ORACLE

INSURANCE

Oracle Insurance Insight

Oracle Insurance Insight Implementation Guide

version 7.0

Part Number: E22075-01

December 2010

ORACLE

Oracle Insurance Insight Implementation Guide

Version 7.0

Part Number: E22075-01

December 2010

Contents

Preface

VERSION

This manual corresponds to Oracle Insurance Insight (OII) version 7.0.

PURPOSE & AUDIENCE

The *OII Implementation Guide* is written for both a non-technical and a technical audience. The chapters are grouped into three content categories or sections:

Section I - Strategic Goals (Big Picture Ideas)

- **Chapter 1 - Introduction**
    - Presents the business objectives that OII was designed to address

Section II - Project Planning (Managing Timelines)

- **Audience** – Project managers
- **Chapter 2 - Implementation Lifecycle**
    - A standardized approach to implementing the OII system from scratch

Section III - Implementation (Technical Details)

- **Audience** – Implementation team
- **Chapter 3 - Program Flow**
    - Familiarizes the user with the OII data architecture
- **Chapter 4 - Source-to-Staging ETL**
    - Outlines the key goals and best practices for building an Extraction-Translation-Loading (ETL) layer that feeds OII
- **Chapter 5 - Data Loading**
    - A step-by-step guide to loading data from Staging to Warehouse to Data Marts
- **Chapter 6 - Data Visualization**
    - A step-by-step guide to updating and linking the Oracle Business Intelligence Enterprise Edition (OBIEE) Repository

For the purpose of the guide, it is assumed that all components have been installed and all environments have been setup. This guide walks the user through the planning and execution of an OII implementation.

For better understanding, the Implementation chapters assume knowledge of:

- Database management systems (DBMS), specifically Oracle 11g,
- Programming Language Structured Query Language (PL/SQL) and
- Oracle Data Integrator (ODI) 11g

List of Acronyms and Abbreviations

To make using this guide easier, all the full name is used the first time an acronym appears. In addition, all acronyms used in the *OII Implementation Guide* are listed below:

- CSI – Customer Support Identifier
- DBMS – Database Management System
- ETL – Extraction, Translation and Loading
- LOB – Line of Business
- OBIEE – Oracle Business Intelligence Enterprise Edition
- ODI – Oracle Data Integrator
- OII_DM – OII Data Mart schema
- OII_ST – OII Staging schema
- OII_WH – OII Warehouse schema
- OII – Oracle Insurance Insight
- OII Core – OII Core Data Model
- OLTP – Online Transaction Processing
- PL/SQL – Programming Language/Structured Query Language
- SCD – Slowly-Changing Dimension
- UAT – User-Acceptance Testing
- UDF – User-Defined Field

NAMING CONVENTIONS

To make using this guide easier, database objects are capitalized and italicized throughout the *OII Implementation Guide*:

- Example: *OII_ST.PLCY_TRANS (Policy Transaction)*
  - *OII_ST* – The relational (i.e. database) schema name
  - *PLCY_TRANS* – The relational (i.e. database) table name
  - *Policy Transaction* – The logical table name

RELATED DOCUMENTS

For more information, refer to the following documents:

- *Oracle Insurance Insight Release Notes*
- *Oracle Insurance Insight Installation Guide*
- *Oracle Insurance Insight Administration Guide*
- *Oracle Insurance Insight Warehouse Palette User Guide*
- *Oracle Insurance Insight User Guide*

CUSTOMER SUPPORT

If you need assistance with OII, you can log a Service Request at My Oracle Support (https://support.oracle.com). You will need your Customer Service Identifier (CSI) to register.

For community support, you can login to Oracle Mix (http://mix.oracle.com) and join the Insurance Business Intelligence and Data Warehousing group (https://mix.oracle.com/groups/18191) to review posts and/or create your own post to request assistance.

Address any additional inquiries to:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Section I - Strategic Goals

Chapter 1

Introduction

The Introduction answers high-level “What” questions. Subsequent sections go into more detail and answer “How” questions. The Introduction serves two purposes:

- For an executive audience, this chapter answers the key questions about what OII can do and what problems it solves. This audience may not need to read beyond the Introduction.
- For a technical audience, this section gives perspective on what is meant to be accomplished in an OII implementation. It gives relevance to the technical details discussed throughout the rest of the guide.

What Lines of Business Are Supported?

Six pre-defined Lines of Business (LOBs) are included with OII out of the box:

- Personal
  - Dwelling Fire
  - Homeowners
  - Personal Auto
  - Personal Umbrella
- Commercial
  - Commercial Auto
  - Commercial Property

Additional LOBs can be created from scratch or by using one of the six pre-defined LOBs as a starting template. Through the flexibility of the OII Adaptive Data Mode, any LOB can be edited in the Warehouse Palette. See the *OII Warehouse Palette User’s Guide* for further information.

What Is Included in the Core Data Model?

Many details about claim and policy transactions are common across different carriers and LOBs. The OII Core Data Model (OII Core) represents that part of the data architecture that is shared by all LOBs. All metrics and universal claim and policy transaction details for all LOBs reside in the OII Core and are exposed in the OII Corporate data marts:

- Monthly Snapshot Marts
  - Claim
  - Combined Policy & Claim
- Transactional Marts
  - Claim
  - Policy

Metrics are also presented with rich, LOB-specific dimensions in an OII LOB detainment per each implemented LOB:

- Combined Policy & Claim Monthly Snapshot

The OII Core provides support for up to six Corporate data marts and one LOB-specific data mart for each of the six included LOBs for a total of potentially 42 data marts:

- Monthly Snapshot Marts
  - Corporate Claim
  - Corporate Combined Policy & Claim
  - Corporate Policy
  - LOB Combined Policy & Claim
- Transactional Marts
  - Corporate Claim
  - Corporate Combined Policy & Claim
  - Corporate Policy

What Are Dimensions?

The concepts and calculations surrounding premiums and losses are contained within the OII Core. Loss Reserve calculations, for instance, remain the same regardless of LOB. The dimensions, or risk items, are what differentiate one LOB from another. A dimension can be a covered item such as a vehicle. A dimension can also be something that provides additional detail about the policy such as a garage on a Personal Auto policy. Each of the supported LOBs is a collection of dimensions that describe that line of business.

For example, the pre-defined dimensions for Personal Auto include Vehicle and Driver. Looking at a metric across more than one dimension such as Paid Loss by Territory and Month is multi-dimensional analysis. A practically infinite number of multi-dimensional views are available in OII.

Creating and viewing reports for OII in OBIEE is covered in-depth in the *OII User Guide*.

What is the Adaptive Data Model?

In adaptive data modeling, the configuration and creation of certain parts of the warehouse (interfaces, ETL, data model, etc.) is automated. LOB dimensions (e.g. risk items) are controlled directly by business users. The IT department has complete flexibility to scale the application appropriately, removing traditional modeling constraints. Business users also manage custom modeling, while internal ETL build-out is auto-generated by the Warehouse Palette.

The Warehouse Palette is a web-based application that allows users to create a LOB and configure the dimensions and associated attributes that make up the LOB. Its purpose is to provide users with an easy-to-use interface that lets them build and modify a LOB's components as they see fit. Once the user "publishes" the LOB, the physical database structures and loading mechanisms (i.e. the interfaces and the internal ETL) are automatically added to OII.

IMPORTANT Once a LOB is published using the Warehouse Palette it is locked for editing. A published LOB must be "unpublished" in order to make additional edits. The process of unpublishing deletes all data for that LOB and removes any LOB-specific structures from the database.

Instructions on how to extend the OII data model are contained in the *OII Warehouse Palette User Guide.*

What Are Slowly-Changing Dimensions?

Dimensions come from three sources in OII:

- Attributes in the OII Core that relate to all LOBs (e.g. Class, Limit, Producer)
- LOB-specific risk items contained in the metadata supplied for a pre-defined LOB (e.g. Driver, Dwelling, Vehicle)
- LOB-specific risk items (e.g. Watercraft) or other descriptive attributes (e.g. Construction Type) built in the Warehouse Palette

With such variability among dimensions, OII offers two widely used Slowly Changing Dimensions (SCDs) to manage changing dimensional data over time:

- Type 1 SCD
- Type 2 SCD

The Type 1 SCD methodology is a simple approach to dealing with dimensional attribute changes. Only the current state of the dimension is preserved. The fact table is unchanged. When a dimensional attribute changes, the previous value is overwritten with the current value.

Dimension bloat is minimized, and performance is optimized for a Type 1 SCD. The obvious disadvantage to this method is that no record of historical attribute values is kept.

A Type 1 SCD is a good choice if there is no value in maintaining old attribute values. Type 1 may also be a good choice if the dimensional attributes are expected to change frequently and there is little value in maintaining history.

The Type 2 SCD methodology is the primary technique for tracking historical data. Given a Natural Key supplied in Staging, Type 2 SCDs generate a new dimensional record whenever a new attribute value is loaded. These new records, identified by a surrogate key, are linked to the fact table. With each new dimensional record added, the fact table is automatically partitioned in time and permanently bound to the dimensional attributes that define that time slice.

Type 2 SCDs preserve unlimited history, allowing for powerful data analysis. As a consequence, storage of unlimited history and query performance across an expansive table may affect performance.

A Type 2 SCD is a good choice if historical analysis offers value and the dimension truly is slowly changing. Type 2 may not be a good choice if the attribute in question is subject to frequent change.

Each individual dimension can be set as a Type 1 or Type 2 SCD in the Warehouse Palette. This setting affects how changing dimensional data is handled in the Data Mart schema. In the Warehouse schema, full transactional history is always preserved for detailed analysis. A Natural Key is required for a dimension table if Type 1 or Type 2 is selected. History cannot be maintained without the presence of a Natural Key in every row of the dimension table.

What Are Metrics?

Metrics are quantifiable measures that represent a piece of business data as seen in relation to one or more dimensions. An example would be Earned Premium by Month. In this case, Earned Premium is the metric and the dimension would be time (i.e. Month). All metrics are contained in the OII Core. There are too many metrics in OII to name them all here. Examples of other metrics include:

- Claim Count
- Future Written Premium
- In Force Premium
- Incurred Loss
- Loss Reserve

See the *OII User Guide* for a complete list of metrics.

Earned Premium deserves a special note because OII can calculate it in a number of ways. If raw premiums are included for policy transactions, OII can calculate Earned Premium using one of these methods:

- Monthly (1/24th)
- Daily
- Seasonal (utilizes a pre-defined table with seasonal factors)
- Original Premium

Alternatively OII can also accept pre-calculated Earned Premium amounts and forgo internal calculations. Earned Exposure is also treated in this manner.

What Is ODI (Data Loading)?

ODI is an ETL tool that is used to load the OII data model from Staging to Warehouse to Data Marts. Custom dimensions built in the Warehouse Palette are published to the Oracle database and added to the OII Core. In the publish process, ODI is updated to handle new dimensions just as it would the OII Core components. Using ODI to load OII is covered in-depth in *Chapter 5: Data Loading*.

What Is OBIEE (Data Visualization)?

OBIEE is a data analysis tool that sits on top of the OII data model, providing dashboards, reports and custom analytic queries that help bring out novel characteristics inherent in the data. OBIEE use and configuration is covered in-depth in the *OII User Guide*. In OBIEE, dimensions are known as “filters” because of their ability to focus analysis on limited, relevant data.

See the *OII User Guide* for a complete listing of pre-defined filters, including those supplied with the six pre-defined LOBs. Linking OBIEE to the OII data model is covered in-depth in *Chapter 6: Data Visualization*.

Section II - Project Planning

Chapter 2

Implementation Lifecycle

STANDARD OII PROJECT STAGES

The standard OII implementation process is separated into four project stages as follows:

1. Initiation
2. Planning
3. Execution
4. Deployment

Initiation

In the Initiation stage, the customer's current source system(s) and requirements are assessed. Based on the estimated transactional data volume and allowing for growth hardware sizing can be assessed to determine configurations needs.

Planning

In the Planning stage, the following occurs:

- A detailed data analysis is performed to determine which fields will be exposed within OII, where they come from in the client's source system(s) and establish valid values.
- Data Mapping is performed in order to map the source system(s) data to OII Staging tables. This involves mapping of policy and claims data.
- Source codes are defined and standardized by the client in order to accommodate the OII data model and to ensure data consistency over time.
- Entities, operation, distribution and other hierarchies are defined prior to Source-to-Staging ETL development in order to enforce the correct parent-child relationships.
- The client's Earned Premium, Earned Exposure (if implemented) and Incurred Loss calculation methods are determined in order to configure OII for the selected pre-defined calculation methods.
- Signoff Metrics/Tolerances are established in order to commit to criteria for successful testing.
- Hardware is procured and configured. Required software is installed.

Execution

In the Execution stage the following occurs:

- The Source-to-Staging ETL is also designed and developed.
- Testing/balancing is done from the source system(s) data to the staging tables. The Balancing Reports provided with OII can be used to balance from the staging tables to the warehouse.
- The Source-to-Staging ETL is re-tooled, as necessary. No calculations are performed on the Staging tables, so balancing from the source to the Staging tables is an iterative process of loading, testing and re-tooling the Source-to-Staging ETL.
- Warehouse and Data Mart loads are run followed by balancing. The Balancing Reports provided with OII can be used to balance transaction amounts and counts between the OII schemas: Staging, Warehouse and Data Mart.
- Routine data loads are configured in ODI. The loads must be configured to ensure there are no data gaps or duplications and that the process can integrate with the source system, month-end processing cycles, etc.
- User Acceptance Testing (UAT) is performed. After the OII's numbers match the agreed tolerances, OII is presented to the end users.
- Based on end-user feedback and request from the end users the OII/OBIEE data views can be customized.

Deployment

In the Deployment stage migration to production, administrative training, end-user training and transition to support takes place.

Standard OII Implementation Process

The timelines below are estimates for a standard OII implementation of a single LOB (e.g. Commercial Auto or Homeowners) that includes policies and claims from a single source system. The time ranges could increase or decrease based upon the number of LOBs, the number of source systems and the complexity of data extraction from the source system. Coordinating multiple source systems and multiple LOBs in an OII environment often requires coordinated data mapping and ETL design so that a consistent view can be presented across all LOBs.

Data Mapping

- **Scope:** During this phase, the required and desired fields to be exposed within OII and the corresponding fields in the source system(s) are identified. Customizations needed for the Adaptive Data Model are identified. Sign-off metrics and tolerances are established during this period. Allocation of server resources and storage space are arranged prior to the ETL Development phase. Accommodations are also made to store the staging data.
- **Client Resources:**
  - One or more BAs who are familiar with the data from an insurance and end-user perspective
  - One technical resource who is familiar with accessing data and data architecture at a high level in the source system(s)

- **Assumptions:**
  - N/A
- **Risks:**
  - Multiple source systems increase the complexity and required time for this phase.

SOURCE-TO-STAGING ETL DEVELOPMENT

- **Scope:** During this phase, the OII Warehouse Palette is used to customize the Adaptive Data Model. From here, the ETL is designed, developed and re-tooled in order to perform any necessary data cleansing, move data into the staging tables (a.k.a. templates), load the warehouse tables, define the standard codes and balance from warehouse to staging tables.
- **Client Resources:**
  - One Business Analyst who can validate the relevance and appropriateness of the data being extracted from the source system(s) and sign off on the data reconciliation metrics between source and staging
  - One technical resource who is familiar with accessing data and data architecture at a high level in the source system(s)
  - One technical resource who can code and test the ETL using SQL and/or an ETL tool and execute the data load to staging
- **Assumptions:**
  - The Business Analyst resource should be readily available to answer questions concerning the appropriateness of the data being extracted from the source system(s).
  - The two technical resources may be combined into one job role if a single resource has the breadth of knowledge of the data architecture and the skill set to code the ETL.
- **Risks:**
  - Poor data quality will require that more time be spent cleaning the data. Ideally, data cleansing will be done in the source system(s) so that the source is consistent with what is loaded into OII and so that data manipulation is reduced in the ETL.
  - Multiple source systems increase the time needed to develop separate loading mechanisms and convert different code values from various systems into one consistent representation of the data.
  - If the required (Policy or Claims) fields were not properly identified and addressed in the ETL, then it will take additional time to re-tool the ETL or investigate ways to accommodate the missing data in OII.

Data Mart Load

- **Scope:** The transactional and monthly snapshot data marts are loaded, and the OBIEE Repository is customized to accommodate changes per the Adaptive Data Model.
- **Client Resources:**
  - One BA who can communicate with the end users and sign off on the data reconciliation metrics between the warehouse tables and data marts
  - One technical resource who is familiar with accessing data and data architecture at a high level in the source system(s)
  - One technical resource who can re-tool the ETL using SQL and/or an ETL tool and execute the data load beyond the warehouse
- **Assumptions:**
  - OII reconciles to data in the templates and not necessarily to historical customer reports.
- **Risks:**
  - Validation of mart metrics often involves a process of discovery where missing data is identified, the ETL code is updated and data is reloaded through the system (staging, warehouse and data marts). Focus on data reconciliation at the warehouse level will identify any issues earlier when it takes less time to reload. Waiting until the Data Mart Load to perform detailed data analysis may lead to additional iterations of this phase, increasing the amount of time necessary beyond the time estimate given.
  - There can be confusion around calculated mart metrics (counts, earned premium, etc.) because customer reports may represent a different view of the same data. Both views may be correct and true representations, but the measure of success for the marts is being able to reconcile the OII metrics to the warehouse and staging tables not necessarily to a customer report. Any differences that cannot be reconciled may ultimately be due to incomplete staging data (via the ETL).

Customize OII Data Views

- **Scope:** In this phase, user requests are translated into custom OBIEE reports and data views.
- **Client Resources:**
  - End-user feedback (dashboard/report requests) and availability for training

Section III - Implementation

Chapter 3

Program Flow

The focus of this chapter is an overview of the flow of data between the processing stages in OII.

The core data model provided with OII is a reference implementation that is adaptable and customizable, therefore, an understanding of program flow is required to properly plan and implement an OII solution.

Note Keep in mind that the content presented herein is representative of the *reference implementation* of OII. The reference implementation is a model upon which an actual implementation is based. As such, actual implementation details may differ from the model.

The OII program flow utilizes three different data models for processing: *staging*, *warehouse* and *data mart*. Generally speaking, data is in slightly denormalized form within the Staging model. Depending on the source system data structures, the loading process may or may not include additional denormalization. During the flow of data from staging into warehouse some renormalization occurs. Finally, during the flow of data from warehouse into data mart data undergoes denormalization. The purpose of the de- and renormalization is to reduce redundancies, flatten hierarchies, and prepare for the final load into fact and dimension tables in the data mart.

Staging Model Concepts

The Staging model introduces several components which are important to understand:

- *Party Model*
- *Contact*
- *Postal Address*
- *Natural Keys*
- *Row Number*
- *Relationship Model*
- *Normalization and Denormalization*
- User-Defined Fields
- Concepts in Practice

These concepts are used to allow for flexibility and reuse in the system while maintaining a high level of performance in the data mart

Party Model

The Party Model is used to establish roles and relationships for entities referenced in transaction data. In this context, an entity can refer to a person or an organization, while a role is describes the entity with respect to the transaction. This concept is illustrated in the following example:

> *John Smith is a personal auto policy holder for Alamere Insurance Company. For data relating to his personal auto policy, John is a PERSON entity, and holds the role of INSURED for the Personal Auto LOB. Similarly, Alamere Insurance Company is an ORG entity, and holds the role of INSURANCE COMPANY in this LOB with respect to this transaction.*

The Party Model allows for an entity to hold more than one role or relationship within the system, and allows for the relationship to be established across LOB boundaries, which results in reduced data volume and enhanced reporting capabilities. This concept is illustrated by expanding on the previous example:

> *John Smith calls his agent and during usual customer service interaction, learns that Alamere Insurance Company also offers Homeowner's insurance policies. John requests a quote. In the system data, John is already established as a PERSON entity, however in the Homeowner's LOB he now holds the role of PROSPECT. His PERSON data is common to both LOBs.*

Finally, the Party Model introduces context and hierarchies for entities and roles within the system. The *relationship code* of the party establishes an entity's relationship to the transaction, or to other parties. See the concept illustrated below, built upon the previous examples:

> *John Smith's insurance agency is established as an ORG entity; similarly the agent is a PERSON entity. With respect to transactions involving John's policy, the agency holds the PRODUCER role while the agent holds the SUBPRODUCER role. The hierarchy of PRODUCER and SUBPRODUCER is expressed via the relationship code in the Party model.*

Contact

The CONTACT table provides flexibility for defining contact endpoints for physical entities such as people or office locations. Currently, the CONTACT table extends only a physical address as a contact endpoint, or entity type, which is housed in the PSTL_ADDR table. In the future, additional entity types can be added, such as phone numbers and email addresses. A row in the CONTACT table is associated with a row in the PARTY table and a row in the PARTY table can be associated with one or more rows in the CONTACT table. A row in the CONTACT table is also associated with a row in the PSTL_ADDR table, where the location information is stored.

> *John Smith is a Personal Auto policy holder with Alamere Insurance Company. He pays a monthly premium for his policy. When his monthly payment transactions are loaded to the OII system, John's home address is listed as a row in the PSTL_ADDR table. This row is then referenced by a row in the CONTACT table. The row in the CONTACT table is then referenced in the PARTY table, which contains a reference to John's entry in the PERSON table.*

Note Since CONTACT is related to PARTY, it is important to note that if there are no rows in the PARTY table, there will be no rows in the CONTACT table.

Postal Address

The PSTL_ADDR table is used to store address location. During the migration process to the warehouse schema, the geographic boundaries present in the address are moved into a separate table called *GEOG_BNDRY* and a reference is created between the PSTL_ADDR and GEOG_BNDRY tables. There are multiple types of geographic boundaries:

- State or Province
- Region
- Country

Regionalization of data by geographic boundary provides the OII user with a sufficiently granular level of detail while not burdening the dimensions in the data mart schema with street information (i.e. Address 1 and Address 2 fields) that is typically more useful in an operational environment (e.g. the generation of mailing labels). Once data for Address 1 and Address 2 are migrated to the Warehouse, it undergoes a regionalization transformation. All data is preserved at the Warehouse level, while only the geographic boundary information is moved to the dimensions in the data mart schema.

.

Note The basic installation for OII is a *model*, so if requirements dictate a need to maintain street address information in the data mart, the implementation team can readily accommodate this request easily.

The PSTL_ADDR table is used by the CONTACT table when storing address information related to the PARTY table. In addition, the PSTL_ADDR table is also used to store address information related to vehicles and claims.

Natural Keys

Natural Keys are unique data identifiers of specific elements within the Staging data model. Natural Keys typically associated with several types of data elements including Entities, Parties, Claims and Policies. A Natural Key serves multiple purposes, one of which is the ability to associate rows in one table with one or more rows in one or more other tables. Natural Keys are one method to keep data loads reduced which increase the performance of a data warehouse system by eliminating data duplication. Additionally, Natural Keys are a good method to use for ensuring the correlation between one or more tables for reporting purposes.

For example, a Natural Key is used in conjunction with the Party Model to allow association of a single entity with more than one role. Without the Party Model and Natural Keys, data duplication would occur – over time, this duplication results in reduced system performance and increased overhead. The Natural Key ensures that the correlation between an entity and multiple roles is valid. It is important in reporting to identify when roles are held by the same person or organization.

During the loading process, if a Natural Key is not supplied by the source system, a Natural Key is generated as a checksum of the *identifying columns* – that is, the collection of columns that uniquely identify a row – and is assigned to each transaction that is loaded from the customer data. If one or more rows of incoming data have the same values in the identifying columns, the generated checksum, the Natural Key, will be the same. Thus, it is important to ensure that the identifying columns are indeed serving the purpose to identify unique rows. The alternative is for the customer to supply a Natural Key.

The use of Natural Keys in OII has several caveats that are important to note for implementation teams:

- Every column with the exception of Natural Keys in the Staging model tables and Natural Keys used for counts in the monthly snapshot tables within the marts must be populated with data – that is, not null. Special notes on null values – with respect to fields with data type of string, a single character, default a single space is used to represent a lack of data for a column that does not allow nulls. This single character can be configured in the OII_SYS_CONFIG table.
- Customers can supply their own Natural Keys from source systems provided the unique string requirement is met.
- Supplying Natural Keys must be performed in an all-or-nothing approach per entity; either all rows within the entity will have a supplied Natural Key or none of them will.
- If Natural Keys are empty, OII will compute Natural Keys in the Staging-to-Warehouse data population routines.
- If Natural Keys are computed, there must be enough sufficient non-empty data to establish a unique distinction between data rows. Otherwise, logical errors will occur.

If Natural Keys are computed, Slowly Changing Dimension (SCD) support is not available – it is impossible to implement Type 1 or Type 2 SCD. An SCD is a dimension that, as the name implies, may change over long periods of time. If a Natural Key is not supplied, then it is computed based on a collection of fields that uniquely identify a row in a table. Since it is then possible for one or more fields in the collection to change, the computed Natural Key would then be different when data changes. As such, there is no method to allow correlation of the changed data to its original state.

These concepts are illustrated in the following example:

Alamere Insurance Company designs a data structure that defines "person" entities. The PERSON table that houses data rows for these entities consists of only two columns, FirstName and LastName. Alamere decides to use First Name and Last Name rather than tax identification numbers as keys to identifying entities in the person table for legal reasons. Alamere is not currently computing natural keys. During data load to OII, the natural key is computed for each entity (row) in the table. If the "person" table contains John Smith and Jane Smith, the computed natural key will suffice to uniquely identify the two individuals. However, if a separate John Smith is entered in the "person" table, a conflict will result in computing natural keys – there are not enough columns to uniquely identify each row.

Alamere has a "location" table which gives a physical location to a row in the "person" table. If Jane Smith changes her last name to Jones, which results in an update to the "person" table, the ability to correlate the location data will be gone – the computed natural key for Jane Jones will be different from the computed natural key for Jane Smith.

The solution then is to either add more columns to the "person" table which guarantee uniqueness, or to have the source system generate a natural key that is added to each row prior to import into OII.

Row Number

In a typical relational database schema, a primary key is used to uniquely identify a row of data in a table. Such keys are then used to provide references between different tables.For example, a primary key in an Insured table would be referenced as a foreign key in a Policy table, indicating a relationship between data in the Policy and Insured tables where the foreign key matches the primary key. Because the database schema is particular to the company and in many cases the business units therein, there is the potential to have as many different table definitions as there are business units and companies.

To simplify creation and maintenance of ODI Interfaces which populate the Staging model from customer data, the interfaces have been programmed to modify imported data by removing primary and foreign keys which results in destruction of referential integrity.

However, there still exists a need to be able to generate a holistic view of a transaction across the various tables in the model, and as such the data in Staging has introduced a unique identifier column: row number (*ROW_NUM)*. The row number is always present for every staging load, regardless of how natural keys are provided or generated. The row number is simply a unique number, very similar to an identity column in SQL Server, that increments per row (or rows, when relationship codes are needed for uniqueness) within a given staging load to uniquely identify a transaction.

The ROW_NUM is used in each table in the Staging environment for all rows applicable to that unique business transaction. As such, a *virtual transaction* can be built from the tables in the Staging model using only the ROW_NUM.

Relationship Model

In OII the relationship model is used to provide context and hierarchy in which multiple rows in a given table can be related. Relationship codes are maintained in the OII_SYS schema.The relationship code is expressed in tables within the Staging schema using a column named like *object_RLAT_CD*, where *object* is the type of entity contained in the table. This column is used to provide a part of the primary key of the table in addition to context and hierarchy. An example is shown below:

> *The OII_ST.PRODR table contains Producer entities, and features the PRODR_RLAT_CD column. In practical applications, a Producer may have Sub-producer that is related. The Sub-producer would also have a row entry in the PRODR table, and the relationship between the two rows for the Producer and Sub-producer is represented in the values contained in the PRODR_RLAT_CD.*

The relationship model works in conjunction with the entities in the Party model to establish context between parties and transactions, and establish relationships between parties with respect to transactions.

Normalization and Denormalization

Relational Databases used in *online transaction processing (OLTP)* are generally designed in a systematic way that ensures the data structures are useful for querying and possess performance characteristics that support OLTP. OLTP databases are characterized by a high volume of small transactions, such as payment processing for policies.

Normalization describes the design pattern in which the data structures meet the needs used by OLTP databases. For the purposes of OII, the main normalization characteristics that are import to understand are table relationships built by keys, and the use of joined tables. Normalization attempts to optimize database performance for inserting, updating or deleting records by storing different but related data in separate tables.

By contrast, *denormalization* attempts to organize database structures in such a way as to provide optimized performance for reading data. Databases used for *online analytical processing (OLAP)* are usually designed with denormalized database structures. Since OLAP applications are used to analyze trends in historical data over long periods of time, the sheer volume of accumulated data requires either a high-performance database or a database structure that is built for fast reading. An OII implementation provides both of these.

OII is an OLAP system, so the end-result of loading and program flow will result in denormalized data in the data mart.

User-Defined Fields

Since the data model provided by OII is a reference implementation, it is expected that the schemas will be customized for each implementation. As such, most tables in the Staging and Warehouse schemas feature five fields that can be used to house data not contained in other areas. These User-Defined Fields (UDFs) allow for the data model to accommodate minor additions without having to unpublish, edit and republish an LOB. The naming of these fields will always contain “UDF” in the field name (e.g. PARTY_UDF_CD_1). The UDFs are normalized in Staging and Warehouse. During the migration to Data Mart, the UDFs are denormalized in the dimension tables. As such, the dimension tables in the Data Mart may contain up to 30 UDF columns to accommodate the 1-5 UDF columns in each entity table. None of the UDF’s are included in the presentation layer. This means that the end user will not see anything in the OBIEE user interface for those UDF’s that have been utilized unless they are given a meaningful name, data is loaded to them and they are added to the presentation layer in the OBIEE Repository to make them available to the end user. *Chapter 6 - Data Visualization* gives detailed instructions on how to enable custom dimensions, fields and UDFs in OBIEE.

Concepts in Practice

The Party Model works in concert with Natural Keys to support the concept of multiple roles per entity. In the Staging model, there are no Natural Keys established for roles (e.g. INSURED or INSURANCE CO) or entities (e.g. PERSON or ORG). Instead, Natural Keys exist for each PARTY row. The relationship between the ENTITY and ROLE is established by ROW_NUM, and the hierarchy/context of the relationship is established by the relationship code (PARTY_RLAT_CD). In this manner, the Natural Keys for PARTY need not be reloaded for each ROLE, since an ENTITY may have more than one ROLE. The end result of the application of these concepts in OII:

- Less data is duplicated
- An ENTITY can hold more than one ROLE, and can be utilized across LOBs and subject areas (Policy Transactions, Claims Transactions, Quote Transactions, etc.)
- Business transactions can be presented holistically despite denormalization

The importance of Natural Keys for identifying rows cannot be understated – in order to ensure that the initial data load and subsequent loads have consistency, the Natural Key process must be defined accurately. This means that the customer must supply a Natural Key for each unique entity or must identify the collection of uniquely-identifying rows. The Natural Key identifies an entity in the data mart that is unique in its lifecycle. This concept is illustrated in the following example:

John Smith holds a Personal Auto policy with Alamere Insurance Company. Alamere chose to generate Natural Keys during the load to the Staging environment. When John's policies are loaded to the Warehouse, each Policy entity and related items (e.g. Vehicle entities, Driver entities) are related by the Natural Keys associated with these items. This includes the John Smith PERSON entity.

Subsequent to the historical load, Alamere issues a new Homeowner's policy to John. During the next load to the data mart, the Homeowner's Policy and related entities will be loaded, with their computed Natural Keys. John's PERSON entity, however, will use its existing Natural Key, and as such, will not be duplicated during the data load. Instead, the new Policy and related entities will be linked to the existing PERSON entity, thereby reducing data redundancy.

The sample example applies if Alamere chose to use identifying columns to compute a Natural Key.

Flow Diagrams

The overall program flow uses Oracle Data Integrator (ODI) as the ETL tool for moving data between schemas. The OII installation includes several packages in ODI which are used to load Warehouse and Data Mart schemas. For the reference implementation, there is an additional package used to load the Staging schema, however in practical application, the implementation team or customer is responsible for designing and implementing the Staging load process. ODI is a natural choice for this process; however the customer may have existing ETL tools and procedures that can be used as well.

It is important to note that while ETL tools provide *translation* capabilities as part of their functionality, it is not recommended to perform translation on data loaded to Staging unless absolutely necessary. If translation becomes necessary, the customer and implementation team are advised to consider applying a similar translation to the source data so there are no discrepancies between the source data and the data contained in OII.

The tables and processes described in the following program flow diagrams are illustrative of the overall process. The actual program flow includes many more tables and denormalization procedures.

Loading to Staging

Data from customer systems is generally maintained in one or more database management systems, and is composed of tables that are related by one or more keys. This arrangement of data is well-suited for maintaining transactional data for daily operations but is regarded as less than optimal for data warehousing. As such, the initial data load to the staging environment begins the process of denormalizing relational data. The following diagram illustrates a typical denormalization of a customer's relational data during the load to the staging environment.

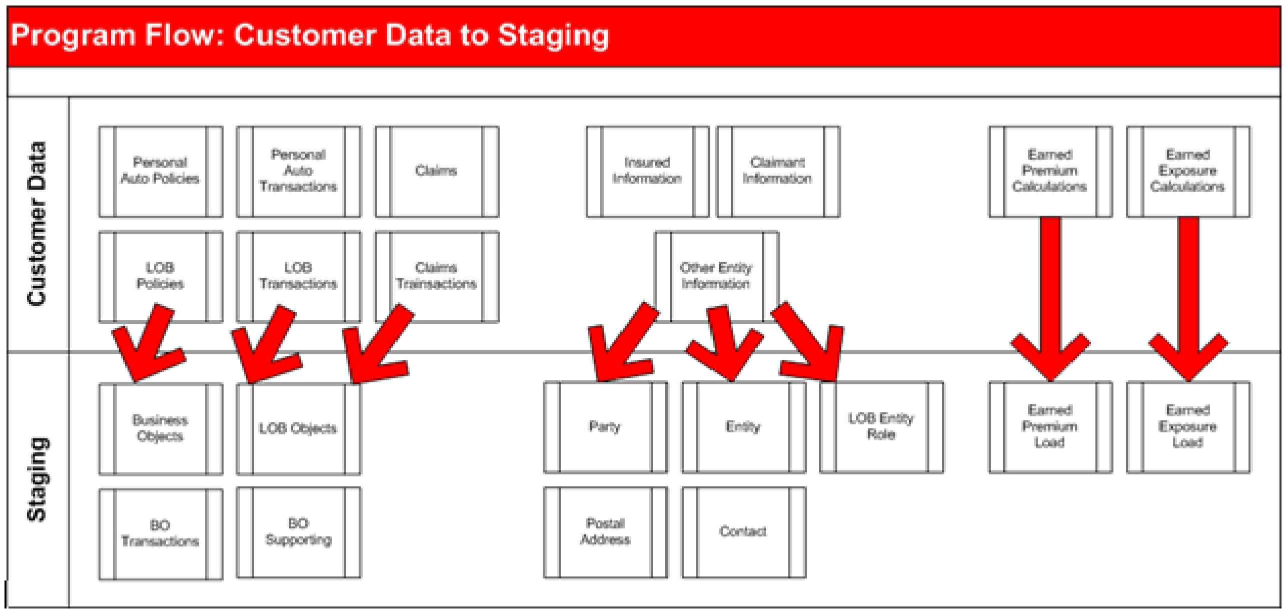


Figure 1: Loading to Staging Flow

The Customer Data row at the top of the diagram above illustrates multiple tables in a customer's enterprise data store. The leftmost cluster in this swim lane represents typical tables used for LOB-specific transactions and business data:

- **Business Data Tables** – data related LOB-specific business objects.
  - **Personal Auto Policies** – exemplifies an LOB-specific table housing business data.
  - **LOB Policies** – a representative placeholder object that is used to denote other groupings of LOB-specific tables (e.g. Homeowners Policies, Commercial Auto Policies, Umbrella Policies, etc).
  - **Claims** – data related to claims.

Note Claims are a special case since a claim may not be related to a policy, so claims are separate from policies.

- **Transactional Data Tables** – transactional data related to LOB-specific business objects.
  - **Personal Auto Transactions** – transactions for the Personal Auto LOB.
  - **LOB Transactions** – a representative placeholder that denotes other groupings of LOB-specific transaction tables.
  - **Claims Transactions** – transactions for Claims.

The center cluster in the top swim lane illustrates multiple tables containing information about *referenced entities* in the customer's data. In this context, an *entity* is an object which is referenced in LOB-specific data. The entities may or may not be LOB-specific in this case. Typical entities are:

- External Customers (Insureds, Policyholders, Claimants)
- Internal customers (Producers, Underwriters)
- Businesses (Insurers, Reinsurers, Suppliers)

Finally, the rightmost cluster illustrates the earned premium and earned exposure information that may exist in a customer's enterprise data structure.

As mentioned previous, in typical customer data structures the information contained therein is linked with enforced key relationships to provide referential integrity between the tables. A transaction in the LOB-Specific Policy Transaction table is related to a transaction in the Insured Information table, as well as the Policy Information table. These tables are typically wide, meaning they contain many columns to house the entity-specific data needed, such as name, address, contact information or additional attributes of the policy related to the transaction. This describes the typical relational database design that most customer data systems employ.

During the *ETL* process which loads the Staging environment, the key relationships are removed as data is denormalized into a collection of tables. Policy data that is universal across LOBs is segmented into a Policy Data table, while LOB-specific data is moved into separate tables. Data that references parties such as Insureds and Insurance Companies, is denormalized according to the party type (person or organization) and the role that entity plays in the transaction (Insured or Insurance Company). Additionally, address and contact information is denormalized into postal address information.

Of special importance is the handling of earned premium and earned exposure data. The latest release of OII includes the ability to import these types of transactions directly to the Staging tables. There is no denormalization that occurs during the load to Staging; instead the reporting data for earned premium and earned exposure is loaded according to the amount and reporting time period. It is not necessary to populate these tables – if they are left empty, OII will perform the necessary calculations during the Staging to Warehouse flow step.

STAGING TO WAREHOUSE

During load into the Staging schema, data is denormalized from the original references. This process uses the aforementioned ROW_NUM construct to maintain a consistency across the tables. When applying the denormalization across LOBs, a reduction in data redundancy can be achieved, and hierarchical data structures can be flattened.

When data is loaded from staging into the warehouse re-normalization is applied to the data in preparation for the final load into the fact and dimension tables in the data mart. The following diagram illustrates a typical renormalization flow from staging to warehouse.

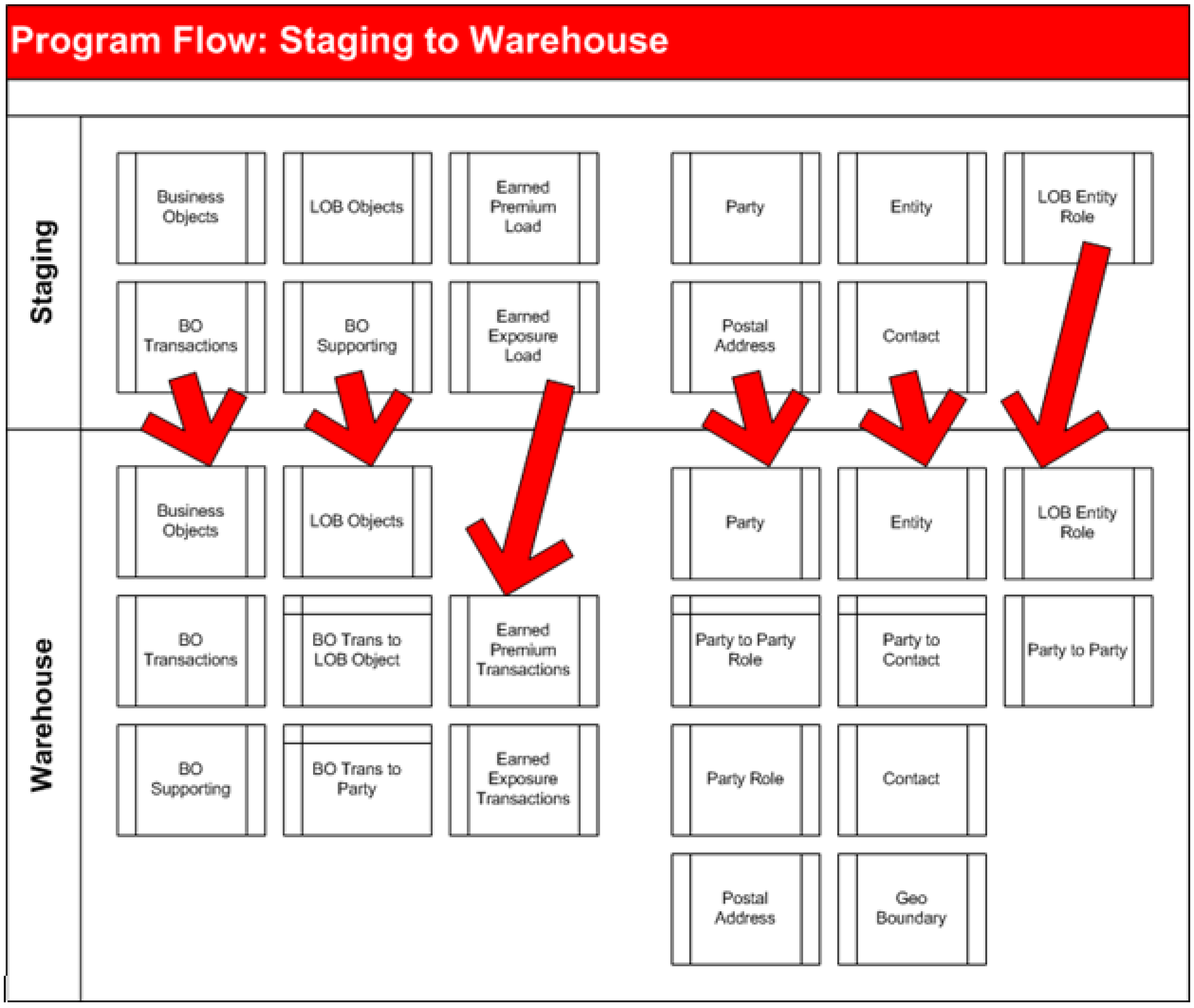


Figure 2: Staging to Warehouse Flow

The top row in the diagram above shows the representative tables present in the Staging schema, which is migrated to tables in the lower swim lane, the Warehouse schema. During the migration, the previously denormalized data is renormalized into OII data structures. This process includes several important steps:

- Creating *Bridge Tables*
- Introduction of ID columns
- Earned Exposure/Earned Premium Calculation

- Translation of relationship code
- Separation of Addresses

Bridge tables (also called many-to-many tables) provide a relationship between two tables, and in some cases provide additional data about the relationship. In the diagram, bridge tables are represented by the boxes with top and left borders, whereas data tables have left and right borders. By convention the bridge tables are generally named according to the two tables that are being joined in the relationship. In OII, bridge tables that have "_TO_" in the name.The following example illustrates a relationship-only bridge table:

> *The OII_WH.CLM_TRANS table contains a list of claims transactions. The OII_WH.PA_VEH table contains a list vehicles referenced by Personal Auto policies. The OII_WH.CLM_TRANS_TO_PA_VEH bridge table relates one or more vehicles in the OII_WH.PA_VEH table to a claim transaction in OII_WH.CLM_TRANS.*

The following example illustrates a bridge table that includes a relationship and additional data about the relationship:

> *The OII_WH.PARTY table contains a list of entities. The OII_WH.PARTY_ROLE table contains a list of roles that an entity may possess. The OII_WH.PARTY_TO_PARTY_ROLE bridge table supplies the relationship between a OII_WH.PARTY and a OII_WH.PARTY_ROLE, while also including a time span that indicates the duration that the entity held the role.*

In the load to the Staging schema, a ROW_NUM was used to provide a virtual transaction across the denormalized tables. When data is loaded to the Warehouse, ROW_NUM is replaced by an ID column which is then used as a reference for rows in other tables that refer to that specific entry.

> *The OII_WH.PARTY table contains a list of entities. As new rows are inserted into the OII_WH.PARTY table, OII checks that the new row is not a duplicate of an existing row. Each row in the OII_WH.PARTY table must be unique, a characteristic enforced by the PARTY_ID primary key column. By using this method, data redundancy is eliminated – each entity in the OII_WH.PARTY table is unique, and as new data is loaded, OII will reference existing entities rather than creating new rows.*
>
> *When OII_WH.PARTY is referenced in the OII_WH.PARTY_TO_PARTY_ROLE bridge table, the reference is made using the PARTY_ID column as a foreign key in the bridge table.*

If the earned premium and earned exposure tables were populated in the Staging schema, the information is carried over to the Warehouse schema. Otherwise, OII will calculate the earned exposure and earned premium values and populate the transactions into the Warehouse schema. OII supports multiple methods of calculating earned premium and earned exposure, such as seasonal and original premium.

Note Transactions for earned premium and earned exposure have granularity to months only. This applies to calculated and loaded earned premium and earned exposure transactions.

When the migration to Warehouse occurs, the relationship codes in the **_RLAT_CD* columns are translated into *type codes*, represented as columns named like **_TYP_CD*. The same hierarchical relationships and context present in Staging are preserved during the migration to Warehouse.

In the Staging schema, addressing information is stored in the OII_ST.PSTL_ADDR table. During the migration process to the Warehouse schema, the geographic boundaries present in the address are moved into a separate table called *OII_WH.GEOG_BNDRY*. A reference is created between the OII_WH.PSTL_ADDR and OII_WH.GEOG_BNDRY tables using the ID scheme mentioned previously.

Warehouse to Data Mart

The Data Mart schema is designed as a star-schema architecture with two types of tables: *Facts* and *Dimensions*. A *fact* is generally recognized to be a transaction. A *dimension* is an attribute of a fact. Put simply, if a transaction occurs today at noon for $5, the fact is the transaction amount ($5) and the dimension is the date and time (today at noon).

The data model for the data mart has been designed using the *normalized star schema* pattern, as opposed to the *snowflake pattern*. In practice, this means that the base dimension tables are linked directly to base fact tables and are normalized to reduce redundant data. The star schema is used to realize performance benefits offered by the database and query platform.

Note The snowflake pattern includes dimension tables that have related dimension tables, rather than all dimensions relating directly to the fact tables.

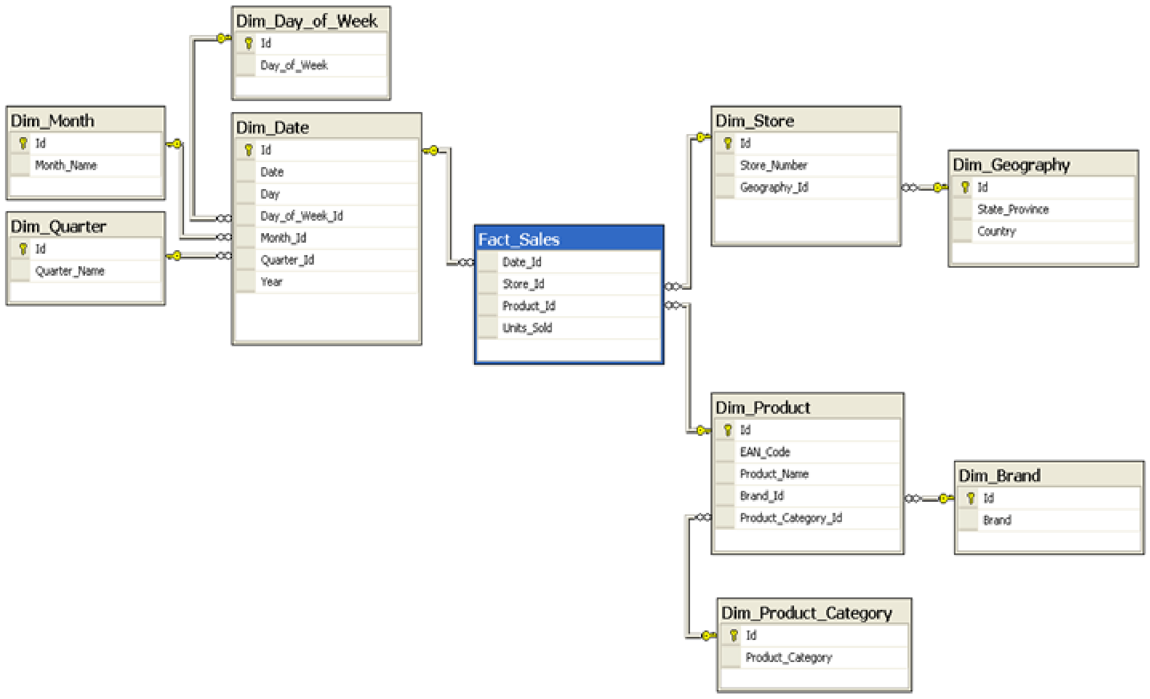


Figure 3: Snowflake Schema Example

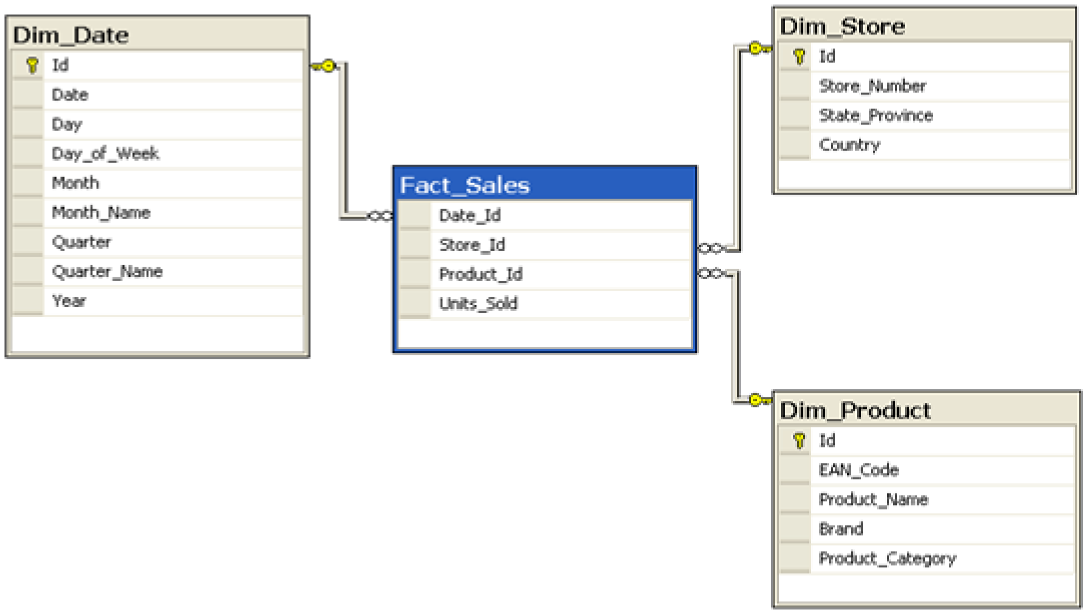


Figure 4: Star Schema Example

The data mart model includes four core fact tables. Of these, two tables contain transactions for policies and claims: PLCY_TRANS_FACT and CLM_TRANS_FACT. The remaining two core tables contain the monthly snapshots of transaction data for policies and claims: PLCY_MTH_FACT and CLM_MTH_FACT. Within these four core tables are all the metrics for all LOBs. Since all metrics are part of the core tables and do not vary, the same interface can be used to load the transaction fact tables regardless of LOB.

Through the use of LOB-specific *extension* fact tables, the four core fact tables are kept abstract and generic, and LOB-specific facts are stored in the extension tables. As such, core fact tables contain the common facts OF the related core common dimensions. This means much more efficient loading of adapted content, ease in creation of LOBs, and allows for significantly better performance during loads. An additional benefit is reduced storage needs, since there are no empty fact keys for unused LOBs. Keep in mind, of course, that this design is the reference implementation only and can be changed.

Fact-to-fact join keys are used to give a 1-to-1 extension of base fact tables to deal with LOB-specific dimension tables. For example, the PLCY_TRANS_FACT table has an LOB_FACT_ID column. This column is used to relate the PLCY_TRANS_FACT table to the LOB-specific table, PA_FACT via the PA_FACT_ID column. The relationship between the core fact table and the LOB-specific extension fact table is expressed by LOB_FACT_ID = PA_FACT_ID.

The relationship between the LOB-specific extension fact table and LOB-specific Dimension table is expressed in the PA_VEH_ID and PA_DRV_ID columns. The PA_FACT_ID column relates back to the core fact table PLCY_TRANS_FACT which provides a consistent LOB-specific view of transactions that is accommodated by the adaptive data model.

In practice, OII uses *views* to present data in the dashboards. *Corporate* views are based on the core fact and dimension tables – as illustrated above, the core tables are not LOB-specific and provide a holistic view of the business without the details specific to any LOB. By contrast, the LOB-specific views provide a holistic view of an LOB such that the core and extended fact tables appear as one LOB-specific fact table, thus allowing a drill-down from high-level cross-LOB metrics into specific metrics for a designated LOB.

Data loaded from the warehouse is denormalized when loaded into the data mart. The following diagram illustrates a typical denormalization flow from warehouse to data mart:

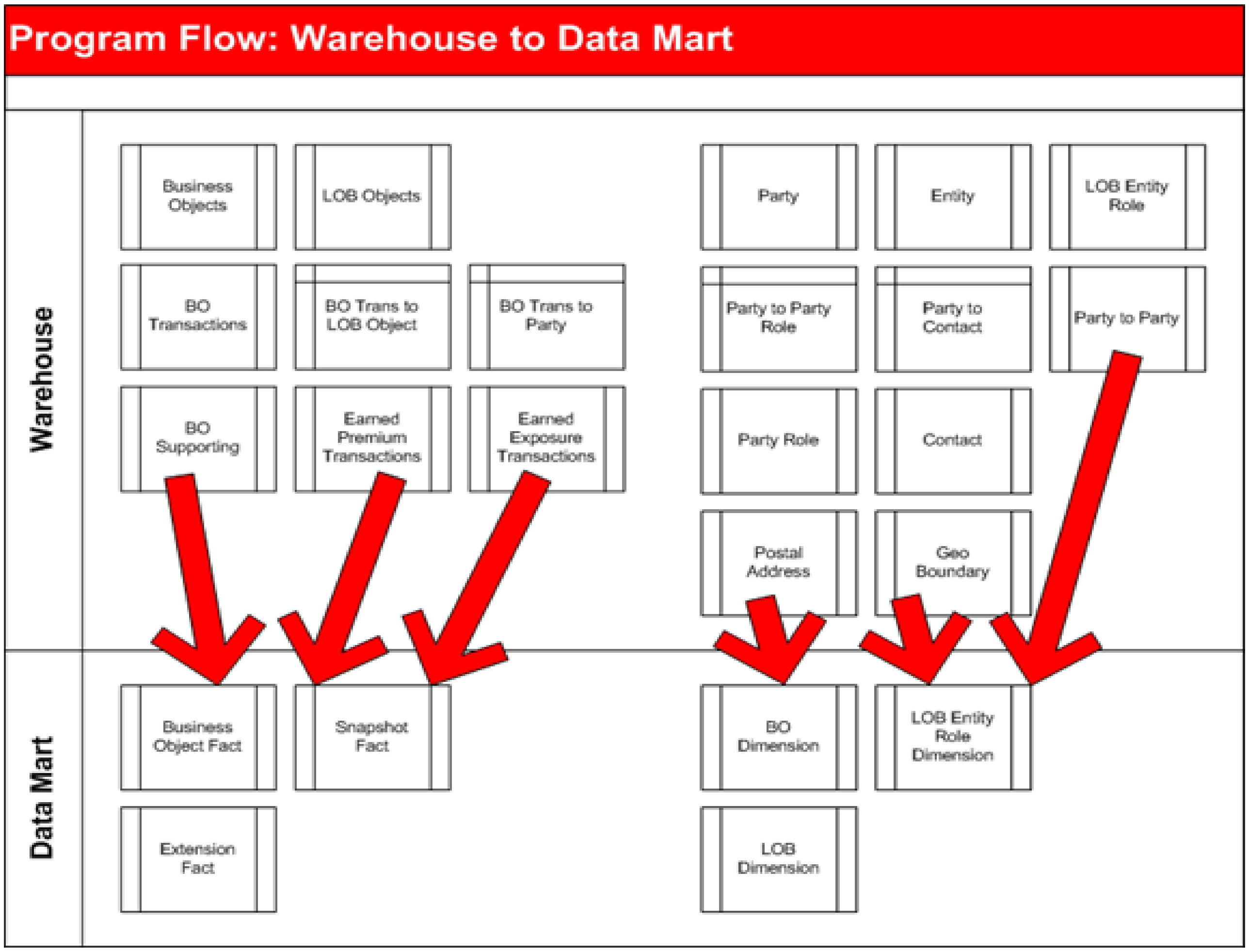


Figure 5: Warehouse to Data Mart Flow

As mentioned previously, transactional information from the warehouse schema is contained in multiple normalized tables, represented in the diagram above as:

- Business Objects (BO)
  - BO Transactions
  - BO Supporting
- LOB Objects
  - Related BO Transactions to LOB Object
  - Related BO Transactions to Party

- Earned Premium Transactions
- Earned Exposure Transactions

The transactions in the Warehouse are denormalized into the data mart schema, using the methodology to extend core metrics described previously, into the following types of tables:

- Business Object Fact
- Snapshot Fact
- Extension Fact

Similarly, normalized entity tables in the warehouse schema are denormalized into the data mart schema as dimension tables. The denormalization process is based upon the *role* held by the Party – as shown in the Party to Party Role bridge table – and as such, the parties are created in the appropriate role-based dimension table, illustrated in the example below:

> *A person in the warehouse PARTY table that is related to the role of Producer will be expressed in the PRODR_DIM table in data mart.*

Additional normalization occurs with PSTL_ADDR and GEOG_BNDRY tables. As mentioned previously the localized address information is discarded upon migration to the data mart. Regional information is maintained and written in to the role-based dimension table. Building on the previous example:

> *A person in the warehouse PARTY table has referenced CONTACT information and is also related to addressing information in PSTL_ADDR and GEOG_BNDRY. During normalization into the data mart, the resulting row in the PRODR_DIM contains the contact and regionalization information for the person.*

Dates are also normalized into dimension tables using the following method. The tables which contain date information are DT_DIM,MTH_DIM and TIME_DIM.

When a date value is normalized from the warehouse, the various attributes of that date are created or referenced. All required attributes of the date, month and time are created In the DT_DIM table, the primary key is the date itself, with the additional columns used to provide attributes of the date such as the context of the date with respect to calendar year, month and week. The same method applies to month and time. For example:

> *A transaction is migrated from warehouse to data mart. The transaction effective date is expressed as 1/1/2010 1:30:00 PM. The following information is contained in the DT_DIM table:*

- *DT_ID = 1/1/2010*
    - *DT_FULL_DESC = Friday, January 1, 2010*
    - *DAY_NUM_IN_WK = 6*
    - *DAY_NM = Fri*
    - *DAY_LONG_NM = Friday*
- *DAY_NUM_IN_CAL_MTH = 1*
    - *DAY_NUM_IN_CAL_YR = 1*
    - *Etc…*
- *MTH_ID = 1*

    - *CAL_MTH_NM = Jan*
    - *CAL_MTH_LONG_NM = January*
    - *CAL_MTH_NUM_IN_YR = 1*
    - *Etc…*
- *TIME_ID = 1:30:00 PM*
    - *HR_12_NM = One o'clock*
    - *HR_24_NM = 13 hrs*
    - *HR_12_SHORT_NM =1*
    - *HR_12_NUM = 1*
    - *HR_24_NUM = 13*
    - *AM_PM_CD = PM*
    - *Etc…*

Currently, columns in data mart tables that are named like *_DT_ID relate back to the DT_DIM tables, but this may change.

Chapter 4

Source-to-Staging ETL

Master Data Management Strategy

The task associated with mapping data from your source systems to the OII staging tables is one of the most critical steps in a successful implementation. Listed below are some best practices to use as guidance during this process.

- Data quality analysis or data profiling should be performed on the source data to ensure the data quality and completeness for business requirements.
- Logical data mapping describes the source elements, target elements (i.e. staging tables) and any transformations that need to occur between the source and target. Transformation rules should be stated explicitly for each target column and address any necessary data conversion. When converting a source date to a target date column, validate that the source date is in the expected format.
- NULL values retrieved from the source must be set to a default value before loading the staging tables. The Natural Key column in each staging table is the only exception to the “No NULLs” rule. A Nature Key must be supplied in order to support history (i.e. Type 1 or Type 2 SCDs).
- Reconcile the count of records at the end of the loading step against the records at the end of the transform step. This check ensures that all transformed records are accounted for and nothing is missed out during the loading phase.

Metadata

In OII, metadata refers to the data (e.g. currency conversion rates) and settings (e.g. the earned premium calculation method for an LOB) that is necessary to process the transactional and dimensional data loaded in staging. The meta-data tables include:

- Configuration (OII_SYS schema)
    - OII_SYS.SYS_CONFIG – This table contains several user-configurable settings for the OII system.
    - OII_SYS.SYS_CURR_CNV – OII loads this system table with the currency conversion data provided by the user in the OII_ST.CURR_CNV staging table.
    - OII_SYS.SYS_ERND_PREM_CONFIG – OII loads this system table with the Earned Premium method chosen by the user per LOB. There are four methods to chose from:
        - **D**’ – Daily method. This method is standardized and formula-based. This is the default method for OII.

        - ‘M’ – Monthly (1/24) method. This method is standardized and formula-based.
        - ‘O’ – Original Premium method. This method is standardized and formula-based.
        - ‘S’ – Seasonal method. This method uses metadata that is loaded in the OII_SYS.SYS_SEASONAL_CALC_CONFIG table. The FORMULA_CD column in the OII_SYS.SYS_ERND_PREM_CONFIG table will need to be loaded with one of the two provided standardized seasonal methods (‘1’ or ‘2’). The FORMULA_CD column should contain ‘%’ for all other Earned Premium Methods, (i.e. Daily, Monthly and Original).
    - OII_SYS. SYS_ERND_EXPO_CONFIG – This table is pre-loaded with metadata that is used for the different Earned Exposure calculation methods. These methods are standardized and formula-based. The TYP_CD options are ‘W’ for written exposure and ‘E’ for earned exposure.
    - OII_SYS.SYS_RNG – This table contains system-wide range lookup metadata.
    - OII_SYS.SYS_SEASONAL_CALC_CONFIG – This table is pre-loaded with seasonal-calculation metadata that is used when the Seasonal Earned Premium method is selected.
    - OII_SYS.SYS_XLT_NM – OII System Code Columns are pre-loaded in this table. Customer Code Columns that need corresponding descriptions are defined by the user in this table**.**
    - OII_SYS.SYS_XLT_VAL – Descriptions for OII System Codes are pre-loaded in this table. Translations and descriptions for source codes are configured by the user in this table.
- Warehouse (OII_WH schema)
    - OII_WH.PARTY_ROLE – This table is pre-loaded with the system-defined Party Roles. For more information, see Appendix A: Relationship Codes.
- Dimensions (OII_DM schema)
    - OII_DM.DT_DIM – This table is pre-loaded with dates from 1900 – 2099 plus dummy dates.
    - OII_DM.MTH_DIM – OII loads this table with month information generated from the DT_DIM table.
    - OII_DM.TIME_DIM – This table is pre-loaded with hour information.

All Warehouse entities and data mart dimensions have a default row loaded with a dummy ID of zero (0). These entries should remain in place once data is loaded.

REQUIRED TABLES

Listed below are the tables that are required in order for the dimension tables to be loaded correctly.

Required Staging Tables for a Policy, Earned Exposure or Earned Premium Transaction:

- OII_ST.BILL_ACCT (Billing Account)
- OII_ST.CLASS (Class)
- OII_ST.COVRG (Coverage)
- OII_ST.COVRG_STATUS (Coverage Status)
- OII_ST.DC (Distribution Channel)

- OII_ST.DED (Deductible)
- OII_ST.GRP_PRG (Group Program)
- OII_ST.INSUR_CO (Insurance Company)
- OII_ST.INSD (Insured)
- OII_ST.LMT (Limit)
- OII_ST.LOB (Line of Business)
- OII_ST.PLCY (Policy)
- OII_ST.PRODR (Producer)
- OII_ST.REINS_CNTRCT (Reinsurance Contract)
- OII_ST.REINS_CO (Reinsurance Company)
- OII_ST.UW (Underwriter)

Required Staging Tables for a Claims Transaction:

- OII_ST.BILL_ACCT (Billing Account)
- OII_ST.CAT (Catastrophe)
- OII_ST.CLASS (Class)
- OII_ST.CLMT (Claimant)
- OII_ST.CLM_ADJ (Claim Adjuster)
- OII_ST.CLM_EXAMR (Claim Examiner)
- OII_ST.CLM_MGR (Claim Manager)
- OII_ST.CLM (Claim)
- OII_ST.COVRG (Coverage)
- OII_ST.COVRG_STATUS (Coverage Status)
- OII_ST.DC (Distribution Channel)
- OII_ST.DED (Deductible)
- OII_ST.FEAT (Feature)
- OII_ST.FEAT_STATUS (Feature Status)
- OII_ST.GRP_PRG (Group Program)
- OII_ST.INSD (Insured)
- OII_ST.INSUR_CO (Insurance Company)
- OII_ST.LITG (Litigation)
- OII_ST.LMT (Limit)
- OII_ST.LOB (Line of Business)

- OII_ST.PLCY (Policy)
- OII_ST.PRODR (Producer)
- OII_ST.REINS_CNTRCT (Reinsurance Contract)
- OII_ST.REINS_CO (Reinsurance Company)
- OII_ST.SUPPLIER (Supplier)
- OII_ST.UW (Underwriter)

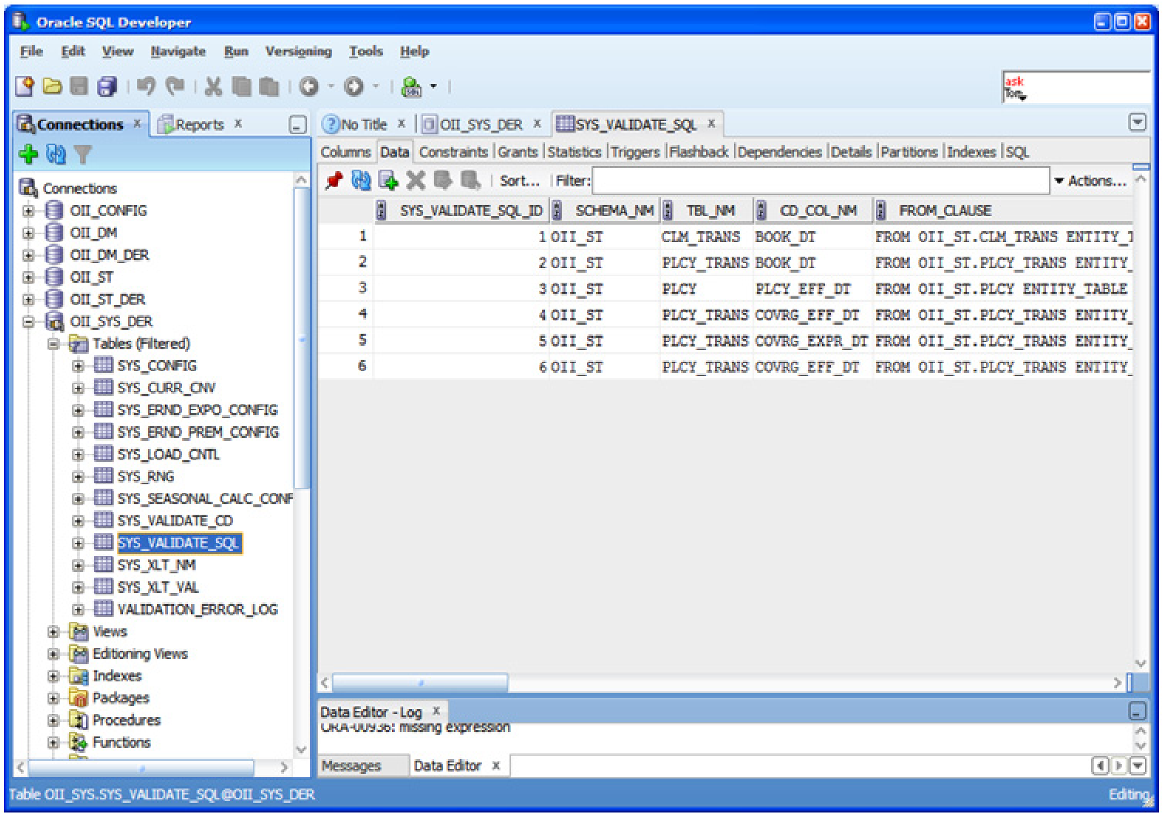
Data Validation

OII supports data validation in line with the ETL processes. OII supports two types of data validation: Code Lookup Validations and Date Validations. Data validation is performed in ODI prior to loading any data to the Warehouse or Data Mart schemas. This process is critical to ensure the data quality and completeness for business requirements. The data validations listed below are pre-loaded in OII.

- **Code Lookup Validations:**
  - LOB_CD
  - PLCY_TRANS_CD
  - COVRG_PART_CD
  - CLM_TRANS_CD
- **Date Validations:**
  - BOOK_DT must be greater than or equal to SRC_SYS_PROC_DT.
  - BOOK_DT must be greater than or equal to TRANS_EFF_DT.
  - PLCY_EFF_DT must be less than or equal to PLCY_EXPR_DT.
  - COVRG_EFF_DT must be less than or equal to PLCY_EXPR_DT.
  - COVRG_EXPR_DT must be equal to PLCY_EXPR_DT.
  - COVRG_EFF_DT must be greater than or equal to PLCY_EFF_DT.

There are several OII_SYS tables that are used to facilitate the Data Validation:

- **OII_SYS.SYS_VALIDATE_CD:**
    - This table defines all code-based validations.
    - The PROC_TYP_FLG column is used to enable/disable each validation. A value of '**V**' indicates a validation is enabled and a value of '**X**' indicates a validation is disabled.

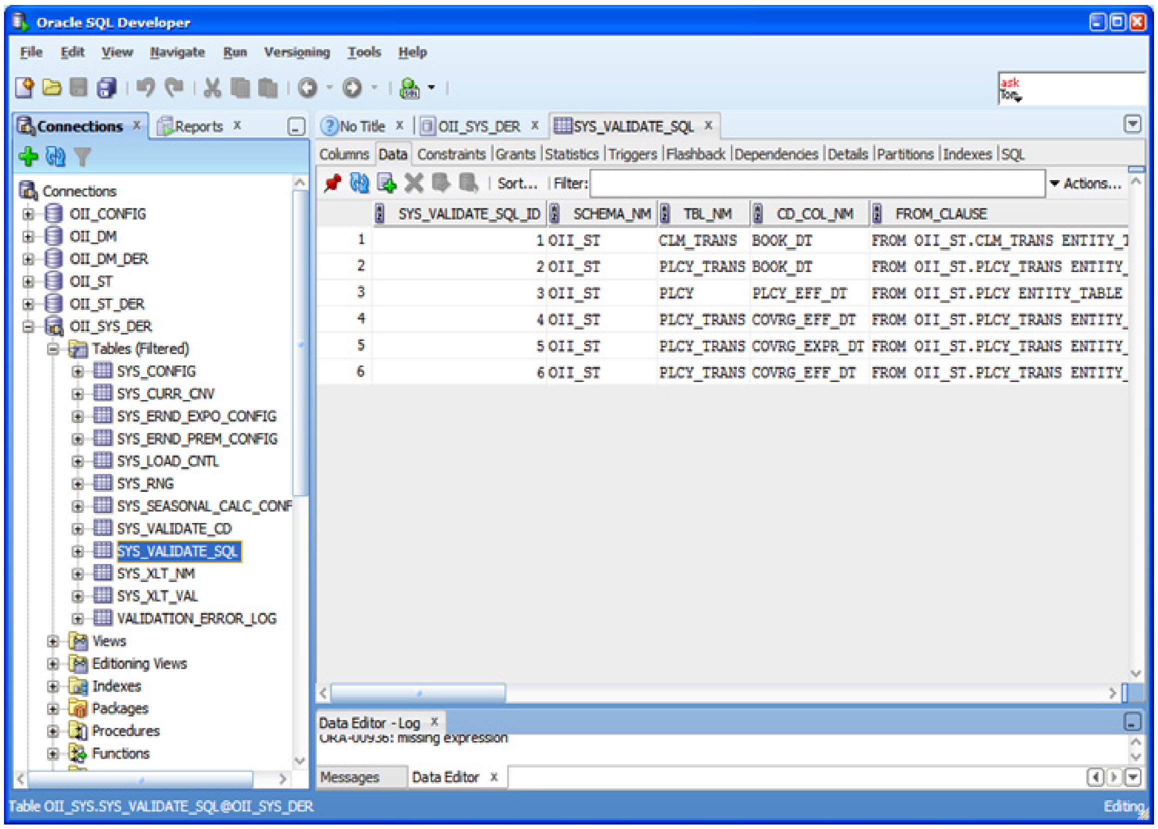


	SYS_VALIDATE_SQL_ID	SCHEMA_NM	TBL_NM	CD_COL_NM	FROM_CLAUSE
1	1	OII_ST	CLM_TRANS	BOOK_DT	FROM OII_ST.CLM_TRANS ENTITY_1
2	2	OII_ST	PLCY_TRANS	BOOK_DT	FROM OII_ST.PLCY_TRANS ENTITY_
3	3	OII_ST	PLCY	PLCY_EFF_DT	FROM OII_ST.PLCY ENTITY_TABLE
4	4	OII_ST	PLCY_TRANS	COVRG_EFF_DT	FROM OII_ST.PLCY_TRANS ENTITY_
5	5	OII_ST	PLCY_TRANS	COVRG_EXPR_DT	FROM OII_ST.PLCY_TRANS ENTITY_
6	6	OII_ST	PLCY_TRANS	COVRG_EFF_DT	FROM OII_ST.PLCY_TRANS ENTITY_

```
ORA-00936: missing expression
```

Figure 6: OII_SYS.SYS_VALIDATE_CD

- **OII_SYS.SYS_VALIDATE_SQL:**
    - This table defines all SQL based validations.
    - The PROC_TYP_FLG column is used to enable/disable each validation. A value of '**V**' indicates a validation is enabled and a value of '**X**' indicates a validation is disabled.



	SYS_VALIDATE_SQL_ID	SCHEMA_NM	TBL_NM	CD_COL_NM	FROM_CLAUSE
1	1	OII_ST	CLM_TRANS	BOOK_DT	FROM OII_ST.CLM_TRANS ENTITY_
2	2	OII_ST	PLCY_TRANS	BOOK_DT	FROM OII_ST.PLCY_TRANS ENTITY_
3	3	OII_ST	PLCY	PLCY_EFF_DT	FROM OII_ST.PLCY ENTITY_TABLE
4	4	OII_ST	PLCY_TRANS	COVRG_EFF_DT	FROM OII_ST.PLCY_TRANS ENTITY_
5	5	OII_ST	PLCY_TRANS	COVRG_EXPR_DT	FROM OII_ST.PLCY_TRANS ENTITY_
6	6	OII_ST	PLCY_TRANS	COVRG_EFF_DT	FROM OII_ST.PLCY_TRANS ENTITY_

```
ORA-00936: missing expression
```

Table OII_SYS.SYS_VALIDATE_SQL@OII_SYS_DER

Figure 7: OII_SYS.SYS_VALIDATE_SQL

The Data Validation can be executed from a pre-defined ODI Package or it can be executed directly from ODI:

- Data Validation is integrated into the ODI Package – **Warehouse Load Conditional.**

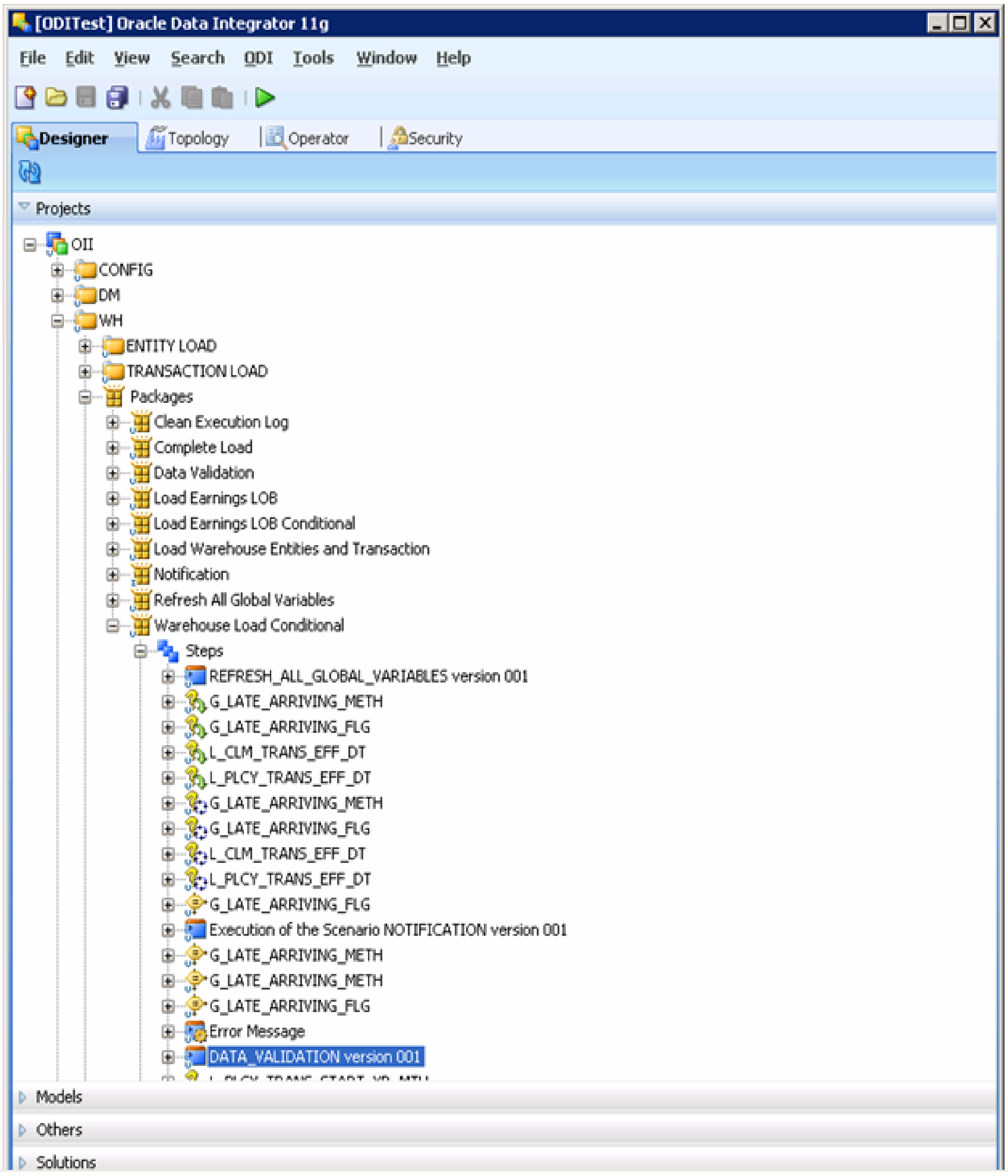


Figure 8: Data Validation

- To directly execute the Data Validation from ODI, locate the DATA_VALIDATION Scenario, right mouse click and select **Execute** from the menu.

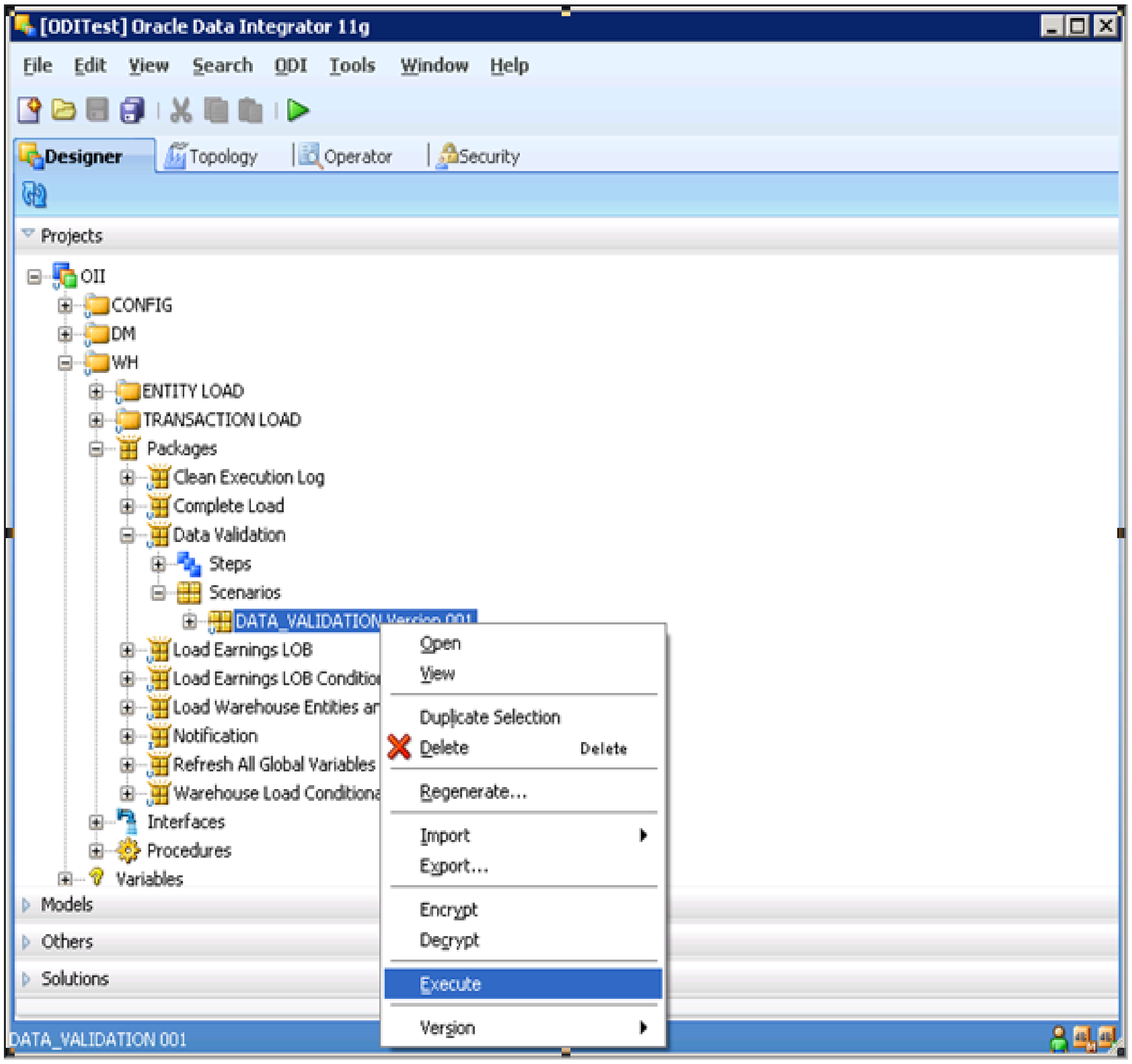
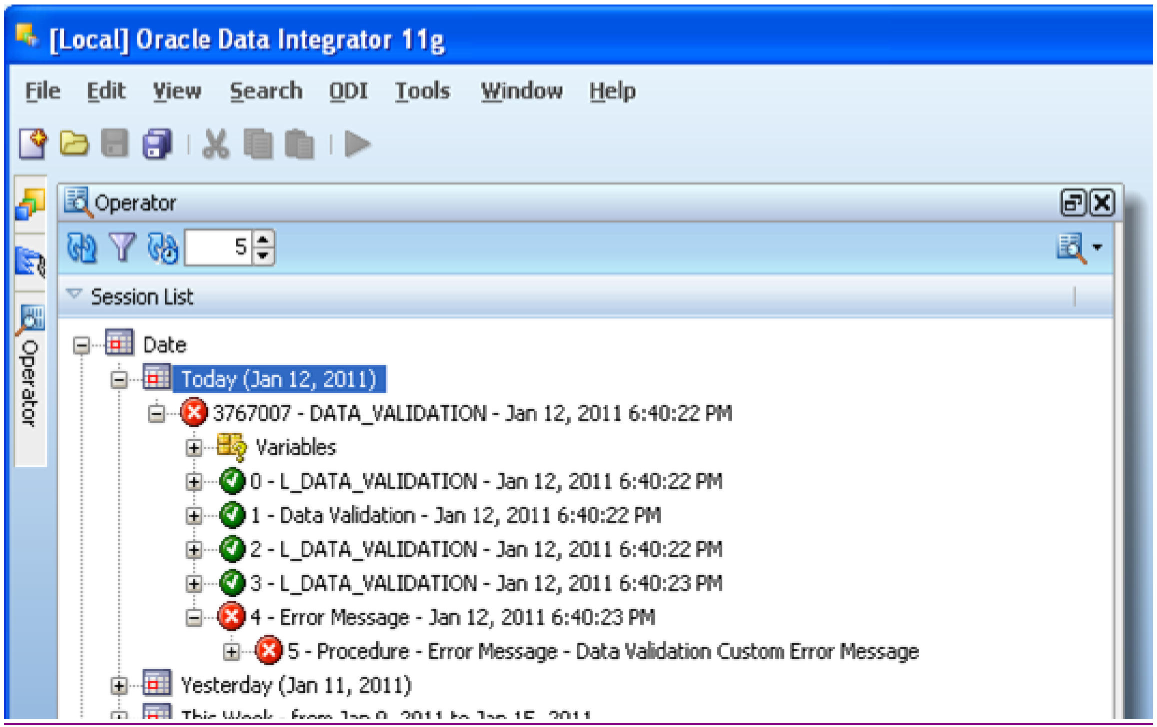


Figure 9: Execute Data Validation from ODI

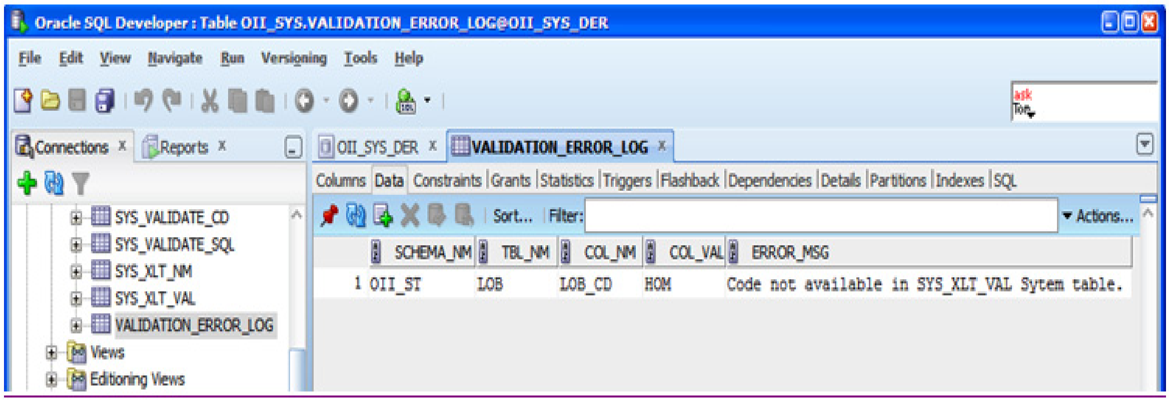
If there are data validation errors, the Operator console will show the error as shown in the figure below.


[Local] Oracle Data Integrator 11g
File Edit View Search ODI Tools Window Help
Operator
5
Session List

```
Date
  Today (Jan 12, 2011)
    3767007 - DATA_VALIDATION - Jan 12, 2011 6:40:22 PM
      Variables
      0 - L_DATA_VALIDATION - Jan 12, 2011 6:40:22 PM
      1 - Data Validation - Jan 12, 2011 6:40:22 PM
      2 - L_DATA_VALIDATION - Jan 12, 2011 6:40:22 PM
      3 - L_DATA_VALIDATION - Jan 12, 2011 6:40:23 PM
      4 - Error Message - Jan 12, 2011 6:40:23 PM
        5 - Procedure - Error Message - Data Validation Custom Error Message
  Yesterday (Jan 11, 2011)
```

Figure 10: Data Validation Errors

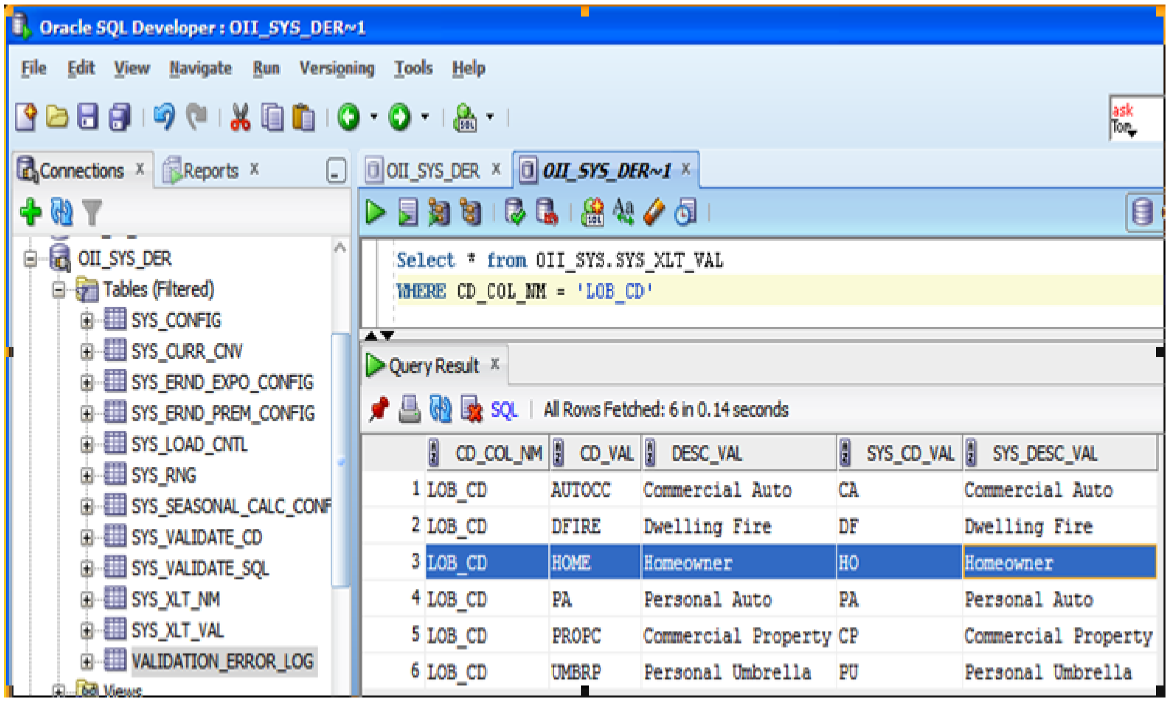
The table, OII_SYS.VALIDATION_ERROR_LOG, contains details of validations errors. Using this information, further analysis on the staging data can be performed.


Oracle SQL Developer : Table OII_SYS.VALIDATION_ERROR_LOG@OII_SYS_DER
File Edit View Navigate Run Versioning Tools Help
Connections
Reports
SYS_VALIDATE_CD
SYS_VALIDATE_SQL
SYS_XLT_NM
SYS_XLT_VAL
VALIDATION_ERROR_LOG
Views
Editioning Views
OII_SYS_DER
VALIDATION_ERROR_LOG
Columns Data Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL
Sort... Filter:
Actions...

	SCHEMA_NM	TBL_NM	COL_NM	COL_VAL	ERROR_MSG
1	OII_ST	LOB	LOB_CD	HOM	Code not available in SYS_XLT_VAL Sytem table.

Figure 11: OII_SYS.VALIDATION_ERROR_LOG

With this example the validation error indicates a failure with translating an LOB_CD value of 'HOM'. With this type of error, you will need to verify that the staging data contains a code value that is represented in the OII_SYS.SYS_XLT_VAL table.



```
Select * from OII_SYS.SYS_XLT_VAL
WHERE CD_COL_NM = 'LOB_CD'
```

	CD_COL_NM	CD_VAL	DESC_VAL	SYS_CD_VAL	SYS_DESC_VAL
1	LOB_CD	AUTOCC	Commercial Auto	CA	Commercial Auto
2	LOB_CD	DFIRE	Dwelling Fire	DF	Dwelling Fire
3	LOB_CD	HOME	Homeowner	HO	Homeowner
4	LOB_CD	PA	Personal Auto	PA	Personal Auto
5	LOB_CD	PROPC	Commercial Property	CP	Commercial Property
6	LOB_CD	UMBRP	Personal Umbrella	PU	Personal Umbrella

Figure 12: Failure in Translating the LOB_CD Value

With this example, the validation process has identified at least one row of data in the OII_ST.LOB table with an invalid LOB_CD value of 'HOM'. This value should be 'HOME' which translates to the 'Homeowner' line of business with a SYS_CD_VAL of 'HO'. This error must be corrected in staging before the data is allowed to be loaded into the warehouse.

Data Relationships

Unique Keys

In the staging tables, there are specific transactions for Policies, Claims, Earned Premiums and Earned Exposures. For OII_ST.PLCY and OII_ST.CLM tables natural keys are used to uniquely identify a policy and a claim. Natural keys are discussed in detail in *Chapter 5: Program Flow*. Unique keys are used to connect Policy, Claim, Earned Premium and Earned Exposure transactions within the warehouse tables.

- OII_WH.PLCY_TRANS
  - Includes PLCY_TRANS_ID to uniquely identify a specific policy transaction.
- OII_WH.CLM_TRANS
  - Includes CLM_TRANS_ID to uniquely identify a specific claim transaction.
  - Includes PLCY_ID to establish a relationship between a specific claim transaction and its matching policy.
- OII_WH.ERND_PREM_TRANS
  - Includes ERND_PREM_TRANS_ID as well as PLCY_TRANS_ID to create the relationship between a specific earned premium transaction and its matching policy transaction.
- OII_WH.ERND_EXPO_TRANS
  - Includes ERND_EXPO_TRANS_ID as well as PLCY_TRANS_ID to create the relationship between a specific earned exposure transaction and its matching policy transaction.

Common/Shared Keys

In the staging environment, there are four possible types of transactions that can be loaded from the customer data: Policy Transactions, Claim Transactions, Earned Premium Transactions and Earned Exposure Transactions. The staging table that is critical in making a connection between all of these transactions is the OII_ST.PLCY. For each of the previously mentioned transaction types, a unique policy should be associated with that transaction.

When the data from the table OII_ST.PLCY is loaded to the warehouse, A POLICY_ID and a POLICY_TRANS_ID is generated in the table OII_WH.PLCY_TRANS. The POLICY_ID is shared between the tables OII_WH.PLCY_TRANS and OII_WH.CLMS_TRANS, creating a connection between a specific claim and its matching policy. The POLICY_TRANS_ID is shared between the tables OII_WH.PLCY_TRANS and OII_WH.ERND_PREMIUM as well as OII_WH.ERND_EXPOSURE, creating a connection between a specific earned premium and earned exposure transaction and its matching policy.

Data Loading Considerations

In coding the Source-to-Staging ETL, it is important to know how OII expects the transactions to relate to one another and the policy as a whole. One of these considerations is how to distribute premium in a Personal Auto Policy.

Personal Auto Premiums

In the Personal Auto LOB, each transaction has one associated driver. If that transaction includes any Written Premium, it will be assigned to the vehicle and/or driver associated with that transaction.

In order to perform premium or loss analysis in Personal Auto at the Vehicle and/or Driver level (i.e. a sub-policy level), the premium should be apportioned across multiple transactions. The premium assigned to that transaction will be associated with the particular vehicle and/or driver for that transaction when the data is loaded into the Staging tables. Transactions loaded with zero premium will not assign any premium to the vehicle and/or driver associated with that transaction, so sub-policy level analysis will not be available.

Chapter 5

Data Loading

After the Source-to-Staging ETL has loaded the transactional data, the OII system is ready to be configured to handle the customer data, loaded and prepared for presentation to the end user. The Data Loading chapter introduces the OII System Codes that control processing of transactions, load configuration options, currency configuration, load execution and OBIEE Repository linkage.

System Codes

System Codes are functional codes in OII that tell the system how to route and process transactions. System Codes are defined in the OII_SYS.SYS_XLT_NM table. System code translations and description are contained in the OII_SYS.SYS_XLT_VAL table. Four categories are covered:

- **Claim Transaction Code (CLM_TRANS_CD)** – This system code represents what type of loss activity occurred in the claim transaction. The functional codes supplied with OII cannot be changed. They are described below. Additional Claim Transaction Codes can be added by the user to group related transaction types, but these transactions will never be included in the monthly snapshot data marts.
- **Coverage Part Code (COVRG_PART_CD)** – This system code represents an LOB within a package policy. By default, there is a single code defined for each of the six pre-defined LOBs. Additional Coverage Part Codes can be added by the user.
- **Line Of Business Code (LOB_CD)** – This system code identifies the LOB classification. By default, there is a single code defined for each of the six pre-defined LOBs. Additional Line of Business Codes can be set by the user in Warehouse Palette. When an LOB is published, its code is added to the OII_SYS.SYS_XLT_VAL table.
- **Policy Transaction Code (PLCY_TRANS_CD)** – This system code identifies the policy transaction. The functional codes supplied with OII cannot be changed. They are described below. Additional Policy Transaction Codes can be added by the user to group related transaction types, but these transactions will never be included in the monthly snapshot data marts.

Source Code Translation and Description

In addition to providing descriptions for the pre-defined OII System Codes, the OII_SYS.SYS_XLT_VAL table translates client source codes present in the staging data into OII System Codes. No additional configuration is needed in the OII_SYS.SYS_XLT_NM table if a source code is being translated to one of the four System Codes defined above.

If descriptions are needed for source codes outside of the four categories above, an entry is needed in the OII_SYS.SYS_XLT_NM table to define the staging table and column that are being translated. The target code and description columns are also defined here. Source codes are translated and described in the OII_SYS.SYS_XLT_VAL table exactly as it is done for the System Codes.

Claim Transaction Codes

Claim Transaction Codes are required to process claim transactions correctly. These codes also allow for the proper display of claim transaction types in OBIEE. There are currently 19 Claim Transaction Codes.

These consist of three basic categories of transactions:

- **Reserve** – The initial reserve set up on a claim, including the initial reserve on a reopened claim.
- **Change** – Any change in reserves on a claim, including the final reserve transaction (eliminating the loss reserve) in a closed or reclosed claim.
- **Paid/Recovered** – Any paid amounts on a claim, including any financial adjustments or modifications to paid losses. This also includes any recoveries made for Salvage, Subrogation or Deductible.

For each category, there are six claim transaction types:

- **Loss** – Amounts reserved, paid or recovered in relation to the settlement of the claim.
- **Allocated Adjustment Expense (ALAE)** – Amounts reserved, paid or recovered in relation to adjustment expenses specifically allotted to a claim.
- **Unallocated Adjustment Expense (ULAE)** – Amounts reserved, paid or recovered in relation to adjustment expenses that are not specifically allocated to a claim. These amounts are not usually used on a claim transaction basis, but rather on a financial accounting basis.
- **Salvage** – Amounts reserved, paid or recovered in relation to property collected as salvage in the settlement of a claim.
- **Subrogation** – Amounts reserved, paid or recovered in relation to reimbursement from a third party responsible for damage or liability in the settlement of a claim.
- **Deductible** – Amounts reserved, paid or recovered in relation to reimbursement of a deductible in the settlement of a claim.

In addition to these eighteen codes, there is a special code that does not fit into the categories above.

- **Non Financial** – The Non Financial code allows for a transaction to be entered without affecting the reserve amount. When loaded, the transaction is available in the transaction data marts, but it will never be included in the monthly snapshot data marts.

The 19 functional Claim Transaction Codes are grouped in the three categories (Reserve, Change and Paid\Recovered) along with the special Non Financial code below:

Claim Transaction Type	Code	Description
Loss Reserve	LR	Initial or reopened Expense Reserve
Allocated Expense Reserve	AR	Initial or reopened ALAE Reserve
Unallocated Expense Reserve	UR	Initial or reopened ULAE Reserve
Salvage Reserve	SR	Initial or reopened Salvage Reserve
Subrogation Reserve	RS	Initial or reopened Subrogation Reserve
Loss Deductible Recovery Reserve	DR	Initial or reopened Deductible Reserve
Loss Change	LC	Change in Loss Reserve
Allocated Expense Change	AC	Change in ALAE Reserve
Unallocated Expense Change	UC	Change in ULAE Reserve
Salvage Change	SC	Change in Salvage Reserve
Subrogation Change	RC	Change in Subrogation Reserve
Loss Deductible Change	DC	Change in Deductible Reserve
Paid Loss	PL	Paid Loss Amount
Allocated Expenses Paid	AE	Paid ALAE Amount
Unallocated Expenses Paid	UE	Paid ULAE Amount
Salvage Recovered	SL	Recovered Salvage Amount
Subrogation Recovered	SB	Recovered Subrogation Amount
Loss Deductible Recovered	DE	Recovered Deductible Amount
Non Financial	NF	Non-Financial Amount (may be zero or non-zero dollar amount)

Policy Transaction Codes

Policy Transaction Codes are required to process premiums and policy transactions correctly. Like the claim codes, the policy codes allow for the proper display of policy transaction types in OBIEE. There are currently 9 Policy Transaction Codes.

Policy Transaction Type	Code
Additional Premium Audit	AP
Cancelled Flat	CF
Cancellation	CN
Cancelled Other	CO
New Business	NB
Non-Premium Endorsement	NP
Renewal Business	RB
Reinstatement Business	RE
Return Premium Audit	RP

Load Configuration

The load parameters listed below are found in the OII_SYS.SYS_CONFIG table. These values should be set prior to executing the data load in ODI.

Analysis Options

- **Code Names:** ANALYZE_WH, ANALYZE_DM
- **Description:** These parameters determine if the warehouse and/or data mart need to be analyzed as part of internal loading (ETL) process. A “Y” value (default) will invoke the analysis process (dbms_stats) as part of ETL. Oracle recommends running stats on a regularly scheduled basis. The analysis option is provided as a way to automate this feature. The analysis options are used in the packages below:
- Load Warehouse Entities and Transactions
- Load Monthly Snapshot Facts
- Load Data Mart Dimensions and Transactional Facts

Currencies

- **Code Names:** LOCAL_CURR, GLOBAL_CURR_1, GLOBAL_CURR_2, and GLOBAL_CURR_3
- **Description:** These parameters define the Local Reporting Currency (i.e. the default currency; LOCAL_CURR) and the optional Global Reporting Currencies. Up to three Global Reporting Currencies can be defined (i.e. GLOBAL_CURR_1, GLOBAL_CURR_2 and GLOBAL_CURR_3). Transaction amounts are input to the OII system in as Document Currency. The Local Reporting Currency and the Global Reporting Currencies can be utilized along with the currency conversion rate data (see *Currency Configuration* on page 51) to convert the Document Currency to the corresponding Reporting Currency equivalents.
- **Example Value:** “USD”

Default Begin Date

- **Code Name:** DFLT_BEGIN_DT
- **Description:** This date represents the lowest date value that the system can support. This value will also serve as a default date value if an input date is NULL. The date format “MM-DD-YYYY” is recommended. The format of the value needs to match the DFLT_DT_FORMAT value.
- **Example Value:** “01-01-0001”

Default Date Format

- **Code Name:** DFLT_DT_FORMAT
- **Description:** Default date format for the system. The date format “MM-DD-YYYY” is recommended.

Default Date/Time Format

- **Code Name:** DFLT_DT_TIME_FORMAT
- **Description:** Default date time format. The format “MM-DD-YYYY HH24:MI:SS” is recommended.

Default End Date

- **Code Name:** DFLT_END_DT
- **Description:** This is the default end date for the system. This date will be used to mark the end date of active record in case of slowly changing record. The format of the value needs to match the DFLT_DT_FORMAT value.
- **Example Value:** “12-31-9999”

Default Numeric Value

- **Code Name:** NUM_DFLT_VAL
- **Description:** Default number value if input number is null. This has to be a single digit number.
- **Example Value:** “0”

Default String Value

- **Code Name:** STR_DFLT_VAL
- **Description:** Default string value if input string is NULL. This has to be a single character and cannot be a single quote character.

Incurred Loss Calculation Method

- **Code Name:** INCRD_CALC_METH is the Incurred Loss Calculation Method, value 1 – 5, default is 1
- **Description:** The following options are available.
  - Method 1: Loss Reserve + Paid Loss
  - Method 2: Loss Reserve + Paid Loss + Salvage + Salvage Reserve + Subrogation + Subrogation Reserve

- Method 3: Loss Reserve + Paid Loss + Salvage + Salvage Reserve + Subrogation + Subrogation Reserve + Deductible + Deductible Reserve
- Method 4: Loss Reserve + Paid Loss - Salvage + Salvage Reserve - Subrogation + Subrogation Reserve + Deductible + Deductible Reserve
- Method 5: Loss Reserve + Paid Loss + Salvage + Subrogation + Deductible

Incurred Loss with Expense Calculation Method

- **Code Name:** INCRD_WEXPNS_CALC_METH is the Incurred Loss with Expense Calculation Method, value 1 – 2, default is 1
- **Description:** The following options are available.
  - 1: Incurred Loss + Allocated Expense + Allocated Expense Reserve + Unallocated Expense + Unallocated Expense Reserve
  - 2: Incurred Loss + Allocated Expense + Allocated Expense Reserve

Late Arriving Methodology

- **Code Name:** LATE_ARRIVING_METH
- **Description:** This parameter determines the loading method to be used. The loading methods listed below are supported.
  - 1: Ignore Late Arriving Transaction, just load the regular transactions
  - 2: Hold the load and wait for user's decision whether to continue if there is late arriving data. Otherwise, continue as normal
  - 3: Load Late Arriving Transactions, but do not delete/reload Earned Premium/Earned Exposure/MTH_SNAP_SHOT for the affected period
  - 4: Load Late Arriving Transactions and automatically reload Earned Premium/Earned Exposure/MTH_SNAP_SHOT for the affected period

Log Name/Path

- **Code Name:** LOG_FILE_NM_PATH
- **Description:** Path and File name of the OS log file to be generated in case of late arriving notification. File will be created on the machine hosting the agent, hence path should be for the machine hosting agent responsible for invoke the processes.

Currency Configuration

OII supports loading transactional amounts in an unlimited number of Document Currencies and reporting in up to four Reporting Currencies.

Transactions are always loaded in denominations of the Document Currency that is defined at the transactional level. For claim transactions, a code for the Document Currency is entered in the DOC_CURR_CD column of the OII_ST.CLM_TRANS (Claim Transaction) table. If a policy transaction, earned exposure transaction or earned premium transaction is loaded, a code for the Document Currency must be entered in the DOC_CURR_CD column of the OII_ST.PLCY_TRANS (Policy Transaction) table.

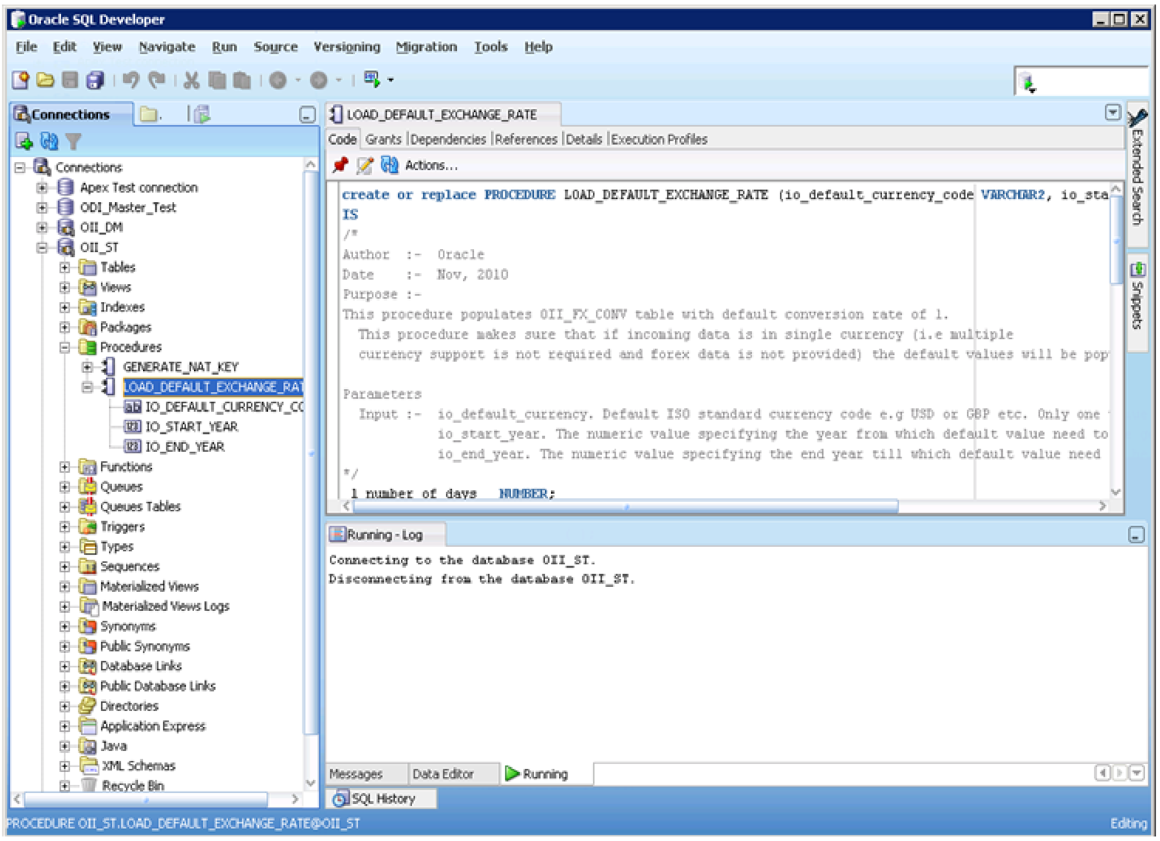
OII can support up to four Reporting Currencies: one Local Reporting Currency (LOCAL_CURR) and up to three additional Global Reporting Currencies (GLOBAL_CURR_1, GLOBAL_CURR_2 and GLOBAL_CURR_3), as defined above under “Currencies.”

A daily exchange rate must be entered for the Document Currency and each Reporting Currency that is defined. This exchange rate data is transferred to the OII_SYS.CURR_CNV (Currency Conversion) table whenever the warehouse tables are loaded. See *Load Execution* on page 55 for definitions of which scenarios involve loading warehouse tables.

Default Currency Conversion Rates

Before loading the warehouse tables for the first time, the OII_ST.LOAD_DEFAULT_EXCHANGE_RATE procedure should be executed for any currency that is both a Document Currency and a Reporting Currency. This will populate the necessary rows in the OII_ST.CURR_CNV table so that conversion rates will not need to be loaded to convert a currency to itself. Executing this procedure sets the self-to-self conversion rate at 1 for the specified time span.

Connect to the OII_ST schema in Oracle SQL Developer. Under Procedures in OII_ST, find the LOAD_DEFAULT_EXCHANGE_RATE procedure, as shown in Figure 13 on page 52.



```
create or replace PROCEDURE LOAD_DEFAULT_EXCHANGE_RATE (io_default_currency_code VARCHAR2, io_sta
IS
/*
Author  :-  Oracle
Date    :-  Nov, 2010
Purpose :-
This procedure populates OII_FX_CONV table with default conversion rate of 1.
  This procedure makes sure that if incoming data is in single currency (i.e multiple
  currency support is not required and forex data is not provided) the default values will be pop

Parameters
  Input :-  io_default_currency. Default ISO standard currency code e.g USD or GBP etc. Only one
            io_start_year. The numeric value specifying the year from which default value need to
            io_end_year. The numeric value specifying the end year till which default value need
*/
 l_number_of_days   NUMBER;
```

```
Connecting to the database OII_ST.
Disconnecting from the database OII_ST.
```

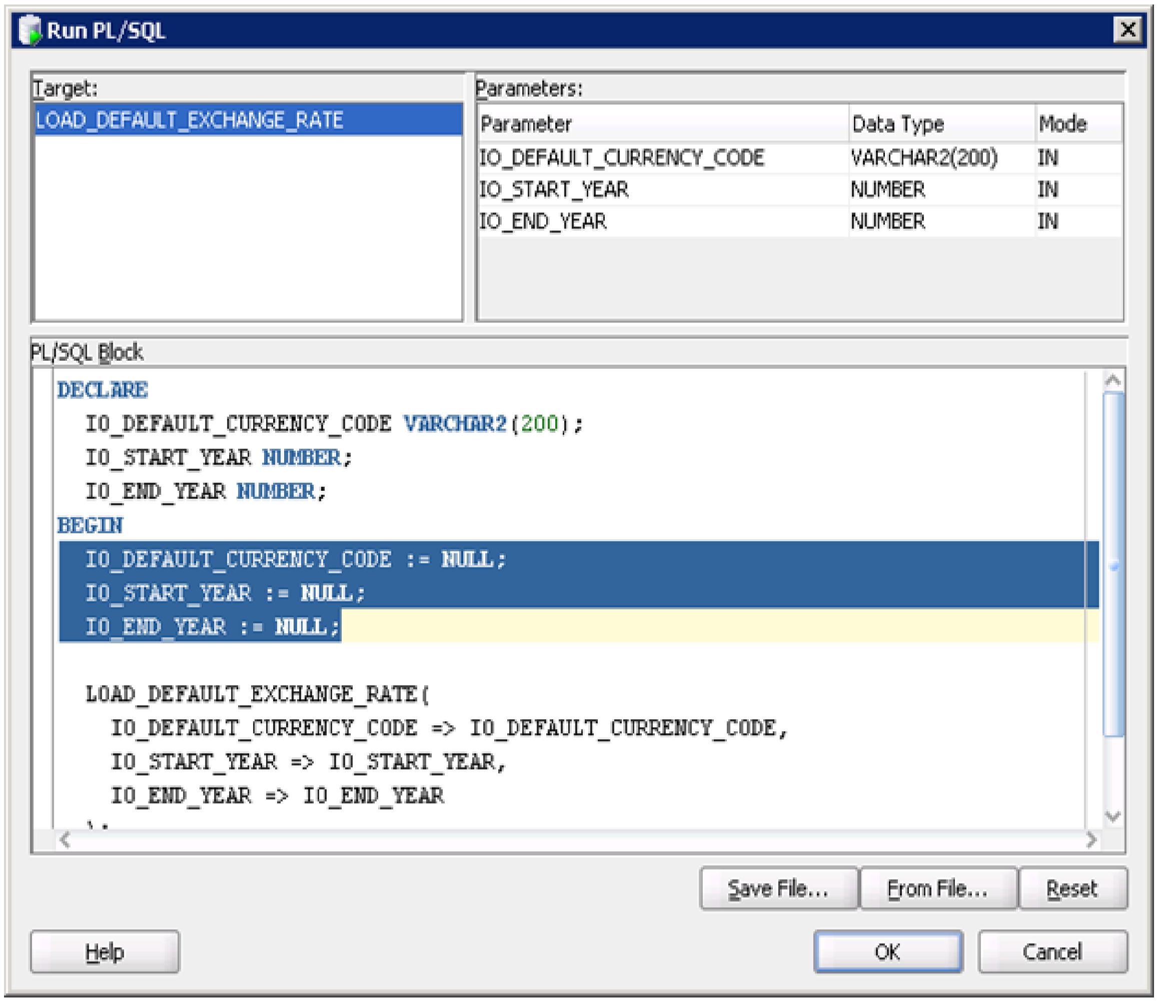
Figure 13: The LOAD_DEFAULT_EXCHANGE_RATE Procedure

The inputs are:

- **IO_DEFAULT_CURRENCY_CODE** – This is the Reporting Currency Code (e.g. 'USD' or 'GBP') as defined in the OII_SYS.SYS_CONFIG table.
- **IO_START_YEAR** – This numeric value specifies the year from which the default conversion rate needs to be generated.
- **IO_END_YEAR** – This numeric value specifies the year to which the default conversion rate needs to be generated.

When specifying a start year and end year, select years that encompass the entire historical and future reporting period. This will populate all necessary conversion rate data for the currency so that this procedure only needs to be run once.

Right click on the procedure, LOAD_DEFAULT_EXCHANGE_RATE, and select **Run**. The **Run PL/SQL** dialog box will appear.



LOAD_DEFAULT_EXCHANGE_RATE

Parameter	Data Type	Mode
IO_DEFAULT_CURRENCY_CODE	VARCHAR2(200)	IN
IO_START_YEAR	NUMBER	IN
IO_END_YEAR	NUMBER	IN

```
DECLARE
  IO_DEFAULT_CURRENCY_CODE VARCHAR2(200);
  IO_START_YEAR NUMBER;
  IO_END_YEAR NUMBER;
BEGIN
  IO_DEFAULT_CURRENCY_CODE := NULL;
  IO_START_YEAR := NULL;
  IO_END_YEAR := NULL;

  LOAD_DEFAULT_EXCHANGE_RATE(
    IO_DEFAULT_CURRENCY_CODE => IO_DEFAULT_CURRENCY_CODE,
    IO_START_YEAR => IO_START_YEAR,
    IO_END_YEAR => IO_END_YEAR
```

Figure 14: Run PL/SQL Dialog Box

Replace the NULL values with the desired values, as shown in the highlighted inputs above. Click the “OK” button to run. Note that the IO_DEFAULT_CURRENCY_CODE input value will need single quotes (e.g. ‘USD’) because it is a VARCHAR2 data type.

MULTIPLE CURRENCIES AND CONVERSION RATES

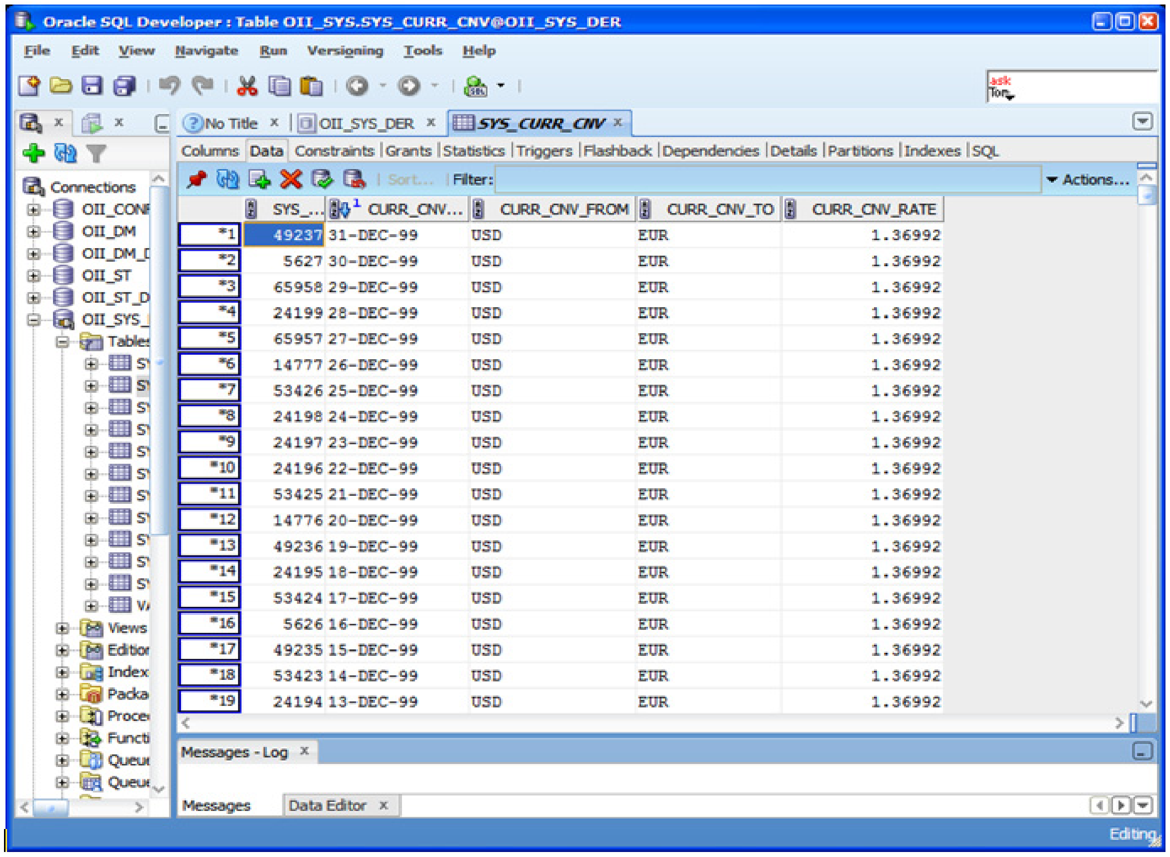
If a single currency is used for the Document Currency and the Reporting Currencies, loading the default currency conversion rate is sufficient. If transactional amounts will be reported in currencies other than the currency used for loading, daily conversion rate data will need to be provided for each Document-Currency-to-Reporting-Currency pair.

If needed, daily currency conversion rate data is loaded in the OII_ST.SYS_CURR_CNV (Currency Conversion) table. The columns in the OII_ST.SYS_CURR_CNV table are listed below:

- **CURR_CNV_DT** – This is the date for which the exchange rate applies.

- **CURR_CNV_FROM_CD** – This is the code for the Document Currency (e.g. USD or GBP). This code will be defined in the OII_SYS.SYS_CONFIG table if it is also a Reporting Currency. Otherwise, Document Currency codes will only appear in the DOC_CURR_CD column of the appropriate transaction table and here, in the OII_ST.SYS_CURR_CNV table.
- **CURR_CNV_TO_CD** – This is the code for the appropriate Reporting Currency as defined in the OII_SYS.SYS_CONFIG table (e.g. EUR or INR).
- **CURR_CNV_RATE** – This numeric value represents the coefficient multiplied by the Document Currency (CURR_CNV_FROM_CD) to produce an equal value in terms of the Reporting Currency (CURR_CNV_TO_CD).

Loading of the OII_ST.SYS_CURR_CNV table with daily currency conversion rates should be incorporated into or alongside the Source-to-Staging ETL process that loads transactional data.


Oracle SQL Developer : Table OII_SYS.SYS_CURR_CNV@OII_SYS_DER
File Edit View Navigate Run Versioning Tools Help
No Title
OII_SYS_DER
SYS_CURR_CNV
Columns Data Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL
Sort...
Filter:
Actions...
Connections
OII_CONF
OII_DM
OII_DM_D
OII_ST
OII_ST_D
OII_SYS_
Tables
Views
Editior
Index
Packa
Proce
Functi
Queue
Queue
Messages - Log
Messages
Data Editor
Editing

	SYS_...	CURR_CNV...	CURR_CNV_FROM	CURR_CNV_TO	CURR_CNV_RATE
*1	49237	31-DEC-99	USD	EUR	1.36992
*2	5627	30-DEC-99	USD	EUR	1.36992
*3	65958	29-DEC-99	USD	EUR	1.36992
*4	24199	28-DEC-99	USD	EUR	1.36992
*5	65957	27-DEC-99	USD	EUR	1.36992
*6	14777	26-DEC-99	USD	EUR	1.36992
*7	53426	25-DEC-99	USD	EUR	1.36992
*8	24198	24-DEC-99	USD	EUR	1.36992
*9	24197	23-DEC-99	USD	EUR	1.36992
*10	24196	22-DEC-99	USD	EUR	1.36992
*11	53425	21-DEC-99	USD	EUR	1.36992
*12	14776	20-DEC-99	USD	EUR	1.36992
*13	49236	19-DEC-99	USD	EUR	1.36992
*14	24195	18-DEC-99	USD	EUR	1.36992
*15	53424	17-DEC-99	USD	EUR	1.36992
*16	5626	16-DEC-99	USD	EUR	1.36992
*17	49235	15-DEC-99	USD	EUR	1.36992
*18	53423	14-DEC-99	USD	EUR	1.36992
*19	24194	13-DEC-99	USD	EUR	1.36992

Figure 15: OII_SYS.SYS_CURR_CNV Table

Load Execution

Several execution options are available to load OII. The scenarios listed below can be executed to load the appropriate layers of the OII data model.

Complete Load

The COMPLETE_LOAD scenario loads the Warehouse, Dimensions, Transactional Fact tables and the Monthly Snapshot Fact tables.

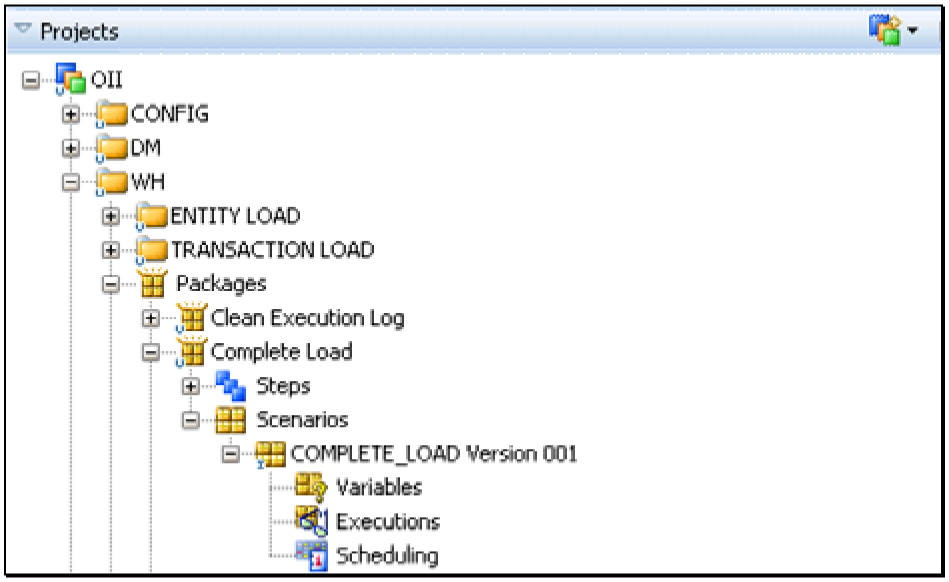
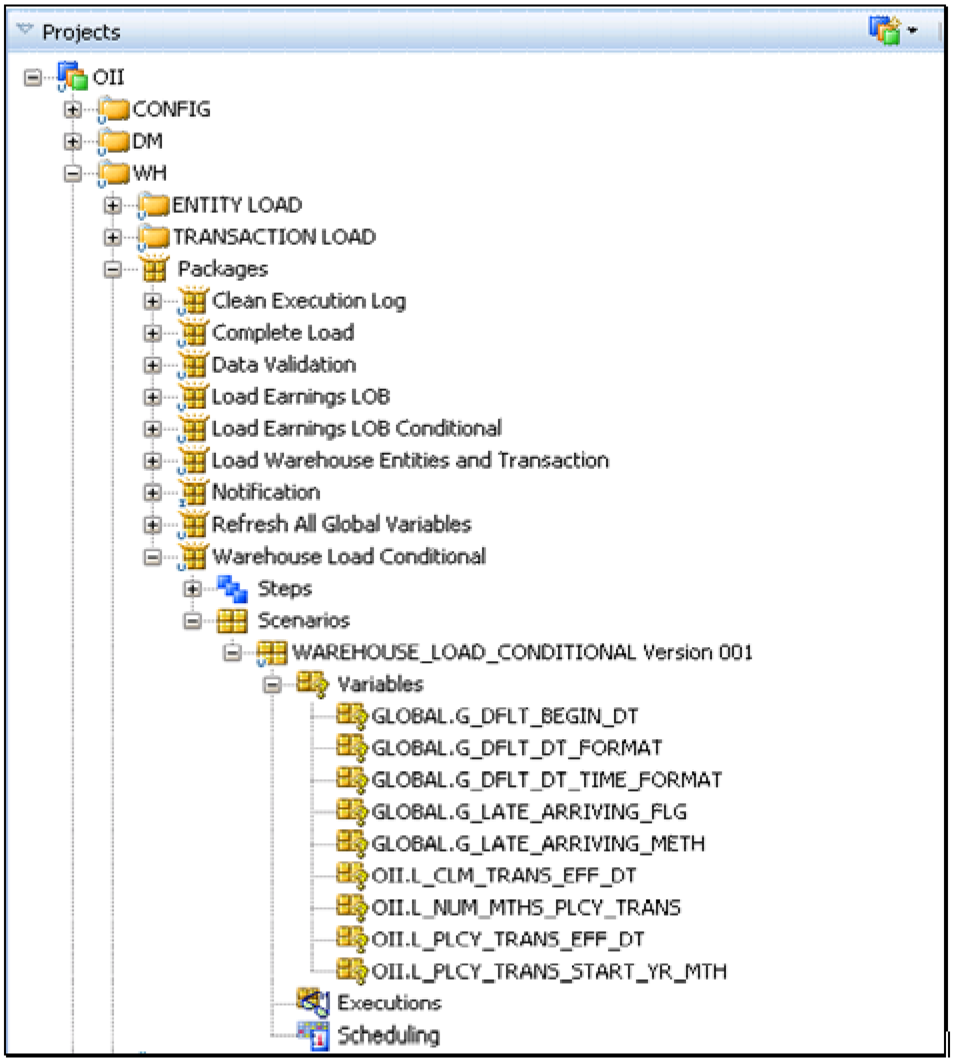


Figure 16: Complete Load

WAREHOUSE LOAD

The WAREHOUSE_LOAD_CONDITIONAL scenario loads the Warehouse tables. This scenario is useful in debugging the data produced from the Source-to-Staging ETL..


Projects

- OII
  - CONFIG
  - DM
  - WH
    - ENTITY LOAD
    - TRANSACTION LOAD
    - Packages
      - Clean Execution Log
      - Complete Load
      - Data Validation
      - Load Earnings LOB
      - Load Earnings LOB Conditional
      - Load Warehouse Entities and Transaction
      - Notification
      - Refresh All Global Variables
      - Warehouse Load Conditional
        - Steps
        - Scenarios
          - WAREHOUSE_LOAD_CONDITIONAL Version 001
            - Variables
              - GLOBAL.G_DFLT_BEGIN_DT
              - GLOBAL.G_DFLT_DT_FORMAT
              - GLOBAL.G_DFLT_DT_TIME_FORMAT
              - GLOBAL.G_LATE_ARRIVING_FLG
              - GLOBAL.G_LATE_ARRIVING_METH
              - OII.L_CLM_TRANS_EFF_DT
              - OII.L_NUM_MTHS_PLCY_TRANS
              - OII.L_PLCY_TRANS_EFF_DT
              - OII.L_PLCY_TRANS_START_YR_MTH
            - Executions
            - Scheduling

Figure 17: Warehouse Load

DATA MART LOAD

The LOAD_DATA_MART_DIMENSIONS_AND_FACTS scenario loads the Dimensions, Transactional Fact tables and the Monthly Snapshot Fact tables. This scenario can be used to complete the loading after the Warehouse Load has been executed.

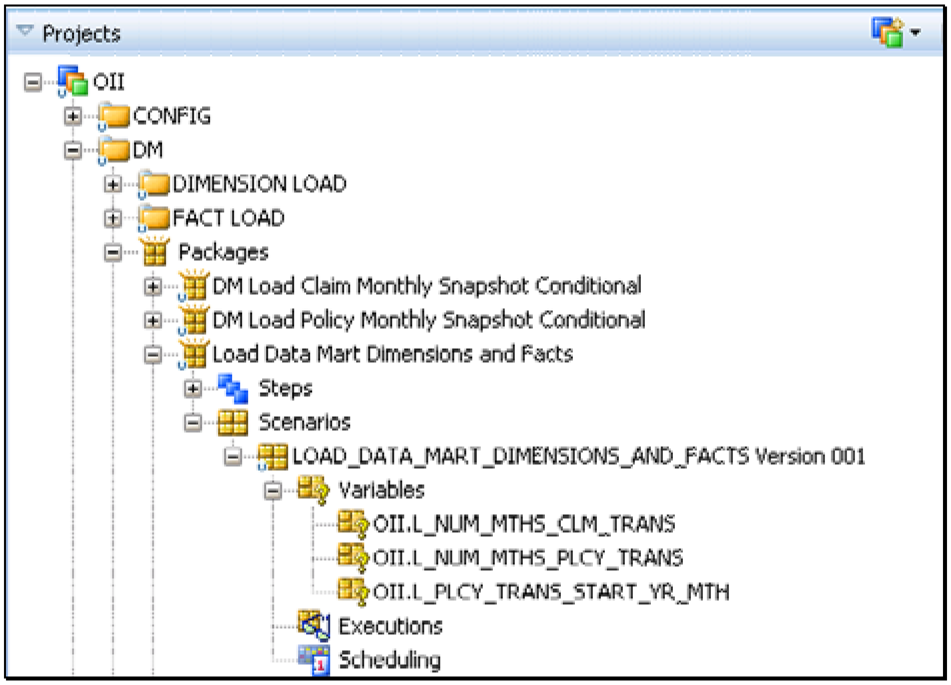

Projects
OII
CONFIG
DM
DIMENSION LOAD
FACT LOAD
Packages
DM Load Claim Monthly Snapshot Conditional
DM Load Policy Monthly Snapshot Conditional
Load Data Mart Dimensions and Facts
Steps
Scenarios
LOAD_DATA_MART_DIMENSIONS_AND_FACTS Version 001
Variables
OII.L_NUM_MTHS_CLM_TRANS
OII.L_NUM_MTHS_PLCY_TRANS
OII.L_PLCY_TRANS_START_YR_MTH
Executions
Scheduling

Figure 18: Data Mart Load

Dimensions and Transactional Facts Load

The LOAD_DATA_MART_DIMENSIONS_AND_FACTS_DAILY scenario loads the Dimensions and the Transactional Fact tables. This scenario can be used after the Warehouse Load_ whenever the Monthly Snapshot Fact tables are not ready to be loaded. This may be the case if routine loads are performed multiple times within a given month.

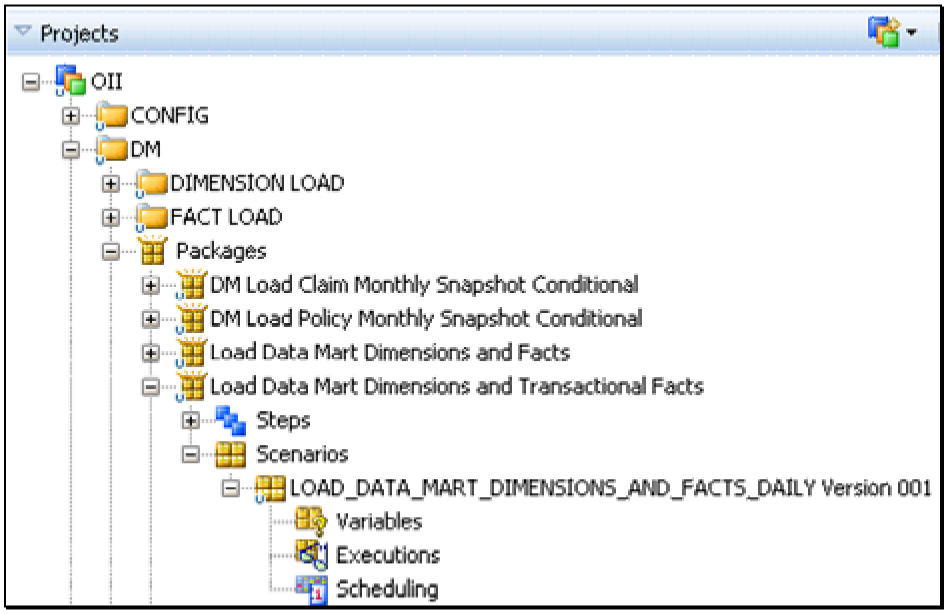
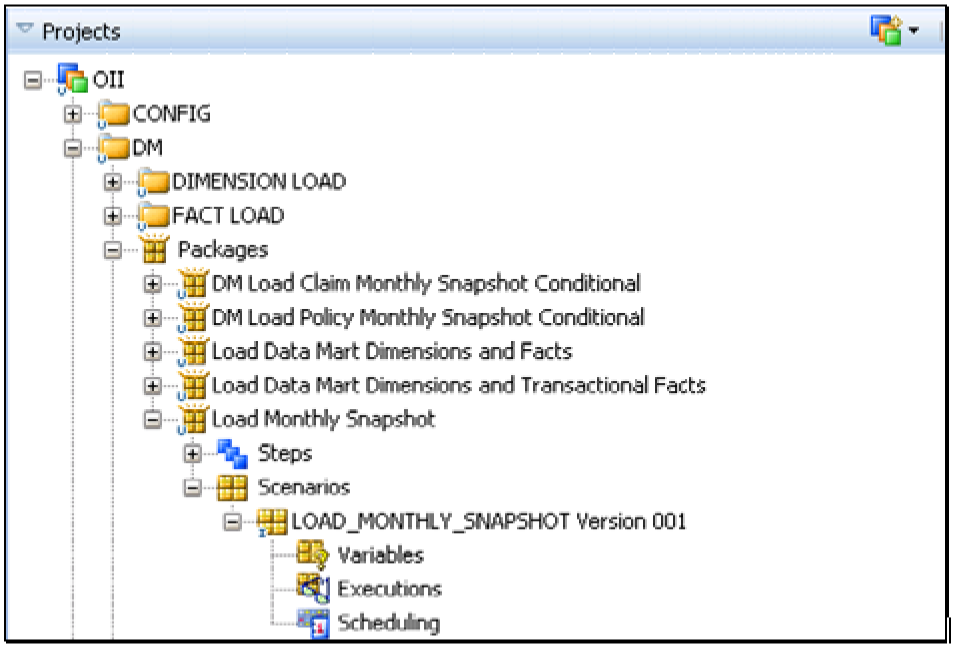

Projects
OII
CONFIG
DM
DIMENSION LOAD
FACT LOAD
Packages
DM Load Claim Monthly Snapshot Conditional
DM Load Policy Monthly Snapshot Conditional
Load Data Mart Dimensions and Facts
Load Data Mart Dimensions and Transactional Facts
Steps
Scenarios
LOAD_DATA_MART_DIMENSIONS_AND_FACTS_DAILY Version 001
Variables
Executions
Scheduling

Figure 19: Dimensions and Transactional Facts Load

Monthly Snapshot Facts Load

The LOAD_MONTHLY_SNAPSHOT scenario loads the Monthly Snapshot Fact tables. his scenario can be used to populate the Monthly Snapshot Fact tables after the Warehouse, Dimensions and Transactional Fact tables have been loaded.



Projects

- OII
  - CONFIG
  - DM
    - DIMENSION LOAD
    - FACT LOAD
    - Packages
      - DM Load Claim Monthly Snapshot Conditional
      - DM Load Policy Monthly Snapshot Conditional
      - Load Data Mart Dimensions and Facts
      - Load Data Mart Dimensions and Transactional Facts
      - Load Monthly Snapshot
        - Steps
        - Scenarios
          - LOAD_MONTHLY_SNAPSHOT Version 001
            - Variables
            - Executions
            - Scheduling

Figure 20: Monthly Snapshot Facts Load

Chapter 6

Data Visualization

OBIEE Repository

OII offers pre-defined OBIEE subject areas that match each LOB reference implementation in Warehouse Palette. Any modifications to the reference implementation must be manually updated in the OBIEE Repository.

Any UDFs that are utilized and contain data will need to be added to the presentation layer in the OBIEE Repository in order for them to be available to the end user in the reports and dashboards.

Please reference the following manuals in the OBIEE documentation set:

- *Oracle Business Intelligence Server Administration Guide*
- *Oracle Business Intelligence Presentation Services Administration Guide*
- *Oracle Business Intelligence Answers, Delivers, and Interactive Dashboards User Guide*

OBIEE Repository Linkage

Once data has been loaded to the Data Mart schema, the OII data model must be linked to the OBIEE Repository before logging into OBIEE Analytics to view the data. This section details those steps.

IMPORTANT	In the case that an LOB was created that was not included in the six that are available out of the box there are a number of steps that need to be performed in order to have that LOB included in the AnalyticsWeb repository. If this is the case then skip to *OBIEE Repository Linkage for New Or Updated LOB* on page 69. Complete the steps in that section before executing the steps in this section.

Step 1: Open the OBIEE Repository

1. Open the OBIEE Repository Administration Tool by selecting:

   Start>All Programs>Oracle Business Intelligence>Administration

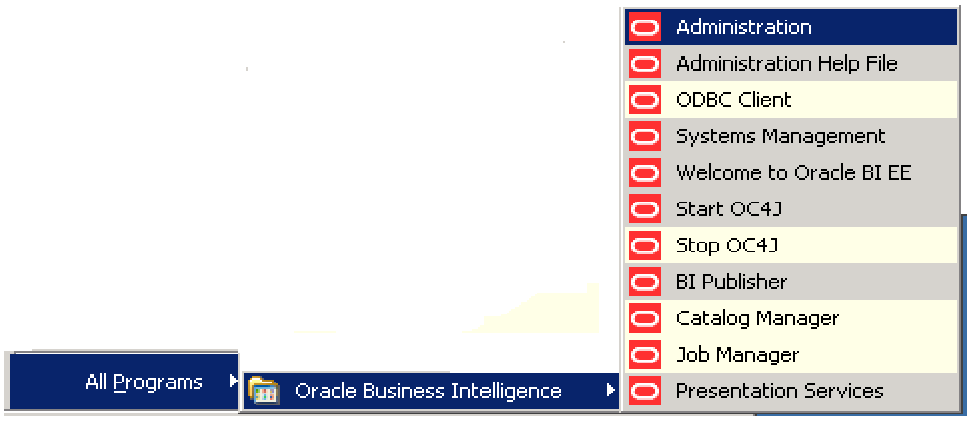


Figure 21: Open OBIEE Administration Tool

2. When the Administration Tool opens, select **File>Open>Online** or click the **Open Online** button.

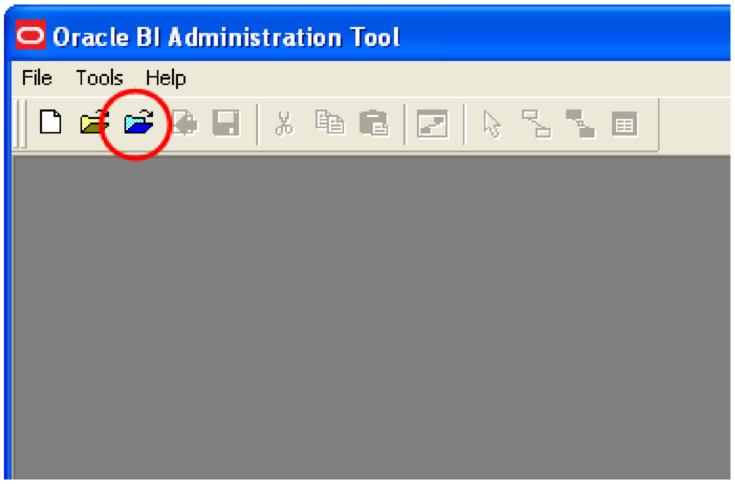
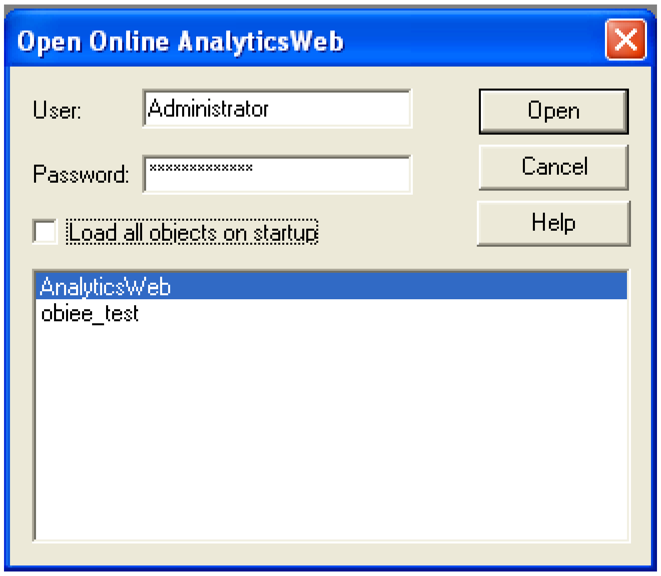


Figure 22: Open Online Button

3. In the login dialog box that appears, enter the password (the default password is *Administrator*) and click on the **Open** button.



User: Administrator

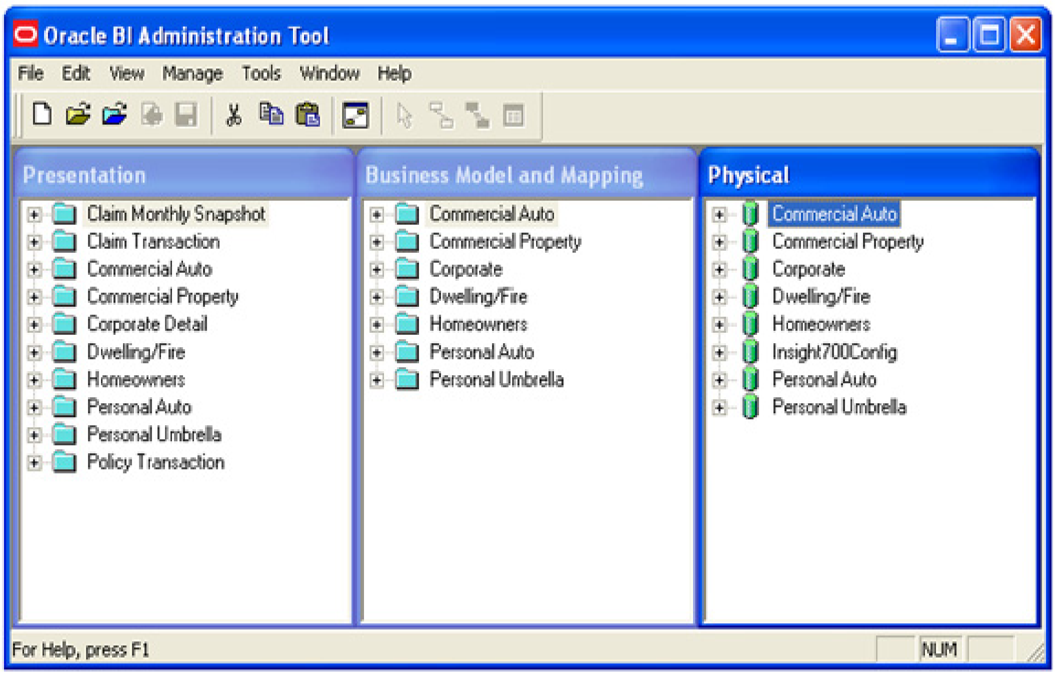
Password: **************

Load all objects on startup

AnalyticsWeb
obiee_test

Figure 23: Enter the Administration Tool Password

The following screen will open:


Oracle BI Administration Tool
File Edit View Manage Tools Window Help
For Help, press F1
NUM

Presentation
- Claim Monthly Snapshot
- Claim Transaction
- Commercial Auto
- Commercial Property
- Corporate Detail
- Dwelling/Fire
- Homeowners
- Personal Auto
- Personal Umbrella
- Policy Transaction

Business Model and Mapping
- Commercial Auto
- Commercial Property
- Corporate
- Dwelling/Fire
- Homeowners
- Personal Auto
- Personal Umbrella

Physical
- Commercial Auto
- Commercial Property
- Corporate
- Dwelling/Fire
- Homeowners
- Insight700Config
- Personal Auto
- Personal Umbrella

Figure 24: Layers of the OBIEE Repository for OII

Step 2: Configure the Line of Business Subject Areas

1. Go to the Physical Layer pane. This pane contains the Subject Areas for each Line of Business (Commercial Auto, Homeowners, etc.) and an additional Subject Area called **Insight700Config**. You will need to configure the settings under each of the Line of Business Subject Areas and the **Insight700Config** Subject Area.

Important The following configuration steps apply only to the Subject Areas for each Line of Business. The steps to configure the **Insight700Config** Subject Area are on page 67.

2. Expand each of the Line of Business Subject Areas under the Physical Layer pane. The Connection Pool icon will appear underneath each Subject Area:

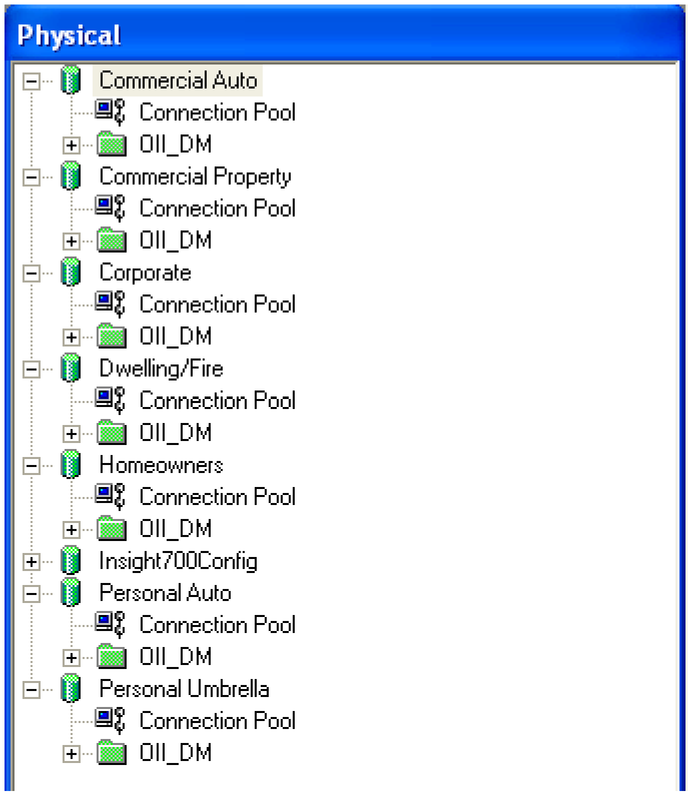


Figure 25: Subject Areas under the Physical Layer

3. Starting with the Commercial Auto Subject Area, click on the Connection Pool icon.

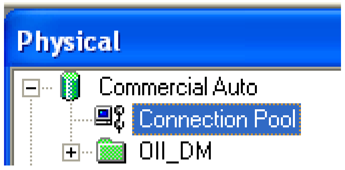
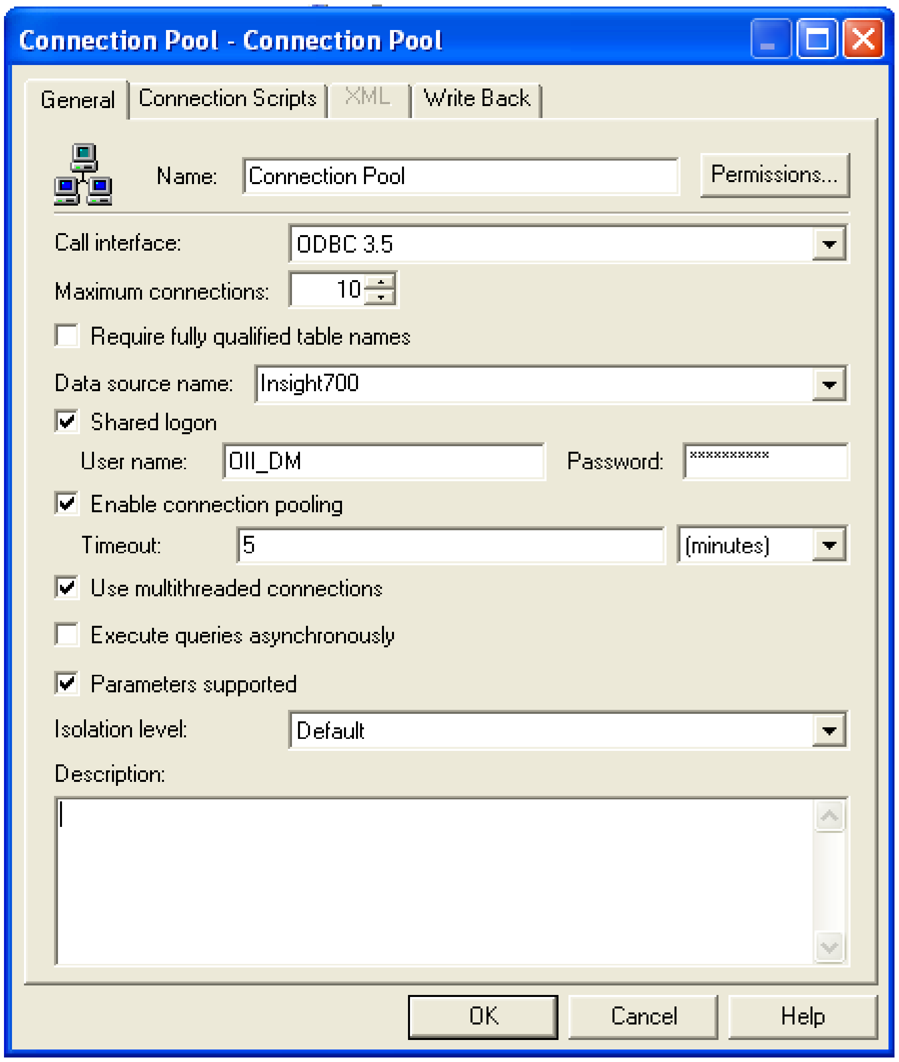


Figure 26: Connection Pool Icon

4. A message box will appear and ask you if you want to check out the Connection Pool.
5. Select **Yes**. The Connection Pool screen opens:



Name: Connection Pool

Call interface: ODBC 3.5

Maximum connections: 10

☐ Require fully qualified table names

Data source name: Insight700

☑ Shared logon

User name: OII_DM Password: **********

☑ Enable connection pooling

Timeout: 5 (minutes)

☑ Use multithreaded connections

☐ Execute queries asynchronously

☑ Parameters supported

Isolation level: Default

Description:

Figure 27: The Connection Pool Screen

6. Make sure that the **Require fully qualified table names** option is unchecked.

7. Under the “Shared Login” section, update the user name and password with the same ones that you entered for the **Insight700** data source.

Note Recall that as part of the post-installation configuration steps for OBIEE (as described in *Chapter 12: OBIEE Post-Installation Steps* in the *Oracle Insurance Insight Installation Guide*) you created two ODBC data sources: **Insight700** and **Insight700Config.** The default user name for **Insight700** is **OII_DM.**

---

8. Leave all of the other settings as is and click **OK** to close the screen.
9. To ensure the user name and password entered in step 7 above is correct:
   - **a.** Open any table under one of the Subject Areas.
   - **b.** Right click on the table you selected and choose ‘Update Row Count’—. You will be prompted to check out the table’s objects—. Click ‘Yes’.

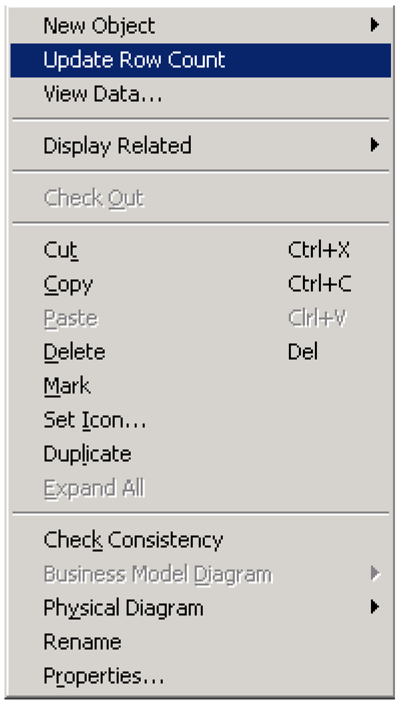
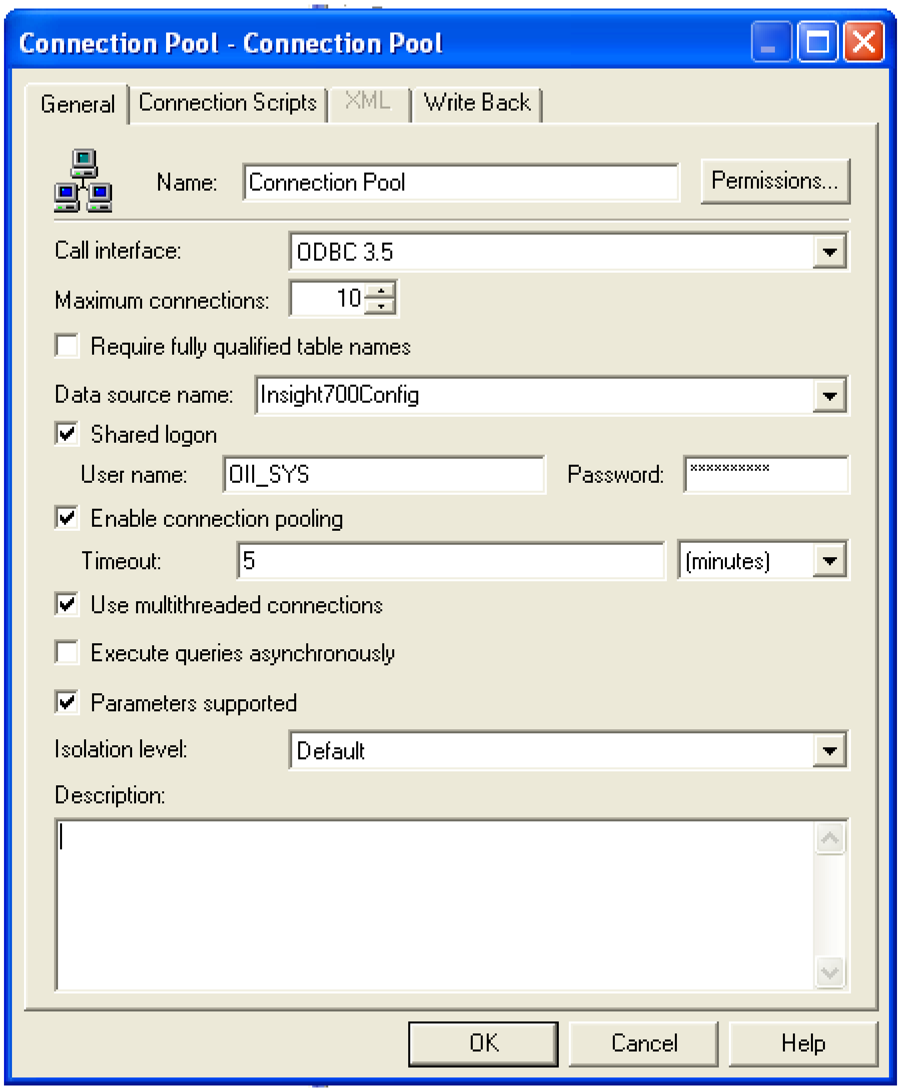


Figure 28: Update Row Count

   - **c.** If an error occurs it means that the user name/password is incorrect or the ODBC connection information supplied is incorrect.
10. Repeat steps 1-8 for each of the remaining Line of Business Subject Areas:
    - Commercial Property
    - Corporate
    - Dwelling Fire
    - Homeowners
    - Personal Auto
    - Personal Umbrella

Step 3: Configure the Insight700Config Subject Area

1. Click on the Connection Pool icon under the **Insight700Config** Subject Area to open the Connection Pool window.



Connection Pool - Connection Pool

General | Connection Scripts | XML | Write Back

Name: Connection Pool — Permissions...

Call interface: ODBC 3.5

Maximum connections: 10

☐ Require fully qualified table names

Data source name: Insight700Config

☑ Shared logon

User name: OII_SYS — Password: ***********

☑ Enable connection pooling

Timeout: 5 (minutes)

☑ Use multithreaded connections

☐ Execute queries asynchronously

☑ Parameters supported

Isolation level: Default

Description:

OK | Cancel | Help

Figure 29: Insight700Config Connection Pool Window

2. Make sure the data source name is **Insight700Config.**
3. Make sure that the **Require fully qualified table names** option is unchecked.
4. Under the **Shared Login** section, update the user name and password with the same ones that you entered for the **Insight700Config** data source.

Note Recall that as part of the post-installation configuration steps for OBIEE (as described in *Chapter 12: OBIEE Post-Installation Steps* in the *Oracle Insurance Insight Installation Guide*) you created two ODBC data sources: **Insight700** and **Insight700Config**. The default user name for **Insight700Config** is **OII_SYS**.

5. Click **OK** to close the screen.

Step 4: Check in the Changes to the Physical Layer

1. Check in all the changes by click on “Check-in” button at the top tool bar:

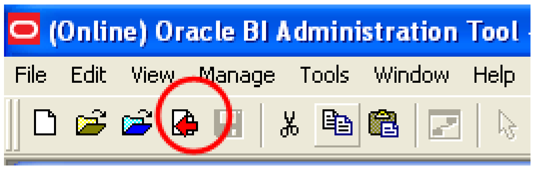
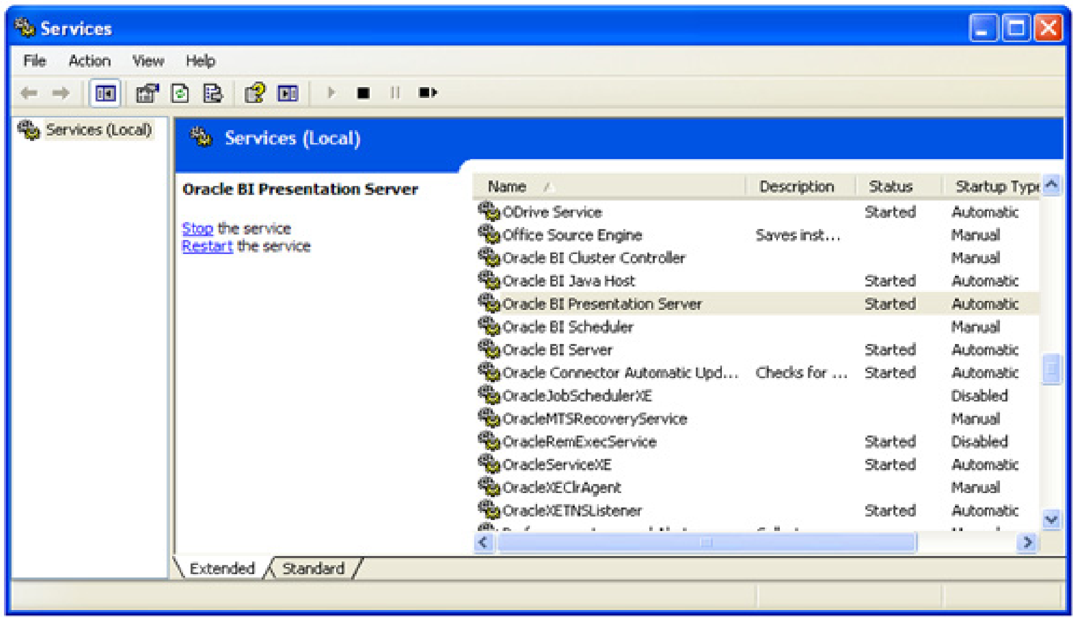


Figure 30: Check in Changes

2. Click the **Save** button.
3. Click the **Close** and **Exit** buttons to exit the OBIEE Administration Tool.

Step 5: Stop and Restart the Oracle BI Server and Oracle BI Presentation Server

1. From the **Start** menu select **Control Panel>Administrative Tools>Services** to open the Services window.
2. Locate the Oracle BI Server and the Oracle BI Presentation Server in the list of services.



Oracle BI Presentation Server

Stop the service
Restart the service

Name	Description	Status	Startup Typ
ODrive Service		Started	Automatic
Office Source Engine	Saves inst...		Manual
Oracle BI Cluster Controller			Manual
Oracle BI Java Host		Started	Automatic
Oracle BI Presentation Server		Started	Automatic
Oracle BI Scheduler			Manual
Oracle BI Server		Started	Automatic
Oracle Connector Automatic Upd...	Checks for ...	Started	Automatic
OracleJobSchedulerXE			Disabled
OracleMTSRecoveryService			Manual
OracleRemExecService		Started	Disabled
OracleServiceXE		Started	Automatic
OracleXEClrAgent			Manual
OracleXETNSListener		Started	Automatic

Figure 31: Locate Oracle BI Presentation Server and Oracle BI Server

3. Separately highlight the name of the server in the list and click the **Stop** link.
4. Separately highlight the name of the server in the list and click the **Restart** link.

OBIEE Repository Linkage for New Or Updated LOB

Before you perform these steps it is assumed that the following tasks have been performed in the Warehouse Palette:

- a new LOB has been created or an existing LOB has been modified,
- all Risk Entities for the LOB have been defined,
- all Attributes for the Risk Entities have been defined, and
- the LOB has been successfully published.

For instructions on how to do this please refer to the *Oracle Insurance Insight Warehouse Palette User Guide*.

Step 1: Perform an Import

1. Open the OBIEE Repository Administration Tool by selecting:

   Start>All Programs>Oracle Business Intelligence>Administration

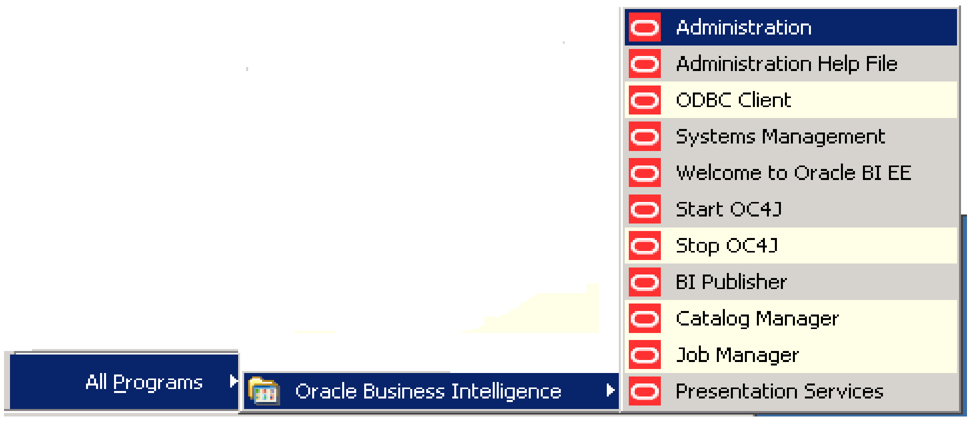


Figure 32: Open OBIEE Administration Tool

2. When the Administration Tool opens, select **File>Open>Online** or click the **Open Online** button.

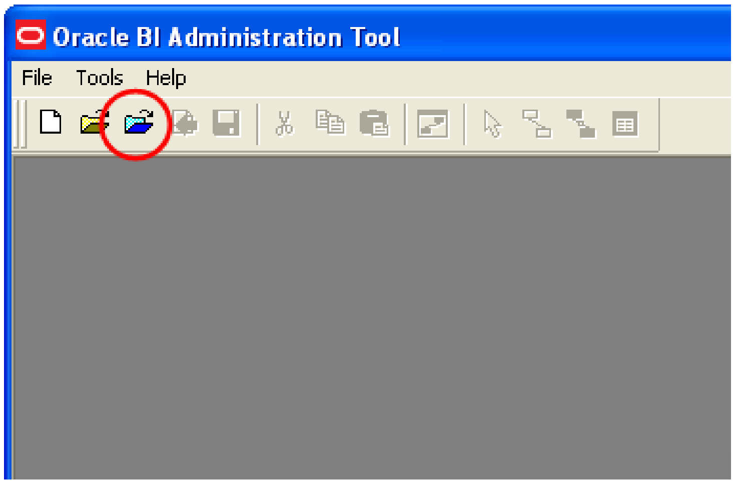
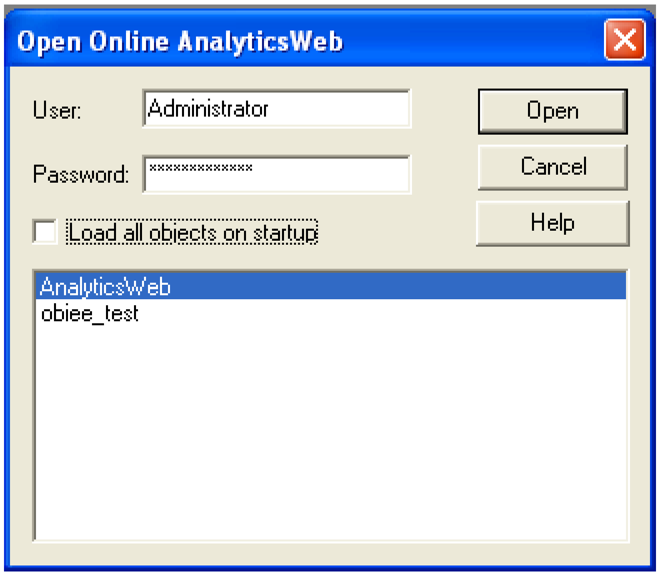


Figure 33: Open Online Button

3. In the login dialog box that appears, enter the password (the default password is *Administrator*) and click on the **Open** button.



User: Administrator

Password: **************

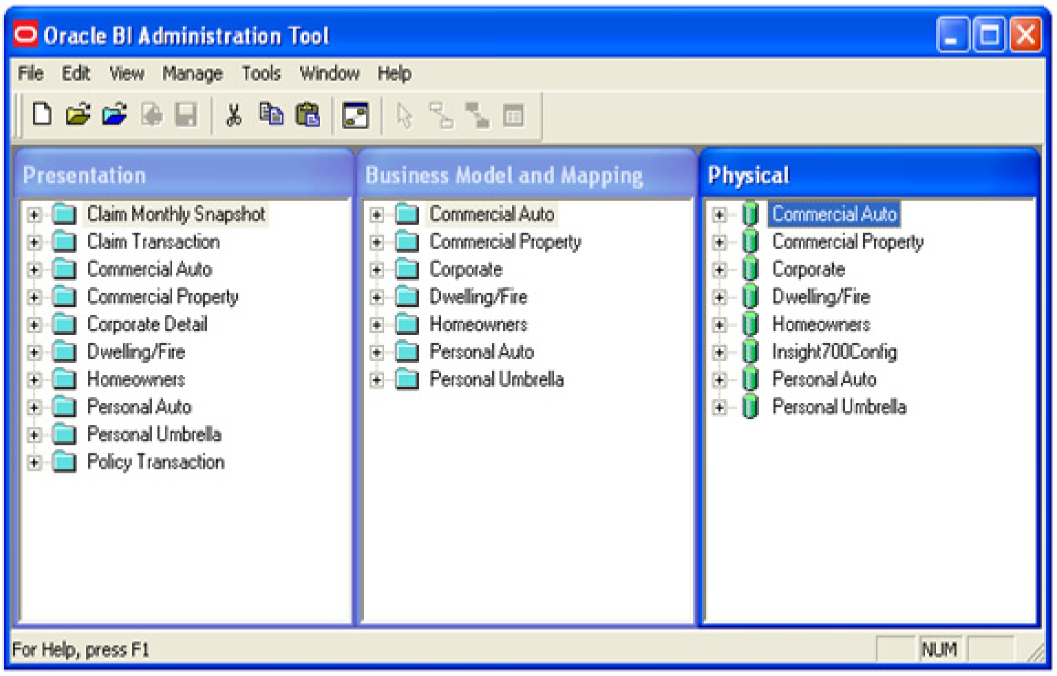
Load all objects on startup

AnalyticsWeb

obiee_test

Figure 34: Enter the Administration Tool Password

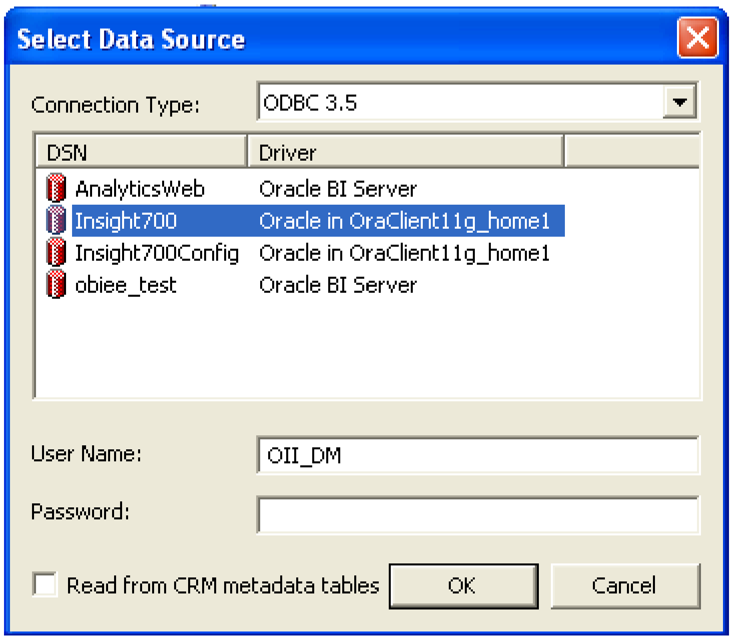
The following Repository screen will open:


Oracle BI Administration Tool
File Edit View Manage Tools Window Help
For Help, press F1
NUM

Presentation	Business Model and Mapping	Physical
Claim Monthly Snapshot	Commercial Auto	Commercial Auto
Claim Transaction	Commercial Property	Commercial Property
Commercial Auto	Corporate	Corporate
Commercial Property	Dwelling/Fire	Dwelling/Fire
Corporate Detail	Homeowners	Homeowners
Dwelling/Fire	Personal Auto	Insight700Config
Homeowners	Personal Umbrella	Personal Auto
Personal Auto		Personal Umbrella
Personal Umbrella		
Policy Transaction		

Figure 35: Layers of the OBIEE Repository for OII

4. Select **File>Import>from Database.** The **Select Data Source** dialog will open.



Connection Type: ODBC 3.5

DSN	Driver
AnalyticsWeb	Oracle BI Server
Insight700	Oracle in OraClient11g_home1
Insight700Config	Oracle in OraClient11g_home1
obiee_test	Oracle BI Server

User Name: OII_DM

Password:

☐ Read from CRM metadata tables

Figure 36: Select "Insight700"

5. Select **Insight700**.

6. Enter the **user name** and **password** for the **Insight700** data source (the same ones you used when you created **Insight700** as described in *Chapter 12: OBIEE Post-Installation Steps* in the *Oracle Insurance Insight Installation Guide*)**.** The default user name for the **Insight700** is **OII_DM** as shown in the figure above.
7. Click **OK**. The **Import** dialog will open.

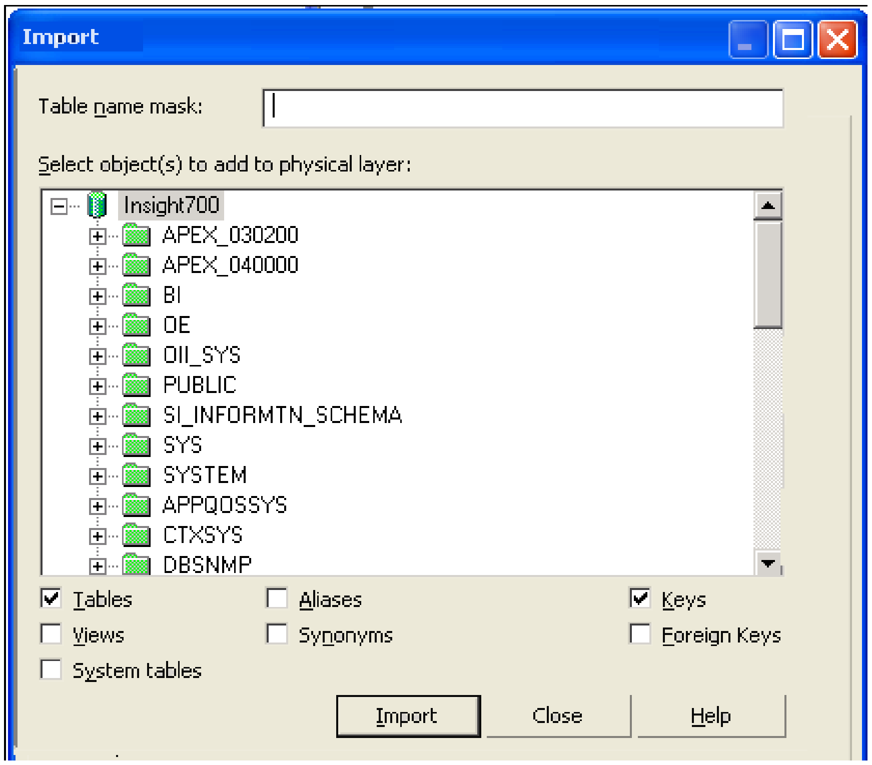
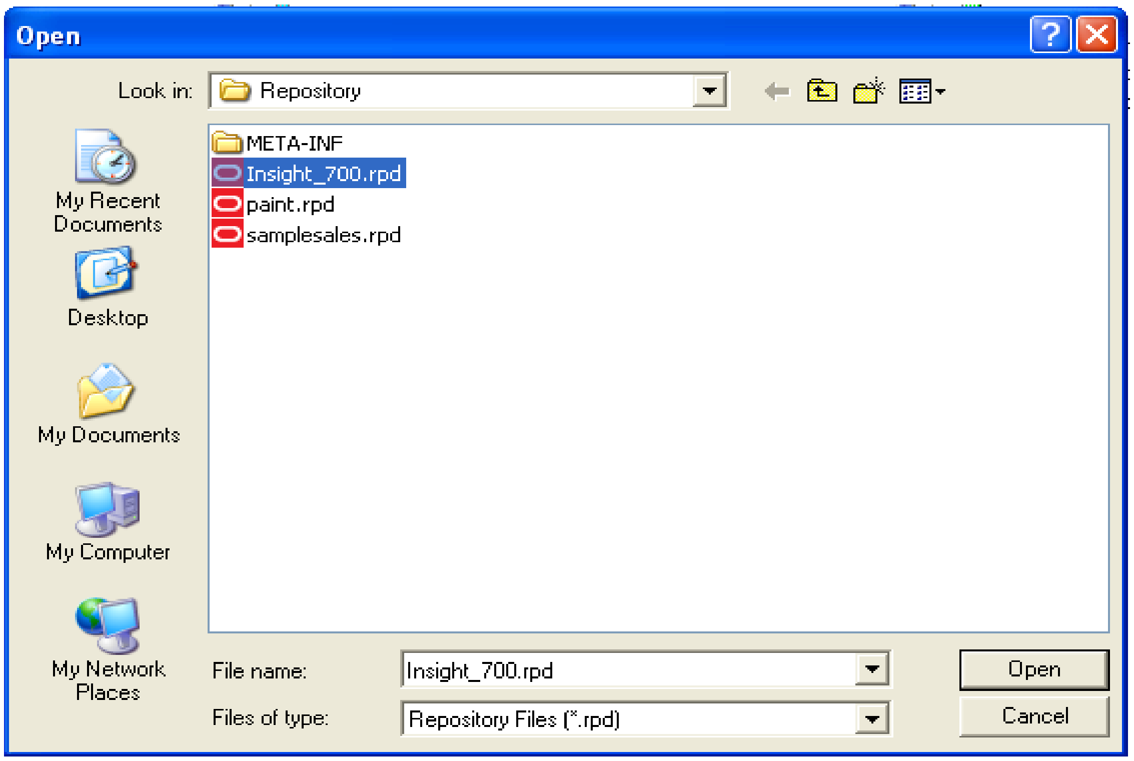


Figure 37: Import Dialog Box

8. To complete the import steps refer to the *Oracle Business Intelligence Server Administration Guide*.

STEP 2: REGENERATE THE METADATA DICTIONARY

1. From the Oracle BI Administration Tool menu select **File>Open>Offline.** The **Open Dialog** box appears.

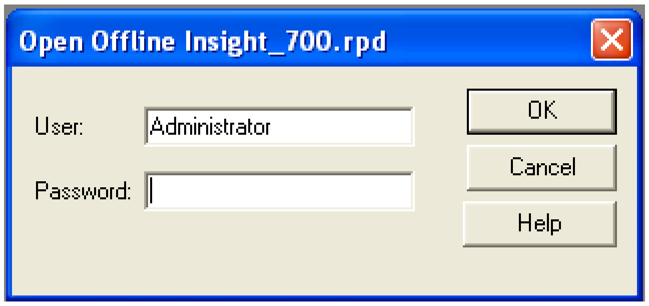


META-INF
Insight_700.rpd
paint.rpd
samplesales.rpd

File name: Insight_700.rpd
Files of type: Repository Files (*.rpd)

Figure 38: Select Insight_700.rpd

2. In the Open Dialog box select the **Insight_700.rpd** repository file and click **Open**.
3. You will get the information dialog box stating that the repository file can only be opened as Read-Only.
4. Click **Yes**. You will be prompted for a password (the default password is *Administrator).*



User: Administrator
Password:

Figure 39: Log Into the Repository

5. Enter “Administrator” as the password and click **OK** The Oracle BI Administration Tool screen will open.

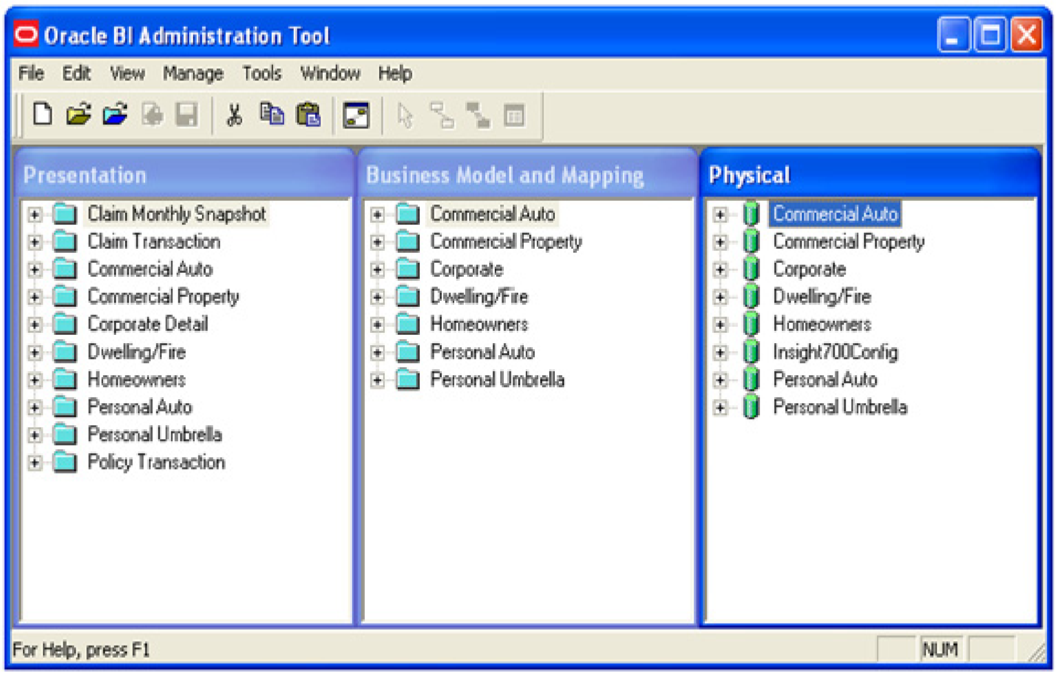


Figure 40: Oracle BI Administration Tool Screen

6. From the Oracle BI Administration Tools menu select **Tools>Utilities**. The **Utilities** dialog will open.
7. From the list of utilities select **Generate Metadata Dictionary** and click the **Execute** button.

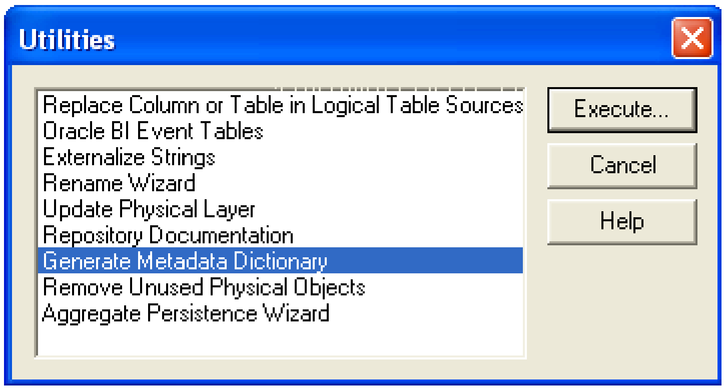
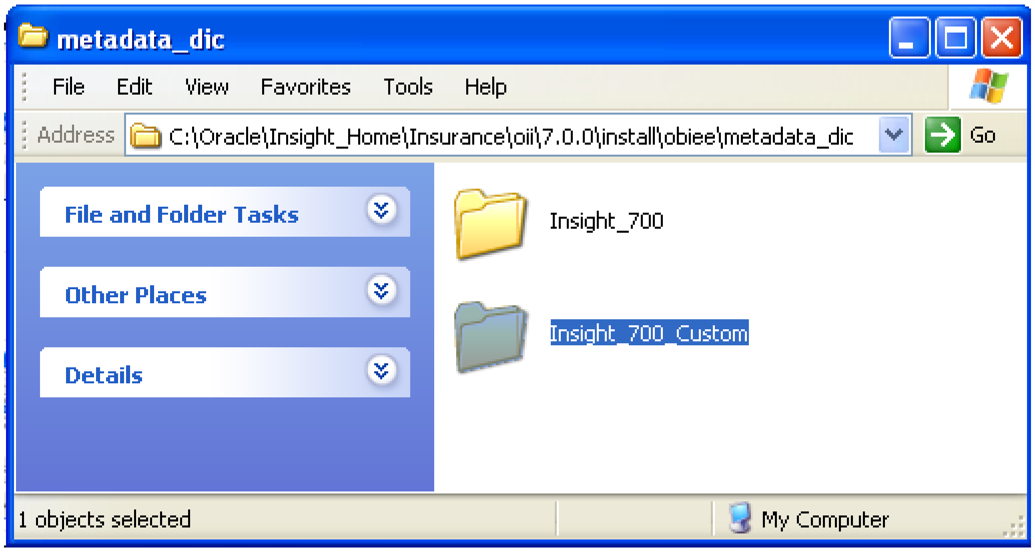

Utilities
Replace Column or Table in Logical Table Sources
Oracle BI Event Tables
Externalize Strings
Rename Wizard
Update Physical Layer
Repository Documentation
Generate Metadata Dictionary
Remove Unused Physical Objects
Aggregate Persistence Wizard
Execute...
Cancel
Help

Figure 41: Select “Generate Metadata Dictionary”

8. The choose directory dialog box will open and you will need to choose a directory for your metadata dictionary. In this example, we have created a directory named **Insight_700_Custom** in order to preserve the original out of the box metadata dictionary.

C:\Oracle\Insight_Home\Insurance\oii\7.0.0\install\obiee\metadata_dic\Insight_700_Custom



C:\Oracle\Insight_Home\Insurance\oii\7.0.0\install\obiee\metadata_dic

1 objects selected

Figure 42: Insight_700_Custom Directory

9. Using the OBIEE Administration Manual, define the proper logical layer and presentation layer object for the newly added or modified LOB's.

Step 3: Update the OBIEE analytics.war File

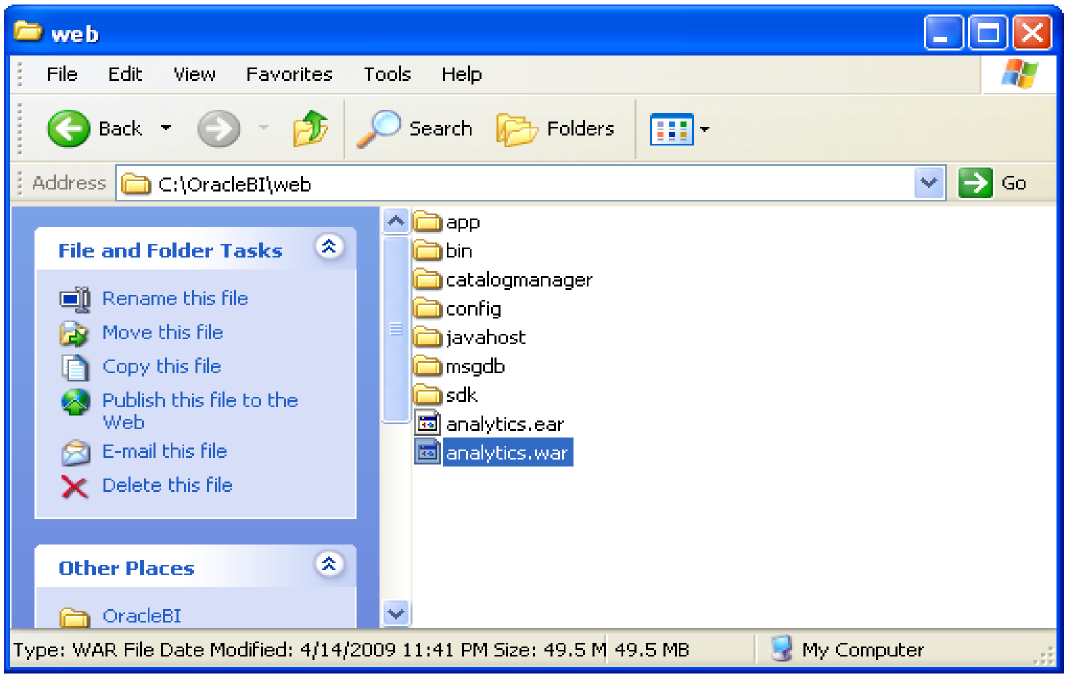
This step requires you to copy the new **Insight_700_Custom** folder into the **dictionary** folder in the **analytics.war** file.

1. Locate the OBIEE **analytics.war** file. Normally this file is located under:

   <OBIEE_Root>\web

   For example:

   C:\OracleBI\web



Type: WAR File Date Modified: 4/14/2009 11:41 PM Size: 49.5 M 49.5 MB

Figure 43: <OBIEE_Root>\Web\analytics.war

2. Copy **analytics.war** to a temporary location on your computer.

3. Use **7-Zip** to open the **analytics.war** file (right-click on **analytics.war a**nd from the pop-up menu select **7-Zip>Open Archive**).

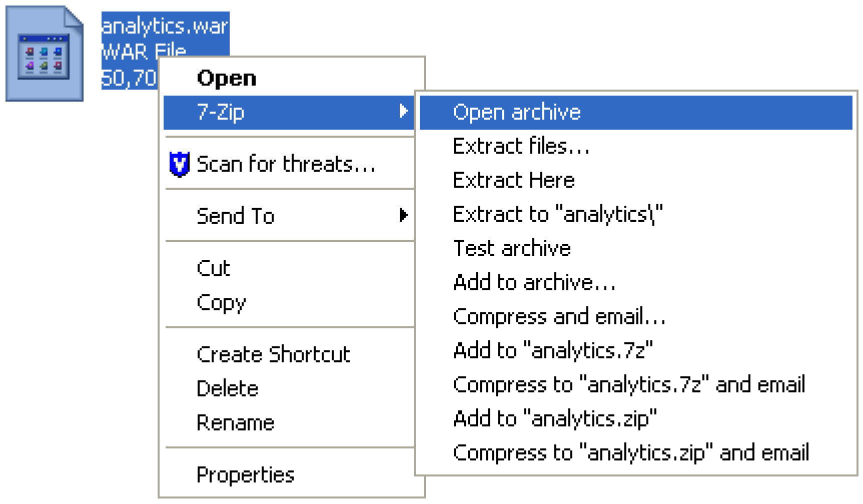
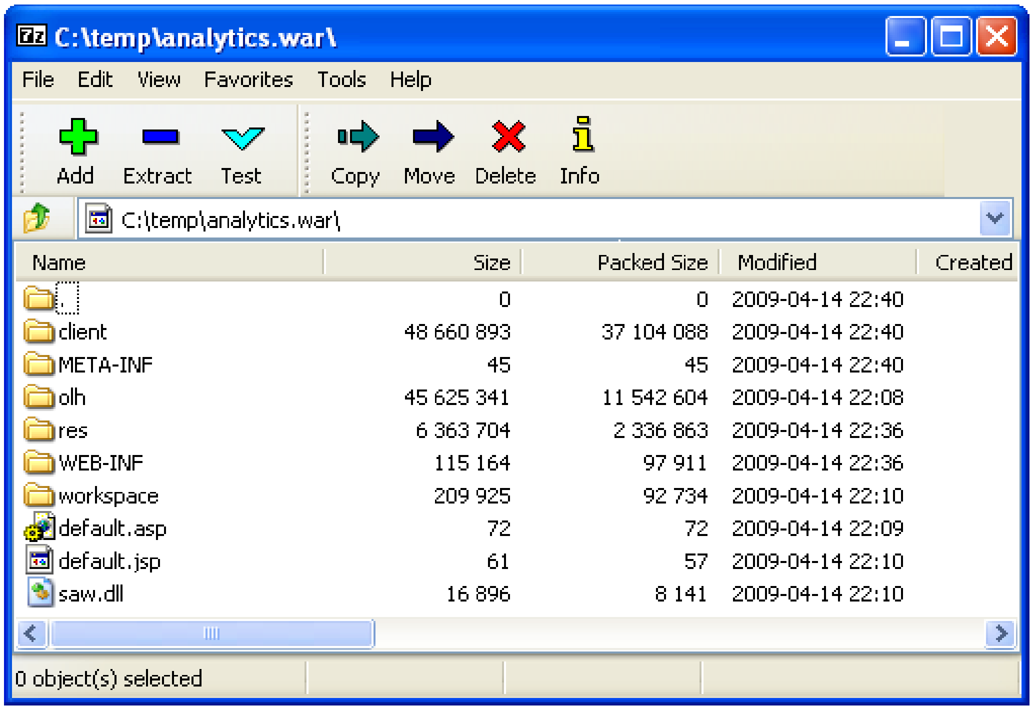


Figure 44: Open the Analytics.war File

The contents of the **analytics.war** file will be similar to the figure below:


C:\temp\analytics.war\
File Edit View Favorites Tools Help
Add Extract Test Copy Move Delete Info
C:\temp\analytics.war\

Name	Size	Packed Size	Modified	Created
..	0	0	2009-04-14 22:40	
client	48 660 893	37 104 088	2009-04-14 22:40	
META-INF	45	45	2009-04-14 22:40	
olh	45 625 341	11 542 604	2009-04-14 22:08	
res	6 363 704	2 336 863	2009-04-14 22:36	
WEB-INF	115 164	97 911	2009-04-14 22:36	
workspace	209 925	92 734	2009-04-14 22:10	
default.asp	72	72	2009-04-14 22:09	
default.jsp	61	57	2009-04-14 22:10	
saw.dll	16 896	8 141	2009-04-14 22:10	

0 object(s) selected

Figure 45: Contents of analytics.war

4. Within this file create a folder called **dictionary**.

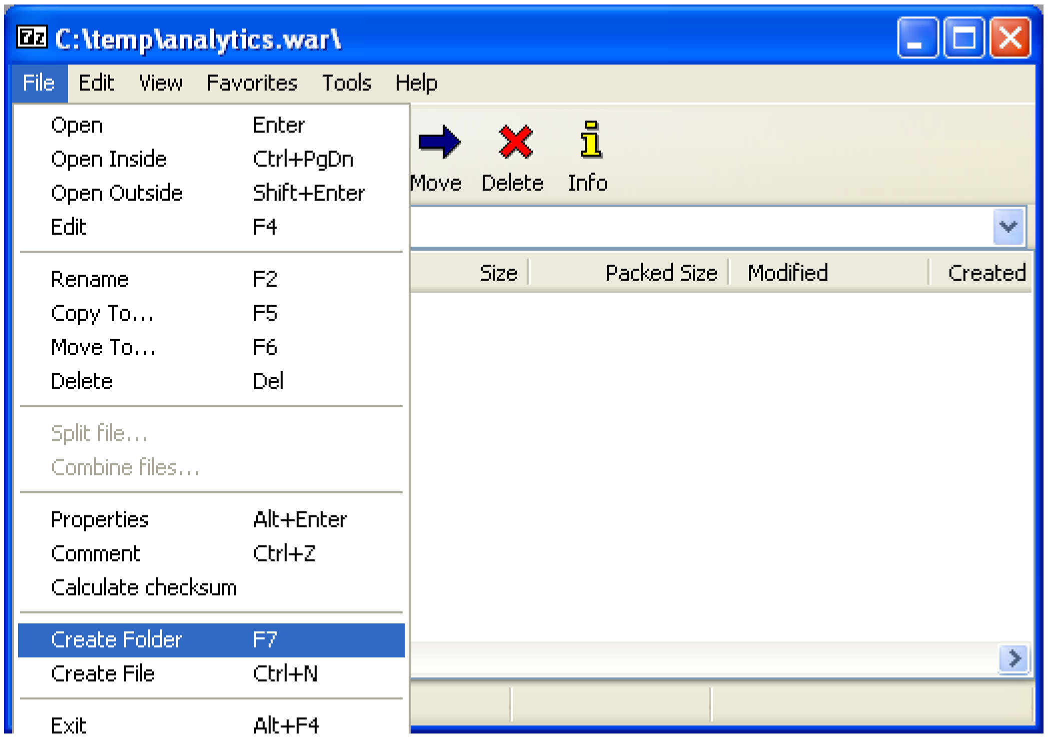


Figure 46: Create a New Folder Called "dictionary"

5. Leave the **7-Zip** window open and go to the folder:

<OII_Root>\install\obiee\metadata_dic\Insight_700_Custom

For example:

C:\Oracle\Insight_Home\Insurance\oii\7.0.0\install\obiee\metadata_dic\Insight_700_Custom

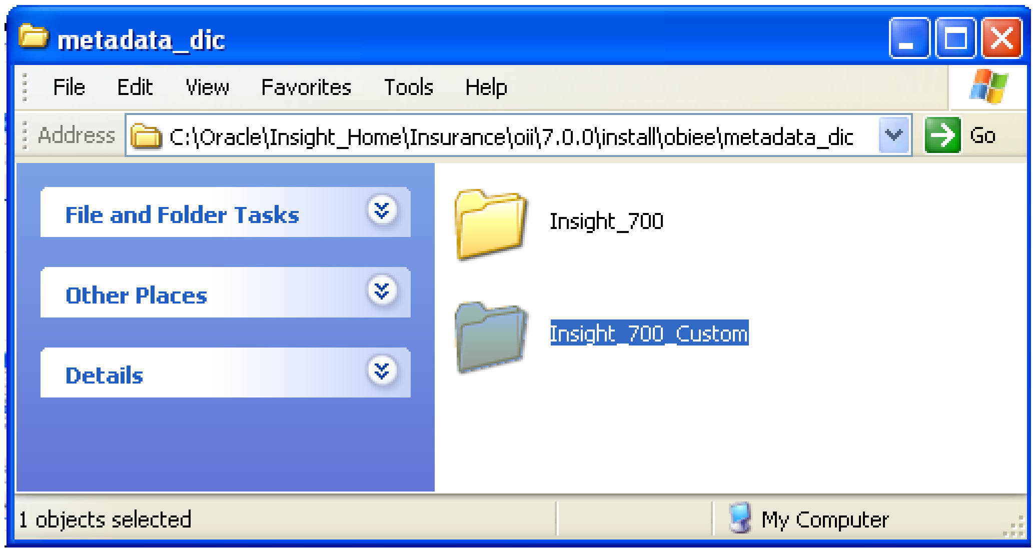


Figure 47: Insight_700_Custom Directory

6. Copy the entire **Insight_700_Custom** folder into the **dictionary** folder in the **7-Zip** window.
7. Save **analytics.war** and close the **7-Zip** window.

STEP 4: DEPLOY OBIEE ANALYTICS.WAR FILE

This step requires you to deploy the **analytics.war** file that you just updated in the previous section on WebLogic.

Important If the **analytics.war** file has previously been deployed on WebLogic you must delete it before deploying the updated one.

1. If it is not already started, start the WebLogic server from the Start Menu:

   Start Menu>Oracle Fusion Middleware>User Projects<Your Server Domain>Start Admin Server for WebLogic Server Domain

Note <Your Server Domain> is the name of the domain you created when you installed WebLogic.

   A separate window will open and display a series of messages to indicate that the domain server is up and running.

2. Once the WebLogic server domain is running, select:

   Start Menu>All Programs>Oracle WebLogic>User Projects><Your Server Domain>>Admin Server Console

   The WebLogic Server Administration Console login screen will open:

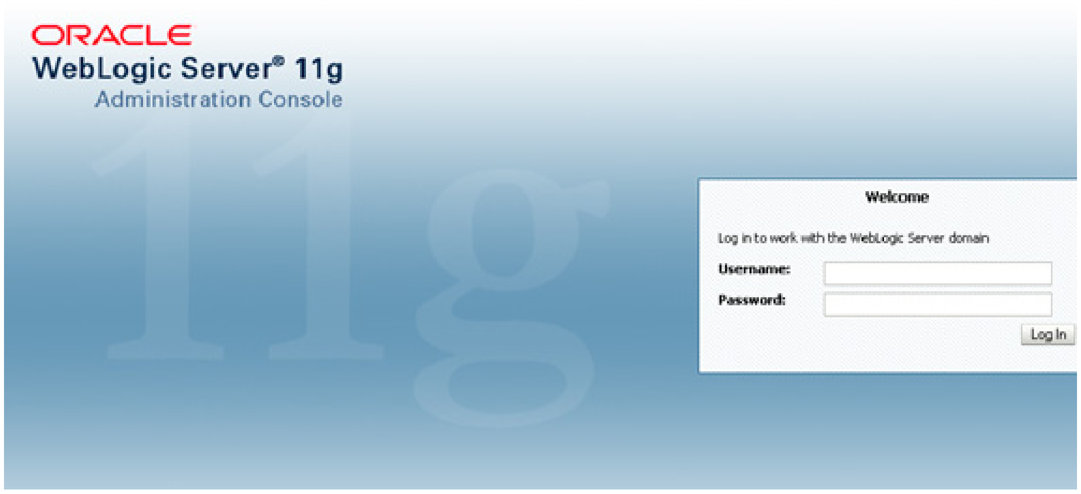
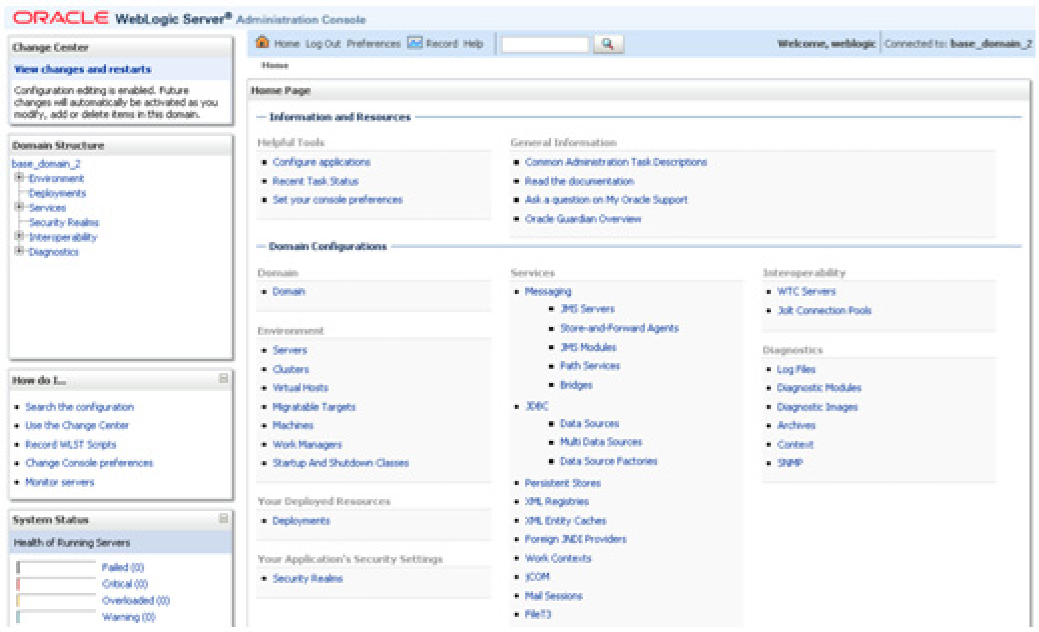


Figure 48: WebLogic Server Administration Console Login Screen

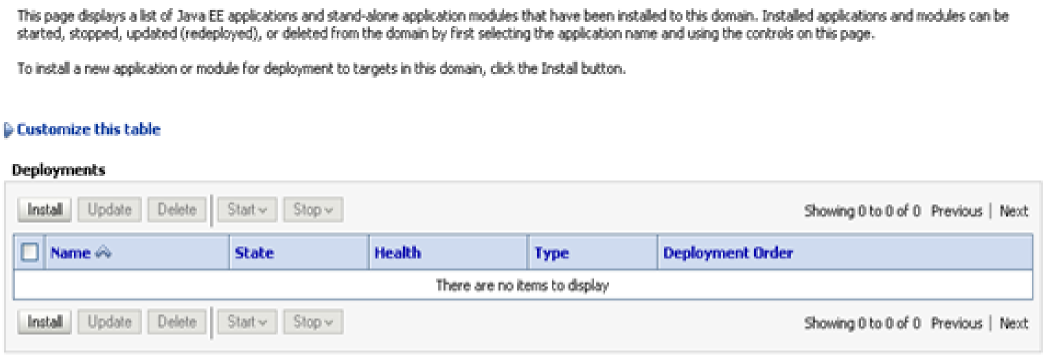
3. Enter the administrator user name and password that you created when you set up the server domain. The WebLogic administration console opens:



Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

Figure 49: WebLogic Server Administration Console

4. Click the **Depolyments** link under the Domain Structure pane. The Deployments screen will open.



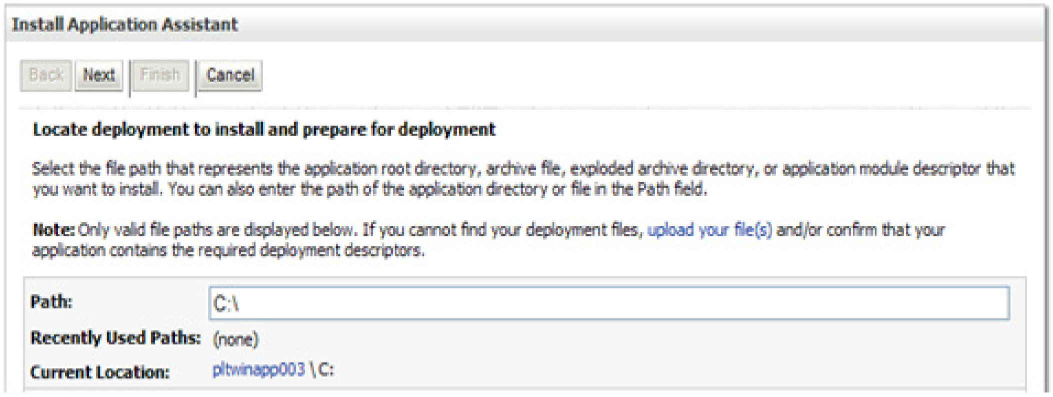
This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

To install a new application or module for deployment to targets in this domain, click the Install button.

☐	Name	State	Health	Type	Deployment Order
	There are no items to display				

Figure 50: Deployments Screen

5. Click the **Install** button. The Install Application Assistant screen will open.



Install Application Assistant

Locate deployment to install and prepare for deployment

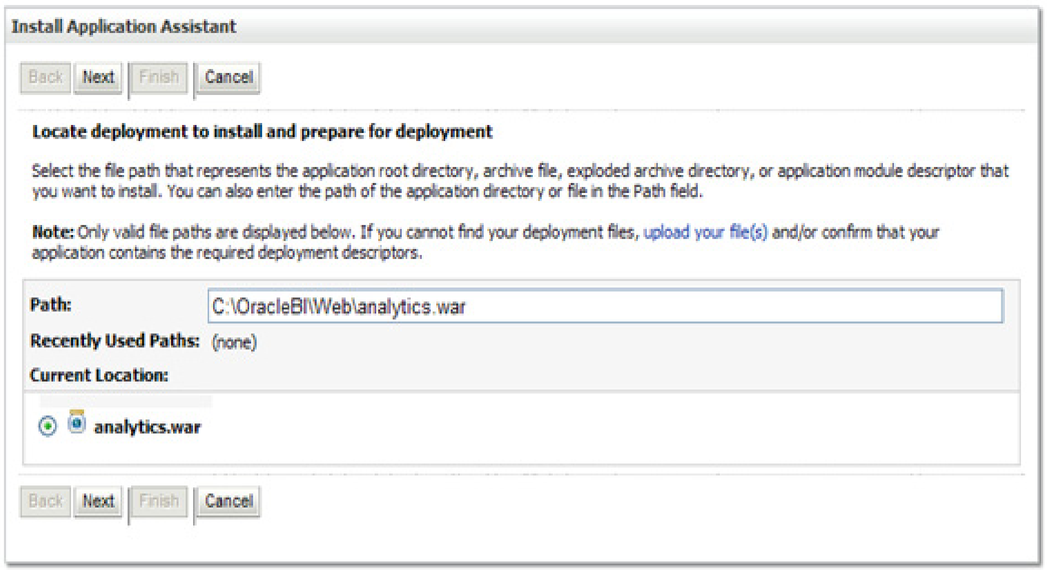
Select the file path that represents the application root directory, archive file, exploded archive directory, or application module descriptor that you want to install. You can also enter the path of the application directory or file in the Path field.

Note: Only valid file paths are displayed below. If you cannot find your deployment files, upload your file(s) and/or confirm that your application contains the required deployment descriptors.

Path: C:\
Recently Used Paths: (none)
Current Location: pltwinapp003 \ C:

Figure 51: Install Application Assistant Screen

6. In the **Path:** box, enter the path of the **analytics.war**file that you updated in *Step 3: Update the OBIEE analytics.war File* on page 76.
7. Click **Enter** on your keyboard. The **analytics.war** file will appear as one of the selections.



Install Application Assistant

Locate deployment to install and prepare for deployment

Select the file path that represents the application root directory, archive file, exploded archive directory, or application module descriptor that you want to install. You can also enter the path of the application directory or file in the Path field.

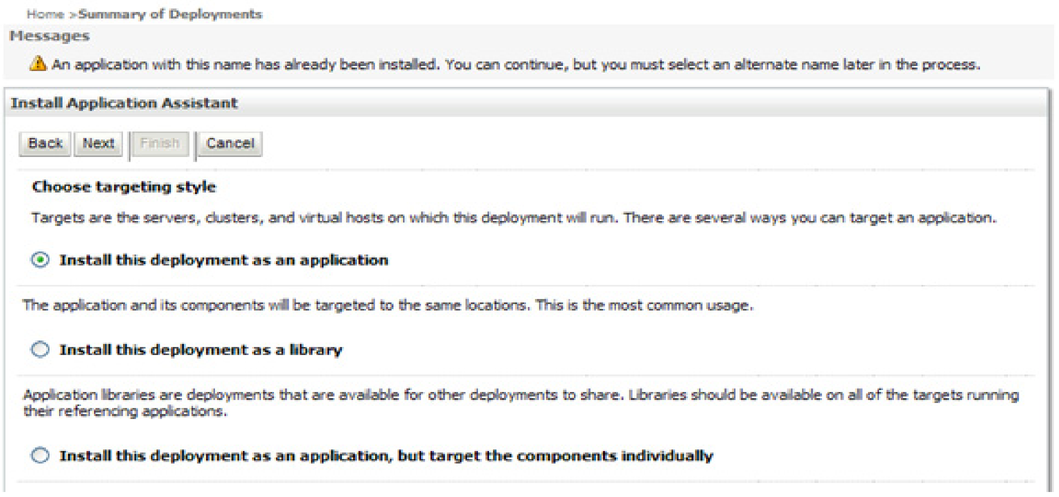
Note: Only valid file paths are displayed below. If you cannot find your deployment files, upload your file(s) and/or confirm that your application contains the required deployment descriptors.

Path: C:\OracleBI\Web\analytics.war
Recently Used Paths: (none)
Current Location:

● analytics.war

Figure 52: The Path to the analytics.war File

8. Select the **analytics.war**file and click the **Next** button. The following screen will open.



Home >Summary of Deployments

Messages

⚠ An application with this name has already been installed. You can continue, but you must select an alternate name later in the process.

Install Application Assistant

Choose targeting style

Targets are the servers, clusters, and virtual hosts on which this deployment will run. There are several ways you can target an application.

[✓ Install this deployment as an application]

The application and its components will be targeted to the same locations. This is the most common usage.

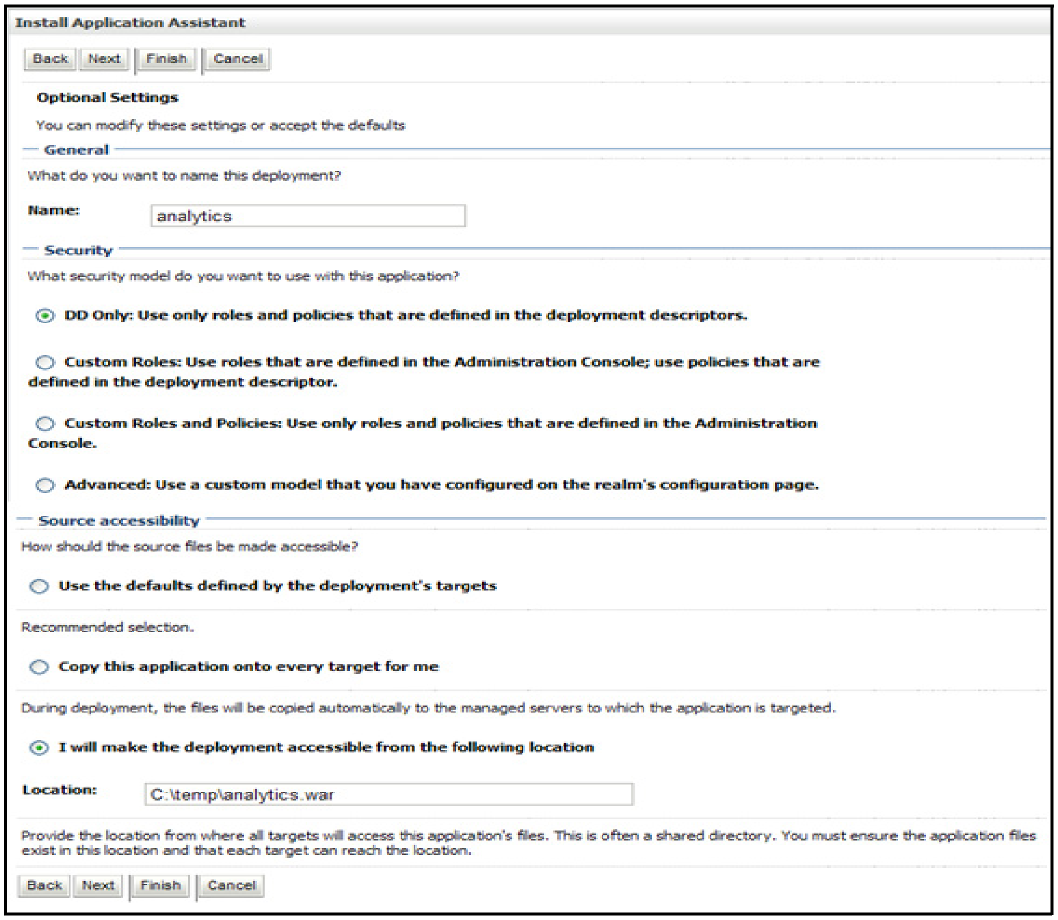
Install this deployment as a library

Application libraries are deployments that are available for other deployments to share. Libraries should be available on all of the targets running their referencing applications.

Install this deployment as an application, but target the components individually

Figure 53: Choose Targeting Style

9. Select **Install this deployment as an application** and click **Next**. The Optional Settings screen opens:



Install Application Assistant

Optional Settings

You can modify these settings or accept the defaults

General

What do you want to name this deployment?

Name: analytics

Security

What security model do you want to use with this application?

[✓ DD Only: Use only roles and policies that are defined in the deployment descriptors.]

Custom Roles: Use roles that are defined in the Administration Console; use policies that are defined in the deployment descriptor.

Custom Roles and Policies: Use only roles and policies that are defined in the Administration Console.

Advanced: Use a custom model that you have configured on the realm's configuration page.

Source accessibility

How should the source files be made accessible?

Use the defaults defined by the deployment's targets

Recommended selection.

Copy this application onto every target for me

During deployment, the files will be copied automatically to the managed servers to which the application is targeted.

[✓ I will make the deployment accessible from the following location]

Location: C:\temp\analytics.war

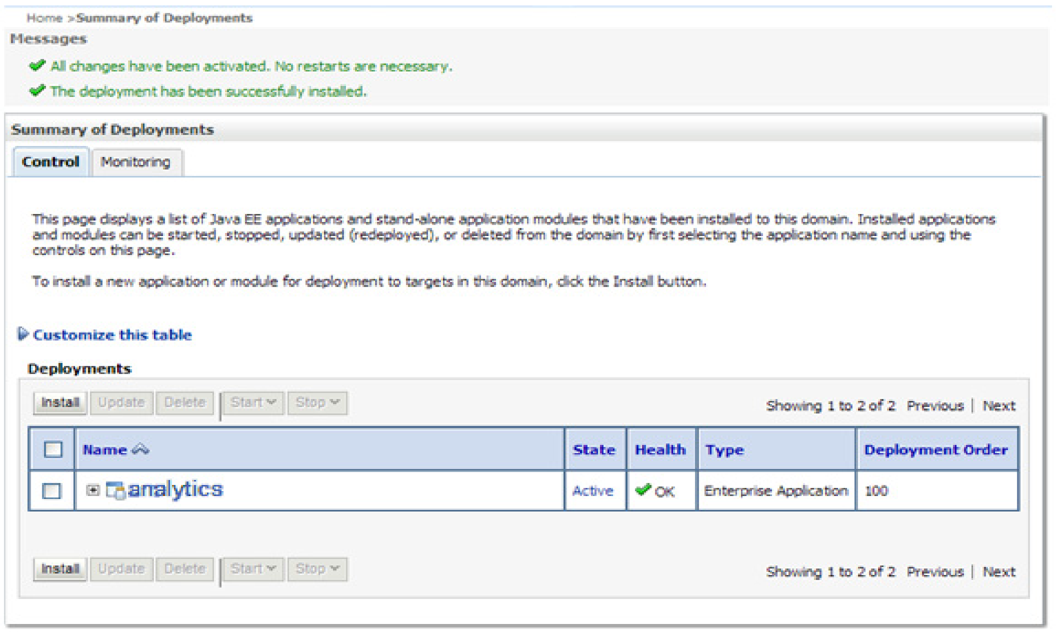
Provide the location from where all targets will access this application's files. This is often a shared directory. You must ensure the application files exist in this location and that each target can reach the location.

Figure 54: Optional Settings

10. Scroll down to the Source accessibility section of the screen and select:

    I will make the deployment accessible from the following location.

11. Leave the rest of the settings on this screen. The deployment name should appear as **analytics**.
12. Click the **Finish** button.
13. After the **analytics.war** file has been added, you will be returned to the Summary of Deployments screen. The **analytics.war** file will be listed. Read the messages at the top of the screen and correct any errors.



✓ All changes have been activated. No restarts are necessary.
✓ The deployment has been successfully installed.

This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

To install a new application or module for deployment to targets in this domain, click the Install button.

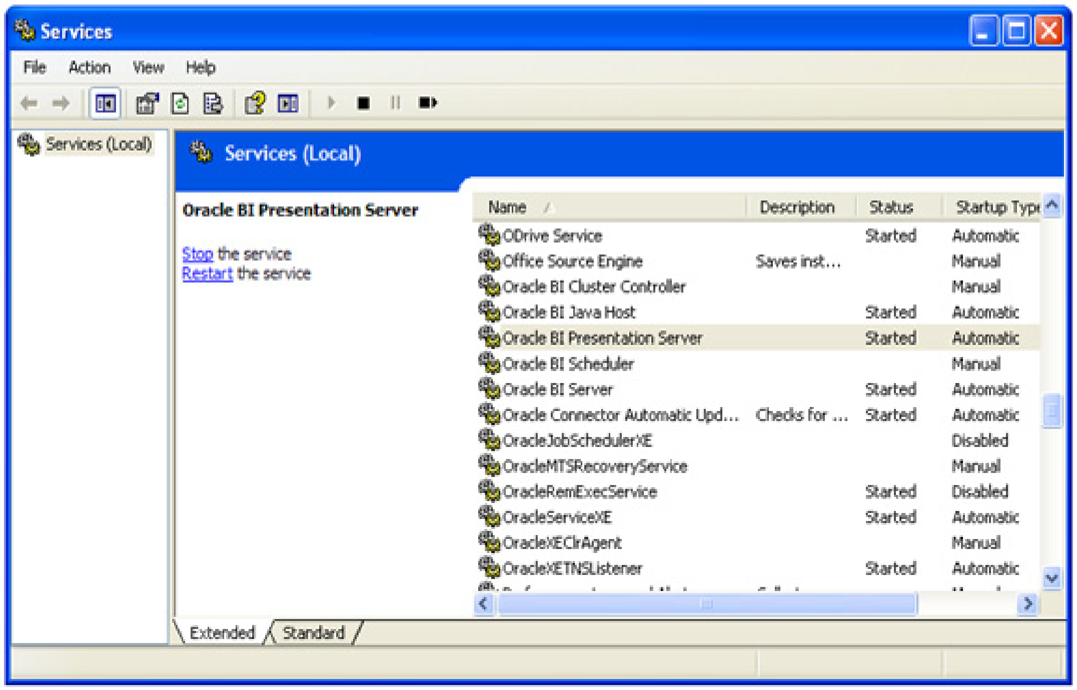
	Name	State	Health	Type	Deployment Order
☐	analytics	Active	✓ OK	Enterprise Application	100

14. Successful Deployment

STEP 5: STOP AND RESTART THE ORACLE BI SERVERS

This step requires you to start and restart the Oracle BI Server and the Oracle BI Presenting Server.

1. From the **Start** menu select **Control Panel>Administrative Tools>Services** to open the Services window.
2. Locate the **Oracle BI Server** and the **Oracle BI Presentation Server** in the list of services. They should be together in the list.



Oracle BI Presentation Server

Stop the service
Restart the service

Name	Description	Status	Startup Type
ODrive Service		Started	Automatic
Office Source Engine	Saves inst...		Manual
Oracle BI Cluster Controller			Manual
Oracle BI Java Host		Started	Automatic
Oracle BI Presentation Server		Started	Automatic
Oracle BI Scheduler			Manual
Oracle BI Server		Started	Automatic
Oracle Connector Automatic Upd...	Checks for ...	Started	Automatic
OracleJobSchedulerXE			Disabled
OracleMTSRecoveryService			Manual
OracleRemExecService		Started	Disabled
OracleServiceXE		Started	Automatic
OracleXEClrAgent			Manual
OracleXETNSListener		Started	Automatic

Figure 55: Locate Oracle BI Presentation Server and Oracle BI Server

3. Separately highlight the name of the server in the list and click the **Stop** link.
4. Separately highlight the name of the server in the list and click the **Restart** link.

STEP 6: PERFORM OBIEE REPOSITORY LINKAGE STEPS

Return to the section, *OBIEE Repository Linkage* on page 62, and complete the steps in that section.

Appendix A

Relationship Codes

Party-Level Codes

OII_ST.PARTY (Party)

- CA_DRV
- CLMT
- CLM_ADJ
- CLM_EXAMR
- CLM_MGR
- INSD
- INSUR_CO
- PAR_INSUR_CO
- PA_DRV
- PRODR
- REINS_CO
- SUB_PRODR
- SUPPLIER
- UW
- UW_ASST

Type-Level Codes

OII_ST.ORG (Organization)

- CLMT
- CLM_ADJ
- INSD
- INSUR_CO

- PAR_INSUR_CO
- PRODR
- REINS_CO
- SUB_PRODR
- SUPPLIER

OII_ST.PERS (Person)

- CA_DRV
- CLMT
- CLM_ADJ
- CLM_EXAMR
- CLM_MGR
- INSD
- PA_DRV
- PRODR
- SUB_PRODR
- SUPPLIER
- UW
- UW_ASST

ROLE-LEVEL CODES

OII_ST.INSUR_CO (Insurance Company)

- INSUR_CO
- PAR_INSUR_CO

OII_ST.PRODR (Producer)

- PRODR
- SUB_PRODR

OII_ST.UW (Underwriter)

- UW
- UW_ASST

CONTACT CODES

OII_ST.CNTCT (Contact)

- CA_VEH
- CLM
- CLMT
- CLM_ADJ
- CLM_EXAMR
- CLM_MGR
- INSD
- INSUR_CO
- PA_VEH
- PRODR
- PROSPECT
- REINS_CO
- SUPPLIER
- UW

OII_ST. PSTL_ADDR (Postal Address)

- CA_VEH
- CLM
- CLMT
- CLM_ADJ
- CLM_EXAMR
- CLM_MGR
- CP_BLDG
- DF_DWLNG
- HO_DWLNG
- INSD
- INSUR_CO
- PA_VEH
- PRODR
- PROSPECT
- REINS_CO
- SUPPLIER
- UW

INDEX