Oracle® Fusion Middleware

Getting Started with Oracle Data Integrator

Release 11*g* (11.1.1)

E12641-01

September 2010

ORACLE®

Oracle Fusion Middleware Getting Started with Oracle Data Integrator, Release 11*g* (11.1.1)

E12641-01

Primary Author: Laura Hofman Miquel

Contents

10 Going Further with Oracle Data Integrator

Preface

This manual describes how to get started with Oracle Data Integrator. It provides general background information and detailed examples to help you learn how to use Oracle Data Integrator

This preface contains the following topics:.

Audience

This document is intended for users interested in learning how to use Oracle Data Integrator as a development tool for their integration processes.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Related Documents

For more information, see the following Oracle resources:

- *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*
- *Oracle Fusion Middleware Installation Guide for Oracle Data Integrator*
- *Oracle Fusion Middleware Upgrade Guide for Oracle Data Integrator*
- *Oracle Fusion Middleware Connectivity and Modules Guide for Oracle Data Integrator*
- *Oracle Fusion Middleware Knowledge Module Developer's Guide for Oracle Data Integrator*
- *Oracle Data Integrator 11g Online Help*
- *Oracle Data Integrator 11g Release Notes, included with your Oracle Data Integrator 11g installation, and on Oracle Technology Network*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Oracle Data Integrator Overview

This chapter provides an introduction to Oracle Data Integrator, the technical architecture, and the contents of this Getting Started guide.

This chapter includes the following sections:

1.1 Introduction to Oracle Data Integrator

A widely used data integration software product, Oracle Data Integrator provides a new declarative design approach to defining data transformation and integration processes, resulting in faster and simpler development and maintenance. Based on a unique *E-LT architecture (Extract - Load Transform)*, Oracle Data Integrator not only guarantees the highest level of performance possible for the execution of data transformation and validation processes but is also the most cost-effective solution available today.

Oracle Data Integrator provides a unified infrastructure to streamline data and application integration projects.

1.1.1 The Business Problem

In today's increasingly fast-paced business environment, organizations need to use more specialized software applications; they also need to ensure the coexistence of these applications on heterogeneous hardware platforms and systems and guarantee the ability to share data between applications and systems. Projects that implement these integration requirements need to be delivered on-spec, on-time and on-budget.

1.1.2 A Unique Solution

Oracle Data Integrator employs a powerful declarative design approach to data integration, which separates the declarative rules from the implementation details. Oracle Data Integrator is also based on a unique E-LT (Extract - Load Transform) architecture which eliminates the need for a standalone ETL server and proprietary engine, and instead leverages the inherent power of your RDBMS engines. This combination provides the greatest productivity for both development and maintenance, and the highest performance for the execution of data transformation and validation processes.

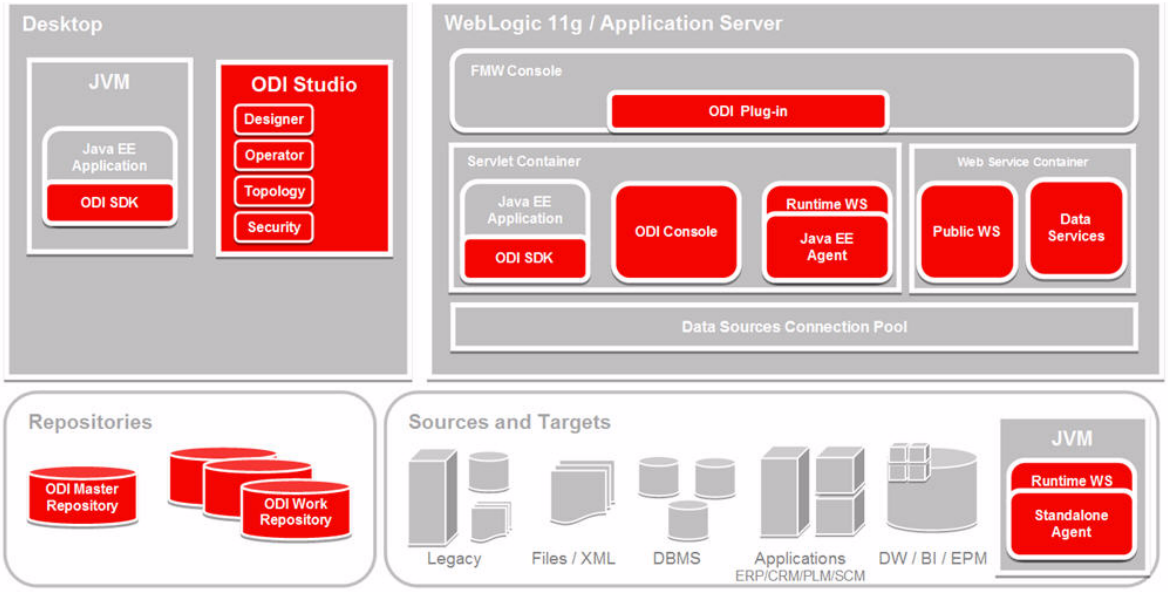
Here are the key reasons why companies choose Oracle Data Integrator for their data integration needs:

- **Faster and simpler development and maintenance**: The declarative rules driven approach to data integration greatly reduces the learning curve of the product and increases developer productivity while facilitating ongoing maintenance. This approach separates the definition of the processes from their actual implementation, and separates the declarative rules (the "what") from the data flows (the "how").
- **Data quality firewall**: Oracle Data Integrator ensures that faulty data is automatically detected and recycled before insertion in the target application. This is performed without the need for programming, following the data integrity rules and constraints defined both on the target application and in Oracle Data Integrator.
- **Better execution performance**: traditional data integration software (ETL) is based on proprietary engines that perform data transformations row by row, thus limiting performance. By implementing an E-LT architecture, based on your existing RDBMS engines and SQL, you are capable of executing data transformations on the target server at a set-based level, giving you much higher performance.
- **Simpler and more efficient architecture**: the E-LT architecture removes the need for an ETL Server sitting between the sources and the target server. It utilizes the source and target servers to perform complex transformations, most of which happen in batch mode when the server is not busy processing end-user queries.
- **Platform Independence**: Oracle Data Integrator supports all platforms, hardware and OSs with the same software.
- **Data Connectivity**: Oracle Data Integrator supports all RDBMSs including all leading Data Warehousing platforms such as Oracle, Exadata, Teradata, IBM DB2, Netezza, Sybase IQ and numerous other technologies such as flat files, ERPs, LDAP, XML.
- **Cost-savings**: the elimination of the ETL Server and ETL engine reduces both the initial hardware and software acquisition and maintenance costs. The reduced learning curve and increased developer productivity significantly reduce the overall labor costs of the project, as well as the cost of ongoing enhancements.

1.2 ODI Component Architecture

The Oracle Data Integrator platform integrates in the broader Fusion Middleware platform and becomes a key component of this stack. Oracle Data Integrator provides its run-time components as Java EE applications, enhanced to fully leverage the capabilities of the Oracle WebLogic Application Server. Oracle Data Integrator components include exclusive features for Enterprise-Scale Deployments, high availability, scalability, and hardened security. Figure 1–1 shows the ODI component architecture.

Figure 1–1 Oracle Data Integrator Component Architecture



1.2.1 Repositories

The central component of the architecture is the Oracle Data Integrator Repository. It stores configuration information about the IT infrastructure, metadata of all applications, projects, scenarios, and the execution logs. Many instances of the repository can coexist in the IT infrastructure, for example *Development*, *QA*, *User Acceptance*, and *Production*. The architecture of the repository is designed to allow several separated environments that exchange metadata and scenarios (for example: Development, Test, Maintenance and Production environments). The repository also acts as a version control system where objects are archived and assigned a version number.

The Oracle Data Integrator Repository is composed of one *Master Repository* and several *Work Repositories*. Objects developed or configured through the user interfaces are stored in one of these repository types.

There is usually only one master repository that stores the following information:

- Security information including users, profiles and rights for the ODI platform
- Topology information including technologies, server definitions, schemas, contexts, languages and so forth.
- Versioned and archived objects.

The work repository is the one that contains actual developed objects. Several work repositories may coexist in the same ODI installation (for example, to have separate environments or to match a particular versioning life cycle). A Work Repository stores information for:

- Models, including schema definition, datastores structures and metadata, fields and columns definitions, data quality constraints, cross references, data lineage and so forth.
- Projects, including business rules, packages, procedures, folders, Knowledge Modules, variables and so forth.

- Scenario execution, including scenarios, scheduling information and logs.

When the Work Repository contains only the execution information (typically for production purposes), it is then called an Execution Repository.

1.2.2 ODI Studio and User Interfaces

Administrators, Developers and Operators use the Oracle Data Integrator Studio to access the repositories. This Fusion Client Platform (FCP) based UI is used for administering the infrastructure (security and topology), reverse-engineering the metadata, developing projects, scheduling, operating and monitoring executions.

ODI Studio provides four Navigators for managing the different aspects and steps of an ODI integration project:

- *Designer Navigator* is used to design data integrity checks and to build transformations such as for example:
  - Automatic reverse-engineering of existing applications or databases
  - Graphical development and maintenance of transformation and integration interfaces
  - Visualization of data flows in the interfaces
  - Automatic documentation generation
  - Customization of the generated code
- *Operator Navigator* is the production management and monitoring tool. It is designed for IT production operators. Through Operator Navigator, you can manage your interface executions in the sessions, as well as the scenarios in production.
- *Topology Navigator* is used to manage the data describing the information system's physical and logical architecture. Through Topology Navigator you can manage the topology of your information system, the technologies and their datatypes, the data servers linked to these technologies and the schemas they contain, the contexts, the languages and the agents, as well as the repositories. The site, machine, and data server descriptions will enable Oracle Data Integrator to execute the same integration interfaces in different physical environments.
- *Security Navigator* is the tool for managing the security information in Oracle Data Integrator. Through Security Navigator you can create users and profiles and assign user rights for methods (edit, delete, etc) on generic objects (data server, datatypes, etc), and fine-tune these rights on the object instances (Server 1, Server 2, and so forth).

Oracle Data Integrator also provides a Java API for performing all these run-time and design-time operations. This *Oracle Data Integrator Software Development Kit (SDK)* is available for standalone Java applications and application servers.

1.2.3 Run-Time Agent

At design time, developers generate scenarios from the business rules that they have designed. The code of these scenarios is then retrieved from the repository by the Run-Time Agent. This agent then connects to the data servers and orchestrates the code execution on these servers. It retrieves the return codes and messages for the execution, as well as additional logging information – such as the number of processed records, execution time and so forth - in the Repository.

The Agent comes in two different flavors:

- The Java EE Agent can be deployed as a web application and benefit from the features of an application server.
- The Standalone Agent runs in a simple Java Machine and can be deployed where needed to perform the integration flows.

Both these agents are multi-threaded java programs that support load balancing and can be distributed across the information system. This agent holds its own execution schedule which can be defined in Oracle Data Integrator, and can also be called from an external scheduler. It can also be invoked from a Java API or a web service interface.

1.2.4 Oracle Data Integrator Console

Business users (as well as developers, administrators and operators), can have read access to the repository, perform topology configuration and production operations through a web based UI called *Oracle Data Integrator Console*. This web application can deployed in a Java EE application server such as Oracle WebLogic.

To manage and monitor the Java EE and Standalone Agents as well as the ODI Console, Oracle Data Integrator provides a new plug-in that integrates in Oracle Fusion Middleware Control Console.

1.3 Get Started with Oracle Data Integrator

Table 1–1 summarizes the contents of this guide.

Table 1–1 Content Summary

This chapter	Describes how to...
Chapter 2, "Installing Oracle Data Integrator and the Demonstration Environment"	Install Oracle Data Integrator and the demonstration environment
Chapter 3, "Working with the ETL Project"	Provides an introduction to the demonstration environment delivered with Oracle Data Integrator Studio
Chapter 4, "Starting Oracle Data Integrator"	Start the demonstration environment and Oracle Data Integrator Studio
Chapter 5, "Implementing Data Quality Control"	Implement data quality control
Chapter 6, "Working with Integration Interfaces"	Create and work with integration interfaces in Oracle Data Integrator
Chapter 7, "Working with Packages"	Create and work with Packages in Oracle Data Integrator
Chapter 8, "Executing Your Developments and Reviewing the Results"	Execute your developments, follow the execution, and interpret the execution results
Chapter 9, "Deploying Integrated Applications"	Run an ODI Package automatically in a production environment
Chapter 10, "Going Further with Oracle Data Integrator"	Perform advanced tasks with Oracle Data Integrator

2

Installing Oracle Data Integrator and the Demonstration Environment

This chapter provides an overview of how to install Oracle Data Integrator and the demonstration environment. The instructions in this chapter are the instructions required for using the demonstration environment with Oracle Data Integrator Studio.

This chapter includes the following sections:

- Section 2.1, "Preparing to Install"
- Section 2.2, "Installing Oracle Data Integrator"
- Section 2.3, "Installing the Demonstration Environment"

> **Note:** Oracle Data Integrator Studio and the demonstration environment must be installed on your system to perform the tasks described in this Getting Started guide.

2.1 Preparing to Install

Review the information in this section before you begin:

- Review System Requirements and Certification
- Understand Oracle Fusion Middleware Support of 64-bit JDK
- Create ODI Repositories

2.1.1 Review System Requirements and Certification

Before installing any Oracle Data Integrator (ODI) components, you should read the system requirements and certification documentation to ensure that your environment meets the minimum installation requirements. Both of these documents are available on Oracle Technology Network (OTN).

The system requirements document covers information such as hardware and software requirements, minimum disk space and memory requirements, and required system libraries, packages, or patches:

`http://www.oracle.com/technology/software/products/ias/files/fusion_requirements.htm`

The certification document covers supported installation types, platforms, operating systems, databases, JDKs, and third-party products:

`http://www.oracle.com/technology/software/products/ias/files/fusion_`

certification.html

> **Note:** If you are installing the 32-bit version of the product, the system on which you are installing must also be a supported 32-bit system. Installing a 32-bit version of the product on a 64-bit system is not supported.

2.1.2 Understand Oracle Fusion Middleware Support of 64-bit JDK

If you are using a 64-bit Java Virtual Machine (JVM) in your environment, ensure that all your Oracle Fusion Middleware components are using the 64-bit JVM. You cannot mix components using a 32-bit JVM with those using a 64-bit JVM.

Refer to the Oracle Fusion Middleware Certifications matrix for information on the platforms that support a 64-bit JDK:

http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html

2.1.3 Create ODI Repositories

You do not need to create ODI repositories for using the demonstration environment in ODI Studio. The demonstration environment provides the necessary preconfigured repositories.

2.2 Installing Oracle Data Integrator

This section contains information and instructions for installing Oracle Data Integrator Studio. The installation instructions in this chapter are the instructions required for using the demonstration environment in Oracle Data Integrator Studio.

For detailed installation instructions, see the *Oracle Fusion Middleware Installation Guide for Oracle Data Integrator*.

This section contains the following topics:

- Downloading the Installer and the Demo Environment
- Starting the Installer
- Installation Types
- Installation Instructions

> **Note:** If you are installing on a UNIX system for the first time, you may be asked to run the *ORACLE_HOME*/oracleRoot.sh script as root user to create all of the necessary installation directories.

2.2.1 Downloading the Installer and the Demo Environment

The installer and the demo environment are available on the Oracle Data Integrator Downloads page on Oracle Technology Network (OTN).

To download the installer and the demo environment:

1. Go to the Oracle Data Integrator Downloads page on OTN at http://otn.oracle.com/goto/odi

2. Download the 11*g* Oracle Data Integrator Media Pack and the 11*g* demo environment for your platform to a temporary folder.

2.2.2 Starting the Installer

To start the installer, uncompress the Oracle Data Integrator Media Pack you have downloaded in a temporary folder and run the following command from this folder:

- On UNIX operating systems:

```
./runInstaller
```

- On Windows operating systems:

```
setup.exe
```

> **Note:** The minimum JDK required for Oracle Data Integrator is JDK 1.6. Refer to the Oracle Fusion Middleware Certification documentation to see the JDKs supported for your system:
>
> http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html

2.2.3 Installation Types

You need to install the following components to work with the demo environment:

- ODI Studio
- Standalone Agent

This corresponds to the Developer and the Standalone installation type.

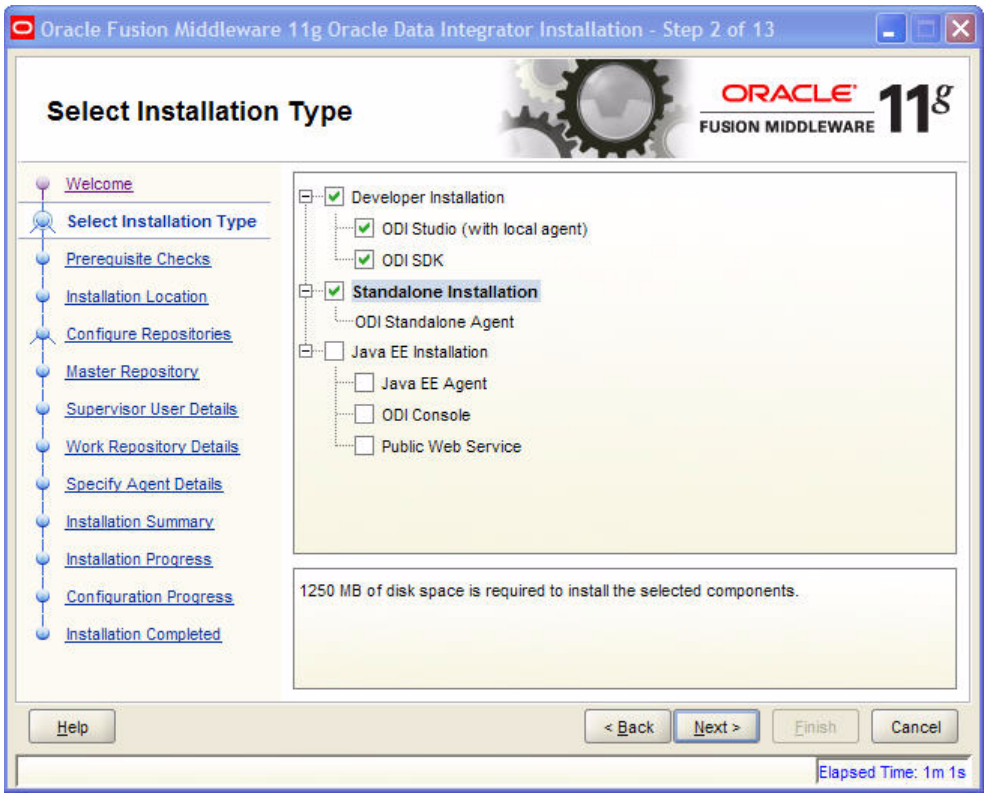
2.2.4 Installation Instructions

Follow these instructions to install and configure Oracle Data Integrator Studio:

> **Note:** If you need additional help with any of the installation screens, refer to the *Oracle Fusion Middleware Installation Guide for Oracle Data Integrator* or click **Help** to access the online help.

1. In the Welcome Screen, click **Next** to continue.
2. In the Select Installation Type Screen, select both **Developer Installation** and **Standalone Installation** as shown in Figure 2–1. Note that ODI Studio and ODI SDK are automatically selected.

Figure 2–1 Select Installation Type Screen



3. Click **Next** to continue.
4. In the Prerequisite Checks Screen, click **Next** to continue.
5. In the Specify Installation Location Screen, enter the absolute path for the Oracle home location (referred to later in this guide as `ODI_HOME`). For example:

   ```
   C:\oracle\ODI_HOME1
   ```

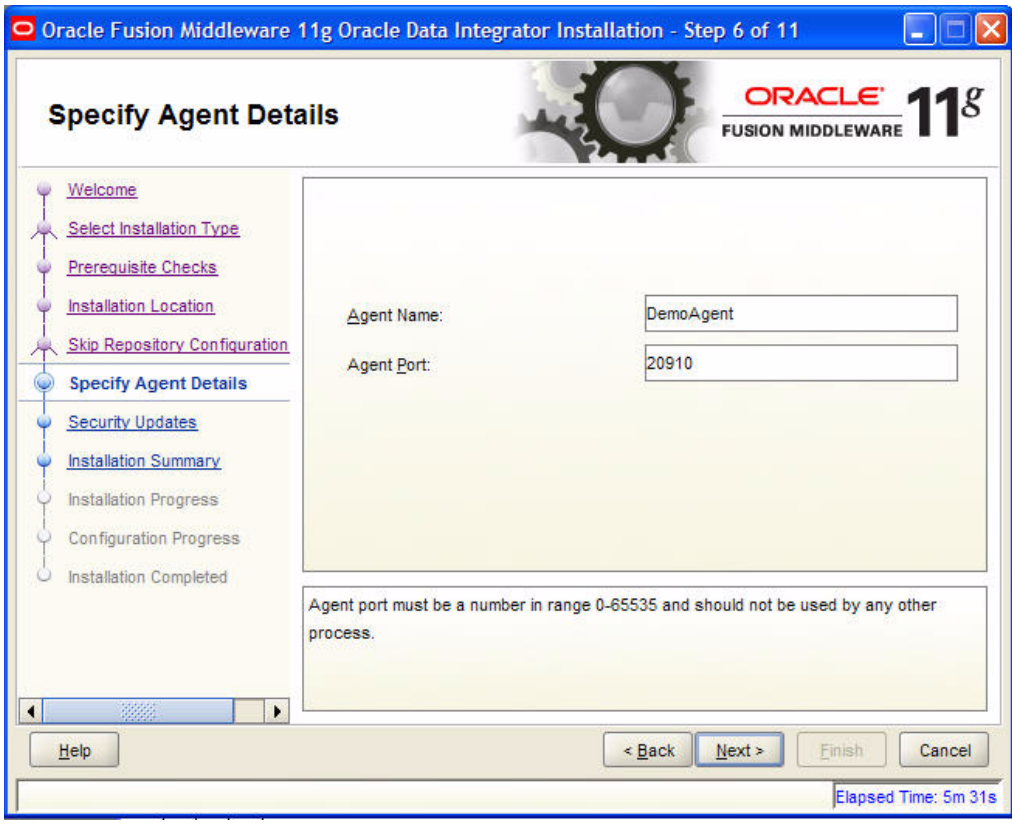
   > **Note:** The specified directory must be an empty directory or an existing Oracle Data Integrator home location.

6. Click **Next** to continue.
7. In the Repository Configuration Screen, select **Skip Repository Configuration**.
8. Click **Next** to continue.
9. In the Specify Agent Details Screen, enter the following agent details:
   - Agent Name: `DemoAgent`
   - Agent Port: `20910`

     Note that the port number should not be used by any other process and that the agent name cannot be the same as another agent already declared in the topology.

   The Specify Agent Details screen is shown in Figure 2–2.

Figure 2–2 Specify Agent Details Screen



10. In the Specify Security Updates Screen, choose how you want to be notified about security issues:
    - If you want to be notified about security issues through E-mail, enter your E-mail address in the E-mail field.
    - If you want to be notified about security issues through My Oracle Support (formerly MetaLink), select the My Oracle Support option and enter your My Oracle Support Password.
    - If you do not want to be notified about security issues, leave all fields empty. You will see the following message: "My Oracle Support Username/E-mail address not specified". Click **Yes** to continue.
11. In the Specify Security Updates Screen, click **Next** to continue.
12. In the Installation Summary Screen, verify the information.
13. Click **Install** to begin the installation.
14. In the Installation Progress Screen, click **Next** to continue.
15. In the Configuration Progress Screen, click **Next** to continue.
16. In the Installation Completed Screen, click **Finish** to dismiss the installer.

Oracle Data Integrator is now installed.

2.3 Installing the Demonstration Environment

The Oracle Data Integrator demonstration environment is delivered on the Oracle Data Integrator companion CD and can be downloaded from the Oracle Data Integrator Downloads page on Oracle Technology Network (OTN) at

http://otn.oracle.com/goto/odi

The demonstration files and samples of the demonstration environment are located in the /demo folder of the companion CD.

> **Note:** The demonstration (demo) environment should be installed with an existing installation that includes the ODI Studio component.

To manually install the Demonstration environment, do the following:

1. Unzip oracledi-demo.zip in the ODI_HOME folder.
2. Verify that the JAVA_HOME environment variable is set and contains the path of a JVM suitable for Oracle Data Integrator.

   If this variable is not set correctly, set it to a valid java machine location.

   For example:

   On UNIX operating systems:

   ```
   setenv JAVA_HOME/usr/local/java
   ```

   On Windows operating systems:

   Set the JAVA_HOME variable graphically

   For a list of certified JVM versions, see http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html.

You can now use the demonstration environment in Oracle Data Integrator Studio.

3

Working with the ETL Project

This chapter provides an introduction to the *ETL* (Extract Transform Load) project that is delivered in the demonstration environment with Oracle Data Integrator Studio.

This chapter includes the following sections:

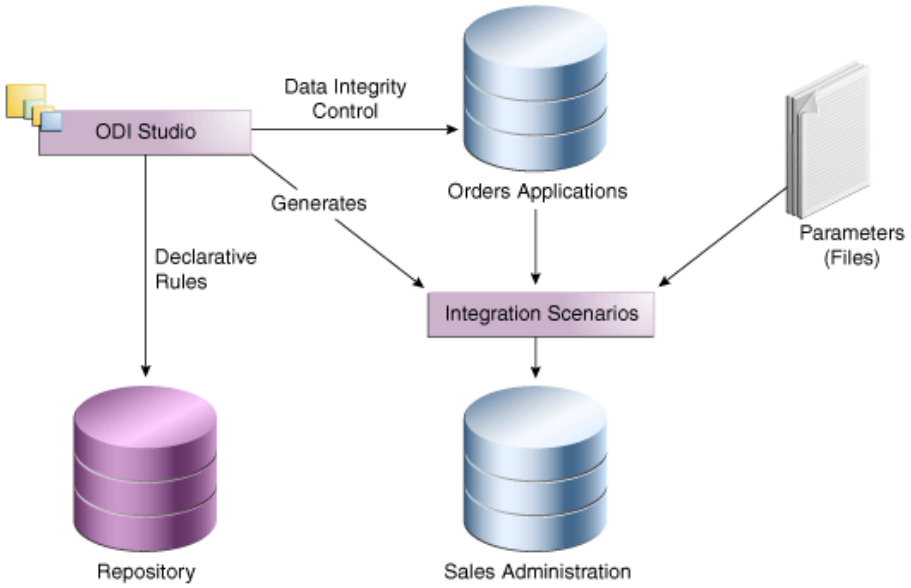
- Section 3.1, "The Example Environment"
- Section 3.2, "The Data Models"
- Section 3.3, "Integration Challenges"

3.1 The Example Environment

The *ETL* project is an example to help you understand how to transform and check the integrity of the data in your information systems.

The examples in this getting started guide track sales from various heterogeneous data sources issued from the production systems. Figure 3–1 shows the example environment.

Figure 3–1 Example Environment



The example environment uses the following elements:

- *The Repository:* The Repository contains all of the metadata required for the training examples. It is hosted, for these evaluation purposes, in a supplied database.
- *Orders Application:* An application for tracking customer orders, hosted in a supplied database (the "srcdemo" sample database).
- *Parameters (File):* Flat files (ASCII) issued from the production system containing a list of sales representatives and the segmentation of ages into age ranges.
- *Sales Administration:* The administration or tracking of sales, hosted in another supplied database (the "trgdemo" sample database). This data warehouse is populated with our transformations.

3.2 The Data Models

The demonstration environment includes three ODI data models:

- Orders Application - HSQL
- Parameters - FILE
- Sales Administration - HSQL

This section provides the schema diagrams for these data models.

3.2.1 Orders Application - HSQL

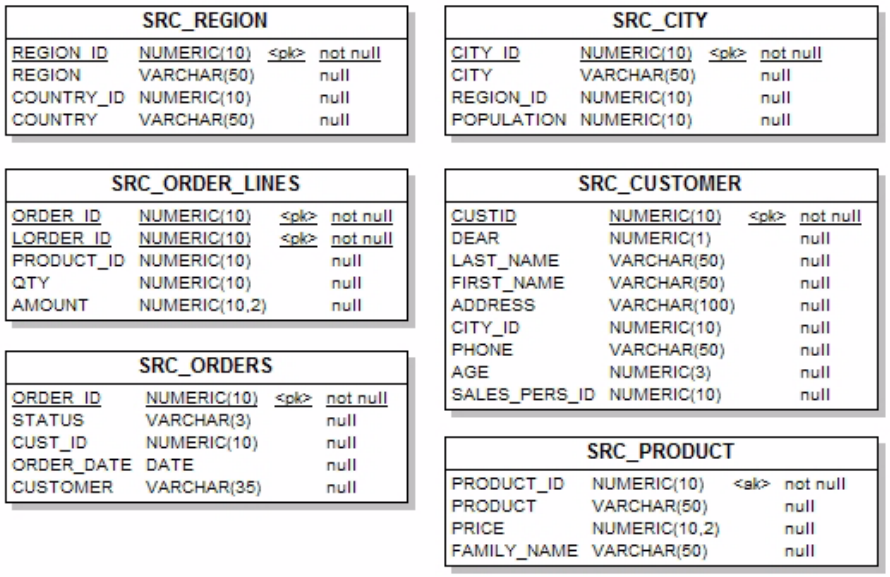
The *Orders Application* data model is based on the HSQL technology and includes five datastores:

- SRC_CITY
- SRC_CUSTOMER
- SRC_ORDERS
- SRC_ORDER_LINES
- SRC_PRODUCT
- SRC_REGION

Figure 3–2 shows the schema diagram of this data model.

Note that this data model does not enforce any foreign key constraints, even if some functional relations exist between the data.

Figure 3–2 Orders Application Schema Diagram



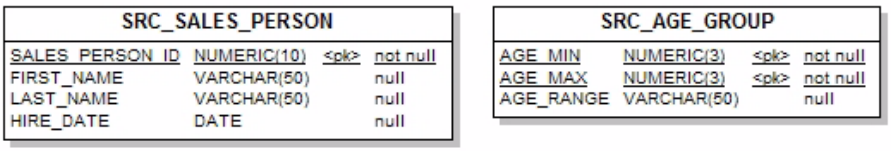
3.2.2 Parameters - FILE

The *Parameters* data model is based on the File technology and includes two datastores:

- SRC_SALES _PERSON
- SRC_AGE_GROUP

Figure 3–3 shows the schema diagram of this data model.

Figure 3–3 Parameters Schema Diagram



3.2.3 Sales Administration - HSQL

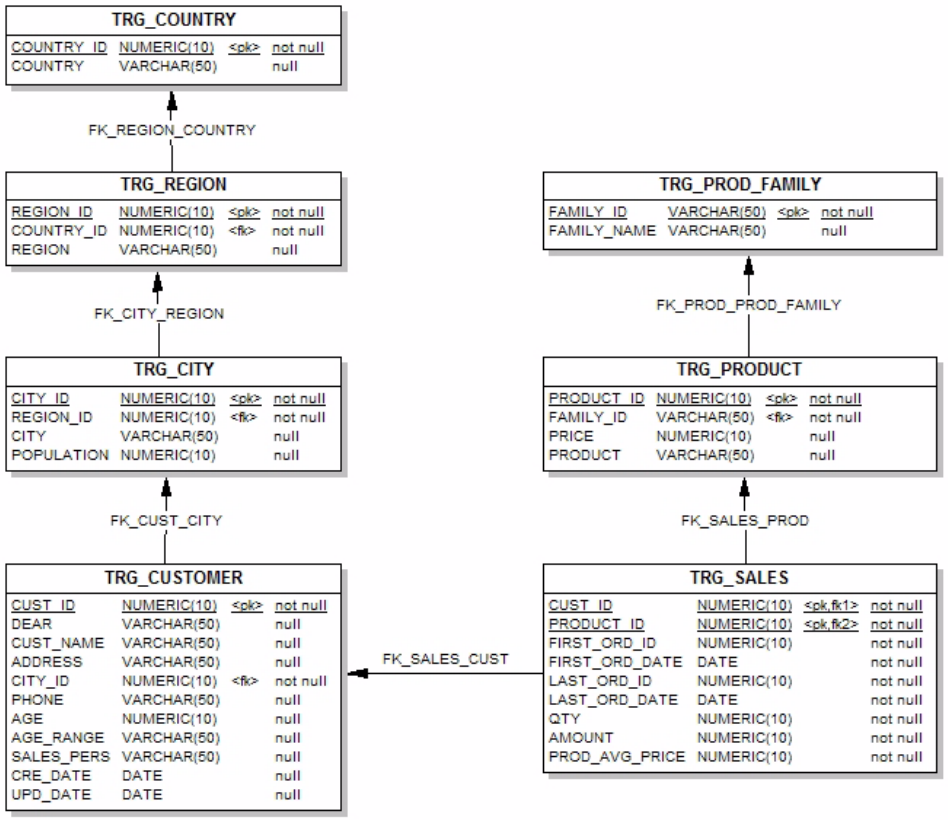
The *Sales Administration* data model is based on the HSQL technology and includes six datastores:

- TRG_CITY
- TRG_COUNTRY
- TRG_CUSTOMER
- TRG_PRODUCT
- TRG_PROD_FAMILY

- TRG_REGION
- TRG_SALES

Figure 3–4 shows the schema diagram of this data model.

Figure 3–4 Sales Administration Schema Diagram



3.3 Integration Challenges

The challenges common to all data integration and transformation projects are:

- Checking and improving the quality of your application data
- Accurately and easily exchanging data between your applications while respecting the business rules of your information system

The examples used in this guide illustrate how to address these issues. During this getting started guide, you will learn how to:

- **Implement Data Quality Control to check data in a database**

  By implementing two examples, you will learn how Oracle Data Integrator enables you to ensure the quality of the data in your applications while segregating invalid rows. The *Orders Application* tables contain a number of data inconsistencies that you will detect.

- **Create integration interfaces to move and transform data**

Two simple examples will show you how to improve productivity by loading the data from the *Orders Application* and *Parameters (File)* applications into the *Sales Administration* data warehouse.

- **Automate the execution of these interfaces into packages**

  This part of the Getting Started guide will show you how to automate your Oracle Data Integrator processes. The aim of this exercise is to load the entire *Sales Administration* data warehouse with a single click.

- **Execute the package and review the execution results**

  You will learn how to execute the Load Sales Administration Package and the integration interfaces Pop. TRG_CUSTOMER and Pop. TRG_SALES you have created and how to review the results of these executions.

- **Prepare the developed components for deployment**

  You will learn how to run the Load Sales Administration Package automatically in a production environment.

Note: In this guide, we will be looking at processes that focus on ETL. While it is beyond the scope of this document, implementing different integration patterns (real-time, for example) can be carried out in the same fashion. For more information on this, see the Oracle Data Integrator documentation after completing this guide.

Now that you have been introduced to the concepts of the ETL-Project and its components, you can move on to Starting Oracle Data Integrator.

4

Starting Oracle Data Integrator

This chapter describes how to start the demonstration environment and the first steps in Oracle Data Integrator Studio.

This chapter includes the following sections:

4.1 Starting the Demonstration Environment

Oracle Data Integrator demonstration environment provides the initial repository and the databases that contain the data used in the examples of this Getting Started guide.

To start the demonstration environment:

1. Change directory to the `ODI_HOME/oracledi/demo/bin` directory.
2. Enter the following command:
   - On UNIX operating systems:

     `./startdemo.sh`
   - On Windows operating systems:

     `startdemo.bat`

The source and target data servers as well as the demo repository data server are started.

To stop these database servers, you can use the *stopdemo* command in the same directory:

Enter the following command:

- On UNIX operating systems:

  `./stopdemo.sh`
- On Windows operating systems:

  `stopdemo.bat`

Caution: Do not shut down the databases by using the [CTRL-C] keyboard shortcut, or by closing their execution windows. This may leave the source, target, and repository databases in an unstable state. Always use the *stopdemo* scripts.

4.2 Starting Oracle Data Integrator Studio

This section describes how to start Oracle Data Integrator Studio.

Note: Before starting ODI Studio, the demonstration environment must be started as described in Section 4.1, "Starting the Demonstration Environment".

4.2.1 Starting ODI Studio

In the demonstration environment, the connection to the demonstration repository is already defined. To perform the tasks of this getting started guide, you only need to login as SUPERVISOR to start Oracle Data Integrator Studio.

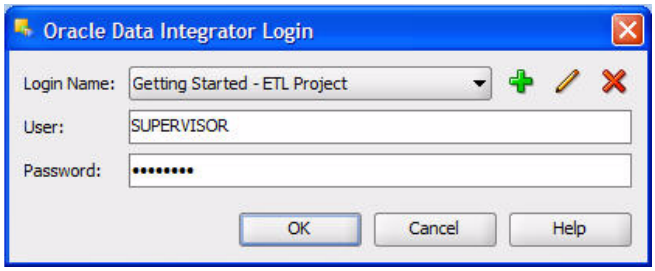
To connect to the demonstration repository and launch ODI Studio:

1. Select **Start Menu** > **All Programs** > **Oracle** > **Oracle Data Integrator** > **ODI Studio**.
2. In Designer Navigator, click **Connect To Repository...**

   The Oracle Data Integrator Login Dialog is displayed
3. Select **Getting Started - ETL Project** from the Login Name dropdown menu.

   Note: If this Login Name does not exist, go to Section 4.2.2, "Defining a new Getting Started Login".
4. In the User field, enter `SUPERVISOR`. Note that the username is case sensitive.
5. In the Password field, enter `SUNOPSIS`. Note that the password is case sensitive
6. The Oracle Data Login Dialog should look as shown in Figure 4–1.

Figure 4–1 Getting Started ODI Login



7. Click **OK**.

ODI Studio connects to the demonstration environment repository. You can now work in the Oracle Data Integrator demonstration environment.

4.2.2 Defining a new Getting Started Login

This section describes how to define a new connection to the demonstration repository and how to create a new ODI login.

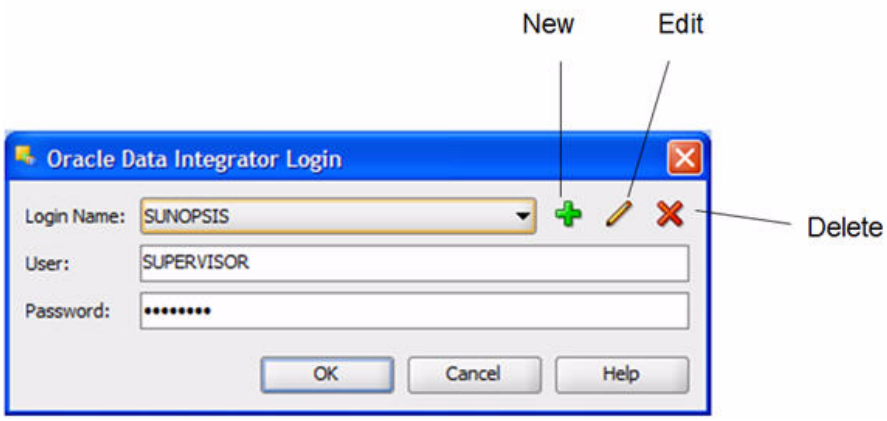
You only need to perform the tasks described in this section, if the Getting Started - ETL Project Login Name does not appear in the Login Name list of the Oracle Data Integrator Login Dialog shown in Figure 4–1.

To define a new connection to the demonstration repository:

1. In the Oracle Data Integrator Login Dialog, click **New**.

   Figure 4–2 shows the Oracle Data Integrator Login Dialog.

Figure 4–2 Default ODI Login



   The Repository Connection Information Dialog is displayed.

2. Click **New**.

   The Repository Connection Information Dialog is displayed.

3. Specify the Oracle Data Integrator connection details as follows:
   - Login name: `Getting Started - ETL Project`
   - User: `SUPERVISOR`
   - Password: `SUNOPSIS`

   Note that username and password are case sensitive.

4. Specify the Database Connection (Master Repository) details as follows:
   - User: `sa`

     This is the database user ID/login of the schema (database, library) that contains the ODI master repository
   - Password: Leave this field empty.
   - Driver List: Select *Hypersonic SQL Driver* from the list.

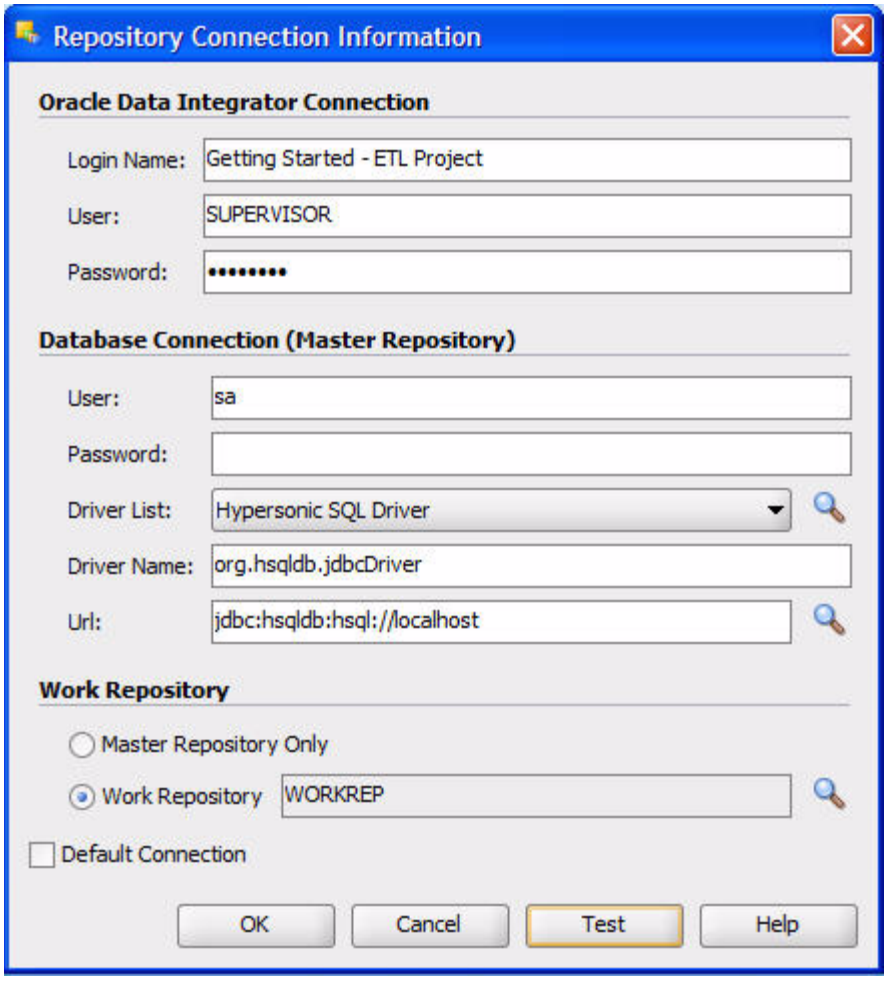
     This driver is required to connect to the DBMS supporting the master repository.
   - Driver Name: `org.hsqldb.jdbcDriver`
   - URL: `jdbc:hsqldb:hsql://localhost`

This URL is used to establish the JDBC connection to the database hosting the repository. Note that the driver name is case sensitive and make sure that the URL does not contain any extra characters, in particular spaces.

5. Select **Work Repository** and enter `WORKREP` in the Work Repository field.

   The Repository Connection Information Dialog should look as shown in Figure 4–3.

Figure 4–3 Repository Connection Information Dialog



6. Click **Test** to verify that the connection is working.

   The Information dialog opens and informs you if the connection has been established. If the connection fails, fix the connection parameters to your repository and make sure that the startdemo script is running (see Section 4.1, "Starting the Demonstration Environment") before moving to next step.

7. Click **OK** to close the Information dialog.

8. In the Oracle Data Integrator Login Dialog, select **Getting Started - ETL Project** from the Login Name dropdown menu.

   The SUPERVISOR user and the SUNOPSIS password are automatically set.

   The Oracle Data Login Dialog should look as shown in Figure 4–1.

9. In the Oracle Data Integrator Login Dialog click **OK**.

ODI Studio connects to the demonstration environment repository. You can now work in the Oracle Data Integrator demonstration environment

4.3 Introduction to Using the ODI Navigators

ODI Studio provides four Navigators for managing the different aspects and steps of an ODI integration project:

- Designer Navigator
- Operator Navigator
- Topology Navigator
- Security Navigator

The tasks performed in this getting started guide take place in Designer Navigator (to create and execute your developments) and in Operator Navigator (to monitor the execution of your developments). This section only describes the Navigators that are used in this getting started guide. See the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for information about the Topology and Security Navigators.

4.3.1 Designer Navigator

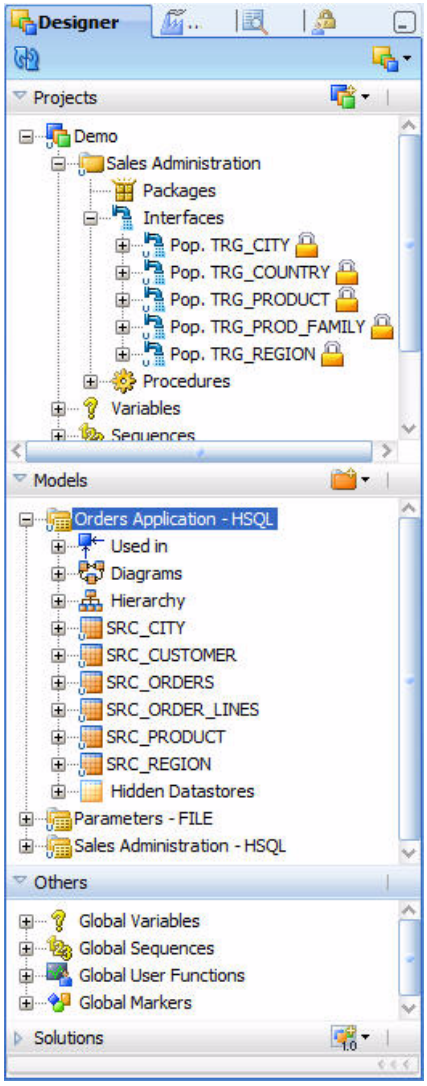
Designer Navigator is used to manage metadata, to design data integrity checks, and to build transformations.

The main objects you handle through Designer Navigator are *models* and *projects*.

- The data models for your applications contain all of the metadata in your data servers (tables, columns, constraints, descriptions, cross-references, etc.)
- The projects contain all of the loading and transformation rules for your data servers (interfaces, procedures, variables, etc.)

The Designer Navigator appears as shown in Figure 4–4.

Figure 4–4 Designer Navigator



The Designer Navigator has the following accordions:

- **Projects**

  The Projects accordion contains the developments made with Designer Navigator.

- **Models**

  The Models accordion contains the descriptions of the data and applications structures.

- **Others**

  The Others accordion contains the Global User Functions, Variables, Markers, and Sequences.

- **Solutions**

  The Solutions accordion contains the Solutions that have been created when working with version management.

The demonstration repository provides the objects you will need in this Getting Started guide:

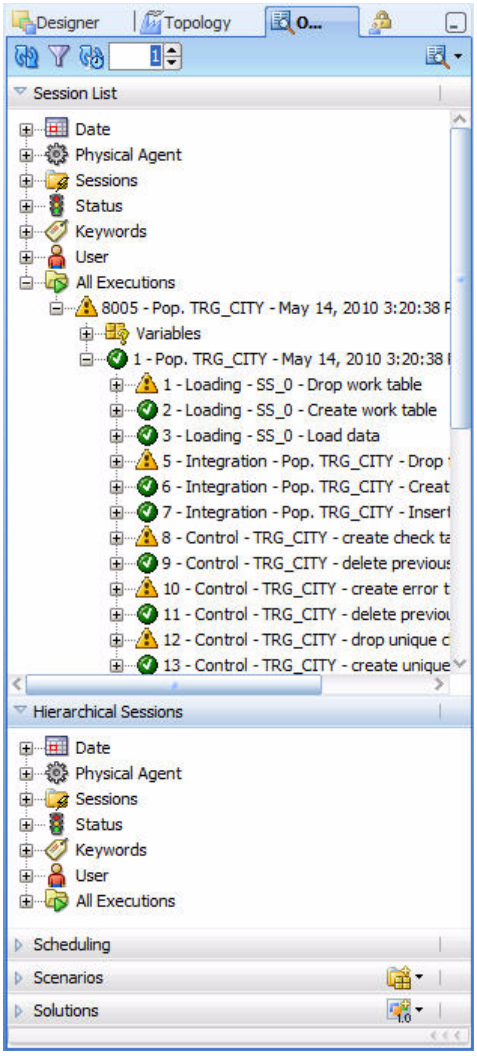
- In the Models accordion, you will find all the data models corresponding to the *Orders Application*, *Parameters*, and *Sales Administration* applications.
- In the Projects accordion, you will find the *Demo* project and the *Sales Administration* folder which already contains several interfaces. You will develop your new interfaces in this folder.
- The necessary Knowledge Modules (KM) are already imported in the Demo Project:
  - LKM File to SQL
  - LKM SQL to SQL
  - CKM HSQL
  - IKM SQL Incremental Update

4.3.2 Operator Navigator

Operator Navigator is the production management and monitoring tool. It is designed for IT production operators and can be used by developers to check code execution and perform debugging operations. Through Operator Navigator, you can manage your development executions in the sessions, as well as the scenarios in production.

The Operator Navigator appears as shown in Figure 4–5.

Figure 4–5 Operator Navigator



The Operator Navigator has the following accordions:

- **Session List**

  The Session List accordion displays all sessions organized per date, physical agent, status, keywords, and so forth.

- **Hierarchical Sessions**

  The Hierarchical Sessions accordion displays the execution sessions organized in a hierarchy with their child sessions.

- **Scheduling**

  The Scheduling accordion displays the list of physical agents and schedules.

- **Scenarios**

  The Scenarios accordion displays the list of scenarios available

- **Solutions**

  The Solutions accordion contains the Solutions that have been created when working with version management.

Now that the demonstration environment and Oracle Data Integrator are started, you can move on to Implementing Data Quality Control.

5

Implementing Data Quality Control

This chapter describes how to implement data quality control. An introduction to data integrity control is provided.

This chapter includes the following sections:

5.1 Introduction to Data Integrity Control

Data integrity control is essential in ensuring the overall consistency of the data in your information system's applications.

Application data is not always valid for the constraints and declarative rules imposed by the information system. You may, for instance, find orders with no customer, or order lines with no product, and so forth.

Oracle Data Integrator provides a working environment to detect these constraint violations and to store them for recycling or reporting purposes.

There are two different types of controls: *Static Control* and *Flow Control*. We will examine the differences between the two.

Static Control

Static Control implies the existence of rules that are used to verify the integrity of your application data. Some of these rules (referred to as constraints) may already be implemented in your data servers (using primary keys, reference constraints, etc.)

With Oracle Data Integrator, you can enhance the quality of your data by defining and checking additional constraints, without declaring them directly in your servers. This procedure is called **Static Control** since it allows you to perform checks directly on existing - or static - data.

Flow Control

The information systems targeted by transformation and integration processes often implement their own declarative rules. The **Flow Control** function is used to verify an application's incoming data according to these constraints before loading the data into these targets. The flow control procedure is detailed in the "Interfaces" chapter.

Benefits

The main advantages of performing data integrity checks are the following:

- *Increased productivity* by using the target database for its entire life cycle. Business rule violations in the data slow down application programming throughout the target database's life-cycle. Cleaning the transferred data can therefore reduce application programming time.
- *Validation of the target database's model*. The rule violations detected do not always imply insufficient source data integrity. They may reveal a degree of incompleteness in the target model. Migrating the data before an application is rewritten makes it possible to validate a new data model while providing a test database in line with reality.
- *Improved quality of service* for the end-users.

Ensuring data integrity is not always a simple task. Indeed, it requires that any data violating declarative rules must be isolated and recycled. This implies the development of complex programming, in particular when the target database incorporates a mechanism for verifying integrity constraints. In terms of operational constraints, it is most efficient to implement a method for correcting erroneous data (on the source, target, or recycled flows) and then to reuse this method throughout the enterprise.

5.2 SRC_CUSTOMER Control Example

This example guides you through the data integrity audit process (Static Control).

The *Orders Application - HSQL* application contains data that do not satisfy business rule constraints on a number of different levels. The objective is to determine which data in this application does not satisfy the constraints imposed by the information system.

This section includes the following topics:

5.2.1 Objective

Some data in our source may be inconsistent. There may be constraints in the target table that are not implemented in the source table or there may be supplementary rules that you wish to add. In our case we have two constraints that we want to enforce on the SRC_CUSTOMER table:

- **Customers must be over 21 years of age.** However there could be some records corresponding to younger customers in the input table.
- **The CITY_ID column must refer to an entry in the SRC_CITY table.** However there could be some values that do not exist in the city table.

We want to determine which rows do not satisfy these two constraints and automatically copy the corresponding invalid records into an error table for analysis.

5.2.2 Interpreting the Problem

Enforcing these types of rules requires the use of a *check constraint* (also referred to as a *condition*), as well as a *reference constraint* between the SRC_CITY and SRC_CUSTOMER tables.

5.2.3 Creating Constraints

This section describes how to create the following constraints:

- Age Constraint
- Reference Constraint

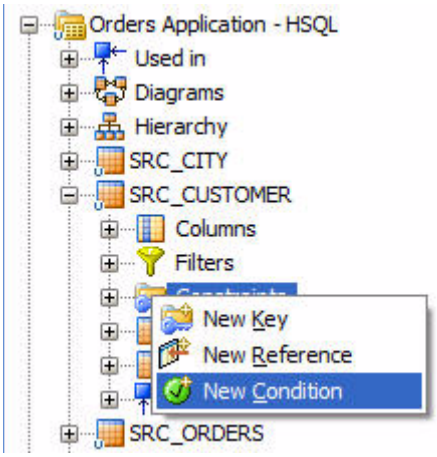
5.2.3.1 Age Constraint

Creating an age constraints consists in adding a data validity condition on a column.

To create the age constraint:

1. In the Models accordion in Designer Navigator, expand the *Orders Application - HSQL* model.
2. Expand the SRC_CUSTOMER datastore.
3. Right-click the Constraints node and select **New Condition** as shown in Figure 5–1.

Figure 5–1 Insert New Condition



4. In the Definition tab of the Condition Editor:
   - In the Name field, enter the name of your condition. For example: `AGE > 21`.
   - From the Type list, select **Oracle Data Integrator Condition**.
   - In the Where clause field, enter the following SQL code:

     ```
     SRC_CUSTOMER.AGE > 21
     ```

Notes:

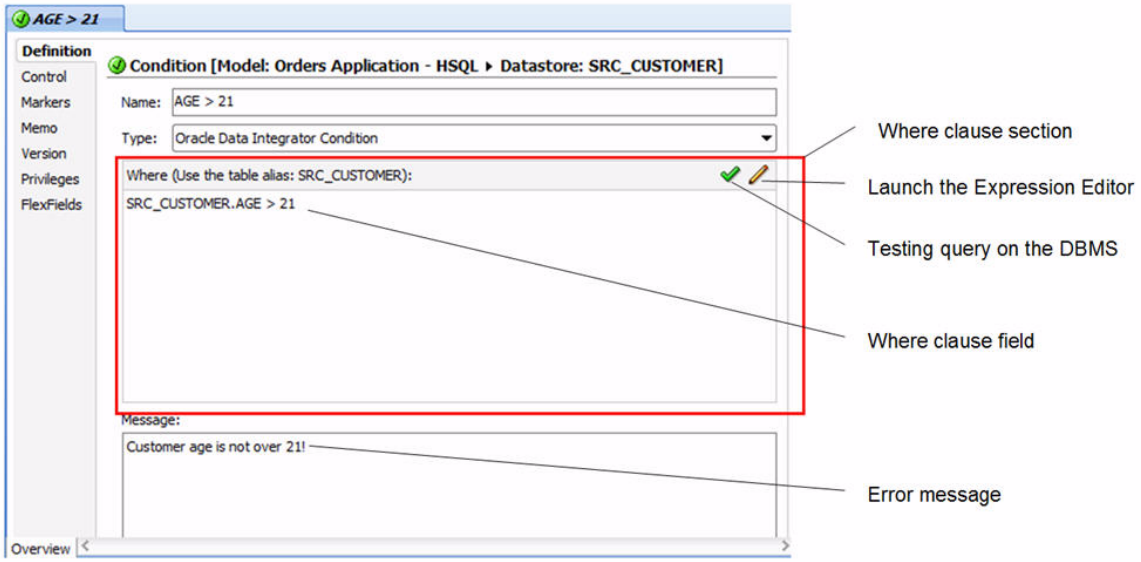
- You can enter this text directly in the Where clause field or you can use the Expression Editor. To open the Expression Editor click **Launch the Expression Editor** in the Where clause toolbar menu.
- The constraints created by Oracle Data Integrator are not actually created on the database. The constraints are stored in the Repository.

- In the Message field, specify the error message as it will appear in your error table:

```
Customer age is not over 21!
```

- Figure 5–2 shows the Condition Editor.

Figure 5–2 Condition Editor



5. From the File main menu, select **Save** to save the condition.

5.2.3.2 Reference Constraint

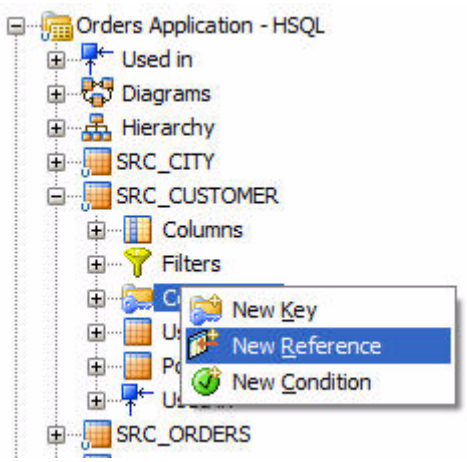
This section describes how to create a reference constraint based on the CITY_ID column between the SRC_CUSTOMER table and the SRC_CITY table.

This constraint allows checking that customers are located in a city that exists in the SRC_CITY table.

To create the reference constraint:

1. In the Models accordion in Designer Navigator, expand the *Orders Application - HSQL* model.
2. Expand the SRC_CUSTOMER datastore.
3. Right-click the Constraints node and select **New Reference** as shown in Figure 5–3.

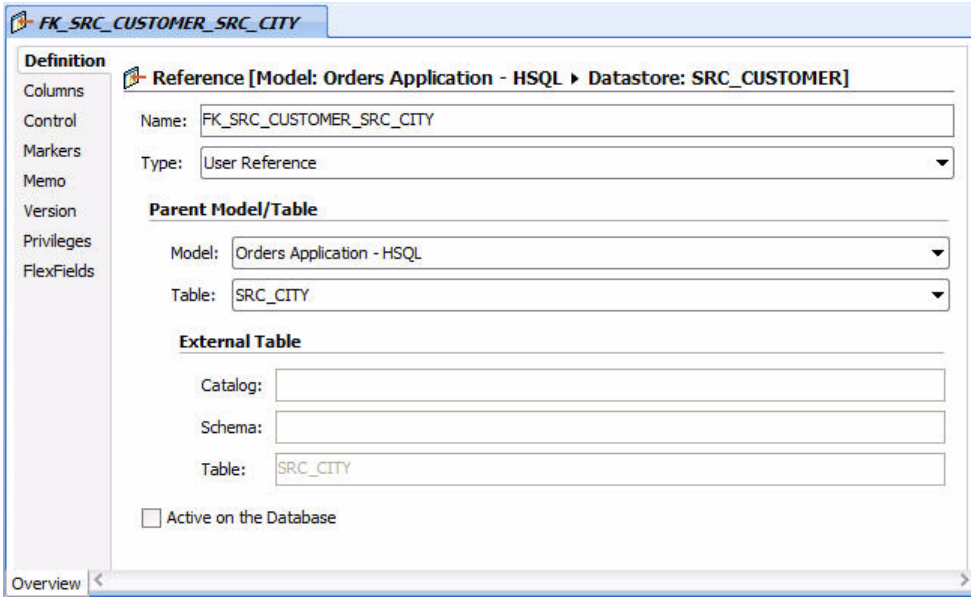
Figure 5–3 Insert New Reference



4. In the Definition tab of the Reference Editor:
   - From the Type list, select **User Reference**.
   - From the Model list in the Parent Model/Table section, select **Orders Application - HSQL**. This is the data model containing the table you want to link to.
   - From the Table list, select **SRC_CITY**. This is the table you want to link to.

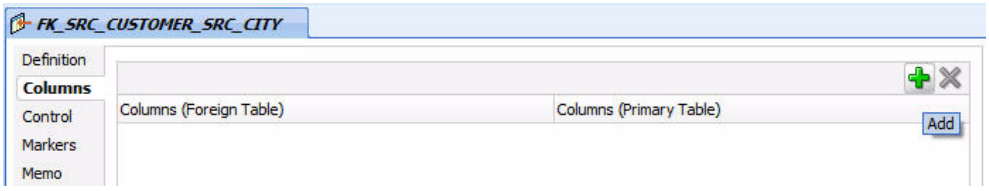
   Figure 5–4 shows the Reference Editor.

Figure 5–4 Reference Editor



5. In the Reference Editor, go to the Columns tab.
6. On the Columns tab, click **Add** as shown in Figure 5–5.

Figure 5–5 Columns tab of the Reference Editor

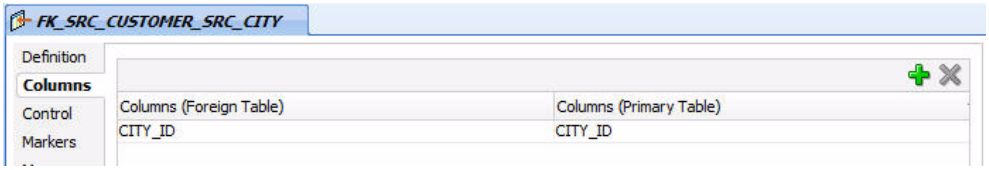


A new row is inserted in the columns table.

7. In this step you define the matching columns:
   - Click on the row that appears. This will bring up a drop-down list containing all of the columns in the appropriate table.
   - From the Columns (Foreign Table) list, select **CITY_ID**.
   - From the Columns (Primary Table) list, select **CITY_ID**.

   Figure 5–6 shows the Columns tab of the Reference Editor with the selected matching columns.

Figure 5–6 Columns tab of the Reference Editor with matching columns



Note that in this example the Foreign Table is SRC_CUSTOMER and the Primary Table is SRC_CITY. Note also that it is not required for foreign keys that the column names of the Foreign Table and the Primary Table match. It just happens that they do in this example.

8. Select **File** > **Save** to save this reference.

   Tip: You can alternately use the [CTRL - S] shortcut to save the current Editor.

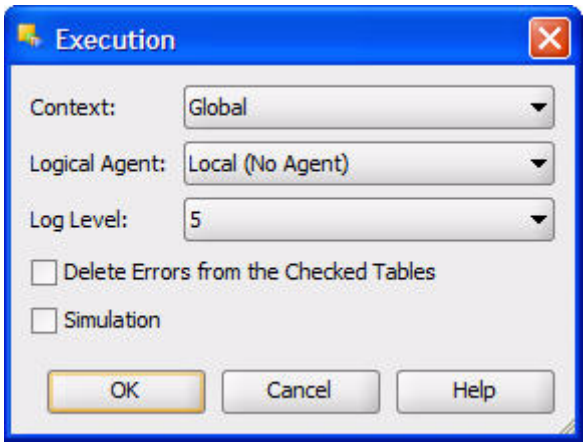
5.2.4 Run the Static Control

Running the static control verifies the constraints defined on a datastore. You can now verify the data in the SRC_CUSTOMER datastore against the constraints defined in Section 5.2.3, "Creating Constraints".

To run the static control:

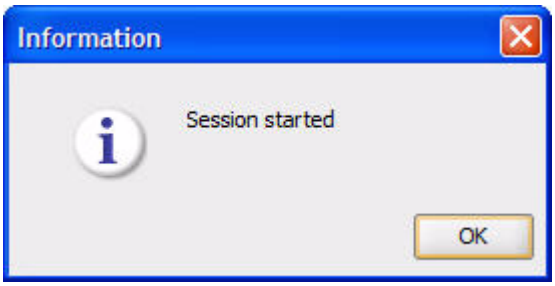
1. In the Models accordion in Designer Navigator, right-click the SRC_CUSTOMER datastore.
2. Select **Control** > **Check**.
3. The Execution dialog is displayed as shown in Figure 5–7.

Figure 5–7 Execution Dialog



4. Click **OK** in the Execution dialog.
5. The Information Dialog is displayed as shown in Figure 5–8.

Figure 5–8 Information Dialog



6. Click **OK** in the Information Dialog.

Oracle Data Integrator automatically generates all of the code required to check your data and start an execution session.

5.2.5 Follow the Execution of the Control in Operator Navigator

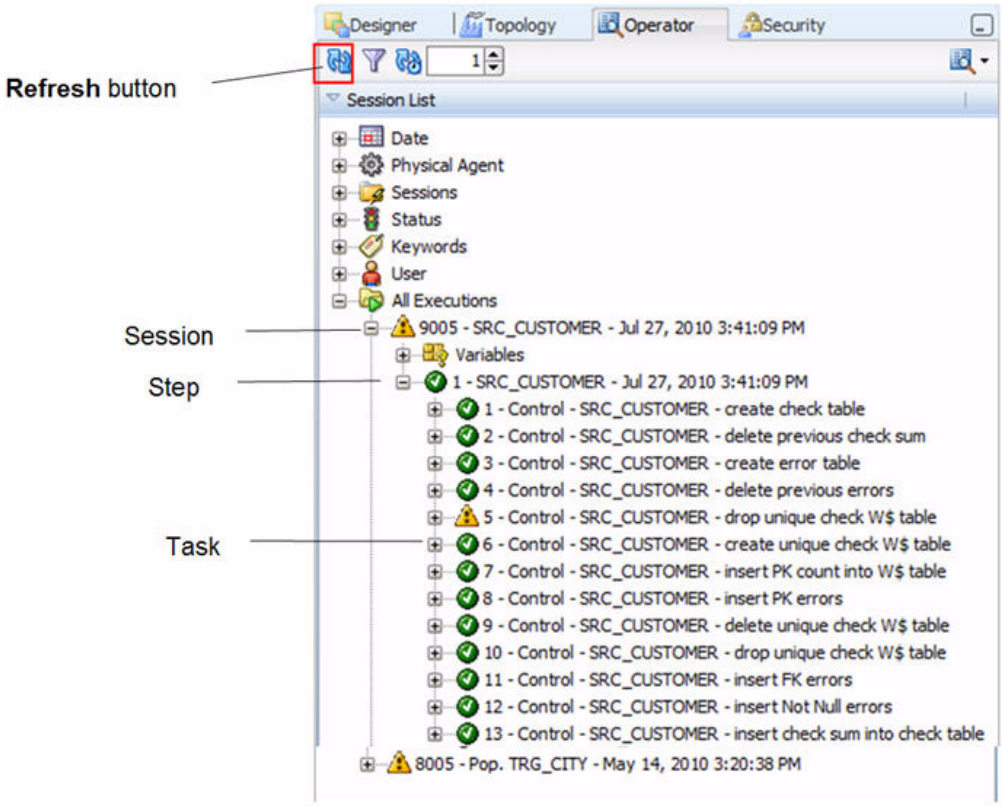
Through Operator Navigator, you can view your execution results and manage your development executions in the sessions.

To view the execution results of your control:

1. In the Session List accordion in Operator Navigator, expand the All Executions node.

   The Session List displays all sessions organized per date, physical agent, status, keywords, and so forth.
2. Refresh the displayed information clicking **Refresh** in the Operator Navigator toolbar.
3. The log for one execution session appears as shown in Figure 5–9.

Figure 5–9 Session List in Operator Navigator



The log comprises 3 levels:

- The session (corresponds to an execution of a scenario, an interface, a package or a procedure undertaken by an execution agent)
- The step (corresponds to a checked datastore, an interface, a procedure or a step in a package or in a scenario)
- The task (corresponds to an elementary task of the interface, process or check)

5.2.6 Interpreting the Results in Operator Navigator

This section describes how to determine the invalid records. These are the records that do not satisfy the constraints and has been rejected by the static control.

This section includes the following topics:

- Determining the Number of Invalid Records
- Reviewing the Invalid Records

5.2.6.1 Determining the Number of Invalid Records

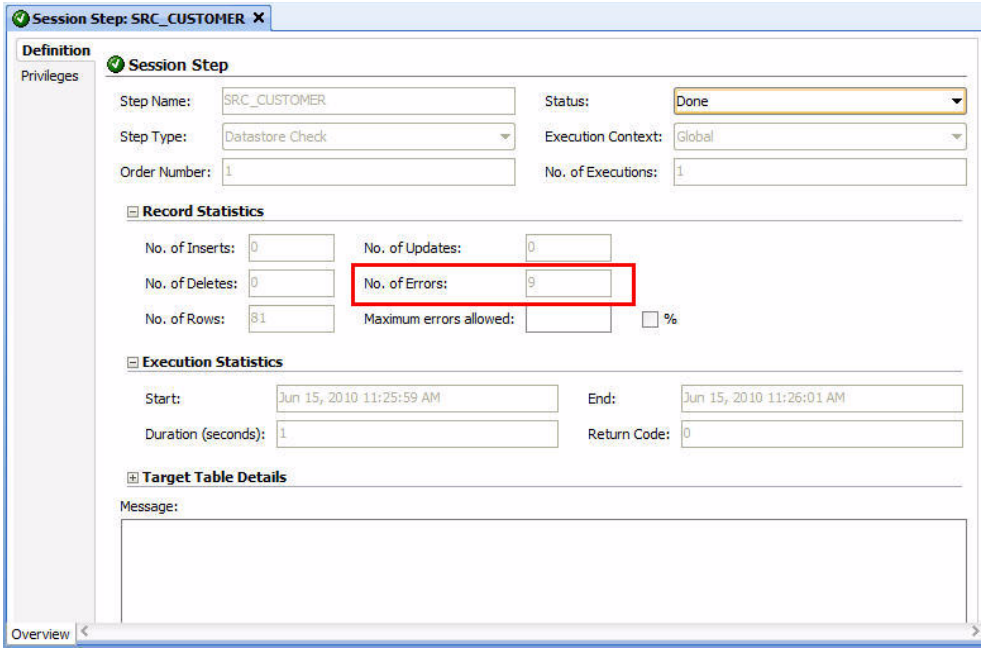
To determine the number of invalid records:

1. In the Session List accordion in Operator Navigator, expand the All Executions node and the SRC_CUSTOMER session.
2. Double-click the SRC_CUSTOMER step to open the Session Step Editor.

3. The Record Statistics section details the changes performed during the static control. These changes include the number of inserts, updates, deletes, errors, and the total number of rows handled during this step.

   Figure 5–10 shows the Session Step Editor of the SRC_CUSTOMER step.

Figure 5–10 SRC_CUSTOMER Session Step Editor



   The number of invalid records is listed in the No. of Errors field. Note that the static control of the SRC_CUSTOMER table has revealed **9** invalid records. These records have been isolated in an error table. See Section 5.2.6.2, "Reviewing the Invalid Records" for more information.

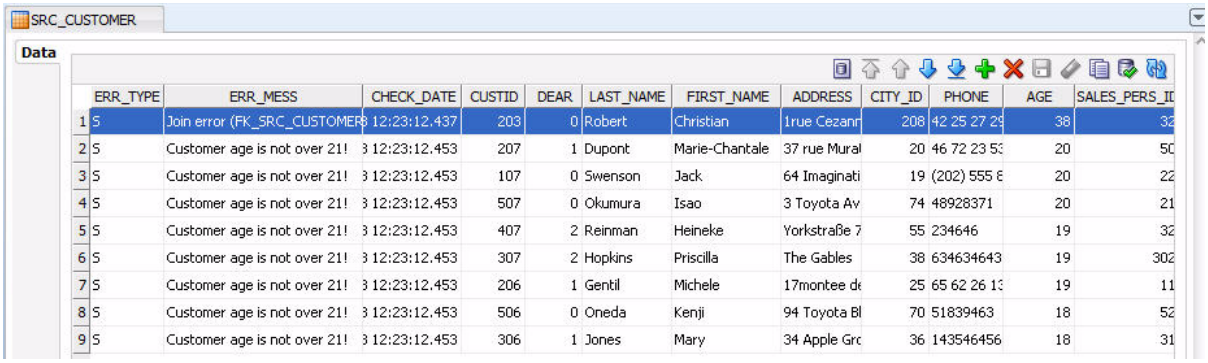
5.2.6.2 Reviewing the Invalid Records

You can access the invalid records by right-clicking on the table in your model and selecting **Control** > **Errors...**

To review the error table of the static control on the SRC_CUSTOMER table:

1. In Designer Navigator, expand the *Orders Application - HSQL* model.
2. Right-click the SRC_CUSTOMER datastore.
3. Select **Control** > **Errors...**
4. The Error Table Editor is displayed as shown in Figure 5–11.

Figure 5–11 Error Table of SRC_CUSTOMER Table



	ERR_TYPE	ERR_MESS	CHECK_DATE	CUSTID	DEAR	LAST_NAME	FIRST_NAME	ADDRESS	CITY_ID	PHONE	AGE	SALES_PERS_ID
1	S	Join error (FK_SRC_CUSTOMER	12:23:12.437	203	0	Robert	Christian	1rue Cezanr	208	42 25 27 29	38	32
2	S	Customer age is not over 21!	12:23:12.453	207	1	Dupont	Marie-Chantale	37 rue Mural	20	46 72 23 53	20	50
3	S	Customer age is not over 21!	12:23:12.453	107	0	Swenson	Jack	64 Imaginati	19	(202) 555 8	20	22
4	S	Customer age is not over 21!	12:23:12.453	507	0	Okumura	Isao	3 Toyota Av	74	48928371	20	21
5	S	Customer age is not over 21!	12:23:12.453	407	2	Reinman	Heineke	Yorkstraße 7	55	234646	19	32
6	S	Customer age is not over 21!	12:23:12.453	307	2	Hopkins	Priscilla	The Gables	38	634634643	19	302
7	S	Customer age is not over 21!	12:23:12.453	206	1	Gentil	Michele	17montee de	25	65 62 26 13	19	11
8	S	Customer age is not over 21!	12:23:12.453	506	0	Oneda	Kenji	94 Toyota Bl	70	51839463	18	52
9	S	Customer age is not over 21!	12:23:12.453	306	1	Jones	Mary	34 Apple Grc	36	143546456	18	31

The records that were rejected by the check process are the following:

- **8** records in violation of the AGE > 21 constraint (the actual age of the customer is 21 or younger, see the AGE column for details).
- **1** record in violation of the FK_CITY_CUSTOMER constraint (The CITY_ID value does not exist in the SRC_CITY table).

You can view the entire record in this Editor. This means that you can instantly see which values are incorrect, for example the invalid CITY_ID value in the top record.

Note that the error message that is displayed is the one that you have defined when setting up the AGE > 21 constraint in Section 5.2.3.1, "Age Constraint".

Now that the static controls have been run on the source data, you are ready to move on to the implementation of integration interfaces.

6

Working with Integration Interfaces

This chapter describes how to work with integration interfaces in Oracle Data Integrator. The demonstration environment includes several example interfaces. In this chapter you learn how to create the following interfaces:

- Pop.TRG_CUSTOMER integration interface: This interface loads the data from the SRC_CUSTOMER table in the *Orders Application - HSQL* model into the TRG_CUSTOMER target table in the *Sales Administration - HSQL* model.
- Pop.TRG_SALES integration interface: This interface loads the data from the SRC_ORDERS table and from the SRC_ORDER_LINES table in the *Orders Application - HSQL* model into the TRG_SALES target table in the *Sales Administration - HSQL* model.

This chapter includes the following sections:

6.1 Pop. TRG_CUSTOMER Interface Example

This section contains the following topics:

6.1.1 Purpose and Integration Requirements

This section describes the integration features and requirements the integration interface Pop. TRG_CUSTOMER is expected to meet.

The purpose of the Pop. TRG_CUSTOMER interface is to load the data from the SRC_CUSTOMER table in the *Orders Application - HSQL* model into the TRG_CUSTOMER target table in the *Sales Administration - HSQL* model.

However, the SRC_CUSTOMER table does not contain all of the data that is required for this operation. The following information has to be added to the target table:

- The age range (AGE_RANGE) that is defined in the SRC_AGE_GROUP flat file in the *Parameters - FILE* model corresponds to the AGE attribute in the source table.
- The last and first names of the customer sales rep. (LAST_NAME and FIRST_NAME) that is defined in the SRC_SALES_PERSON file in the *Parameters - FILE* model correspond to the sales rep. number (SALES_PERS_ID) in the source table.

- The transformed value of the numeric data (0, 1, 2) from the DEAR column in the source table into an standard salutation text string in the target (Mr, Mrs, or Ms).
- The concatenated first and last names of the source customers.

The source data is not always consistent with the integrity rules implemented in the target environment. For this interface, the data has to be cleansed by verifying that all constraints are satisfied and by storing invalid rows in an error table rather than in our target database. In this example, two important integrity rules must be satisfied:

- Customers must be older than 21 (condition AGE > 21)
- The customers must be associated with a city (CITY_ID) that exists in the TRG_CITY table (reference FK_CUST_CITY)

The functional details for these rules and the procedure to follow are given in Section 6.1.3, "Creating the Integration Interface".

6.1.2 Interface Definition

This section describes the integration interface Pop. TRG_CUSTOMER that will be created in this example. See Section 6.1.3, "Creating the Integration Interface" for more information.

The Pop. TRG_CUSTOMER interface uses the following data and transformations:

- One target datastore. Table 6–1 lists the details of the target datastore.

Table 6–1 Target Datastore Details of Pop. TRG_CUSTOMER

Model	Datastore	Description	Type
Sales Administration - HSQL	TRG_CUSTOMER		HSQL table

- Three source datastores. Table 6–2 lists the details of the source datastores.

Table 6–2 Source Datastore Details of Pop. TRG_CUSTOMER

Model	Datastore	Description	Type
Orders Application - HSQL	SRC_CUSTOMER	Customers in the source system	HSQL table
Parameters - FILE	SRC_AGE_GROUP	Age bracket file	File delimited by semicolons
Parameters - FILE	SRC_SALES_PERSON	Salesperson file	File of fixed-size records

- One join. Table 6–3 lists the details of the join.

Table 6–3 Joins used in Pop. TRG_CUSTOMER

Join	Description	SQL Rule	Execution Location
Sales Representatives and Customers	Join SRC_SALES_PERSON and SRC_CUSTOMER	`SRC_CUSTOMER.SALES_PERS_ID = SRC_SALES_PERSON.SALES_PERS_ID`	Staging area

- One lookup table. Table 6–4 lists the details of the lookup table.

Table 6–4 Lookups used in Pop. TRG_CUSTOMER

Lookup	Description	SQL Rule	Execution Location
Customers and age range	The customer's age must between the minimum and maximum ages in the file	`SRC_CUSTOMER.AGE between SRC_AGE_GROUP.AGE_MIN and SRC_AGE_GROUP.AGE_MAX`	Staging area

- Several transformation rules. Table 6–5 lists the details of the transformation rules.

Table 6–5 Transformation Rules used in Pop. TRG_CUSTOMER

Target Column	Origin	SQL Rule	Execution Location
CUST_ID	SRC_CUSTOMER.CUSTID	`SRC_CUSTOMER.CUSTID`	Source
DEAR	If SRC_CUSTOMER.DEAR = 0 then 'MR' If SRC_CUSTOMER.DEAR = 1 then 'MRS' else 'MS'	`CASEWHEN(SRC_CUSTOMER.DEAR=0, 'MR', CASEWHEN(SRC_CUSTOMER.DEAR=1, 'MRS', 'MS') )`	Source
CUST_NAME	Concatenation of SRC_CUSTOMER.FIRST_NAME and SRC_CUSTOMER.LAST_NAME in upper case	`SRC_CUSTOMER.FIRST_NAME || ' ' || UCASE(SRC_CUSTOMER.LAST_NAME)`	Source
ADDRESS	SRC_CUSTOMER.ADDRESS	`SRC_CUSTOMER.ADDRESS`	Source
CITY_ID	SRC_CUSTOMER.CITY_ID	`SRC_CUSTOMER.CITY_ID`	Source
PHONE	SRC_CUSTOMER.PHONE	`SRC_CUSTOMER.PHONE`	Source
AGE	SRC_CUSTOMER.AGE	`SRC_CUSTOMER.AGE`	Source
AGE_RANGE	SRC_AGE_GROUP.AGE_RANGE	`SRC_AGE_GROUP.AGE_RANGE`	Staging area
SALES_PERS	Concatenation of SRC_SALES_PERSON.FIRST_NAME and SRC_SALES_PERSON.LAST_NAME in uppercase	`SRC_SALES_PERSON.FIRST_NAME || ' ' || UCASE(SRC_SALES_PERSON.LAST_NAME)`	Staging area
CRE_DATE	Today's date	`CURDATE()`	Target
UPD_DATE	Today's date	`CURDATE()`	Target

6.1.3 Creating the Integration Interface

This section describes how to create the Pop. TRG_CUSTOMER integration interface. To create the Pop. TRG_CUSTOMER interface perform the following procedure:

1. Insert a New Integration Interface

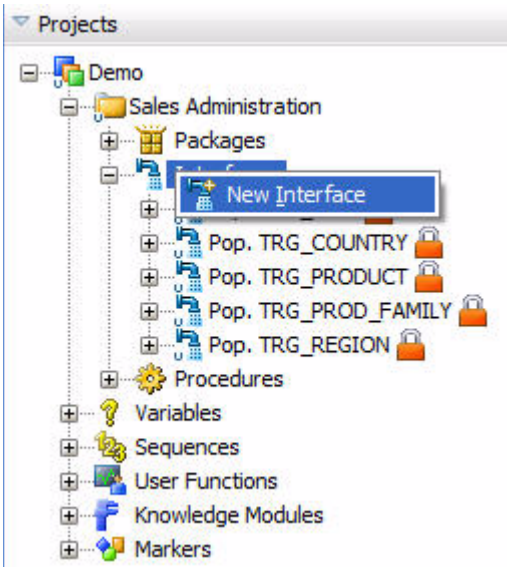
2. Define the Target Datastore
3. Define the Source Datastores
4. Define the Lookup Table
5. Define the Join between the Source Datastores
6. Define the Mappings
7. Define the Data Loading Strategies (LKM)
8. Define the Data Integration Strategies (IKM)
9. Define the Data Control Strategy

6.1.3.1 Insert a New Integration Interface

To create a new integration interface:

1. In Designer Navigator, expand the Demo project node in the Projects accordion.
2. Expand the Sales Administration node.
3. In the Sales Administration folder, right-click the Interfaces node and select **New Interface** as shown in Figure 6–1.

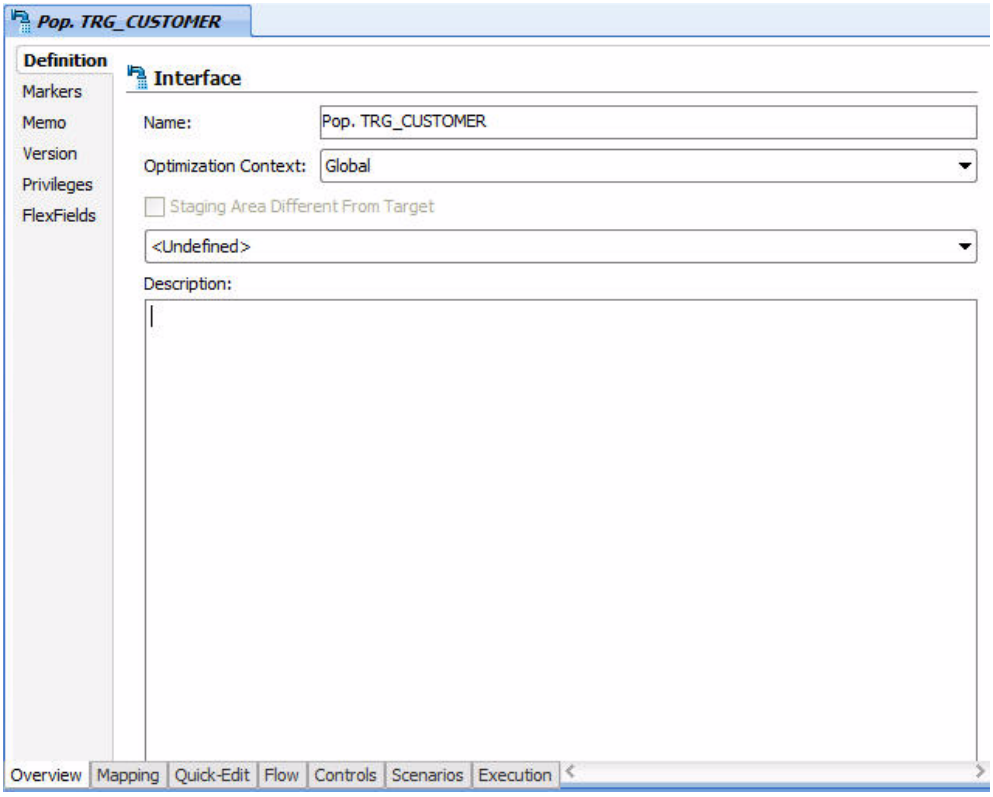
Figure 6–1 Insert New Interface



The Interface Editor is displayed.

4. On the Definition tab of the Interface Editor, enter the name of your interface (Pop. TRG_CUSTOMER) in the Name field as shown in Figure 6–2.

Figure 6–2 Interface Editor



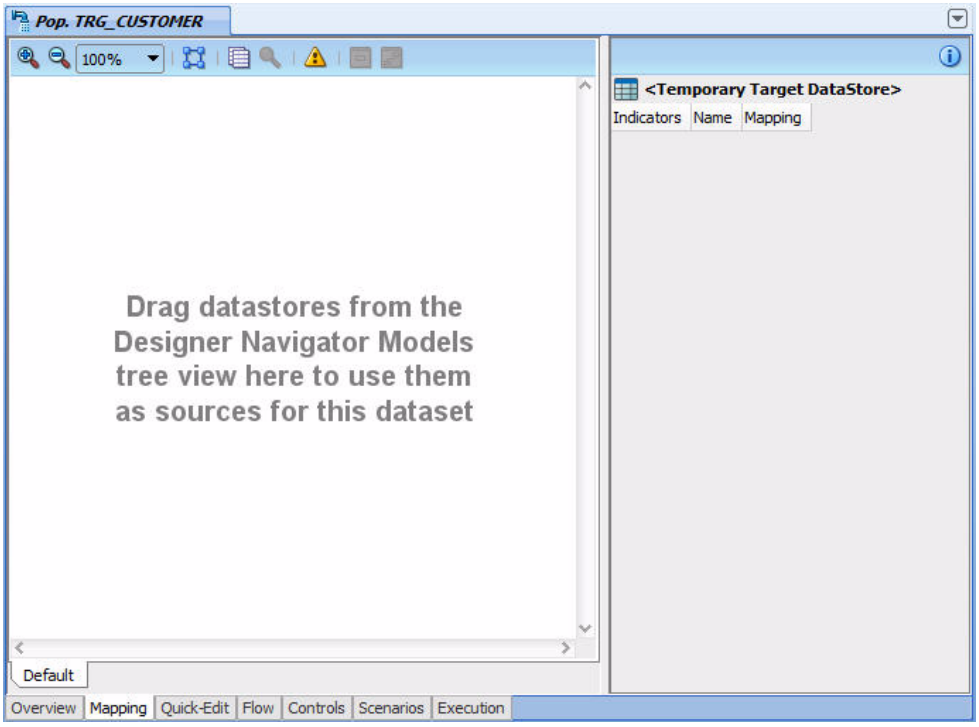
6.1.3.2 Define the Target Datastore

The target datastore is the element that will be loaded by the interface.

To insert the target datastore in the Pop. TRG_CUSTOMER interface:

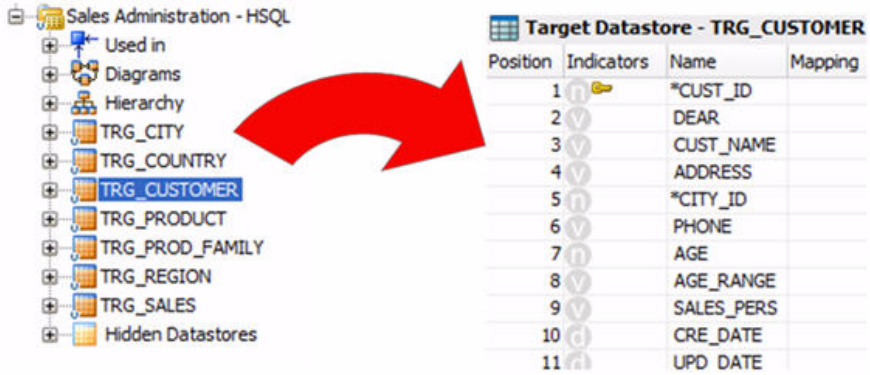
1. Go to the Mapping tab of the Interface Editor.
2. The Mapping tab displays in the interface diagram as shown in Figure 6–3.

Figure 6–3 Mapping Tab of Interface Editor



3. In the Designer Navigator, expand the Models accordion and the *Sales Administration - HSQL* model.
4. Select the TRG_CUSTOMER datastore under the *Sales Administration - HSQL model* and drag it into the Target Datastore panel as shown in Figure 6–4.

Figure 6–4 The Target Datastore



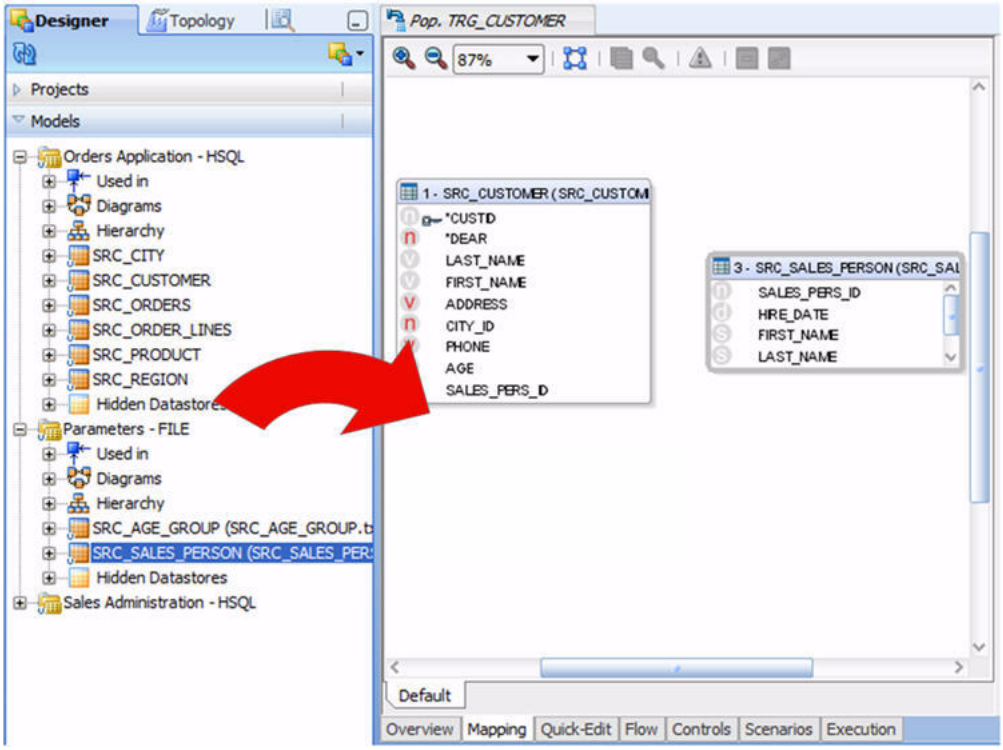
6.1.3.3 Define the Source Datastores

The source datastores contain data used to load the target datastore. Two types of datastores can be used as an interface source: datastores from the models and temporary datastores that are the target of an interface. This example uses datastores from the *Orders Application - HSQL* and *Parameters - FILE* models.

To add source datastores to the Pop. TRG_CUSTOMER interface:

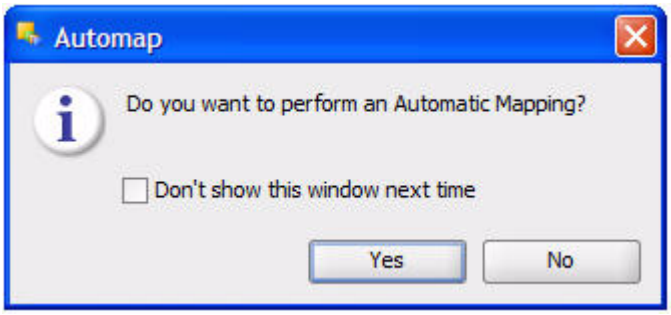
1. In the Mapping tab, drag the following source datastores into the Source Diagram:
   - SRC_CUSTOMER from the *Orders Application - HSQL* model
   - SRC_SALES_PERSON from the *Parameters - FILE* model
2. The Mapping tab of your Interface Editor should look like shown in Figure 6–5.

Figure 6–5 The Source Datastores



3. The Automap Dialog appears as shown in Figure 6–6.

Figure 6–6 Automap Dialog



Click **Yes** to confirm the use of automatic field to field mapping by Oracle Data Integrator. The automatic mapping is performed when you drop a source datastore in the Source Diagram.

6.1.3.4 Define the Lookup Table

This section describes how to create a lookup that defines that the customer's age must be between the minimum and maximum ages in the file.

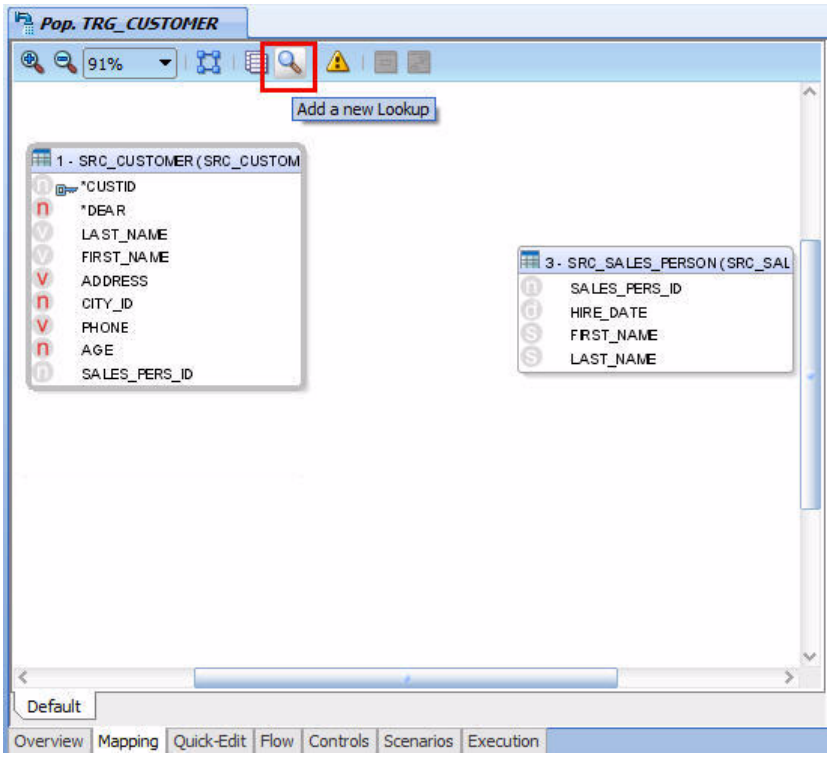
A lookup is a datastore (from a model or the target datastore of an interface) - called the *lookup table* - associated to a source datastore - the *driving table* - via a join expression and from which data can be fetched and used in mappings.

Lookup tables are added with the Lookup Wizard.

To create a lookup in the Pop. TRG_CUSTOMER interface:

1. From the Source Diagram toolbar menu, select **Add a new Lookup** as shown in Figure 6–7.

Figure 6–7 Launching the Lookup Table Wizard in the Source Diagram



The Lookup Wizard opens.

2. In the Lookup Wizard, select **SRC_CUSTOMER(SRC_CUSTOMER)** from the Driving Table list.

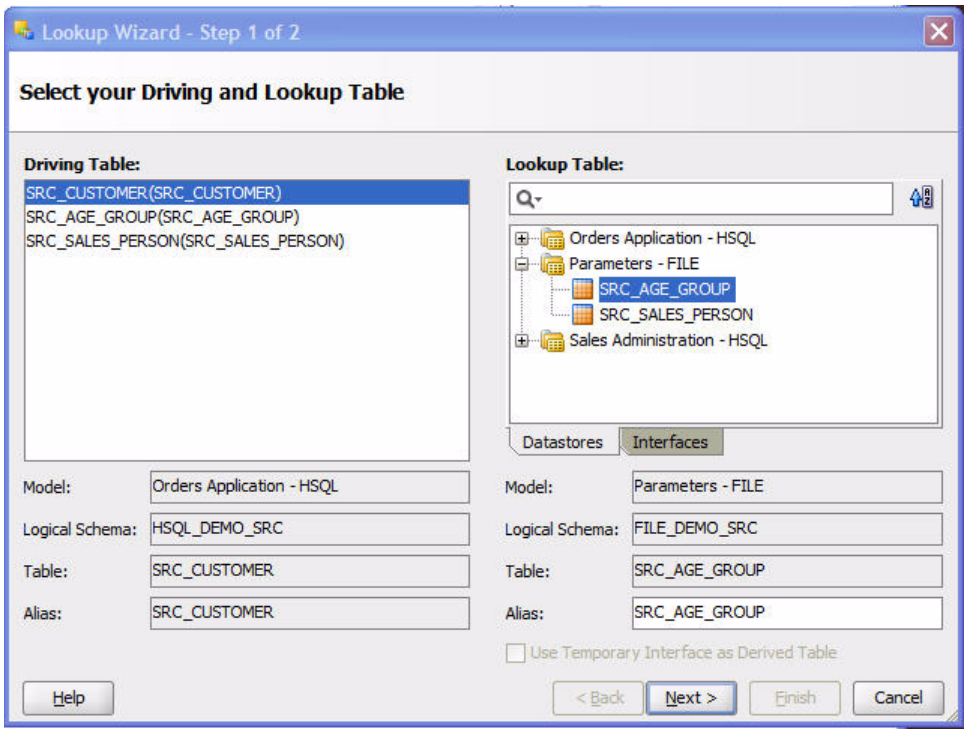
   Note that source datastores for the current diagram appear here and that lookups do not appear in the list.

3. From the Lookup Table pane select the SRC_AGE_GROUP datastore from the *Parameters - FILE* model on the Datastores tab.

   The SRC_AGE_GROUP datastore will be used as a lookup table.

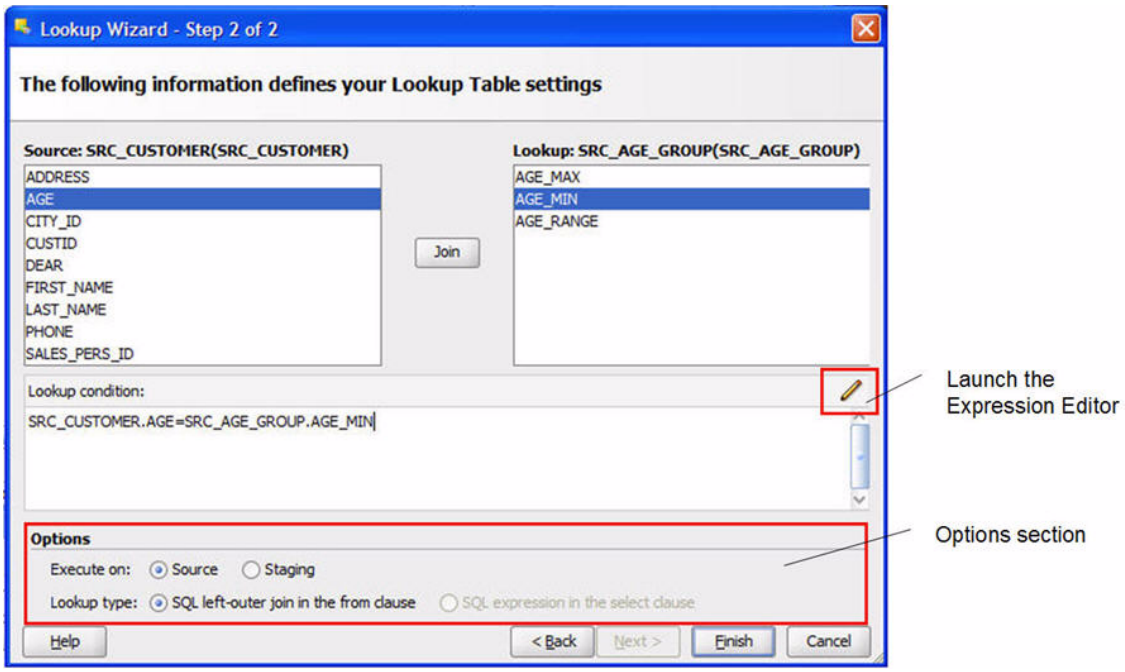
   Figure 6–8 shows the first screen of the Lookup Wizard.

Figure 6–8 First screen of the Lookup Wizard



4. Click **Next**.
5. On the left pane, select the **AGE** source column from the driving table.
6. On the right pane, select the **AGE_MIN** column of the lookup table.
7. Click **Join**. The join condition appears in the Lookup condition text field as shown in Figure 6–9.

Figure 6–9 Second Screen of the Lookup Wizard



8. In the Options section, select **Staging** for the execution location.
9. Click **Launch the Expression Editor** and modify the lookup condition as follows:
   - Replace the equals sign (=) with the string `between`
   - Add the following string at the end of the expression:

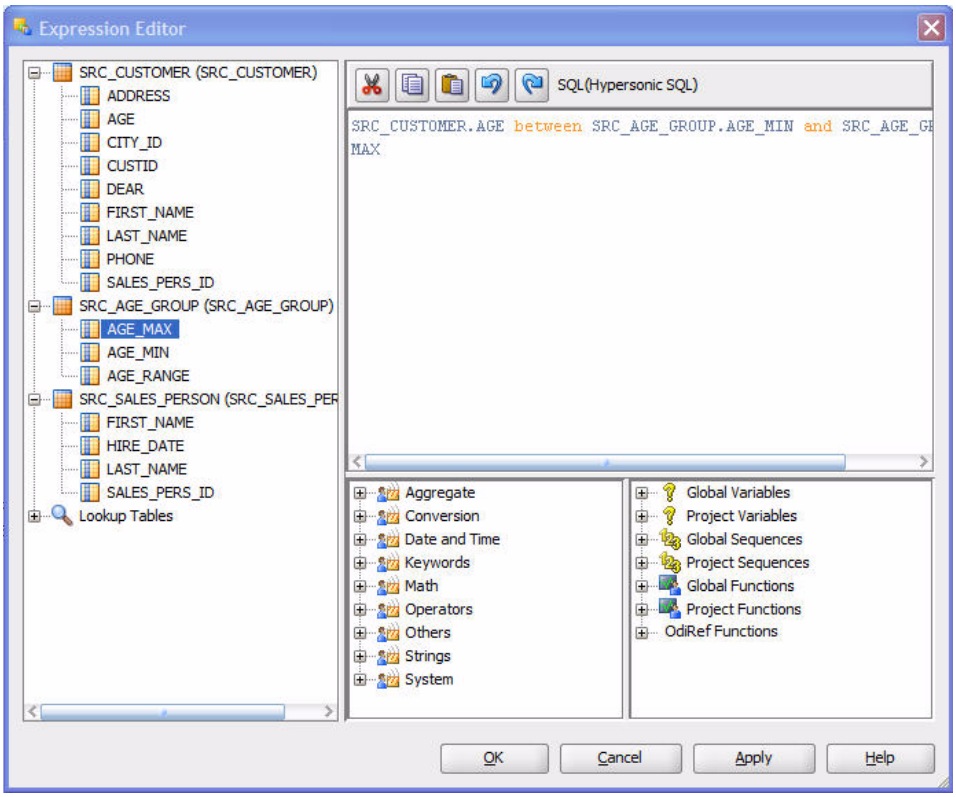
     ```
     and SRC_AGE_GROUP.AGE_MAX
     ```

     This adds the AGE_MAX column from the SRC_AGE_GROUP datastore.
10. You should have the following join expression:

    ```
    SRC_CUSTOMER.AGE between SRC_AGE_GROUP.AGE_MIN and SRC_AGE_
    GROUP.AGE_MAX
    ```

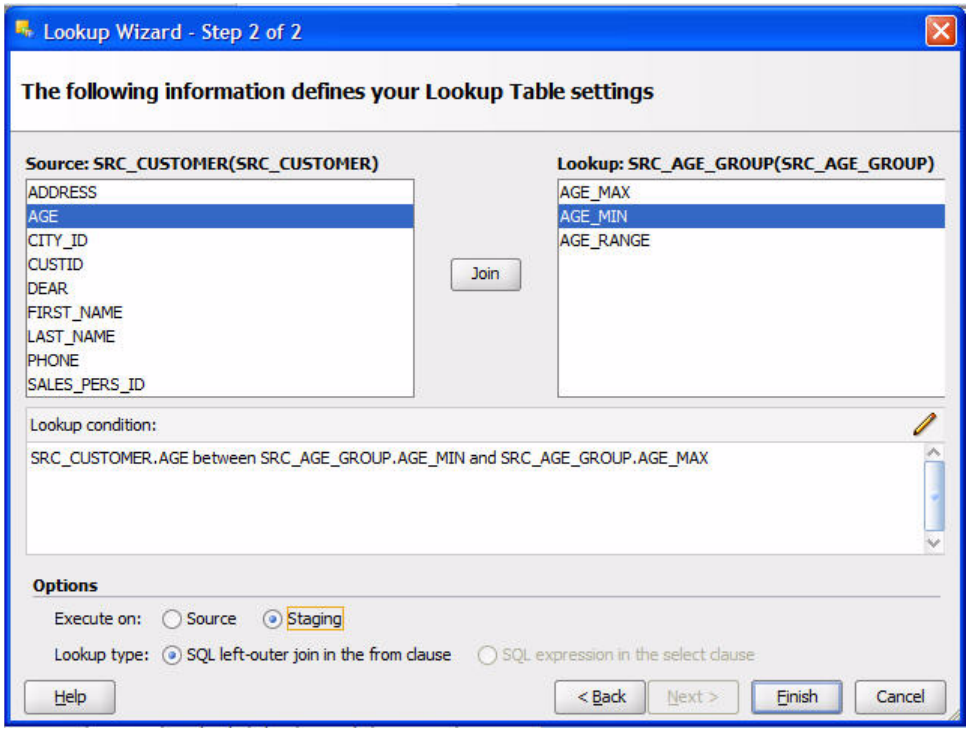
    This corresponds to a join between the SRC_CUSTOMER and the SRC_AGE_GROUP datastore and defines that the customer's age must between the minimum and maximum ages in the file. Figure 6–10 shows the Expression Editor with the lookup condition.

Figure 6–10 Expression Editor with modified lookup condition



11. In the Expression Editor, click **OK**.
12. The modified lookup condition appears in the Lookup Wizard as shown in Figure 6–11.

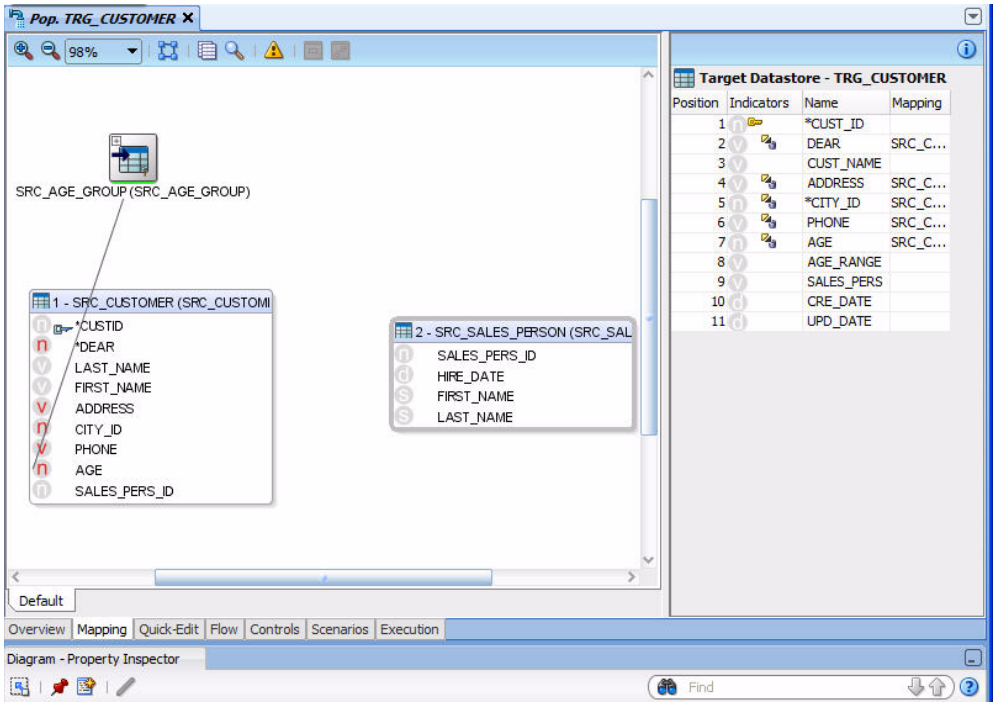
Figure 6–11 Second Screen of the Lookup Wizard with modified lookup condition



13. Click **Finish**.

The Source Diagram appears as shown in Figure 6–12.

Figure 6–12 Source Diagram of Pop. TRG_CUSTOMER Interface



Note: If references were already defined in the models to link the source datastores, these references would have appeared automatically as joins in the source diagram.

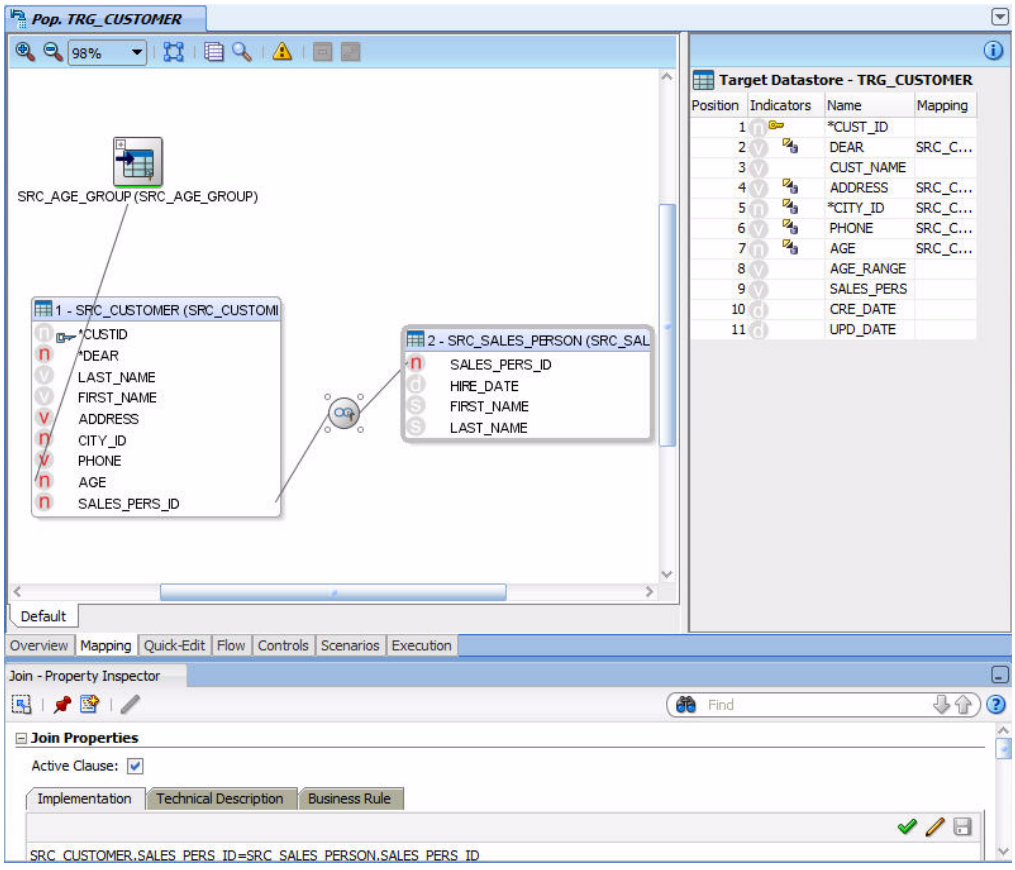
6.1.3.5 Define the Join between the Source Datastores

This section describes how to define a join between the source datastores.

To create the join defined in Table 6–3:

1. In the Source Diagram, select the SALES_PERS_ID column of the SRC_ CUSTOMER datastore.
2. Drag and drop it on the SALES_PERS_ID column of the SRC_SALES_PERSON datastore. A join linking the two datastores appears as shown in Figure 6–13. This is the join on the sales representative identifier.

Figure 6–13 Source Diagram of the Pop.TRG_CUSTOMER Interface with a Lookup and a Join



6.1.3.6 Define the Mappings

In the Target Datastore panel of your interface, columns with names that match their sources are automatically mapped. The automatic mapping is done by the matching of the column names. Most of the transformation rules listed in Table 6–5 have been defined by the automatic mapping. In addition to this automatic mappings, you have to define the transformation rules for the following fields: CUST_ID, DEAR, CUST_ NAME, AGE_RANGE, SALES_PERS, CRE_DATE and UPD_DATE.

The transformation rules, also called *mappings*, are defined on the target column.

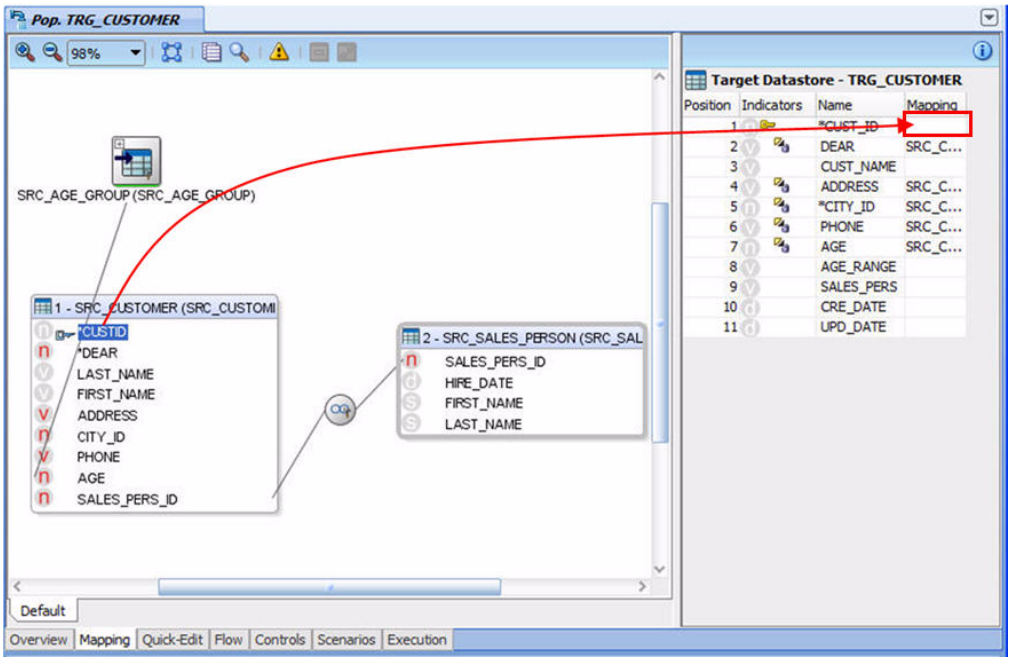
CUST_ID Mapping

The CUST_ID mapping maps the SRC_CUSTOMER.CUSTID source column to the TRG_CUSTOMER.CUST_ID target column. Note that these 2 columns have not been automatically mapped, since their names are slightly different.

To define the mapping for the CUST_ID target column:

1. In the Source Diagram, select the CUSTID column in the SRC_CUSTOMER datastore.
2. Drag it into the Mapping field in the Target Datastore panel as shown in Figure 6–14.

Figure 6–14 CUST_ID Mapping



3. Select the Mapping field in the Target Datastore to display its properties in the Property Inspector.
4. Verify that the execution location is set to **Source** as shown in Figure 6–15.

Figure 6–15 Execution Locations

Note: Possible execution locations are: **Source**, **Target**, and **Staging Area**. Make sure that you select the environment in which your transformation will be executed as specified in Table 6–5. Select this environment by clicking on one of the radio buttons as shown in Figure 6–15.

DEAR Mapping

This transformation rule maps the source datastore's DEAR column (numeric) as a string expression (0 -->'MR', 1 -->'MRS', 2 -->'MS').

To define the mapping for the DEAR target column:

1. In the Target Datastore panel, select the Mapping field of the DEAR target column to display the mapping properties in the Property Inspector.

   Tip: Click **Freeze View** in the Property Inspector toolbar to continue displaying the current contents of the Property Inspector even if you select a different component that would normally change the contents of the Property Inspector. The Freeze View button is:

   To unfreeze a frozen instance of the Property Inspector and allow it to track the active selection, click **Freeze View** again.

2. In the Implementation field, enter the following mapping expression:

   ```
   CASEWHEN(SRC_CUSTOMER.DEAR=0, 'MR', CASEWHEN( SRC_
   CUSTOMER.DEAR=1, 'MRS', 'MS') )
   ```

   Tip: You can drag source columns, for example the SRC_CUSTOMER.DEAR column, into the Implementation field.

3. Verify that the execution location is set to **Source**.

CUST_NAME Mapping

This transformation rule maps the concatenated value of the first name and uppercase last name of each customer.

To define the mapping for the CUST_NAME target column:

1. In the Target Datastore panel, select the Mapping field of the CUST_NAME target column to display the mapping properties in the Property Inspector.
2. In the Implementation field, enter the following mapping expression:

   ```
   SRC_CUSTOMER.FIRST_NAME || ' ' || UCASE(SRC_CUSTOMER.LAST_
   NAME)
   ```

   Tip: Use the Expression Editor to create this rule. By using the Expression Editor, you can avoid most common syntax errors.

3. Verify that the execution location is set to **Source**.

AGE_RANGE Mapping

This mapping maps the SRC_AGE_GROUP.AGE_RANGE to the TRG_CUSTOMER.AGE_RANGE.

To define the mapping for the AGE_RANGE target column:

1. In the Target Datastore panel, select the Mapping field of the AGE_RANGE target column to display the mapping properties in the Property Inspector.
2. In the Implementation field, enter the following mapping expression:

```
SRC_AGE_GROUP.AGE_RANGE
```

3. Verify that the execution location is set to **Staging Area**.

Note: This rule must be executed in the staging area! The source in this example is a flat file, and as such is not associated to an engine that supports concatenation.

SALES_PERS Mapping

This will map the concatenated value of the first name and uppercase last name of each salesperson.

To define the mapping for the SALES_PERS target column:

1. In the Target Datastore panel, select the Mapping field of the SALES_PERS target column to display the mapping properties in the Property Inspector.
2. In the Implementation field, enter the following mapping expression:

   ```
   SRC_SALES_PERSON.FIRST_NAME || ' ' || UCASE(SRC_SALES_
   PERSON.LAST_NAME)
   ```

3. Verify that the execution location is set to **Staging Area**.

Note: This rule must be executed in the staging area! The source in this example is a flat file, and as such is not associated to an engine that supports concatenation.

CRE_DATE Mapping

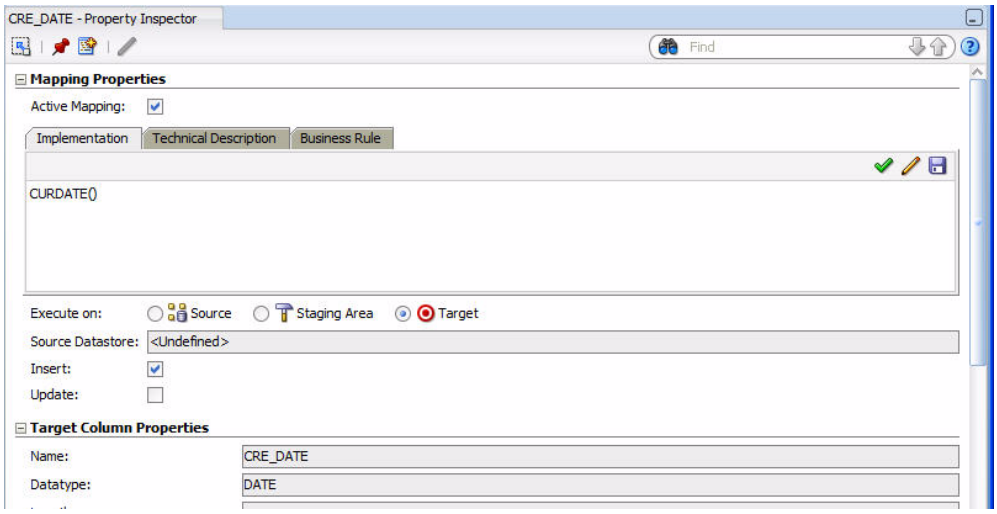
To define the mapping for the CRE_DATE target column:

1. In the Target Datastore panel, select the Mapping field of the CRE_DATE target column to display the mapping properties in the Property Inspector.
2. In the Implementation field, enter the following mapping expression:

   CURDATE()

3. Verify that **Active Mapping** is selected.
4. Verify that the execution location is set to **Target**.
5. Unselect **Update**. The mapping will be performed only on Insert.
6. The Property Inspector of the CRE_DATE mapping appears as shown in Figure 6–16.

Figure 6–16 Property Inspector of the CRE_DATE Mapping



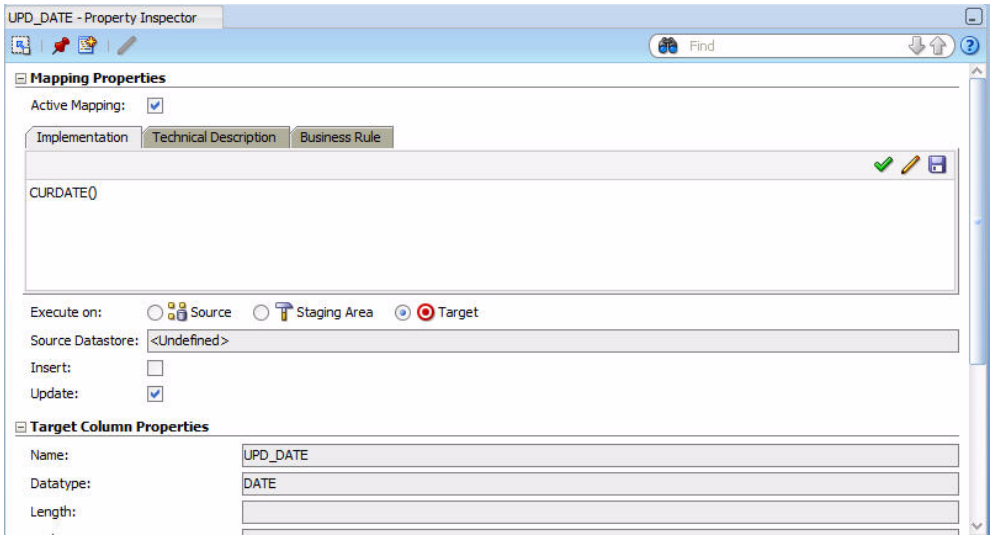
UPD_DATE Mapping

To define the mapping for the UPD_DATE target column:

1. In the Target Datastore panel, select the Mapping field of the UPD_DATE target column to display the mapping properties in the Property Inspector.
2. In the Implementation field, enter the following mapping expression:

   CURDATE()
3. Verify that **Active Mapping** is selected.
4. Verify that the execution location is set to **Target**.
5. Unselect **Insert**. The mapping will be performed only on Update.
6. The Property Inspector of the UPD_DATE mapping appears as shown in Figure 6–17.

Figure 6–17 Property Inspector of the UPD_DATE Mapping



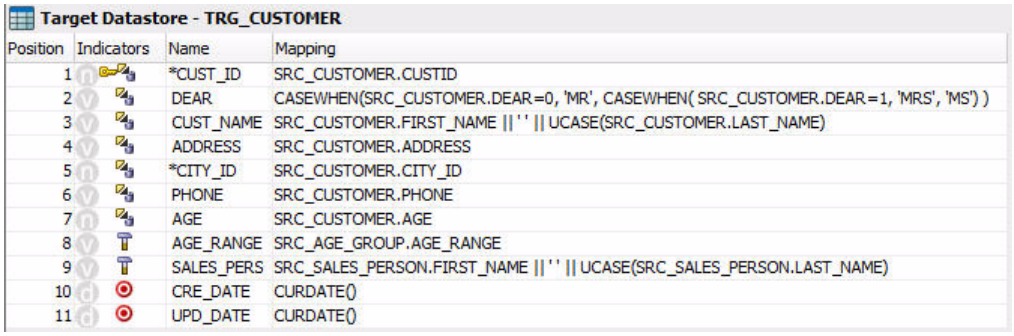
Notes on the Expression Editor

- The Expression Editor that is used to build the Expressions does not contain all of the functions specific to a technology. It contains only functions that are common to a large number of technologies. The fact that a function does not appear in the Expression Editor does not prevent it from being entered manually and used in an Expression.
- If you were to execute this mapping on the target, the Expression Editor would give you the syntax for your target system (also Hypersonic SQL in this case).
- Clicking **Check the expression in the DBMS** calls your source server to check the syntax of the SQL code you have entered. This check can only be performed when your rule is entirely executed on the source server.

The Target Datastore Panel

Your transformation rules appear in the Target Datastore panel as shown in Figure 6–18.

Figure 6–18 Target Datastore Mappings



Two types of icons are used in the Indicators column of the Target Datastore panel:

- The first letter of the data type in the target column (n: numeric, v: varchar, d: date)
- The execution location of the expression.

Table 6–6 Execution Location Icons

Icon	Description
	Source
	Staging area
	Target
	Error in the mapping. If this icon appears, select the target column in error and verify your input in the Property Inspector.

Note that you can also use the Quick-Edit Editor to create and view an integration interface. See "Using the Quick-Edit Editor" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for more information.

6.1.3.7 Define the Data Loading Strategies (LKM)

The data loading strategies are defined in the Flow tab of the Interface Editor. Oracle Data Integrator automatically computes the flow depending on the configuration in the interface's diagram. It proposes default KMs for the data flow. The Flow tab enables you to view the data flow and select the KMs used to load and integrate data.

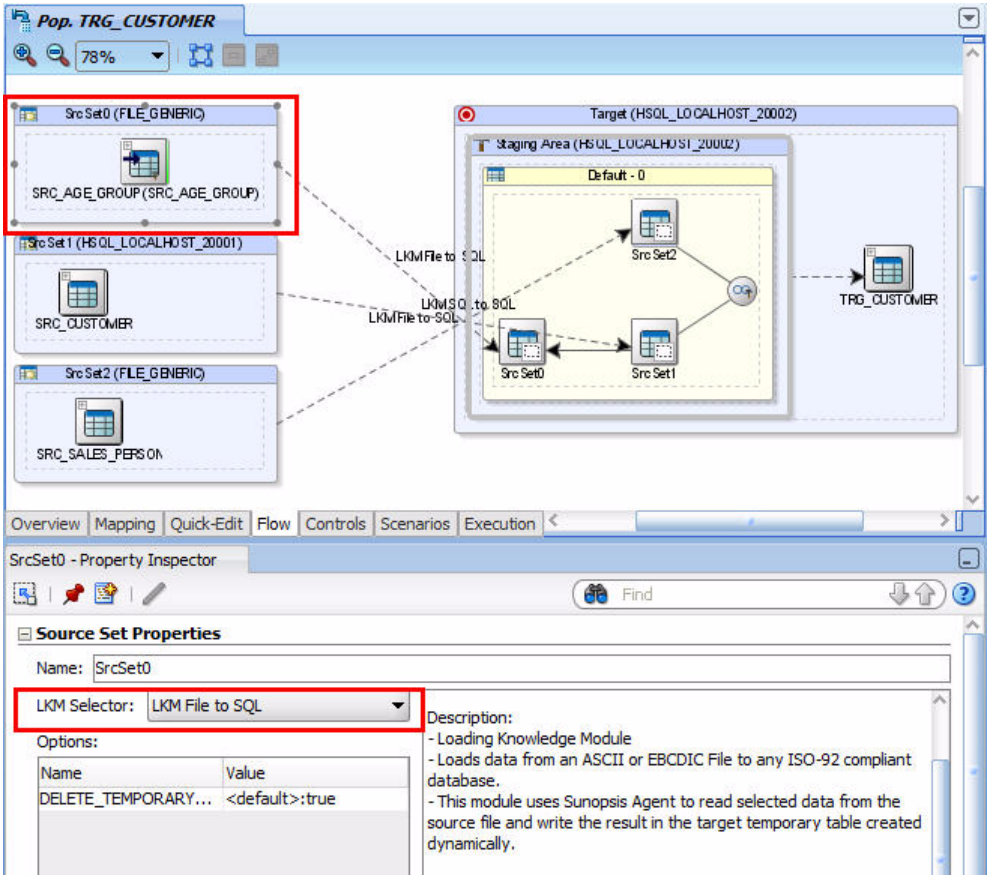
Loading Knowledge Modules (LKM) are used for loading strategies and Integration Knowledge Modules (IKM) are used for integration strategies.

You have to define the way to retrieve the data from the SRC_AGE_GROUP, SRC_SALES_PERSON files and from the SRC_CUSTOMER table in your source environment.

To define the loading strategies:

1. In the Flow tab of the Interface Editor, select the source set that corresponds to the loading of the SRC_AGE_GROUP file. In this example, this is the SrcSet0 (FILE_GENERIC). The Property Inspector should display the properties of this source set.
2. In the Property Inspector, verify that the **LKM File to SQL** is selected in the LKM Selector list as shown in Figure 6–19.

Figure 6–19 Flow tab of the Pop. TRG_CUSTOMER Interface Editor



3. Select the source set that corresponds to the loading of the SRC_CUSTOMER table.
4. In the Property Inspector, verify that the **LKM SQL to SQL** is selected in the LKM Selector list.

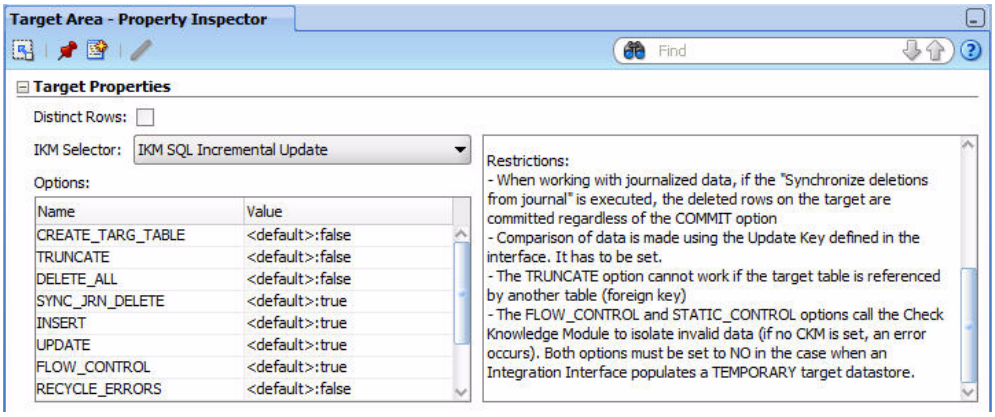
6.1.3.8 Define the Data Integration Strategies (IKM)

After defining the loading phase, you need to define the strategy to adopt for the integration of the data into the target table.

To define the integration strategies:

1. In the Flow tab of the Interface Editor, select the Target object in the Flow Diagram. The Property Inspector should display the properties of the target.
2. In the Property Inspector, verify that the **IKM SQL Incremental Update** is selected in the IKM Selector list.
3. In the knowledge module options, leave the default values. The Property Inspector appears as shown in Figure 6–20.

Figure 6–20 Property Inspector for Target Area of Pop.TRG_CUSTOMER



The KM options enable to control certain aspects of the integration strategy. For example, the FLOW_CONTROL option triggers the flow control operations of the data before inserting it into the target table.

Note: Only Knowledge Modules imported to your Project appear in the KM Selector lists. The demonstration environment already includes the Knowledge Modules required for the getting started examples. You do not need to import KMs into the demonstration Project.

For more information on importing KMs into your Projects, see "Importing a KM" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

6.1.3.9 Define the Data Control Strategy

In Section 6.1.3.7, "Define the Data Loading Strategies (LKM)" and Section 6.1.3.8, "Define the Data Integration Strategies (IKM)" you have specified the data flow from the source to the target. You must now define how to check your data (CKM) and the constraints and rules that must be satisfied before integrating the data.

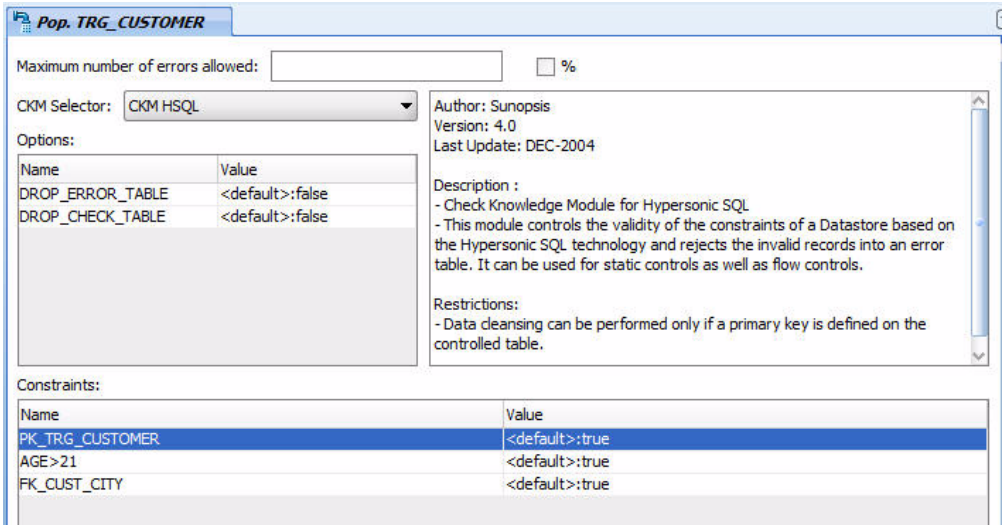
To define the data control strategy:

1. In the Controls tab of the Interface Editor, verify that the **CKM HSQL** is selected.
2. In the Constraints section, set the constraints that you wish to verify to `true`.

- PK_TRG_CUSTOMER
- AGE >21
- FK_CUST_CITY

The Controls tab appears as shown in Figure 6–21.

Figure 6–21 Controls Tab of Pop. TRG_CUSTOMER Interface



3. From **File** main menu, select **Save**.

The Pop. TRG_CUSTOMER interface is now ready to be executed.

6.2 Pop. TRG_SALES Interface Example

This section contains the following topics:

- Purpose and Integration Requirements
- Interface Definition
- Creating the Integration Interface

6.2.1 Purpose and Integration Requirements

This section describes the integration features and requirements the integration interface Pop. TRG_SALES is expected to meet.

The purpose of this interface is to load the SRC_ORDERS table of orders and the SRC_ORDER_LINES table of order lines from the *Orders Application - HSQL* model into the TRG_SALES target table in the *Sales Administration - HSQL* model. The data must be aggregated before it is integrated into the target table. Only orders whose status is `CLO` are to be used.

However, the source data is not always consistent with the integrity rules present in the target environment. For this transformation, we want to cleanse the data by verifying that all of the constraints are satisfied. We want to place any invalid rows into an error table rather that into our target database. In our case, two important integrity rules must be satisfied:

- The sales must be associated with a customer (CUST_ID) that exists in the TRG_CUSTOMER table (reference FK_SALES_CUST)
- The sales must be associated with a product (PRODUCT_ID) that exists in the TRG_PRODUCT table (reference FK_SALES_PROD)

The functional details for these rules and the procedure to follow are given in Section 6.2.3, "Creating the Integration Interface".

6.2.2 Interface Definition

This section describes the integration interface Pop. TRG_SALES that will be created in this example. See Section 6.2.3, "Creating the Integration Interface" for more information.

The Pop.TRG_SALES interface uses the following data and transformations:

- One target datastore. Table 6–7 lists the details of the target datastore.

Table 6–7 Target Datastore Details of Pop. TRG_SALES

Model	Datastore	Description	Type
Sales Administration - HSQL	TRG_SALES		HSQL table

- Two source datastores. Table 6–8 lists the details of the source datastores.

Table 6–8 Source Datastore Details of Pop. TRG_SALES

Model	Datastore	Description	Type
Orders Application - HSQL	SRC_ORDERS	Orders table in the source systems	HSQL table
Orders Application - HSQL	SRC_ORDER_LINES	Order lines table in the source system	HSQL table

- One join. Table 6–9 lists the details of the join.

Table 6–9 Joins used in Pop. TRG_SALES

Join	Description	SQL Rule	Execution Location
Commands and Order lines	Join SRC_ORDERS and SRC_ORDER_LINES	`SRC_ORDERS.ORDER_ID = SRC_ORDER_LINES.ORDER_ID`	Source

- One Filter. Table 6–10 lists the details of the filter.

Table 6–10 Filters used in Pop. TRG_SALES

Description	SQL Rule	Execution Location
JOnly retrieve completed orders (CLOSED)	`SRC_ORDERS.STATUS = 'CLO'`	Source

- Several transformation rules. Table 6–11 lists the details of the transformation rules.

Table 6–11 Transformation Rules used in Pop. TRG_CUSTOMER

Target Column	Origin	SQL Rule	Execution Location
CUST_ID	CUST_ID from SRC_ORDERS	SRC_ORDERS.CUST_ID	Source
PRODUCT_ID	PRODUCT_ID from SRC_ORDER_LINES	SRC_ORDER_LINES.PRODUCT_ID	Source
FIRST_ORD_ID	Smallest value of ORDER_ID	MIN(SRC_ORDERS.ORDER_ID)	Source
FIRST_ORD_DATE	Smallest value of the ORDER_DATE from SRC_ORDERS	MIN(SRC_ORDERS.ORDER_DATE)	Source
LAST_ORD_ID	Largest value of ORDER_ID	MAX(SRC_ORDERS.ORDER_ID)	Source
LAST_ORD_DATE	Largest value of the ORDER_DATE from SRC_ORDERS	MAX(SRC_ORDERS.ORDER_DATE)	Source
QTY	Sum of the QTY quantities from the order lines	SUM(SRC_ORDER_LINES.QTY)	Source
AMOUNT	Sum of the amounts from the order lines	SUM(SRC_ORDER_LINES.AMOUNT)	Source
PROD_AVG_PRICE	Average amount from the order lines	AVG(SRC_ORDER_LINES.AMOUNT)	Source

6.2.3 Creating the Integration Interface

This section describes how to create the Pop. TRG_SALES integration interface. To create the Pop. TRG_SALES interface perform the following procedure:

1. Insert a New Integration Interface
2. Define the Target Datastore
3. Define the Source Datastores
4. Define Joins between the Source Datastores
5. Define the Order Filter
6. Define the Transformation Rules
7. Define the Data Loading Strategies (LKM)
8. Define the Data Integration Strategies (IKM)
9. Define the Data Control Strategy

Note that you can also use the Quick-Edit Editor to create an integration interface. See "Using the Quick-Edit Editor" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for more information.

6.2.3.1 Insert a New Integration Interface

To create a new integration interface:

1. In Designer Navigator, expand the Demo project node in the Projects accordion.
2. Expand the Sales Administration node.

3. In the Sales Administration folder, right-click the Interfaces node and select **New Interface**.

   The Interface Editor is displayed.

4. On the Definition tab of the Interface Editor, enter the name of your interface (Pop. TRG_SALES) in the Name field.

6.2.3.2 Define the Target Datastore

To insert the target datastore in the Pop. TRG_SALES interface:

1. Go to the Mapping tab of the Interface Editor.
2. In the Designer Navigator, expand the Models accordion and the *Sales Administration - HSQL* model.
3. Select the TRG_SALES datastore under the *Sales Administration - HSQL model* and drag it into the Target Datastore panel.

6.2.3.3 Define the Source Datastores

The Pop. TRG_SALES interface example uses datastores from the *Orders Application - HSQL* model.

To add source datastores to the Pop. TRG_SALES interface:

1. In the Mapping tab, drag the following source datastores into the Source Diagram:
   - SRC_ORDERS from the *Orders Application - HSQL* model
   - SRC_ORDER_LINES from the *Orders Application - HSQL* model
2. In the Automap Dialog click **Yes**.

6.2.3.4 Define Joins between the Source Datastores

This section describes how to define joins between the source datastores.

To create the join defined in Table 6–9:

1. In the Source Diagram, select the ORDER_ID column of the SRC_ORDERS datastore.
2. Drag and drop it on the ORDER_ID column of the SRC_ORDER_LINES datastore.

   A join linking the two datastores appears. This is the join on the order number.

   The join has the following expression:

   ```
   SRC_ORDERS.ORDER_ID=SRC_ORDER_LINES.ORDER_ID
   ```

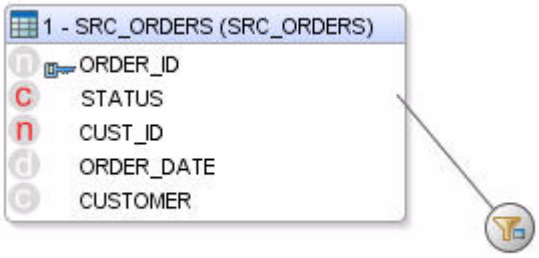
6.2.3.5 Define the Order Filter

In this example, only completed orders should be retrieved. A filter needs to be defined on the SRC_ORDERS datastore.

To define the filter:

1. In the Source Diagram, select the STATUS column of the SRC_ORDERS datastore and drag it onto the Source Diagram.
2. The filter appears as shown in Figure 6–22.

Figure 6–22 Filter on SRC_ORDERS



3. Select the filter in the Source Diagram to display the filter properties in the Property Inspector.
4. In the Implementation tab of the Property Inspector, modify the filter rule by typing:

```
SRC_ORDERS.STATUS = 'CLO'
```

6.2.3.6 Define the Transformation Rules

In the Target Datastore panel of your interface, columns with names that match their sources are automatically mapped. The automatic mapping is done by the matching of the column names. Most of the transformation rules listed in Table 6–11 have been defined by the automatic mapping. In addition to this automatic mappings, you have to define the transformation rules for the following fields: FIRST_ORD_ID, FIRST_ORD_DATE, LAST_ORD_ID, LAST_ORD_DATE, QTY, AMOUNT, and PROD_AVG_PRICE.

To manually define a mapping of the target column:

1. In the Target Datastore Panel, select the target column to display the mapping properties in the Property Inspector.
2. In the Property Inspector, click **Freeze View** so that the mapping of the target column is displayed.
3. Create the transformation rule either by:
   - Dragging the source column into the Mapping field in the Target Datastore panel
   - Dragging the required columns from the source datastores into the Implementation field in the Property Inspector
   - Editing the mapping expression in the Expression Editor
4. Select the execution location: **Source**, **Target** or **Staging Area**. Make sure that you select the environment in which your transformation will be executed as specified in Table 6–11.
5. Validate the syntax by clicking **Check the expression in the DBMS**.
6. Save the expression by clicking **Save your expression**.

Implement the following rules in the mappings:

- **FIRST_ORD_ID**: Drag the SRC_ORDERS.ORDER_ID column into the Implementation field. Enter the following text in the Implementation field:

```
MIN(SRC_ORDERS.ORDER_ID)
```

This transformation rule maps the minimum value of the ORDER_ID column in your SRC_ORDERS table to the FIRST_ORD_ID column in your target table.

- **FIRST_ORD_DATE**: Drag the SRC_ORDERS.ORDER_DATE column into the Implementation field. Enter the following text in the Implementation field:

  ```
  MIN(SRC_ORDERS.ORDER_DATE)
  ```

  This transformation rule maps the minimum value of the ORDER_DATE column in your SRC_ORDERS table to the FIRST_ORD_DATE column in your target table.

- **LAST_ORD_ID**: Drag-and-drop the SRC_ORDERS.ORDER_ID column into the Implementation field. Enter the following text in the Implementation field:

  ```
  MAX(SRC_ORDERS.ORDER_ID)
  ```

  This transformation rule maps the maximum value of the ORDER_ID column in your SRC_ORDERS table to the LAST_ORD_ID column in your target table.

- **LAST_ORD_DATE**: Drag the SRC_ORDERS.ORDER_DATE column into the Implementation field. Enter the following text in the Implementation field:

  ```
  MAX(SRC_ORDERS.ORDER_DATE)
  ```

  This transformation rule maps the maximum value of the ORDER_DATE column in your SRC_ORDERS table to the LAST_ORD_DATE column in your target table.

- **QTY**: Enter the following text in the Implementation field:

  ```
  SUM(SRC_ORDER_LINES.QTY)
  ```

  This transformation rule maps the sum of the product quantities to the QTY column in your target table.

- **AMOUNT**: Enter the following text in the Implementation field:

  ```
  SUM(SRC_ORDER_LINES.AMOUNT)
  ```

  This transformation rule maps the sum of the product prices to the AMOUNT column in your target table.

- **PROD_AVG_PRICE**: Drag the SRC_ORDERLINES.AMOUNT column into the Implementation field. Enter the following text in the Implementation field:

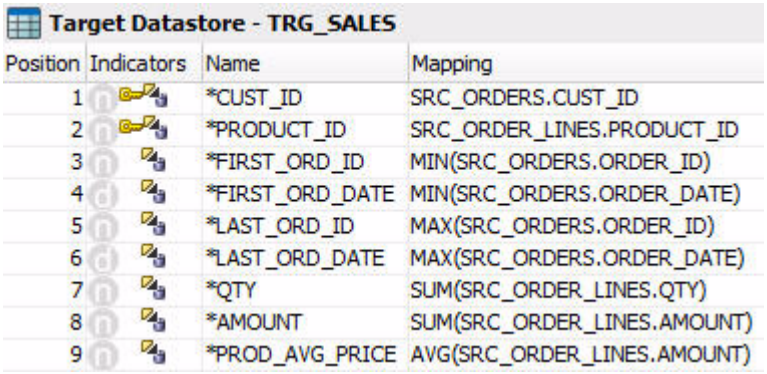
  ```
  AVG(SRC_ORDER_LINES.AMOUNT)
  ```

  This transformation rule maps the average of the product prices to the PROD_AVG_PRICE column in your target table.

Review carefully your mapping rules and make sure that you have defined the rules as shown in Figure 6–23.

Note that even though this example uses aggregation functions, you do not have to specify the group by rules: Oracle Data Integrator will infer that from the mappings, applying SQL standard coding practices.

Figure 6–23 Target Datastore Mappings



Target Datastore - TRG_SALES

Position	Indicators	Name	Mapping
1		*CUST_ID	SRC_ORDERS.CUST_ID
2		*PRODUCT_ID	SRC_ORDER_LINES.PRODUCT_ID
3		*FIRST_ORD_ID	MIN(SRC_ORDERS.ORDER_ID)
4		*FIRST_ORD_DATE	MIN(SRC_ORDERS.ORDER_DATE)
5		*LAST_ORD_ID	MAX(SRC_ORDERS.ORDER_ID)
6		*LAST_ORD_DATE	MAX(SRC_ORDERS.ORDER_DATE)
7		*QTY	SUM(SRC_ORDER_LINES.QTY)
8		*AMOUNT	SUM(SRC_ORDER_LINES.AMOUNT)
9		*PROD_AVG_PRICE	AVG(SRC_ORDER_LINES.AMOUNT)

6.2.3.7 Define the Data Loading Strategies (LKM)

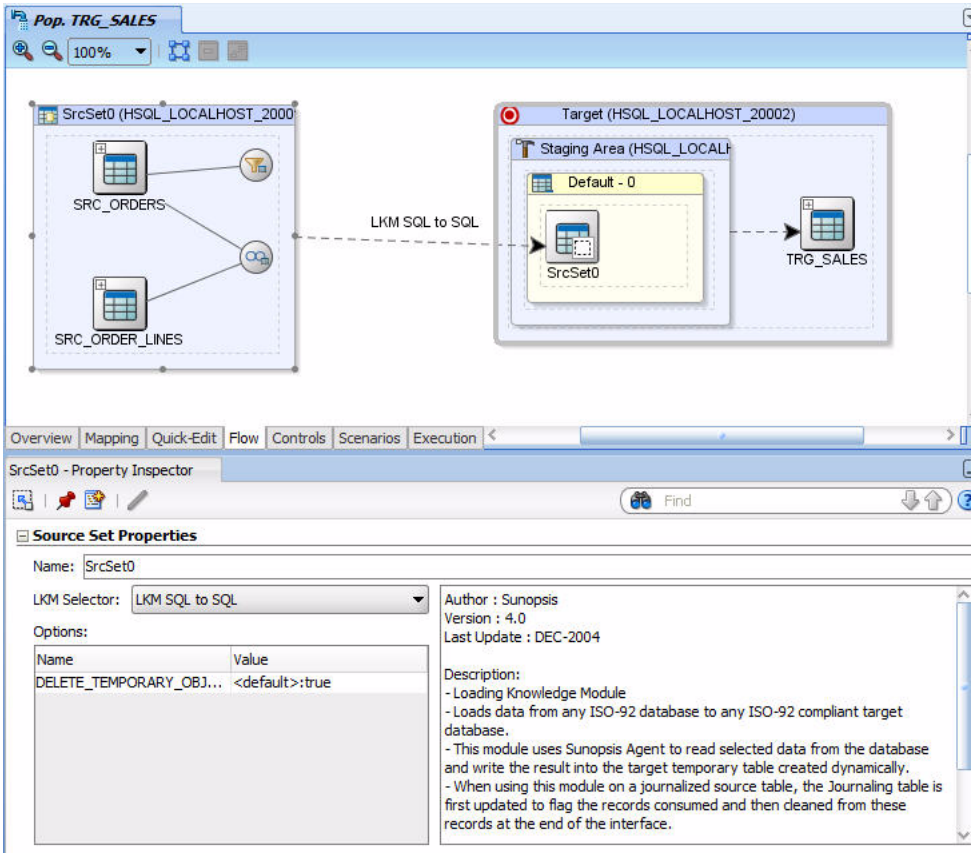
In the Flow tab, Oracle Data Integrator indicates the various steps that are performed when the interface is executed.

In the Flow tab you define how to load the result of the orders and order line aggregates into your target environment with a Loading Knowledge Module (LKM).

To define the loading strategies:

1. In the Flow tab of the Interface Editor, select the source set that corresponds to the loading of the order line's filtered aggregate results. In this example, this is the SrcSet0 (HSQL_LOCALHOST_2000).
2. In the Property Inspector, verify that the **LKM SQL to SQL** is selected in the LKM Selector as shown in Figure 6–24.

Figure 6–24 Flow tab of Pop.TRG_SALES Interface



6.2.3.8 Define the Data Integration Strategies (IKM)

After defining the loading phase, you need to define the strategy to adopt for the integration of the data into the target table.

To define the integration strategies:

1. In the Flow tab of the Interface Editor, select the Target object in the Flow Diagram. The Property Inspector should display the properties of the target.
2. In the Property Inspector, verify that the **IKM SQL Incremental Update** is selected in the IKM Selector list.
3. In the knowledge module options, leave the default values.

6.2.3.9 Define the Data Control Strategy

In Section 6.2.3.7, "Define the Data Loading Strategies (LKM)" and Section 6.2.3.8, "Define the Data Integration Strategies (IKM)" you have specified the data flow from the source to the target. You must now define how to check your data (CKM) and the constraints and rules that must be satisfied before integrating the data.

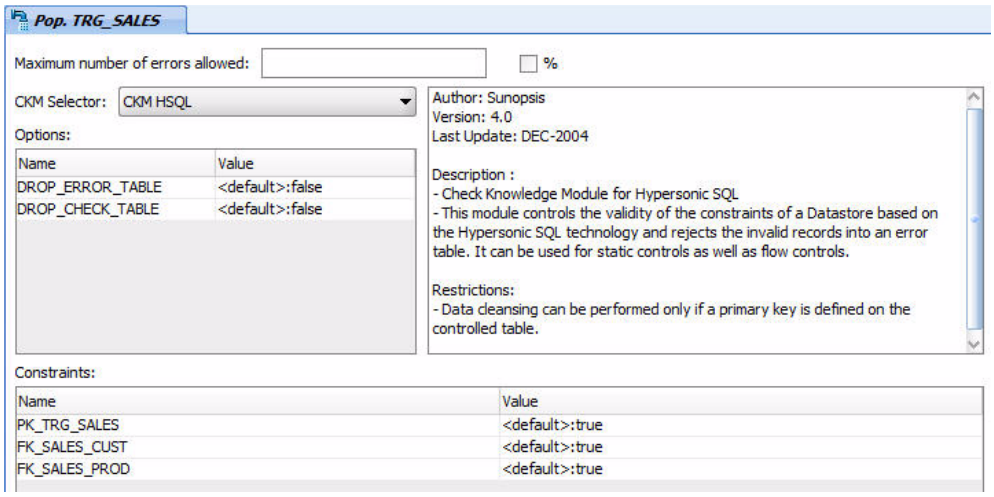
To define the data control strategy:

1. In the Controls tab of the Interface Editor, verify that the **CKM HSQL** is selected.
2. In the Constraints section, set the constraints that you wish to verify to `true`:
   - PK_TRG_SALES

- FK_SALES_CUST
- FK_SALES_PROD

The Controls tab appears as shown in Figure 6–25.

Figure 6–25 Controls tab of Pop.TRG_SALES Interface



3. From **File** main menu, select **Save**.

The Pop. TRG_SALES interface is now ready to be executed.

7

Working with Packages

This chapter describes how to work with Packages in Oracle Data Integrator. The *Load Sales Administration* package is used as an example. An introduction to Packages and automating data integration between applications is provided.

This chapter includes the following sections:

7.1 Introduction

This section provides an introduction to automating data integration using packages in Oracle Data Integrator.

7.1.1 Automating Data Integration Flows

The automation of the data integration flows is achieved by sequencing the execution of the different steps (interfaces, procedures, and so forth) in a package and by producing a production scenario containing the ready-to-use code for each of these steps.

This chapter describes how to sequence the execution of the different steps. How to produce the production scenario is covered in Chapter 9, "Deploying Integrated Applications".

7.1.2 Packages

A *Package* is made up of a sequence of steps organized into an execution diagram. Packages are the main objects used to generate scenarios for production. They represent the data integration workflow and can perform, for example, the following jobs:

- Start a reverse-engineering process on a datastore or a model
- Send an email to an administrator
- Download a file and unzip it
- Define the order in which interfaces must be executed
- Define loops to iterate over execution commands with changing parameters

In this Getting Started exercise, you will load your *Sales Administration* application using a sequence of interfaces. Since referential constraints exist between tables of this application, you must load target tables in a predefined order. For example, you

cannot load the TRG_CUSTOMER table if the TRG_CITY table has not been loaded first.

In the Section 7.2, "Load Sales Administration Package Example", you will create and run a package that includes interfaces that are included in the Demo project and interfaces that you've created in Chapter 6, "Working with Integration Interfaces".

7.1.2.1 Scenarios

A *scenario* is designed to put a source component (interface, package, procedure, variable) into production. A scenario results from the generation of code (SQL, shell, and so forth) for this component.

Once generated, the code of the source component is frozen and the scenario is stored inside the Work repository. A scenario can be exported and then imported into different production environments.

Note: Once generated, the scenario's code is frozen, and all subsequent modifications of the package and/or data models which contributed to its creation will not affect it. If you want to update a scenario - for example because one of its interfaces has been changed - then you must generate a new version of the scenario from the package.

See "Working with Scenarios" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for more information.

In Chapter 9, "Deploying Integrated Applications", you will generate the *LOAD_SALES_ADMINISTRATION* scenario from a package and run this scenario from Oracle Data Integrator Studio.

7.2 Load Sales Administration Package Example

This section contains the following topics:

- Purpose
- Developments Provided with Oracle Data Integrator
- Problem Analysis
- Creating the Package

7.2.1 Purpose

The purpose of the Load Sales Administration package is to define the complete workflow for the loading of the *Sales Administration* application and to set the execution sequence.

7.2.2 Developments Provided with Oracle Data Integrator

The demo repository is delivered with a number of developments. The Demo project now contains the following developments as shown in Figure 7–1:

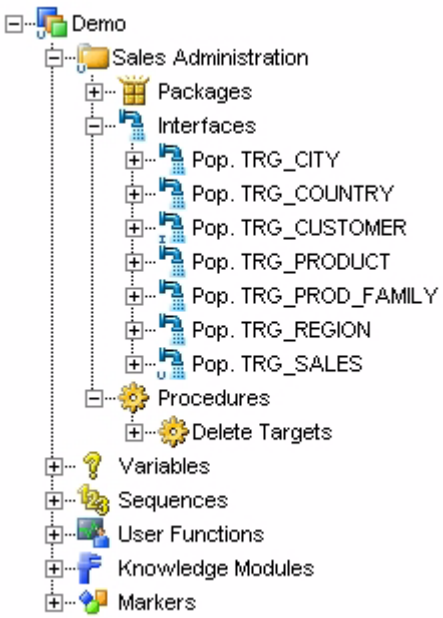
- Seven interfaces:
  - **Pop. TRG_CITY**: an interface that populates the TRG_CITY table. This interface is delivered with the demo repository.

- **Pop. TRG_COUNTRY**: an interface that populates the TRG_COUNTRY table. This interface is delivered with the demo repository.
- **Pop. TRG_CUSTOMER**: an interface that populates the TRG_CUSTOMER table. This interface is created in Section 6.1, "Pop. TRG_CUSTOMER Interface Example".
- **Pop. TRG_PRODUCT**: an interface populates the TRG_PRODUCT table. This interface is delivered with the demo repository.
- **Pop. TRG_PROD_FAMILY**: an interface that populates the TRG_PROD_FAMILY table. This interface is delivered with the demo repository.
- **Pop. TRG_REGION**: an interface that populates the TRG_REGION table. This interface is delivered with the demo repository.
- **Pop. TRG_SALES**: an interface that populates the TRG_SALES table. This interface is created in Section 6.2, "Pop. TRG_SALES Interface Example".

- One procedure:

  The **Delete Targets** procedure empties all of the tables in the *Sales Administration* application. This operation is performed by using a *Delete* statement on each table.

Figure 7–1 Demo Project



7.2.3 Problem Analysis

In order to load the *Sales Administration* application correctly (in accordance with the referential integrity constraints), the tasks must be executed in the following order:

1. Empty the Sales Administration tables with the Delete Targets procedure
2. Load the TRG_COUNTRY table with the Pop. TRG_COUNTRY interface
3. Load the TRG_REGION table with the Pop. TRG_REGION interface
4. Load the TRG_CITY table with the Pop. TRG_CITY interface

5. Load the TRG_PROD_FAMILY table with the Pop. TRG_PROD_FAMILY interface
6. Load the TRG_PRODUCT table with the Pop. TRG_PRODUCT interface
7. Load the TRG_CUSTOMER table with the Pop. TRG_CUSTOMER interface
8. Load the TRG_SALES table with the Pop. TRG_SALES interface

Such an integration process is built in Oracle Data Integrator in the form of a Package.

7.2.4 Creating the Package

This section describes how to create the Load Sales Administration Package. To create the Load Sales Administration Package perform the following procedure:

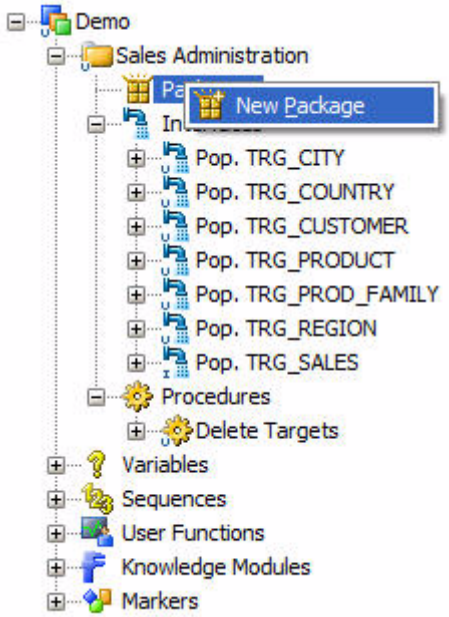
1. Create a New Package
2. Insert the Steps in the Package
3. Define the Sequence of Steps in the Package

7.2.4.1 Create a New Package

To create a new Package:

1. In Designer Navigator, expand the Demo project node in the Projects accordion.
2. Expand the Sales Administration node.
3. In the Sales Administration folder, right-click the Packages node and select **New Package** as shown in Figure 7–2.

Figure 7–2 Insert New Package



The Package Editor is displayed.

4. On the Definition tab of the Package Editor, enter the name of your Package (Load Sales Administration) in the Name field.
5. From the File main menu, select **Save**.

7.2.4.2 Insert the Steps in the Package

To insert the steps in the Load Sales Administration Package:

1. In the Package Editor, go the Diagram tab.
2. In the Diagram tab, select the following components one by one from the Projects accordion and drag-and-drop them into the diagram:
   - Delete Targets (Procedure)
   - Pop. TRG_COUNTRY
   - Pop. TRG_REGION
   - Pop. TRG_CITY
   - Pop. TRG_CUSTOMER
   - Pop. TRG_PROD_FAMILY
   - Pop. TRG_PRODUCT
   - Pop. TRG_SALES

   These components are inserted in the Package and appear as steps in the diagram. Note that the steps are not sequenced yet.

7.2.4.3 Define the Sequence of Steps in the Package

Once the steps are created, you must reorder them into a data processing chain. This chain has the following rules:

- It starts with a unique step defined as the *First Step*.
- Each step has two termination states: Success or Failure.
- A step in failure or success can be followed by another step, or by the end of the Package.
- In case of failure, it is possible to define a number of retries.

A Package has one entry point, the First Step, but several possible termination steps.

The Load Sales Administration Package contains only steps on Success.

Defining the First Step

To define the first step in the Load Sales Administration Package:

> **Note:** If you have dragged and dropped the Package components in the order defined in Section 7.2.4.2, "Insert the Steps in the Package", the Delete Target procedure is already identified as the first step and the first step symbol is displayed on the step's icon. If this is the case, define the next steps on success.

1. Select and right-click the *Delete Target* procedure step.
2. Select **First Step** from the contextual menu.

A small green arrow appears on this step.

Defining the Next Steps on Success

To define the next steps on success:

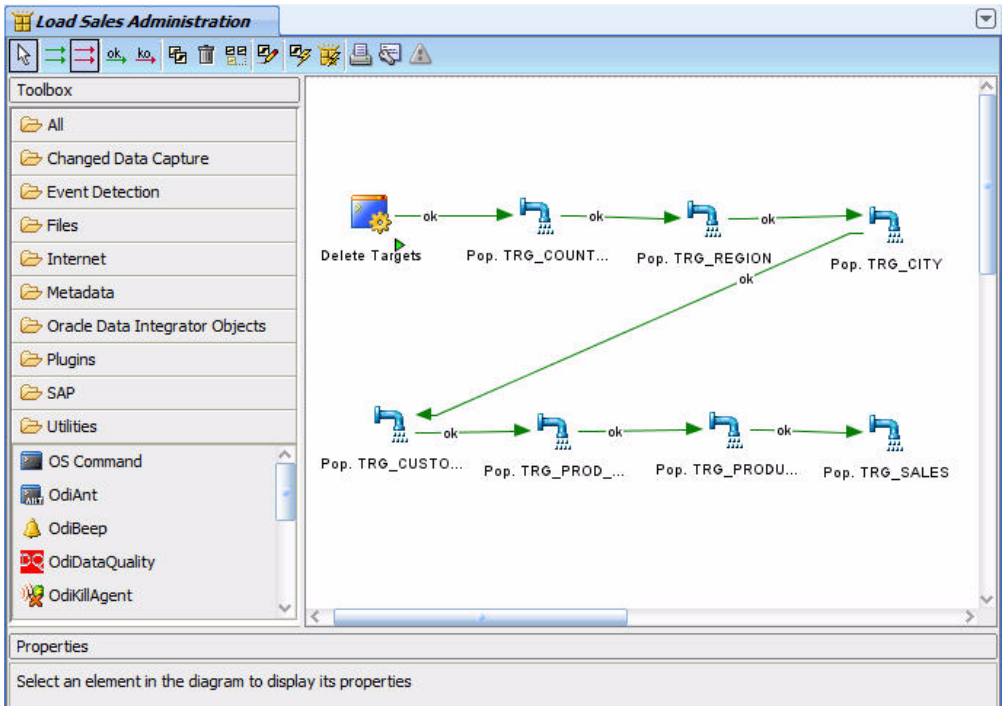
1. In the Package toolbar tab, select **Next Step on Success**.

2. Select the Delete Targets step.
3. Keep the mouse button pressed and move the cursor to the icon of the step that must follow in case of a success (here the Pop. TRG_COUNTRY step) and release the mouse button.

   A green arrow representing the success path between the steps, with an ok label on it appears.
4. Repeat this operation to link all your steps in a success path sequence. This sequence should be:
   - Delete Targets (First Step)
   - Pop. TRG_COUNTRY
   - Pop. TRG_REGION
   - Pop. TRG_CITY
   - Pop. TRG_CUSTOMER
   - Pop. TRG_PROD_FAMILY
   - Pop. TRG_PRODUCT
   - Pop. TRG_SALES

   The resulting sequence appears in the Package diagram as shown in Figure 7–3.

Figure 7–3 Load Sales Administration Package Diagram



5. From the File main menu, select **Save**.

The package is now ready to be executed.

8

Executing Your Developments and Reviewing the Results

This chapter describes how to execute the Load Sales Administration Package you have created in Chapter 7, "Working with Packages" and the intergration interfaces Pop. TRG_CUSTOMER and Pop. TRG_SALES you have created in Chapter 6, "Working with Integration Interfaces". This chapter also describes how to follow the execution and how to interpret the execution results.

This chapter includes the following sections:

8.1 Executing the Load Sales Administration Package

This section contains the following topics:

8.1.1 Run the Package

To run the Load Sales Administration Package:

1. In Designer Navigator, expand the Packages node under the Sales Administration node.
2. Select the Load Sales Administration Package.
3. Right-click and select **Execute**.
4. In the Confirm Dialog click **Yes**.
5. In the Execution Dialog, leave the default settings and click **OK**.
6. The Session Started Information Dialog is displayed. Click **OK**.

Oracle Data Integrator now starts an execution session.

8.1.2 Follow the Execution of the Package in Operator Navigator

Through Operator Navigator, you can view your execution results and manage your development executions in the sessions.

To view the execution results of the Load Sales Administration Package:

1. In the Session List accordion in Operator Navigator, expand the All Executions node.
2. Refresh the displayed information by clicking **Refresh** in the Operator Navigator toolbar. The Refresh button is:

3. The log for the execution session of the Load Sales Administration Package appears as shown in Figure 8–1.

Figure 8–1 Load Sales Administration Package Session Log

8.1.3 Interpreting the Results of the Pop. TRG_CUSTOMER Session Step

This section describes how to determine the invalid records detected by the Pop. TRG_CUSTOMER interface. These are the records that do not satisfy the constraints and have been rejected by the flow control of the Pop. TRG_CUSTOMER interface.

This section includes the following topics:

- Determining the Number of Processed Records
- Viewing the Resulting Data
- Reviewing the Invalid Records and Incorrect Data
- Correcting Invalid Data
- Review the Processed Records

8.1.3.1 Determining the Number of Processed Records

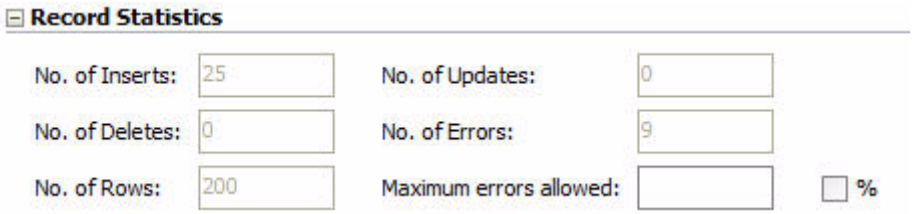
To determine the number of records that have been processed by the Pop. TRG_CUSTOMER interface (this is the number of inserts, updates, deletes, and errors):

1. In the Session List accordion in Operator Navigator, expand the All Executions node.
2. Refresh the displayed information clicking **Refresh** in the Operator Navigator toolbar menu.
3. Expand the Load Sales Administration Package Session and open the Session Step Editor for the Pop. TRG_CUSTOMER step. This is step 4.

4. On the Definition tab of the Session Step Editor, you can see in the Record Statistics section that the loading of the TRG_CUSTOMER table produced 25 inserts and isolated 9 errors in an error table.

   Figure 8–2 shows the Record Statistics section of the Session Step Editor:

Figure 8–2 Record Statistics in the Session Step Editor



8.1.3.2 Viewing the Resulting Data

In this example, the resulting data are the 25 rows that have been inserted in the TRG_CUSTOMER table during the interface execution.

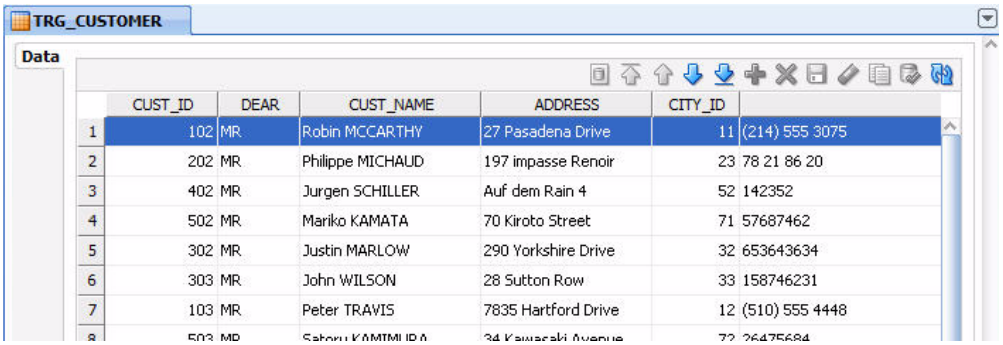
To view the data resulting of your interface execution:

1. In Designer Navigator, expand the Models accordion and the *Sales Administration - HSQL* model.
2. Select the TRG_CUSTOMER datastore.
3. Right-click and select **View Data** to view the data in the target table.

   Note that you can also select **Data...** to view and edit the data of the target table.

   The View Data Editor is displayed as shown in Figure 8–3.

Figure 8–3 View Data Editor



8.1.3.3 Reviewing the Invalid Records and Incorrect Data

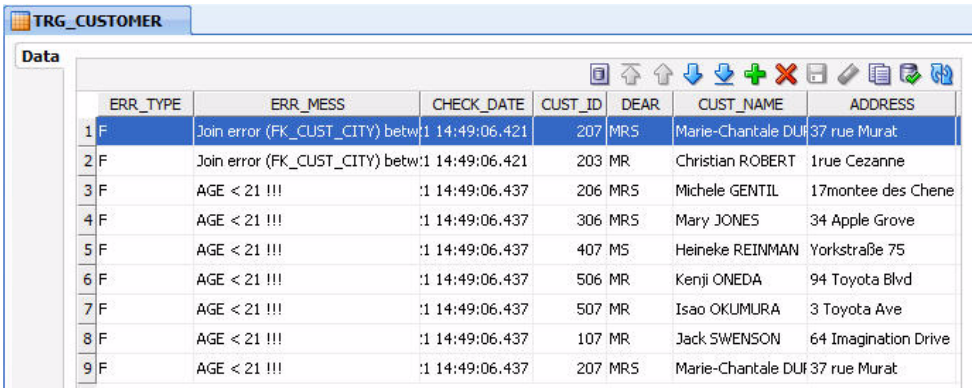
You can access the invalid records by right-clicking on the datastore in your model and selecting **Control** > **Errors...**

To review the error table of the TRG_CUSTOMER datastore:

1. In Designer Navigator, expand the *Sales Administration - HSQL* model.
2. Select the TRG_CUSTOMER datastore.
3. Right-click and select **Control** > **Errors...**

4. The Error Table Editor is displayed as shown in Figure 8–4.

Figure 8–4 Error Table of TRG_CUSTOMER



	ERR_TYPE	ERR_MESS	CHECK_DATE	CUST_ID	DEAR	CUST_NAME	ADDRESS
1	F	Join error (FK_CUST_CITY) betw	1 14:49:06.421	207	MRS	Marie-Chantale DUI	37 rue Murat
2	F	Join error (FK_CUST_CITY) betw	1 14:49:06.421	203	MR	Christian ROBERT	1rue Cezanne
3	F	AGE < 21 !!!	1 14:49:06.437	206	MRS	Michele GENTIL	17montee des Chene
4	F	AGE < 21 !!!	1 14:49:06.437	306	MRS	Mary JONES	34 Apple Grove
5	F	AGE < 21 !!!	1 14:49:06.437	407	MS	Heineke REINMAN	Yorkstraße 75
6	F	AGE < 21 !!!	1 14:49:06.437	506	MR	Kenji ONEDA	94 Toyota Blvd
7	F	AGE < 21 !!!	1 14:49:06.437	507	MR	Isao OKUMURA	3 Toyota Ave
8	F	AGE < 21 !!!	1 14:49:06.437	107	MR	Jack SWENSON	64 Imagination Drive
9	F	AGE < 21 !!!	1 14:49:06.437	207	MRS	Marie-Chantale DUI	37 rue Murat

The interface that you have executed has identified and isolated **9** invalid records in an error table that was automatically created for you.

In this error table, you can see that the interface rejected:

- **2** records that did not satisfy the FK_CUST_CITY constraint (for example, the CITY_ID value does not exist in the table of cities SRC_CITY table).
- **7** records that did not satisfy the business rule acting on customers under 21 (AGE > 21 constraint).

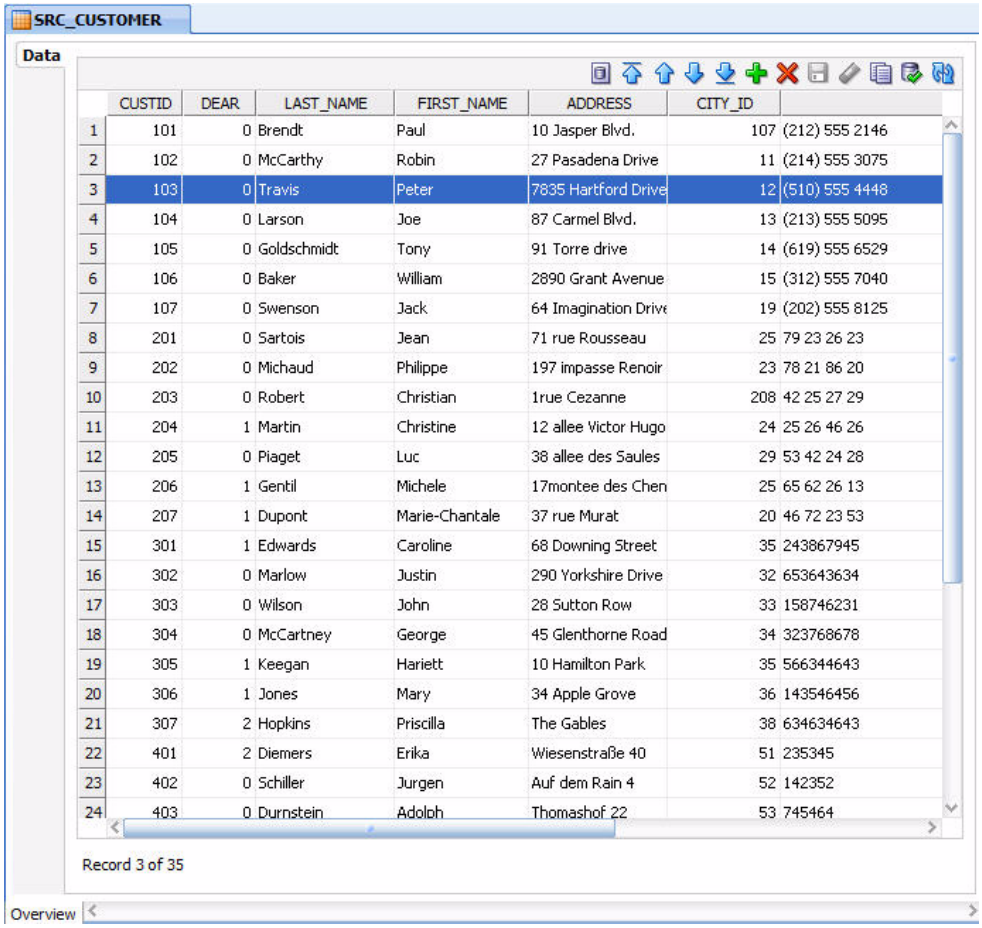
  The invalid records were saved into an error table and were not integrated into the target table.

8.1.3.4 Correcting Invalid Data

To rectify invalid data:

1. In Designer Navigator, expand the *Orders Application - HSQL* model.
2. Select the SRC_CUSTOMER datastore.
3. Right-click and select **Data**.
4. The Data Editor is displayed as shown in Figure 8–5.

Figure 8–5 Data Editor



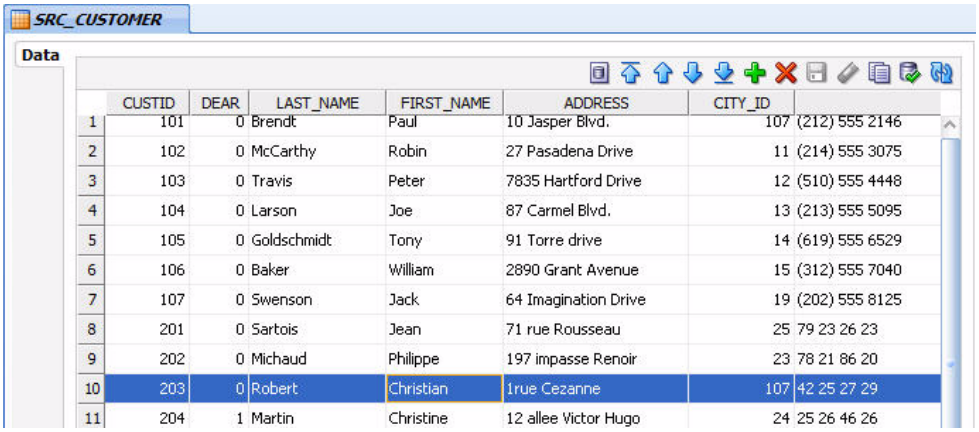
	CUSTID	DEAR	LAST_NAME	FIRST_NAME	ADDRESS	CITY_ID	
1	101	0	Brendt	Paul	10 Jasper Blvd.	107	(212) 555 2146
2	102	0	McCarthy	Robin	27 Pasadena Drive	11	(214) 555 3075
3	103	0	Travis	Peter	7835 Hartford Drive	12	(510) 555 4448
4	104	0	Larson	Joe	87 Carmel Blvd.	13	(213) 555 5095
5	105	0	Goldschmidt	Tony	91 Torre drive	14	(619) 555 6529
6	106	0	Baker	William	2890 Grant Avenue	15	(312) 555 7040
7	107	0	Swenson	Jack	64 Imagination Drive	19	(202) 555 8125
8	201	0	Sartois	Jean	71 rue Rousseau	25	79 23 26 23
9	202	0	Michaud	Philippe	197 impasse Renoir	23	78 21 86 20
10	203	0	Robert	Christian	1rue Cezanne	208	42 25 27 29
11	204	1	Martin	Christine	12 allee Victor Hugo	24	25 26 46 26
12	205	0	Piaget	Luc	38 allee des Saules	29	53 42 24 28
13	206	1	Gentil	Michele	17montee des Chen	25	65 62 26 13
14	207	1	Dupont	Marie-Chantale	37 rue Murat	20	46 72 23 53
15	301	1	Edwards	Caroline	68 Downing Street	35	243867945
16	302	0	Marlow	Justin	290 Yorkshire Drive	32	653643634
17	303	0	Wilson	John	28 Sutton Row	33	158746231
18	304	0	McCartney	George	45 Glenthorne Road	34	323768678
19	305	1	Keegan	Hariett	10 Hamilton Park	35	566344643
20	306	1	Jones	Mary	34 Apple Grove	36	143546456
21	307	2	Hopkins	Priscilla	The Gables	38	634634643
22	401	2	Diemers	Erika	Wiesenstraße 40	51	235345
23	402	0	Schiller	Jurgen	Auf dem Rain 4	52	142352
24	403	0	Durnstein	Adolph	Thomashof 22	53	745464

In the Data table, search for the client row having a CUSTID equal to `203`.

Note that you can sort the table by clicking on the column headers. If the customer 203 is not visible, click **Refresh data** in the menu toolbar to refresh the display.

5. The CITY_ID value of this customer is `208`. This CITY_ID is not listed in the SRC_CITY table. Double-click on the value of the CITY_ID column for this customer in order to modify it. Enter `107` in the CITY_ID field.
6. Press **Enter** to validate your entry.
7. The Data Editor is displayed as shown in Figure 8–6.

Figure 8–6 Data Editor with new CITY_ID value



	CUSTID	DEAR	LAST_NAME	FIRST_NAME	ADDRESS	CITY_ID	
1	101	0	Brendt	Paul	10 Jasper Blvd.	107	(212) 555 2146
2	102	0	McCarthy	Robin	27 Pasadena Drive	11	(214) 555 3075
3	103	0	Travis	Peter	7835 Hartford Drive	12	(510) 555 4448
4	104	0	Larson	Joe	87 Carmel Blvd.	13	(213) 555 5095
5	105	0	Goldschmidt	Tony	91 Torre drive	14	(619) 555 6529
6	106	0	Baker	William	2890 Grant Avenue	15	(312) 555 7040
7	107	0	Swenson	Jack	64 Imagination Drive	19	(202) 555 8125
8	201	0	Sartois	Jean	71 rue Rousseau	25	79 23 26 23
9	202	0	Michaud	Philippe	197 impasse Renoir	23	78 21 86 20
10	203	0	Robert	Christian	1rue Cezanne	107	42 25 27 29
11	204	1	Martin	Christine	12 allee Victor Hugo	24	25 26 46 26

8. In the menu toolbar, click **Post changes to current row**.
9. In the Projects accordion, select the Pop. TRG_CUSTOMER interface in the *Sales Administration* model.
10. Right-click and select **Execute**. This executes only the Pop. TRG_CUSTOMER interface.
11. In the Execution Dialog and in the Information Dialog click **OK**.

The Pop. TRG_CUSTOMER interface is executed.

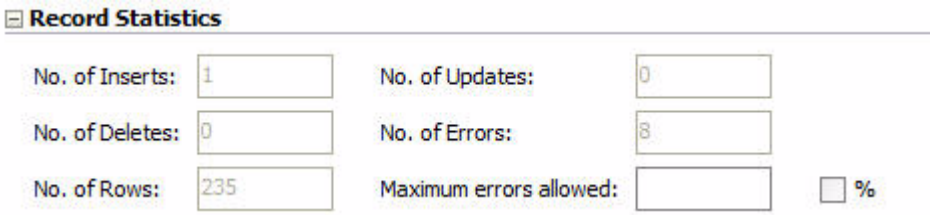
8.1.3.5 Review the Processed Records

To review the processed records:

1. In Operator Navigator, open the Session Step Editor for the Pop. TRG_CUSTOMER step.
2. If required, click **Refresh** in the Operator Navigator menu toolbar.
3. On the Definition tab of the Session Step Editor, you can see in the Record Statistics section that the loading of the TRG_CUSTOMER table produced 1 insertion (this is the record that you have modified in Section 8.1.3.4, "Correcting Invalid Data") and isolated 8 errors in an error table.

   Figure 8–8 shows the Record Statistics section of the Session Step Editor.

Figure 8–7 Record Statistics of the Session Step Editor



8.2 Executing the Pop. TRG_SALES Interface

This section contains the following topics:

- Execute the Integration Interface
- Follow the Execution of the Interface in Operator Navigator
- Interpreting the Results

8.2.1 Execute the Integration Interface

The Pop. TRG_SALES integration interface has already been executed by the Load Sales Administration package in Section 8.1.1, "Run the Package". This section describes how to execute only the Pop. TRG_SALES interface.

To run the Pop. TRG_SALES integration interface:

1. In Designer Navigator, expand the Interfaces node under the Sales Administration node.
2. Select the Pop. TRG_SALES Interface.
3. Right-click and select **Execute**.
4. In the Confirm Dialog click **Yes**. The interface is saved and the Execution Dialog is displayed.
5. In the Execution Dialog, leave the default settings and click **OK**.
6. The Session Started Information Dialog is displayed. Click **OK**.

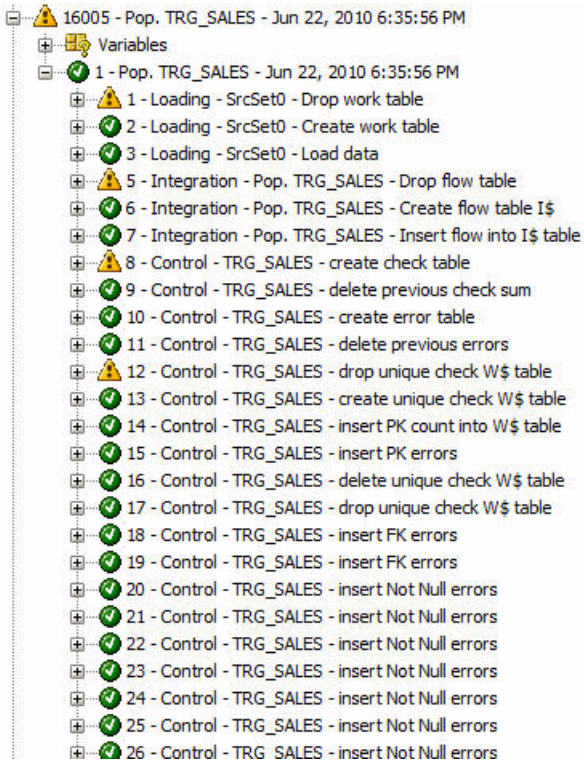
Oracle Data Integrator now starts an execution session.

8.2.2 Follow the Execution of the Interface in Operator Navigator

To view the execution results of your integration interface:

1. In the Session List accordion in Operator Navigator, expand the All Executions node.
2. Refresh the displayed information clicking **Refresh** in the Operator Navigator toolbar menu.
3. The log for the execution session of the Pop. TRG_SALES interface appears as shown in Figure 8–8.

Figure 8–8 Pop. TRG_SALES Interface Session Log



8.2.3 Interpreting the Results

This section describes how to determine the invalid records. These are the records that do not satisfy the constraints and has been rejected by the flow control.

This section includes the following topics:

- Determining the Number of Processed Records
- Viewing the Resulting Data
- Reviewing the Invalid Records and Incorrect Data

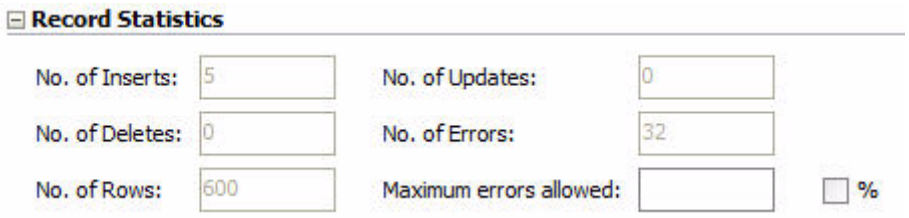
8.2.3.1 Determining the Number of Processed Records

To determine the number of processed records:

1. In Operator Navigator, open the Session Step Editor for the Pop. TRG_SALES step.
2. If required, click **Refresh** in the Operator Navigator menu toolbar.
3. On the Definition tab of the Session Step Editor, you can see in the Record Statistics section that the loading of the TRG_SALES table produced 5 inserts and isolated 32 errors in an error table.

   Figure 8–9 shows the Record Statistics section of the Session Step Editor:

Figure 8–9 Record Statistics in the Session Step Editor

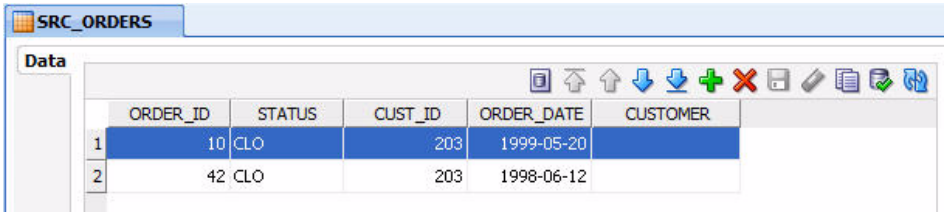


These 5 inserts are the 5 rows that have been inserted because of the changes you have performed in Section 8.1.3.4, "Correcting Invalid Data". Changing the CITY_ID of the customer with the CUST_ID = 203 to a CITY_ID that is listed in SRC_CITY table, adds the sales performed by the customer 203 to the TRG_SALES table. These 5 sales operations are highlighted in Figure 8–12.

Note that the customer with the CUST_ID = 203 actually performed 7 sales operations. You can identify these 7 operations as follows:

1. In the SRC_CUSTOMER table, determine the ORDER_IDs of the customer with the CUST_ID = 203. Figure 8–10 shows that this customer has two ORDER_IDs: 10 and 42.

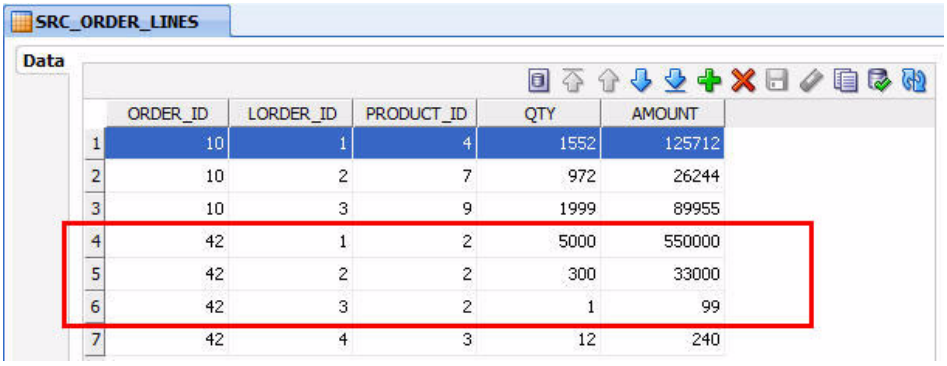
Figure 8–10 ORDER_IDs corresponding to CUST_ID = 203



	ORDER_ID	STATUS	CUST_ID	ORDER_DATE	CUSTOMER
1	10	CLO	203	1999-05-20	
2	42	CLO	203	1998-06-12	

2. As shown in Figure 8–11, the SRC_ORDER_LINES table lists 7 order lines for the ORDER_IDs 10 and 42.

Figure 8–11 ORDER_LINES for ORDER_IDs 10 and 42



	ORDER_ID	LORDER_ID	PRODUCT_ID	QTY	AMOUNT
1	10	1	4	1552	125712
2	10	2	7	972	26244
3	10	3	9	1999	89955
4	42	1	2	5000	550000
5	42	2	2	300	33000
6	42	3	2	1	99
7	42	4	3	12	240

The highlighted order lines have the same PRODCT_ID and are merged into one line (line 23) in the TRG_SALES table shown in Figure 8–12.

8.2.3.2 Viewing the Resulting Data

Note that the Pop. TRG_SALES interface has already been executed in the Load Sales Administration package. This is why the TRG_SALES table now contains 62 rows and not only 5 inserts as shown in Figure 8–9.

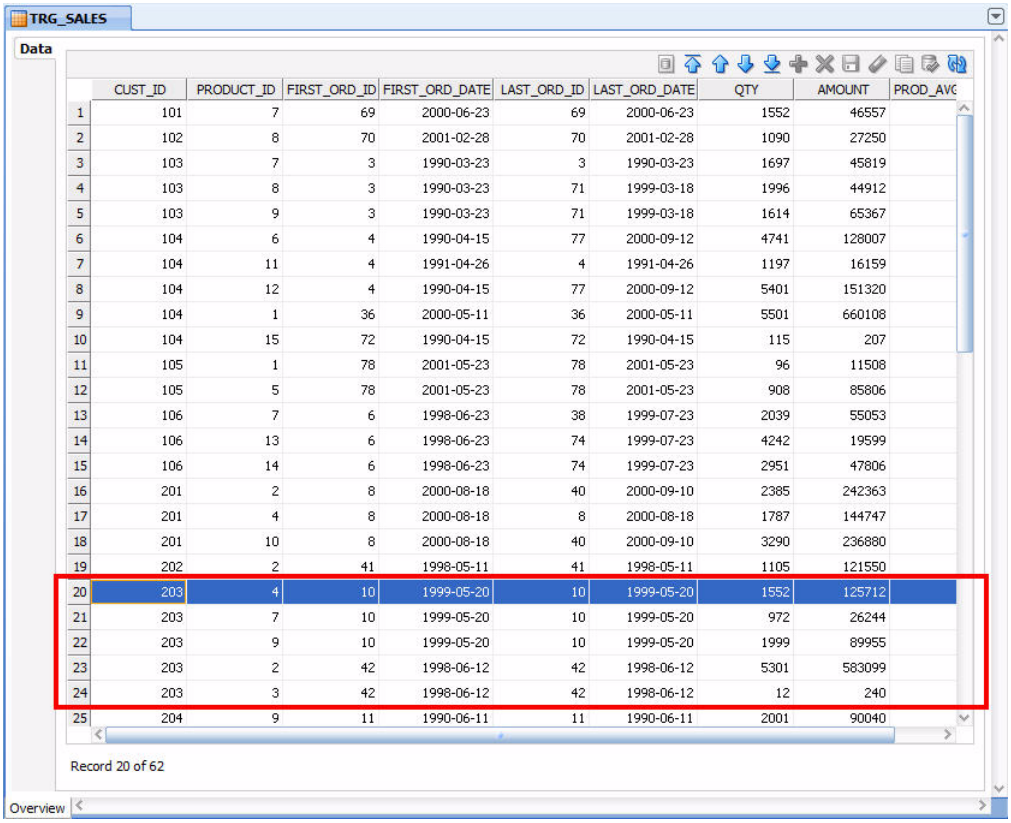
To view the data resulting of your interface execution:

1. In Designer Navigator, expand the Models accordion and the *Sales Administration - HSQL* model.
2. Select the TRG_SALES datastore.
3. Right-click and select **View Data** to view the data in the target table.

   Note that you can also select **Data...** to view and edit the data of the target table.

   The View Data Editor is displayed as shown in Figure 8–12.

Figure 8–12 View Data Editor for TRG_SALES



	CUST_ID	PRODUCT_ID	FIRST_ORD_ID	FIRST_ORD_DATE	LAST_ORD_ID	LAST_ORD_DATE	QTY	AMOUNT	PROD_AVG
1	101	7	69	2000-06-23	69	2000-06-23	1552	46557	
2	102	8	70	2001-02-28	70	2001-02-28	1090	27250	
3	103	7	3	1990-03-23	3	1990-03-23	1697	45819	
4	103	8	3	1990-03-23	71	1999-03-18	1996	44912	
5	103	9	3	1990-03-23	71	1999-03-18	1614	65367	
6	104	6	4	1990-04-15	77	2000-09-12	4741	128007	
7	104	11	4	1991-04-26	4	1991-04-26	1197	16159	
8	104	12	4	1990-04-15	77	2000-09-12	5401	151320	
9	104	1	36	2000-05-11	36	2000-05-11	5501	660108	
10	104	15	72	1990-04-15	72	1990-04-15	115	207	
11	105	1	78	2001-05-23	78	2001-05-23	96	11508	
12	105	5	78	2001-05-23	78	2001-05-23	908	85806	
13	106	7	6	1998-06-23	38	1999-07-23	2039	55053	
14	106	13	6	1998-06-23	74	1999-07-23	4242	19599	
15	106	14	6	1998-06-23	74	1999-07-23	2951	47806	
16	201	2	8	2000-08-18	40	2000-09-10	2385	242363	
17	201	4	8	2000-08-18	8	2000-08-18	1787	144747	
18	201	10	8	2000-08-18	40	2000-09-10	3290	236880	
19	202	2	41	1998-05-11	41	1998-05-11	1105	121550	
20	203	4	10	1999-05-20	10	1999-05-20	1552	125712	
21	203	7	10	1999-05-20	10	1999-05-20	972	26244	
22	203	9	10	1999-05-20	10	1999-05-20	1999	89955	
23	203	2	42	1998-06-12	42	1998-06-12	5301	583099	
24	203	3	42	1998-06-12	42	1998-06-12	12	240	
25	204	9	11	1990-06-11	11	1990-06-11	2001	90040	

8.2.3.3 Reviewing the Invalid Records and Incorrect Data

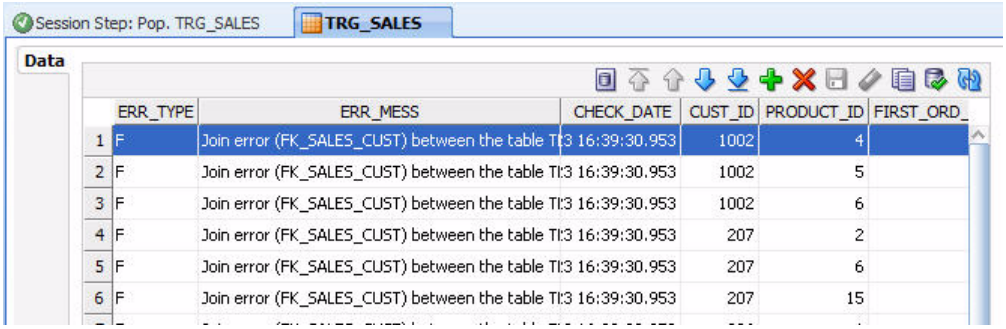
You can access the invalid records by right-clicking on the datastore in your model and selecting **Control** > **Errors...**

To review the error table of the TRG_SALES datastore:

1. In Designer Navigator, expand the *Sales Administration - HSQL* model.
2. Select the TRG_SALES datastore.
3. Right-click and select **Control** > **Errors...**

4. The Error Table Editor is displayed as shown in Figure 8–13.

Figure 8–13 Error Table of TRG_SALES



	ERR_TYPE	ERR_MESS	CHECK_DATE	CUST_ID	PRODUCT_ID	FIRST_ORD_
1	F	Join error (FK_SALES_CUST) between the table TI	3 16:39:30.953	1002	4	
2	F	Join error (FK_SALES_CUST) between the table TI	3 16:39:30.953	1002	5	
3	F	Join error (FK_SALES_CUST) between the table TI	3 16:39:30.953	1002	6	
4	F	Join error (FK_SALES_CUST) between the table TI	3 16:39:30.953	207	2	
5	F	Join error (FK_SALES_CUST) between the table TI	3 16:39:30.953	207	6	
6	F	Join error (FK_SALES_CUST) between the table TI	3 16:39:30.953	207	15	

The interface that you have executed has identified and isolated 32 invalid records in an error table that was automatically created for you.

In this error table, you can see that the interface rejected:

- 31 records in violation of the FK_SALES_CUST constraint (for example, have a customer number that does not exist in the table of customers)
- 1 record in violation of the FK_SALES_PROD constraint (has a product number that does not exist in the table of products)

The invalid records were saved into an error table and not integrated into the target table.

9

Deploying Integrated Applications

This chapter describes how to run the Load Sales Administration Package in a production environment.

This chapter includes the following sections:

9.1 Introduction

The automation of the data integration flows is achieved by sequencing the execution of the different steps (interfaces, procedures, and so forth) in a package and by producing a production scenario containing the ready-to-use code for each of these steps.

Chapter 7, "Working with Packages" describes the first part of the automation process: sequencing the execution of the different processes in a Package.

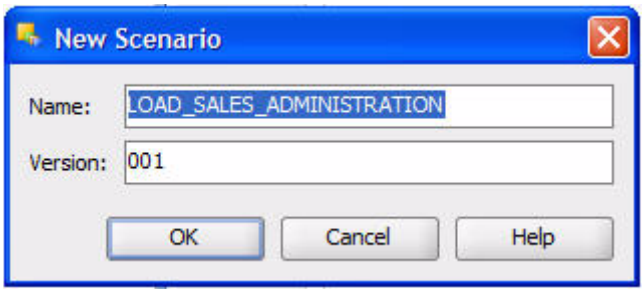
This chapter describes the second part: how to produce a scenario that runs automatically the Load Sales Administration Package in a production environment.

9.2 Scenario Creation

To generate the LOAD_SALES_ADMINISTRATION scenario that executes the Load Sales Administration Package:

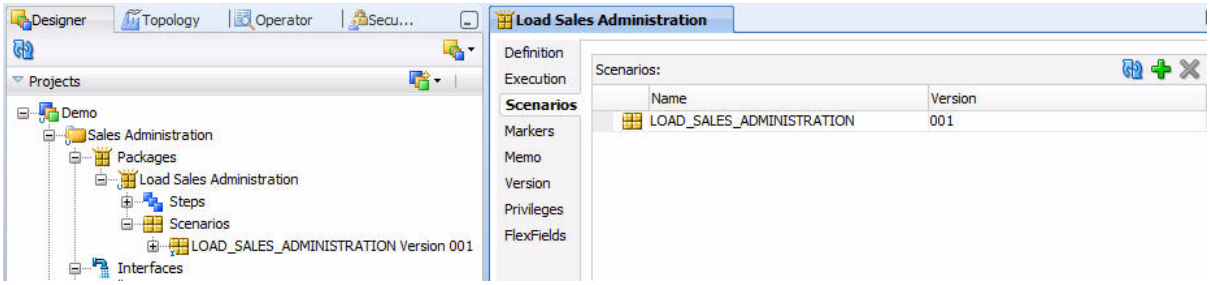
1. In the Load Sales Administration Package Editor, go to the Scenarios tab.
2. In the Scenarios toolbar menu, click **Generate Scenario.** The New Scenario dialog appears as shown in Figure 9–1.

Figure 9–1 New Scenario Dialog



3. The Name and Version fields of the Scenario are preset. Leave these values and click **OK**.
4. Oracle Data Integrator processes and generates the scenario. The new scenario appears on the Scenarios tab of the Package Editor and in the Demo Project as shown in Figure 9–2.

Figure 9–2 LOAD_SALES_ADMINISTRATION Scenario



9.3 Run the Scenario

Scenarios can be executed in several ways:

- Executing a Scenario from ODI Studio
- Executing a Scenario from a Command Line
- Executing a Scenario from a Web Service.

This Getting Started describes how to execute a scenario from ODI Studio. See "Executing a Scenario" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* for more information about how to execute a scenario from a command line and a web service.

9.3.1 Executing a Scenario from ODI Studio

You can start a scenario from Oracle Data Integrator Studio from Designer or Operator Navigator.

To start the LOAD_SALES_ADMINISTRATION scenario from Oracle Data Integrator Studio:

1. Select the LOAD_SALES_ADMINISTRATION scenario in the Projects accordion (in Designer Navigator) or the Scenarios accordion (in Operator Navigator).
2. Right-click, then select **Execute**.

3. In the Execution Dialog, leave the default settings and click **OK**.
4. The Session Started Information Dialog is displayed. Click **OK**.

The scenario is executed.

9.4 Follow the Execution of the Scenario

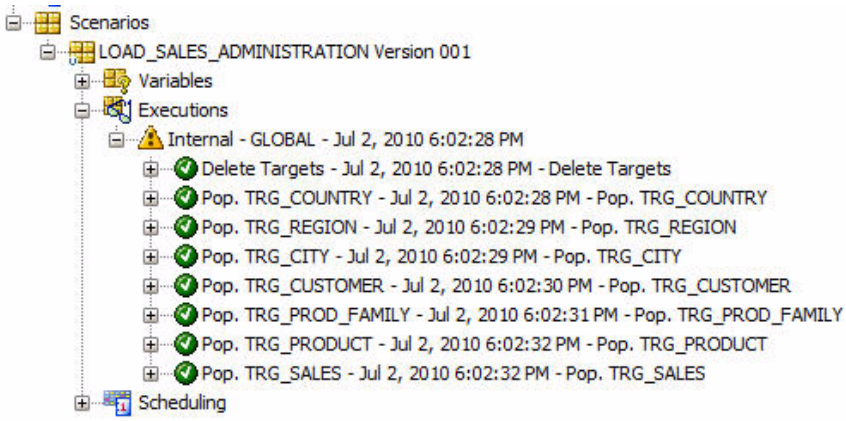
You can review the scenario execution in Operator Navigator, and find the same results as those obtained when the package was executed as described in Section 8.1.1, "Run the Package".

It is also possible to review the scenario execution report in Designer Navigator.

To view the execution results of the LOAD_SALES_ADMINISTRATION scenario in Designer Navigator:

1. In the Projects accordion in Designer Navigator, expand the Scenarios node under the Load Sales Administration package.
2. Refresh the displayed information by clicking **Refresh** in the Designer Navigator toolbar menu.
3. The log for the execution session of the LOAD_SALES_ADMINISTRATION scenario appears as shown in Figure 9–3.

Figure 9–3 LOAD_SALES_ADMINISTRATION Scenario Session Log



10

Going Further with Oracle Data Integrator

This chapter provides information for going further with Oracle Data Integrator.

This chapter includes the following sections:

10.1 Summary

Congratulations! You have now completed an ETL project and learned about the fundamentals of Oracle Data Integrator.

In this Getting Started guide, you learned how to:

- Define and implement data integrity rules in the *Orders Application - HSQL* application (Chapter 5, "Implementing Data Quality Control")
- Create interfaces to load the data from the *Orders Application - HSQL* and *Parameters - FILE* applications into the *Sales Administration* data warehouse (Chapter 6, "Working with Integration Interfaces")
- Sequence your developments (Chapter 7, "Working with Packages")
- Prepare your process for deployment (Chapter 9, "Deploying Integrated Applications")

10.2 What else can you do with Oracle Data Integrator?

You have learned how to use Oracle Data Integrator for a typical Data Warehousing project. But Oracle Data Integrator is capable of addressing any type of data-driven integration, from batch to near-real-time, as for example:

- Data Migration - with or without subsequent replication between the old and the new system
- Point-to-point Data Integration
- Data Replication

Furthermore, in this Getting Started guide you have only seen Oracle Data Integrator connecting to a relational database and files. Oracle Data Integrator can also access and integrate all database systems, ERPs and CRMs, mainframes, flat files, LDAP directories, XML data sources, and so forth - all within the same toolset and using the same methodology.

Oracle Data Integrator is the only integration platform that unifies data, event, and service-based integration with a common declarative rules driven approach. It enables the enterprise to present a single view of its Information System, with a single, unified access model.

Some of the benefits that you will find from using Oracle Data Integrator include:

- **Unified integration support**: Oracle Data Integrator is the only integration application software to support data-, event- and service-oriented integration with the same interface. This unique feature allows IT teams to cover all integration needs: batch and real-time, asynchronous and synchronous - regardless of data volumes or latency requirements.
- **Enhanced productivity and a short learning curve**: the declarative rules driven approach is shared throughout Oracle Data Integrator, regardless of the data, event or service orientation of each integration mechanism. With a common use model and shared user interfaces throughout the platform, the learning curve is shortened and productivity is dramatically increased.
- **Shared, reusable metadata**: with a single metadata repository that is fully integrated with all components of Oracle Data Integrator, the consistency of the integration processes is guaranteed. The repository also promotes the reusability of declarative rules for data transformation and data validation across processes.
- **Support for multiple applications**: Oracle Data Integrator is well suited to a broad range of integration projects- ETL, Data Migration, Master data management, Business Activity Monitoring (BAM), Business Process Management (BPM), Business Process Reengineering (BPR), and Web Services integration - implemented using a combination of Data-oriented, Event-oriented, and Service-oriented mechanisms.

10.3 Learn More

You can learn more about creating your own integration projects with Oracle Data Integrator in the guides listed in Table 10–1.

Table 10–1 Oracle Data Integrator Documentation

Document	Description
Oracle Fusion Middleware Installation Guide for Oracle Data Integrator	Provides Oracle Data Integrator installation information including pre-installation requirements and troubleshooting.
Oracle Fusion Middleware Upgrade Guide for Oracle Data Integrator	Provides 11g upgrade information for Oracle Data Integrator.
Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator	Provides guidelines for developers interested in using Oracle Data Integrator for integration projects.
Oracle Fusion Middleware Connectivity and Knowledge Modules Guide for Oracle Data Integrator	Describes Oracle Data Integrator Knowledge Modules and technologies and how to use them in integration projects.
Oracle Fusion Middleware Knowledge Module Developer's Guide for Oracle Data Integrator	Describes ho to develop your own Knowledge Modules for Oracle Data Integrator.

You can find all Oracle Data Integrator documentation on the Oracle Data Integrator documentation page on the Oracle Technology Network, at:

http://www.oracle.com/technetwork/middleware/data-integrator/documentation/index.html

The Oracle Data Integrator home page on the Oracle Technology Network also provides the following resources to learn more about other features of Oracle Data Integrator:

- View the *Two Minutes Product Tour*. This viewlet provides a short introduction and overview of the main ODI features.
- View the *Oracle by Example Series for ODI*. The Oracle by Example (OBE) series provides step-by-step instructions on how to perform a variety of tasks using Oracle Data Integrator Suite.

To learn more about the new features that have been introduced in Oracle Data Integrator 11*g*, see "What's New in Oracle Data Integrator?" in the *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* and the Release Notes.

Thank you for choosing Oracle Data Integrator!