Oracle® Data Integrator

Getting Started with an ETL Project

10g Release 3 (10.1.3)

September 2008

Oracle® Data Integrator: Getting Started with an ETL Project, 10g Release 3 (10.1.3)

Table Of Contents

About this Manual

The purpose of this document is to provide you with an overview of the possibilities offered by Oracle Data Integrator.

This document provides examples to help you understand how to achieve an ETL project with **Oracle Data Integrator**: loading a Data Warehouse with data from heterogeneous systems, while validating data quality.

This project should take approximately 90 minutes to complete. You may save your work in order to pause and resume at any point in the project.

Product Overview

What is 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, 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.

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.

A Unique Solution

Oracle Data Integrator employs a powerful declarative design approach to ETL, which separates the declarative rules from the implementation details. Oracle Data Integrator is also based on a unique “E-LT” 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 more than 500 companies have chosen Oracle Data Integrator for their ETL needs:

- **Faster and simpler development and maintenance**: The declarative rules driven approach to ETL 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 ETL software 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 hub server sitting between the sources and the target server. It utilizes the target server and

its RDBMS 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 Teradata, IBM DB2, Netezza, Oracle, Sybase IQ and numerous other technologies such as flat files, ERPs, LDAP, XML.
- **Cost-savings**: the elimination of the ETL hub 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.

Technical Architecture

Oracle Data Integrator is built around a modular repository architecture and accessed in client/server or thin client mode by components that are developed entirely in Java: the graphical interface modules and execution agents.

The **Repository** is a comprehensive, open and shared metadata repository. It stores the metadata for the accessed applications involved in the transformation and integration processing, the developed project versions and all of the information required for their use (planning, scheduling and execution reports).

Topology Manager is the graphical interface module used to manage the data describing the information system's physical and logical architecture. The site, machine, and data server descriptions will enable Oracle Data Integrator to execute the same interfaces in different environments.

Designer is a tool used to design data integrity checks and to build transformations. In particular, it incorporates functions for:

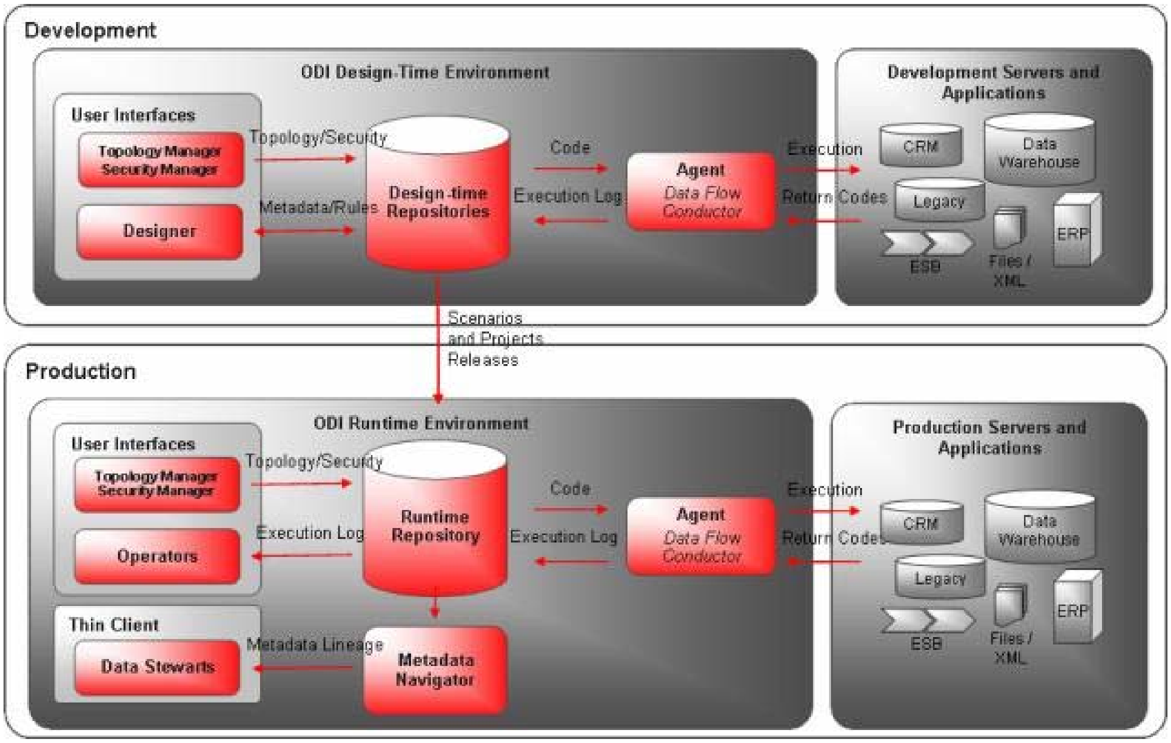
- Automatic reverse-engineering of existing applications or databases
- Graphical development and maintenance of transformation and integration interfaces
- Visualization of data flows in the interfaces
- Automatic documentation generation
- Customization of the generated code

Security Manager is the tool for managing users and permissions in Oracle Data Integrator.

Operator is the production management and monitoring module. It is designed for IT production operators.

The **Agent** is used to execute the inter-application interfaces developed with Designer. It integrates the **Scheduler** to execute the interfaces according to a predefined schedule.

Metadata Navigator is a thin client, browser based interface to access the Oracle Data Integrator Repository, browse, search and analyze all its contents, and perform advanced data lineage tracing, cross referencing, impact analysis, etc. You may even use this tool to manage all of your production processes.


Development
ODI Design-Time Environment
User Interfaces
Topology Manager
Security Manager
Designer
Topology/Security
Metadata/Rules
Design-time Repositories
Code
Execution Log
Agent
Data Flow Conductor
Execution
Return Codes
Development Servers and Applications
CRM
Data Warehouse
Legacy
ESB
Files / XML
ERP
Scenarios and Projects Releases
Production
ODI Runtime Environment
User Interfaces
Topology Manager
Security Manager
Operators
Topology/Security
Execution Log
Runtime Repository
Code
Execution Log
Agent
Data Flow Conductor
Execution
Return Codes
Production Servers and Applications
CRM
Data Warehouse
Legacy
ESB
Files / XML
ERP
Thin Client
Data Stewarts
Metadata Lineage
Metadata Navigator

Installing Oracle Data Integrator

To follow the steps described in this document, you must have Oracle Data Integrator correctly installed on your system. The installation procedure is described below.

Prerequisites

Your workstation must meet the following requirements:

- Any OS with Windows or X/Windows capacities (Windows, Linux, HP-UX, Solaris etc.)
- 256 MB / 512 of RAM recommended
- 150 MB hard disk
- TCP/IP

Installation

Installing Oracle Data Integrator from a CD

If you have an Oracle Data Integrator setup CD:

1. Insert the Oracle Data Integrator CD-ROM in your CD-ROM drive.
2. Go to the `/setup/<system>/` directory on the CD, where `<system>` is your platform name.
3. Run `setup.exe`. on a Windows platform or `sh ./setup.bin` on a Unix platform.
4. Follow the instructions in the setup program.

Installing Oracle Data Integrator from a Downloaded File

1. Run `setup.exe`. on a Windows platform or `sh ./setup.bin` on a Unix platform.
2. Follow the instructions in the setup program.

Note: A Java Virtual Machine is included in the setup program, and thus an external JVM is not required. You should use the JVM supplied by the Oracle Data Integrator installation program.

Note: In order to use web services, Oracle Data Integrator requires a Java Development Kit, not supplied in the installation program. The warning *"A JDK is required to execute Web Services with Oracle Data Integrator. You are currently using a JRE."* may appear when starting Oracle Data Integrator components. Ignore this warning.

Note: This setup program creates shortcuts to the installed Oracle Data Integrator components.

Caution: If you use Microsoft Windows 95 or 98 please refer to the Run Oracle Data Integrator with Windows 95/98 chapter in the online help.

Note: During the installation, you will be given the choice of the type of installation you wish to perform. You can choose the *Evaluation* install set if you want only the basic files or the *Full* install set if you also wish to install extra utilities such as those used to create repositories. Either installation will work for the purposes of this getting started.

Start your project

Oracle Data Integrator is now installed on your machine with the evaluation environment.

The ETL Project

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

Examples in this Document

The Example Environment

Throughout this "getting started" guide, we will be looking at an organization that wishes to track sales from various heterogeneous data sources issued from the production systems. The following data sources will be used:

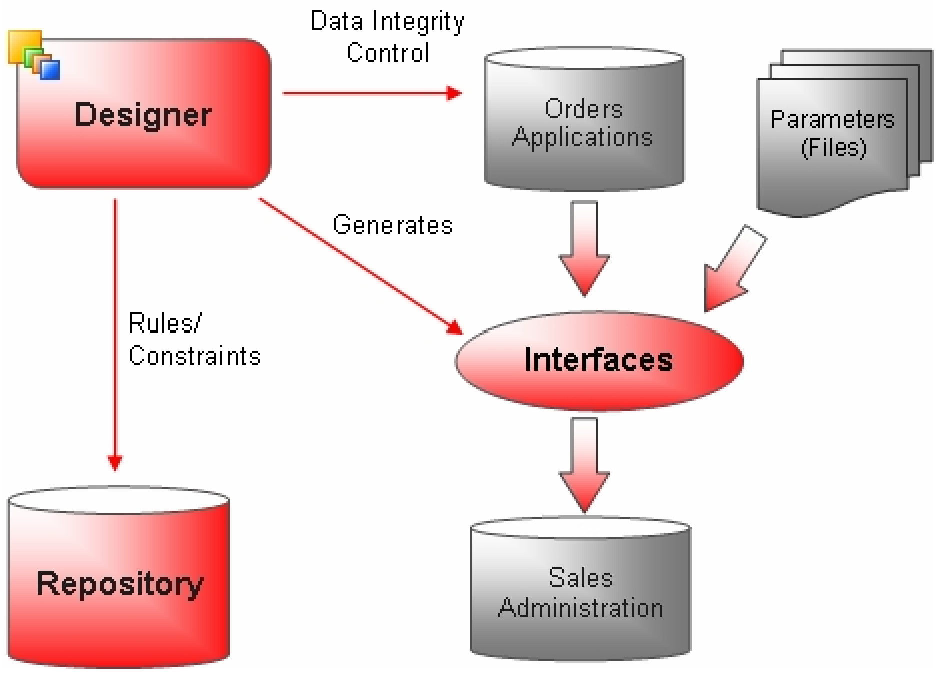
The Repository: The Repository contains all of the metadata required for the training examples. It is hosted, for these evaluation purposes, in a supplied database.

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). We will populate this data warehouse with our transformations.

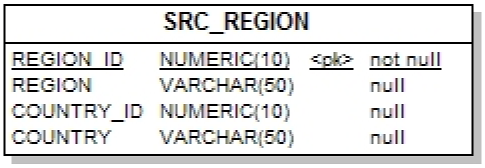
A graphical representation of the environment is given below:



The Data Models

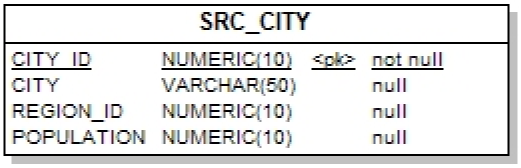
The schema diagrams below define the supplied data models that we will be using throughout this guide:

Orders Application



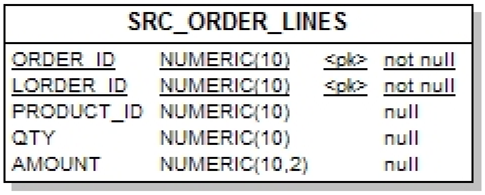
SRC_REGION

REGION_ID	NUMERIC(10)	<pk>	not null
REGION	VARCHAR(50)		null
COUNTRY_ID	NUMERIC(10)		null
COUNTRY	VARCHAR(50)		null



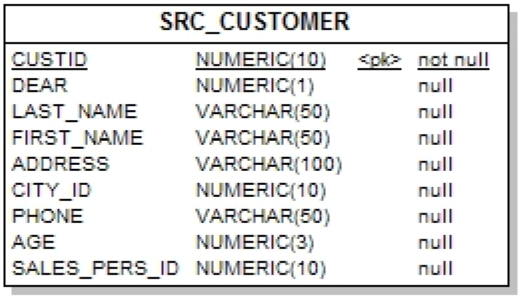
SRC_CITY

CITY_ID	NUMERIC(10)	<pk>	not null
CITY	VARCHAR(50)		null
REGION_ID	NUMERIC(10)		null
POPULATION	NUMERIC(10)		null



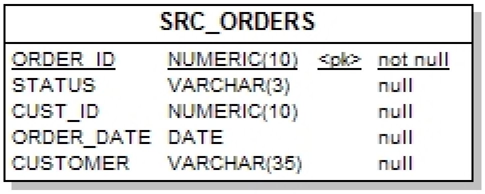
SRC_ORDER_LINES

ORDER_ID	NUMERIC(10)	<pk>	not null
LORDER_ID	NUMERIC(10)	<pk>	not null
PRODUCT_ID	NUMERIC(10)		null
QTY	NUMERIC(10)		null
AMOUNT	NUMERIC(10,2)		null



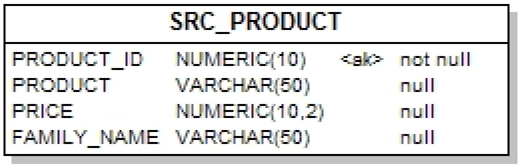
SRC_CUSTOMER

CUSTID	NUMERIC(10)	<pk>	not null
DEAR	NUMERIC(1)		null
LAST_NAME	VARCHAR(50)		null
FIRST_NAME	VARCHAR(50)		null
ADDRESS	VARCHAR(100)		null
CITY_ID	NUMERIC(10)		null
PHONE	VARCHAR(50)		null
AGE	NUMERIC(3)		null
SALES_PERS_ID	NUMERIC(10)		null



SRC_ORDERS

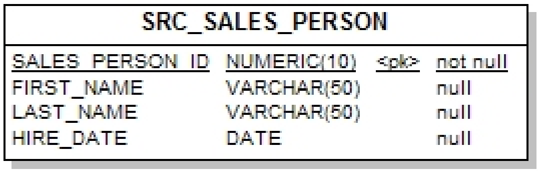
ORDER_ID	NUMERIC(10)	<pk>	not null
STATUS	VARCHAR(3)		null
CUST_ID	NUMERIC(10)		null
ORDER_DATE	DATE		null
CUSTOMER	VARCHAR(35)		null



SRC_PRODUCT

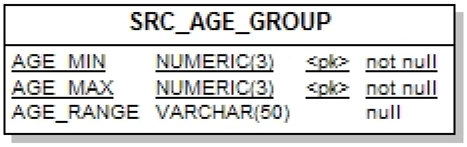
PRODUCT_ID	NUMERIC(10)	<ak>	not null
PRODUCT	VARCHAR(50)		null
PRICE	NUMERIC(10,2)		null
FAMILY_NAME	VARCHAR(50)		null

Parameters (File)



SRC_SALES_PERSON

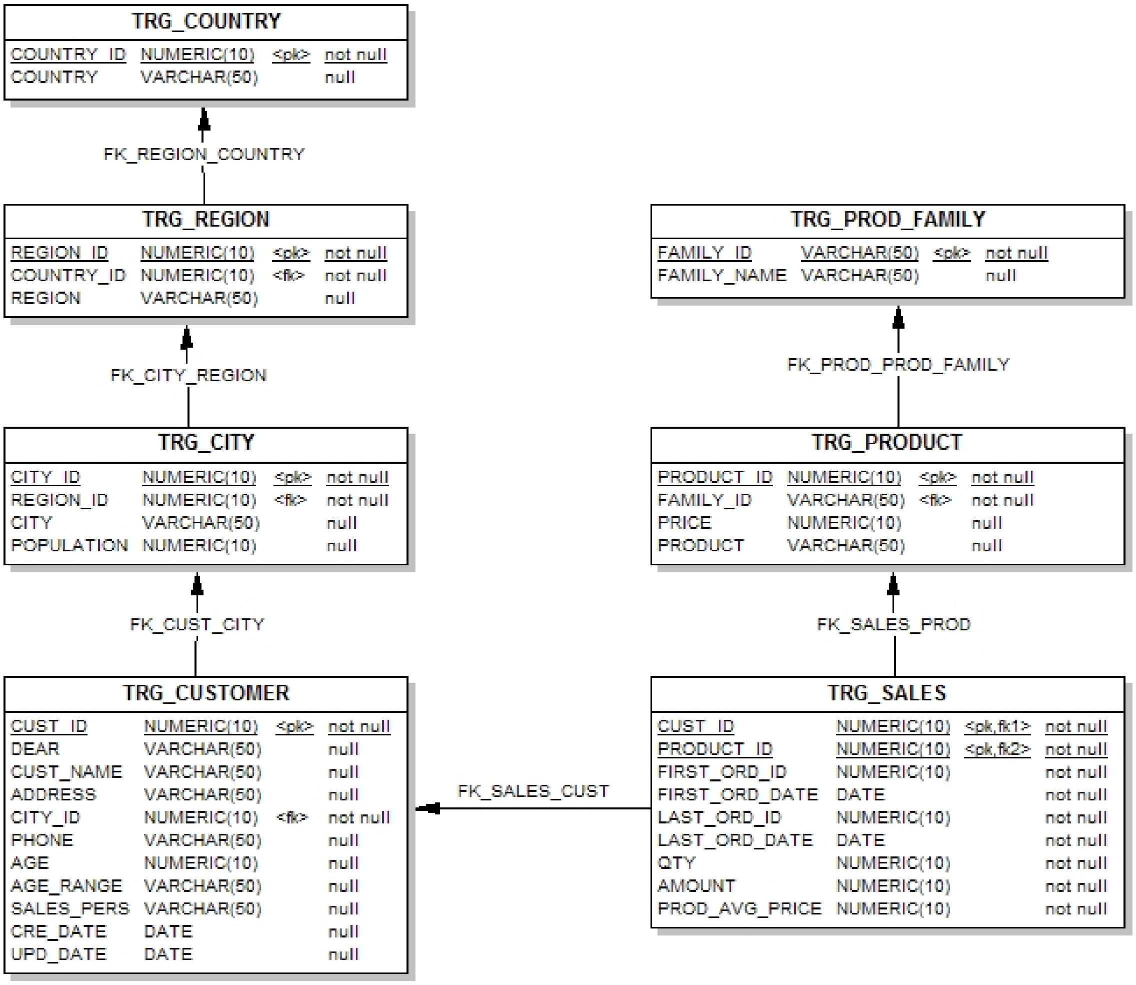
SALES_PERSON_ID	NUMERIC(10)	<pk>	not null
FIRST_NAME	VARCHAR(50)		null
LAST_NAME	VARCHAR(50)		null
HIRE_DATE	DATE		null



SRC_AGE_GROUP

AGE_MIN	NUMERIC(3)	<pk>	not null
AGE_MAX	NUMERIC(3)	<pk>	not null
AGE_RANGE	VARCHAR(50)		null

Sales Administration



TRG_COUNTRY

COUNTRY_ID	NUMERIC(10)	<pk>	not null
COUNTRY	VARCHAR(50)		null

TRG_REGION

REGION_ID	NUMERIC(10)	<pk>	not null
COUNTRY_ID	NUMERIC(10)	<fk>	not null
REGION	VARCHAR(50)		null

TRG_CITY

CITY_ID	NUMERIC(10)	<pk>	not null
REGION_ID	NUMERIC(10)	<fk>	not null
CITY	VARCHAR(50)		null
POPULATION	NUMERIC(10)		null

TRG_CUSTOMER

CUST_ID	NUMERIC(10)	<pk>	not null
DEAR	VARCHAR(50)		null
CUST_NAME	VARCHAR(50)		null
ADDRESS	VARCHAR(50)		null
CITY_ID	NUMERIC(10)	<fk>	not null
PHONE	VARCHAR(50)		null
AGE	NUMERIC(10)		null
AGE_RANGE	VARCHAR(50)		null
SALES_PERS	VARCHAR(50)		null
CRE_DATE	DATE		null
UPD_DATE	DATE		null

TRG_PROD_FAMILY

FAMILY_ID	VARCHAR(50)	<pk>	not null
FAMILY_NAME	VARCHAR(50)		null

TRG_PRODUCT

PRODUCT_ID	NUMERIC(10)	<pk>	not null
FAMILY_ID	VARCHAR(50)	<fk>	not null
PRICE	NUMERIC(10)		null
PRODUCT	VARCHAR(50)		null

TRG_SALES

CUST_ID	NUMERIC(10)	<pk,fk1>	not null
PRODUCT_ID	NUMERIC(10)	<pk,fk2>	not null
FIRST_ORD_ID	NUMERIC(10)		not null
FIRST_ORD_DATE	DATE		not null
LAST_ORD_ID	NUMERIC(10)		not null
LAST_ORD_DATE	DATE		not null
QTY	NUMERIC(10)		not null
AMOUNT	NUMERIC(10)		not null
PROD_AVG_PRICE	NUMERIC(10)		not null

Issues

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

We will use examples to illustrate how to address these issues, over the following topics:

- Checking data in a database
- Creating and executing an ETL transformation
- Cleansing the results of a transformation
- Automating this process

Note: In this guide, we will be looking at processes that focus on ETL. While it is beyond the scope of this document, creating a real-time EAI process can be carried out in the same fashion. For more information on this, please refer to the Oracle Data Integrator documentation after completing this guide.

Cleansing Data

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.

Loading "Sales Administration"

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.

Automating the Processes Developed in Oracle Data Integrator

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.

Starting Data Integrator

First Run

Starting the Demonstration Environment

Oracle Data Integrator provides the databases that contain the data used in these examples as well as the initial repository. Before looking at the different examples, **you must do this first:**

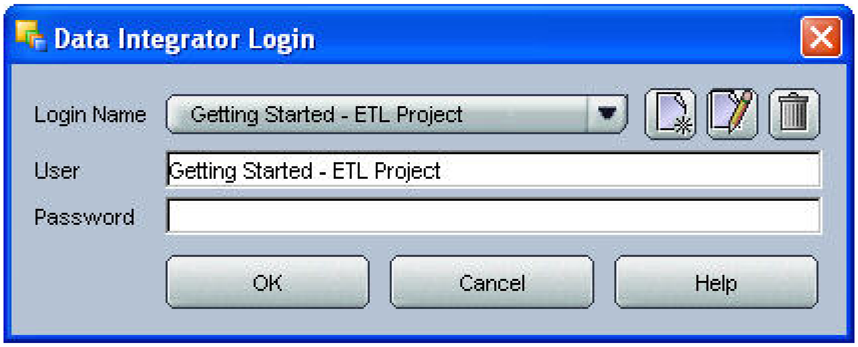
Select **Start Menu** > **Programs** > **Oracle Data Integrator** > **Examples**, then select **Start Demo Environment**. This starts the Repository database server as well as the source and target databases.

To stop these database servers, you can use the **Stop Demo Environment** shortcut in the same folder. **Do not shut down the databases by using the [CTRL-C] keyboard shortcut, or by closing their execution windows**. This may corrupt the data in the database.

If you are in a UNIX environment, you can use the command prompt to run the appropriate .sh files located in your Oracle Data Integrator install/bin directory by adding "`&`" to start the servers as a background task (for example, type: `startdemo.sh&` to start the Repository database and `stopdemo.sh` to stop it.)

Starting Designer

1. Select **Start Menu** > **Programs** > **Oracle Data Integrator** > **Designer**. The following window is displayed, prompting you to select the repository you wish to connect to:



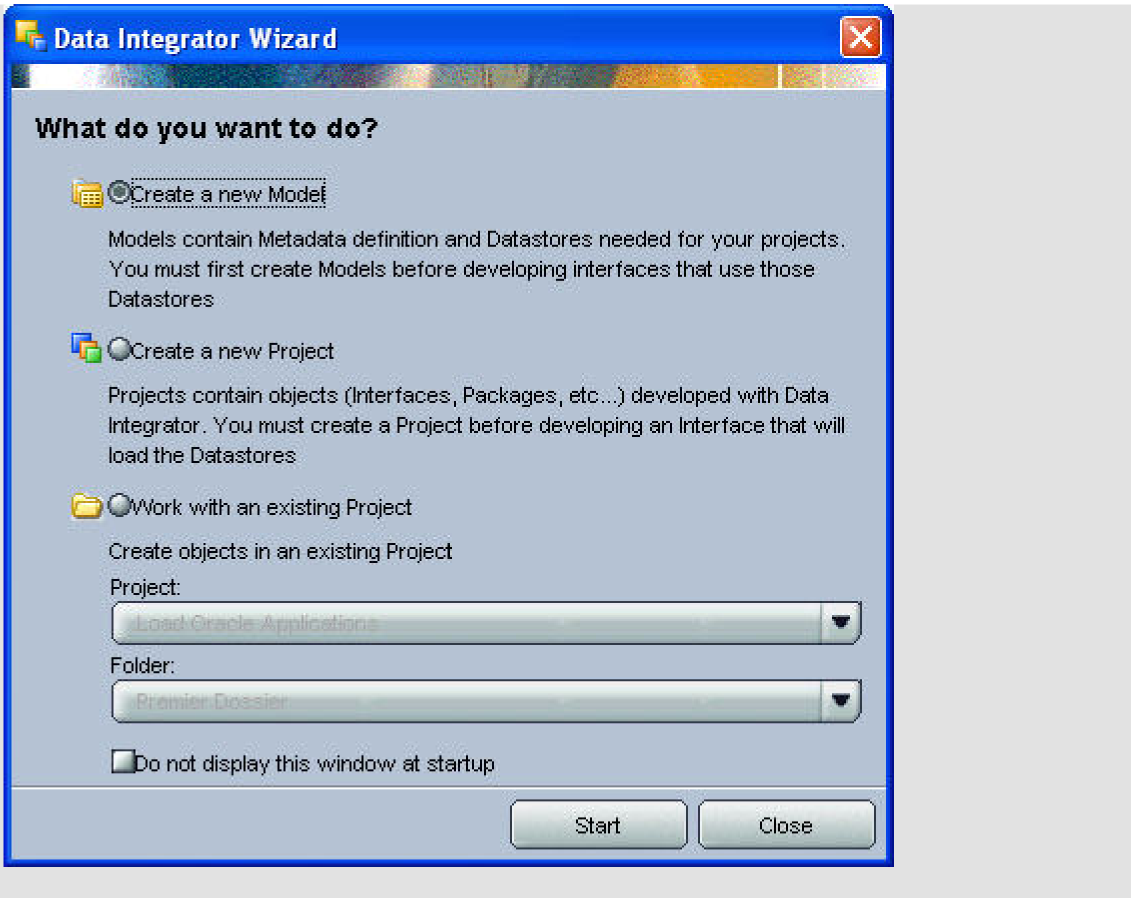
Login Name: Getting Started - ETL Project
User: Getting Started - ETL Project
Password:

2. Select the appropriate login from the **Login Name** from the dropdown menu. Choose *Getting Started – ETL Project*.
3. Click on **OK** to connect to the Repository.

Note: Make sure you have **started the Repository server** as specified in the previous section, as well as the "Orders Application" and "Sales Administration" applications database servers.

4. **Designer** starts

Note: When Designer opens, a Wizard window may appear. You can close this window by clicking **Close**.



Data Integrator Wizard

What do you want to do?

● Create a new Model

Models contain Metadata definition and Datastores needed for your projects. You must first create Models before developing interfaces that use those Datastores

○ Create a new Project

Projects contain objects (Interfaces, Packages, etc...) developed with Data Integrator. You must create a Project before developing an Interface that will load the Datastores

○ Work with an existing Project

Create objects in an existing Project

Project: Load Oracle Applications

Folder: Premier Dossier

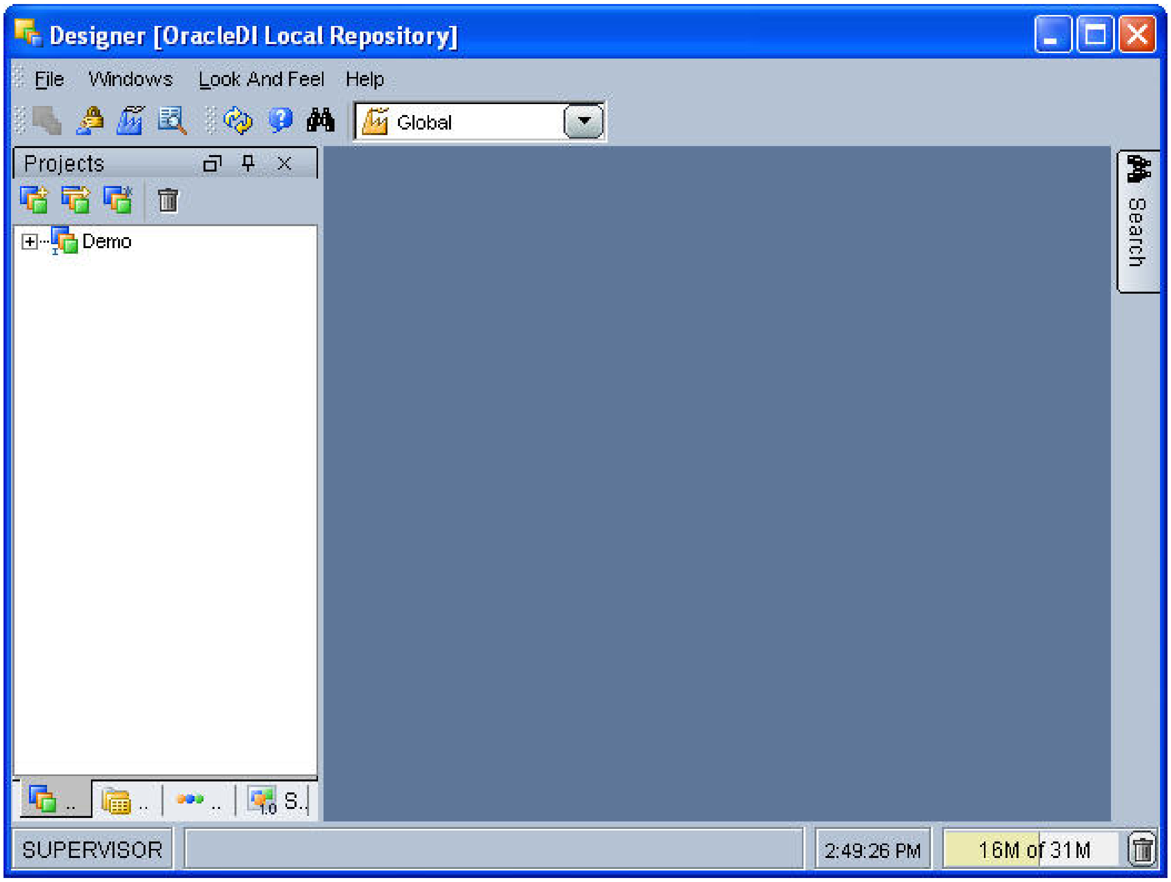
□ Do not display this window at startup

Designer Interface

Designer manages:

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

The Designer GUI appears as follow:



The Menu

The **Menu** contains pull-down menus to access the following features:

- Import/Export
- Wizards
- Display options
- Open modules or tree views
- Change the user's password and options

The Toolbar

The **Toolbar** enables you to:

- Open other Oracle Data Integrator modules
- Refresh the Tree views
- Open the on-online help
- Choose the default **context.** The context used for this project is called *Global*.

The Tree Views

Designer objects available to the current user are organized into the following tree views: **Projects**, **Models**, **Solutions** and **Other** (User Functions, Global Variables and Sequences).

You will perform most actions on the objects via pop-up menus (by selecting the object, then right-clicking to display a menu).

Each tree view appears in a floatable frames that may be docked to the sides of the Designer main window. These frames can also be stacked up. When several frames are stacked up, tabs appear at the bottom of the frame window to access each frame of the stack.

Tree view frames can be moved, docked and stacked by selecting and dragging the frame title or tab. If a tree view frame does not appear in the Designer main window or has been closed, it can be opened using the **Windows** > **Show View** menu.

Objects already exist in the demo repository:

- In the **Models** view, you will find all the data models corresponding to the *Orders Application*, *Parameters* and *Sales Administration* applications.
- In the **Projects** view, 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 Workbench

The windows for object being edited or displayed appear in the **Workbench**.

Data Quality Control

Introduction to Data Integration

With multiple applications in your IT infrastructure reading and writing to and from different data stores in varying formats, it is imperative to implement a process that will let you integrate the data so that they can be easily used by anyone in your company. There are numerous Data Integration methods to accomplish this – ETL, Data Replication/Synchronization, etc. In all of these cases, the first step in successfully integrating your Information System's data with your organization's applications is ensuring the integrity of that data.

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, etc.

Oracle Data Integrator provides a working environment to detect these constraint violation and 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 refine the validation of your data by defining 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 as follow:

- 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 will benefit from using data that has been pre-treated to filter out business rule violations.

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.

Examples

The following example will guide you through the data integrity audit process (Static Control).

The "Orders Application - HSQL" application contains data which 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.

"SRC_CUSTOMER" Control

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:

- **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.

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.

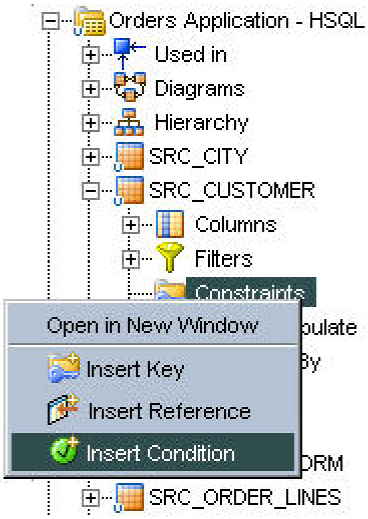
In Oracle Data Integrator, to add a condition you simply specify the rule to be satisfied. To add a reference you just define the columns that have to match. We will now create our two constraints:

Creating Constraints

Age Constraint

In this example, we will add a data validity condition on a column.

1. Expand the "Orders Application - HSQL" model.
2. Expand the "SRC_CUSTOMER" table.
3. Right-click on the **Constraints** node and choose **Insert Condition.**



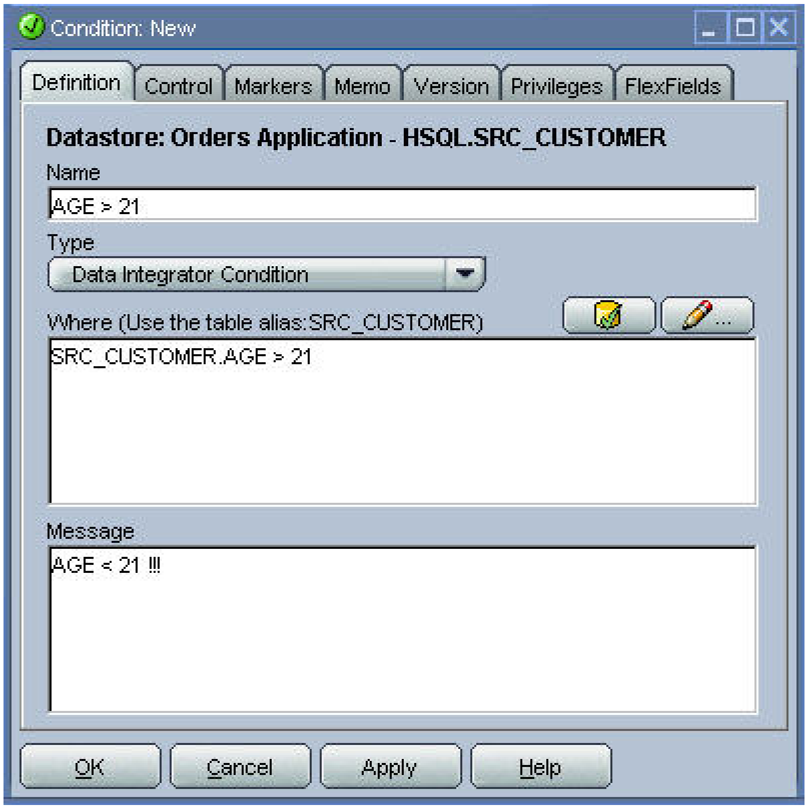
4. In the **Definition** tab:

- Enter the **Name** of your condition (e.g., "`AGE > 21`").
- For the **type** of condition, choose **Oracle Data Integrator Condition.**
- In the **Where** clause, enter the following SQL code: `SRC_CUSTOMER.AGE > 21`. You can enter this text directly or use the expression editor., which can be accessed by clicking the ✎ button.

Note: In order for Data Integrator to generate the right SQL code, you must use the **alias** of the datastore (in this case `SRC_CUSTOMER`) as a prefix for the AGE. This alias is defined in the **Definition** tab in the **Datastore** screen (right-click > **Edit** on a datastore).

Note: 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.
- Your screen should look as follows:



Datastore: Orders Application - HSQL.SRC_CUSTOMER

Name: AGE > 21

Type: Data Integrator Condition

Where (Use the table alias:SRC_CUSTOMER): SRC_CUSTOMER.AGE > 21

Message: AGE < 21 !!!

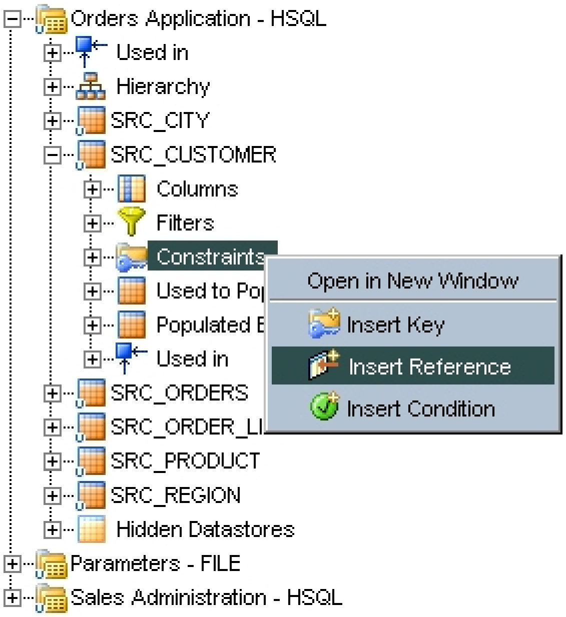
5. Click on **OK** to save the condition.

Reference Constraint

In this example, we wish to add the following referential constraint: The SRC_CUSTOMER table will use the CITY_ID column to reference SRC_CITY.

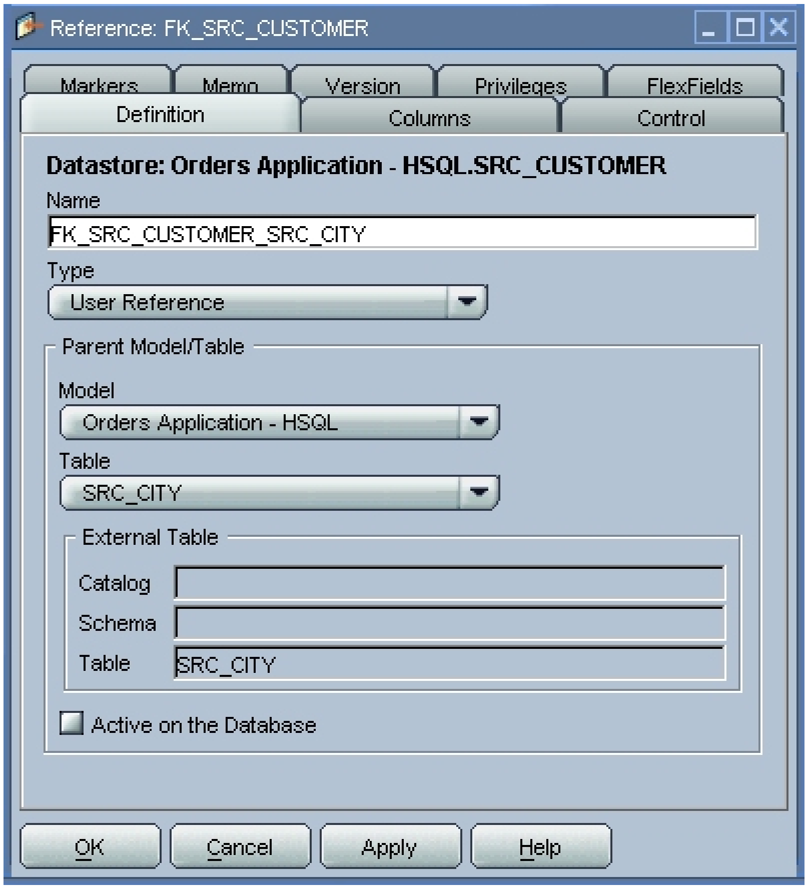
We will now create the constraint to check for valid CITY_ID values. To do this:

1. Expand the tree structure for the "Orders Application - HSQL" model.
2. Expand the tree structure for the "SRC_CUSTOMER" datastore.
3. Right-click on the **Constraints** node and select **Insert Reference.**



4. In the **Definition** tab:

- Select **User Reference** as the **type** of this reference.
- Select "SRC_CITY" in the list box corresponding to the table you want to link to.
- Your screen should look as follows:



Datastore: Orders Application - HSQL.SRC_CUSTOMER

Name: FK_SRC_CUSTOMER_SRC_CITY

Type: User Reference

Parent Model/Table

Model: Orders Application - HSQL

Table: SRC_CITY

External Table

Catalog:

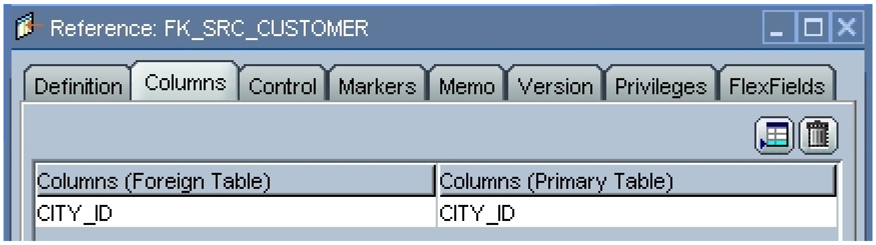
Schema:

Table: SRC_CITY

□ Active on the Database

5. In the **Columns** tab:

- Click on ▦ in order to 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.
- Select "CITY_ID" from the **Foreign Table** ("SRC_CUSTOMER") and from the **Primary Table** ("SRC_CITY"):



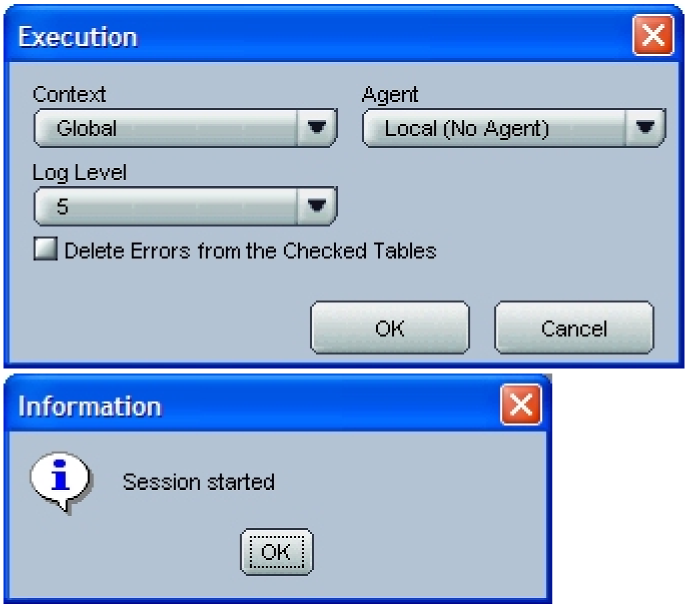
Columns (Foreign Table)	Columns (Primary Table)
CITY_ID	CITY_ID

6. Click **OK** to save this reference.

Run the Static Control

1. Right click the "SRC_CUSTOMER" datastore.
2. Select **Control > Check.**

3. Click **OK** in the **Execution** and **Information** windows.



Execution

Context: Global

Agent: Local (No Agent)

Log Level: 5

☐ Delete Errors from the Checked Tables

Information

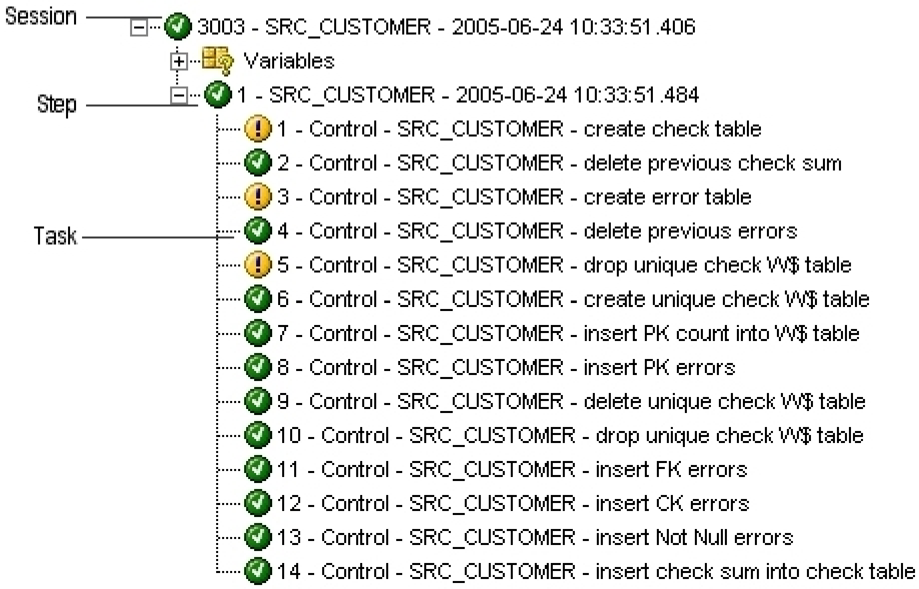
Session started

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

Follow the Execution of the Control in Operator

1. To view the steps in your execution in the **Operator**, click on the **Operator** button in your menu bar.
2. The **Operator** window that subsequently opens details the process tasks under the node **All Executions**, in Operator's **Session List** view.
3. You can refresh the window using the button if it is already opened.

The log for one execution session appears as follows:



```
3003 - SRC_CUSTOMER - 2005-06-24 10:33:51.406
  Variables
  1 - SRC_CUSTOMER - 2005-06-24 10:33:51.484
    1 - Control - SRC_CUSTOMER - create check table
    2 - Control - SRC_CUSTOMER - delete previous check sum
    3 - Control - SRC_CUSTOMER - create error table
    4 - Control - SRC_CUSTOMER - delete previous errors
    5 - Control - SRC_CUSTOMER - drop unique check W$ table
    6 - Control - SRC_CUSTOMER - create unique check W$ table
    7 - Control - SRC_CUSTOMER - insert PK count into W$ table
    8 - Control - SRC_CUSTOMER - insert PK errors
    9 - Control - SRC_CUSTOMER - delete unique check W$ table
    10 - Control - SRC_CUSTOMER - drop unique check W$ table
    11 - Control - SRC_CUSTOMER - insert FK errors
    12 - Control - SRC_CUSTOMER - insert CK errors
    13 - Control - SRC_CUSTOMER - insert Not Null errors
    14 - Control - SRC_CUSTOMER - insert check sum into check table
```

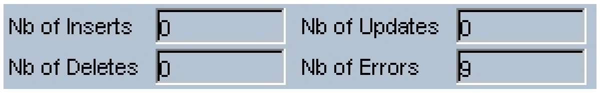
The log comprises 3 levels:

- The **session.**
- The **step** (corresponds to a checked datastore, an interface or a procedure)
- The **task** (corresponds to an elementary task of the interface, process or check)

Interpreting the Results

The Number of Invalid Records

1. In **Operator**, edit the "SRC_CUSTOMER" **step** (level 2) window by double-clicking the corresponding icon in the execution tree. A new window will open to the right of the screen.
2. Select the **Execution** tab. Notice that checking the "SRC_CUSTOMER" table has revealed **9** invalid records. These records have been isolated in an error table.

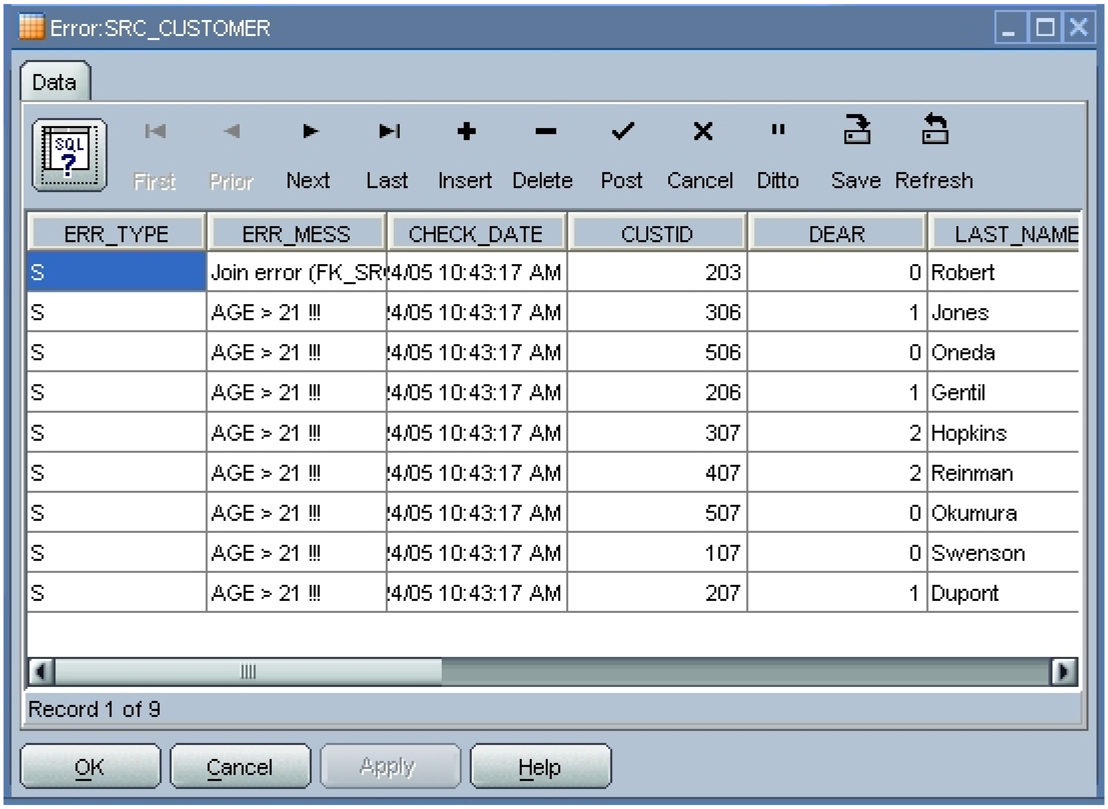


Nb of Inserts	0	Nb of Updates	0
Nb of Deletes	0	Nb of Errors	9

Invalid Records

You can access the invalid records by right-clicking on the table in your model, then choosing **Control > Errors** from the popup menu.

1. Go back to the Designer module by clicking on the **Designer** button in the toolbar
2. Expand the "Orders Application - HSQL" model
3. Right-click on the "SRC_CUSTOMER" datastore
4. Choose **Control > Errors** from the popup menu
5. The window listing the errors in the data is displayed:



ERR_TYPE	ERR_MESS	CHECK_DATE	CUSTID	DEAR	LAST_NAME
S	Join error (FK_SR	4/05 10:43:17 AM	203	0	Robert
S	AGE > 21 !!!	4/05 10:43:17 AM	306	1	Jones
S	AGE > 21 !!!	4/05 10:43:17 AM	506	0	Oneda
S	AGE > 21 !!!	4/05 10:43:17 AM	206	1	Gentil
S	AGE > 21 !!!	4/05 10:43:17 AM	307	2	Hopkins
S	AGE > 21 !!!	4/05 10:43:17 AM	407	2	Reinman
S	AGE > 21 !!!	4/05 10:43:17 AM	507	0	Okumura
S	AGE > 21 !!!	4/05 10:43:17 AM	107	0	Swenson
S	AGE > 21 !!!	4/05 10:43:17 AM	207	1	Dupont

You can now observe the records that were invalidated by the check process:

- **8** records in violation of the "AGE > 21" constraint (i.e., the actual age is 21 or younger).
- **1** record in violation of the "FK_CITY_CUSTOMER" constraint (i.e., its CITY_ID value did not exist in the SRC_CITY table).

Note that you can view the entire record in this window. This means that you can instantly see which values are incorrect (i.e. the invalid CITY_ID value in the top record.)

You might also have noticed that the error message included is the one that was defined when setting up the “AGE > 21” constraint.

Now that the static controls have been set up, we are ready to move on to the implementation of our transformation interfaces

Interfaces for Data Integration

"Pop. TRG_CUSTOMER" Interface

Purpose

The purpose of our first 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. We wish to add the following information to the target table:

- The age range ("AGE_RANGE") – as defined in the "SRC_AGE_GROUP" flat file – that corresponds to the AGE attribute in the source table
- The last and first names of the customer sales rep. ("LAST_NAME" and "FIRST_NAME") – as defined in the "SRC_SALES_PERSON" file – corresponding 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 transformation, we wish to cleanse the data by verifying that all of the constraints are satisfied and by storing invalid rows in an error table rather that in our target database. In our case, 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 below.

Defining the Interface

The following is an overview of the data that we will be using in this transformation.

Target Datastore

Model: Sales Administration - HSQL

Datastore: TRG_CUSTOMER

Source Datastores

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

Joins

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
Customers and age brackets	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

Transformation rules

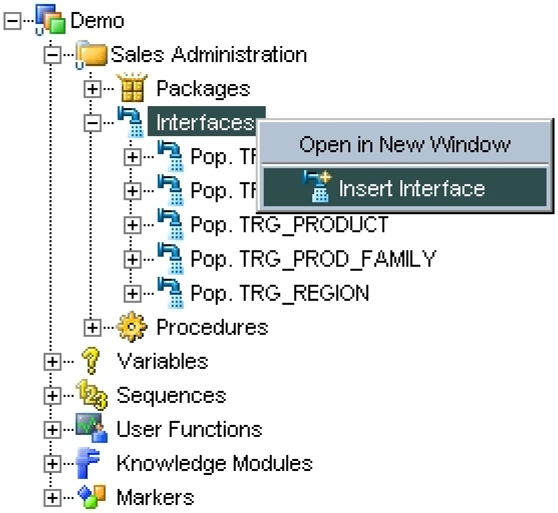
Target column	Origin	SQL rule	Execution location
CUST_ID	SRC_CUSTOMER.CUSTID	`SRC_CUSTOMER.CUSTID`	Source
DEAR	If SRC_CUSTOMER.DEAR = 0 then 'MR' If SRC_CUSTOMER.DEAR = 1 then 'MRS' else 'MS'	`CASEWHEN(SRC_CUSTOMER.DEAR=0 , 'MR', CASEWHEN( SRC_CUSTOMER.DEAR=1, 'MRS', 'MS') )`	Source
CUST_NAME	Concatenation 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

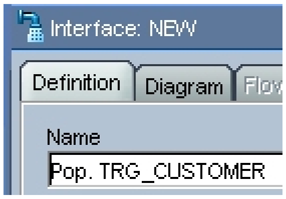
Steps to Follow

Insert a new interface

1. In the "Sales Administration" folder in the "Demo" project, right-click on Interfaces and choose **Insert Interface**



2. In the **Interface** window, enter the name of your interface (Pop. TRG_CUSTOMER) in the **Name** field

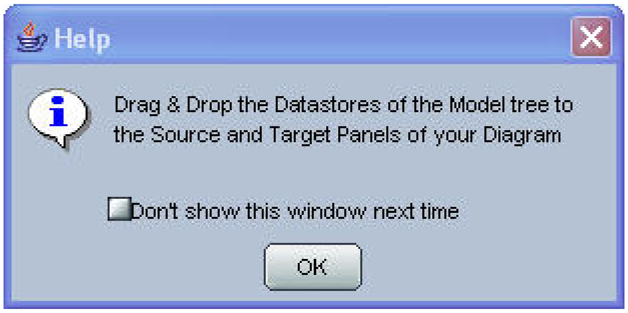


Name
Pop. TRG_CUSTOMER

3. At this point, you should enlarge your interface window for easy viewing.

Define the Target Table

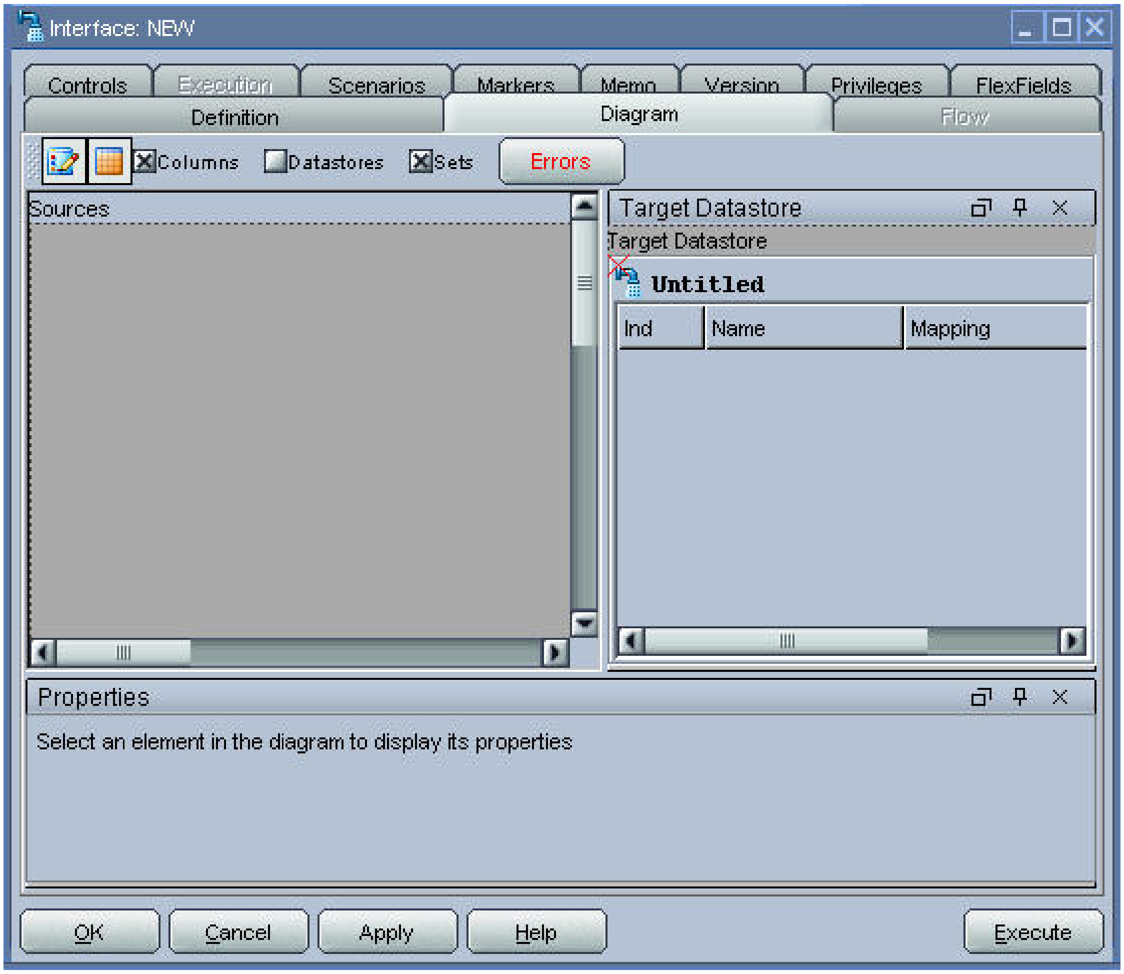
1. Click on the **Diagram** tab to enter your working environment. An information box will appear. You can simply click **OK** to close it.



Drag & Drop the Datastores of the Model tree to the Source and Target Panels of your Diagram

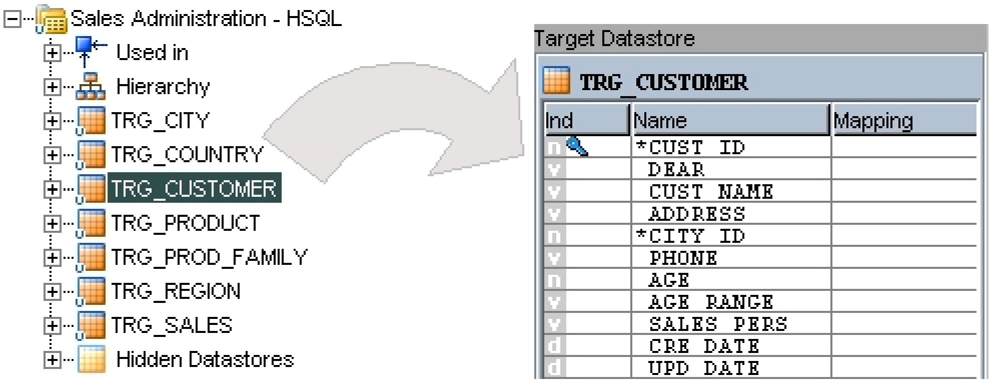
□ Don't show this window next time

2. Your interface **diagram** tab should look as follows:



Select an element in the diagram to display its properties

3. Drag the "TRG_CUSTOMER" table from the "Sales Administration - HSQL" model to the **Target datastore** area on the right of the **Diagram** tab.



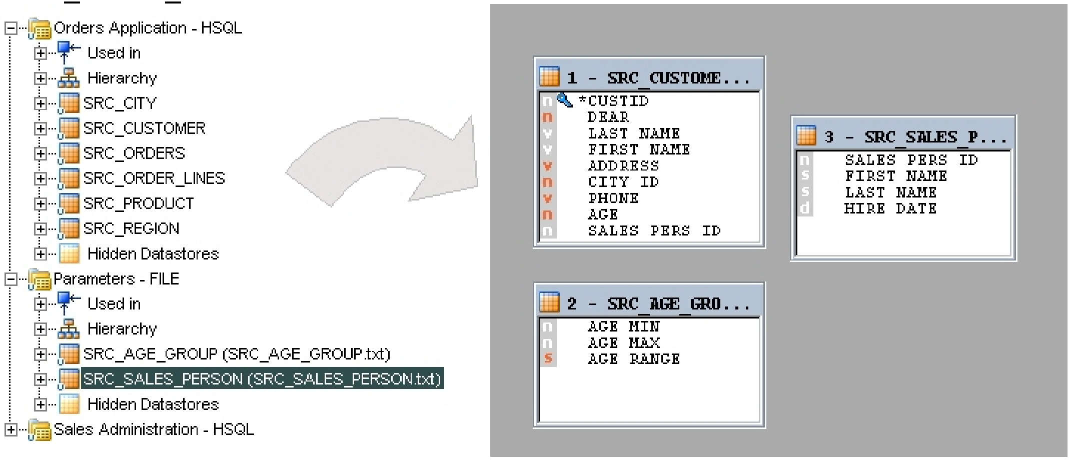
Target Datastore

TRG_CUSTOMER

Ind	Name	Mapping
n	*CUST_ID	
v	DEAR	
v	CUST_NAME	
v	ADDRESS	
n	*CITY_ID	
v	PHONE	
n	AGE	
v	AGE_RANGE	
v	SALES_PERS	
d	CRE_DATE	
d	UPD_DATE	

Define the Source Datastores

1. In the **Diagram** tab, drag the following source datastores into the **composition panel** (left part of the **diagram** tab):
   - "SRC_CUSTOMER" from the "Orders Application - HSQL" model
   - "SRC_AGE_GROUP" from the "Parameters - FILE" model
   - "SRC_SALES_PERSON" from the "Parameters - FILE" model



1 - SRC_CUSTOME...

n	*CUSTID
n	DEAR
v	LAST_NAME
v	FIRST_NAME
v	ADDRESS
n	CITY_ID
v	PHONE
n	AGE
n	SALES_PERS_ID

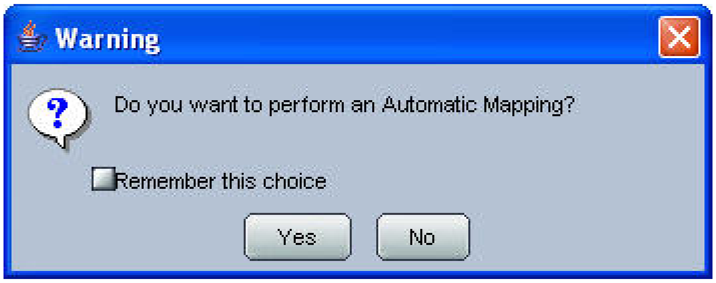
3 - SRC_SALES_P...

n	SALES_PERS_ID
s	FIRST_NAME
s	LAST_NAME
d	HIRE_DATE

2 - SRC_AGE_GRO...

n	AGE_MIN
n	AGE_MAX
s	AGE_RANGE

2. A popup window appears to confirm the use of automatic field to field mapping by Oracle Data Integrator when you drop a datastore source onto the composition panel. Click **Yes** to close this window.

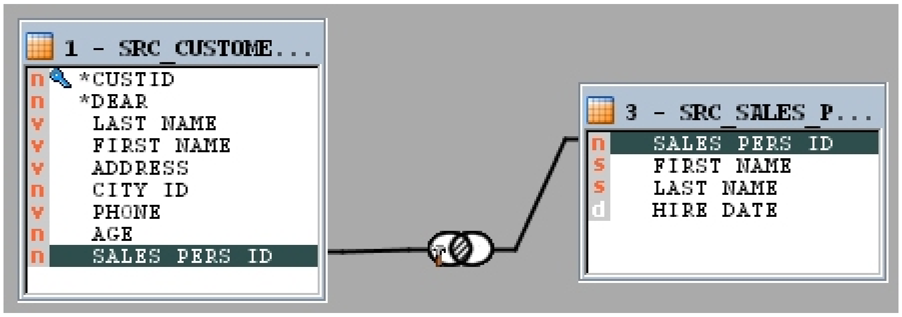

Warning

Do you want to perform an Automatic Mapping?

□ Remember this choice

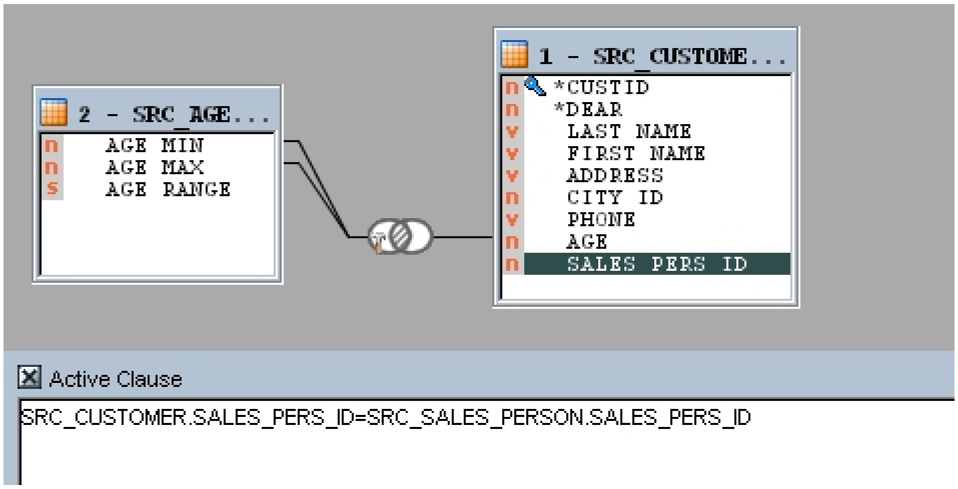
Define the Joins between the Source Datastores

1. In the **composition panel**, drag the "SALES_PERS_ID" column from the "SRC_CUSTOMER" table onto the "SALES_PERS_ID" column in the "SRC_SALES_PERSON" table. This will define the first join on the sales representative identifier.



2. Drag the "AGE" column from the "SRC_CUSTOMER" table onto the "AGE_MIN" column in the "SRC_AGE_GROUP" table.
3. Modify the **implementation** in the **Properties** panel as follows:
   - Replace "=" with "between"
   - Add "and" at the end of the text
   - Drag the "AGE_MAX" column from "SRC_AGE_GROUP" table into the text field.

4. You should have the following text: `SRC_CUSTOMER.AGE between SRC_AGE_GROUP.AGE_MIN and SRC_AGE_GROUP.AGE_MAX`
   This corresponds to a join between "SRC_CUSTOMER" and "SRC_AGE_GROUP"



☒ Active Clause

SRC_CUSTOMER.SALES_PERS_ID=SRC_SALES_PERSON.SALES_PERS_ID

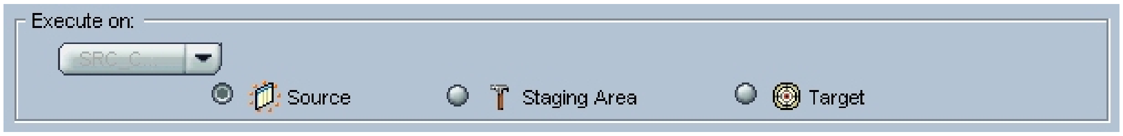
Note: If references were already defined in the models to link the source datastores, these references would have appeared automatically as joins in the source diagram.

Define the Transformation Rules

In your diagram's **Target datastore** area, columns with names that match their sources are mapped automatically (by matching column names). All you have to do in this case is to define the transformation rules for the following fields: CUSTID, DEAR, CUST_NAME, SALES_PERS, CRE_DATE and UPD_DATE.

The transformation rules are defined on each target column as follows:

1. Select the target column that you wish to load by clicking on its name.
2. In the **implementation** field in the bottom left, create the rule either by dragging the required columns from the source tables in the text field, or by opening the expression editor by clicking on the ✎ button.
3. Make sure that you select the environment in which your transformation will be executed. Select this environment by clicking on one of the radio buttons as shown below:



Execute on:

SRC_C...

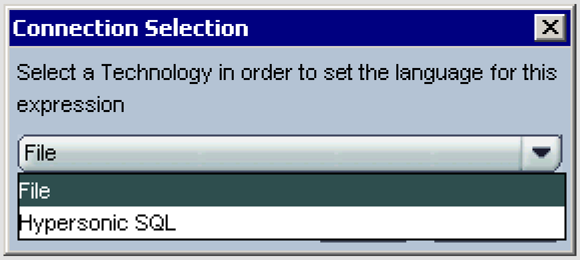
● Source ○ Staging Area ○ Target

Implement the following rules in the mappings:

- **CUST_ID** (executed on Source): drag the "SRC_CUSTOMER.CUSTID" column to the **implementation** field. This will simply map the source table's CUST_ID column name into our target table.
- **DEAR** (executed on Source): enter the following text: `CASEWHEN(SRC_CUSTOMER.DEAR=0, 'MR', CASEWHEN( SRC_CUSTOMER.DEAR=1, 'MRS', 'MS') )` You can drag the "SRC_CUSTOMER.DEAR" column each time you need it. Save your expression with ↲ and validate your syntax with ✓. This will map the source table's DEAR column as a string expression (0>'MR', 1>'MRS', 2>'MS')
- **CUST_NAME** (executed on Source): open the expression editor and create the following rule: `SRC_CUSTOMER.FIRST_NAME || ' ' || UCASE(SRC_CUSTOMER.LAST_NAME)` This will map the concatenated value of the first name and uppercase last name of each customer.

Caution: 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.

Note: When opening the expression editor, the following window may appear:



Connection Selection

Select a Technology in order to set the language for this expression

File

File

Hypersonic SQL

Select "Hypersonic SQL" and click **OK**. This window lets you select the SQL Language Syntax to be used by the expression Editor.

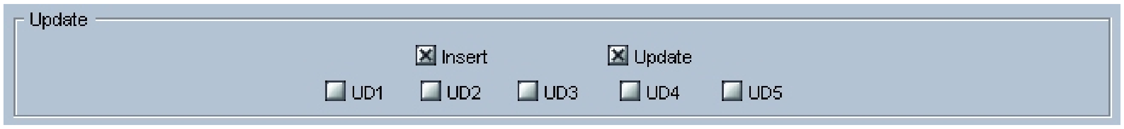
Note: If you were to execute this mapping on the target, the expression editor would give you the syntax for your target system (also Hypersonic SQL in this case).

- **SALES_PERS** (executed in the staging area): open the expression editor and create the following rule: `SRC_SALES_PERSON.FIRST_NAME || ' ' || UCASE(SRC_SALES_PERSON.LAST_NAME).`

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

Caution: 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** and **UPD_DATE** (both executed on Target): enter the `CURDATE()` function, check the "Active Mapping" box, and choose to execute the expression on the target. For CRE_DATE, you only want the mapping performed on Insert. To do this, scroll to the bottom of the mapping area and uncheck Update. For UPD_DATE, you only want the date where a record is updated. To do this, uncheck the Insert checkbox in the UPD_DATE mapping.



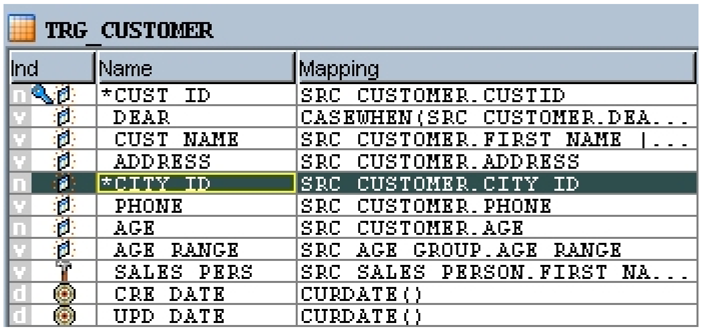
Update

☒ Insert ☒ Update

☐ UD1 ☐ UD2 ☐ UD3 ☐ UD4 ☐ UD5

Note: The ✓ button 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. In the event of an error, an error window will appear. Click on the **Details** button to display the error message sent by the source server.

Your transformation rules should now resemble the screen below.



TRG_CUSTOMER

Ind	Name	Mapping
n	*CUST ID	SRC CUSTOMER.CUSTID
v	DEAR	CASEWHEN(SRC CUSTOMER.DEA...
v	CUST NAME	SRC CUSTOMER.FIRST NAME |...
v	ADDRESS	SRC CUSTOMER.ADDRESS
n	*CITY ID	SRC CUSTOMER.CITY ID
v	PHONE	SRC CUSTOMER.PHONE
n	AGE	SRC CUSTOMER.AGE
v	AGE RANGE	SRC AGE GROUP.AGE RANGE
v	SALES PERS	SRC SALES PERSON.FIRST NA...
d	CRE DATE	CURDATE()
d	UPD DATE	CURDATE()

The icons on the left indicate the first letter of the data type in the target column (n: numeric, v: varchar, d: date). The other icons indicate the expressions' execution location:

▯	source
T	staging area
◎	target

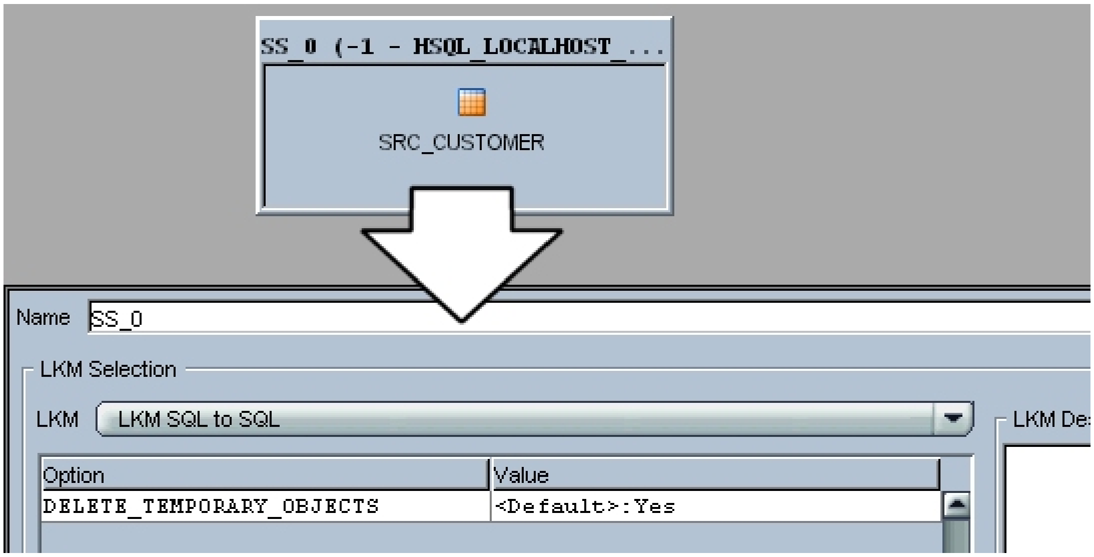
Note: If the mapping **Ind** indicator displays a ⛔ icon, you have an error in your mapping. In this case, select the target field at fault and verify your input. Alternately, you can click the **Errors** button at the top of the **Diagram** tab, then click on the **Details** button in the displayed window for more detailed information on the type of error.

Choose the Data Loading Strategies (LKM)

In the **Flow** tab, Oracle Data Integrator indicates the various steps it must perform to execute your interface.

You have to choose the way in which to proceed (**Loading Knowledge Module** - LKM) to retrieve the data from the "SRC_AGE_GROUP", "SRC_SALES_PERSON" files and from the "SRC_CUSTOMER" table in your source environment.

1. Click on the **Source Set** (SS_x) that corresponds to the loading of the "SRC_AGE_GROUP" file and choose the "**LKM File to SQL**" module from the list of loading knowledge modules.

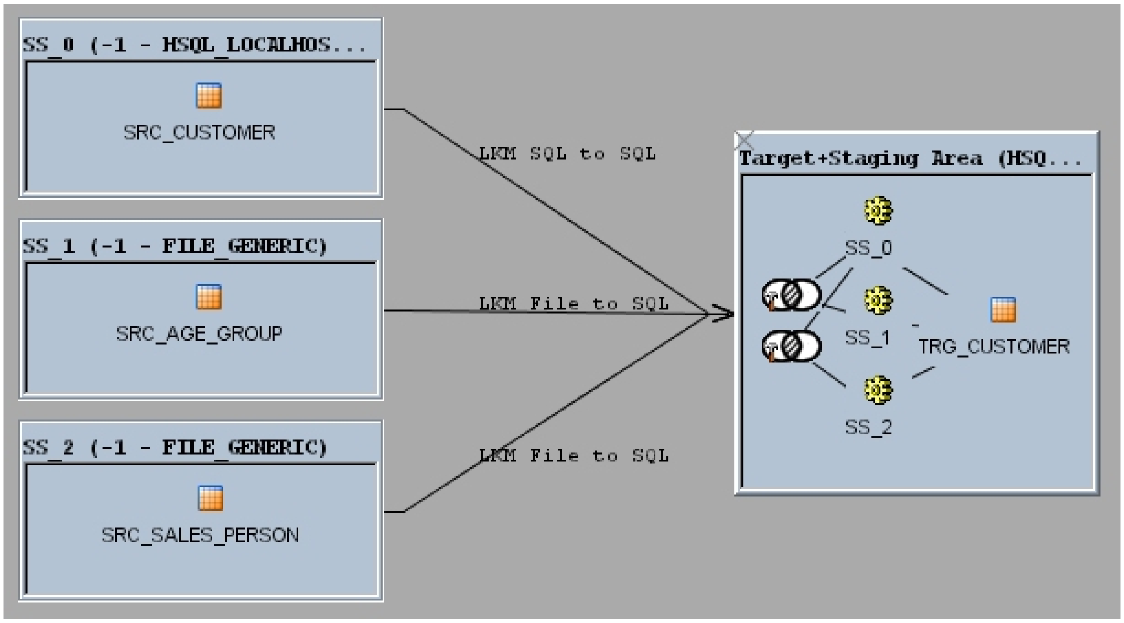


Name: SS_0

LKM: LKM SQL to SQL

Option	Value
DELETE_TEMPORARY_OBJECTS	<Default>:Yes

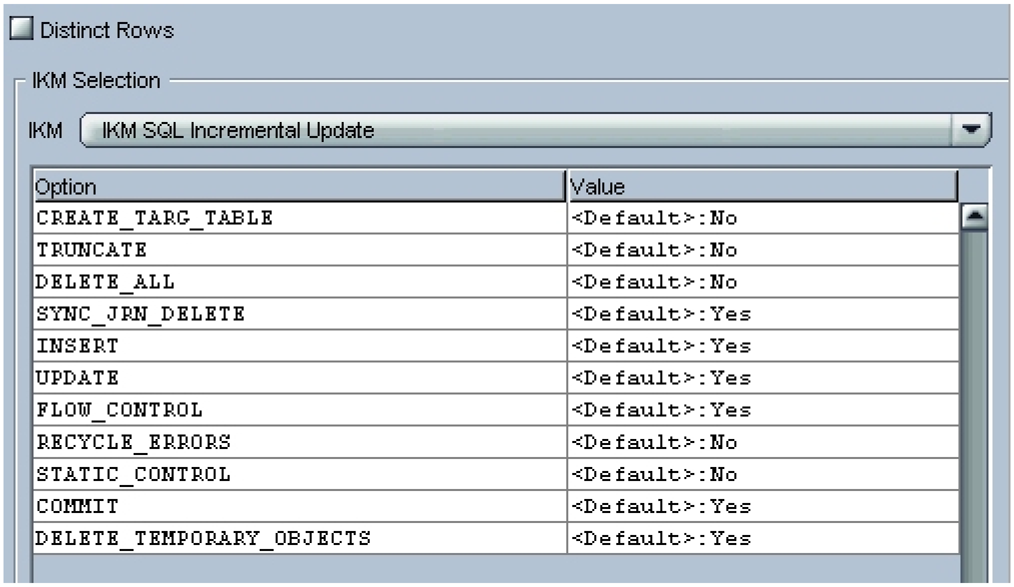
2. Click on the second **Source Set** (SS_x) that corresponds to the loading of the "SRC_SALES_PERSON" file and choose the "**LKM File to SQL**" module from the list of loading knowledge modules.
3. Click on the last **Source Set** (SS_x) that corresponds to the loading of the "SRC_CUSTOMER" table and choose the "**LKM SQL to SQL**" module from the list of loading knowledge modules.
4. Your **Flow** screen should now look like the image below:



Choose the Data Integration Strategy (IKM)

Upon completion of the loading phase, you must choose the strategy to adopt for the integration of the data into the target table. To do this, you must specify the Integration Knowledge Module (IKM) that will be used.

1. Click on the **Target+Staging Area** object and choose the "**IKM SQL Incremental Update**" knowledge module.



□ Distinct Rows

IKM Selection

IKM: IKM SQL Incremental Update

Option	Value
CREATE_TARG_TABLE	<Default>:No
TRUNCATE	<Default>:No
DELETE_ALL	<Default>:No
SYNC_JRN_DELETE	<Default>:Yes
INSERT	<Default>:Yes
UPDATE	<Default>:Yes
FLOW_CONTROL	<Default>:Yes
RECYCLE_ERRORS	<Default>:No
STATIC_CONTROL	<Default>:No
COMMIT	<Default>:Yes
DELETE_TEMPORARY_OBJECTS	<Default>:Yes

2. In the knowledge module **options**, leave the default values. They should be as above.

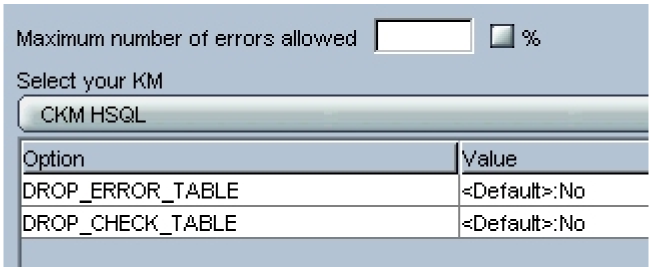
These options enable to control certain aspects of the integration strategy. For example, with the `FLOW_CONTROL` option, you have ordered Oracle Data Integrator to trigger the flow control operations of the data before inserting it into the target table.

Note: When you are choosing a knowledge module, you may notice that only few of them are available. To use all of the knowledge modules that are included with Oracle Data Integrator, you must import them into your project. This step is not required for the purposes of this guide. For more information on doing this for your own projects, please refer to the Oracle Data Integrator User Guide, which is part of the online help of the product documentation.

Choose the Data Control Strategy

Up until now you have only 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.

1. In the **Controls** tab, choose the "CKM HSQL" Check Knowledge Module.



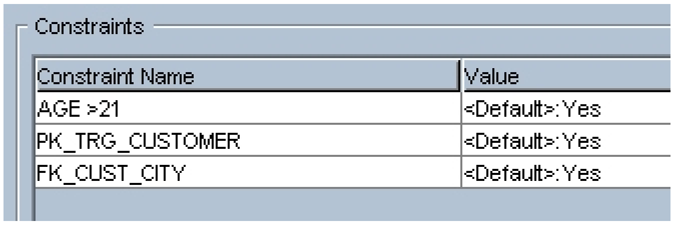
Maximum number of errors allowed □ %

Select your KM

CKM HSQL

Option	Value
DROP_ERROR_TABLE	<Default>:No
DROP_CHECK_TABLE	<Default>:No

2. Set the constraints that you wish to verify ("AGE > 21", , 'PK_TRG_CITY' and "FK_CUST_CITY") to ‘Yes’, as is seen below:



Constraints

Constraint Name	Value
AGE >21	<Default>:Yes
PK_TRG_CUSTOMER	<Default>:Yes
FK_CUST_CITY	<Default>:Yes

You are now ready to execute your interface.

Execute the Interface

1. Click the **Execute** button in the **Interface** window to run your interface.
2. A window prompting to save the interface appears. Click **Yes**.
3. Click **OK** in the **Execution** and **Information** windows.

Oracle Data Integrator now automatically generates the SQL code and starts an execution session.

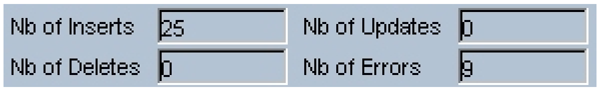
Follow the Interface’s Execution in Operator

1. To follow the result of your execution in the **Operator**, click on the icon in your menu bar.
2. The **Operator** window that subsequently opens (and can be refreshed using the button if the window is already open) details the execution process tasks.

Interpreting the Results

The Processed Records

1. In **Operator**, open the window for the "Pop. TRG_CUSTOMER" **step** by double-clicking on the corresponding icon in the execution tree. A new window opens to the right of the screen.
2. In the **Execution** tab, you will notice that the loading of the "TRG_CUSTOMER" table produced 25 inserts and isolated 9 errors in an error table.



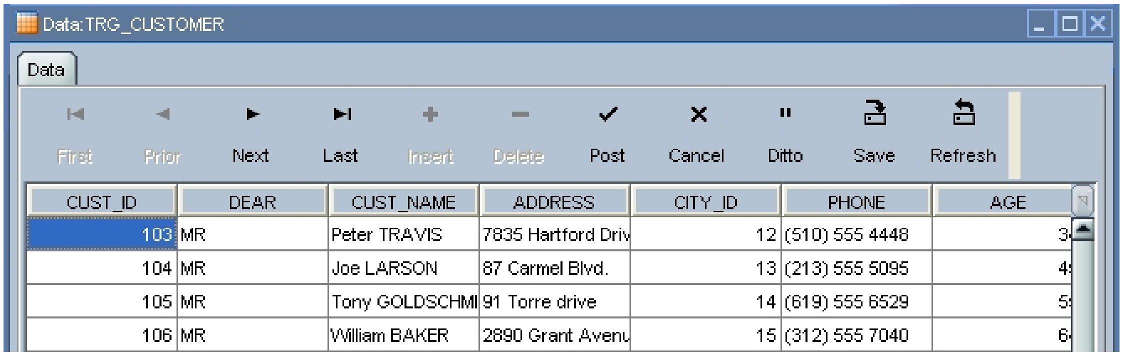
Nb of Inserts	25	Nb of Updates	0
Nb of Deletes	0	Nb of Errors	9

The Resulting Data

You can view the data resulting from your interface in the **Designer** module.

1. Expand the "Sales Administration - HSQL" model and select the "TRG_CUSTOMER" table.

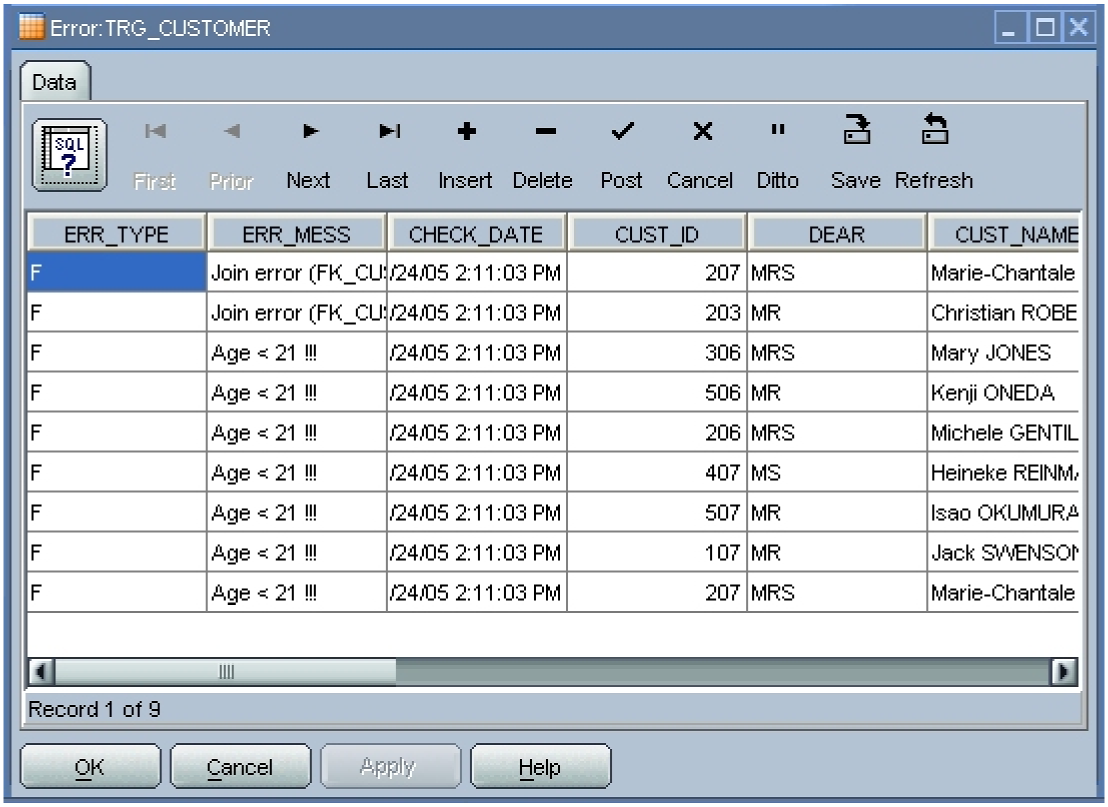
2. Right-click and choose **Data** to view the data in the target table.



CUST_ID	DEAR	CUST_NAME	ADDRESS	CITY_ID	PHONE	AGE
103	MR	Peter TRAVIS	7835 Hartford Driv	12	(510) 555 4448	3
104	MR	Joe LARSON	87 Carmel Blvd.	13	(213) 555 5095	4
105	MR	Tony GOLDSCHMI	91 Torre drive	14	(619) 555 6529	5
106	MR	William BAKER	2890 Grant Avenu	15	(312) 555 7040	6

Invalid Records and Incorrect Data

The interface you have executed has identified and isolated **9** invalid records in an error table that was automatically created for you. You can access the invalid records by selecting the target table (TRG_CUSTOMER) from your model, then choosing **Control > Errors** from the popup menu.



ERR_TYPE	ERR_MESS	CHECK_DATE	CUST_ID	DEAR	CUST_NAME
F	Join error (FK_CU	/24/05 2:11:03 PM	207	MRS	Marie-Chantale
F	Join error (FK_CU	/24/05 2:11:03 PM	203	MR	Christian ROBE
F	Age < 21 !!!	/24/05 2:11:03 PM	306	MRS	Mary JONES
F	Age < 21 !!!	/24/05 2:11:03 PM	506	MR	Kenji ONEDA
F	Age < 21 !!!	/24/05 2:11:03 PM	206	MRS	Michele GENTIL
F	Age < 21 !!!	/24/05 2:11:03 PM	407	MS	Heineke REINM
F	Age < 21 !!!	/24/05 2:11:03 PM	507	MR	Isao OKUMURA
F	Age < 21 !!!	/24/05 2:11:03 PM	107	MR	Jack SWENSON
F	Age < 21 !!!	/24/05 2:11:03 PM	207	MRS	Marie-Chantale

Record 1 of 9

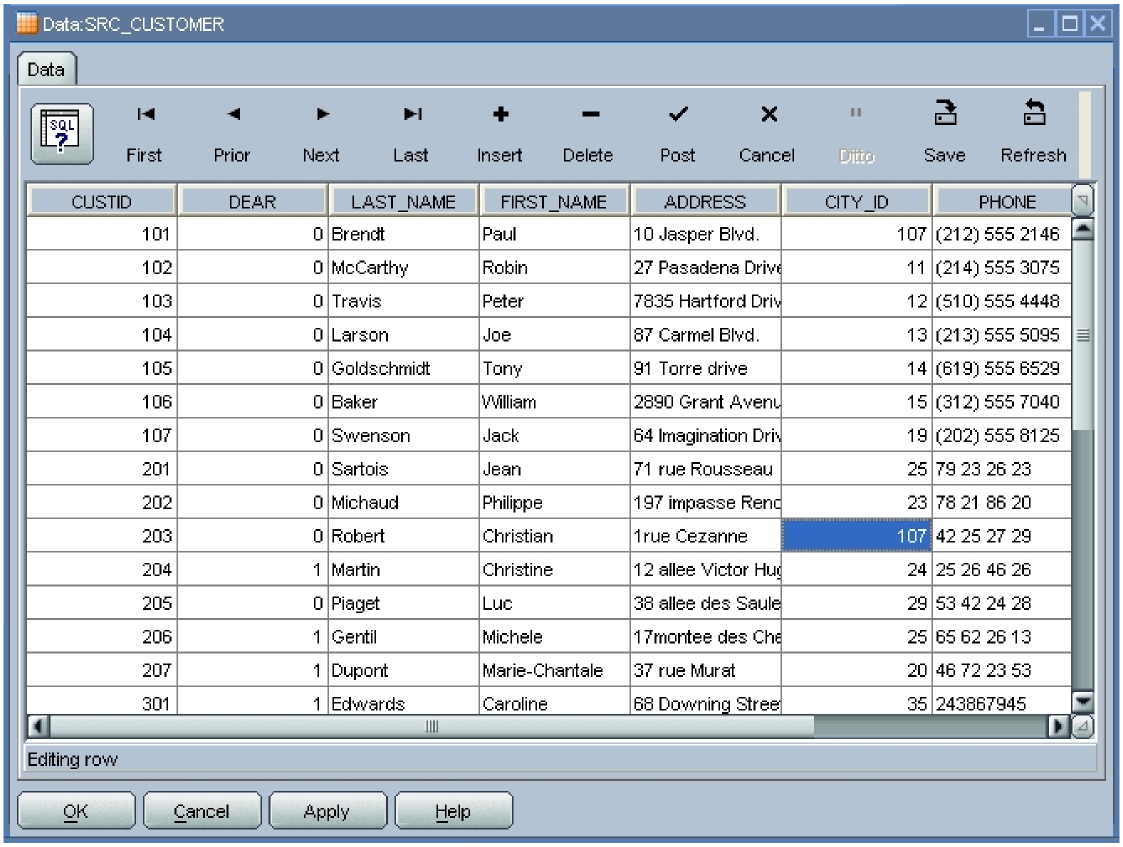
From here, you can see that the interface rejected:

- **2** records that did not satisfy the "FK_CUST_CITY" constraint (i.e. have a city number that does not exist in the table of cities).
- **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 not integrated into the target table.

Correcting Invalid data

1. Expand the tree structure for the "Orders Application - HSQL" model
2. Right-click on "SRC_CUSTOMER" and select **Data**
3. Search on the screen for the client row having a CUSTID equal to "203". You can sort the table by clicking on the column headers. If the customer 203 is not visible, click the **Refresh** button to refresh the display.
4. Double-click on the value of the "CITY_ID" column for this customer in order to modify it. The new value has to be 107 (rather than 208)



CUSTID	DEAR	LAST_NAME	FIRST_NAME	ADDRESS	CITY_ID	PHONE
101	0	Brendt	Paul	10 Jasper Blvd.	107	(212) 555 2146
102	0	McCarthy	Robin	27 Pasadena Drive	11	(214) 555 3075
103	0	Travis	Peter	7835 Hartford Driv	12	(510) 555 4448
104	0	Larson	Joe	87 Carmel Blvd.	13	(213) 555 5095
105	0	Goldschmidt	Tony	91 Torre drive	14	(619) 555 6529
106	0	Baker	William	2890 Grant Avenu	15	(312) 555 7040
107	0	Swenson	Jack	64 Imagination Driv	19	(202) 555 8125
201	0	Sartois	Jean	71 rue Rousseau	25	79 23 26 23
202	0	Michaud	Philippe	197 impasse Renc	23	78 21 86 20
203	0	Robert	Christian	1rue Cezanne	107	42 25 27 29
204	1	Martin	Christine	12 allee Victor Hug	24	25 26 46 26
205	0	Piaget	Luc	38 allee des Saule	29	53 42 24 28
206	1	Gentil	Michele	17montee des Che	25	65 62 26 13
207	1	Dupont	Marie-Chantale	37 rue Murat	20	46 72 23 53
301	1	Edwards	Caroline	68 Downing Stree	35	243867945

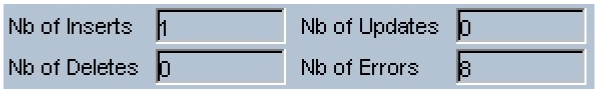
Editing row

5. Click on **Apply** and then **OK**
6. Re-execute the "Pop. TRG_CUSTOMER" interface by selecting the **Execute** option in the contextual menu.

The Records that were Processed

1. In **Operator**, click on the to refresh the screen.

2. Edit the "Pop. TRG_CUSTOMER" **step** window by double-clicking on the corresponding icon in the execution tree. A new window will open on the right side of the screen.
3. In the **Execution** tab, you will notice that the loading of the "TRG_CUSTOMER" table produced 1 insertion (the record that you have just modified) and isolated 8 invalid records in the error table.



Nb of Inserts	1	Nb of Updates	0
Nb of Deletes	0	Nb of Errors	8

Interfaces for Data Integration (Continued)

"Pop. TRG_SALES" Interface

Purpose

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 below.

Defining the Interface

Target Datastore

Model: Sales Administration - HSQL

Datastore: TRG_SALES

Source datastores

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

Joins

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

Filters

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

Transformation rules

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

Steps to Follow

Insert a New Interface

1. In the "Sales Administration" folder in the "Demo" project, right-click on Interfaces and choose **Insert Interface**
2. In the **Interface** window, enter the name of your interface (Pop. TRG_SALES) in the **Name** field
3. At this point, you should enlarge your interface window for easy viewing.

Define the Target Table

1. In the **Diagram** tab, drag the "TRG_SALES" table from the "Sales Administration - HSQL" model to the **Target datastore** area on the right of the **Diagram** tab.

Define the Source Datastores

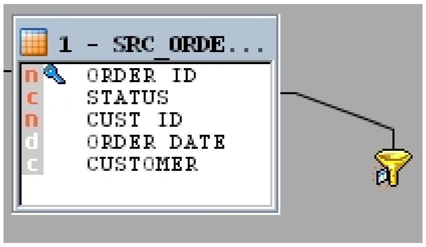
1. In the **Diagram** tab, drag the following source datastores into the **composition panel** (left hand side of the **diagram** tab):
   - "SRC_ORDERS" from the "Orders Application - HSQL" model
   - "SRC_ORDER_LINES" from the "Orders Application - HSQL" model

Define the Joins between the Source Datastores

1. In the **composition panel**, drag the "ORDER_ID" column in "SRC_ORDERS" onto the "ORDER_ID" column in "SRC_ORDER_LINES". This defines a join clause on the order number.

Define the Order Filter

1. In the **Diagram tab**, drag the "STATUS" column in "SRC_ORDERS" onto the **composition panel**.
2. Modify the **implementation** of the filter rule by typing: `SRC_ORDERS.STATUS = 'CLO'`



Define the Transformation Rules

In the diagram's **Target datastore** area, columns with names that match their sources are mapped automatically (using column name matching). All that remains to do is to define the mapping rules for the following fields: FIRST_ORD_ID, FIRST_ORD_DATE, LAST_ORD_ID, LAST_ORD_DATE, QTY, AMOUNT and PROD_AVG_PRICE.

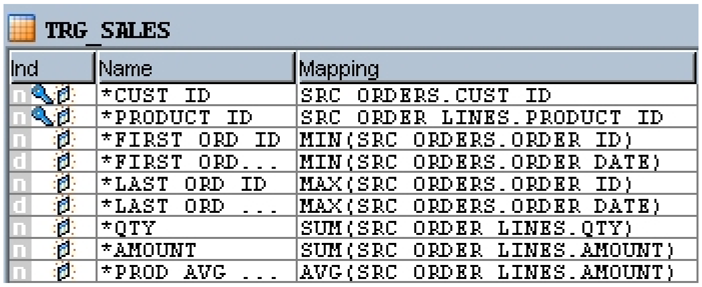
As with the previous exercise, each column's name is represented in a table where each row corresponds to a column name. The transformation rules are defined for each field in the target table as follows:

1. Select the field in the target table that you wish to load
2. In the **Implementation** field on the bottom left side of your diagram, create the rule either by dragging the required columns from the diagram of source tables into the text field, or by calling the expression editor by clicking on the ✎ button.

Implement the following rules:

- FIRST_ORD_ID: Drag-and-drop the "SRC_ORDERS.ORDER_ID" column into the **Implementation** field. Change the rule to:
  `MIN(SRC_ORDERS.ORDER_ID)`
  This will map 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-and-drop the "SRC_ORDERS.ORDER_DATE" column into the **Implementation**. Change the rule to:
  `MIN(SRC_ORDERS.ORDER_DATE)`
  This will map 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. Change the rule to:
  `MAX(SRC_ORDERS.ORDER_ID)`
  This will map 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-and-drop the "SRC_ORDERS.ORDER_DATE" column into the **Implementation** field. Change the rule to:
  `MAX(SRC_ORDERS.ORDER_DATE)`
  This will map the maximum value of the ORDER_DATE column in your SRC_ORDERS table to the LAST_ORD_DATE column in your target table
- QTY: Change the rule to:
  `SUM(SRC_ORDER_LINES.QTY)`
  This will map the sum of the product quantities to the QTY column in your target table
- AMOUNT: Change the rule to:
  `SUM(SRC_ORDER_LINES.AMOUNT)`
  This will map the sum of the product prices to the AMOUNT column in your target table
- PROD_AVG_PRICE: Drag-and-drop the "SRC_ORDERLINES.AMOUNT" column into the **Implementation** field. Change the rule to:
  `AVG(SRC_ORDER_LINES.AMOUNT)`
  This will map the average of the product prices to the PROD_AVG_PRICE column in your target table

Your target area should now look like this:



TRG_SALES

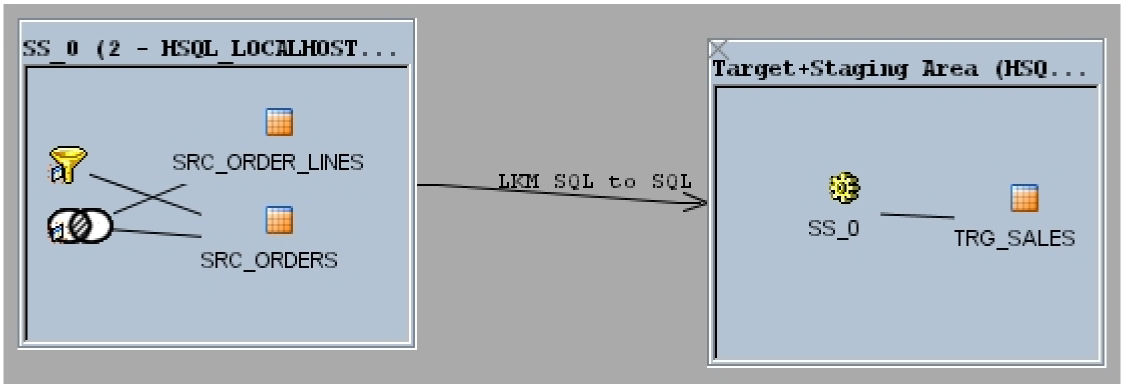
Ind	Name	Mapping
n 🔑	*CUST ID	SRC ORDERS.CUST ID
n 🔑	*PRODUCT ID	SRC ORDER LINES.PRODUCT ID
n	*FIRST ORD ID	MIN(SRC ORDERS.ORDER ID)
d	*FIRST ORD...	MIN(SRC ORDERS.ORDER DATE)
n	*LAST ORD ID	MAX(SRC ORDERS.ORDER ID)
d	*LAST ORD ...	MAX(SRC ORDERS.ORDER DATE)
n	*QTY	SUM(SRC ORDER LINES.QTY)
n	*AMOUNT	SUM(SRC ORDER LINES.AMOUNT)
n	*PROD AVG ...	AVG(SRC ORDER LINES.AMOUNT)

Choose the Data Loading Strategy (LKM)

In the **Flow** tab, Oracle Data Integrator indicates the various steps that must be performed in order to execute your interface.

You must choose the way in which to load the result of the orders and order line aggregates into your target environment with a **Loading Knowledge Module** (LKM)

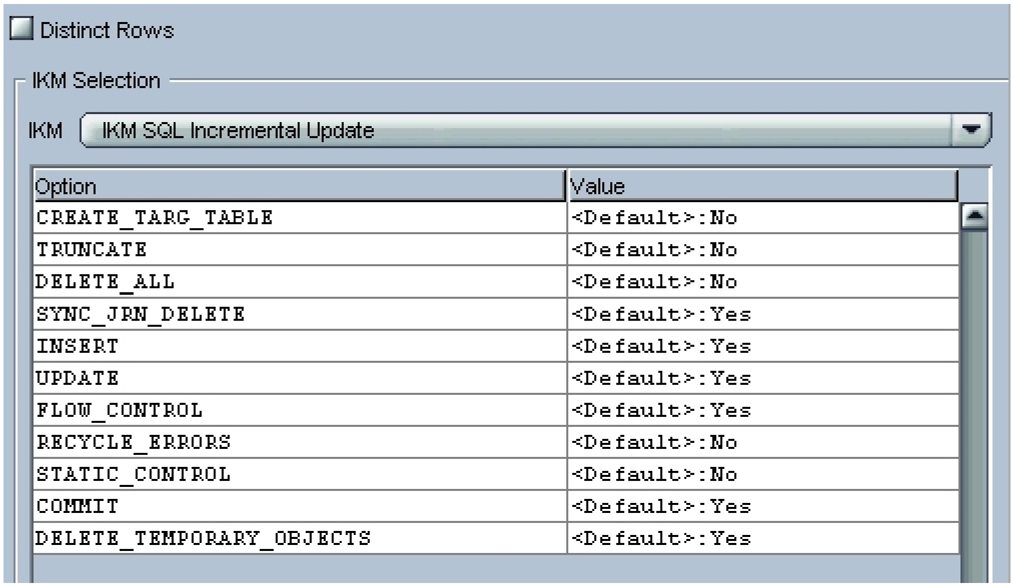
1. Click on the (SS_0) **Source Set** (SS_0) that corresponds to the loading of the order line's filtered aggregate results. Choose the "**LKM SQL to SQL**" module from the list of loading knowledge modules.
2. Your Flow screen should now look like this:



Choose the Data Integration Strategy (IKM)

You must now choose the strategy for integrating your data into the target table. To do this, you must specify the Integration Knowledge Module (IKM) that will be used.

1. Click on the **Target+Staging Area** object and choose the "**IKM SQL Incremental Update**" knowledge module



□ Distinct Rows

IKM Selection

IKM: IKM SQL Incremental Update

Option	Value
CREATE_TARG_TABLE	<Default>:No
TRUNCATE	<Default>:No
DELETE_ALL	<Default>:No
SYNC_JRN_DELETE	<Default>:Yes
INSERT	<Default>:Yes
UPDATE	<Default>:Yes
FLOW_CONTROL	<Default>:Yes
RECYCLE_ERRORS	<Default>:No
STATIC_CONTROL	<Default>:No
COMMIT	<Default>:Yes
DELETE_TEMPORARY_OBJECTS	<Default>:Yes

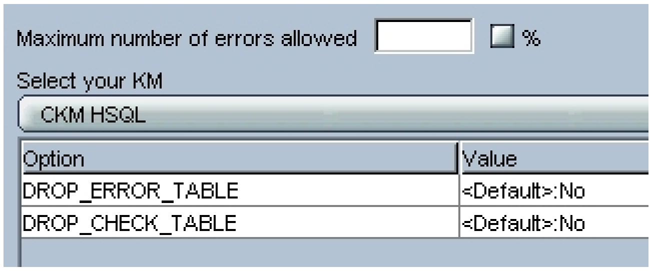
2. In the knowledge module options, leave the default values as above.

As before, Oracle Data Integrator will cleanse the data in your flow before inserting that data into your target table.

Choose the Data Control Strategy

Up until now, you have only specified the data flow from the source to the target. You must now define how to check your data (CKM) and the constraints or conditions that have to be satisfied before integrating the data.

1. In the **Control** tab, choose the "CKM HSQL" Check Knowledge Module:

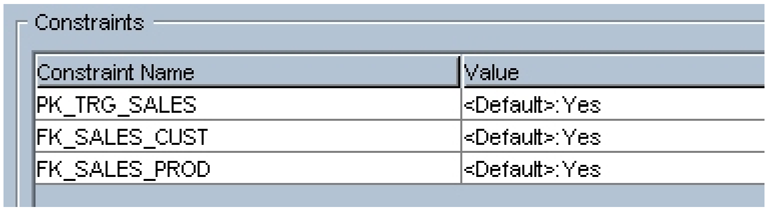


Maximum number of errors allowed □ %

Select your KM: CKM HSQL

Option	Value
DROP_ERROR_TABLE	<Default>:No
DROP_CHECK_TABLE	<Default>:No

2. Define the constraints to be verified: "PK_TRG_SALES", "FK_SALES_CUST" and "FK_SALES_PROD"



Constraints

Constraint Name	Value
PK_TRG_SALES	<Default>:Yes
FK_SALES_CUST	<Default>:Yes
FK_SALES_PROD	<Default>:Yes

You are now ready to execute your interface.

Execute the Interface

1. Click the **Execute** button in the **Interface** window to run your interface.
2. A window prompting to save the interface appears. Click **Yes.**
3. Click **OK** in the **Execution** and **Information** windows.

Oracle Data Integrator now automatically generates the SQL code and starts an execution session.

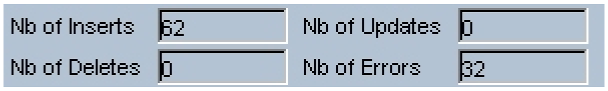
Follow the Interface’s Execution in Operator

1. To follow the result of your execution in the **Operator**, click the 🔍 icon in your menu bar.
2. The **Operator** window that subsequently opens (and can be refreshed using the 🔄 button if the window is already open) details the execution process tasks.

Interpreting the Results

The Processed Records

1. In **Operator**, open the window for the "Pop. TRG_SALES" **step** by double-clicking on the corresponding icon in the execution tree. A new window opens to the right of the screen.
2. In the **Execution tab**, you will notice that the loading of the "TRG_SALES" table produced **62** inserts and isolated **32** invalid records in an error table.

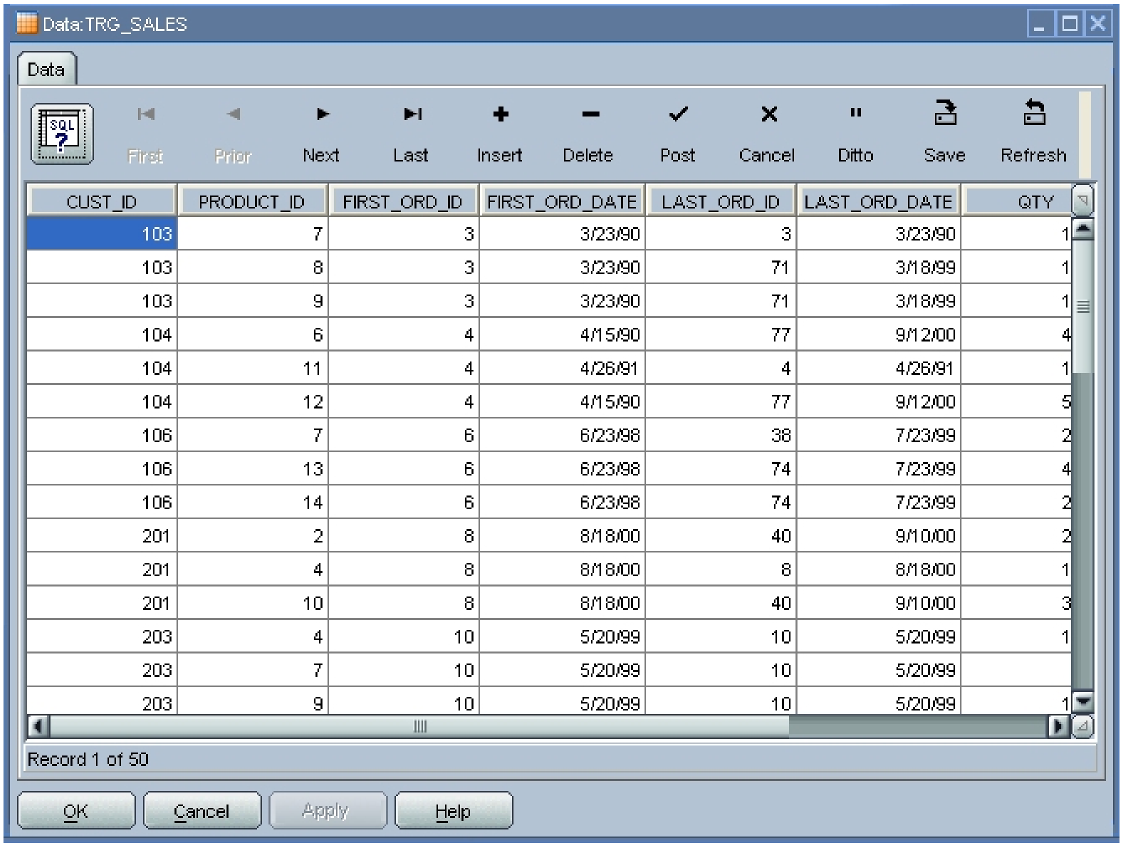


Nb of Inserts	62	Nb of Updates	0
Nb of Deletes	0	Nb of Errors	32

The Resulting Data

You can view the data that results from your interface in the **Designer** module.

1. Expand the "Sales Administration - HSQL" model and select the "TRG_SALES" table.
2. Right-click and choose **Data**, to view the data in the target table:

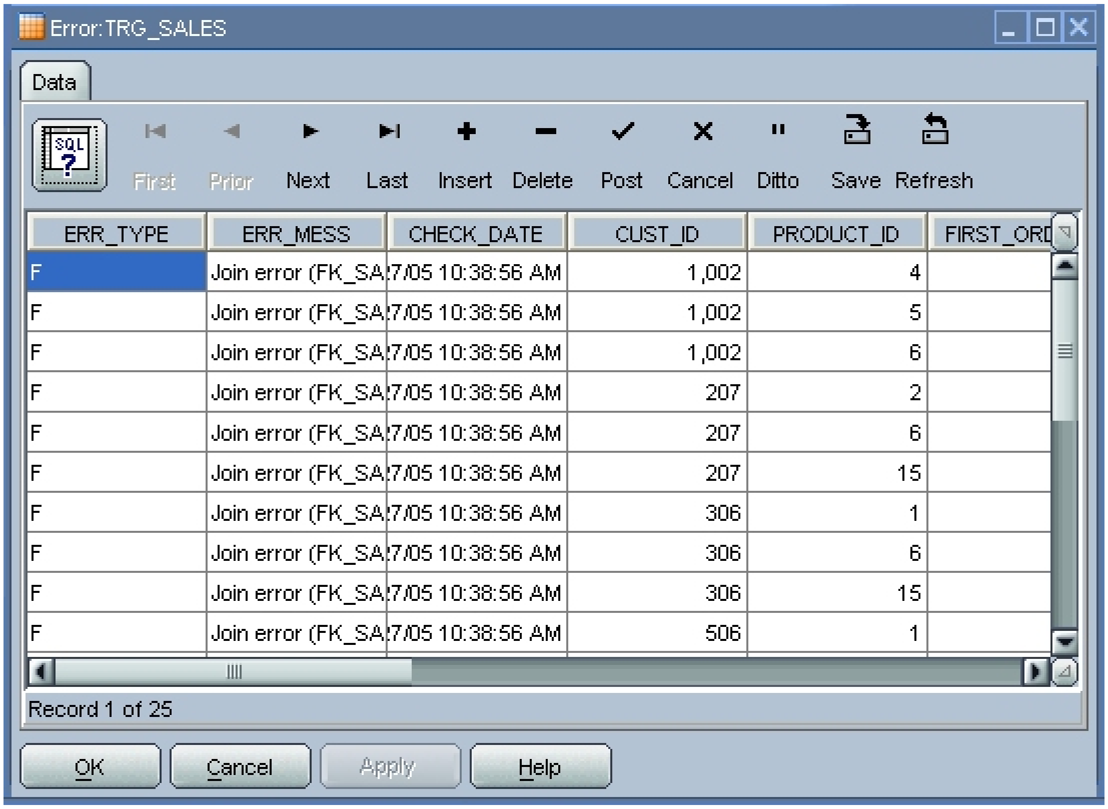


CUST_ID	PRODUCT_ID	FIRST_ORD_ID	FIRST_ORD_DATE	LAST_ORD_ID	LAST_ORD_DATE	QTY
103	7	3	3/23/90	3	3/23/90	1
103	8	3	3/23/90	71	3/18/99	1
103	9	3	3/23/90	71	3/18/99	1
104	6	4	4/15/90	77	9/12/00	4
104	11	4	4/26/91	4	4/26/91	1
104	12	4	4/15/90	77	9/12/00	5
106	7	6	6/23/98	38	7/23/99	2
106	13	6	6/23/98	74	7/23/99	4
106	14	6	6/23/98	74	7/23/99	2
201	2	8	8/18/00	40	9/10/00	2
201	4	8	8/18/00	8	8/18/00	1
201	10	8	8/18/00	40	9/10/00	3
203	4	10	5/20/99	10	5/20/99	1
203	7	10	5/20/99	10	5/20/99	
203	9	10	5/20/99	10	5/20/99	1

Record 1 of 50

Invalid Records

The interface you executed isolated **32** invalid records in an error table which was created for you automatically. You can access these records by selecting your target table (TRG_SALES) from your model, then choosing **Control > Errors** in the popup menu.



ERR_TYPE	ERR_MESS	CHECK_DATE	CUST_ID	PRODUCT_ID	FIRST_ORI
F	Join error (FK_SA	7/05 10:38:56 AM	1,002	4	
F	Join error (FK_SA	7/05 10:38:56 AM	1,002	5	
F	Join error (FK_SA	7/05 10:38:56 AM	1,002	6	
F	Join error (FK_SA	7/05 10:38:56 AM	207	2	
F	Join error (FK_SA	7/05 10:38:56 AM	207	6	
F	Join error (FK_SA	7/05 10:38:56 AM	207	15	
F	Join error (FK_SA	7/05 10:38:56 AM	306	1	
F	Join error (FK_SA	7/05 10:38:56 AM	306	6	
F	Join error (FK_SA	7/05 10:38:56 AM	306	15	
F	Join error (FK_SA	7/05 10:38:56 AM	506	1	

Record 1 of 25

From here, you can see that the interface rejected:

- **31** records in violation of the "FK_SALES_CUST" constraint (i.e. have a customer number that does not exist in the table of customers)
- **1** record in violation of the "FK_SALES_PROD" constraint (i.e. 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.

Operating

Introduction to Operating

The goal of Oracle Data Integrator is to automate information exchange between applications whether for full Enterprise Application Integration (EAI) or simply to populate different tables in a Data Warehouse.

For this it is necessary to sequence the execution of the different processes (interfaces, procedures, etc.) and to produce a production scenario containing the ready-to-use code for each of these processes.

Packages

A package is a pre-defined sequence of steps, designed to be executed in order. There are many types of steps, such as:

- Interfaces
- Procedures
- Actions/evaluations on variables
- Actions on models, sub-models or datastores
- OS Commands
- Oracle Data Integrator Tools

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

In the **"Load Sales Administration" Package** section, you will create and run a package that includes interfaces that you developed as well as some that came included in the "Demo" project.

Scenarios

A scenario is the partially-generated code (SQL, shell, etc) for the objects (interfaces, procedures, etc.) contained in a package.

Note: Once generated, the scenario is stored inside the Work repository. The scenario can be exported then re-imported into another Repository for use in one or more different contexts.

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.

The scenario's properties are displayed in the **Scenario** window’s **Scenario** window in the **Designer** module or via the **Operator** module. Operation of a scenario takes place from the same window or from an operating system command window. In the Designer module, scenarios are grouped under their respective packages.

For the purposes of this Getting Started guide, you will generate the scenario **"Load Sales Administration"** from a package that we will build, and run it from Oracle Data Integrator and later from a command prompt.

"Load Sales Administration" Package

Objective

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

Developments Provided with Oracle Data Integrator

In addition to the two interfaces you have developed, a number of developments are provided with the demo repository:

Interfaces

- **Pop. TRG_COUNTRY**: an interface that populates the TRG_COUNTRY table
- **Pop. TRG_REGION**: an interface that populates the TRG_REGION table
- **Pop. TRG_CITY**: an interface that populates the TRG_CITY table
- **Pop. TRG_PROD_FAMILY**: an interface that populates the TRG_PROD_FAMILY table
- **Pop. TRG_PRODUCT**: an interface populates the TRG_PRODUCT table

Procedures

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

Problem Analysis

In order to load the "Sales Administration" application correctly (in accordance with the referential integrity constraints), we must execute the tasks 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

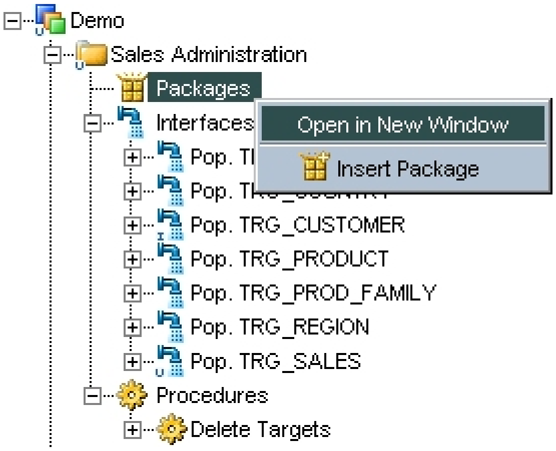
6. Load the TRG_PRODUCT table with the "Pop. TRG_PRODUCT" interface
7. Load the TRG_CUSTOMER table with the "Pop. TRG_CUSTOMER" interface
8. Load the TRG_SALES table with the "Pop. TRG_SALES" interface

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

Steps

Creation of the Package

1. Select the **Package** object, right-click and select **insert package**



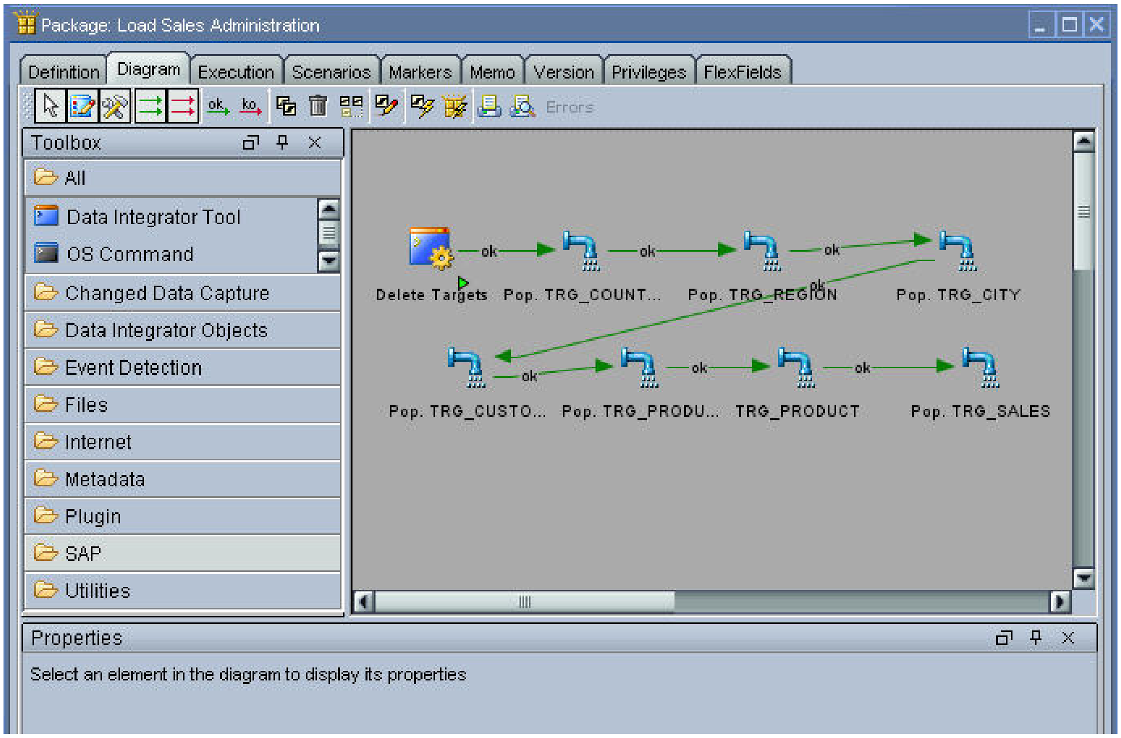
2. In the **Name** field type "Load Sales Administration".
3. Click **Apply** to save this empty package.

Define the Steps of the Package

1. In the **Package** window, click on the **Diagram** tab
2. To define the steps in the package, select each component that you wish to integrate into the package from the **Projects** tree and drag-and-drop it into the **Diagram.**
   These components appear as steps in the package. They are not sequenced yet.
3. To indicate the first step to execute in the package (the *Delete Target* procedure), select and right-click this step, then select **First Step** from the contextual menu. A small green arrow will appear on this step.
4. Select the **Next Step on Success** ok→icon in the diagram toolbar.
5. To sequence the steps with this tool, click on one step, then click on the step that should be executed next.
   Repeat this operation to sequence all the steps, to sequence developments as below:
   1. Delete Targets (First Step)
   2. Pop. TRG_COUNTRY
   3. Pop. TRG_REGION
   4. Pop. TRG_CITY
   5. Pop. TRG_CUSTOMER

6. Pop. TRG_PROD_FAMILY
7. Pop. TRG_PRODUCT
8. Pop. TRG_SALES

6. The resulting package should look as follows:



Select an element in the diagram to display its properties

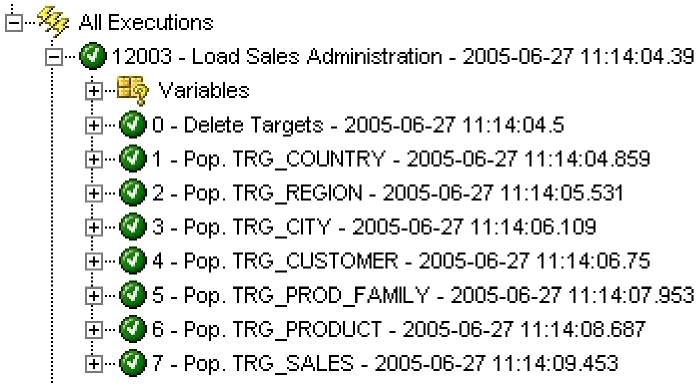
7. Click **Apply** to save the changes to the package

Run the Package

1. To run this package, click on the **Execute** button.
2. Click **OK** in the **Execution** and **Information** windows.

Follow the Execution of the Package in Operator

By reviewing the execution in Operator, you will find the same results as those obtained when you executed the individual interfaces.



```
All Executions
    12003 - Load Sales Administration - 2005-06-27 11:14:04.39
        Variables
        0 - Delete Targets - 2005-06-27 11:14:04.5
        1 - Pop. TRG_COUNTRY - 2005-06-27 11:14:04.859
        2 - Pop. TRG_REGION - 2005-06-27 11:14:05.531
        3 - Pop. TRG_CITY - 2005-06-27 11:14:06.109
        4 - Pop. TRG_CUSTOMER - 2005-06-27 11:14:06.75
        5 - Pop. TRG_PROD_FAMILY - 2005-06-27 11:14:07.953
        6 - Pop. TRG_PRODUCT - 2005-06-27 11:14:08.687
        7 - Pop. TRG_SALES - 2005-06-27 11:14:09.453
```

Deployment

Purpose

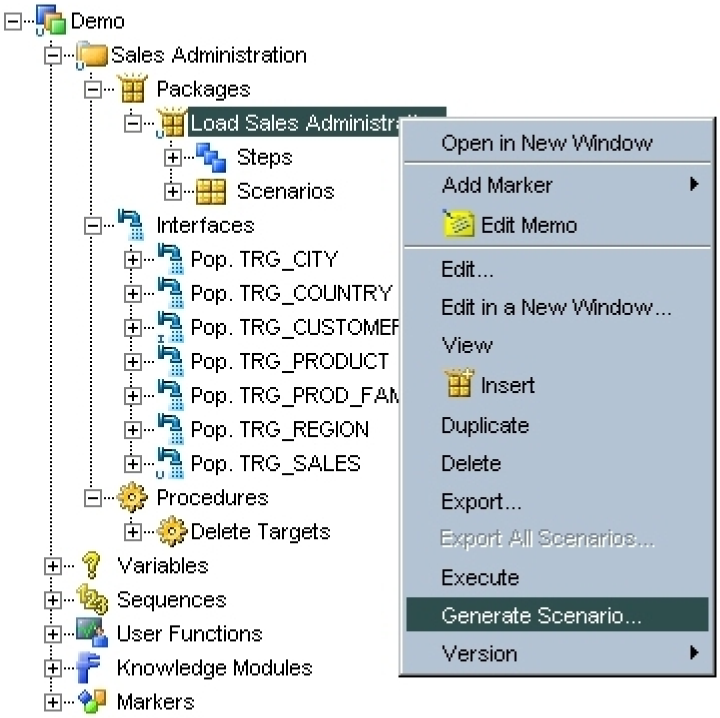
The goal of this section is to be able to run this package automatically in a production environment.

To do this we must build a scenario "LOAD_SALES_ADMINISTRATION" that we will use to execute the "Load Sales Administration" package.

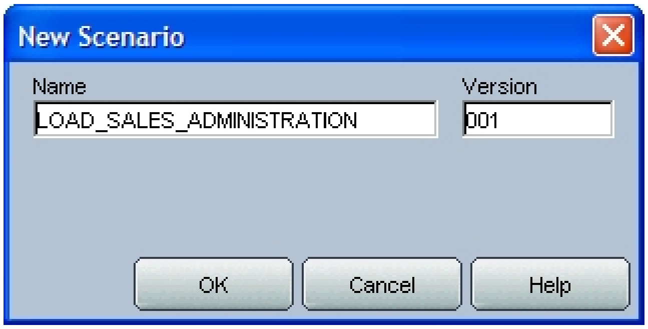
Steps

Scenario Creation

1. Select the **"**Load Sales Administration" **Package**
2. Right-click and select **Generate scenario.**



3. The following window appears:

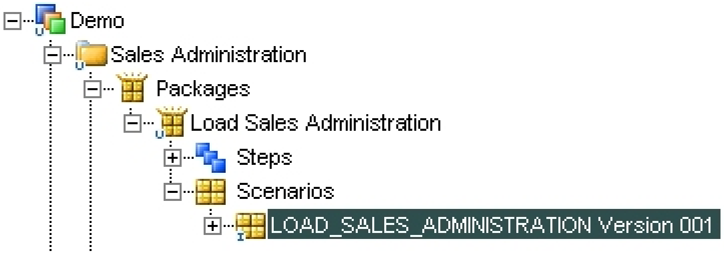


New Scenario

Name: LOAD_SALES_ADMINISTRATION

Version: 001

4. Click **OK.** Oracle Data Integrator processes and generates the scenario.
5. The result is shown below:



```
Demo
  Sales Administration
    Packages
      Load Sales Administration
        Steps
        Scenarios
          LOAD_SALES_ADMINISTRATION Version 001
```

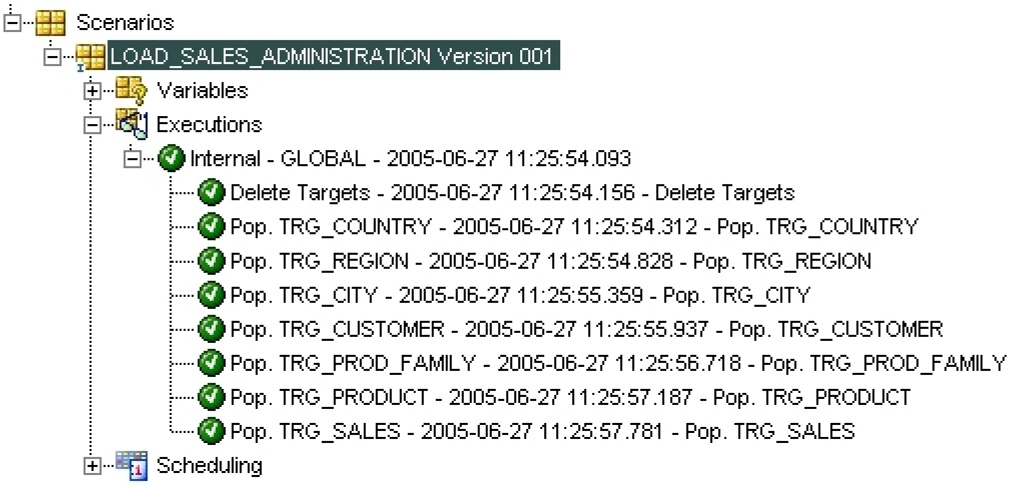
Run the Scenario from Oracle Data Integrator

1. Right-click on your scenario
2. Select **Execute**
3. Click **OK** in the **Execution** and **Information** windows.

Follow the Scenario's Execution

You can review the scenario execution in **Operator**, and find the same results as those obtained when the package was executed.

It is also possible to review the scenario execution report from the **Designer** module by expanding the scenario tree view as shown below:



```
Scenarios
  LOAD_SALES_ADMINISTRATION Version 001
    Variables
    Executions
      Internal - GLOBAL - 2005-06-27 11:25:54.093
        Delete Targets - 2005-06-27 11:25:54.156 - Delete Targets
        Pop. TRG_COUNTRY - 2005-06-27 11:25:54.312 - Pop. TRG_COUNTRY
        Pop. TRG_REGION - 2005-06-27 11:25:54.828 - Pop. TRG_REGION
        Pop. TRG_CITY - 2005-06-27 11:25:55.359 - Pop. TRG_CITY
        Pop. TRG_CUSTOMER - 2005-06-27 11:25:55.937 - Pop. TRG_CUSTOMER
        Pop. TRG_PROD_FAMILY - 2005-06-27 11:25:56.718 - Pop. TRG_PROD_FAMILY
        Pop. TRG_PRODUCT - 2005-06-27 11:25:57.187 - Pop. TRG_PRODUCT
        Pop. TRG_SALES - 2005-06-27 11:25:57.781 - Pop. TRG_SALES
    Scheduling
```

Run the Scenario from an OS Command

1. Open an MS-DOS command prompt window or a console in UNIX.
2. In the Oracle Data Integrator installation folder, open the "bin" directory.
3. Enter the following command:
   `startscen LOAD_SALES_ADMINISTRATION 001 GLOBAL "-v=2"`

Note: The parameters for the startscen command are:
- Scenario name
- Scenario version
- Scenario context
- Logging level
These parameters are separated by spaces.

4. When execution is finished, the command prompt should look like this:

```
C:\oracledi\bin>startscen LOAD_SALES_ADMINISTRATION 001 GLOBAL "-
v=2"
Oracle Data Integrator: Starting scenario
LOAD_SALES_ADMINISTRATION 001 in context GLOBAL ...
06/27/2005 11:33:05 AM(main): Creating session for
scenario:LOAD_SALES_ADMINISTRATION - 001
06/27/2005 11:33:06 AM(main): Session : 14003 is running
06/27/2005 11:33:16 AM(main): Session : 14003 finished with
return code : 0
DwgJv.main: Exit. Return code:0
```

Note: In this example, we use a command prompt to call a scenario. This can also be done from virtually any outside application. Conversely, it is also possible for Oracle Data Integrator to call outside applications that you may already own. Interaction with another application can thus be integrated as part of your processes.

Conclusion

You have now completed an ETL project, easily and quickly!

Throughout these examples, you have:

- Defined and implemented data integrity rules in the "Orders Application" application
- Created two interfaces to load the data from the "Orders Application" and "Parameters (File)" applications into the "Sales Administration" data warehouse.
- Sequenced your developments and automated the execution of your processes.

These examples correspond to just one type of problem for which Oracle Data Integrator provides an answer.

What else can you do with Oracle Data Integrator?

You have seen Oracle Data Integrator used 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. Other examples can include:

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

Furthermore, in this example 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, etc. – 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.

Going further with Oracle Data Integrator

Use the demonstration environment to familiarize yourself with Oracle Data Integrator.

The following documents will assist you in learning Oracle Data Integrator in more detail:

Document	Description
Installation Guide	Describes the system requirements and explains how to install Oracle Data Integrator.
Reference manual	Describes the details of Oracle Data Integrator graphical user interface.
User's Guide	Describes how to work with Oracle Data Integrator.

Contacting Oracle

If you require assistance or more information, please contact Oracle.

www.oracle.com