.00.00.000000' )
        IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "DS      "."PRODUCT_DIM"
```

```
ALTER TABLE "DS      "."PRODUCT_DIM"
    ADD PRIMARY KEY
        ("PRODUCT_DIM_KEY");

ALTER TABLE "DS      "."PRODUCT_DIM" ALTER COLUMN "PRODUCT_DIM_KEY" RESTART WITH 799;

------------------------------------------------
-- DDL Statements for table "DS      "."SALES_FACT"
------------------------------------------------
CREATE TABLE "DS      "."SALES_FACT"  (
          "CUSTOMER_DIM_KEY" INTEGER NOT NULL ,
          "DATE_DIM_KEY" INTEGER NOT NULL ,
          "PRODUCT_DIM_KEY" INTEGER NOT NULL ,
          "QUANTITY" INTEGER ,
          "PRICE_USD" DECIMAL(10,2) ,
          "SELLING_PRICE_USD" DECIMAL(10,2) ,
          "TOTAL_USD" DECIMAL(10,2) ,
          "STORE_DIM_KEY" INTEGER NOT NULL ,
          "TOTAL_LOCAL_CURRENCY" DECIMAL(10,2) ,
          "COUNTRY_ISO_CODE" CHAR(3) )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "DS      "."SALES_FACT"

ALTER TABLE "DS      "."SALES_FACT"
    ADD PRIMARY KEY
        ("CUSTOMER_DIM_KEY",
         "DATE_DIM_KEY",
         "PRODUCT_DIM_KEY",
         "STORE_DIM_KEY");


------------------------------------------------
-- DDL Statements for table "DS      "."STORE_DIM"
------------------------------------------------
CREATE TABLE "DS      "."STORE_DIM"  (
          "STORE_DIM_KEY" INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY (
            START WITH +700
            INCREMENT BY +1
            MINVALUE +700
            MAXVALUE +2147483647
            NO CYCLE
            CACHE 20
            NO ORDER ) ,
          "STORE_ID" INTEGER ,
          "ADDRESS" VARCHAR(50) ,
          "CITY" VARCHAR(50) ,
          "CITY_POPULATION" DECIMAL(8,0) ,
          "STATE" CHAR(2) ,
          "STATE_POPULATION" DECIMAL(8,0) ,
          "ZIP" VARCHAR(15) ,
          "COUNTRY" VARCHAR(50) ,
          "MANAGER_NAME" VARCHAR(50) ,
          "CURRENT_IND" CHAR(1) WITH DEFAULT 'Y' ,
```

```
      "EFFECTIVE_TS" TIMESTAMP WITH DEFAULT CURRENT TIMESTAMP ,
      "EXPIRATION_TS" TIMESTAMP WITH DEFAULT '2099-12-31-00.00.00.000000' )
     IN "USERSPACE1" ;


-- DDL Statements for primary key on Table "DS      "."STORE_DIM"

ALTER TABLE "DS      "."STORE_DIM"
   ADD PRIMARY KEY
      ("STORE_DIM_KEY");

ALTER TABLE "DS      "."STORE_DIM" ALTER COLUMN "STORE_DIM_KEY" RESTART WITH 779;

-- DDL Statements for foreign keys on Table "DS      "."SALES_FACT"

ALTER TABLE "DS      "."SALES_FACT"
   ADD CONSTRAINT "SQL071121141338930" FOREIGN KEY
      ("PRODUCT_DIM_KEY")
   REFERENCES "DS      "."PRODUCT_DIM"
      ("PRODUCT_DIM_KEY")
   ON DELETE NO ACTION
   ON UPDATE NO ACTION
   ENFORCED
   ENABLE QUERY OPTIMIZATION;

ALTER TABLE "DS      "."SALES_FACT"
   ADD CONSTRAINT "SQL071121141338950" FOREIGN KEY
      ("STORE_DIM_KEY")
   REFERENCES "DS      "."STORE_DIM"
      ("STORE_DIM_KEY")
   ON DELETE NO ACTION
   ON UPDATE NO ACTION
   ENFORCED
   ENABLE QUERY OPTIMIZATION;

ALTER TABLE "DS      "."SALES_FACT"
   ADD CONSTRAINT "SQL071121141338970" FOREIGN KEY
      ("CUSTOMER_DIM_KEY")
   REFERENCES "DS      "."CUSTOMER_DIM"
      ("CUSTOMER_DIM_KEY")
   ON DELETE NO ACTION
   ON UPDATE NO ACTION
   ENFORCED
   ENABLE QUERY OPTIMIZATION;

ALTER TABLE "DS      "."SALES_FACT"
   ADD CONSTRAINT "SQL071121141338980" FOREIGN KEY
      ("DATE_DIM_KEY")
   REFERENCES "DS      "."DATE_DIM"
      ("DATE_DIM_KEY")
   ON DELETE NO ACTION
   ON UPDATE NO ACTION
   ENFORCED
   ENABLE QUERY OPTIMIZATION;
```

```
COMMIT WORK;

CONNECT RESET;

TERMINATE;
```

Example: B-2 DDL statements for the interim tables for the sales transaction

```
------------------------------------------------
-- DDL Statements for table "DS      "."SALES_ST1"
------------------------------------------------
CREATE TABLE "DS      "."SALES_ST1"  (
         "SALES_ID" INTEGER NOT NULL ,
         "DATE" TIMESTAMP ,
         "QUANTITY" INTEGER ,
         "PRICE_USD" DECIMAL(10,2) ,
         "SELLING_PRICE_USD" DECIMAL(10,2) ,
         "TOTAL_USD" DECIMAL(10,2) ,
         "TOTAL_LOCAL_CURRENCY" DECIMAL(10,2) ,
         "CUSTOMER_ID" INTEGER ,
         "STORE_ID" INTEGER ,
         "PRODUCT_ID" INTEGER ,
         "COUNTRY_ISO_CODE" CHAR(3) )
        IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "DS      "."SALES_ST1"

ALTER TABLE "DS      "."SALES_ST1"
    ADD PRIMARY KEY
        ("SALES_ID");


------------------------------------------------
-- DDL Statements for table "DS      "."SALES_ST33"
------------------------------------------------
CREATE TABLE "DS      "."SALES_ST33"  (
         "SALES_ID" INTEGER NOT NULL ,
         "DATE" TIMESTAMP ,
         "QUANTITY" INTEGER ,
         "PRICE_USD" DECIMAL(10,2) ,
         "SELLING_PRICE_USD" DECIMAL(10,2) ,
         "TOTAL_USD" DECIMAL(10,2) ,
         "TOTAL_LOCAL_CURRENCY" DECIMAL(10,2) ,
         "CUSTOMER_ID" INTEGER ,
         "STORE_ID" INTEGER ,
         "PRODUCT_ID" INTEGER ,
         "COUNTRY_ISO_CODE" CHAR(3) )
        IN "USERSPACE1" ;
```

```
-- DDL Statements for primary key on Table "DS      "."SALES_ST33"

ALTER TABLE "DS      "."SALES_ST33"
    ADD PRIMARY KEY
        ("SALES_ID");



------------------------------------------------
-- DDL Statements for table "DS      "."SALES_ST9"
------------------------------------------------


CREATE TABLE "DS      "."SALES_ST9"  (
          "SALES_ID" INTEGER NOT NULL ,
          "DATE" TIMESTAMP ,
          "QUANTITY" INTEGER ,
          "PRICE_USD" DECIMAL(10,2) ,
          "SELLING_PRICE_USD" DECIMAL(10,2) ,
          "TOTAL_USD" DECIMAL(10,2) ,
          "TOTAL_LOCAL_CURRENCY" DECIMAL(10,2) ,
          "CUSTOMER_ID" INTEGER ,
          "STORE_ID" INTEGER ,
          "PRODUCT_ID" INTEGER ,
          "COUNTRY_ISO_CODE" CHAR(3) )
         IN "USERSPACE1" ;


-- DDL Statements for primary key on Table "DS      "."SALES_ST9"

ALTER TABLE "DS      "."SALES_ST9"
    ADD PRIMARY KEY
        ("SALES_ID");
```

C

Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

`ftp://www.redbooks.ibm.com/redbooks/SG247576`

Alternatively, you can go to the IBM Redbooks Web site at:

`ibm.com``/redbooks`

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247576.

Using the Web material

The additional Web material that accompanies this book includes the following files:

File name	*Description*
SG247576.zip	Zipped Code Samples

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

For information about ordering these publications, see "How to get Redbooks" on page 608. Note that some of the documents referenced here may be available in softcopy only.

- *SOA Solutions Using IBM Information Server,* SG24-7402
- *IBM WebSphere Information Analyzer Data Quality Assessment,* SG24-7508
- *IBM WebSphere QualityStage Methodologies, Standandarization, and Matching,* SG24-7546

Other publications

These publications are also relevant as further information sources:

- *IBM Information Server - Delivering information you can trust,* IBM United States Announcement 206-308 dated December 12, 2006
- *IBM Information Server Version 8.0.1 Planning, Installation, and Configuration Guide,* GC19-1048-01.
- *IBM Information Server Version 8.0.1 Information Server Introduction,* SC19-1049-01.
- *IBM Information Server Version 8.0.1 IBM Information Server Administration Guide,* SC19-9929-00.
- *IBM Information Server Version 8.0.1 Reporting Guide,* SC19-1162-01.
- *IBM Information Server — Delivers next generation data profiling analysis and monitoring through the new IBM WebSphere Information Analyzer module,* IBM United States Announcement 207-043 dated March 13th 2007.
- *IBM Information Management Software Profiling: Take the first step toward assuring data quality,* December 2006, IMW11808-USEN-00.
- *IBM WebSphere DataStage and QualityStage Version 8 Parallel Job Developer Guide,* SC18-9891-00.
- *IBM WebSphere DataStage and QualityStage Version 8.0.1 Parallel Job Advanced Developer Guide,* LC18-9892-01.

- *IBM WebSphere DataStage and QualityStage Version 8 Designer Client Guide,* SC18-9893-00.
- *IBM WebSphere DataStage and QualityStage Version 8 Director Client Guide,* SC18-9894-00.
- *IBM WebSphere DataStage and QualityStage Version 8 Administrator Client Guide,* SC18-9895-00.
- *IBM WebSphere DataStage and QualityStage Version 8 Basic Reference Guide,* SC18-9897-00.
- *IBM WebSphere DataStage and QualityStage Version 8 Server Job Developer Guide,* SC18-9898-00.
- *IBM WebSphere DataStage and QualityStage Version 8 Parallel Engine Message Reference,* LC18-9931-00.
- *IBM WebSphere DataStage and QualityStage Version 8 Connectivity Guide for DB2 Databases,* SC18-9932-00.

Online resources

These Web sites are also relevant as further information sources:

- IBM Information Server information center

  `http://publib.boulder.ibm.com/infocenter/iisinfsv/v8r0/index.jsp`

- *IBM Information Server Quick Start Guide*

  `http://www-1.ibm.com/support/docview.wss?uid=swg27009391&aid=1`

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

X

IBM InfoSphere DataStage
Data Flow and Job Design

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

IBM InfoSphere DataStage Data Flow and Job Design

IBM InfoSphere DataStage overview

Retail industry scenario

IBM Information Server setups

IBM Information Server is a revolutionary new software platform that helps organizations derive more value from the complex heterogeneous information spread across their systems. It enables organizations to integrate disparate data and deliver trusted information wherever and whenever needed, in line and in context, to specific people, applications, and processes.

IBM InfoSphere DataStage is a critical component of the IBM Information Server, and the parallel framework of IBM InfoSphere DataStage is also the foundation for IBM InfoSphere QualityStage and IBM InfoSphere Information Analyzer components.

This IBM Redbooks publication develops usage scenarios that describe the implementation of IBM InfoSphere DataStage flow and job design with special emphasis on the new features such as the distributed transaction stage (DTS) in Version 8.0.1, slowly changing dimensions stage (Version 8.0.1), complex flat file stage (Version 8.0.1), and access to mainframe data.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7576-00 ISBN 0738431117