

ETL Proof of Concept Written Response

Course: Evaluating ETL Tools and Technologies, afternoon session
ETL Vendors in Action

Table of Contents

Proof of Concept Overview	2
Scenario Overview	2
Demo Scenarios / Topics	4
1. View of ETL / Data Integration and Product Architecture Overview	4
2. Extract Scenario 1: Customer Dimension Incremental Extract	6
3. Extract Scenario 2: Shipments Fact Table Extract	10
4. Extraction Scenario 3: Open Case	15
5. Extraction Scenario 4: Time Dimension.....	22
6. Maintenance Features	15
7. Operations and Deployment	19
8. Pricing	22
9. Performance Features.....	25

Proof of Concept Overview

The scenarios for the proof of concept are all based on a wholesale business that supplies specialty products to retailers. The scenarios are based on the items that one might consider important when evaluating an ETL solution for a single data warehouse.

The examples are all built around a wholesale shipments schema, with a set of source tables loaded with data and a set of target tables to be populated by the tools. The extract rules for the schema are simple, but should be enough to demonstrate basic and some advanced capabilities in the products.

The afternoon will be a mix of discussion and demo, with the emphasis on showing how the products are used to accomplish specific tasks. While the focus is on extraction, some of the scenarios or presentation topics involve showing other features like metadata management, data profiling or monitoring job execution.

Because there's no way to show the entire set of ETL for three vendors in the time allotted, we'll be using different elements to show different features. For the scenarios listed we expect to see the features used to accomplish the task live. It isn't expected that we can see the entire extract constructed for each scenario in the time given. However, a complete set of extracts is required in order to show how dependencies, scheduling and monitoring work.

Demo time is limited so there are topics/scenarios labeled "time permitted" which we may not be able to show. They are included in case we have extra time at the end of the class.

Scenario Overview

In a proof of concept you provide to vendors all the source and target table definitions, extract rules and source data. Since this is meant to reflect the real ETL you'll be doing, it's a good idea to select both simple extracts and complex extracts or extracts that have problem data. When you provide this information, it should be formally documented so the vendor understands in detail what they are supposed to show.

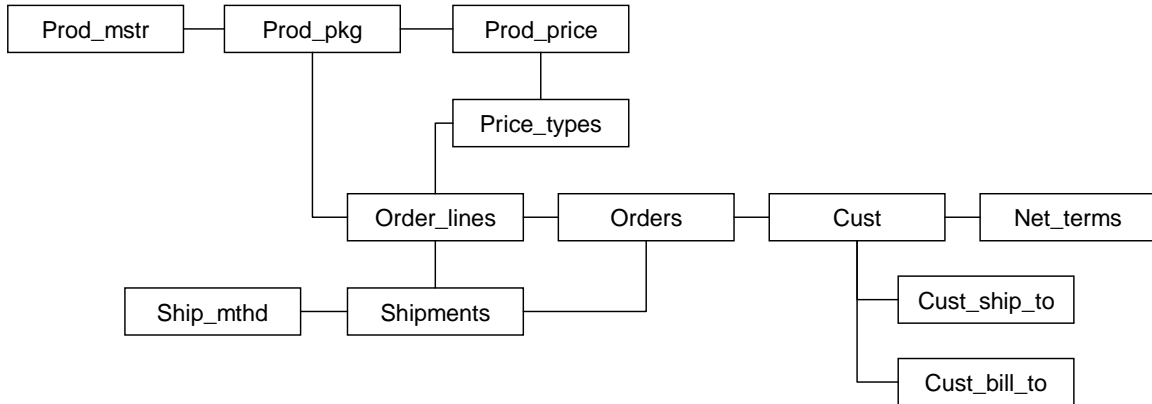
Part of the reason for selecting source data with quality problems is that this will show how a developer is expected to work within the tool. If all the extracts are based on ideal tables and data, as with standard vendor demos, then you won't see what a developer really has to face when dealing with data exceptions.

As a rule, you should have imperfect data, tables with relationship problems like different types on join or lookup columns, and you should always require the use of relational database in the proof of concept.

Using a database is important because it will show you how the tool interacts with a database. Many vendor demos you see are based on text files as input and output, which

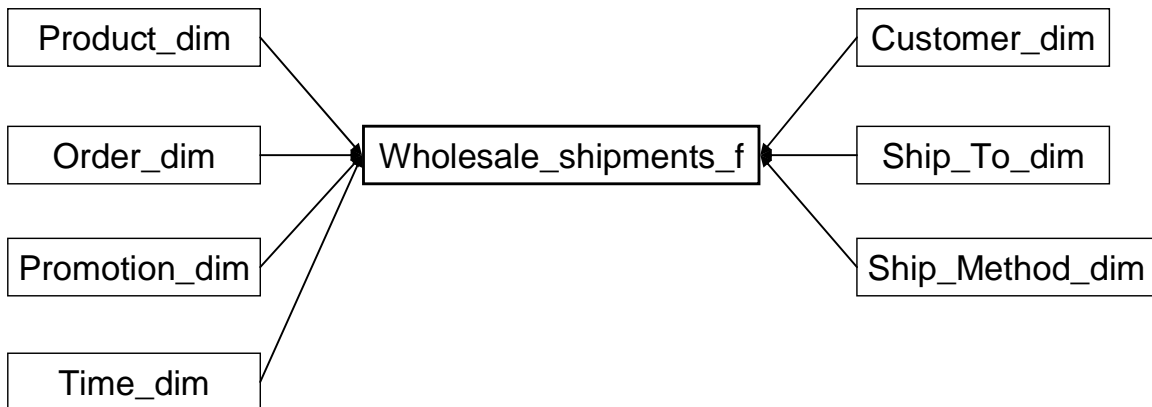
can avoid some of the difficulties when dealing with SQL since all the work is done directly in the ETL engine.

For our scenarios, we will be using the following source and target schemas. The source schema consists of 12 tables with some of the common design problems found in source databases.



Among the problems are a mix of second and third normal form, multi-part keys, invalid foreign key constraints, and multiple join paths to the same data elements. In addition to the above tables there are two change-data-capture tables for the shipment and customer data. These are used for incremental extract examples.

The target schema has seven dimension tables and one fact table. The fact table contains all shipments of order lines by day to customer addresses in the ship_to dimension.



There are several things in this target schema that complicate the extracts. The time dimension is based on a fiscal calendar using 4 and 5 week periods rather than a simple date-based dimension. There is no source for this data, so it must be constructed by the ETL job. A non-data-driven extract is a challenge for some ETL products. The ship_method dimension has unique rows based on a multi-part key which can cause trouble for some ETL tools' lookup functions. The specific details about extract rules and data are available at the end of this document.

Demo Scenarios / Topics

The following section describes each of the presentation topics or scenarios that will be reviewed during the course. For each scenario, there is a list of directions or questions to be answered and a short view of the schema and data (if applicable).

Included with the descriptions are sample criteria you might use during an evaluation so that you can score the vendors during the class. After this section there are responses from each of the vendors to all of the scenario questions so you have something to refer back to.

1. Product Architecture Overview and ETL Perspective

This is a short (4 minutes or less) presentation about the company, its products and its view of the ETL and data integration market. You should expect to hear answers to the following types of questions:

- The product name or components used for this demo, whether all components are included in the base product and / or how it's packaged.
 - a. The product used for the demonstration is Oracle Data Integrator. Only components of the basic offering are used for the demonstration (i.e. no optional elements are used).
 - b. The graphical components that will be shown during the demonstrations are ODI Topology Manager (definition of the connectivity to the different systems), ODI Designer (Definition of the mappings, transformations, data movement strategies, and metadata management), ODI Operator (follow-up of processes execution, execution statistics and review of generated code). If time permits, the Metadata Navigator will also be presented (web front-end to navigate the ODI repository – including data lineage, impact analysis, and operator access).
- What is the core ETL product offering?
 - ODI EE is Oracle's data integration solution. It is comprised of ODI (Oracle Data Integrator, previously known as Sunopsis) and OWB EE (Oracle Warehouse Builder Enterprise Edition). Both components are included with the same license.
- What optional components are provided?
 - Optional components are:
 - i. Oracle Data Quality
 - ii. Oracle Data Profiling
 - iii. Application connectors for SAP, PeopleSoft, Siebel, eBusiness Suite and JDEdwards.
 - It must also be noted that many Oracle applications are using ODI for their data integration layer. This represents over 50 development teams at Oracle, comprised of hundreds of developers)

- How are components / editions bundled?
 - i. ODI EE, comprised of ODI and OWB. See above for optional components for this offering
 - ii. ODI Suite, comprised of ODI EE, BPEL Process Manager, BPEL Human Workflow Manager, Oracle Enterprise Service Bus, Oracle DRM (Data Relationship Manager). Certain restrictions apply to BPEL, ESB and DRM.
- What requirements are there for server installs, client installs, etc.?
 - i. ODI: 200 Mb storage space in a database for the ODI repository
 - ii. ODI GUI: 200Mb of disk space on the end-user machine, 1 Gb of RAM (Usually uses less than 100Mb) and Java 1.4.2 or above (installed by the ODI installer)
 - iii. ODI Agent: 200Mb of disk drive on a file server or database server. 1Gb of RAM (usually uses less than 1Mb)
 - iv. ODP and ODQ typically require a dedicated 2 or 4 CPU server.
- What external dependencies are there? For example, a license for a database to be used as a metadata repository.
 - i. A database is required to store the ODI metadata repository. Most customers re-use an existing database as ODI requirements are minimal. This database can be Oracle, Microsoft SQL Server, DB2 UDB, Informix, Sybase.
 - ii. Java 1.4.2 or above
 - iii. If the Metadata Navigator is used, an application server is required (Weblogic, Tomcat, etc)
 - iv. If ODI is used to generate web services, Axis2 is required on the application server.
- What do you consider your sweet spot in applications or in industries?
 - i. ODI will serve enterprises in all verticals and industries. Existing customers expand the usage of ODI in all departments and tend to normalize on ODI at an enterprise level.
 - ii. Oracle is standardizing on ODI for all data integration requirements within the Oracle stack.
- What are your product's strongest points?
 - i. ELT versus ETL to better leverage the database engines
 - ii. Developer productivity and maintenance of the developed code, thanks to a declarative design and a template based approach (knowledge modules).
 - iii. Fully integrated solution for batch, real-time, event-driven integration
 - iv. Changed Data Capture
 - v. Errors capture and re-cycling
 - vi. Best of breed data profiling and data quality
 - vii. Heterogenous support (all database vendors are supported)
 - viii. Fully integrated in the Oracle stack
 - ix. Flexibility, modularity, and light weight architecture
- Why would a customer choose your product over others?

- i. Best of breed solution
- ii. Leverage of existing environment
- iii. Better productivity, easier maintenance
- iv. More and more integration with other Oracle products that will require ODI knowledge

2. Extract Scenario 1: Customer Dimension Incremental Extract

This scenario demonstrates the extract logic and facilities for a slowly changing dimension where history is preserved in the rows using start and end dates.

Scenario

Change rows are stored in a change data capture table. This is a table that mirrors the source table, but has additional columns that indicate whether a row is an insert, update or delete, and a timestamp for the activity. This is what you might see as the result of change replication using database replication facilities. The expectation is that the change table provides the activity, while the source table shows the current state of the data.

To properly test the extract, there are four rows for three different conditions. The test cases are as follows:

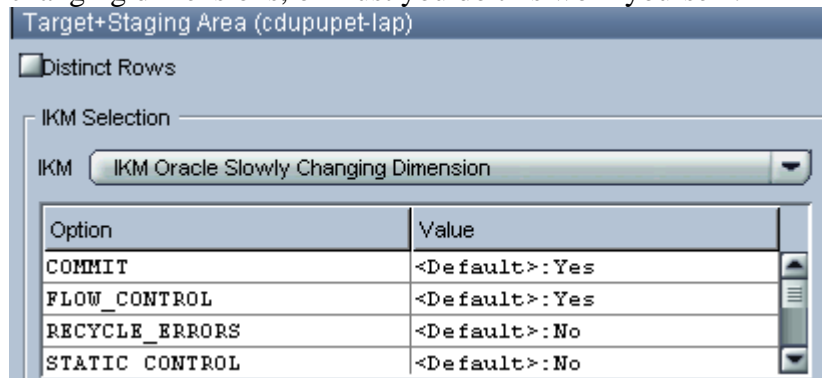
- Insert of a new customer
- Update of an existing customer, changing address line 2.
- Same day insert and later update of a row (within the same batch).

The goal of this scenario is to show how one would address the issues associated with slowly-changing dimension tables where a changed-data capture table is used as input. The intent is to also show some of the features available for looking at data values before and after update from within the developer's interface. The demonstration should answer the types of questions outlined below.

- How do you perform a simple lookup from another table?

Ind	Name	Mapping
	*xcustomer key	<%=odiRef.get
	*customer nbr	CUS.cust nbr
	*customer ...	upper(trim(CU
	*customer ...	upper(trim(CU
	*customer ...	force not nul
	*customer ...	upper(trim(CU
	*customer ...	force not nul
	*customer ...	force not nul
	*customer ...	force not nul
	*city	force not nul
	*state	force not nul
	*postal code	case when cus
	*country	force not nul
	*phone	case when CUS
	*fax	case when CUS
	*default ...	upper(CUS.cus

- Lookups are defined graphically. Users drag and drop columns and define the type of join that is required. Outer joins are defined graphically.
- Are there any features that automate or simplify the maintenance of slowly changing dimensions, or must you do this work yourself?



ODI Knowledge Module Selection

ODI separates the notion of transformations from the notion of the corresponding integration strategy. The different integration strategies are compiled into “Knowledge Modules” that are in essence templates of code that will define best practices for any given integration strategy: creation / drop of staging tables, creation / drop of indexes if and where needed, dynamic generation of scripts for loading utilities, etc.

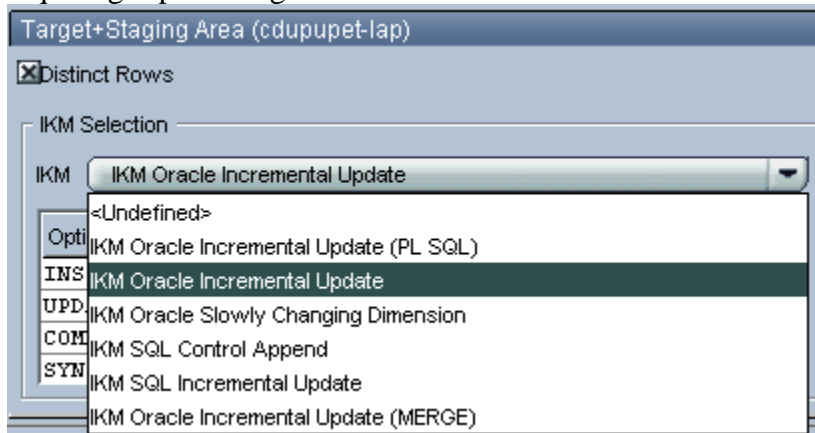
ODI does provide out of the box templates (the Knowledge Modules) that automate the implementation of Slowly Changing Dimension, leveraging native code for most database technologies.

Knowledge Module: IKM DB2 400 Slowly Changing Dimension

Ord...	Command	Context	Logical ...	Transaction	Commit	Ignore Errors	Log
12	Create target table					<input checked="" type="checkbox"/>	
52	Delete target table					<input type="checkbox"/>	
72	Drop flow table (I\$)					<input checked="" type="checkbox"/>	
92	Create flow table (I\$)					<input type="checkbox"/>	
122	Lock Journalized Table					<input type="checkbox"/>	
132	Insert flow into I\$ table					<input type="checkbox"/>	
152	Recycle previous errors					<input checked="" type="checkbox"/>	
182	Flow control					<input type="checkbox"/>	
192	Create unique index on flow table					<input type="checkbox"/>	
212	Flag rows to update					<input type="checkbox"/>	
232	Flag useless rows					<input type="checkbox"/>	
252	Update existing rows					<input type="checkbox"/>	
262	Historize old rows					<input type="checkbox"/>	
272	Insert changing and new dimensi...					<input type="checkbox"/>	
292	Cleanup journalized table					<input type="checkbox"/>	
302	Post-integration control					<input type="checkbox"/>	
312	Drop flow table (I\$)					<input checked="" type="checkbox"/>	

Sample ODI Knowledge Module Code.

- How do you deal with both inserts and updates against a target table without requiring separate logic?



ODI provides out of the box templates that automate the processing of inserts and updates as part of the same extract. For each mapping, the developers can specify if the transformation will apply for inserts, updates, or both, as show in the screenshot below.

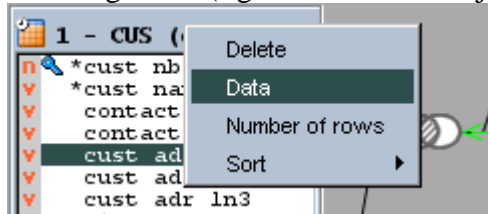


- Do you have any problems with locking or staging when retrieving and updating rows in the target table as part of the extract?

The ODI knowledge modules are designed to automatically generate the appropriate staging tables where and when needed. The tables are entirely maintained by ODI, and do not require any manual intervention. From that perspective, a target table can be used as a source table in any part of the loads

- How do you preview source data from within the design interface?

Source data, target data, results from joins and filters can be viewed directly from the design area (right-click on the object and select “Data”).



It is also possible to view row counts, and to view changed data vs. data when changed data capture is enabled.

- Can you look at both the input and output data from within the developer interface to see how updates were applied?
Source and target systems can be investigated. It is also possible to use the GUI to investigate staging tables by modifying the SQL query used to render the data.
- In the case where you do not preserve history, what steps do you have to take for a destructive load?
All knowledge modules come with a set of options, two of which can be set for this purpose: “Delete All” and “Truncate”. The knowledge Modules could also be modified to enforce a “drop table and create table” if it makes more sense in a given environment.

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

- Is there any support for managing hierarchies in dimensions?
Management in hierarchies can be done with successive loads for the different elements of the hierarchy, or by leveraging hierarchy functions of the database (such as “connect by” in Oracle).

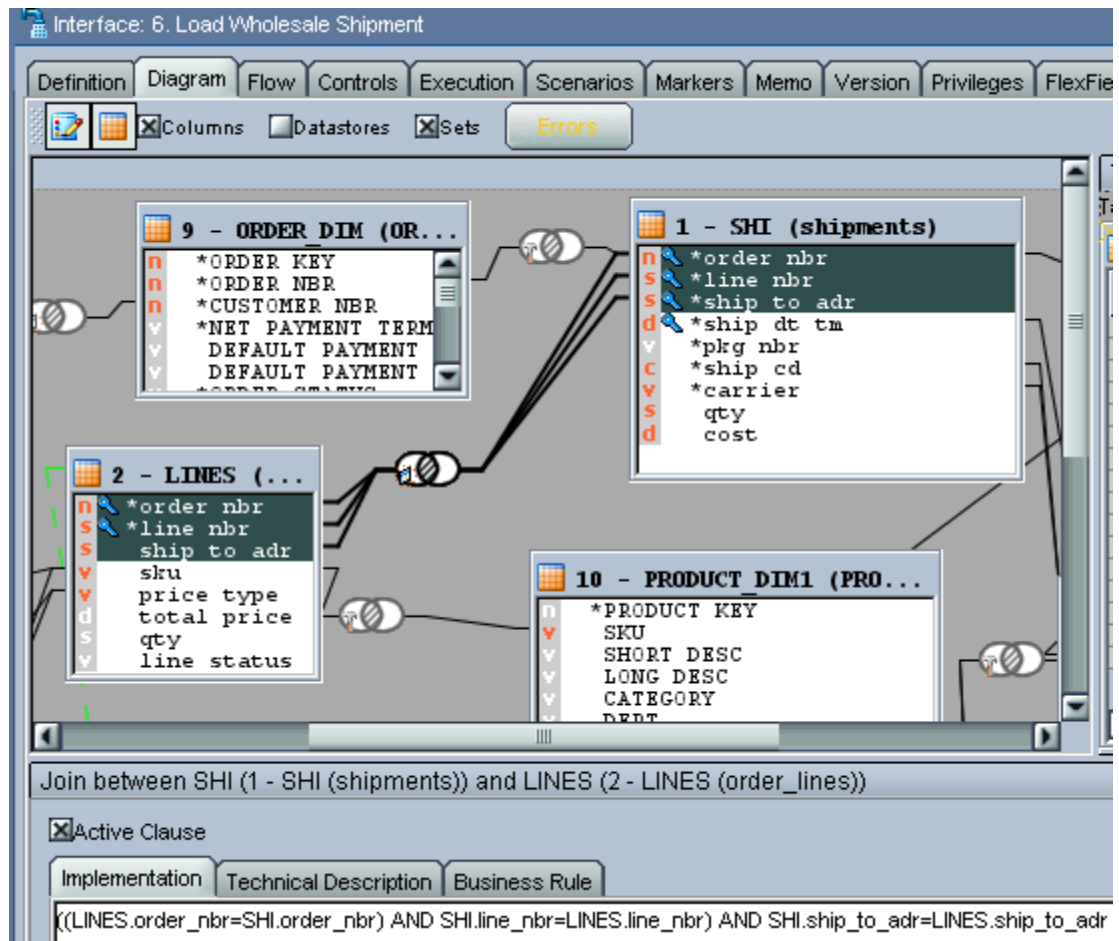
3. Extract Scenario 2: Shipments Fact Table Extract

The purpose of this scenario is to show a more complex extract involving conditional logic, calculations, many source tables, multiple lookups, and available facilities for dealing with surrogate key matching and validation for dimensional fact tables.

Scenario

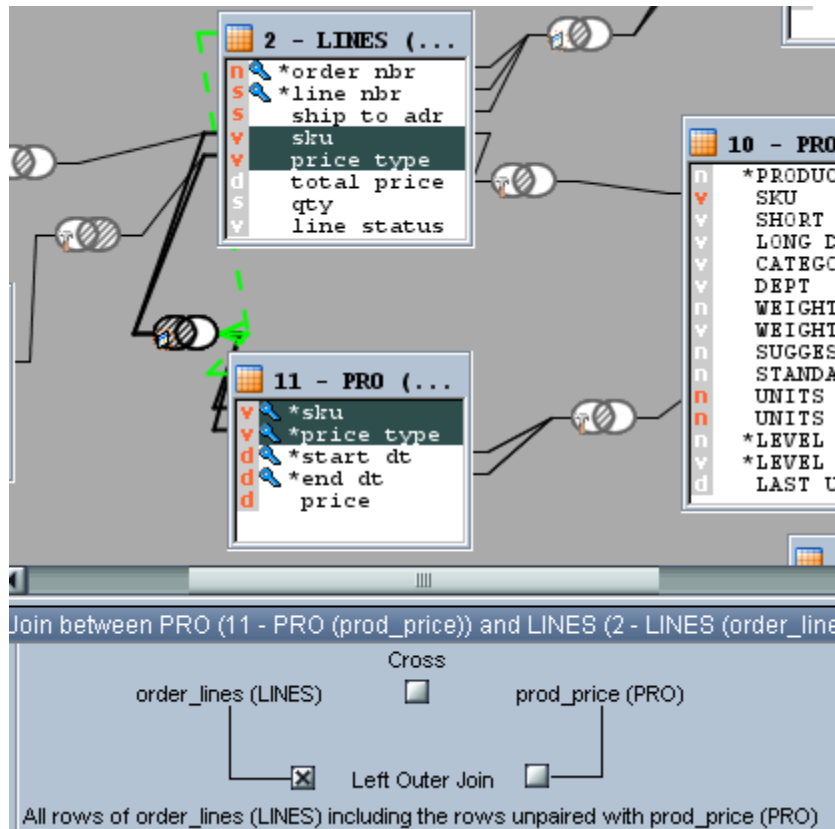
The goal of this scenario is to show how one would address the more complex issues associated with building a fact table. Added complications in the data include common problems faced by developers. This includes dealing with missing data (forcing outer joins or conditional lookups), varchar and char data type conversions and comparisons, and missing values. The demonstration should provide answers to the following types of questions.

- How do you do a lookup based on a multi-column input key? (ship_method_key uses both ship_cd and carrier)
Multi-column lookups are as easy as single column lookup: simply drag and drop the columns that make up the relationship from one table to the next. The GUI will graphically connect the columns to represent the lookup, and the developer can modify the relationship between the columns if needed (not all joins are equijoins!)



- How would you accomplish the equivalent of an outer join between two tables? (the shipments table has some bad product values that won't join to the product tables, there is a non-existent customer number, and the ship_to address is missing for a shipment)

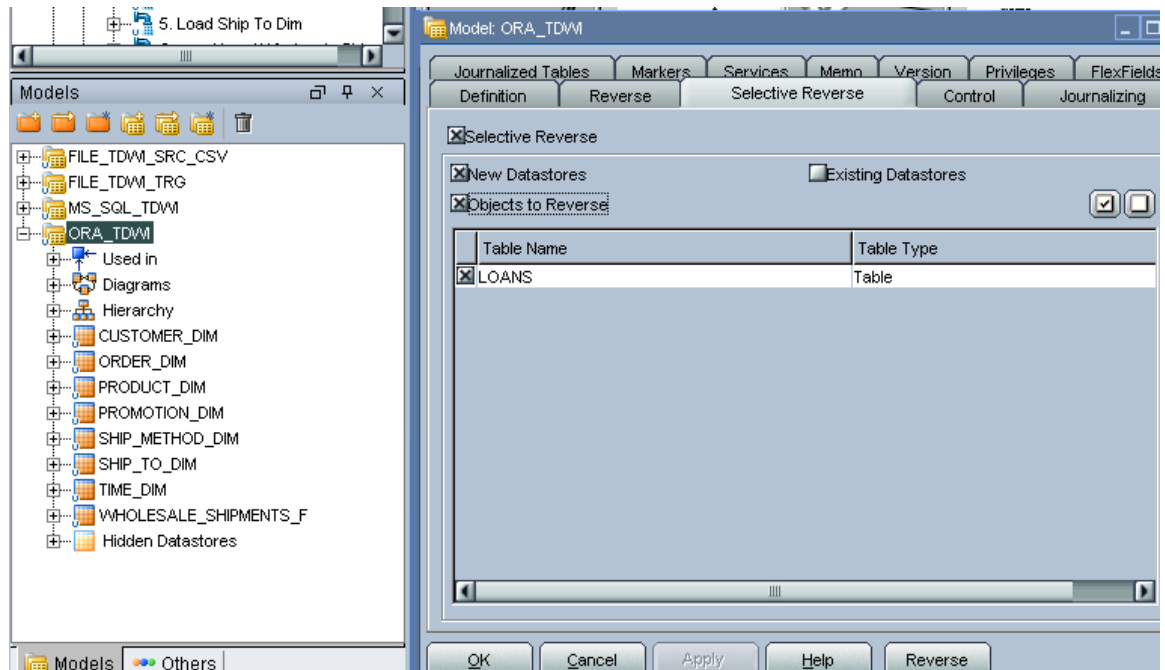
ODI lets you select the type of join you want: inner join, left or right outer join, full join or Cartesian product (cross join). The developer select the type of join, and ODI will generate the appropriate SQL code for the underlying database. A plain English description of the nature of the join will help non SQL developers understand what data will be brought back by the lookup. The screenshot below shows a left outer join, with the plain English description at the bottom.



- Assume the developer forgot to include a table in the metadata so it isn't present but is needed at design time. How does the developer import the metadata so the table can be used in this extract?

This process is called reverse engineering of Metadata in ODI. Tables are organized in Models in ODI. The developer will simply edit the model to see which tables are available in the database, and simply select the missing table to add it to the list of available tables in ODI.

The screenshot below shows the tables that are already available in ODI on the left hand side, and an additional table that could be added to the list on the right hand side (that table exists in the database, but its definitions has not been imported yet in ODI). To add the table to the list, all the developer has to do is hit the "Reverse" button on the bottom right. If multiple tables were available in the database, the developer could import them all at once, or import a selection of the tables, by either creating filters on the tables names, or by manually selecting the tables to be reverse engineered.

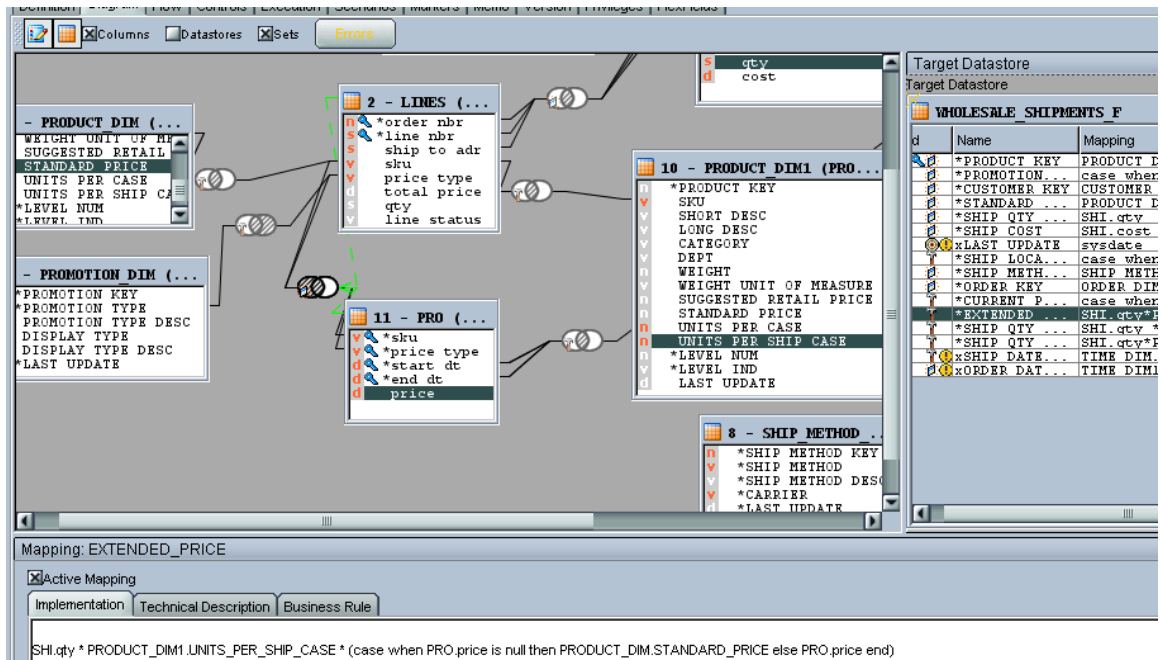


- How do you do calculations involving values from multiple tables, or aggregation? (unit calculations are an example, and missing rows in one table can cause difficulties with these calculations)

Simply use the columns from the source tables as part of the formulas. ODI will stage the source data as needed, rename the columns as needed for the staging tables, and make sure that the formula designed by the developer is applied properly.

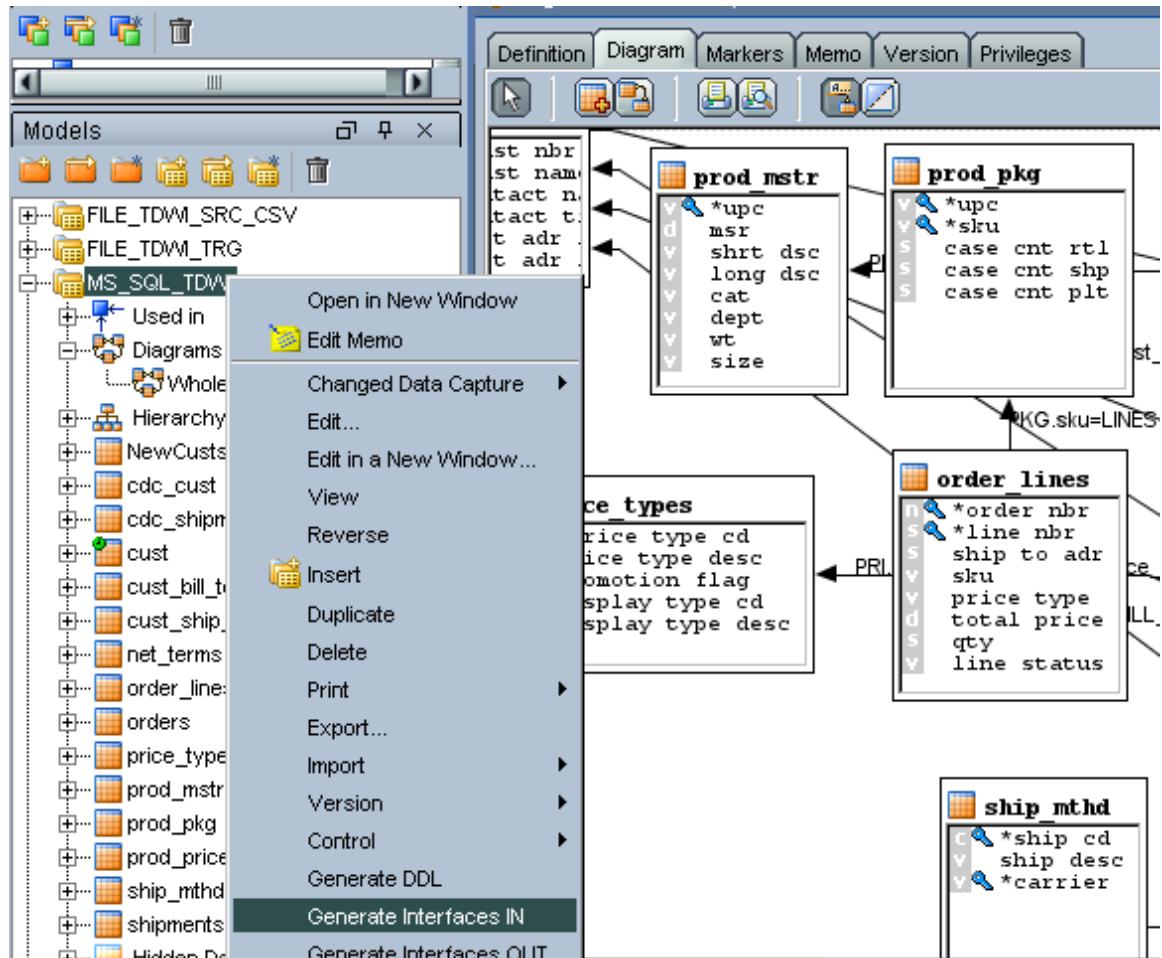
In the example below, when the developer clicked on the target column (right hand side) ODI highlights the source columns that are part of the formula in the different source tables, and displays the formula in the property panel at the bottom. Here the formula reads:

*SHI.qty * PRODUCT_DIM1.UNITS_PER_SHIP_CASE * (case when PRO.price is null then PRODUCT_DIM.STANDARD_PRICE else PRO.price end)*



- What features does the tool have to automate common tasks like keying fact tables?

ODI provides automatic mapping for columns that have the same name. Beyond that, the fact table can be designed in ODI from the definition of dimensions and source tables, and ODI can automatically generate the code (including the default mappings) for the table.



4. Extraction Scenario 3: Open Case

This is an open case. The vendors have been asked to demonstrate something that is unique or compelling about their products. The specific features they show aren't always known in advance.

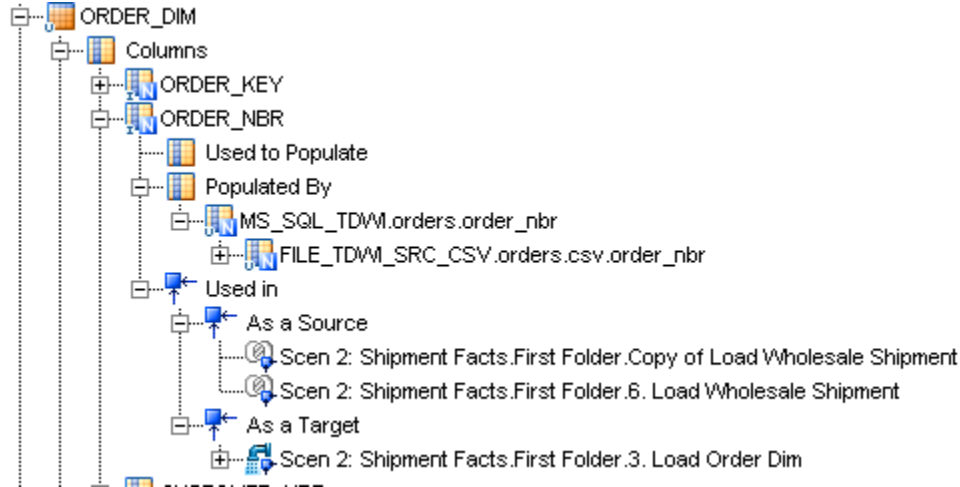
5. Maintenance Features

Post-deployment maintenance is an important aspect of the ETL process. There is no specific scenario. Instead, the vendors have been asked to describe and demonstrate features available to developers that address the types of questions outlined below.

- Assume there is a change to the data type of the `order_nbr` column in the Orders dimension. What is the process of tracing the impact of this change and how would a developer change the affected extracts?

The first step before allowing for the change will be to see where the `order_nbr` column is used in the different transformations. The column could be used as a source to target tables, could retrieve data from other tables and be a target, or

could be used as a lookup value. All references to the column are available in the GUI under the column definition, as illustrated below.



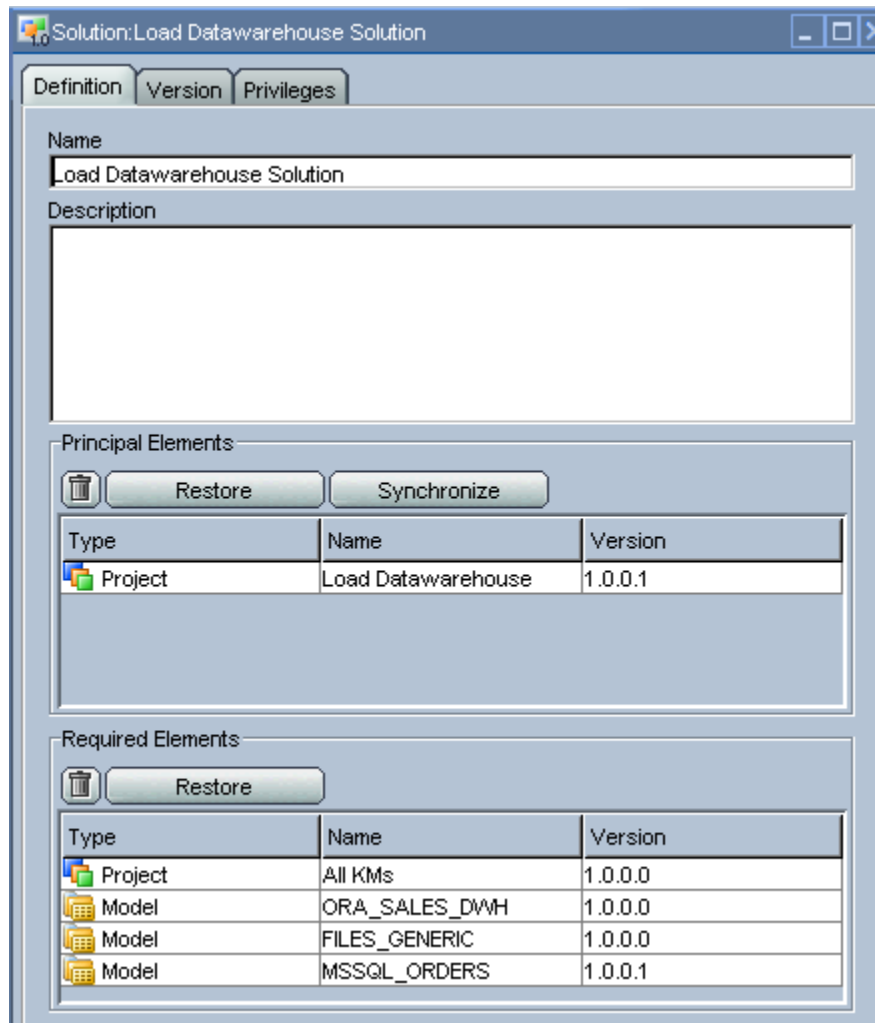
Here, the impact analysis tells us that the column is used as a target only and receives data from MS SQL Server. Data in SQL Server originally comes from a flat file.

We also see that the column is used in 2 different lookups. We can double-click on these lookup to investigate the other tables and columns involved in this lookup. This would show that this column is joined with the order_nbr from the shipment table in SQL server.

Once the impact has been assessed, the developer can update the data type either manually or by re-importing the table definition from the database (this would overwrite the current data types). Then the developer can double-click on any of the references above that require code modification due to the change of data type.

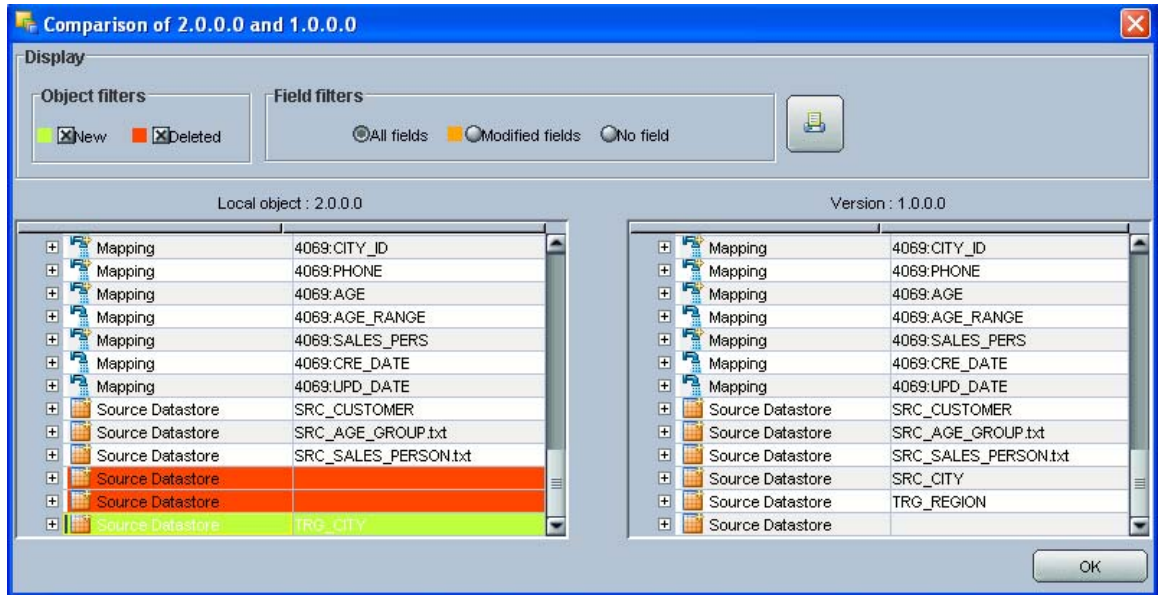
- What source control or version control features are available?

Versioning is an integral part of ODI: all objects can be versioned. All versions of the objects are saved in the ODI repository and can be restored later as needed. The major issue in Data Integration environments is that as projects evolve, not only do the transformations evolve, but the metadata evolves as well. When restoring a transformation in a previous release, it is important to be able to retrieve the then current version of the metadata. ODI provides a concept called “Solutions” that takes care of these dependencies.



All objects, including solutions, can be exported as XML files and later imported in the same (or another) ODI repository.

Different versions of the same object can be compared thanks to the version browser as illustrated below:



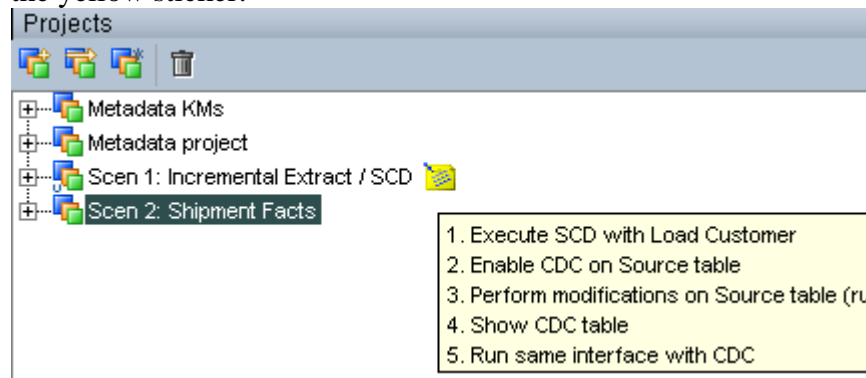
- What facilities are available for documenting or annotating extracts?

Everything done by the developers can be documented. All objects have a “Description” field that will let developers describe what they did. In addition, they can attach “memos” to the different objects, to either document their work in progress or communicate with one another while developing.

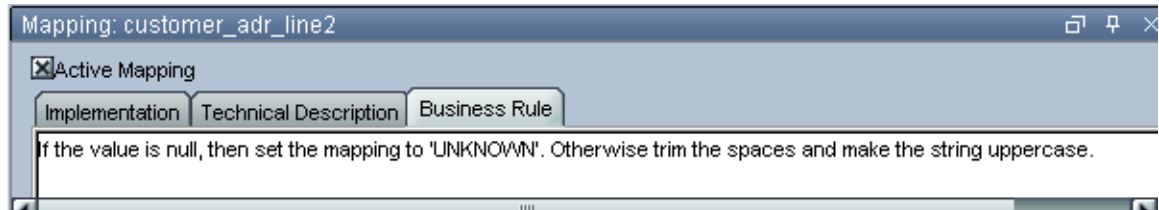
As mappings are developed, developers can also add a plain English “business rule” that describes what their formula is doing.

All these documentation elements are retrieved by ODI and compiled with the actual metadata and transformations when the documentation is generated from the GUI. Documentation can be generated on the metadata, on the interfaces (definition of the mappings and data flows), on the packages (definition of the workflows) and on the projects (combination of interfaces, packages, variables, user functions, etc.)

The screen shot below shows the content of a memo when the mouse hovers over the yellow sticker:



This other screenshot shows the business rule associated to a mapping in an interface:



- How does a developer compare and find differences between an extract that has been deployed in production and one in the repository / development environment?

The developer can use the version browser described above

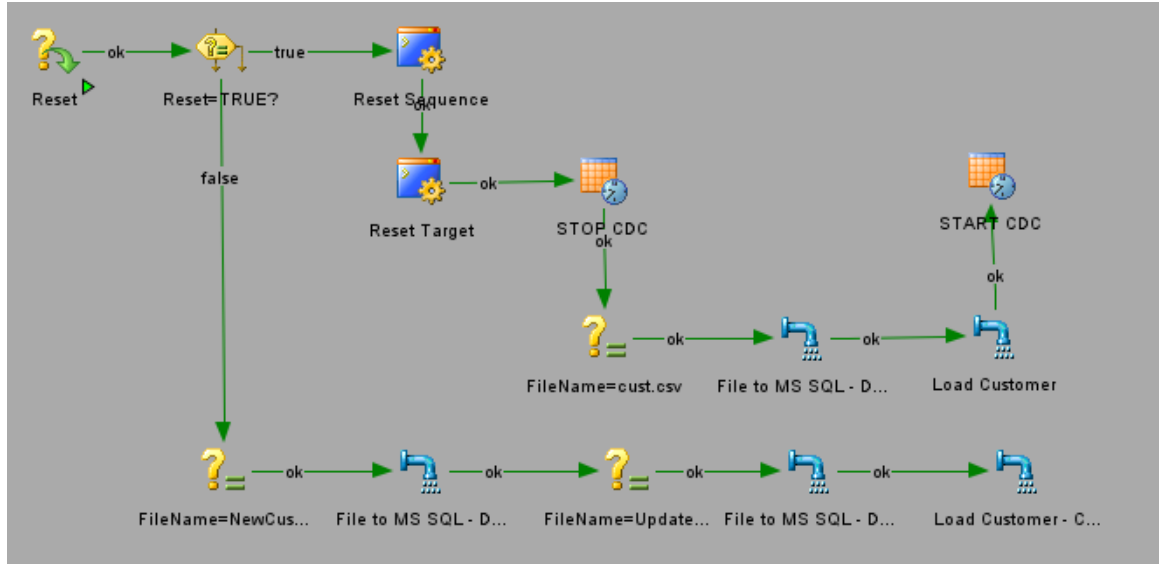
- How does a developer address production deployment and rollback of jobs?

Version management will allow the developers to deploy validated versions in production (restored into the production repository) as well as the rollback to previous versions (restoring a previous version in the repository).

6. Operations and Deployment

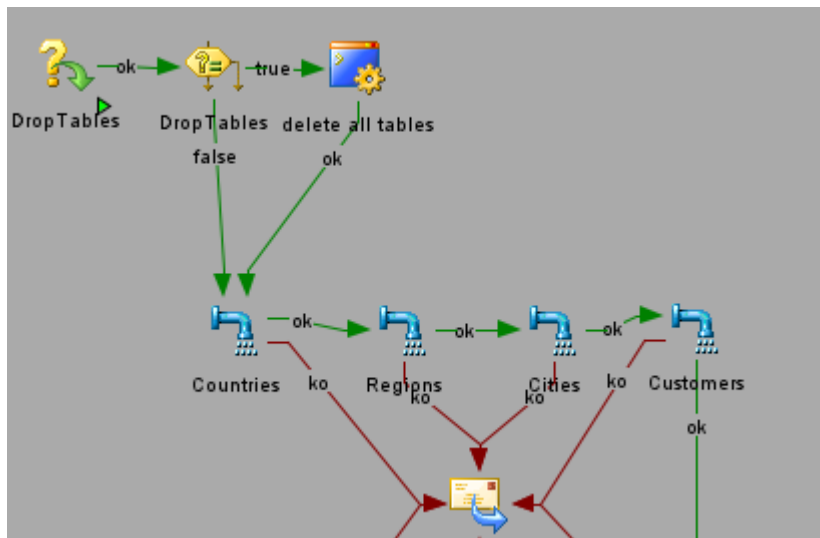
To show features available for scheduling and monitoring, we are using the complete set of extracts for the dimension and fact tables. The vendors have been asked to show how dependencies between the various extracts are configured, how a schedule for execution is created, and how the extract jobs are monitored. The goal is to show product features related to the types of questions outlined below.

- What is the executable unit of work in the product?
There are multiple levels of execution units. Developers will typically execute “interfaces” one by one to validate the logic to load data into individual tables. Once these loads have been validated, the developers will combine the interfaces into “packages” where they can define the workflow, define the behavior in case of error, make the process event-driven, notify administrators, define loops, set parameters, and invoke other processes. ODI Tools can be used in the workflows to enhance the integration process: web services calls, FTP, emails (receive-send).



- Demonstrate how execution dependencies between jobs are set up (e.g. fact table doesn't load until dimension tables have loaded).

In the illustration of the previous point, we can see that all actions are defined in a serial fashion (i.e. we only execute a step after the previous step has been executed). An improvement to this scenario would be to process errors in that same workflow (this would show up as red arrows between the icons). The illustration below shows decision points and error processing (or in this specific case, error notification)



- How do you make a change to the dependencies?
Simply re-order the icons in the workflow, or add new ones as needed.
- How do schedule-based initiation and event-based initiation of jobs work?

ODI provides its own scheduler. ODI jobs can also be invoked from other schedulers through a command line interface or through a Web Service call. Event detection objects can be used in the definition of ODI processes to identify new files in a directory, new emails, new records in a table, new tables in a schema, completion of another ODI process, messages on a JMS queue.

- How can execution be monitored by a developer within the development environment, and by an administrator outside the development environment?

The same interface, the Operator interface, can be used by developers, administrators and operators to monitor the execution of the processes. This interface will detail all the steps the processes are going through, will display the actual code that is executed, and will display statistics for the tasks being executed.

The screenshot displays the 'All Executions' window in ODI. It shows a list of job executions with their status (success or failure) and timestamps. Below the list, the 'Variables' section is expanded to show the detailed steps of a failed job (ID 214690).

Job ID	Description	Status	Timestamp
252690	Copy of 6. Load Wholesale Shipment	Success	Jul 30, 2009 5:23:53 PM
251690	Copy of 6. Load Wholesale Shipment	Success	Jul 30, 2009 5:21:56 PM
250690	Copy of 6. Load Wholesale Shipment	Success	Jul 14, 2009 8:11:41 PM
249690	Copy of 6. Load Wholesale Shipment	Success	Jul 14, 2009 7:47:43 PM
248690	Copy of 6. Load Wholesale Shipment	Failure	Jul 14, 2009 7:46:18 PM
227690	6. Load Wholesale Shipment	Failure	Jul 14, 2009 4:25:04 AM
217690	6. Load Wholesale Shipment	Failure	Jul 14, 2009 3:03:28 AM
216690	6. Load Wholesale Shipment	Success	Jul 14, 2009 2:53:24 AM
215690	6. Load Wholesale Shipment	Success	Jul 14, 2009 2:35:16 AM
214690	6. Load Wholesale Shipment	Failure	Jul 14, 2009 2:34:31 AM

Step ID	Description	Status
1	1 - 6. Load Wholesale Shipment - Jul 14, 2009 2:34:31 AM	Failure
1	1 - Loading - SS_0 - Drop work table	Warning
2	2 - Loading - SS_0 - Create work table	Success
3	3 - Loading - SS_0 - Load data	Success
4	4 - Loading - SS_0 - Analyze work table	Success
6	6 - Loading - SS_1 - Drop work table	Warning
7	7 - Loading - SS_1 - Create work table	Success
8	8 - Loading - SS_1 - Load data	Success
9	9 - Loading - SS_1 - Analyze work table	Success
11	11 - Integration - 6. Load Wholesale Shipment - Drop flow table	Warning
12	12 - Integration - 6. Load Wholesale Shipment - Create flow table I\$	Success
13	13 - Integration - 6. Load Wholesale Shipment - Insert flow into I\$ table	Failure
14	14 - Control - WHOLESale_SHIPMENTS_F - create check table	Warning
15	15 - Control - WHOLESale_SHIPMENTS_F - delete previous check sum	Warning
16	16 - Control - WHOLESale_SHIPMENTS_F - create error table	Warning
17	17 - Control - WHOLESale_SHIPMENTS_F - delete previous errors	Warning
18	18 - Control - WHOLESale_SHIPMENTS_F - insert Not Null errors	Warning

The same information is available through a web interface called the Metadata Navigator, so that operators and administrators do not have to install the complete GUI on their machines.

- Explain the mechanisms available for monitoring execution and sending alerts if there are problems.

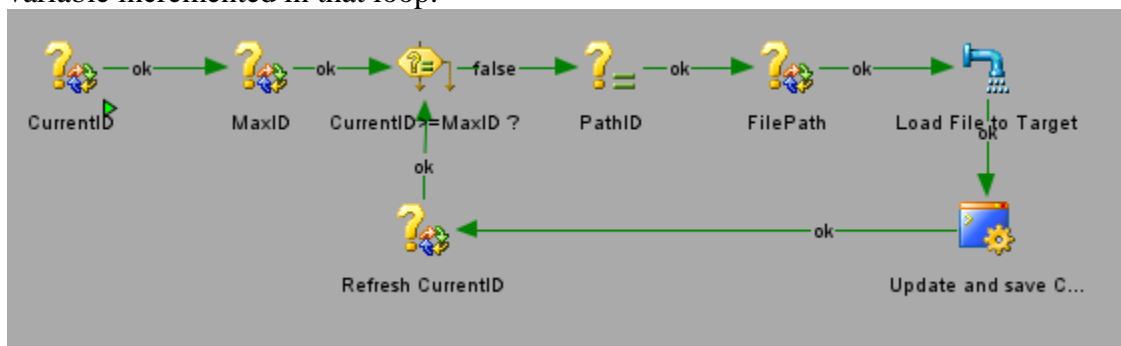
Monitoring of the executions is done through the operator interface described above. Sending alerts can be embedded in the workflows: when errors are detected, ODI can send an alert by email or through a message queue.

7. Extraction Scenario 4: Time Dimension

This scenario involves building a derived table where there is no data source. The job must construct a time dimension based on a fiscal (4-4-5) calendar with a number of derived date attributes. The vendors have been asked to explain and demonstrate how one would construct a derived table like this where there is no data source and the data must be created by the ETL program.

This sort of extract also shows how vendors deal with looping constructs and conditional logic, both of which are needed to work out problems like leap years and fiscal years crossing into new calendar years.

- What facilities are there to generate data?
The generation of data is done by using system tables in the different databases (dual in Oracle, or random generations in other databases)
- Ease of addressing conditional logic and loops.
Conditional logic and loops are usually defined as part of the packages, when the orchestration of the processes is defined. Variables are often used in that environment.
The example below illustrates a loop that stops with a test on the value of the variable incremented in that loop.



- Date management and attribute functionality.
ODI provides an API to manage the dates. All database date functions can also be used. In the case where ODI is used to generate a time dimension, leveraging database date functions makes the exercise a lot easier.
- Does the product recognize time as a specific type of dimension?
No – because to ODI a time dimension is just another table.

8. Pricing

Since pricing is so variable we asked for information about the licensing policies and how costs are calculated.

For ETL , there is the added complication of sizing. Depending on the tools, you may need a large, small, or no ETL server. You may also need to expand the warehouse or source servers to accommodate the workload. How you configure your environment can affect the pricing of the software.

Basic criteria:

Is there a per seat cost for developers?

No

Is there a per seat cost for administrators?

No

Is there a price per server by CPU? Per core?

Pricing is based on target CPU, 23K. This abides by Oracle core factor rules.

Is the price different for different server operating systems?

It can be – again, regular Oracle Core Factor metrics apply.

Are there per source or per target server / instance charges?

No (CPU count only, independently of the number of servers or databases on the servers)

Are there additional source / target connector charges?

SAP, Siebel, EBS, Peoplesoft and JDE ‘adapters’ are available at 2,300K per CPU. All other KMs and functionality is out of the box.

Is there a charge for development or test environments? If so, is it the same cost?

Yes. All environments do need to be licensed, same ‘target’ CPU metric at 23K per CPU applies.

How is maintenance charged? What is the range if it is a percent of some set of costs?

Maintenance is 22% of final price.

How many different editions or bundles are offered?

ODI-EE (ODI and OWB) is offered, along with the ODI Suite (Data Integration Suite)

Are there additional charges for team-based development, e.g. source control, project-level security, role-based access?

No.

Please provide an estimated list price and support cost for these two scenarios:

Scenario 1: Department / project level ETL

A single data warehouse, with one ETL project

3 ETL developers, 1 administrator / operations role

2 different (non-legacy) database source types, and some file-based extracts

1 target database server, 4 CPUs

One production environment

One test and development environment

Small volumes of raw data moved through standard star-schema style batch extract, with the total target warehouse size of 180 GB of data (60GB of data loaded per year).

Standard next-business-day support

Note: please specify the number of servers and CPUs used to support this configuration

ODI CPU counts are for the existing servers in the described environment. ODI does not require any additional server.

List price is \$23,000 per CPU.

Support is 22% of the final price

Scenario 2: Enterprise ETL

Multiple data warehouses/marts with several different ETL projects

10 ETL developers, 3 administrator / operations roles

3 different (non-legacy) database source types, file-based extracts, one SAP source system, requirement to consume XML formatted data

3 target database servers for warehouses / marts - 12 4, 4 CPUs respectively

3 production environments for the three projects (but infrastructure, repositories, etc. is centrally managed)

3 half-scale development environments

3 half-scale test environments

Moderate volumes of raw data moved through standard star-schema style batch extracts for two projects, with the total target sizes of each at 500 GB of data (~160 GB of data loaded per year).

One larger project environment with 2 TB of data (~650GB of data loaded per year), through more complex rules in batch mode, plus small amounts of data streaming in

through your choice of either message queues / ESB / event publishing and then processed through your choice of either on-demand or in mini-batches. Specify the preferred option for streaming data consumption for the purpose of this pricing exercise. Assume an 8 CPU database server is this is important.

Note: this environment requires team-based development support and project-level security and roles.

Enterprise level (same-day response) support including off-hours and weekends

Note: please specify the number of servers and CPUs used to support this configuration

ODI CPU counts are for the existing servers in the described environment. ODI does not require any additional server.

List price is \$23,000 per CPU.

CDC usage would be needed or use overall micro-batch scenarios.

SAP source – 2,300K per CPUs

Support is 22% or the final price

9. Performance Features

There is no easy way in a class setting to demonstrate performance. Instead, the vendors have been asked to describe features in the product that specifically address performance needs and answer common performance questions like the following.

- What features are available to deal with large volumes of source data?

ODI will leverage native SQL for the underlying databases as well as the native utilities provided by the database vendors, hence guarantying the best possible performance.

- How does a developer set up or create an extract that can execute in parallel? Does it require special transforms or is it the same ETL logic regardless of parallel degree? Does it require changes to the database like table partitioning?

The exact same extracts can be run sequentially or in parallel and will not require any changes to the database – unless the database itself does not support parallel loads of course.

- What features are available for caching of source data or lookup values, and what are the limitations?

ODI will automatically create and maintain staging tables as required by the different processes. It is possible to configure ODI to keep the staging tables once

they have been created to re-use them for other processes – and thus cache the results from previous extracts.

Keep in mind though that thanks to the ELT architecture, caching is not as needed as it would be with an ETL architecture. Building a lookup with a dimension is simply generated as a join in the target database.

- How can the product be configured to run extracts on more than one physical server, if this capability is available? What must the developer do differently to take advantage of these features?

ODI provides a notion of context. Administrators will define as many contexts as needed (for instance Development, QA, Production – or Boston, Paris, San Diego, Tokyo). They will then define the connectivity to all the databases and file servers that are needed in these environments, and attach them to “logical schemas” – in essence aliases that represent the data model that the developers will use. From then on, the developers will design one single set of code, which will run in any context as defined by the administrators.

The additional benefit of this architecture is that administrators can change the location of the databases, can modify the connection information as well. As long as they update the information in the ODI topology, these changes are completely transparent to the developers.

- Can individual extracts be parallelized to run across servers?
Yes. Parallel extracts can be explicit (different models) or implicit (same model on several servers). In the later case, developers and administrators can either use contexts, or leverage variables in the topology definition.
We have customers running the same process in parallel against thousands of schemas...
- Are there grid / concurrent processing capabilities? If so, how do they work?
Yes – leveraging database grids. In most cases, the database grid is transparent to ODI.