Oracle® Airlines Data Model

Implementation and Operations Guide

Release 12.2

E79018-02

December 2016

ORACLE®

Oracle Airlines Data Model Implementation and Operations Guide, Release 12.2

E79018-02

Contents

3 Access Layer Customization

4 ETL Implementation and Customization

5 Report and Query Customization

6 Oracle Airlines Data Model Security

A Sizing and Configuring an Oracle Airlines Data Model Warehouse

Index

List of Examples

List of Figures

x

List of Tables

Preface

Oracle Airlines Data Model Implementation and Operations Guide describes best practices for implementing a data warehouse based on Oracle Airlines Data Model.

This preface contains the following topics:

Audience

This guide is intended for business analysts, data modelers, data warehouse administrators, IT staff, and ETL developers who implement an Oracle Airlines Data Model warehouse.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Oracle Resources

Oracle provides many resources for you when implementing Oracle Airlines Data Model.

Oracle Airlines Data Model Documentation Set

For more information on Oracle Airlines Data Model, see the following documents:

- *Oracle Airlines Data Model Installation Guide*

- *Oracle Airlines Data Model Reference*

- *Oracle Airlines Data Model Release Notes*

Oracle Technology Network

Visit the Oracle Technology Network (OTN) to access to demos, whitepapers, Oracle By Example (OBE) tutorials, updated Oracle documentation, and other collateral.

Registering on OTN

You must register online before using OTN, Registration is free and can be done at

www.oracle.com/technetwork/index.html

Oracle Documentation

The Oracle Help Center provides access to Oracle documentation. For more information see the web site

docs.oracle.com

Oracle Learning Library on OTN

The Oracle Learning Library provides free online training content (OBEs, Demos and Tutorials). After you have a user name and password for OTN, you can go directly to the Oracle Learning Library Web site at

www.oracle.com/technetwork/tutorials/index.html

Then you can search for the tutorial or demo (within "All") by name.

Conventions

The following text conventions are used in this document:

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

1

Introduction to Oracle Airlines Data Model Customization

Provides an introduction to customizing the Oracle Airlines Data Model.

Overview of Oracle Airlines Data Model (page 1-1)
> Oracle Airlines Data Model is a pre-built approach to airline data warehousing enabling an airline to realize the power of insight quickly. Oracle Airlines Data Model reduces costs for both immediate and on-going operations by leveraging out-of-box Oracle based data warehouse and business intelligence solutions, making world-class database and business intelligence technology solutions available with an airline specific data model. You can use Oracle Airlines Data Model in any application environment.

Implementing an Oracle Airlines Data Model Warehouse (page 1-3)
> Although Oracle Airlines Data Model was designed following best practices for airlines, usually the model requires some customization to meet your business needs. The reasons that you might customize Oracle Airlines Data Model include: your business does not have a business area that is in the logical model of Oracle Airlines Data Model, or you must apply a new or different business rule.

Before You Begin Customizing the Oracle Airlines Data Model (page 1-4)
> Before you begin customizing Oracle Airlines Data Model, ensure that certain teams and committees exist.

Managing the Metadata for Oracle Airlines Data Model (page 1-7)
> Metadata is any data about data and is an important aspect of the data warehouse environment. Metadata allows the end user and the business analyst to navigate through the possibilities without knowing the context of the data or what the data represents.

Performing Fit-Gap Analysis for Oracle Airlines Data Model (page 1-10)
> Fit-gap analysis is where you compare your information needs and airline business requirements with the structure that is available with Oracle Airlines Data Model. You identify any required functionality that is not included in the logical model and the default schema, as well as other modifications that are necessary to meet your requirements. The result of your fit-gap analysis is a customization report which is a brief explanation of the adaptations and adjustments required to customize Oracle Airlines Data Model to fit your airline environment.

1.1 Overview of Oracle Airlines Data Model

Oracle Airlines Data Model is a pre-built approach to airline data warehousing enabling an airline to realize the power of insight quickly. Oracle Airlines Data Model

reduces costs for both immediate and on-going operations by leveraging out-of-box Oracle based data warehouse and business intelligence solutions, making world-class database and business intelligence technology solutions available with an airline specific data model. You can use Oracle Airlines Data Model in any application environment.

Using Oracle Airlines Data Model you can jump-start the design and implementation of an Oracle Airlines Data Model warehouse to quickly achieve a positive ROI for your data warehousing and business intelligence project with a predictable implementation effort.

Oracle Airlines Data Model provides much of the data modeling work that you must do for an airline business intelligence solution. The Oracle Airlines Data Model logical and physical data models were designed following best practices for airlines.

Components of the Oracle Airlines Data Model (page 1-2)
> Oracle Airlines Data Model includes several components.

Oracle Products That Make Up Oracle Airlines Data Model (page 1-3)
> Several Oracle technologies are involved in building the infrastructure for Oracle Airlines Data Model.

1.1.1 Components of the Oracle Airlines Data Model

Oracle Airlines Data Model includes several components.

- Logical model which is a third normal form (3NF) entity-object model. The logical model follows aviation industry standards and is described in *Oracle Airlines Data Model Reference*.

- Physical model defined as an Oracle Database schema.

- Intra-ETL database packages and SQL scripts to extract, transform, and load (ETL) data from the Oracle Airlines Data Model 3NF physical tables to the derived and aggregate tables in Oracle Airlines Data Model.

- Sample reports and dashboards developed using Oracle Business Intelligence Suite Extended Edition.

- DDL and installation scripts

Note:

When you use the Oracle Installer to install Oracle Airlines Data Model, you have the choice of performing two different types of installations:

- Installation of the Oracle Airlines Data Model component

- Installation of sample reports (and schemas)

Related Topics:

Oracle Airlines Data Model Reference

Oracle Airlines Data Model Installation Guide

1.1.2 Oracle Products That Make Up Oracle Airlines Data Model

Several Oracle technologies are involved in building the infrastructure for Oracle Airlines Data Model.

- Oracle Database with OLAP, Advanced Analytics and Partitioning Option (page 1-3)

- Oracle Development Tools (page 1-3)

- Oracle Business Intelligence Suite Extended Edition Presentation Tools (page 1-3)

Oracle Database with OLAP, Advanced Analytics and Partitioning Option

Oracle Airlines Data Model uses a complete Oracle technical stack. It leverages the following data warehousing features of the Oracle Database: SQL model, compression, partitioning, materialized views, data mining, and online analytical processing (OLAP).

Oracle Development Tools

You can use the following Oracle tools to customize the predefined physical models provided with Oracle Airlines Data Model, or to populate the target relational tables and materialized cube views.

Table 1-1   Oracle Development Tools Used with Oracle Airlines Data Model

Name	Use
SQL Developer or SQL*Plus	To modify, customize, and extend database objects
Analytic Workspace Manager	To view, create, develop, and manage OLAP dimensional objects.

Oracle Business Intelligence Suite Extended Edition Presentation Tools

Oracle Business Intelligence Suite Extended Edition is a comprehensive suite of enterprise BI products that delivers a full range of analysis and reporting capabilities. You can use Oracle Business Intelligence Suite Extended Edition Answers and Dashboard presentation tools to customize the predefined sample dashboard reports that are provided with Oracle Airlines Data Model.

Related Topics:

Reporting Approaches in Oracle Airlines Data Model (page 5-1)
Describes the methods to create reports from the data in an Oracle Airlines Data Model warehouse.

1.2 Implementing an Oracle Airlines Data Model Warehouse

Although Oracle Airlines Data Model was designed following best practices for airlines, usually the model requires some customization to meet your business needs. The reasons that you might customize Oracle Airlines Data Model include: your business does not have a business area that is in the logical model of Oracle Airlines Data Model, or you must apply a new or different business rule.

Typical physical model modifications include: adding, deleting, modifying, or renaming tables and columns; or altering foreign keys, constraints, or indexes.

To implement an Oracle Airlines Data Model warehouse, perform the following steps:

1. Perform the required organizational tasks.

2. Create a fit-gap analysis report.

3. In a development environment, install a copy of the Oracle Airlines Data Model.

4. Customize Oracle Airlines Data Model by making the changes you documented in the fit-gap analysis report. Make the changes in the following order:

   a. Foundation layer of the physical model and the ETL to populate the foundation layer.

   b. Access layer of the physical model and the ETL to populate the access layer.

5. In a test environment, make a copy of your customized version of Oracle Airlines Data Model. Then, following the documentation you created in Step 2, test the customized version of Oracle Airlines Data Model.

6. Following your typical procedures, roll the tested customized version of Oracle Airlines Data Model out into pre-production and, then, production.

   Tip:

   Keep 'clean' copies of the components delivered with Oracle Airlines Data Model components. This is important when upgrading to later versions of Oracle Airlines Data Model.

   See Also:

   - For more information, see "Before You Begin Customizing the Oracle Airlines Data Model (page 1-4)"

   - For more information, see "Performing Fit-Gap Analysis for Oracle Airlines Data Model (page 1-10)"

   - When customizing the physical objects, follow the guidelines in, "Foundation Layer Customization (page 2-7)"

   - When writing the ETL, follow the guidelines in "ETL for the Foundation Layer of an Oracle Airlines Data Model Warehouse (page 4-2)"

   - When designing the physical objects, follow the guidelines in "Access Layer Customization (page 3-1)"

   - When writing the ETL, follow the guidelines in "Customizing Intra-ETL for Oracle Airlines Data Model (page 4-10)"

1.3 Before You Begin Customizing the Oracle Airlines Data Model

Before you begin customizing Oracle Airlines Data Model, ensure that certain teams and committees exist.

- Data warehouse governance steering committee.

- Implementation team.

- Fit-gap analysis team. This team consists of business analysts who can identify the business requirements and scope of the Oracle Airlines Data Model and at least some engineers in the Implementation team. Business members of this team must understand logical data modeling so that they can evaluate what changes must be made to the foundation and access layers of the physical model.

After these teams and committees are formed:

- Discuss the approach and determine the involvement and roles of every party involved in the customization (for example, business and IT).

- Agree on the scope of the project (that is, agree on what new data must be in the data warehouse and why it is needed).

- Agree on the timing and the working arrangements.

> Prerequisite Knowledge for Implementors (page 1-6)
> > The Oracle Airlines Data Model uses much of the Oracle stack. Consequently, to successfully implement the Oracle Airlines Data Model, the implementation team needs certain prerequisite knowledge.

> Responsibilities of a Data Warehouse Governance Committee (page 1-6)
> > Governance is of concern to any enterprise, executive team or individual with an interest in the processes, standards, and compliance. It is even more important to organizations that have invested in data warehousing. Data warehouse governance occurs within the context of overall IT governance. It provides the necessary policies, process and procedures, which must be clearly communicated to the entire corporation, from the IT employees to the front-end operational personnel.

Related Topics:

> Responsibilities of a Data Warehouse Governance Committee (page 1-6)
> > Governance is of concern to any enterprise, executive team or individual with an interest in the processes, standards, and compliance. It is even more important to organizations that have invested in data warehousing. Data warehouse governance occurs within the context of overall IT governance. It provides the necessary policies, process and procedures, which must be clearly communicated to the entire corporation, from the IT employees to the front-end operational personnel.

> Prerequisite Knowledge for Implementors (page 1-6)
> > The Oracle Airlines Data Model uses much of the Oracle stack. Consequently, to successfully implement the Oracle Airlines Data Model, the implementation team needs certain prerequisite knowledge.

> Implementing an Oracle Airlines Data Model Warehouse (page 1-3)
> > Although Oracle Airlines Data Model was designed following best practices for airlines, usually the model requires some customization to meet your business needs. The reasons that you might customize Oracle Airlines Data Model include: your business does not have a business area that is in the logical model of Oracle Airlines Data Model, or you must apply a new or different business rule.

Fit-gap analysis is where you compare your information needs and airline business requirements with the structure that is available with Oracle Airlines Data Model. You identify any required functionality that is not included in the logical model and the default schema, as well as other modifications that are necessary to meet your requirements. The result of your fit-gap analysis is a customization report which is a brief explanation of the adaptations and adjustments required to customize Oracle Airlines Data Model to fit your airline environment.

1.3.1 Prerequisite Knowledge for Implementors

The Oracle Airlines Data Model uses much of the Oracle stack. Consequently, to successfully implement the Oracle Airlines Data Model, the implementation team needs certain prerequisite knowledge.

- Experience performing information and data analysis and data modeling. (Experience using Oracle SQL Data Modeler, is a plus.)

- An understanding of the Oracle technology stack, especially data warehouse (Database, Data Warehouse, OLAP, Data Mining, Oracle Business Intelligence Suite Extended Edition)

- Hands-on experience using:

  - Oracle Database

  - PL/SQL

  - SQL DDL and DML syntax

  - Analytic Workspace Manager

  - Oracle SQL Developer

  - Oracle Business Intelligence Suite Extended Edition Administrator, Answers, and Dashboards

1.3.2 Responsibilities of a Data Warehouse Governance Committee

Governance is of concern to any enterprise, executive team or individual with an interest in the processes, standards, and compliance. It is even more important to organizations that have invested in data warehousing. Data warehouse governance occurs within the context of overall IT governance. It provides the necessary policies, process and procedures, which must be clearly communicated to the entire corporation, from the IT employees to the front-end operational personnel.

Before you customize Oracle Airlines Data Model, setup a data warehouse governance steering committee if one does not exist. The role of this steering committee is to oversee the data warehouse to provide an environment that reaches across the enterprise and drives the best business value.

Data Warehouse Governance Committee: Overall Responsibilities

The data warehouse governance steering committee sets direction and response for the governance framework and covers the follow areas:

- The entire data warehouse life cycle.

- Agree on the data to process and make available to end-users.

- Determine what is the minimum quality criteria for the data that is available to end users and determine how to measure and analyze these criteria against the quality of the data that is the source data for the data warehouse.

- The business goals of the organization to apply core information from data warehouse.

- The policies, procedures and standards for data resource and data access.

- The life cycle of data warehouse component management.

Data Warehouse Governance Committee: Data Governance Responsibilities

The more detailed focus in data warehouse governance is data governance. Data governance tasks include:

- Approving the data modeling standards, metadata standards and other related standards. This includes determining a metadata strategy and identifying the data modeling tools to use that support these standards.

- Determining the data retention policy.

- Designing a data access policy based on legal restrictions and data security rules.

- Designing a data backup strategy that aligns with the impact analysis to the business unit.

- Monitoring and reporting on data usage, activity, and alerts.

Related Topics:

Managing the Metadata for Oracle Airlines Data Model (page 1-7)
> Metadata is any data about data and is an important aspect of the data warehouse environment. Metadata allows the end user and the business analyst to navigate through the possibilities without knowing the context of the data or what the data represents.

1.4 Managing the Metadata for Oracle Airlines Data Model

Metadata is any data about data and is an important aspect of the data warehouse environment. Metadata allows the end user and the business analyst to navigate through the possibilities without knowing the context of the data or what the data represents.

Metadata management is a comprehensive, ongoing process of overseeing and actively managing metadata in a central environment which helps an enterprise to identify how data is constructed, what data exists, and what the data means. It is particularly helpful to have good metadata management when customizing Oracle Airlines Data Model so that you can do impact analysis to ensure that changes do not adversely impact data integrity anywhere in your data warehouse.

Metadata Categories and Standards (page 1-8)
> Metadata is organized into three major categories: Business metadata, Technical metadata, and Process Execution metadata.

Working with a Metadata Repository (page 1-8)
> You manage metadata using a Metadata Repository. At the highest level, a Metadata Repository includes three layers of information.

To customize the Oracle Airlines Data Model, you must understand the dependencies among components, especially how the report KPIs are mapped to the physical tables and columns.

1.4.1 Metadata Categories and Standards

Metadata is organized into three major categories: Business metadata, Technical metadata, and Process Execution metadata.

- **Business metadata** describes the meaning of data in a business sense. The business interpretation of data elements in the data warehouse is based on the actual table and column names in the database. Gather this mapping information and business definition and rules information into business metadata.

- **Technical metadata** represents the technical aspects of data, including attributes such as data types, lengths, lineage, results from data profiling, and so on.

- **Process execution metadata** presents statistics on the results of running the ETL process itself, including measures such as rows loaded successfully, rows rejected, amount of time to load, and so on.

Because metadata is so important in information management, many organizations attempt to standardize metadata at various levels, such as:

- Metadata Encoding and Transmission Standard (METS): A standard for encoding descriptive, administrative, and structural metadata regarding objects within a digital library.

- American National Standards Institute (ANSI): The organization that coordinates the US voluntary standardization and conformity-assessment systems.

- International Organization for Standardization (ISO): The body that establishes, develops, and promotes standards for international exchange.

- Common Warehouse Metamodel (CWM): A specification, released and owned by the Object Management Group, for modeling metadata for relational, non-relational, multi-dimensional, and most other objects found in a data warehousing environment.

When you implement your metadata management solution, reference your data warehouse infrastructure environment and make the decision about which standard to follow.

1.4.2 Working with a Metadata Repository

You manage metadata using a Metadata Repository. At the highest level, a Metadata Repository includes three layers of information.

The layers are defined in the following order:

1. A Physical layer: This metadata layer identifies the source data.

2. A Business Model and Mapping layer: This metadata layer organizes the physical layer into logical categories and records the appropriate metadata for access to the source data.

3. The Presentation layer: This metadata layer exposes the business model entities for end-user access.

The first step in creating a Metadata Repository is to scope your metadata management needs by:

- Identifying the metadata consumers. Typically, there are business consumers and technical consumers.

- Determine the business and technical metadata requirements.

- Aligning metadata requirements to specific data elements and logical data flows.

Then:

- Decide how important each part is.

- Assign responsibility to someone for each piece.

- Decide what constitutes a consistent and working set of metadata

- Where to store, backup, and recover the metadata.

- Ensure that each piece of metadata is available only to those people who need it.

- Quality-assure the metadata and ensure that it is complete and up to date.

- Identify the Metadata Repository to use and how to control that repository from one place

After creating the metadata definitions, review your data architecture to ensure you can acquire, integrate, and maintain the metadata.

As the data keeps on changing in your data warehouse day by day, update the Metadata Repository. When you want to change business rules, definitions, formulas or process (especially when customizing the Oracle Airlines Data Model), your first step is to survey the metadata and do an impact analysis to list all of the attributes in the data warehouse environment that is affected by a proposed change.

1.4.3 Browsing the Metadata Repository Supplied With Oracle Airlines Data Model

To customize the Oracle Airlines Data Model, you must understand the dependencies among components, especially how the report KPIs are mapped to the physical tables and columns.

Oracle Airlines Data Model provides a tool called the "OADM Metadata" browser that helps you discover these dependencies. When you install Oracle Airlines Data Model with its sample reports, the metadata browser is delivered as a sample Dashboard in the webcat.

To browse the metadata repository:

1. In the browser, open the login page at `http://servername:9704/analytics` where `servername` is the server on which the webcat is installed.

2. Log in with username of `oadm`, and provide the password.

3. Select the Metadata Browser dashboard.

4. Use the tabs in the Metadata browser to explore the metadata:

- **Measure-Entity tab**

  On the Measure-Entity tab you can see the business areas (relational, OLAP, mining), the measures description, corresponding formula, responsible entities, and attributes for the measure.

  To browse the data, select the business area and measure description that you are interested in.

- **Entity-Measure tab**

  Using the Entity-Measure tab, you can discover the mappings between entities, attributes, supported measures, and calculations of the measures. You can discover information about particular entities and attributes.

  For example, perform the following steps to learn more about an entity:

  1. Select the entity.

  2. Click **GO**.

- **Program-Table tab**

  Using the Program-Table tab you can browse for information on the intra-ETL mappings and report information. Perform the following steps:

  1. Select the program type (that is, intra-ETL or report) and program name.

  2. Select **GO**.

- **Table-Program tab**

  By default when you go to the Table-Program tab you see all of the tables used for all the reports.

  To discover what reports use a particular table, you must move a particular table from the right pane to the left (Selected) pane.

  For example, to see the reports that use a particular table, perform the following steps:

  1. In the right pane of the Table-Program tab, select the table.

  2. Move the table to the Selected list on the left by clicking on **<** (left arrow).

  3. Click **OK**.

  4. Select **GO**.

  The reports for the selected table are displayed.

Related Topics:

Oracle Airlines Data Model Installation Guide

1.5 Performing Fit-Gap Analysis for Oracle Airlines Data Model

Fit-gap analysis is where you compare your information needs and airline business requirements with the structure that is available with Oracle Airlines Data Model. You identify any required functionality that is not included in the logical model and the default schema, as well as other modifications that are necessary to meet your requirements. The result of your fit-gap analysis is a customization report which is a brief explanation of the adaptations and adjustments required to customize Oracle Airlines Data Model to fit your airline environment.

The fit-gap analysis team writes the customization report by performing the following steps:

1. If you have performed previous evaluations, review the documentation from the previous phases, and if necessary add team members with the required business and technical expertise.

2. Review the data and map your logical entities and data structure with the Oracle Airlines Data Model logical model and schema:

   - Starting from business requirements, questions, and rules, identify any entities and attributes that are *not* in the Oracle Airlines Data Model.

   - Compare the Oracle Airlines Data Model to your existing application model if have one.

   - Compare the Oracle Airlines Data Model to the OLTP data that you are using as a data source to the Oracle Airlines Data Model warehouse.

3. Determine the differences between your needs and Oracle Airlines Data Model schema. To help you with this task, produce a list of actions people may take with the system (examples rather than models), and create use cases for appraising the functionality of the Oracle Airlines Data Model Warehouse. Answer the following questions about the differences you find:

   - What differences you can live with, and what must be reconciled?

   - What can you do about the differences you cannot live with?

4. Identify the changes you must make to the default design of Oracle Airlines Data Model to create the customized warehouse. Identify these changes in the following order:

   a. Physical model. Follow the guidelines outlined in Physical Model Customization (page 2-1).

   b. ETL mapping. Follow the guidelines outlined in ETL Implementation and Customization (page 4-1) to identify and design the source-ETL that you must create and to identify and make any changes to the intra-ETL provided with Oracle Airlines Data Model.

      Tip:

      When identifying changes, ensure that the changes meet your security and metadata requirements.

5. Write the customization report, detailing what changes are required to make the Oracle Airlines Data Model match your business needs. This includes any additions and changes to interfaces to existing systems.

6. Based on the customization report, update the Project Plan and perform the steps outlined in "Implementing an Oracle Airlines Data Model Warehouse (page 1-3)".

Related Topics:

Physical Model Customization (page 2-1)
> Provides general information about customizing the physical model of Oracle Airlines Data Model and more detailed information about customizing the foundation layer of the physical model.

ETL Implementation and Customization (page 4-1)

Implementing an Oracle Airlines Data Model Warehouse (page 1-3)

Although Oracle Airlines Data Model was designed following best practices for airlines, usually the model requires some customization to meet your business needs. The reasons that you might customize Oracle Airlines Data Model include: your business does not have a business area that is in the logical model of Oracle Airlines Data Model, or you must apply a new or different business rule.

2

Physical Model Customization

Provides general information about customizing the physical model of Oracle Airlines Data Model and more detailed information about customizing the foundation layer of the physical model.

Characteristics of the Default Physical Model (page 2-1)
> The default physical data model of Oracle Airlines Data Model defines tables, KPIs, data mining models, and OLAP cubes.

Customizing the Oracle Airlines Data Model Physical Model (page 2-4)
> The starting point for the Oracle Airlines Data Model physical data model is the 3NF logical data model. The physical data model mirrors the logical model as much as possible, (although some changes in the structure of the tables or columns may be necessary) and defines database objects (such as tables, cubes, and views).

Foundation Layer Customization (page 2-7)
> The first step in customizing the physical model of Oracle Airlines Data Model is customizing the foundation layer of the physical data model. The foundation layer of the physical model mirrors the 3NF logical model of Oracle Airlines Data Model, so you might choose to customize the foundation layer to reflect differences between your logical model needs and the default logical model of Oracle Airlines Data Model. Additionally, you might need to customize the physical objects in the foundation layer to improve performance (for example, you might choose to compress some foundation layer tables).

General Recommendations When Designing Physical Structures (page 2-10)
> The `oadm_sys` schema delivered with Oracle Airlines Data Model was designed following the best practices for data access and performance. Continue to use these practices when you add new physical objects.

Related Topics:

Access Layer Customization (page 3-1)
> Provides information about customizing the access layer of Oracle Airlines Data Model.

2.1 Characteristics of the Default Physical Model

The default physical data model of Oracle Airlines Data Model defines tables, KPIs, data mining models, and OLAP cubes.

370 tables and 8,500 columns
220 industry-specific measures and KPIs
6 pre-built data mining models
7 pre-built OLAP cubes

The default physical model of the Oracle Airlines Data Model shares characteristics of a multischema "traditional" data warehouse, as described in "Layers in a "Traditional" Data Warehouse (page 2-2)", but defines all data structures in a single schema as described in "Layers in the Default Oracle Airlines Data Model Warehouse (page 2-2)".

Layers in a "Traditional" Data Warehouse

Historically, the following layers are defined for a data warehouse environment:

- **Staging layer:** This layer is used when moving data from the OLTP system and other data sources into the data warehouse itself. It consists of temporary loading structures and rejected data. Having a staging layer enables the speedy extraction, transformation and loading (ETL) of data from your operational systems into data warehouse without disturbing any of the business users. It is in this layer that much of the complex data transformation and data quality processing occurs. The most basic approach for the design of the staging layer is using a schema identical to the one that exists in the source operational system.

  > **Note:**
  >
  > In some implementations this layer is not necessary, because all data transformation processing is done "on the fly" as data is extracted from the source system before it is inserted directly into the foundation layer.

- **Foundation or integration layer:** This layer is traditionally implemented as a Third Normal Form (3NF) schema. A 3NF schema is a neutral schema design independent of any application, and typically has many tables. It preserves a detailed record of each transaction without any data redundancy and allows for rich encoding of attributes and all relationships between data elements. Users typically require a solid understanding of the data to navigate the more elaborate structure reliably. In this layer data begins to take shape and it is not uncommon to have some end-user application access data from this layer especially if they are time sensitive, as data becomes available here before it is transformed into the Access and Performance layer.

- **Access layer:** This layer is traditionally defined as a snowflake or star schema that describes a "flattened" or a dimensional view of the data.

Layers in the Default Oracle Airlines Data Model Warehouse

Oracle Airlines Data Model warehouse environment also consists of three layers. However, as indicated by the dotted line in Figure 2-1 (page 2-3), in the Oracle Airlines Data Model the definitions of the foundation and access layers are combined in a single schema.

Figure 2-1   Layers in an Oracle Airlines Data Model Warehouse
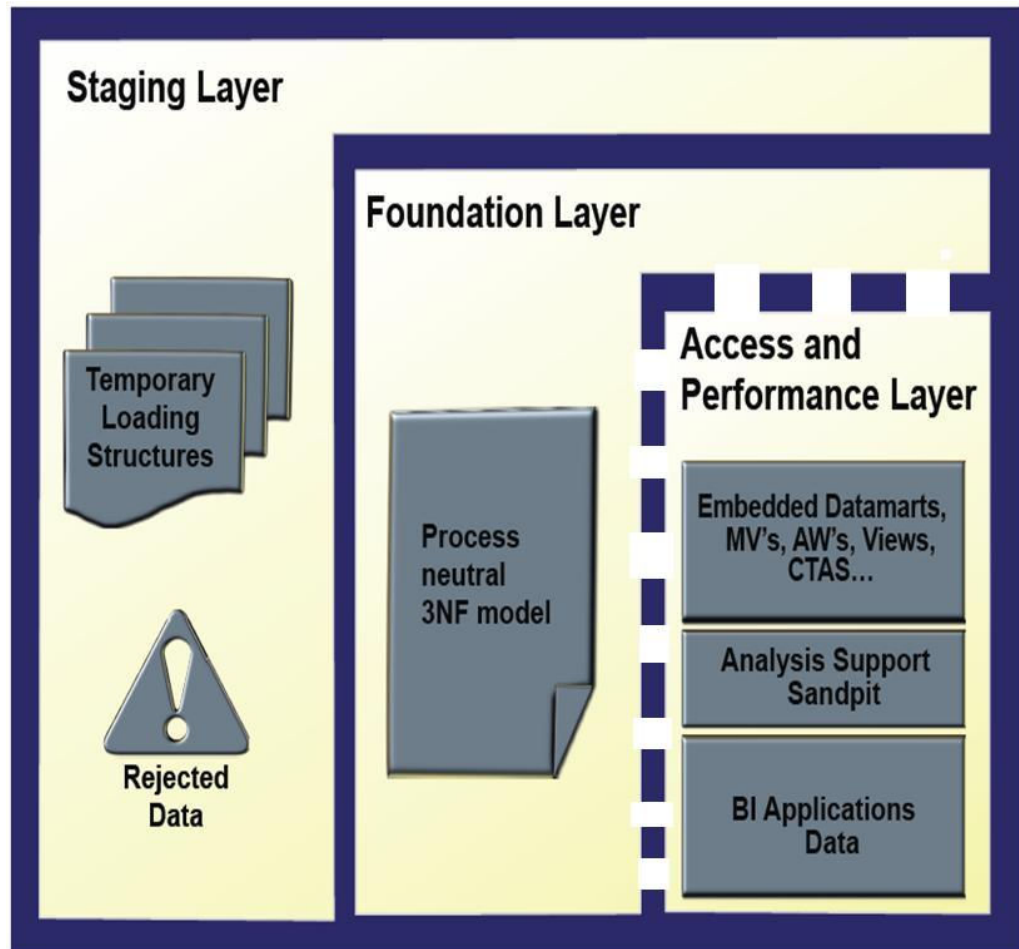
The layers in the Oracle Airlines Data Model warehouse are:

- **Staging layer**. As in a "traditional" data warehouse environment, an Oracle Airlines Data Model warehouse environment can have a staging layer. Because the definition of this layer varies by customer, a definition of this area is not provided as part of Oracle Airlines Data Model.

- **Foundation and Access layers.** The physical objects for these layers are defined in a single schema, the oadm_sys schema:

  - **Foundation layer.** The foundation layer of the Oracle Airlines Data Model is defined by base tables that present the data in 3NF (that is, tables that have the DWB_ prefix). This layer also includes reference, lookup, and control tables defined in the oadm_sys schema (that is, the tables that have the DWR_ , DWL_, DWC_ prefixes).

  - **Access layer.** The access layer of Oracle Airlines Data Model is defined by derived and aggregate tables (defined with DWD_ and DWA_ prefixes), cubes (defined with a CB$ prefix), and views (that is, views defined with the _VIEW suffix). These structures provide a summarized or "flattened" perspectives of the data in the foundation layer.

    This layer also contains the results of the data mining models which are stored in derived (DWD_) tables.

Related Topics:

Oracle Airlines Data Model Reference

2.2 Customizing the Oracle Airlines Data Model Physical Model

The starting point for the Oracle Airlines Data Model physical data model is the 3NF logical data model. The physical data model mirrors the logical model as much as possible, (although some changes in the structure of the tables or columns may be necessary) and defines database objects (such as tables, cubes, and views).

To customize the default physical model of the Oracle Airlines Data Model perform the following steps:

1. Answer the physical customization questions.

2. Familiarize yourself with the characteristics of the logical and physical model of Oracle Airlines Data Model

3. Modify the foundation level of your physical model of Oracle Airlines Data Model, as needed.

   When defining physical structures:

   • Keep the foundation layer in 3NF form.

   • Follow the conventions used when creating the default physical model of Oracle Airlines Data Model.

   > **Tip:**
   >
   > Package the changes you make to the physical data model as a patch to the `oadm_sys` schema.

4. Modify the access layer of your physical model of Oracle Airlines Data Model.

   Questions to Answer Before You Customize the Physical Model (page 2-5)
     When designing the physical model, remember that the logical data model is not one-to-one with the physical data model. Consider the load, query, and maintenance requirements when you convert the logical data model into the physical layer. For example, answer the following questions before you design the physical data model:

   Conventions When Customizing the Physical Model (page 2-5)
     When developing the physical model for Oracle Airlines Data Model, the conventions outlined below were followed. Continue to follow these conventions as you customize the physical model.

Related Topics:

Questions to Answer Before You Customize the Physical Model (page 2-5)
  When designing the physical model, remember that the logical data model is not one-to-one with the physical data model. Consider the load, query, and maintenance requirements when you convert the logical data model into the physical layer. For example, answer the following questions before you design the physical data model:

Characteristics of the Default Physical Model (page 2-1)
  The default physical data model of Oracle Airlines Data Model defines tables, KPIs, data mining models, and OLAP cubes.

Common Change Scenarios When Customizing the Foundation Layer of Oracle Airlines Data Model (page 2-8)

When changing the foundation layer objects to reflect your logical model design, make as few changes as possible.

General Recommendations When Designing Physical Structures (page 2-10)

The `oadm_sys` schema delivered with Oracle Airlines Data Model was designed following the best practices for data access and performance. Continue to use these practices when you add new physical objects.

Conventions When Customizing the Physical Model (page 2-5)

When developing the physical model for Oracle Airlines Data Model, the conventions outlined below were followed. Continue to follow these conventions as you customize the physical model.

Access Layer Customization (page 3-1)

Provides information about customizing the access layer of Oracle Airlines Data Model.

2.2.1 Questions to Answer Before You Customize the Physical Model

When designing the physical model, remember that the logical data model is not one-to-one with the physical data model. Consider the load, query, and maintenance requirements when you convert the logical data model into the physical layer. For example, answer the following questions before you design the physical data model:

- Do you need the physical data model to cover the full scope of the logical data model, or only part of the scope?

  "Common Change Scenarios When Customizing the Foundation Layer of Oracle Airlines Data Model (page 2-8)" provides an overview discussion of making physical data model changes when your business needs do not result in a logical model that is the same as the Oracle Airlines Data Model logical model.

- What is the result of the source data profile?

- What is the data load frequency for each table?

- How many large tables are there and which tables are these?

- How will the tables and columns be accessed? What are the common joins?

- What is your data backup strategy?

2.2.2 Conventions When Customizing the Physical Model

When developing the physical model for Oracle Airlines Data Model, the conventions outlined below were followed. Continue to follow these conventions as you customize the physical model.

General Naming Conventions for Physical Objects (page 2-6)

Describes the general naming guidelines for naming physical objects that you define in the data model.

Domain Definition Standards (page 2-7)

A domain is a set of values allowed for a column. The domain can be enforced by a foreign key, check constraints, or the application on top of the database.

2.2.2.1 General Naming Conventions for Physical Objects

Describes the general naming guidelines for naming physical objects that you define in the data model.

When naming the physical objects follow the naming guidelines for naming objects within an Oracle Database schema. For example:

- Table and column names must start with a letter, can use only 30 alphanumeric characters or less, cannot contain spaces or some special characters such as "!" and cannot use reserved words.

- Table names must be unique within a schema that is shared with views and synonyms.

- Column names must be unique within a table.

- Although it is common to use abbreviations in the physical modeling stage, as much as possible, use names for the physical objects that correspond to the names of the entities in the logical model. Use consistent abbreviations to avoid programmer and user confusion.

- When naming columns, use short names if possible. Short column names reduce the time required for SQL command parsing.

The `oadm_sys` schema delivered with Oracle Airlines Data Model uses the prefixes and suffixes shown in the following table to identify object types. Use these prefixes and suffixes for any new tables, views, and cubes that you define.

Table 2-1   Table and View Naming Conventions

Prefix or Suffix	Used for Name of These Objects
CB$	Materialized view of an OLAP cube. This materialized view is automatically created by the OLAP server. **Note:** Do not report or query against this object. Instead access the corresponding `_VIEW` object.
DMV_	Materialized views used for as the source data of data mining model.
DWA_	Aggregate tables.
DWB_	Base transaction data (3NF) tables.
DWC_	Control tables.
DWD_	Derived tables -- including data mining result tables.
DWL_	Lookup tables.
DWV_	Relational view of time dimension.
DWR_	Reference data tables used as dimension tables in a foundation layer fact table (that is, for a `DWB_` table).
_VIEW	Relational views of OLAP cubes, dimensions, or hierarchies.

Related Topics:

Oracle Airlines Data Model Reference

2.2.2.2 Domain Definition Standards

A domain is a set of values allowed for a column. The domain can be enforced by a foreign key, check constraints, or the application on top of the database.

Define the standards for each domain across the model such as:

- Date and time type, such as `'YYYY-MM-DD'`.

- Numeric value in different situations.

- Character string length in different situations.

- Coded value definition such as key or description.

2.3 Foundation Layer Customization

The first step in customizing the physical model of Oracle Airlines Data Model is customizing the foundation layer of the physical data model. The foundation layer of the physical model mirrors the 3NF logical model of Oracle Airlines Data Model, so you might choose to customize the foundation layer to reflect differences between your logical model needs and the default logical model of Oracle Airlines Data Model. Additionally, you might need to customize the physical objects in the foundation layer to improve performance (for example, you might choose to compress some foundation layer tables).

When making changes to the foundation layer, keep the following points in mind:

- When changing the foundation layer objects to reflect your logical model design, make as few changes as possible.

- When defining new foundation layer objects or when redesigning existing foundation layer objects for improved performance, follow the general recommendations and adhere to the physical model customization conventions..

- Remember that changes to the foundation layer objects can also impact the access layer objects.

Note:

Approach any attempt to change the Oracle Airlines Data Model with caution. The foundation layer of the physical model of the Oracle Airlines Data Model has (at its core) a set of generic structures that allow it to be flexible and extensible. Before making extensive additions, deletions, or changes, ensure that you understand the full range of capabilities of Oracle Airlines Data Model and that you cannot handle your requirements using the default objects in the foundation layer

---

Common Change Scenarios When Customizing the Foundation Layer of Oracle Airlines Data Model (page 2-8)

When changing the foundation layer objects to reflect your logical model design, make as few changes as possible.

Example: Changing the Foundation Layer of the Oracle Airlines Data Model (page 2-9)

> When defining new foundation layer objects or when redesigning existing foundation layer objects for improved performance, follow the general recommendations.

Related Topics:

Common Change Scenarios When Customizing the Foundation Layer of Oracle Airlines Data Model (page 2-8)

> When changing the foundation layer objects to reflect your logical model design, make as few changes as possible.

General Recommendations When Designing Physical Structures (page 2-10)

> The `oadm_sys` schema delivered with Oracle Airlines Data Model was designed following the best practices for data access and performance. Continue to use these practices when you add new physical objects.

Conventions When Customizing the Physical Model (page 2-5)

> When developing the physical model for Oracle Airlines Data Model, the conventions outlined below were followed. Continue to follow these conventions as you customize the physical model.

2.3.1 Common Change Scenarios When Customizing the Foundation Layer of Oracle Airlines Data Model

When changing the foundation layer objects to reflect your logical model design, make as few changes as possible.

There are several common change scenarios when customizing the foundation layer of the physical data model:

- **Additions to Existing Structures**

  If you identify business areas or processes that are not supported in the default foundation layer of the physical data model of Oracle Airlines Data Model, add new tables and columns.

  Carefully study the default foundation layer of the physical data model of Oracle Airlines Data Model (and the underlying logical data model) to avoid building redundant structures when making additions. If these additions add high value to your business value, communicate the additions back to the Oracle Airlines Data Model Development Team for possible inclusion in future releases of Oracle Airlines Data Model.

- **Deletions of Existing Structures**

  If there are areas of the model that cannot be matched to any of the business requirements of your legacy systems, it is safer to keep these structures and not populate that part of the warehouse.

  Deleting a table in the foundation layer of the physical data model can destroy relationships needed in other parts of the model or by applications based on it. Some tables may not be needed during the initial implementation, however you can use these structures at a later time. If this is a possibility, keeping the structures now saves re-work later. If tables are deleted, perform a thorough analysis to identify all relationships originating from that entity.

- **Changes to Existing Structures**

In some situations some structures in the foundation layer of the physical data model of the Oracle Airlines Data Model may not exactly match the corresponding structures that you use before implementing changes, identify the impact that the changes would have on the database design of Oracle Airlines Data Model. Also identify the impact on any applications based on the new design.

Related Topics:

When defining new foundation layer objects or when redesigning existing foundation layer objects for improved performance, follow the general recommendations.

2.3.2 Example: Changing the Foundation Layer of the Oracle Airlines Data Model

When defining new foundation layer objects or when redesigning existing foundation layer objects for improved performance, follow the general recommendations.

As an example, take a look at how Oracle Airlines Data Model supports the various airlines services, what a sample customer might discover during fit-gap analysis, and how that customer extends Oracle Airlines Data Model to fit the discovered gaps.

Entities supporting airline services

The entities provided with the logical model of Oracle Airlines Data Model that support the airline services are:

- AIRPORT: Airport means an IATA recognized location that serves as an origin or destination of one or more flights. This entity stores the details of the airport like city, country, region of the airport.

- FLIGHT: This entity stores information about the flight (for example, the carrier to which it belongs, and the flight number).

- AIRCRAFT: This entity stores the type of aircraft (for example. Boeing 737).

- AIRCRAFT VERSION: This entity stores the details of the aircraft version used for flights. For example, if the aircraft type is Boeing 737 then the version will be Boeing 737-800.

- CARRIER: This entity stores the details about the carrier (for example, carrier code and description).

- CARRIER TYPE: This entity stores Carrier type details (for example, airline, railway, on road transport, and ship).

The differences discovered during fit-gap analysis

Assume that during the fit-gap analysis, you discover the following needs that are not supported by the logical model delivered with Oracle Airlines Data Model:

- Your company serves more than one service in airline arena. You discover that you track not only airports as represented in the standard logical model of Oracle Airlines Data Model, but also you need to define airports by categories of airport activities (that is, airport categories). These airport categories are, for example, commercial service, primary, cargo service, reliever, and general aviation airports.

- In the flight operations, the airline needs to maintain a history of airport categories used to operate the various lines of services (for example, commercial, cargo, reliever or general aviation).

Extending the logical and physical model to support the differences

To support the differences, extend the logical and physical model in the following ways:

- To store details about category of the airport you must modify the logical model. The classic way to do this is to add another entity to hold this information. For example, you can add an entity named AIRPORT CATEGORY that has a 1:M relationship with the AIRPORT entity. Then, extend the physical data model in a corresponding manner. The specific steps to extend the physical model are given in Example 2-1 (page 2-10).

- To maintain the history of flight operations which are operated from respective airport categories, you can find the same based on the existing relationship between flight schedule and AIRPORT entity for corresponding Leg entity.

Example 2-1   Extending the Physical Data Model of Oracle Airlines Data Model to Support Multiple Categories for AIRPORT

To extend the design of the foundation layer of Oracle Airlines Data Model to support multiple categories for AIRPORT, perform the following steps:

1. Create a table named DWL_ARPRT_CTGRS to hold the multiple categories information for airport by executing the following statements:

```
CREATE TABLE DWL_ARPRT_CTGRS
(
    ARPRT_CTGRS_ID          INTEGER NOT NULL ,
    ARPRT_CTGRS_NM          VARCHAR2(50) NULL ,
    ARPRT_CTGRS_DESC        VARCHAR2(500) NULL ,
    ARPRT_CTGRS_CD          CHAR(18) NULL ,
    DWFEED_ID               INTEGER NULL ,
    SRC_SYS_ID              VARCHAR2(30) NULL ,
    SRC_SYS_CRTD_TMSTMP     TIMESTAMP NULL ,
    SRC_SYS_UPD_TMSTMP      TIMESTAMP NULL ,
    SRC_SYS_DEL_IND         VARCHAR2(1) NULL
);


ALTER TABLE DWL_ARPRT_CTGRS   ADD CONSTRAINT  XPKARPRT_CTGRS PRIMARY KEY
(ARPRT_CTGRS_ID);
```

2. Add a column named ARPRT_CTGRS_ID to the DWR_AIP table by issuing the following statement:

```
ALTER TABLE DWR_AIP ADD COLUMN ARPRT_CTGRS_ID INTEGER NULL;
```

2.4 General Recommendations When Designing Physical Structures

The oadm_sys schema delivered with Oracle Airlines Data Model was designed following the best practices for data access and performance. Continue to use these practices when you add new physical objects.

This section provides information about the important physical design aspects in the default Oracle Airlines Data Model.

Tablespaces in Oracle Airlines Data Model (page 2-12)
> A tablespace consists of one or more data files, which are physical structures within the operating system you are using.

Data Compression in Oracle Airlines Data Model (page 2-12)
> A key decision that you must make is whether to compress your data. Using table compression reduces disk and memory usage, often resulting in a better scale-up performance for read-only operations. Table compression can also speed up query execution by minimizing the number of round trips required to retrieve data from the disks. Compressing data however imposes a performance penalty on the load speed of the data. Most of the base tables in the Oracle Airlines Data Model are compressed tables.

Tables for Supertype and Subtype Entities in Oracle Airlines Data Model (page 2-14)
> A supertype is a generic entity type that has a relationship with one or more subtypes. A subtype is a sub-grouping of the entities in an entity type that is meaningful to the organization and that shares common attributes or relationships distinct from other subgroups.

Surrogate Keys in the Physical Model (page 2-15)
> The surrogate key method for primary key construction involves taking the natural key components from the source systems and mapping them through a process of assigning a unique key value to each unique combination of natural key components (including source system identifier). The resulting primary key value is completely non-intelligent and is typically a numeric data type for maximum performance and storage efficiency.

Integrity Constraints in Oracle Airlines Data Model (page 2-15)
> Integrity constraints are used to enforce business rules associated with your database and to prevent invalid information in the tables. The most common types of constraints include:

Indexes and Partitioned Indexes in Oracle Airlines Data Model (page 2-16)
> Indexes are optional structures associated with tables or clusters. In addition to the classical B-tree indexes, bitmap indexes are very common in data warehousing environments.

Partitioned Tables in the Oracle Airlines Data Model (page 2-17)
> Partitioning allows a table, index or index-organized table to be subdivided into smaller pieces. Each piece of the database object is called a partition. Each partition has its own name, and may optionally have its own storage characteristics. From the perspective of a database administrator, a partitioned object has multiple pieces that can be managed either collectively or individually. This gives the administrator considerable flexibility in managing partitioned objects. However, from the perspective of the application, a partitioned table is identical to a nonpartitioned table. No modifications are necessary when accessing a partitioned table using SQL DML commands.

Parallel Execution in Oracle Airlines Data Model (page 2-20)
> Parallel Execution enables a database task to be parallelized or divided into smaller units of work, thus allowing multiple processes to work concurrently. By using parallelism, a terabyte of data can be scanned and processed in minutes or less, not hours or days.

2.4.1 Tablespaces in Oracle Airlines Data Model

A tablespace consists of one or more data files, which are physical structures within the operating system you are using.

Recommendations: Defining Tablespaces

If possible, define tablespaces so that they represent logical business units.

Use ultra large data files for a significant improvement in very large Oracle Airlines Data Model warehouse.

Changing the Tablespace and Partitions Used by Tables

You can change the tablespace and partitions used by Oracle Airlines Data Model tables. What you do depends on whether the Oracle Airlines Data Model table has partitions:

- For tables that do not have partitions (that is, lookup tables and reference tables), you can change the existing tablespace for a table.

  By default, Oracle Airlines Data Model defines the partitioned tables as interval partitioning, which means the partitions are created only when new data arrives.

  Consequently, for Oracle Airlines Data Model tables that have partitions (that is, Base, Derived, and Aggregate tables), for the new interval partitions to be generated in new tablespaces rather than current ones, issue the following statements:

  ```
  ALTER TABLE table_name MODIFY DEFAULT ATTRIBUTES
  TABLESPACE new_tablespace_name;
  ```

  When new data is inserted in the table specified by *table_name*, a new partition is automatically created in the tablespace specified by *tablespace new_tablespace_name*.

- For tables that have partitions (that is, base, derived, and aggregate tables), you can specify that new interval partitions be generated into new tablespaces.

  For Oracle Airlines Data Model tables that do not have partitions (that is, lookup tables and reference tables), to change the existing tablespace for a table then issue the following statement:

  ```
  ALTER TABLE table_name MOVE TABLESPACE new_tablespace_name;
  ```

2.4.2 Data Compression in Oracle Airlines Data Model

A key decision that you must make is whether to compress your data. Using table compression reduces disk and memory usage, often resulting in a better scale-up performance for read-only operations. Table compression can also speed up query execution by minimizing the number of round trips required to retrieve data from the disks. Compressing data however imposes a performance penalty on the load speed of the data. Most of the base tables in the Oracle Airlines Data Model are compressed tables.

Recommendations: Data Compression

In general, choose to compress the data. The overall performance gain typically outweighs the cost of compression.

If you decide to use compression, consider sorting your data before loading it to achieve the best possible compression rate. The easiest way to sort incoming data is to load it using an ORDER BY clause on either your CTAS or IAS statement. Specify an ORDER BY a NOT NULL column (ideally non numeric) that has many distinct values (1,000 to 10,000).

Types of Data Compression Available (page 2-13)
> Oracle Database offers the following types of compression:

Related Topics:

Types of Data Compression Available (page 2-13)
> Oracle Database offers the following types of compression:

Compressing Materialized Views (page 3-31)
> Using data compression for a materialized view can provide dramatic performance improvement.

2.4.2.1 Types of Data Compression Available

Oracle Database offers the following types of compression:

- Basic or Standard Compression (page 2-13)

- Advanced Row Compression (page 2-13)

- Hybrid Columnar Compression (HCC) (page 2-14)

Basic or Standard Compression

With standard compression Oracle Database compresses data by eliminating duplicate values in a database block. Standard compression only works for direct path operations (CTAS or IAS). If the data is modified using any kind of conventional DML operation (for example updates), the data within that database block is uncompressed to make the modifications and is written back to the disk uncompressed.

By using a compression algorithm specifically designed for relational data, Oracle Database can compress data effectively and in such a way that Oracle Database incurs virtually no performance penalty for SQL queries accessing compressed tables.

Oracle Airlines Data Model leverages the compress feature for all base, derived, and aggregate tables which reduces the amount of data being stored, reduces memory usage (more data per memory block), and increases query performance.

You can specify table compression by using the COMPRESS clause of the CREATE TABLE statement or you can enable compression for an existing table by using ALTER TABLE statement as shown below.

```
alter table <tablename> move compress;
```

Advanced Row Compression

Advanced row compression is a component of the Advanced Compression option. With advanced row compression, just like standard compression, Oracle Database compresses data by eliminating duplicate values in a database block. But unlike standard compression advanced row compression allows data to remain compressed during all types of data manipulation operations, including conventional DML such as INSERT and UPDATE.

> **See Also:**
>
> For information about Oracle Advanced Compression, see the "Using Table Compression to Save Storage Costs" OBE tutorial.
>
> To access the tutorial, open the Oracle Learning Library in your browser by following the instructions in "Oracle Technology Network (page xiv)"; and, then, search for the tutorial by name.

Hybrid Columnar Compression (HCC)

HCC is available with some storage formats and achieves its compression using a logical construct called the compression unit which is used to store a set of hybrid columnar-compressed rows. When data is loaded, a set of rows is pivoted into a columnar representation and compressed. After the column data for a set of rows has been compressed, it is fit into the compression unit. If a conventional DML is issued against a table with HCC, the necessary data is uncompressed to do the modification and then written back to the disk using a block-level compression algorithm.

> **Tip:**
>
> If your data set is frequently modified using conventional DML, then the use of HCC is not recommended; instead, the use of advanced row compression is recommended.

HCC provides different levels of compression, focusing on query performance or compression ratio respectively. With HCC optimized for query, fewer compression algorithms are applied to the data to achieve good compression with little to no performance impact. However, compression for archive tries to optimize the compression on disk, irrespective of its potential impact on the query performance.

Related Topics:

Oracle Database Administrator's Guide

Oracle Database Concepts

2.4.3 Tables for Supertype and Subtype Entities in Oracle Airlines Data Model

A supertype is a generic entity type that has a relationship with one or more subtypes. A subtype is a sub-grouping of the entities in an entity type that is meaningful to the organization and that shares common attributes or relationships distinct from other subgroups.

A supertype is a generic entity type that has a relationship with one or more subtypes.

- Subtypes inherit all supertype attributes

- Subtypes have attributes that are different from other subtypes

For example, in the default Oracle Airlines Data Model, the table `DWB_PRTY_INTRATN` (Party Interaction) is a supertype that has a number of subtypes including `DWB_PRTY_INTRATN_CALL` (Party Interaction Call) and `DWB_PRTY_INTRATN_EML` (Party Interaction Email).

Recommendations: Tables for Supertype and Subtype Entities

Create separate tables for the super type and all sub type entities for the following reasons:

- Data integrity enforced at database level. (using `NOT NULL` column constraints)

- Relationships can be accurately modeled and enforced including those which apply to only one subtype

- Physical model closely resembles the logical data model.

- It is easier to correlate the logical data model with the physical data model and support the logical data model enhancements and changes.

- Physical data model reflects true business rules (for example, if there are some attributes or relationships mandatory for only one subtype.)

2.4.4 Surrogate Keys in the Physical Model

The surrogate key method for primary key construction involves taking the natural key components from the source systems and mapping them through a process of assigning a unique key value to each unique combination of natural key components (including source system identifier). The resulting primary key value is completely non-intelligent and is typically a numeric data type for maximum performance and storage efficiency.

Advantages of Surrogate keys include:

- Ensure uniqueness: data distribution

- Independent of source systems

- Re-numbering

- Overlapping ranges

- Uses the numeric data type which is the most performant data type for primary keys and joins

Disadvantages of Surrogate keys:

- Have to allocate during ETL

- Complex and expensive re-processing and data quality correction

- Not used in queries – performance impact

- The operational business intelligence requires natural keys to join to operational systems

2.4.5 Integrity Constraints in Oracle Airlines Data Model

Integrity constraints are used to enforce business rules associated with your database and to prevent invalid information in the tables. The most common types of constraints include:

- `PRIMARY KEY` constraints, this is usually defined on the surrogate key column to ensure uniqueness of the record identifiers. In general, it is recommended that you specify the `ENFORCED ENABLED RELY` mode.

- `UNIQUE` constraints, to ensure that a given column (or set of columns) is unique. For slowly changing dimensions, it is recommended that you add a unique constraint on the Business Key and the Effective From Date columns to allow

tracking multiple versions (based on surrogate key) of the same Business Key record.

- `NOT NULL` constraints, to ensure that no null values are allowed. For query rewrite scenarios, it is recommended that you have an inline explicit `NOT NULL` constraint on the primary key column in addition to the primary key constraint.

- `FOREIGN KEY` constraints, to ensure that relation between tables are being honored by the data. Usually in data warehousing environments, the foreign key constraint is present in `RELY DISABLE NOVALIDATE` mode.

The Oracle Database uses constraints when optimizing SQL queries. Although constraints can be useful in many aspects of query optimization, constraints are particularly important for query rewrite of materialized views. Under some specific circumstances, constraints need space in the database. These constraints are in the form of the underlying unique index.

Unlike data in many relational database environments, data in a data warehouse is typically added or modified under controlled circumstances during the extraction, transformation, and loading (ETL) process.

2.4.6 Indexes and Partitioned Indexes in Oracle Airlines Data Model

Indexes are optional structures associated with tables or clusters. In addition to the classical B-tree indexes, bitmap indexes are very common in data warehousing environments.

- Bitmap indexes are optimized index structures for set-oriented operations. Additionally, they are necessary for some optimized data access methods such as star transformations. Bitmap indexes are typically only a fraction of the size of the indexed data in the table.

- B-tree indexes are most effective for high-cardinality data: that is, for data with many possible values, such as customer name or phone number. However, fully indexing a large table with a traditional B-tree index can be prohibitively expensive in terms of disk space because the indexes can be several times larger than the data in the table. B-tree indexes can be stored specifically in a compressed manner to enable huge space savings, storing more keys in each index block, which also leads to less I/O and better performance.

Recommendations: Indexes and Partitioned Indexes

Make the majority of the indexes in your customized Oracle Airlines Data Model bitmap indexes.

Use B-tree indexes only for unique columns or other columns with very high cardinalities (that is, columns that are almost unique). Store the B-tree indexes in a compressed manner.

Partition the indexes. Indexes are just like tables in that you can partition them, although the partitioning strategy is not dependent upon the table structure. Partitioning indexes makes it easier to manage the data warehouse during refresh and improves query performance.

Typically, specify the index on a partitioned table as local. Bitmap indexes on partitioned tables must always be local. B-tree indexes on partitioned tables can be global or local. However, in a data warehouse environment, local indexes are more common than global indexes. Use global indexes only when there is a specific requirement which cannot be met by local indexes (for example, a unique index on a nonpartitioning key, or a performance requirement).

Related Topics:

> Partitioning allows a table, index or index-organized table to be subdivided into smaller pieces. Each piece of the database object is called a partition. Each partition has its own name, and may optionally have its own storage characteristics. From the perspective of a database administrator, a partitioned object has multiple pieces that can be managed either collectively or individually. This gives the administrator considerable flexibility in managing partitioned objects. However, from the perspective of the application, a partitioned table is identical to a nonpartitioned table. No modifications are necessary when accessing a partitioned table using SQL DML commands.

> The two most common operations on a materialized view are query execution and fast refresh, and each operation has different performance requirements.

> Partitioning is a method of physically storing the contents of a cube. It improves the performance of large cubes in the following ways:

> Because of the large volume of data held in a data warehouse, partitioning is an extremely useful option when designing a database. Partitioning the fact tables improves scalability, simplifies system administration, and makes it possible to define local indexes that can be efficiently rebuilt. Partitioning the fact tables also improves the opportunity of fast refreshing the materialized view because this may enable partition change tracking refresh on the materialized view.

2.4.7 Partitioned Tables in the Oracle Airlines Data Model

Partitioning allows a table, index or index-organized table to be subdivided into smaller pieces. Each piece of the database object is called a partition. Each partition has its own name, and may optionally have its own storage characteristics. From the perspective of a database administrator, a partitioned object has multiple pieces that can be managed either collectively or individually. This gives the administrator considerable flexibility in managing partitioned objects. However, from the perspective of the application, a partitioned table is identical to a nonpartitioned table. No modifications are necessary when accessing a partitioned table using SQL DML commands.

As discussed in the following topics, partitioning can provide tremendous benefits to a wide variety of applications by improving manageability, availability, and performance:

> Range partitioning helps improve the manageability and availability of large volumes of data.

> Range partitioning also helps ensure that only the necessary data to answer a query is scanned. Consider the case where business users predominately access the sales data on a weekly basis (for example, total

sales per week) then range partitioning this table by day ensures that the data is accessed in the most efficient manner, as only seven partitions must be scanned to answer the business users query instead of the entire table. The ability to avoid scanning irrelevant partitions is known as partition pruning.

Partitioning the Oracle Airlines Data Model for Join Performance (page 2-19)
Sub-partitioning by hash is used predominately for performance reasons. Oracle Database uses a linear hashing algorithm to create sub-partitions.

Related Topics:

Indexes and Partitioned Indexes in Oracle Airlines Data Model (page 2-16)
Indexes are optional structures associated with tables or clusters. In addition to the classical B-tree indexes, bitmap indexes are very common in data warehousing environments.

Choosing a Cube Partitioning Strategy for Oracle Airlines Data Model (page 3-24)
Partitioning is a method of physically storing the contents of a cube. It improves the performance of large cubes in the following ways:

Partitioning and Materialized Views (page 3-30)
Because of the large volume of data held in a data warehouse, partitioning is an extremely useful option when designing a database. Partitioning the fact tables improves scalability, simplifies system administration, and makes it possible to define local indexes that can be efficiently rebuilt. Partitioning the fact tables also improves the opportunity of fast refreshing the materialized view because this may enable partition change tracking refresh on the materialized view.

See Also:

To understand the various partitioning techniques in Oracle Database, see the "Manipulating Partitions in Oracle Database 11*g*" OBE tutorial.

To access the tutorial, open the Oracle Learning Library in your browser by following the instructions in "Oracle Technology Network (page xiv)"; and, then, search for the tutorial by name.

2.4.7.1 Partitioning the Oracle Airlines Data Model for Manageability

Range partitioning helps improve the manageability and availability of large volumes of data.

Consider the case where two year's worth of sales data or 100 terabytes (TB) is stored in a table. At the end of each day a new batch of data must be to loaded into the table and the oldest days worth of data must be removed. If the Sales table is range partitioned by day then the new data can be loaded using a partition exchange load. This is a sub-second operation that has little or no impact on end user queries.

2.4.7.2 Partitioning the Oracle Airlines Data Model for Easier Data Access

Range partitioning also helps ensure that only the necessary data to answer a query is scanned. Consider the case where business users predominately access the sales data on a weekly basis (for example, total sales per week) then range partitioning this table

by day ensures that the data is accessed in the most efficient manner, as only seven partitions must be scanned to answer the business users query instead of the entire table. The ability to avoid scanning irrelevant partitions is known as partition pruning.

2.4.7.3 Partitioning the Oracle Airlines Data Model for Join Performance

Sub-partitioning by hash is used predominately for performance reasons. Oracle Database uses a linear hashing algorithm to create sub-partitions.

A major performance benefit of hash partitioning is partition-wise joins. Partition-wise joins reduce query response time by minimizing the amount of data exchanged among parallel execution servers when joins execute in parallel. This significantly reduces response time and improves both CPU and memory resource usage. In a clustered data warehouse, this significantly reduces response times by limiting the data traffic over the interconnect (IPC), which is the key to achieving good scalability for massive join operations. Partition-wise joins can be full or partial, depending on the partitioning scheme of the tables to be joined.

As illustrated a full partition-wise join divides a join between two large tables into multiple smaller joins. Each smaller join, performs a joins on a pair of partitions, one for each of the tables being joined. For the optimizer to choose the full partition-wise join method, both tables must be equi-partitioned on their join keys. That is, they have to be partitioned on the same column with the same partitioning method. Parallel execution of a full partition-wise join is similar to its serial execution, except that instead of joining one partition pair at a time, multiple partition pairs are joined in parallel by multiple parallel query servers. The number of partitions joined in parallel is determined by the Degree of Parallelism (DOP).

Figure 2-2   Partitioning for Join Performance

Both tables have the same degree of parallelism and are partitioned the same way on the join column (cust_id)

A large join is divided into multiple smaller joins, each joins a pair of partitions in parallel
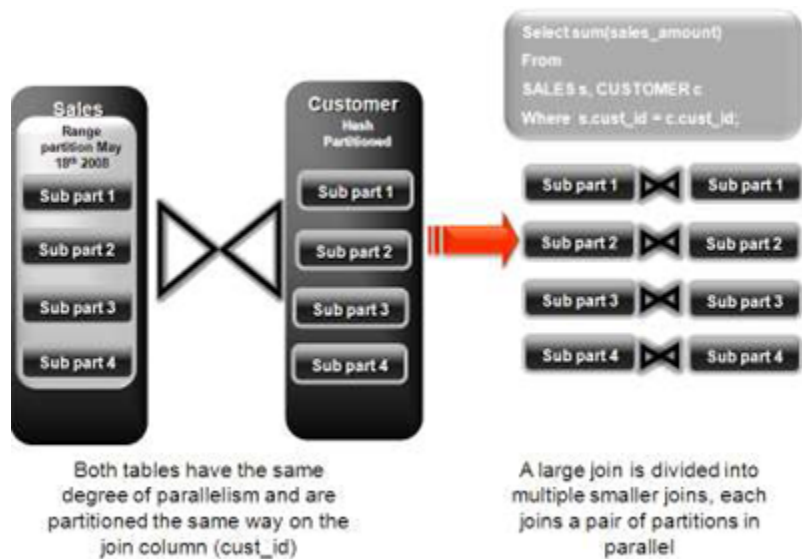
Recommendations: Number of Hash Partitions

To ensure that the data gets evenly distributed among the hash partitions, it is highly recommended that the number of hash partitions is a power of 2 (for example, 2, 4, 8, and so on). A good rule of thumb to follow when deciding the number of hash partitions a table should have is `2 X # of CPUs` rounded to up to the nearest power of 2.

If your system has 12 CPUs, then 32 can be a good number of hash partitions. On a clustered system the same rules apply. If you have 3 nodes each with 4 CPUs, then 32 can still be a good number of hash partitions. However, ensure that each hash partition is at least 16MB in size. Many small partitions do not have efficient scan rates with parallel query. Consequently, if using the number of CPUs makes the size of the hash partitions too small, use the number of Oracle RAC nodes in the environment (rounded to the nearest power of 2) instead.

2.4.8 Parallel Execution in Oracle Airlines Data Model

Parallel Execution enables a database task to be parallelized or divided into smaller units of work, thus allowing multiple processes to work concurrently. By using parallelism, a terabyte of data can be scanned and processed in minutes or less, not hours or days.

Figure 2-3 (page 2-20) illustrates the parallel execution of a full partition-wise join between two tables, Sales and Customers. Both tables have the same degree of parallelism and the same number of partitions. They are range partitioned on a date field and sub partitioned by hash on the cust_id field. As illustrated in the picture, each partition pair is read from the database and joined directly.

There is no data redistribution necessary, thus minimizing IPC communication, especially across nodes. Below figure shows the execution plan for this join.

Figure 2-3   Parallel Execution of a Full Partition-Wise Join Between Two Tables

Partition Hash All above the join &
single PQ set indicate partition-wise join

ID	Operation	Name	Pstart	Pstop	TQ	PQ Distrib
0	SELECT STATEMENT					
1	PX COORDINATOR					
2	PX SEND QC (RANDOM)	:TQ10001			Q1,01	QC (RAND)
3	SORT GROUP BY				Q1,01	
4	PX RECEIVE				Q1,01	
5	PX SEND HASH	:TQ10000			Q1,00	HASH
6	SORT GROUP BY				Q1,00	
7	PX PARTITION HASH ALL		1	128	Q1,00	
8	HASH JOIN				Q1,00	
9	TABLE ACCESS FULL	Customers	1	128	Q1,00	
10	TABLE ACCESS FULL	Sales	1	128	Q1,00	

To ensure that you get optimal performance when executing a partition-wise join in parallel, specify a number for the partitions in each of the tables that is larger than the degree of parallelism used for the join. If there are more partitions than cluster database servers, each cluster database is given one pair of partitions to join, when the cluster database completes that join, it requests another pair of partitions to join. This process repeats until all pairs are processed. This method enables the load to be balanced dynamically (for example, 128 partitions with a degree of parallelism of 32).

What happens if only one table that you are joining is partitioned? In this case the optimizer picks a partial partition-wise join. Unlike full partition-wise joins, partial partition-wise joins can be applied if only one table is partitioned on the join key. Hence, partial partition-wise joins are more common than full partition-wise joins. To

execute a partial partition-wise join, Oracle Database dynamically repartitions the other table based on the partitioning strategy of the partitioned table.

After the other table is repartitioned, the execution is similar to a full partition-wise join. The redistribution operation involves exchanging rows between parallel execution servers. This operation leads to interconnect traffic in Oracle RAC environments, since data must be repartitioned across node boundaries.

Figure 2-4   Partial Partition-Wise Join
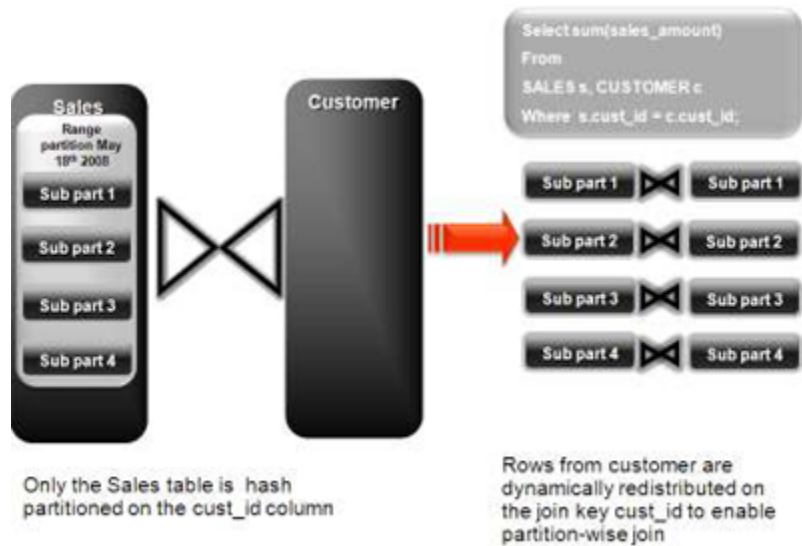
Figure 2-4 (page 2-21) illustrates a partial partition-wise join. It uses the same example as in Figure 2-3 (page 2-20), except that the customer table is not partitioned. Before the join operation is executed, the rows from the customers table are dynamically redistributed on the join key.

Enabling Parallel Execution for a Session (page 2-21)
> Parallel query is the most commonly used parallel execution feature in Oracle Database. Parallel execution can significantly reduce the elapsed time for large queries.

Enabling Parallel Execution of DML Operations (page 2-22)
> Data Manipulation Language (DML) operations such as `INSERT`, `UPDATE`, and `DELETE` can be parallelized by Oracle Database. Parallel execution can speed up large DML operations and is particularly advantageous in data warehousing environments.

Enabling Parallel Execution at the Table Level (page 2-22)
> The setting of parallelism for a table influences the optimizer.

2.4.8.1 Enabling Parallel Execution for a Session

Parallel query is the most commonly used parallel execution feature in Oracle Database. Parallel execution can significantly reduce the elapsed time for large queries.

To enable parallelization for an entire session, execute the following statement:

```
alter session enable parallel query;
```

2.4.8.2 Enabling Parallel Execution of DML Operations

Data Manipulation Language (DML) operations such as INSERT, UPDATE, and DELETE can be parallelized by Oracle Database. Parallel execution can speed up large DML operations and is particularly advantageous in data warehousing environments.

To enable parallelization of DML statements, execute the following statement:

```
alter session enable parallel dml;
```

When you issue a DML statement such as an INSERT, UPDATE, or DELETE, Oracle Database applies a set of rules to determine whether that statement can be parallelized. The rules vary depending on whether the statement is a DML INSERT statement, or a DML UPDATE or DELETE statement.

- The following rules apply when determining how to parallelize DML UPDATE and DELETE statements:

    - Oracle Database can parallelize UPDATE and DELETE statements on partitioned tables, but only when multiple partitions are involved.

    - You cannot parallelize UPDATE or DELETE operations on a nonpartitioned table or when such operations affect only a single partition.

- The following rules apply when determining how to parallelize DML INSERT statements:

    - Standard INSERT statements using a VALUES clause cannot be parallelized.

    - Oracle Database can parallelize only INSERT . . . SELECT . . . FROM statements.

2.4.8.3 Enabling Parallel Execution at the Table Level

The setting of parallelism for a table influences the optimizer.

When using a parallel query. enable parallelism at the table level by issuing the following statement:

```
alter table <table_name> parallel 32;
```

3

Access Layer Customization

Provides information about customizing the access layer of Oracle Airlines Data Model.

Introduction to Customizing the Access Layer of Oracle Airlines Data Model (page 3-2)

> The access layer of Oracle Airlines Data Model provides the calculated and summarized ("flattened") perspectives of the data needed by business intelligence tools. Access layer objects are populated using the data from the foundation layer 3NF objects.

Dimension Tables in Oracle Airlines Data Model (page 3-2)

> Describes the dimension tables in Oracle Airlines Data Model that store the dimension data for fact tables.

Derived Tables in the Oracle Airlines Data Model (page 3-3)

> Derived tables are tables that have as values the result of a non-aggregate calculation against the data in the foundation layer tables. Derived tables have a `DWD_` prefix.

Aggregate Tables in Oracle Airlines Data Model (page 3-11)

> Aggregate tables are tables that aggregate or "roll up" the data to one level higher than a base or derived table. The aggregate tables in the default Oracle Airlines Data Model have a `DWA_` prefix. These aggregate tables provide a view of the data similar to the view provided by a fact table in a snowflake schema.

Dimensional Components in Oracle Airlines Data Model (page 3-12)

> The foundation layer of the Oracle Airlines Data Model is a 3NF model. The default Oracle Airlines Data Model also provides a dimensional model of the data. This dimensional model of the data is a perspective that summarizes and aggregates data, rather than preserving detailed transaction information.

Materialized Views in Oracle Airlines Data Model (page 3-26)

> Materialized views are query results that are stored or "materialized" in advance as schema objects. From a physical design point of view, materialized views resemble tables or partitioned tables and behave like indexes in that they are used transparently and improve performance.

Related Topics:

Physical Model Customization (page 2-1)

> Provides general information about customizing the physical model of Oracle Airlines Data Model and more detailed information about customizing the foundation layer of the physical model.

3.1 Introduction to Customizing the Access Layer of Oracle Airlines Data Model

The access layer of Oracle Airlines Data Model provides the calculated and summarized ("flattened") perspectives of the data needed by business intelligence tools. Access layer objects are populated using the data from the foundation layer 3NF objects.

The access layer objects in the `oadm_sys` schema include: derived and aggregate tables, tables that provide the dimension for the derived and aggregate tables, OLAP cubes, and materialized views. This layer also contains data mining models. The results of the these models are stored in derived tables.

When designing and customizing access layer objects:

* Follow the general guidelines for customizing physical objects.

* Design the access layer objects to support the business intelligence reports and queries that your site makes.

The following topics provide specialized information about designing and customizing access layer objects:

* Dimension Tables in Oracle Airlines Data Model (page 3-2)

* Derived Tables in the Oracle Airlines Data Model (page 3-3)

* Aggregate Tables in Oracle Airlines Data Model (page 3-11)

* Dimensional Components in Oracle Airlines Data Model (page 3-12)

* Materialized Views in Oracle Airlines Data Model (page 3-26)

Related Topics:

General Recommendations When Designing Physical Structures (page 2-10)
    The `oadm_sys` schema delivered with Oracle Airlines Data Model was designed following the best practices for data access and performance. Continue to use these practices when you add new physical objects.

Report and Query Customization (page 5-1)
    Provides information about creating reports, queries, and dashboards against the data in an Oracle Airlines Data Model warehouse.

3.2 Dimension Tables in Oracle Airlines Data Model

Describes the dimension tables in Oracle Airlines Data Model that store the dimension data for fact tables.

The following types of tables act as dimensions to other tables in Oracle Airlines Data Model:

* In the access layer, `DWR_` tables act as dimension tables to derived (`DWD_`) tables.

* In the access layer, `DWR_` tables act as dimension tables to aggregate (`DWA_`) tables.

3.3 Derived Tables in the Oracle Airlines Data Model

Derived tables are tables that have as values the result of a non-aggregate calculation against the data in the foundation layer tables. Derived tables have a `DWD_` prefix.

Oracle Airlines Data Model includes the following types of derived tables. The way you customize these tables varies by type:

- Tables that hold the results of a calculation. For information on customizing these tables.

- Result tables for the data mining models. For information on customizing data mining models.

Creating New Derived Tables for Calculated Data (page 3-3)
> If, during a fit-gap analysis, you identified a need for calculated data that is not provided by the default derived tables, you can meet this need by defining new tables. When designing these tables, name the tables following the convention of using the `DWD_` prefix for derived tables.

Customizing Data Mining Models in Oracle Airlines Data Model (page 3-4)
> Some derived (`DWD_`) tables define in the `oadm_sys` schema are the result tables for the data mining models defined in the Oracle Airlines Data Model. All Oracle Airlines Data Model data mining models use materialized views as source input. Those materialized views are defined in the `oadm_mining_init.sql` script in `$ORACLE_HOME/oadm/pdm/mining`. Different data mining models use different source materialized views.

Related Topics:

Creating New Derived Tables for Calculated Data (page 3-3)
> If, during a fit-gap analysis, you identified a need for calculated data that is not provided by the default derived tables, you can meet this need by defining new tables. When designing these tables, name the tables following the convention of using the `DWD_` prefix for derived tables.

Customizing Data Mining Models in Oracle Airlines Data Model (page 3-4)
> Some derived (`DWD_`) tables define in the `oadm_sys` schema are the result tables for the data mining models defined in the Oracle Airlines Data Model. All Oracle Airlines Data Model data mining models use materialized views as source input. Those materialized views are defined in the `oadm_mining_init.sql` script in `$ORACLE_HOME/oadm/pdm/mining`. Different data mining models use different source materialized views.

Oracle Airlines Data Model Reference

3.3.1 Creating New Derived Tables for Calculated Data

If, during a fit-gap analysis, you identified a need for calculated data that is not provided by the default derived tables, you can meet this need by defining new tables. When designing these tables, name the tables following the convention of using the `DWD_` prefix for derived tables.

3.3.2 Customizing Data Mining Models in Oracle Airlines Data Model

Some derived (`DWD_`) tables define in the `oadm_sys` schema are the result tables for the data mining models defined in the Oracle Airlines Data Model. All Oracle Airlines Data Model data mining models use materialized views as source input. Those materialized views are defined in the `oadm_mining_init.sql` script in `$ORACLE_HOME/oadm/pdm/mining`. Different data mining models use different source materialized views.

When creating a customized Oracle Airlines Data Model warehouse, you can customize the data mining models in the following ways:

Creating a New Data Mining Model for Oracle Airlines Data Model (page 3-4)
> To create a data mining model perform the following steps:

Modifying a Data Mining Model in the Oracle Airlines Data Model (page 3-4)
> To customize an Oracle Airlines Data Model data mining model, perform the following steps:

Tutorial: Customizing the Customer Life Time Value Prediction Data Mining Model (page 3-5)
> After you have populated Oracle Airlines Data Model foundation layer and executed the intra-ETL to populate derived tables, you can leverage the prebuilt Oracle Airlines Data Model data mining models for more advanced analysis and predictions.

3.3.2.1 Creating a New Data Mining Model for Oracle Airlines Data Model

To create a data mining model perform the following steps:

1. Ensure that the `oadm_sys` schema includes a definition for materialized view that you can use as input to the new data mining model. Define a new materialized view, if necessary.

2. Create the new data mining model as you would any data mining model.

3. Add any physical tables needed by the data mining model into the `oadm_sys` schema. When naming a data mining source table, use the prefix `DM_` . When naming a data mining results table, use the prefix `DWD_` .

Related Topics:

> *Oracle Data Mining User's Guide*

3.3.2.2 Modifying a Data Mining Model in the Oracle Airlines Data Model

To customize an Oracle Airlines Data Model data mining model, perform the following steps:

1. Change the definition for the source materialized views used as an input to the data mining model.

2. Train the data mining model again by calling Oracle Airlines Data Model mining package.

3. Ensure that the data mining model reflects the new definition (for example, that a new column has been added).

Example 3-1   Adding a New Column to the create_cust_ltv_svm_rgrsn_model Data Mining Model

The `create_cust_ltv_svm_rgrsn_model` data mining model uses the `dmv_cust_ltv_src` materialized view as source input. To customize the `create_cust_ltv_svm_rgrsn_model` data mining model by adding a new column to the model, perform the following steps:

1. Modify the following materialized views:

    -   `dmv_bkg_fact_src`

    -   `dmv_lylty_acct_bal_src`

    -   `dmv_cust_profile_src`

2. Train customer life time value regression data mining model, execute the following statement:

    `pkg_oadm_mining.create_cust_ltv_svm_rgrsn`

3. Issue the following statement to query the result table and ensure the new column is included in the query result:

    ```
    SELECT attribute_name FROM TABLE( SELECT attribute_set FROM
    TABLE(DBMS_DATA_MINING.GET_MODEL_DETAILS_SVM('CUST_LTV_SVM_RGRSN')));
    ```

Related Topics:

> Refreshing Oracle Airlines Data Model Data Mining Models (page 4-16)
> > The `PKG_INTRA_ETL_PROCESS.Populate_Mining` procedure triggers the data mining model refreshment as part of the initial load of the warehouse. After the initial load of the warehouse, Oracle recommends that you refresh the data mining models monthly.

> Troubleshooting Data Mining Model Creation (page 4-20)
> > After the data mining models are created, check the error log in `oadm_sys.dwc_intra_etl_activity` table.

3.3.2.3 Tutorial: Customizing the Customer Life Time Value Prediction Data Mining Model

After you have populated Oracle Airlines Data Model foundation layer and executed the intra-ETL to populate derived tables, you can leverage the prebuilt Oracle Airlines Data Model data mining models for more advanced analysis and predictions.

This tutorial shows how to predict the Life Time Value of customers, who are frequent flier passengers, for next six months based on populated Oracle Airlines Data Model warehouse. Using prebuilt Oracle Airlines Data Model data mining models, you can easily and very quickly see the prediction results of your customers, without having to go through all of the data preparation, training, testing, and applying process that you must perform in a traditional from-scratch mining project.

After initially generating a data mining model, as time goes by, the customer information, behavior, and usage change. Consequently, you must refresh the previous trained data mining models based on the latest customer and usage data. You can follow the process in this tutorial to refresh the data mining models to acquire predictions on latest customer information.

This tutorial shows you how to investigate the Customer Life Time Value Prediction model through Oracle Airlines Data Model mining APIs. To use different parameters

in the training process, or customize the model in more advanced fashion, you can either modify mining settings tables (with `DM_` as prefix) or use the Oracle Data Miner GUI tool.

This tutorial consists of the following:

- Tutorial Prerequisites (page 3-6)

- Preparing Your Environment (page 3-6)

- Generating the Model (page 3-9)

- Checking the Result (page 3-10)

Tutorial Prerequisites

Before starting this tutorial:

1. Review the Oracle by Example (OBE) tutorial "Using Oracle Data Miner 11*g* Release 2."

See Also:

To access this tutorial, open the Oracle Learning Library in your browser by following the instructions in "Oracle Technology Network (page xiv)"; and, then, search for the tutorial by name

---

2. Install Oracle Airlines Data Model.

3. Populate the base, reference, and lookup tables.

4. Execute the intra-ETL.

   Ensure that at least the following tables contain valid data:

   ```
   DWB_LYLTY_ACCT_BAL_HIST
   DWR_CLNDR
   DWR_FREQ_FLYR
   DWR_GEO
   ```

Note:

If you have not populated the real data, and only want to learn the Oracle Airlines Data Model data mining model, you can use the sample data by performing the following steps:

1. Ensure that during the install, you generated the calendar data covering range of 2005~2011. For example, the parameters of starting from `20050101` for seven years satisfy this condition.

2. Download the sample data (`oadm_sample_mining.dmp.zip`) and import the data into your new `oadm_sys` schema.

Preparing Your Environment

This tutorial requires a valid, populated Oracle Airlines Data Model warehouse.

To prepare the environment, perform the following steps:

1. In SQL Developer, connect to the `oadm_sys` schema.

   Tip:

   SQL Developer can be found on any Oracle Database Installation under `$ORACLE_HOME/sqldeveloper`.
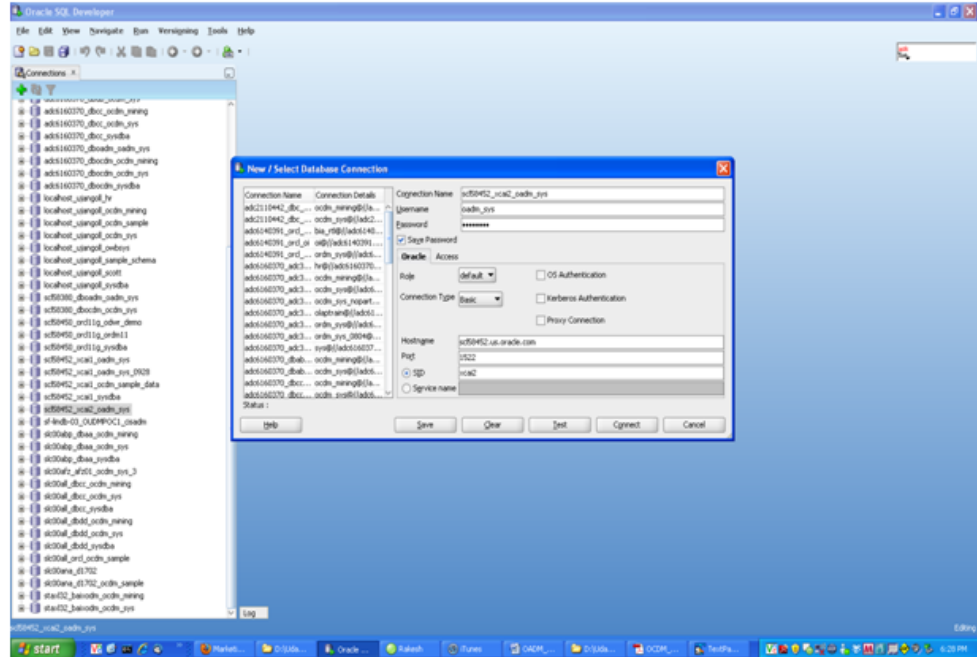
   Note:

   For more information about using SQL Developer, refer to the tutorial "Getting Started with Oracle SQL Developer 3.0".

   See Also:

   To access this tutorial, open the Oracle Learning Library in your browser by following the instructions in "Oracle Technology Network (page xiv)"; and, then, search for the tutorial by name

2. After you connect to the `oadm_sys` schema, you can see all of the tables that have been created. You can narrow down the list by right clicking "Tables" and then applying filters.
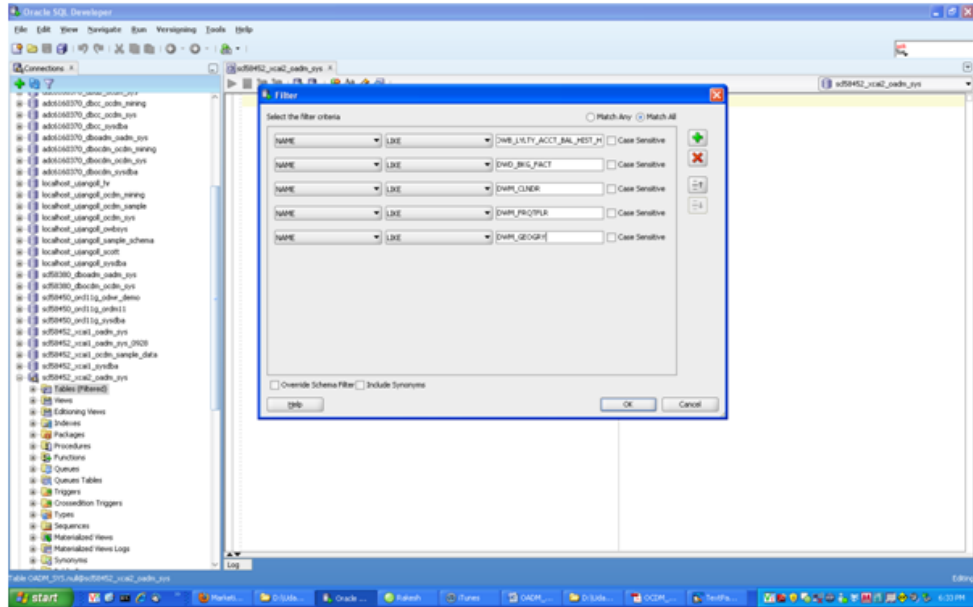
3. (Optional) If you have not populated those tables with your own data, you can use the sample data. After you download the sample data, perform the following steps to import the data.

See Also:

"Tutorial Prerequisites (page 3-6)"

---

a. Grant dba to `oadm_sys` by issuing the following command:

```
grant dba to oadm_sys
```

b. Disable all foreign keys on those tables required by the tutorial.

First, issue the following statement that generates SQL statements:

```
SELECT 'ALTER TABLE ' || table_name || ' DISABLE CONSTRAINT ' ||
  CONSTRAINT_NAME ' CASCADE;' FROM all_constraints
 WHERE status='ENABLED' AND owner='OADM_SYS'
   AND constraint_type=
     'R' and table_name IN
       ('DWB_LYLTY_ACCT_BAL_HIST','DWD_BKG_FACT','DWR_CLNDR',
          'DWR_FREQ_FLYR','DWR_GEO') ;
```

Then, to actually disable the foreign keys for the following tables, execute the SQL statements generated by the previous SELECT statement:

```
DWB_LYLTY_ACCT_BAL_HIST
DWD_BKG_FACT
DWR_CLNDR
DWR_FREQ_FLYR
DWR_GEO
```

c. Ensure that the sample dump, `oadm_sample.dmp`, is in default data dump directory, `DATA_PUMP_DIR`. Then, import the sample mining dump into `oadm_sys` schema by issuing the following statement:

> **Note:**
>
> Replace *password* with your password for `oadm_sys`.

```
impdp oadm_sys/password directory=DATA_PUMP_DIR dumpfile=oadm_sample.dmp
   content=DATA_ONLY table_exists_action=truncate
   TABLES=oadm_sample.DWB_LYLTY_ACCT_BAL_HIST,
   oadm_sample.DWD_BKG_FACT,oadm_sample.DWR_CLNDR,
   oadm_sample.DWR_FREQ_FLYR,oadm_sample.DWR_GEO
```

4. Review the tables to ensure that they contain valid data (either your own customer data or sample mining data).
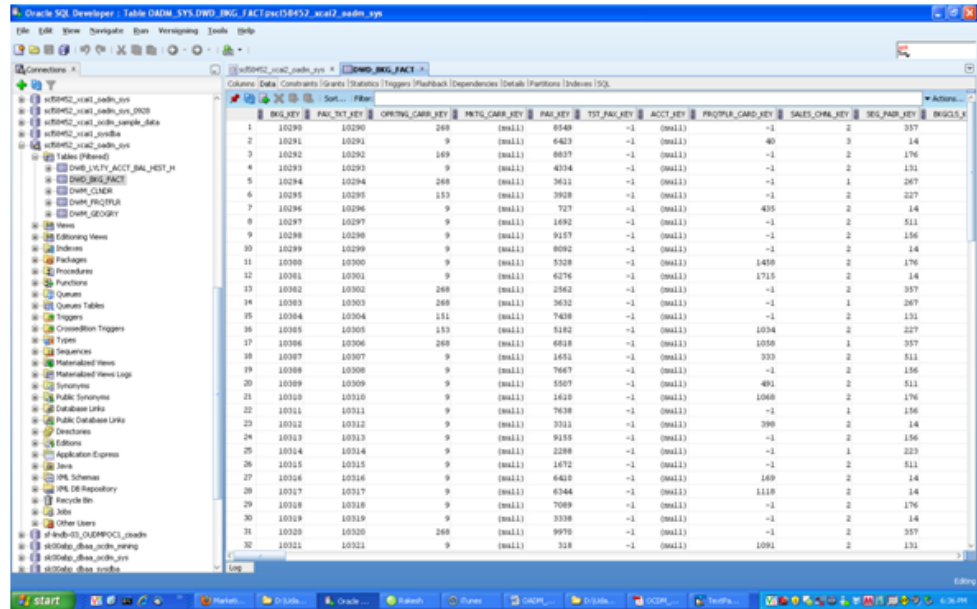
5. Check the mining result table `DWD_CUST_MNNG` is empty before executing the model procedure in Oracle Airlines Data Model Mining APIs.

Generating the Model

This tutorial uses the following procedures from Oracle Airlines Data Model Mining APIs:

- `pkg_oadm_mining.refresh_mining_source` that refreshes all mining source materialized views.

- `pkg_oadm_mining.create_cust_ltv_svm_rgrsn` that generates the Customer Life Time Value Prediction Model.

Perform the following steps to use the procedures:

1. Refresh the Oracle Airlines Data Model mining source materialized views by executing the following SQL statements:

```
SELECT count(*) FROM dmv_cust_ltv_src;
exec pkg_oadm_mining.refresh_mining_source;
SELECT count(*)FROM dmv_cust_ltv_src;
```

These statements:

    a. Displays the number of records in `dmv_cust_ltv_src` materialized view before materialized view refresh.

    b. Refreshes mining source materialized views.

    c. Displays the number of records in `dmv_cust_ltv_src` materialized view after materialized view refresh.

2. Generate the Customer Life Time Value Prediction Model by executing the following statements:

```
SELECT count(*) FROM dwd_cust_mnng;
SELECT count(*) FROM dwd_cust_ltv_svm_factor;
EXEC pkg_oadm_mining. create_cust_ltv_svm_rgrsn;
SELECT count(*) FROM dwd_cust_mnng;
SELECT count(*) FROM dwd_cust_ltv_svm_factor;
```

These statements:

    a. Shows the records count in `dwd_cust_mnng` table before data mining model build.

    b. Shows the records counts in the `dwd_cust_ltv_svm_factor` table before data mining model build.

    c. Trains data mining model.

    d. Shows the records count in the `dwd_cust_mnng` table after data mining model build.

    e. Shows the records count in the `dwd_cust_ltv_svm_factor` table after data mining model build.

Checking the Result

After refreshing the mining source materialized views and building the data mining model, check the mining prediction results in `dwd_cust_mnng` table as shown in the following steps:

1. Issue the following query:

```
SELECT frqtflr_card_key, ltv_value, ltv_band_cd FROM dwd_cust_mnng;
```

2. For each customer identified by `frqtflr_card_key`, the `ltv_value` column gives the prediction of customer life time value, a continuous value. The `ltv_band_cd` column is populated by binning the prediction value, `ltv_value`.

Related Topics:

Oracle Data Mining Concepts

3.4 Aggregate Tables in Oracle Airlines Data Model

Aggregate tables are tables that aggregate or "roll up" the data to one level higher than a base or derived table. The aggregate tables in the default Oracle Airlines Data Model have a `DWA_` prefix. These aggregate tables provide a view of the data similar to the view provided by a fact table in a snowflake schema.

The default Oracle Airlines Data Model defines several aggregate tables. For example, the `DWA_DLY_BKG_FACT` table aggregates the values of the `DWD_BKG_FACT` table to the day, segment, and traffic category levels.

If, during fit-gap analysis, you identified a need for aggregated data that is not provided by the default aggregate tables, you can define new materialized views. When designing these tables, keep the following points in mind:

- Create a query for the materialized view that aggregates up only a single level. For example, if aggregating over time, then aggregate only from day to month.

Note:

When you must aggregate up many levels (for example in time, month, quarter, and year) or different hierarchies (for example, the fiscal and calendar hierarchies for a time dimension), do not define a `DWA_` table; instead, define the aggregations by creating OLAP cubes.

- Name the tables following the conventions for physical objects and use a `DWA_` prefix.

Related Topics:

Materialized Views in Oracle Airlines Data Model (page 3-26)
> Materialized views are query results that are stored or "materialized" in advance as schema objects. From a physical design point of view, materialized views resemble tables or partitioned tables and behave like indexes in that they are used transparently and improve performance.

Defining New Oracle OLAP Cubes for Oracle Airlines Data Model (page 3-21)
> You can add new OLAP cubes to the `oadm_sys` schema.

General Naming Conventions for Physical Objects (page 2-6)
> Describes the general naming guidelines for naming physical objects that you define in the data model.

Oracle Airlines Data Model Reference

3.5 Dimensional Components in Oracle Airlines Data Model

The foundation layer of the Oracle Airlines Data Model is a 3NF model. The default Oracle Airlines Data Model also provides a dimensional model of the data. This dimensional model of the data is a perspective that summarizes and aggregates data, rather than preserving detailed transaction information.

Familiarize yourself with dimensional modeling by reading the following topics before you begin to customize the dimensional model of the default Oracle Airlines Data Model:

Characteristics of a Dimensional Model (page 3-13)
> The simplicity of a dimensional model is inherent because it defines objects that represent real-world business entities. Analysts know which business measures they are interested in examining, which dimensions and attributes make the data meaningful, and how the dimensions of their business are organized into levels and hierarchies.

About the Characteristics of Relational Star and Snowflake Tables (page 3-14)
> For relational tables, the dimensional model has historically been implemented as a star or snowflake schema. Dimension tables (which contain information about hierarchies, levels, and attributes) join one or more fact tables. Fact tables are the large tables that store quantifiable business measurements (such as Booking or Partner Earning) and typically have foreign keys to the dimension tables. Dimension tables, also known as lookup or reference tables contain the relatively static or descriptive data in the data warehouse.

Characteristics of the OLAP Dimensional Model (page 3-16)
> Oracle OLAP Cubes logically represent data similar to relational star tables, although the data is actually stored in multidimensional arrays. Like dimension tables, cube dimensions organize members into hierarchies, levels, and attributes. The cube stores the measure (fact) data. The dimensions form the edges of the cube.

Characteristics of the OLAP Cubes in Oracle Airlines Data Model (page 3-20)
> The default access layer of Oracle Airlines Data Model provides a dimensional perspective of the data using Oracle OLAP cubes.

Defining New Oracle OLAP Cubes for Oracle Airlines Data Model (page 3-21)
> You can add new OLAP cubes to the `oadm_sys` schema.

Changing an Oracle OLAP Cube in Oracle Airlines Data Model (page 3-23)
> Common customizations to Oracle Airlines Data Model cubes are
> changing the dimensions or the measures of the cube.

Creating a Forecast Cube for Oracle Airlines Data Model (page 3-23)
> Provides steps to create a forecast cube for Oracle Airlines Data Model.

Choosing a Cube Partitioning Strategy for Oracle Airlines Data Model
(page 3-24)
> Partitioning is a method of physically storing the contents of a cube. It
> improves the performance of large cubes in the following ways:

Choosing a Cube Data Maintenance Method for Oracle Airlines Data Model
(page 3-25)
> While developing a dimensional model of your data, it is a good idea to
> map and load each object immediately after you create it so that you can
> immediately detect and correct any errors that you made to the object
> definition or the mapping.

3.5.1 Characteristics of a Dimensional Model

The simplicity of a dimensional model is inherent because it defines objects that
represent real-world business entities. Analysts know which business measures they
are interested in examining, which dimensions and attributes make the data
meaningful, and how the dimensions of their business are organized into levels and
hierarchies.

In the simplest terms, a dimensional model identifies the following objects:

- **Measures:** Measures store quantifiable business data (such as Booking or Partner
  Earning). Measures are sometimes called "facts". Measures are organized by one or
  more dimensions and may be stored or calculated at query time:

  - **Stored Measures:** Stored measures are loaded and stored at the leaf level.
    Commonly, there is also a percentage of summary data that is stored. Summary
    data that is not stored is dynamically aggregated when queried.

  - **Calculated Measures:** Calculated measures are measures whose values are
    calculated dynamically at query time. Only the calculation rules are stored in
    the database. Common calculations include measures such as ratios, differences,
    moving totals, and averages. Calculations do not require disk storage space, and
    they do not extend the processing time required for data maintenance.

- **Dimensions:** A dimension is a structure that categorizes data to enable users to
  answer business questions. Commonly used dimensions are Account, Airport, and
  Flight, Geography, and Calendar. A dimension's structure is organized
  hierarchically based on parent-child relationships. These relationships enable:

  - Navigation between levels

    Hierarchies on dimensions enable drilling down to lower levels or navigation
    (rolling up) to higher levels. Drilling down on a Calendar Year dimension
    member 2005 typically navigates you to quarters Q1 2005 through Q4 2005. In a
    calendar year hierarchy, drilling down on Q1 2005 would navigate you to the
    months, January 05 through March 05. These kinds of relationships make it easy
    for users to navigate large volumes of multidimensional data.

  - Aggregation from child values to parent values

The parent represents the aggregation of its children. Data values at lower levels aggregate into data values at higher levels. Dimensions are structured hierarchically so that data at different levels of aggregation are manipulated efficiently for analysis and display.

– Allocation from parent values to child values

The reverse of aggregation is allocation and is heavily used by planning budgeting, and similar applications. Here, the role of the hierarchy is to identify the children and descendants of particular dimension members of "top-down" allocation of budgets (among other uses).

– Grouping of members for calculations

Share and index calculations take advantage of hierarchical relationships (for example, the percentage of total ticket booking count contributed by each frequent flyer member level, or the percentage share of flown revenue for a certain segment, or sales revenue as a percentage of the geographical region for a sales agent location).

A dimension object helps to organize and group dimensional information into hierarchies. This represents natural 1:n relationships between columns or column groups (the levels of a hierarchy) that cannot be represented with constraint conditions. Going up a level in the hierarchy is called rolling up the data and going down a level in the hierarchy is called drilling down the data.

You can implement a dimensional model by using the following options:

• **Relational tables in a star schema configuration:** This traditional method of implementing a dimensional model.

• **Oracle OLAP Cubes:**. The physical model provided with Oracle Airlines Data Model provides a dimensional perspective of the data using Oracle OLAP cubes.

Related Topics:

About the Characteristics of Relational Star and Snowflake Tables (page 3-14)
> For relational tables, the dimensional model has historically been implemented as a star or snowflake schema. Dimension tables (which contain information about hierarchies, levels, and attributes) join one or more fact tables. Fact tables are the large tables that store quantifiable business measurements (such as Booking or Partner Earning) and typically have foreign keys to the dimension tables. Dimension tables, also known as lookup or reference tables contain the relatively static or descriptive data in the data warehouse.

Characteristics of the OLAP Dimensional Model (page 3-16)
> Oracle OLAP Cubes logically represent data similar to relational star tables, although the data is actually stored in multidimensional arrays. Like dimension tables, cube dimensions organize members into hierarchies, levels, and attributes. The cube stores the measure (fact) data. The dimensions form the edges of the cube.

3.5.2 About the Characteristics of Relational Star and Snowflake Tables

For relational tables, the dimensional model has historically been implemented as a star or snowflake schema. Dimension tables (which contain information about hierarchies, levels, and attributes) join one or more fact tables. Fact tables are the large tables that store quantifiable business measurements (such as Booking or Partner

Earning) and typically have foreign keys to the dimension tables. Dimension tables, also known as lookup or reference tables contain the relatively static or descriptive data in the data warehouse.

A star schema borders on a physical model, as drill paths, hierarchy and query profile are embedded in the data model itself rather than the data. This in part at least, is what makes navigation of the model so straightforward for end users. Star schemas usually have a large fact table surrounded by smaller dimension tables. Dimension tables do not change very much. Most of the information that the users need are in the fact tables. Therefore, star schemas have fewer table joins than do 3NF models.

A star schema is so called because the diagram resembles a star, with points radiating from a center. The center of the star consists of one or more fact tables and the points of the star are the dimension tables.

Figure 3-1   Star Schema Diagram
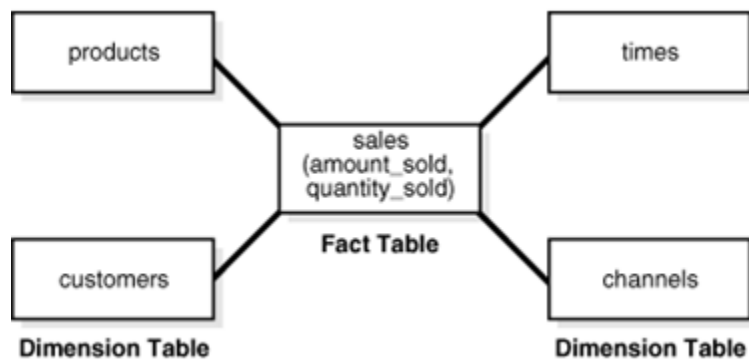
Snowflake schemas are slight variants of a simple star schema where the dimension tables are further normalized and broken down into multiple tables. The snowflake aspect only affects the dimensions and not the fact table and is therefore considered conceptually equivalent to star schemas. Snowflake dimensions are useful and indeed necessary when there are fact tables of differing granularity. A month-level derived or aggregate table (or materialized view) must be associated with a month level snowflake dimension table rather than the default (lower) Day level star dimension table.

> Declaring Relational Dimension Tables (page 3-15)
>> When a relational table acts as a dimension to a fact table, Oracle recommends that you declare that table as a dimension (even though it is not necessary). Defined dimensions can yield significant performance benefits, and support the use of more complex types of rewrite.

> Validating Relational Dimension Tables (page 3-16)
>> To improve the data quality of the dimension data in the data warehouse, Oracle recommends that you validate the declarative information about the relationships between the dimension members after any modification to the dimension data.

3.5.2.1 Declaring Relational Dimension Tables

When a relational table acts as a dimension to a fact table, Oracle recommends that you declare that table as a dimension (even though it is not necessary). Defined dimensions can yield significant performance benefits, and support the use of more complex types of rewrite.

To define and declare the structure of the dimension use the `CREATE DIMENSION` command. Use the `LEVEL` clause to identify the names of the dimension levels.

> **Note:**
>
> In the default Oracle Airlines Data Model, relational tables used as dimension tables are not defined using the `CREATE DIMENSION` command.

3.5.2.2 Validating Relational Dimension Tables

To improve the data quality of the dimension data in the data warehouse, Oracle recommends that you validate the declarative information about the relationships between the dimension members after any modification to the dimension data.

To perform this validation, use the `VALIDATE_DIMENSION` procedure of the `DBMS_DIMENSION` package. When the `VALIDATE_DIMENSION` procedure encounters any errors, the procedure places the errors into the `DIMENSION_EXCEPTIONS` table. To find the exceptions identified by the `VALIDATE_DIMENSION` procedure, query the `DIMENSION_EXCEPTIONS` table.

You can schedule a call to the `VALIDATE_DIMENSION` procedure as a post-process step to the regular Incremental Dimension load script. This can be done before the call to refresh the derived or aggregate tables of the data model through materialized view refresh, intra-ETL package calls.

> **Note:**
>
> In the ETL delivered with the default Oracle Airlines Data Model, relational dimension tables are not validated.

3.5.3 Characteristics of the OLAP Dimensional Model

Oracle OLAP Cubes logically represent data similar to relational star tables, although the data is actually stored in multidimensional arrays. Like dimension tables, cube dimensions organize members into hierarchies, levels, and attributes. The cube stores the measure (fact) data. The dimensions form the edges of the cube.

Oracle OLAP is an OLAP server embedded in the Oracle Database. Oracle OLAP provides native multidimensional storage and speed-of-thought response times when analyzing data across multiple dimensions. The database provides rich support for analytics such as time series calculations, forecasting, advanced aggregation with additive and nonadditive operators, and allocation operations.

By integrating multidimensional objects and analytics into the database, Oracle provides the best of both worlds: the power of multidimensional analysis along with the reliability, availability, security, and scalability of the Oracle Database.

Oracle OLAP is fully integrated into Oracle Database. At a technical level, this means:

- The OLAP engine runs within the kernel of Oracle Database.

- Dimensional objects are stored in Oracle Database in their native multidimensional format.

- Cubes and other dimensional objects are first class data objects represented in the Oracle data dictionary.

- Data security is administered in the standard way, by granting and revoking privileges to Oracle Database users and roles.

- OLAP cubes, dimensions, and hierarchies are exposed to applications as relational views. Consequently, applications can query OLAP objects using SQL.

- Oracle OLAP cubes can be enhanced so that they are materialized views.

Benefits of Using Oracle OLAP

The benefits to your organization are significant. Oracle OLAP offers the power of simplicity: One database, standard administration and security, standard interfaces and development tools.

The Oracle OLAP dimensional data model is highly structured. Structure implies rules that govern the relationships among the data and control how the data can be queried. Cubes are the physical implementation of the dimensional model, and thus are highly optimized for dimensional queries. The OLAP engine leverages this innate dimensionality in performing highly efficient cross-cube joins for inter-row calculations, outer joins for time series analysis, and indexing. Dimensions are pre-joined to the measures. The technology that underlies cubes is based on an indexed multidimensional array model, which provides direct cell access.

The OLAP engine manipulates dimensional objects in the same way that the SQL engine manipulates relational objects. However, because the OLAP engine is optimized to calculate analytic functions, and dimensional objects are optimized for analysis, analytic and row functions can be calculated much faster in OLAP than in SQL.

The dimensional model enables Oracle OLAP to support high-end business intelligence tools and applications such as OracleBI Discoverer Plus OLAP, OracleBI Spreadsheet Add-In, Oracle Business Intelligence Suite Extended Edition, BusinessObjects Enterprise, and Cognos ReportNet.

Oracle OLAP Dimensional Objects

Oracle OLAP dimensional objects include cubes, measures, dimensions, hierarchies, levels, and attributes. The OLAP dimensional objects are described in detail in *Oracle OLAP User's Guide*. Figure 3-2 (page 3-18) shows the general relationships among the objects.

Figure 3-2   Diagram of the OLAP Dimensional Model
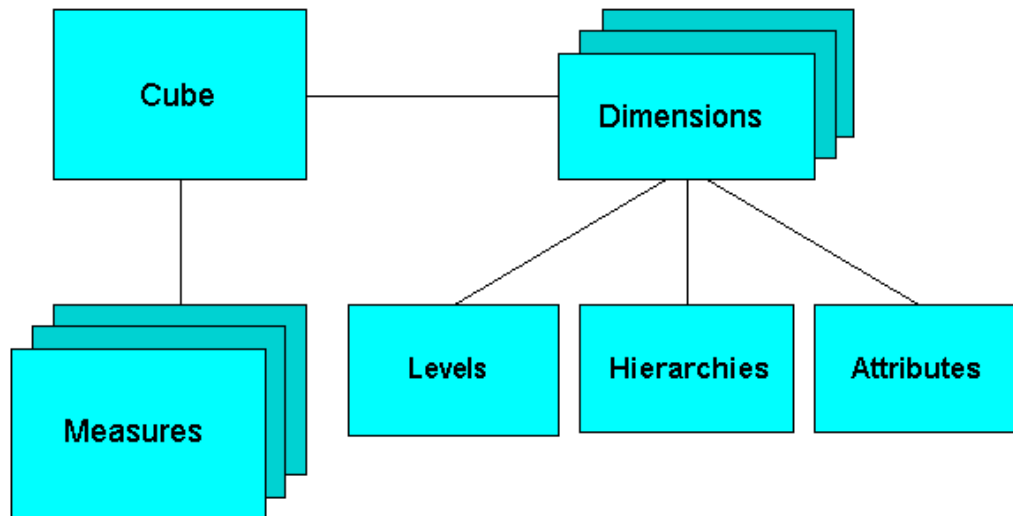
Oracle OLAP Cube Views (page 3-19)
>   When you define an OLAP cube, Oracle OLAP automatically generates a
>   set of relational views on the cube and its dimensions and hierarchies.

Cube Materialized Views (page 3-19)
>   Oracle OLAP cubes can be enhanced so that they are materialized views.
>   A cube that is enhanced in this way is called a cube materialized view
>   and has a `CB$` prefix. Cube materialized views are incrementally
>   refreshed through the Oracle Database materialized view subsystem,
>   and serve as targets for transparent rewrite of queries against the source
>   tables.

Related Topics:

Characteristics of the OLAP Cubes in Oracle Airlines Data Model (page 3-20)
>   The default access layer of Oracle Airlines Data Model provides a
>   dimensional perspective of the data using Oracle OLAP cubes.

Oracle OLAP Cube Views (page 3-19)
>   When you define an OLAP cube, Oracle OLAP automatically generates a
>   set of relational views on the cube and its dimensions and hierarchies.

Report and Query Customization (page 5-1)
>   Provides information about creating reports, queries, and dashboards
>   against the data in an Oracle Airlines Data Model warehouse.

Cube Materialized Views (page 3-19)
>   Oracle OLAP cubes can be enhanced so that they are materialized views.
>   A cube that is enhanced in this way is called a cube materialized view
>   and has a `CB$` prefix. Cube materialized views are incrementally
>   refreshed through the Oracle Database materialized view subsystem,
>   and serve as targets for transparent rewrite of queries against the source
>   tables.

Oracle OLAP User's Guide

3.5.3.1 Oracle OLAP Cube Views

When you define an OLAP cube, Oracle OLAP automatically generates a set of relational views on the cube and its dimensions and hierarchies.

- **Cube view**: Each cube has a cube view that presents the data for all the measures and calculated measures in the cube. You can use a cube view like a fact table in a star or snowflake schema. However, the cube view contains all the summary data in addition to the detail level data. The default name of a cube view is *cube*`_VIEW`.

- **Dimension and Hierarchy views**: Each dimension has one dimension view plus a hierarchy view for each hierarchy associated with the dimension. The default name for a dimension view is *dimension*`_VIEW`. For a hierarchy view, the default name is *dimension_hierarchy*`_VIEW`.

These views are related in the same way as fact and dimension tables in a star schema. Cube views serve the same function as fact tables, and hierarchy views and dimension views serve the same function as dimension tables. Typical queries join a cube view with either a hierarchy view or a dimension view.

SQL applications query these views to display the information-rich contents of these objects to analysts and decision makers. You can also create custom views that follow the structure expected by your applications, using the system-generated views like base tables.

Related Topics:

Report and Query Customization (page 5-1)
> Provides information about creating reports, queries, and dashboards against the data in an Oracle Airlines Data Model warehouse.

See Also:

The discussion on querying dimensional objects in *Oracle OLAP User's Guide*

3.5.3.2 Cube Materialized Views

Oracle OLAP cubes can be enhanced so that they are materialized views. A cube that is enhanced in this way is called a cube materialized view and has a `CB$` prefix. Cube materialized views are incrementally refreshed through the Oracle Database materialized view subsystem, and serve as targets for transparent rewrite of queries against the source tables.

The OLAP dimensions associated with a cube materialized view are also defined with materialized view capabilities.

Necessary Cube Characteristics for Cube Materialized Views

A cube must conform to these requirements, before it can be designated as a cube materialized view:

- All dimensions of the cube have at least one level and one level-based hierarchy. Ragged and skip-level hierarchies are not supported. The dimensions must be mapped.

- All dimensions of the cube use the same aggregation operator, which is either `SUM`, `MIN`, or `MAX`.

- The cube has one or more dimensions and one or more measures.

- The cube is fully defined and mapped. For example, if the cube has five measures, then all five are mapped to the source tables.

- The data type of the cube is `NUMBER`, `VARCHAR2`, `NVARCHAR2`, or `DATE`.

- The source detail tables support dimension and rely constraints. If they are not defined, then use the Relational Schema Advisor to generate a script that defines them on the detail tables.

- The cube is compressed.

- The cube can be enriched with calculated measures, but it cannot support more advanced analytics in a cube script.

Adding Materialized View Capabilities

To add materialized view capabilities to an OLAP cube, perform the following steps:

1. In the Analytic Workspace Manager, connect to the `oadm_sys` schema.

2. From the cube list, select the cube which you want to enable.

3. In the right pane, select the **Materialized Views** tab.

4. Select **Enable Materialized View Refresh of the Cube**. then click **Apply**.

> **Note:**
>
> You cannot enable the cube materialized view for a forecast cube.

> **See Also:**
>
> For more information on working with OLAP cubes, see the following OBE tutorials:
>
> - "Querying OLAP 11*g* Cubes"
>
> - "Using Oracle OLAP 11*g* With Oracle BI Enterprise Edition"
>
> To access the tutorials, open the Oracle Learning Library in your browser by following the instructions in "Oracle Technology Network (page xiv)"; and, then, search for the tutorials by name

Related Topics:

   Oracle OLAP User's Guide

3.5.4 Characteristics of the OLAP Cubes in Oracle Airlines Data Model

The default access layer of Oracle Airlines Data Model provides a dimensional perspective of the data using Oracle OLAP cubes.

There are OLAP cubes defined in the default `oadm_sys` schema. The OLAP cubes in the Oracle Airlines Data Model have the following characteristics:

- All of the default OLAP cubes are loaded with data from `DWA_` tables.

- The cubes were defined and built using the Analytical Workspace Manager (AWM) client tool.

- A relational view (with a `_VIEW` suffix) is defined over each of the OLAP cubes.

- All of the OLAP cubes in the Oracle Airlines Data Model are ready to be enabled as cube materialized views (that is, `CB$` objects).

> **See Also:**
>
> To enable the cube materialized views, follow the steps outlined in "Adding Materialized View Capabilities (page 3-20)"

> **Tip:**
>
> Immediately after installation, all materialized views underlying the OLAP cubes are disabled by default.

Related Topics:

Defining New Oracle OLAP Cubes for Oracle Airlines Data Model (page 3-21)
  You can add new OLAP cubes to the `oadm_sys` schema.

Changing an Oracle OLAP Cube in Oracle Airlines Data Model (page 3-23)
  Common customizations to Oracle Airlines Data Model cubes are changing the dimensions or the measures of the cube.

Creating a Forecast Cube for Oracle Airlines Data Model (page 3-23)
  Provides steps to create a forecast cube for Oracle Airlines Data Model.

Choosing a Cube Partitioning Strategy for Oracle Airlines Data Model (page 3-24)
  Partitioning is a method of physically storing the contents of a cube. It improves the performance of large cubes in the following ways:

Choosing a Cube Data Maintenance Method for Oracle Airlines Data Model (page 3-25)
  While developing a dimensional model of your data, it is a good idea to map and load each object immediately after you create it so that you can immediately detect and correct any errors that you made to the object definition or the mapping.

Characteristics of the OLAP Dimensional Model (page 3-16)
  Oracle OLAP Cubes logically represent data similar to relational star tables, although the data is actually stored in multidimensional arrays. Like dimension tables, cube dimensions organize members into hierarchies, levels, and attributes. The cube stores the measure (fact) data. The dimensions form the edges of the cube.

Oracle OLAP User's Guide

3.5.5 Defining New Oracle OLAP Cubes for Oracle Airlines Data Model

You can add new OLAP cubes to the `oadm_sys` schema.

Perform the following steps to define new cubes:

1. Ensure that there is an aggregate table (DWA_) to use as the "lowest leaf" data for the cube.

2. Use the AWM to define new cubes for a customized version of Oracle Airlines Data Model:

   • Fact tables correspond to cubes.

   • Data columns in the fact tables correspond to measures.

   • Foreign key constraints in the fact tables identify the dimension tables.

   • Dimension tables identify the dimensions.

   • Primary keys in the dimension tables identify the base-level dimension members.

   • Parent columns in the dimension tables identify the higher level dimension members.

   • Columns in the dimension tables containing descriptions and characteristics of the dimension members identify the attributes.

   You can also get insights into the dimensional model by looking at the sample reports included with Oracle Airlines Data Model.

   > **Tip:**
   >
   > While investigating your source data, you may decide to create relational views that more closely match the dimensional model that you plan to create.

3. Add materialized view capabilities to the OLAP cubes.

   > **Note:**
   >
   > For more information on creating OLAP cubes, see the "Building OLAP 11*g* Cubes" OBE tutorial.
   >
   > To access the tutorial, open the Oracle Learning Library in your browser by following the instructions in "Oracle Technology Network (page xiv)"; and, then, search for the tutorial by name.

Related Topics:

Aggregate Tables in Oracle Airlines Data Model (page 3-11)
> Aggregate tables are tables that aggregate or "roll up" the data to one level higher than a base or derived table. The aggregate tables in the default Oracle Airlines Data Model have a DWA_ prefix. These aggregate tables provide a view of the data similar to the view provided by a fact table in a snowflake schema.

Characteristics of the OLAP Dimensional Model (page 3-16)
> Oracle OLAP Cubes logically represent data similar to relational star tables, although the data is actually stored in multidimensional arrays. Like dimension tables, cube dimensions organize members into hierarchies, levels, and attributes. The cube stores the measure (fact) data. The dimensions form the edges of the cube.

About the Characteristics of Relational Star and Snowflake Tables (page 3-14)
> For relational tables, the dimensional model has historically been implemented as a star or snowflake schema. Dimension tables (which contain information about hierarchies, levels, and attributes) join one or more fact tables. Fact tables are the large tables that store quantifiable business measurements (such as Booking or Partner Earning) and typically have foreign keys to the dimension tables. Dimension tables, also known as lookup or reference tables contain the relatively static or descriptive data in the data warehouse.

Characteristics of the OLAP Cubes (page 3-20)
> The default access layer of Oracle Airlines Data Model provides a dimensional perspective of the data using Oracle OLAP cubes.

Oracle Airlines Data Model Installation Guide

Oracle Airlines Data Model Reference

Oracle OLAP User's Guide

3.5.6 Changing an Oracle OLAP Cube in Oracle Airlines Data Model

Common customizations to Oracle Airlines Data Model cubes are changing the dimensions or the measures of the cube.

Since all Oracle Airlines Data Model cubes load data from tables with the `DWA_` prefix, to change the measures or dimensions of one cube, you must perform the following steps:

1. Use the information in *Oracle Airlines Data Model Reference* to identify the DWA_ table from which the OLAP cube is populated.

2. Change the structure of the `DWA_` table identified in Step 1.

3. Change the OLAP cube and cube materialized views to reflect the new structure.

Related Topics:

> *Oracle Airlines Data Model Reference*

3.5.7 Creating a Forecast Cube for Oracle Airlines Data Model

Provides steps to create a forecast cube for Oracle Airlines Data Model.

1. Create a cube to contain the results of the forecast.

Note:

You cannot enable materialized views for an Oracle Airlines Data Model forecast cube.

---

2. Write an OLAP DML forecasting context program.

Related Topics:

Defining New Oracle OLAP Cubes for Oracle Airlines Data Model (page 3-21)
> You can add new OLAP cubes to the `oadm_sys` schema.

Oracle OLAP DML Reference

3.5.8 Choosing a Cube Partitioning Strategy for Oracle Airlines Data Model

Partitioning is a method of physically storing the contents of a cube. It improves the performance of large cubes in the following ways:

- Improves scalability by keeping data structures small. Each partition functions like a smaller measure.

- Keeps the working set of data smaller both for queries and maintenance, since the relevant data is stored together.

- Enables parallel aggregation during data maintenance. Each partition can be aggregated by a separate process.

- Simplifies removal of old data from storage. Old partitions can be dropped, and new partitions can be added.

The number of partitions affects the database resources that can be allocated to loading and aggregating the data in a cube. Partitions can be aggregated simultaneously when sufficient resources have been allocated.

The Cube Partitioning Advisor analyzes the source tables and develops a partitioning strategy. You can accept the recommendations of the Cube Partitioning Advisor, or you can make your own decisions about partitioning.

If your partitioning strategy is driven primarily by life-cycle management considerations, then you should partition the cube on the time dimension. Old time periods can then be dropped as a unit, and new time periods added as a new partition. The Cube Partitioning Advisor has a time option, which recommends a hierarchy and a level in the time dimension for partitioning.

The level on which to partition a cube is determined based on a trade off between load performance and query performance.

Typically, you do not want to partition on too low a level (for example, on the DAY level of a TIME dimension) because if you do then too many partitions must be defined at load time which slows down an initial or historical load. Also, a large number of partitions can result in unusually long Analytic Workspace attach times and slows down the Time Series-based calculations. Also, a Quarterly Cumulative measure (Quarter to Date Measure) needs to access 90 or 91 partitions to calculate a specific value for one Customer and Organization.

All dimension members above the partition level of the partition dimension (including those belonging to nondefault hierarchies) must be present in a single default template. Day level partitioning makes this very heavy because all higher level members are stored in a default template. However, the advantage of partitioning at the DAY level of a TIME dimension is if the OLAP Cube load frequency is daily then you must only load from a new partition in fact table into a single partition in the OLAP cube every day. This greatly improves the load performance because the percentage-based refresh can be enabled if the cube is materialized-view enabled and has materialized-view logs.

Recommendations: Cube Partitioning Strategy

Usually a good compromise between the differing load and query performance requirements is to use an intermediate level like MONTH as the partition level. Time series calculations within a month (week to date, month to date, and so on) are fast and higher level calculation like year to date needs to refer to 12 partitions at most.

Also this way the monthly partition is defined and created only one time (that is during the initial load on first of each month) and is then reused for each subsequent load that month. The aggregation process may be triggered off at the month level (instead of specific day level) and some redundant aggregations (of previously loaded dates of current month) may occur each time but it should result in satisfactory load and query performance.

Related Topics:

Partitioning and Materialized Views (page 3-30)
> Because of the large volume of data held in a data warehouse, partitioning is an extremely useful option when designing a database. Partitioning the fact tables improves scalability, simplifies system administration, and makes it possible to define local indexes that can be efficiently rebuilt. Partitioning the fact tables also improves the opportunity of fast refreshing the materialized view because this may enable partition change tracking refresh on the materialized view.

Indexes and Partitioned Indexes in Oracle Airlines Data Model (page 2-16)
> Indexes are optional structures associated with tables or clusters. In addition to the classical B-tree indexes, bitmap indexes are very common in data warehousing environments.

Oracle OLAP User's Guide

3.5.9 Choosing a Cube Data Maintenance Method for Oracle Airlines Data Model

While developing a dimensional model of your data, it is a good idea to map and load each object immediately after you create it so that you can immediately detect and correct any errors that you made to the object definition or the mapping.

In a production environment, you want to perform routine maintenance as quickly and easily as possible. For this stage, you can choose among data maintenance methods. You can refresh all the cubes using the Maintenance Wizard. This wizard enables you to refresh a cube immediately, or submit the refresh as a job to the Oracle job queue, or generate a PL/SQL script. You can run the script manually or using a scheduling utility, such as Oracle Enterprise Manager Scheduler or the DBMS_SCHEDULER PL/SQL package. The generated script calls the BUILD procedure of the DBMS_CUBE PL/SQL package. You can modify this script or develop one from the start using this package.

The data for a partitioned cube is loaded and aggregated in parallel when multiple processes have been allocated to the build. You can view this in the build log.

In addition, each cube can support these data maintenance methods:

- Custom cube scripts

- Cube materialized views

If you are defining cubes to replace existing materialized views, then you use the materialized views as an integral part of data maintenance.

> **Note:**
>
> Materialized view capabilities restrict the types of analytics that can be performed by a custom cube script.

> **Note:**
>
> See the following OBE tutorial for an example of how Oracle uses cube materialized views for transparent access to a relational star schema:
>
> • "Querying OLAP 11*g* Cubes"
>
> To access the tutorial, open the Oracle Learning Library in your browser by following the instructions in "Oracle Technology Network (page xiv)"; and, then, search for the tutorial by name.

Related Topics:

Refresh options vary by the type of materialized view.

Oracle OLAP User's Guide

3.6 Materialized Views in Oracle Airlines Data Model

Materialized views are query results that are stored or "materialized" in advance as schema objects. From a physical design point of view, materialized views resemble tables or partitioned tables and behave like indexes in that they are used transparently and improve performance.

In the past, organizations using summaries spent a significant amount of time and effort creating summaries manually, identifying which summaries to create, indexing the summaries, updating them, and advising their users on which summaries to use. Using materialized views, a database administrator creates one or more materialized views, which are the equivalent of a summary. Thus, the workload of the database administrator is eased and the user no longer needs to be aware of the summaries that had been defined. Instead, the end user queries the tables and views at the detail data level. The query rewrite mechanism automatically rewrites the SQL query to use the summary tables and reduces response time for returning results from the query.

Materialized views improve query performance by precalculating expensive join and aggregation operations on the database before executing and storing the results in the database. The query optimizer automatically recognizes when an existing materialized view can and should be used to satisfy a request.

The default Oracle Airlines Data Model defines materialized views. In the default `oadm_sys` schema, you can identify these materialized views by looking for objects with the prefixes listed in the following table.

Table 3-1   Naming Conventions for Materialized Views

Prefix	Description
CB$	An OLAP cube enhanced with materialized view capabilities. **Note:** Do `not` report or query against these objects. Instead access the relational view of an OLAP cube (that is, the object with the `_VIEW` suffix).

Table 3-1   (Cont.) Naming Conventions for Materialized Views

Prefix	Description
DMV_	Data mining source materialized views.

Types of Materialized Views and Refresh options (page 3-27)
> Refresh options vary by the type of materialized view.

Choosing Indexes for Materialized Views (page 3-29)
> The two most common operations on a materialized view are query execution and fast refresh, and each operation has different performance requirements.

Partitioning and Materialized Views (page 3-30)
> Because of the large volume of data held in a data warehouse, partitioning is an extremely useful option when designing a database. Partitioning the fact tables improves scalability, simplifies system administration, and makes it possible to define local indexes that can be efficiently rebuilt. Partitioning the fact tables also improves the opportunity of fast refreshing the materialized view because this may enable partition change tracking refresh on the materialized view.

Compressing Materialized Views (page 3-31)
> Using data compression for a materialized view can provide dramatic performance improvement.

Related Topics:

Characteristics of the OLAP Cubes in Oracle Airlines Data Model (page 3-20)
> The default access layer of Oracle Airlines Data Model provides a dimensional perspective of the data using Oracle OLAP cubes.

Oracle Airlines Data Model Reference

3.6.1 Types of Materialized Views and Refresh options

Refresh options vary by the type of materialized view.

Refresh Options for Materialized Views with Aggregates (page 3-28)
> In data warehouses, materialized views normally contain aggregates. The _DWA tables in the default Oracle Airlines Data Model are this type of materialized view.

Refresh Options for Materialized Views Containing Only Joins (page 3-28)
> Some materialized views contain only joins and no aggregates (for example, when a materialized view is created that joins the sales table to the times and customers tables). The advantage of creating this type of materialized view is that expensive joins are precalculated.

Refresh Options for Nested Materialized Views (page 3-29)
> A nested materialized view is a materialized view whose definition is based on another materialized view. A nested materialized view can reference other relations in the database in addition to referencing materialized views.

Related Topics:

Oracle OLAP User's Guide

3.6.1.1 Refresh Options for Materialized Views with Aggregates

In data warehouses, materialized views normally contain aggregates. The `_DWA` tables in the default Oracle Airlines Data Model are this type of materialized view.

For a materialized view with aggregates, and for fast refresh:

- The `SELECT` list must contain all of the `GROUP BY` columns (if present)

- There must be a `COUNT(*)` and a `COUNT(`*column*`)` on any aggregated columns.

- Materialized view logs must be present on all tables referenced in the query that defines the materialized view. The valid aggregate functions are: `SUM`, `COUNT(`*x*`)`, `COUNT(*)`, `AVG`, `VARIANCE`, `STDDEV`, `MIN`, and `MAX`, and the expression to be aggregated can be any SQL value expression.

Fast refresh for a materialized view containing joins and aggregates is possible after any type of DML to the base tables (direct load or conventional `INSERT`, `UPDATE`, or `DELETE`).

You can define that the materialized view be refreshed `ON COMMIT` or `ON DEMAND`. A `REFRESH ON COMMIT` materialized view is automatically refreshed when a transaction that does DML to a materialized view's detail tables commits.

When you specify `REFRESH ON COMMIT`, the table commit can take more time than if you have not. This is because the refresh operation is performed as part of the commit process. Therefore, this method may not be suitable if many users are concurrently changing the tables upon which the materialized view is based.

3.6.1.2 Refresh Options for Materialized Views Containing Only Joins

Some materialized views contain only joins and no aggregates (for example, when a materialized view is created that joins the sales table to the times and customers tables). The advantage of creating this type of materialized view is that expensive joins are precalculated.

Fast refresh for a materialized view containing only joins is possible after any type of DML to the base tables (direct-path or conventional `INSERT`, `UPDATE`, or `DELETE`).

A materialized view containing only joins can be defined to be refreshed `ON COMMIT` or `ON DEMAND`. If it is `ON COMMIT`, the refresh is performed at commit time of the transaction that does DML on the materialized view's detail table.

If you specify `REFRESH FAST`, Oracle Database performs further verification of the query definition to ensure that fast refresh can be performed if any of the detail tables change. These additional checks are:

- A materialized view log must be present for each detail table unless the table supports partition change tracking. Also, when a materialized view log is required, the `ROWID` column must be present in each materialized view log.

- The rowids of all the detail tables must appear in the `SELECT` list of the materialized view query definition.

If some of these restrictions are not met, you can create the materialized view as `REFRESH FORCE` to take advantage of fast refresh when it is possible. If one table does

not meet all of the criteria, but the other tables do the materialized view is still fast refreshable with respect to the other tables for which all the criteria are met.

To achieve an optimally efficient refresh:

- Ensure that the defining query does not use an outer join that behaves like an inner join. If the defining query contains such a join, consider rewriting the defining query to contain an inner join.

- If the materialized view contains *only* joins, the ROWID columns for each table (and each instance of a table that occurs multiple times in the FROM list) must be present in the SELECT list of the materialized view.

- If the materialized view has remote tables in the FROM clause, all tables in the FROM clause must be located on that same site. Further, ON COMMIT refresh is not supported for materialized view with remote tables. Except for SCN-based materialized view logs, materialized view logs must be present on the remote site for each detail table of the materialized view and ROWID columns must be present in the SELECT list of the materialized view.

3.6.1.3 Refresh Options for Nested Materialized Views

A nested materialized view is a materialized view whose definition is based on another materialized view. A nested materialized view can reference other relations in the database in addition to referencing materialized views.

In a data warehouse, you typically create many aggregate views on a single join (for example, rollups along different dimensions). Incrementally maintaining these distinct materialized aggregate views can take a long time, because the underlying join must be performed many times.

Using nested materialized views, you can create multiple single-table materialized views based on a joins-only materialized view and the join is performed just one time. In addition, optimizations can be performed for this class of single-table aggregate materialized view and thus refresh is very efficient.

Some types of nested materialized views cannot be fast refreshed. Use EXPLAIN_MVIEW to identify those types of materialized views.

You can refresh a tree of nested materialized views in the appropriate dependency order by specifying the nested =TRUE parameter with the DBMS_MVIEW.REFRESH parameter.

3.6.2 Choosing Indexes for Materialized Views

The two most common operations on a materialized view are query execution and fast refresh, and each operation has different performance requirements.

- Query execution might need to access any subset of the materialized view key columns, and might need to join and aggregate over a subset of those columns. Consequently, for best performance, create a single-column bitmap index on each materialized view key column.

- For materialized views containing only joins using fast refresh, create indexes on the columns that contain the rowids to improve the performance of the refresh operation.

- If a materialized view using aggregates is fast refreshable, then an index appropriate for the fast refresh procedure is created unless USING NO INDEX is specified in the CREATE MATERIALIZED VIEW statement.

Related Topics:

Indexes are optional structures associated with tables or clusters. In addition to the classical B-tree indexes, bitmap indexes are very common in data warehousing environments.

3.6.3 Partitioning and Materialized Views

Because of the large volume of data held in a data warehouse, partitioning is an extremely useful option when designing a database. Partitioning the fact tables improves scalability, simplifies system administration, and makes it possible to define local indexes that can be efficiently rebuilt. Partitioning the fact tables also improves the opportunity of fast refreshing the materialized view because this may enable partition change tracking refresh on the materialized view.

Partitioning a materialized view has the same benefits as partitioning fact tables. When a materialized view is partitioned a refresh procedure can use parallel DML in more scenarios and partition change tracking-based refresh can use truncate partition to efficiently maintain the materialized view.

Using Partition Change Tracking

It is possible and advantageous to track freshness to a finer grain than the entire materialized view. The ability to identify which rows in a materialized view are affected by a certain detail table partition, is known as partition change tracking. When one or more of the detail tables are partitioned, it may be possible to identify the specific rows in the materialized view that correspond to a modified detail partition(s). those rows become stale when a partition is modified while all other rows remain fresh.

You can use partition change tracking to identify which materialized view rows correspond to a particular partition. Partition change tracking is also used to support fast refresh after partition maintenance operations on detail tables. For instance, if a detail table partition is truncated or dropped, the affected rows in the materialized view are identified and deleted. Identifying which materialized view rows are fresh or stale, rather than considering the entire materialized view as stale, allows query rewrite to use those rows that refresh while in `QUERY_REWRITE_INTEGRITY = ENFORCED` or `TRUSTED` modes.

Several views, such as `DBA_MVIEW_DETAIL_PARTITION`, detail which partitions are stale or fresh. Oracle does not rewrite against partial stale materialized views if partition change tracking on the changed table is enabled by the presence of join dependent expression in the materialized view.

To support partition change tracking, a materialized view must satisfy the following requirements:

- At least one detail table referenced by the materialized view must be partitioned.

- Partitioned tables must use either range, list, or composite partitioning.

- The top level partition key must consist of only a single column.

- The materialized view must contain either the partition key column or a partition marker or `ROWID` or join dependent expression of the detail table.

- If you use a `GROUP BY` clause, the partition key column or the partition marker or `ROWID` or join dependent expression must be present in the `GROUP BY` clause.

- If you use an analytic window function or the MODEL clause, the partition key column or the partition marker or ROWID or join dependent expression must be present in their respective PARTITION BY subclauses.

- Data modifications can only occur on the partitioned table. If partition change tracking refresh is being done for a table which has join dependent expression in the materialized view, then data modifications should not have occurred in any of the join dependent tables.

- The COMPATIBILITY initialization parameter must be a minimum of 9.0.0.0.0.

- Partition change tracking is not supported for a materialized view that refers to views, remote tables, or outer joins.

Related Topics:

Partitioned Tables in the Oracle Airlines Data Model (page 2-17)
> Partitioning allows a table, index or index-organized table to be subdivided into smaller pieces. Each piece of the database object is called a partition. Each partition has its own name, and may optionally have its own storage characteristics. From the perspective of a database administrator, a partitioned object has multiple pieces that can be managed either collectively or individually. This gives the administrator considerable flexibility in managing partitioned objects. However, from the perspective of the application, a partitioned table is identical to a nonpartitioned table. No modifications are necessary when accessing a partitioned table using SQL DML commands.

Indexes and Partitioned Indexes in Oracle Airlines Data Model (page 2-16)
> Indexes are optional structures associated with tables or clusters. In addition to the classical B-tree indexes, bitmap indexes are very common in data warehousing environments.

Choosing a Cube Partitioning Strategy for Oracle Airlines Data Model (page 3-24)
> Partitioning is a method of physically storing the contents of a cube. It improves the performance of large cubes in the following ways:

Oracle Database VLDB and Partitioning Guide

3.6.4 Compressing Materialized Views

Using data compression for a materialized view can provide dramatic performance improvement.

Consider data compression when using highly redundant data, such as tables with many foreign keys. In particular, likely candidates are materialized views created with the ROLLUP clause.

Related Topics:

Data Compression in Oracle Airlines Data Model (page 2-12)
> A key decision that you must make is whether to compress your data. Using table compression reduces disk and memory usage, often resulting in a better scale-up performance for read-only operations. Table compression can also speed up query execution by minimizing the number of round trips required to retrieve data from the disks. Compressing data however imposes a performance penalty on the load

speed of the data. Most of the base tables in the Oracle Airlines Data
Model are compressed tables.

Aggregate Tables in Oracle Airlines Data Model (page 3-11)
Aggregate tables are tables that aggregate or "roll up" the data to one
level higher than a base or derived table. The aggregate tables in the
default Oracle Airlines Data Model have a `DWA_` prefix. These aggregate
tables provide a view of the data similar to the view provided by a fact
table in a snowflake schema.

4

ETL Implementation and Customization

Discusses the ETL (extraction, transformation and loading) programs you use to populate an Oracle Airlines Data Model warehouse.

The Role of ETL in the Oracle Airlines Data Model (page 4-1)
> The three layers in Oracle Airlines Data Model warehouse environment are: the optional staging layer, the foundation layer, and the access layer. You use two types of ETL (extraction, transformation and loading) to populate these layers.

ETL for the Foundation Layer of an Oracle Airlines Data Model Warehouse (page 4-2)
> ETL that populates the foundation layer of an Oracle Airlines Data Model warehouse (that is, the base, reference, and lookup tables) with data from an OLTP system is known as source-ETL.

Customizing Intra-ETL for Oracle Airlines Data Model (page 4-10)
> Intra-ETL is delivered as a component of Oracle Airlines Data Model. This intra-ETL is delivered as a PL/SQL package named `PKG_INTRA_ETL_PROCESS` which is a complete intra-ETL process that has procedures that populate the access layer.

Performing an Initial Load of an Oracle Airlines Data Model Warehouse (page 4-11)
> Performing an initial load of an Oracle Airlines Data Model is a multistep process:

Refreshing the Data in an Oracle Airlines Data Model Warehouse (page 4-13)
> Describes how to perform an initial load of an Oracle Airlines Data Model data warehouse. After this initial load, you must load new data into your Oracle Airlines Data Model data warehouse regularly so that it can serve its purpose of facilitating business analysis.

Managing Errors During Oracle Airlines Data Model Intra-ETL Execution (page 4-17)
> Describes information about identifying and managing errors during intra-ETL execution.

4.1 The Role of ETL in the Oracle Airlines Data Model

The three layers in Oracle Airlines Data Model warehouse environment are: the optional staging layer, the foundation layer, and the access layer. You use two types of ETL (extraction, transformation and loading) to populate these layers.

- **Source-ETL:** ETL that populates the staging layer (if any) and the foundation layer (that is, the base, reference, and lookup tables) with data from the OLTP system is known as source ETL.

Oracle Airlines Data Model does *not* include source-ETL scripts. You must create source-ETL yourself using your understanding of your OLTP system and your customized Oracle Airlines Data Model.

- **Intra-ETL:** ETL that populates the access layer (that is, the derived tables, aggregate tables, materialized views, OLAP cubes, and data mining models) using the data in the foundation layer is known as intra-ETL.

  Oracle Airlines Data Model *does* include intra-ETL. You can modify the default intra-ETL to populate a customized access layer from a customized foundation layer.

Related Topics:

ETL for the Foundation Layer of an Oracle Airlines Data Model Warehouse (page 4-2)
> ETL that populates the foundation layer of an Oracle Airlines Data Model warehouse (that is, the base, reference, and lookup tables) with data from an OLTP system is known as source-ETL.

Customizing Intra-ETL for Oracle Airlines Data Model (page 4-10)
> Intra-ETL is delivered as a component of Oracle Airlines Data Model. This intra-ETL is delivered as a PL/SQL package named `PKG_INTRA_ETL_PROCESS` which is a complete intra-ETL process that has procedures that populate the access layer.

4.2 ETL for the Foundation Layer of an Oracle Airlines Data Model Warehouse

ETL that populates the foundation layer of an Oracle Airlines Data Model warehouse (that is, the base, reference, and lookup tables) with data from an OLTP system is known as source-ETL.

You populate the foundation layer of an Oracle Airlines Data Model warehouse by writing your own source-ETL scripts using Oracle Warehouse Builder or another ETL tool and then use those scripts to populate the foundation layer.

The following topics provide general information about writing source-ETL:

Source-ETL Design Considerations (page 4-3)
> Keep the following points in mind when designing and writing source-ETL for Oracle Airlines Data Model:

ETL Architecture for Oracle Airlines Data Model Source-ETL (page 4-4)
> ETL first extracts data from the original sources, assures the quality of the data, cleans the data, and makes the data consistent across the original sources. ETL then populates the physical objects with the "clean" data so that query tools, report writers, dashboards, and so on can access the data.

Creating a Source to Target Mapping Document for the Source-ETL (page 4-4)
> Before you begin building your extract systems, create a logical data interface document that maps the relationship between original source fields and target destination fields in the tables. This document ties the very beginning of the ETL system to the very end.

Designing a Plan for Rectifying Source-ETL Data Quality Problems (page 4-5)
> Data cleaning consists of all the steps required to clean and validate the data feeding a table and to apply known business rules to make the data consistent. The perspectives of the cleaning and conforming steps are less about the upside potential of the data and more about containment and control.

Designing Source-ETL Workflow and Jobs Control (page 4-5)
> All data movement among ETL processes are composed of jobs. An ETL workflow executes these jobs in the proper sequence and with regard to the necessary dependencies. General ETL tools, such as Oracle Warehouse Builder, support this kind of workflow, job design, and execution control.

Designing Source-ETL Exception Handling (page 4-6)
> Your ETL tool or your developed mapping scripts generate status and error handling tables.

Writing Source-ETL that Loads Efficiently (page 4-6)
> Whether you are developing mapping scripts and loading into a staging layer or directly into the foundation layer the goal is to get the data into the warehouse in the most expedient manner. To achieve good performance during the load you must begin by focusing on where the data to be loaded resides and how you load it into the database. For example, you should not use a serial database link or a single JDBC connection to move large volumes of data. The most common and preferred mechanism for loading large volumes of data is loading from flat files.

4.2.1 Source-ETL Design Considerations

Keep the following points in mind when designing and writing source-ETL for Oracle Airlines Data Model:

- You can populate the calendar data by using the calendar population scripts provided with Oracle Airlines Data Model

- Populate the tables in the following order:

    1. Lookup tables

    2. Reference tables

    3. Base tables

- Analyze the tables in one category before loading the tables in the next category (for example, analyze the lookup tables before loading the reference tables). Additionally, you must analyze all of the tables loaded by the source-ETL process before executing the intra-ETL processes).

Related Topics:

> *Oracle Airlines Data Model Reference*

> *Oracle Database Administrator's Guide*

4.2.2 ETL Architecture for Oracle Airlines Data Model Source-ETL

ETL first extracts data from the original sources, assures the quality of the data, cleans the data, and makes the data consistent across the original sources. ETL then populates the physical objects with the "clean" data so that query tools, report writers, dashboards, and so on can access the data.

The fundamental services upon which data acquisition is constructed are as follows:

- Data sourcing

- Data movement

- Data transformation

- Data loading

From a logical architecture perspective, there are many different ways to configure these building blocks for delivering data acquisition services. The major architectural styles available that cover a range of options to be targeted within a data warehousing architecture include:

- **Batch Extract, Transform, and Load** and **Batch Extract, Load, Transform, Load**

  Batch Extract, Transform and Load (ETL) and Batch Extract, Load, Transform, Load (ELTL) are the traditional architecture sin data warehouse implementation. The difference between them is where the transformation proceed in or out database.

- **Batch Hybrid Extract, Transform, Load, Transform, Load**

  Batch Hybrid Extract, Transform, Load, Transform, Load (ETLTL) is a hybrid strategy. This strategy provides the most flexibility to remove hand coding approaches to transformation design, apply a metadata-driven approach, and still be able to leverage the data processing capabilities of the enterprise warehouse. In this targeted design, the transformation processing is first performed outside the warehouse as a pre-processing step before loading the staging tables, and then further transformation processing is performed within the data warehouse before the final load into the target tables.

- **Real-time Extract, Transform, Load**

  Real-time Extract, Transform, Load (rETL) is appropriate when service levels for data freshness demand more up-to-date information in the data warehousing environment. In this approach, the OLTP system must actively publish events of interest so that the rETL processes can extract them from a message bus (queue) on a timely basis. A message-based paradigm is used with publish and subscribe message bus structures or point-to-point messaging with reliable queues.

When designing source-ETL for Oracle Airlines Data Model, use the architecture that best meets your business needs.

4.2.3 Creating a Source to Target Mapping Document for the Source-ETL

Before you begin building your extract systems, create a logical data interface document that maps the relationship between original source fields and target destination fields in the tables. This document ties the very beginning of the ETL system to the very end.

Columns in the data mapping document are sometimes combined. For example, the source database, table name, and column name can be combined into a single target column. The information within the concatenated column must be delimited with a period. Regardless of the format, the content of the logical data mapping document has been proven to be the critical element required to sufficiently plan ETL processes.

4.2.4 Designing a Plan for Rectifying Source-ETL Data Quality Problems

Data cleaning consists of all the steps required to clean and validate the data feeding a table and to apply known business rules to make the data consistent. The perspectives of the cleaning and conforming steps are less about the upside potential of the data and more about containment and control.

If there are data quality problems, then build a plan, in agreement with IT and business users, for how to rectify these problems.

Answer the following questions:

- Is data missing?

- Is the data wrong or inconsistent?

- Should the problem be fixed in the source systems?

- Set up the data quality reporting and action program and people responsibility.

Then, set up the following processes and programs:

- Set up a data quality measurement process.

- Set up the data quality reporting and action program and people responsibility.

4.2.5 Designing Source-ETL Workflow and Jobs Control

All data movement among ETL processes are composed of jobs. An ETL workflow executes these jobs in the proper sequence and with regard to the necessary dependencies. General ETL tools, such as Oracle Warehouse Builder, support this kind of workflow, job design, and execution control.

Below are some tips when you design ETL jobs and workflow:

- Use common structure across all jobs (source system to transformer to target data warehouse).

- Have a one-to-one mapping from source to target.

- Define one job per Source table.

- Apply generic job structure and template jobs to allow for rapid development and consistency.

- Use an optimized job design to leverage Oracle load performance based on data volumes.

- Design parameterized job to allow for greater control over job performance and behavior.

- Maximize Jobs parallelism execution.

4.2.6 Designing Source-ETL Exception Handling

Your ETL tool or your developed mapping scripts generate status and error handling tables.

As a general principle, all ETL logs status and errors into a table. You monitor execution status using an ETL tool or by querying this log table directly.

4.2.7 Writing Source-ETL that Loads Efficiently

Whether you are developing mapping scripts and loading into a staging layer or directly into the foundation layer the goal is to get the data into the warehouse in the most expedient manner. To achieve good performance during the load you must begin by focusing on where the data to be loaded resides and how you load it into the database. For example, you should not use a serial database link or a single JDBC connection to move large volumes of data. The most common and preferred mechanism for loading large volumes of data is loading from flat files.

The following topics discuss best practices for ensuring your source-ETL loads efficiently:

Using a Staging Area for Flat Files (page 4-7)
> The area where flat files are stored before being loaded into the staging layer of a data warehouse system is commonly known as staging area. The overall speed of your load is determined by:

Preparing Raw Data Files for Source-ETL (page 4-7)
> To parallelize the data load Oracle Database must be able to logically break up the raw data files into chunks, known as granules. To ensure balanced parallel processing, the number of granules is typically much higher than the number of cluster database processes. At any given point in time, a cluster database process is allocated one granule to work on. After a cluster database process completes working on its granule, another granule is allocated until all of the granules are processed and the data is loaded.

Source-ETL Data Loading Options (page 4-8)
> Oracle offers several data loading options.

Parallel Direct Path Load Source-ETL (page 4-8)
> A direct path load parses the input data according to the description given in the external table definition, converts the data for each input field to its corresponding Oracle data type, then builds a column array structure for the data. These column array structures are used to format Oracle data blocks and build index keys. The newly formatted database blocks are then written directly to the database, bypassing the standard SQL processing engine and the database buffer cache.

Partition Exchange Load for Oracle Airlines Data Model Source-ETL (page 4-9)
> A benefit of partitioning is the ability to load data quickly and easily with minimal impact on the business users by using the `EXCHANGE PARTITION` command. The `EXCHANGE PARTITION` command enables swapping the data in a nonpartitioned table into a particular partition in your partitioned table. The `EXCHANGE PARTITION` command does not physically move data, instead it updates the data dictionary to exchange a pointer from the partition to the table and vice versa.

4.2.7.1 Using a Staging Area for Flat Files

The area where flat files are stored before being loaded into the staging layer of a data warehouse system is commonly known as staging area. The overall speed of your load is determined by:

- How quickly the raw data can be read from a staging area.

- How quickly the raw data can be processed and inserted into the database.

Recommendations: Using a Staging Area

Stage the raw data across as many physical disks as possible to ensure that reading it is not a bottleneck during the load.

Also, if you are using the Exadata Database Machine, the best place to stage the data is in an Oracle Database File System (DBFS) stored on the Exadata storage cells. DBFS creates a mountable cluster file system which can you can use to access files stored in the database. Create the DBFS in a separate database on the Database Machine. This allows the DBFS to be managed and maintained separately from the data warehouse.

Mount the file system using the `DIRECT_IO` option to avoid thrashing the system page cache while moving the raw data files in and out of the file system.

Related Topics:

Oracle Database SecureFiles and Large Objects Developer's Guide

4.2.7.2 Preparing Raw Data Files for Source-ETL

To parallelize the data load Oracle Database must be able to logically break up the raw data files into chunks, known as granules. To ensure balanced parallel processing, the number of granules is typically much higher than the number of cluster database processes. At any given point in time, a cluster database process is allocated one granule to work on. After a cluster database process completes working on its granule, another granule is allocated until all of the granules are processed and the data is loaded.

Recommendations: Preparing Raw Data Files for Source-ETL

Follow these recommendations:

- Deliminate each row using a known character such as a new line or a semicolon. This ensures that Oracle can look inside the raw data file and determine where each row of data begins and ends to create multiple granules within a single file.

- If a file is not position-able and seek-able (for example the file is compressed or zip file), then the files cannot be broken up into granules and the whole file is treated as a single granule. In this case, only one cluster database process can work on the entire file. To parallelize the loading of compressed data files, use multiple compressed data files. The number of compressed data files used determines the maximum parallel degree used by the load.

- When loading multiple data files (compressed or uncompressed):

  - Use a single external table, if at all possible

  - Make the files similar in size

  – Make the size of the files a multiple of 10 MB

- If you must have files of different sizes, list the files from largest to smallest. By default, Oracle assumes that the flat file has the same character set as the database. If this is not the case, specify the character set of the flat file in the external table definition to ensure the proper character set conversions can take place.

4.2.7.3 Source-ETL Data Loading Options

Oracle offers several data loading options.

- Oracle Data Integrator

- External table or SQL*Loader

- Oracle Data Pump (import and export)

- Change Data Capture and Trickle feed mechanisms (such as Oracle Golden Gate)

- Oracle Database Gateways to open systems and mainframes

- Generic Connectivity (ODBC and JDBC)

The approach that you take depends on the source and format of the data you receive.

Recommendations: Loading Flat Files

If you are loading from files into Oracle you have two options: SQL*Loader or external tables.

Using external tables offers the following advantages:

- Allows transparent parallelization inside the database.You can avoid staging data and apply transformations directly on the file data using arbitrary SQL or PL/SQL constructs when accessing external tables. SQL Loader requires you to load the data as-is into the database first.

- Parallelizing loads with external tables enables a more efficient space management compared to SQL*Loader, where each individual parallel loader is an independent database sessions with its own transaction. For highly partitioned tables this could potentially lead to a lot of wasted space.

You can create an external table using the standard `CREATE TABLE` statement. However, to load from flat files the statement must include information about where the flat files reside outside the database. The most common approach when loading data from an external table is to issue a `CREATE TABLE AS SELECT (CTAS)` statement or an `INSERT AS SELECT (IAS)` statement into an existing table.

4.2.7.4 Parallel Direct Path Load Source-ETL

A direct path load parses the input data according to the description given in the external table definition, converts the data for each input field to its corresponding Oracle data type, then builds a column array structure for the data. These column array structures are used to format Oracle data blocks and build index keys. The newly formatted database blocks are then written directly to the database, bypassing the standard SQL processing engine and the database buffer cache.

The key to good load performance is to use direct path loads wherever possible:

- A `CREATE TABLE AS SELECT (CTAS)` statement always uses direct path load.

- A simple `INSERT AS SELECT (IAS)` statement does *not* use direct path load. To achieve direct path load with an IAS statement you must add the `APPEND` hint to the command.

Direct path loads can also run in parallel. To set the parallel degree for a direct path load, either:

- Add the `PARALLEL` hint to the CTAS statement or an IAS statement.

- Set the `PARALLEL` clause on both the external table and the table into which the data is loaded.

  After the parallel degree is set:

  - A `CTAS` statement automatically performs a direct path load in parallel.

  - An `IAS` statement does not automatically perform a direct path load in parallel. To enable an `IAS` statement to perform direct path load in parallel, you must alter the session to enable parallel DML by executing the following statement:

    ```
    alter session enable parallel DML;
    ```

4.2.7.5 Partition Exchange Load for Oracle Airlines Data Model Source-ETL

A benefit of partitioning is the ability to load data quickly and easily with minimal impact on the business users by using the `EXCHANGE PARTITION` command. The `EXCHANGE PARTITION` command enables swapping the data in a nonpartitioned table into a particular partition in your partitioned table. The `EXCHANGE PARTITION` command does not physically move data, instead it updates the data dictionary to exchange a pointer from the partition to the table and vice versa.

Because there is no physical movement of data, an exchange does not generate redo and undo. In other words, an exchange is a sub-second operation and far less likely to impact performance than any traditional data-movement approaches such as `INSERT`.

Recommendations: Partitioning Tables

Partition the larger tables and fact tables in the Oracle Airlines Data Model warehouse.

Note:

The assumption being made in this example is that the data integrity was verified at date extraction time. If you are unsure about the data integrity, omit the `WITHOUT VALIDATION` clause so that the Database checks the validity of the data.

Example 4-1   Using Exchange Partition Statement with a Partitioned Table

Assume that there is a large table called `DWB_PNR`, which is range partitioned by day. At the end of each business day, data from the online airlines system is loaded into the `DWB_PNR` table in the warehouse.

The following steps ensure the daily data gets loaded into the correct partition with minimal impact to the business users of the data warehouse and optimal speed:

1. Create external table for the flat file data coming from the online system

2. Using a `CTAS` statement, create a nonpartitioned table called `tmp_pnr` that has the same column structure as `DWB_PNR` table

3. Build any indexes that are on the `DWB_PNR` table on the `tmp_pnr` table

4. Issue the `EXCHANGE PARTITION` command.

   ```
   Alter table dwb_pnr exchange partition SYS_P1926 with    table tmp_pnr including
   indexes without validation;
   ```

5. Gather optimizer statistics on the newly exchanged partition using incremental statistics.

The `EXCHANGE PARTITION` command in this example, swaps the definitions of the named partition and the `tmp_pnr` table, so the data instantaneously exists in the right place in the partitioned table. Moreover, with the inclusion of the `INCLUDING INDEXES` and `WITHOUT VALIDATION` clauses, Oracle swaps index definitions and does not check whether the data actually belongs in the partition - so the exchange is very quick.

4.3 Customizing Intra-ETL for Oracle Airlines Data Model

Intra-ETL is delivered as a component of Oracle Airlines Data Model. This intra-ETL is delivered as a PL/SQL package named `PKG_INTRA_ETL_PROCESS` which is a complete intra-ETL process that has procedures that populate the access layer.

The `PKG_INTRA_ETL_PROCESS` package is composed of individual sub-process procedures and functions that respect the dependency of each individual program. The main procedures execute in the following order:

1. `Populate_Derived` - Populates the derived (`DWD_`) tables based on the content of the base (`DWB_`) tables.

2. `Populate_Aggregate` - Refreshes all of the aggregate (`DWA_`) tables using data from the dimension (`DWR_`) and derived (`DWD_`) tables.

3. `Populate_Aw` - Loads data from Oracle Airlines Data Model aggregate (`DWA_`) tables into the Oracle Airlines Data Model Analytical Workspace and calculates the forecast data. It reads OLAP ETL parameters from `DWC_OLAP_ETL_PARM` table.

4. `Populate_MINING` - Triggers the data mining models.

The `PKG_INTRA_ETL_PROCESS` package is designed to work with the default Oracle Airlines Data Model. If you have customized the model, you also need to customize the intra-ETL. To customize the intra-ETL delivered with Oracle Airlines Data Model, perform the following steps:

1. Familiarize yourself with the `PKG_INTRA_ETL_PROCESS` package. In particular, understand the procedures, data flows, and maps, and how the procedures use the `DWC_ETL_PARAMETER` and `DWC_OLAP_ETL_PARM` control tables, and the `DWC_INTRA_ETL_PROCESS` and `DWC_INTRA_ETL_ACTIVITY` control tables..

2. Identify the changes that you have made to the Oracle Airlines Data Model and what changes need to be made to the intra-ETL needs to support those model changes.

3. Make a copy of the packages that you need to change. Give the copies a different name.

4. Make the changes you identified in Step 2 to the packages you created in Step 3.

Optionally, you can create new intra-ETL from scratch - either by writing your own PL/SQL code or by using an ETL tool such as Oracle Warehouse Builder.

4.4 Performing an Initial Load of an Oracle Airlines Data Model Warehouse

Performing an initial load of an Oracle Airlines Data Model is a multistep process:

Performing an Initial Load of the Foundation Layer (page 4-11)
> You perform the initial load of the foundation layer using source-ETL that you create.

Performing an Initial Load of the Access Layer (page 4-11)
> You perform several steps for an initial load of the access layer of the Oracle Airlines Data Model warehouse (that is, the derived and aggregate tables, materialized views, OLAP cubes, and data mining models).

4.4.1 Performing an Initial Load of the Foundation Layer

You perform the initial load of the foundation layer using source-ETL that you create.

Related Topics:

ETL for the Foundation Layer of an Oracle Airlines Data Model Warehouse (page 4-2)
> ETL that populates the foundation layer of an Oracle Airlines Data Model warehouse (that is, the base, reference, and lookup tables) with data from an OLTP system is known as source-ETL.

4.4.2 Performing an Initial Load of the Access Layer

You perform several steps for an initial load of the access layer of the Oracle Airlines Data Model warehouse (that is, the derived and aggregate tables, materialized views, OLAP cubes, and data mining models).

1. Update the parameters in `DWC_ETL_PARAMETER` control table in the `oadm_sys` schema so that the ETL can use this information (that is, the beginning and end date of the ETL period) when loading the data into the access layer.

For an initial load of an Oracle Airlines Data Model warehouse, specify the values shown in the following table:

Table 4-1   DWC_ETL_PARAMETER Parameter Descriptions

Columns	Value
PROCESS_NAME	'OADM-INTRA-ETL'
FROM_DATE_ETL	The beginning date of the ETL period.
TO_DATE_ETL	The ending date of the ETL period.

2. Update the Oracle Airlines Data Model OLAP ETL parameters in `DWC_OLAP_ETL_PARM` control table in the `oadm_sys` schema to specify the build method and other build characteristics so that the ETL can use this information when loading the OLAP cube data.

For an initial load of the analytic workspace, specify values following the guidelines in Table 4-2 (page 4-12).

Table 4-2   DWC_OLAP_ETL_PARAMETER Values for OLAP ETL Parameters

Column Name	Value
PROCESS_NAME	'OADM-INTRA-ETL'
BUILD_METHOD	C which specifies a complete refresh which clears all dimension values before loading.
CUBENAME	One of the following values that specifies the cubes you want to build: • ALL specifies a build of the cubes in the Oracle Airlines Data Model analytic workspace. • *cubename*[[|*cubename*]...] specifies one or more cubes to build.
MAXJOBQUEUES	A decimal value that specifies the number of parallel processes to allocate to this job. (Default value is 4.) The value that you specify varies depending on the setting of the JOB_QUEUE_PROCESSES database initialization parameter.
CALC_FCST	One of the following values depending on whether you want to calculate forecast cubes: • Y specifies calculate forecast cubes. • N specifies do not calculate forecast cubes.
NO_FCST_YRS	If the value for the CALC_FCST column is Y, specify a decimal value that specifies how many years forecast data you want to calculate; otherwise, specify NULL.
FCST_MTHD	If the value for the CALC_FCST column is Y, then specify AUTO; otherwise, specify NULL.
FCST_ST_YR	If the value for the CALC_FCST column is Y, then specify value specified as *yyyy* which is the "start business year" of a historical period; otherwise, specify NULL.
FCST_END_YR	If the value for the CALC_FCST column is Y, then specify value specified as *yyyy* which is the "end business year" of a historical period; otherwise, specify NULL.
OTHER1	Specify NULL.
OTHER2	Specify NULL.

3. Execute the intra-ETL.

See Also:

Oracle Airlines Data Model Reference for more information on the DWC_ETL_PARAMETER control table

---

Executing the Default Oracle Airlines Data Model Intra-ETL (page 4-13)
You execute the PKG_INTRA_ETL_PROCESS process flow from an Oracle client tool (for example, SQL Plus).

Related Topics:

Describes how to perform an initial load of an Oracle Airlines Data Model data warehouse. After this initial load, you must load new data into your Oracle Airlines Data Model data warehouse regularly so that it can serve its purpose of facilitating business analysis.

4.4.2.1 Executing the Default Oracle Airlines Data Model Intra-ETL

You execute the `PKG_INTRA_ETL_PROCESS` process flow from an Oracle client tool (for example, SQL Plus).

Enter the following command to execute the Intra-ETL process flow:

```
EXEC PKG_INTRA_ETL_PROCESS.RUN( )
```

The PL/SQL code executed by `PKG_INTRA_ETL_PROCESS`:

- Reads the values from the `DWC_ETL_PARAMETER` and `DWC_OLAP_ETL_PARM` control tables in the `oadm_sys` schema before executing the mappings in the correct order.

- The result of each table loading are tracked in the `DWC_INTRA_ETL_PROCESS` and `DWC_INTRA_ETL_ACTIVITY` control tables.

Executing the `PKG_INTRA_ETL_PROCESS` from within an Oracle client provides the ability to monitor the execution of the process. However, you can simply execute the `RUN` procedure within the `PKG_INTRA_ETL_PROCESS` PL/SQL package.

In either case, you can execute the intra-ETL explicitly or invoke its execution in some other program or process (for example, the source-ETL process after its successful execution) or through a predefined schedule (for example, using Oracle Job Scheduling feature and so on).

Related Topics:

The schema control tables in the `oadm_sys` schema: `DWC_INTRA_ETL_PROCESS` and `DWC_INTRA_ETL_ACTIVITY`, monitor the execution of the intra-ETL process.

Describes the steps for recovering from errors when running the `PKG_INTRA_ETL_PROCESS` package.

Describes how to check, monitor, and troubleshoot the performance of the intra-ETL process.

4.5 Refreshing the Data in an Oracle Airlines Data Model Warehouse

Describes how to perform an initial load of an Oracle Airlines Data Model data warehouse. After this initial load, you must load new data into your Oracle Airlines Data Model data warehouse regularly so that it can serve its purpose of facilitating business analysis.

First, perform an initial load of an Oracle Airlines Data Model warehouse. After this initial load, you must load new data into your Oracle Airlines Data Model warehouse regularly so that it can serve its purpose of facilitating business analysis.

To load new data into your Oracle Airlines Data Model warehouse, you extract the data from one or more operational systems and copy that data into the warehouse. The challenge in data warehouse environments is to integrate, rearrange and consolidate large volumes of data over many systems, thereby providing a new unified information base for business intelligence.

The successive loads and transformations must be scheduled and processed in a specific order and are determined by your business needs. Depending on the success or failure of the operation or parts of it, the result must be tracked and subsequent, alternative processes might be started.

You can do a full incremental load of the Oracle Airlines Data Model warehouse, or you can refresh the data sequentially.:

You must also manage errors during the execution of the intra-ETL.

Refreshing the Foundation Layer of Oracle Airlines Data Model Warehouse (page 4-14)
>   You refresh the foundation layer using source-ETL scripts that you wrote using Oracle Warehouse Builder or another ETL tool.

Refreshing the Access Layer of an Oracle Airlines Data Model Warehouse (page 4-14)
>   Refreshing the access layer of an Oracle Airlines Data Model is a multi-step process. You can do a full incremental load of the access layer at once by executing the `PKG_INTRA_ETL_PROCESS` package.

Related Topics:

Performing an Initial Load of the Access Layer (page 4-11)
>   You perform several steps for an initial load of the access layer of the Oracle Airlines Data Model warehouse (that is, the derived and aggregate tables, materialized views, OLAP cubes, and data mining models).

Managing Errors During Oracle Airlines Data Model Intra-ETL Execution (page 4-17)
>   Describes information about identifying and managing errors during intra-ETL execution.

4.5.1 Refreshing the Foundation Layer of Oracle Airlines Data Model Warehouse

You refresh the foundation layer using source-ETL scripts that you wrote using Oracle Warehouse Builder or another ETL tool.

Related Topics:

ETL for the Foundation Layer of an Oracle Airlines Data Model Warehouse (page 4-2)
>   ETL that populates the foundation layer of an Oracle Airlines Data Model warehouse (that is, the base, reference, and lookup tables) with data from an OLTP system is known as source-ETL.

4.5.2 Refreshing the Access Layer of an Oracle Airlines Data Model Warehouse

Refreshing the access layer of an Oracle Airlines Data Model is a multi-step process. You can do a full incremental load of the access layer at once by executing the `PKG_INTRA_ETL_PROCESS` package.

Refreshing the access layer of an Oracle Airlines Data Model is a multi-step process. You can do a full incremental load of the access layer all at one time by executing the `PKG_INTRA_ETL_PROCESS` package, or you can refresh the data sequentially:

> Refreshing the Access Layer Relational Tables in the Oracle Airlines Data Model (page 4-15)
>> After you refresh the foundation layer of Oracle Airlines Data Model, perform these steps to refresh the tables and views in the access layer.

> Refreshing Oracle Airlines Data Model OLAP Cubes (page 4-16)
>> On a scheduled basis you must update the OLAP cube data with the relational data that has been added to the Oracle Airlines Data Model data warehouse since the initial load of the OLAP cubes.

> Refreshing Oracle Airlines Data Model Data Mining Models (page 4-16)
>> The `PKG_INTRA_ETL_PROCESS.Populate_Mining` procedure triggers the data mining model refreshment as part of the initial load of the warehouse. After the initial load of the warehouse, Oracle recommends that you refresh the data mining models monthly.

Related Topics:

> Executing the Default Oracle Airlines Data Model Intra-ETL (page 4-13)
>> You execute the `PKG_INTRA_ETL_PROCESS` process flow from an Oracle client tool (for example, SQL Plus).

> Managing Errors During Oracle Airlines Data Model Intra-ETL Execution (page 4-17)
>> Describes information about identifying and managing errors during intra-ETL execution.

4.5.2.1 Refreshing the Access Layer Relational Tables in the Oracle Airlines Data Model

After you refresh the foundation layer of Oracle Airlines Data Model, perform these steps to refresh the tables and views in the access layer.

You can refresh only the relational tables and views in the access layer of an Oracle Airlines Data Model by performing the following steps:

1. Update the parameters of the `DWC_ETL_PARAMETER` control table in the `oadm_sys` schema. For an incremental load of an Oracle Airlines Data Model warehouse, specify the values shown in the following table (that is, the beginning and end date of the ETL period):

Table 4-3   DWC_ETL_PARAMETER Column Descriptions

Columns	Value
PROCESS_NAME	'OADM-INTRA-ETL'
FROM_DATE_ETL	The beginning date of the ETL period.
TO_DATE_ETL	The ending date of the ETL period.

2. Refresh the tables by executing the following procedures in the `PKG_INTRA_ETL_PROCESS` PL/SQL package. Execute the procedures in the following order:

    a.   `Populate_Dimension`

    b.   `Populate_Derived`

    c.   `Populate_Aggregate`

See Also:

Oracle Airlines Data Model Reference for more information on the `DWC_ETL_PARAMETER` control table

---

Related Topics:

You execute the `PKG_INTRA_ETL_PROCESS` process flow from an Oracle client tool (for example, SQL Plus).

4.5.2.2 Refreshing Oracle Airlines Data Model OLAP Cubes

On a scheduled basis you must update the OLAP cube data with the relational data that has been added to the Oracle Airlines Data Model data warehouse since the initial load of the OLAP cubes.

Perform these steps to refresh *only* the OLAP cube data in the Oracle Airlines Data Model warehouse:

1. Ensure that the underlying aggregate tables are refreshed.

2. Execute the `PKG_INTRA_ETL_PROCESS.Populate_Aw` procedure to load the cube data.

3. If necessary, recover from errors that happen during the execution of `Populate_Aw` by performing the following steps:

    a.   Change the value of the `BUILD_METHOD` column of the `DWC_OLAP_ETL_PARM` table to `"C"`.

    b.   Re-execute `PKG_INTRA_ETL_PROCESS.Populate_Aw`.

Related Topics:

After you refresh the foundation layer of Oracle Airlines Data Model, perform these steps to refresh the tables and views in the access layer.

4.5.2.3 Refreshing Oracle Airlines Data Model Data Mining Models

The `PKG_INTRA_ETL_PROCESS.Populate_Mining` procedure triggers the data mining model refreshment as part of the initial load of the warehouse. After the initial load of the warehouse, Oracle recommends that you refresh the data mining models monthly.

After you have refreshed the OLAP cubes, you can also refresh the data mining models. In this case, the way you refresh a data mining model varies depending on whether you want to refresh all of the data mining models or only one data mining model:

- To manually refresh *all* data mining models, call the following procedure:

  `oadm_sys.pkg_oadm_mining.refresh_model(p_month_code,p_process_no)`

  This procedure performs the following tasks for each data mining model:

  1. Refreshes the source materialized views for the data mining model based on the latest data from `oadm_sys` schema.

  2. Trains each data mining model on the new training data.

  3. Applies each data mining model onto the new apply data set.

- To manually re-create only *one* data mining model, you can call the corresponding `oadm_sys.pkg_oadm_mining.create_` procedure.

  For example, to re-create the Customer Life Time Value regression data mining model, call the following procedure:

  `oadm_sys.pkg_oadm_mining.create_cust_ltv_svm_rgrsn(p_month_cd);`

Related Topics:

Tutorial: Customizing the Customer Life Time Value Prediction Data Mining Model (page 3-5)
> After you have populated Oracle Airlines Data Model foundation layer and executed the intra-ETL to populate derived tables, you can leverage the prebuilt Oracle Airlines Data Model data mining models for more advanced analysis and predictions.

Troubleshooting Data Mining Model Creation (page 4-20)
> After the data mining models are created, check the error log in `oadm_sys.dwc_intra_etl_activity` table.

4.6 Managing Errors During Oracle Airlines Data Model Intra-ETL Execution

Describes information about identifying and managing errors during intra-ETL execution.

Monitoring the Execution of the Intra-ETL Process (page 4-17)
> The schema control tables in the `oadm_sys` schema: `DWC_INTRA_ETL_PROCESS` and `DWC_INTRA_ETL_ACTIVITY`, monitor the execution of the intra-ETL process.

Recovering an Intra ETL Process (page 4-18)
> Describes the steps for recovering from errors when running the `PKG_INTRA_ETL_PROCESS` package.

Troubleshooting Intra-ETL Performance (page 4-19)
> Describes how to check, monitor, and troubleshoot the performance of the intra-ETL process.

4.6.1 Monitoring the Execution of the Intra-ETL Process

The schema control tables in the `oadm_sys` schema: `DWC_INTRA_ETL_PROCESS` and `DWC_INTRA_ETL_ACTIVITY`, monitor the execution of the intra-ETL process.

Each normal run of the `PKG_INTRA_ETL_PROCESS` package (as opposed to an error-recovery run) performs the following steps:

1. Inserts a record into the `DWC_INTRA_ETL_PROCESS` table with a consistently increasing system generated unique process key, `SYSDATE` as process start time, `RUNNING` as the process status, input date range in the fields `FROM_DATE_ETL` and `TO_DATE_ETL`.

2. Invokes each of the individual `PKG_INTRA_ETL_PROCESS` procedures in the appropriate order of dependency. Before the invocation of each program, the procedure inserts a record into the Intra-ETL Activity detail table `DWC_INTRA_ETL_ACTIVITY` with a system generated unique activity key, the process key value corresponding to the Intra-ETL process, individual program name as the `Activity Name`, a suitable activity description, `SYSDATE` as activity start time, `RUNNING` as the activity status.

3. Updates the corresponding record in the `DWC_INTRA_ETL_ACTIVITY` table for the activity end time and activity status after the completion of each individual ETL program (either successfully or with errors. For successful completion of the activity, the procedure updates the status as `'COMPLETED-SUCCESS'`. If an error occurs, the procedure updates the activity status as `'COMPLETED-ERROR'`, and also updates the corresponding error detail in the `ERROR_DTL` column.

4. Updates the record corresponding to the process in the `DWC_INTRA_ETL PROCESS` table for the process end time and status, after the completion of all individual intra-ETL programs. If all the individual programs succeed, the procedure updates the status to `'COMPLETED-SUCCESS'`, otherwise it updates the status to `'COMPLETED-ERROR'`.

You can monitor the execution state of the intra-ETL, including current process progress, time taken by individual programs, or the complete process by viewing the contents of the `DWC_INTRA_ETL_PROCESS` and `DWC_INTRA_ETL_ACTIVITY` tables corresponding to the maximum process key. Monitoring can be done both during and after the execution of the intra-ETL procedure.

4.6.2 Recovering an Intra ETL Process

Describes the steps for recovering from errors when running the `PKG_INTRA_ETL_PROCESS` package.

To recover a run of the `PKG_INTRA_ETL_PROCESS` package:

1. Identify the errors by looking at the corresponding error details that are tracked against the individual programs in the `DWC_INTRA_ETL_ACTIVITY` table.

2. Correct the causes of the errors.

3. Re-execute the `PKG_INTRA_ETL_PROCESS` package.

The procedures in the `PKG_INTRA_ETL_PROCESS` package identify whether it is a normal run or recovery run by referring the `DWC_INTRA_ETL_ACTIVITY` table. During a recovery run, `PKG_INTRA_ETL_PROCESS` executes only the procedures needed for recovery. For example, for a derived population error as a part of the previous run, this recovery run executes the individual derived population programs which produced errors in the previous run. After their successful completion, the run executes the aggregate population programs and materialized view refresh in the appropriate order.

In this way, the intra-ETL error recovery is almost transparent, without involving the data warehouse or ETL administrator. The administrator must correct the causes of the errors and re-invoke the `PKG_INTRA_ETL_PROCESS` package. `PKG_INTRA_ETL_PROCESS` identifies and executes the programs that generated errors.

4.6.3 Troubleshooting Intra-ETL Performance

Describes how to check, monitor, and troubleshoot the performance of the intra-ETL process.

Checking the Execution Plan (page 4-19)
> Use SQLDeveloper or other tools to view the package body of the code generated by Oracle Warehouse Builder.

Monitoring PARALLEL DML Executions (page 4-19)
> Describes how to check that you are running mapping in parallel mode.

Troubleshooting Data Mining Model Creation (page 4-20)
> After the data mining models are created, check the error log in `oadm_sys.dwc_intra_etl_activity` table.

4.6.3.1 Checking the Execution Plan

Use SQLDeveloper or other tools to view the package body of the code generated by Oracle Warehouse Builder.

For example, perform the following steps to examine a map:

1. Copy out the main query statement from code viewer.

   Copy from `"CURSOR "AGGREGATOR_c" IS ...."` to end of the query, which is right above another `"CURSOR "AGGREGATOR_c$1" IS"`.

2. In SQLDeveloper worksheet, issue the following statement to turn on the parallel DML:

   ```
   Alter session enable parallel dml;
   ```

3. Paste the main query statement into another SQL Developer worksheet and view the execution plan by clicking `F6`.

   Carefully examine the execution plan to make the mapping runs according to a valid plan.

4.6.3.2 Monitoring PARALLEL DML Executions

Describes how to check that you are running mapping in parallel mode.

Check the execution mode by executing the following SQL statement to count the executed "Parallel DML/Query" statement:

```
column name format a50
column value format 999,999
SELECT NAME, VALUE
FROM GV$SYSSTAT
WHERE UPPER (NAME) LIKE '%PARALLEL OPERATIONS%'
  OR UPPER (NAME) LIKE '%PARALLELIZED%'
  OR UPPER (NAME) LIKE '%PX%'
;
```

If you run the mapping in parallel mode, then the "DML statements parallelized" increases by 1 (one) every time the mapping is invoked. If you do not see this increase, then the mapping was not invoked as "parallel DML".

If you see "queries parallelized" increased by 1 (one) instead, then typically this means that the SELECT statement inside of the INSERT was parallelized, but that INSERT itself was not parallelized.

Related Topics:

Parallel Execution in Oracle Airlines Data Model (page 2-20)
> Parallel Execution enables a database task to be parallelized or divided into smaller units of work, thus allowing multiple processes to work concurrently. By using parallelism, a terabyte of data can be scanned and processed in minutes or less, not hours or days.

4.6.3.3 Troubleshooting Data Mining Model Creation

After the data mining models are created, check the error log in `oadm_sys.dwc_intra_etl_activity` table.

For example, execute the following code:

```
set line 160
col ACTIVITY_NAME format a30
col ACTIVITY_STATUS format a20
col error_dtl format a80
select activity_name, activity_status,  error_dtl from dwc_intra_etl_activity;
```

If all models are created successfully, the `activity_status` is all `"COMPLETED-SUCCESS"`. If the `activity_status` is `"COMPLETED-ERROR"` for a certain step, check the ERROR_DTL column, and fix the problem accordingly.

The following examples illustrate how to troubleshoot some common error messages returned in ERROR_DTL and ACTIVITY_NAME when working with Oracle Airlines Data Model.

- Example 4-2 (page 4-20)

- Example 4-3 (page 4-20)

- Example 4-4 (page 4-21)

Example 4-2   Troubleshooting the "Message not available ... [Language=ZHS]" Error

Assume that the returned error is `Message not available ... [Language=ZHS]`.

'ZHS' is a code for a language. The language name it relates to can appear as a different name depending on the database environment. This error happens when `oadm_sys.DWC_MESSAGE.LANGUAGE` does not contain messages for the current language.

Check the values in the DWC_MESSAGE table and, if required, update to the language code specified by the Oracle session variable `USERENV('lang')`.

Example 4-3   Troubleshooting ORA-40112: insufficient number of valid data rows,

Assume that the returned error is `ORA-40112: insufficient number of valid data rows, for "create_cust_ltv_svm_rgrsn" model.`

For this model, the target column is
`oadm_sys.dmv_cust_ltv_src.tot_cpn_amt`.

To troubleshoot this error:

1. Execute the following SQL statements:

    ```
    SELECT count(tot_cpn_amt) FROM dmv_cust_ltv_src;
    SELECT count(frqtflr_card_key) FROM dmv_cust_ltv_src;
    ```

2. Check that the values returned by above two queries are same and greater than 0 (zero). If value returned by the first `SELECT` statement is smaller than value returned by the second `SELECT` statement, then check the source tables and materialized views of `dmv_cust_ltv_src`.

Example 4-4   Troubleshooting ORA-40113: insufficient number of distinct target values

Assume that the returned error is `ORA-40113: insufficient number of distinct target values`, for `"create_ffp_pred_svm"` model.For a two-class classification model, the target column should have two distinct values. This error happens when the target column for the training model contains only one value or no value when it is expecting two values.

For example, for frequent fliers prediction among non-frequent fliers svm model, the target column is `oadm_sys.dmv_ffp_pred_src.ff_ind`.

To troubleshoot this error:

1. Execute a SQL query to check if there are enough values in this column. Using the frequent flier prediction svm model as an example, issue the following statement:

    ```
    SELECT ff_ind, count(*) FROM dmv_ffp_pred_src GROUP BY ff_ind;
    ```

    The result of the query is shown below.

    ```
    FF_IND       COUNT(*)
    ------       --------
         1           1296
         0            990
    ```

2. Check the following tables to ensure that there are both frequent fliers and non-frequent fliers by issuing the following query:

    ```
    SELECT NVL2(frqtflr_nbr,'FFP','Non_FFP') AS ffp_ind, count(*)
       FROM oadm_sys.dwd_bkg_fact GROUP BY NVL2(frqtflr_nbr,'FFP','Non_FFP');
    ```

    The result of the query is shown below

    ```
    FFP_IND      COUNT(*)
    -------      --------
    Non-FFP         77353
    FFP             42647
    ```

3. Execute the following statement to refresh the mining source materialized views:

    ```
    exec pkg_oadm_mining.refresh_mining_source;
    ```

5

Report and Query Customization

Provides information about creating reports, queries, and dashboards against the data in an Oracle Airlines Data Model warehouse.

5.1 Reporting Approaches in Oracle Airlines Data Model

Describes the methods to create reports from the data in an Oracle Airlines Data Model warehouse.

- **Relational Reporting**

  With relational reporting, you create reports against the analytical layer entities using the fact tables as the center of the star with the dimension or lookup tables acting as the dimensions of the star. Typically, the facts are derived (`DWD_`) and aggregate (`DWA_`) tables. However in some cases, you may need to use base (`DWB_`) tables along with reference tables (`DWR_`) or lookup (`DWL_`) tables to generate more detailed reports.

  – The reference (`DWR_`) tables typically represent dimensions which contain a business hierarchy and are present in the form of snowflake entities containing a table for each level of the hierarchy. This allows you to attach the appropriate set of reference tables for the multiple subject areas and fact entities composed of differing granularity. For example, you can use the time dimension table `DWR_CLNDR` to query against a `DAY` level Passenger Name Record (PNR) data such as `DWD_PNR`.

  – Lookup (`DWL_`) tables represent the simpler dimensions comprising a single level of a flat list of values. Typically, most reporting tools add a superficial top level to the dimension. These could be individual tables starting with `DWL_` or views (also named `DWL_`) on `DWL_CODE_MASTER` that break out different code types into separate dimensions.

- **OLAP Reporting**

  With OLAP reporting, you access Oracle OLAP cubes using SQL against the dimension and cube (fact) views. Cubes and dimensions are represented using a star schema design. Dimension views form a constellation around the cube (or fact) view. The views of the dimension and cube views are relational views with names ending with `_VIEW`:

  – OLAP dimension views contain information relating to the *whole* dimension, including all the levels of the hierarchy logically partitioned on the basis of a level column (identified as `level_name`). You can think of OLAP dimensions as "flattened" views of snowflake dimension tables. Typically, in Oracle Airlines Data Model, a dimension view is named *`dimension_hierarchy`*`_VIEW`

  – Cube views contain the facts pertaining to the cross-combination of the levels of individual dimensions which are part of the cube definition. Also the join from the cube view and the dimension views are based on the dimension keys along with required dimension level filters. Typically, in Oracle Airlines Data Model, a cube view is named *`cube`*`_VIEW`.

  Although the OLAP views are also modeled as a star schema, there are certain unique features to the OLAP reporting methodology which requires special modeling techniques in Oracle Business Intelligence Suite Enterprise Edition.

Related Topics:

Writing As Is and As Was Queries (page 5-8)
> The common query techniques are "as is" and "as was" queries.

Tutorial: Creating a New Oracle Airlines Data Model Dashboard (page 5-14)
> Explains how to create a dashboard based on dashboards in the Oracle Airlines Data Model webcat included with the sample Oracle Business Intelligence Suite Extended Edition reports delivered with Oracle Airlines Data Model.

Explains how to create a report based on the Oracle Airlines Data Model webcat included with the sample Oracle Business Intelligence Suite Extended Edition reports delivered with Oracle Airlines Data Model.

Oracle Airlines Data Model Reference

See Also:

The Oracle By Example tutorial, entitled "Using Oracle OLAP 11*g* With Oracle BI Enterprise Edition". To access the tutorial, open the Oracle Learning Library in your browser by following the instructions in "Oracle Technology Network (page xiv)"; and, then, search for the tutorials by name

---

5.2 Customizing Oracle Airlines Data Model Sample Reports

Sample reports and dashboards are delivered with Oracle Airlines Data Model. These sample reports illustrate the analytic capabilities provided with Oracle Airlines Data Model -- including the OLAP and data mining capabilities.

Sample reports and dashboards are delivered with Oracle Airlines Data Model. These sample reports illustrate the analytic capabilities provided with Oracle Airlines Data Model -- including the OLAP and data mining capabilities.

The sample reports were developed using Oracle Business Intelligence Suite Extended Edition which is a comprehensive suite of enterprise business intelligence products that delivers a full range of analysis and reporting capabilities. Thus, the reports also illustrate the ease with which you can use Oracle Business Intelligence Suite Extended Edition Answers and Dashboard presentation tools to create useful reports.

You can use Oracle Business Intelligence Suite Extended Edition Answers and Dashboard presentation tools to customize the predefined sample dashboard reports:

- **Oracle BI Answers:** Provides end user ad hoc capabilities in a pure Web architecture. Users interact with a logical view of the information -- completely hidden from data structure complexity while simultaneously preventing runaway queries. Users can easily create charts, pivot tables, reports, and visually appealing dashboards.

- **Oracle BI Interactive Dashboards:** Provide any knowledge worker with intuitive, interactive access to information. The end user can be working with live reports, prompts, charts, tables, pivot tables, graphics, and tickers. The user has full capability for drilling, navigating, modifying, and interacting with these results.

Related Topics:

Oracle Airlines Data Model Installation Guide

Oracle Airlines Data Model Reference

5.3 Writing Your Own Queries and Reports

The `oadm_sys` schema defines the relational tables and views in Oracle Airlines Data Model. You can use any SQL reporting tool to query and report on these tables and views.

Oracle Airlines Data Model also supports On Line Analytic Processing (OLAP) reporting using OLAP cubes defined in the `oadm_sys` schema. You can query and

write reports on OLAP cubes by using SQL tools to query the views that are defined for the cubes or by using OLAP tools to directly query the OLAP components.

Example 5-1   Creating a Relational Query for Oracle Airlines Data Model

For example, assume that you want to know the total booking count for the business class for February 2010. To answer this question, you might have to query the tables described in the following table:

Entity Name	Table Name	Description
Daily Booking Fact	DWA_DLY_BKG_FACT	Stores the booking related fact data.
Booking Class Fact	DWR_BKG_CLS_TYP	The booking class type like business or economy.
Calendar	DWR_CLNDR	Stores the calendar information.

To make the query, you execute the following SQL statement:

```
select
DWR_CLNDR.CLNDR_MONTH_CD,DWR_BKG_CLS_TYP.SVC_CLS_DESC,sum(DWA_DLY_BKG_FACT.BKD) as
Booking_Count
from DWA_DLY_BKG_FACT,DWR_CLNDR,DWR_BKG_CLS_TYP
where DWR_CLNDR.CLNDR_KEY=DWA_DLY_BKG_FACT.BKG_DT_KEY
and DWR_BKG_CLS_TYP.BKG_CLS_KEY=DWA_DLY_BKG_FACT.BKG_CLS_KEY
and DWR_BKG_CLS_TYP.SVC_CLS_DESC='BUSINESS'
and DWR_CLNDR.CLNDR_MONTH_KEY=20100201
group by DWR_CLNDR.CLNDR_MONTH_CD, DWR_BKG_CLS_TYP.SVC_CLS_DESC
```

The result of this query is shown below.

```
CLNDR_MONTH_CD           SVC_CLS_DESC          BOOKING_COUNT
------------------------ --------------------   ----------------------
20100201                 BUSINESS              61
```

Related Topics:

Reporting Approaches in Oracle Airlines Data Model (page 5-1)
> Describes the methods to create reports from the data in an Oracle Airlines Data Model warehouse.

Oracle OLAP Cube Views (page 3-19)
> When you define an OLAP cube, Oracle OLAP automatically generates a set of relational views on the cube and its dimensions and hierarchies.

Oracle OLAP User's Guide

5.4 Optimizing Star Queries

A typical query in the access layer is a join between the fact table and some number of dimension tables and is often referred to as a star query. In a star query each dimension table is joined to the fact table using a primary key to foreign key join. Normally the dimension tables do not join to each other.

Typically, in this kind of query all of the WHERE clause predicates are on the dimension tables and the fact table. Optimizing this type of query is very straight forward.

To optimize, you simply:

- Create a bitmap index on each of the foreign key columns in the fact table or tables

- Set the initialization parameter STAR_TRANSFORMATION_ENABLED to TRUE.

Setting this parameter enables the optimizer feature for star queries which is off by default for backward compatibility.

If your environment meets these two criteria, your star queries should use a powerful optimization technique that rewrites or transforms your SQL called star transformation. Star transformation executes the query in two phases:

1. Retrieves the necessary rows from the fact table (row set).

2. Joins this row set to the dimension tables.

Example 5-2   Star Transformation

A business question that could be asked against the star schema would be "What was the total number of umbrellas sold in Boston during the month of May 2008?"

1. The original query.

    ```
    select SUM(quantity_sold) total_umbrellas_sold_in_Boston
    From Sales s, Customers c, Products p, Times t
    Where s.cust_id=cust_id
    And s.prod_id = p.prod_id
    And s.time_id=t.time_id
    And c.cust_city='BOSTON'
    And p.product='UMBRELLA'
    And t.month='MAY'
    And t.year=2008;
    ```

    As you can see all of the where clause predicates are on the dimension tables and the fact table (Sales) is joined to each of the dimensions using their foreign key, primary key relationship.

2. Perform the following actions:

    a. Create a bitmap index on each of the foreign key columns in the fact table or tables

    b. Set the initialization parameter STAR_TRANSFORMATION_ENABLED to TRUE.

3. The rewritten query: Oracle rewrites and transfers the query to retrieve only the necessary rows from the fact table using bitmap indexes on the foreign key columns

    ```
    select SUM(quantity_sold
    From Sales
    Where cust_id IN
    (select c.cust_id From Customers c Where c.cust_city='BOSTON')
    And s.prod_id IN
    (select p.prod_id From Products p Where  p.product='UMBRELLA')
    And s.time_id IN
    (select t.time_id From Times(Where t.month='MAY' And tyear=2008);
    ```

    By rewriting the query you can now leverage the strengths of bitmap indexes. Bitmap indexes provide set based processing within the database, allowing you to use various fact methods for set operations such as AND, OR, MINUS, and COUNT. So, you use the bitmap index on time_id to identify the set of rows in the fact table corresponding to sales in May 2008. In the bitmap the set of rows are actually represented as a string of 1's and 0's. A similar bitmap is retrieved for the

fact table rows corresponding to the sale of umbrellas and another is accessed for sales made in Boston. At this point there are three bitmaps, each representing a set of rows in the fact table that satisfy an individual dimension constraint. The three bitmaps are then combined using a bitmap AND operation and this newly created final bitmap is used to extract the rows from the fact table needed to evaluate the query.

4. Using the rewritten query, Oracle joins the rows from fact tables to the dimension tables.

The join back to the dimension tables is normally done using a hash join, but the Oracle Optimizer selects the most efficient join method depending on the size of the dimension tables.

The rows from the fact table are retrieved by using bitmap joins between the bitmap indexes on all of the foreign key columns. The end user never needs to know any of the details of STAR_TRANSFORMATION, as the optimizer automatically chooses STAR_TRANSFORMATION when it is appropriate.

Example 5-2 (page 5-5) gives the step by step process to use STAR_TRANSFORMATION to optimize a star query.

The following figure shows the typical execution plan for a star query when STAR_TRANSFORMATION has kicked in. The execution plan may not look exactly how you imagined it. You may have noticed that you can see that there is no join back to the customer table after the rows have been successfully retrieved from the Sales table. If you look closely at the select list, you can see that there is not anything actually selected from the Customers table so the optimizer knows not to bother joining back to that dimension table. You may also notice that for some queries even if STAR_TRANSFORMATION does kick in it may not use all of the bitmap indexes on the fact table. The optimizer decides how many of the bitmap indexes are required to retrieve the necessary rows from the fact table. If an additional bitmap index would not improve the selectivity the optimizer does not use it. The only time you see the dimension table that corresponds to the excluded bitmap in the execution plan is during the second phase or the join back phase.

5.5 Troubleshooting Oracle Airlines Data Model Report Performance

Perform the following actions to identify problems generating a report created using Oracle Business Intelligence Suite Extended Edition:

1. In the (Online) Oracle BI Administrator Tool, select **Manage,** then **Security**, then **Users**, and then **oadm**.

   Ensure that the value for **Logging level** is 7.

2. Open the Oracle Airlines Data Model Repository, select **Manage**, and then **Cache**.

3. In the right-hand pane of the Cache Manager window, select all of the records, then right-click and select **Purge**.

4. Run the report or query that you want to track using the SQL log.

5. Open the query log file (`NQQuery.log`) under `OracleBI\server\Log`.

   The last query SQL is the log of the report you have just run. If an error was returned in your last accessed report, there is an error at the end of this log.

Example 5-3 Troubleshooting an Oracle Airlines Data Model Report

Assume the log file contains the following error:

```
Query Status: Query Failed: [nQSError: 15018] Incorrectly defined logical table
source (for fact table Booking Segment Departure Fact) does not contain mapping for
[Booking Segment Departure Fact.Confirmed Count, Booking Segment Departure
Fact.Belly Cargo Revenue].
```

This error occurs when there is a problem in the Business layer in your Oracle Business Intelligence Suite Extended Edition repository.

In this case, you need to check the mappings for `Booking Segment Departure Fact.Confirmed Count` and `Booking Segment Departure Fact.Belly Cargo Revenue`.

Example 5-4 Troubleshooting a Report: A Table Does Not Exist

Assume the log file contains the following error:

```
Query Status: Query Failed: [encloser: 17001] Oracle Error code:
942, message: ORA-00942: table or view does not exist.
```

This error occurs when the physical layer in your Oracle Business Intelligence Suite Extended Edition repository has the table which actually does not exist in the Database.

To find out which table has a problem:

1. Copy the SQL query to database environment.

2. Execute the query.

The table which does not exist is marked out by the Database client.

Example 5-5 Troubleshooting a Report: When the Database is Not Connected

Assume the log file contains the following error:

Error: `Query Status: Query Failed: [nQSError: 17001] Oracle Error code: 12545, message: ORA-12545: connect failed because target host or object does not exist.`

Meaning: This error occurs when the Database is not connected.

Action: Check connecting information in physical layer and ODBC connection to ensure that the repository is connecting to the correct database.

The following examples illustrate how to use these error messages:

- Example 5-3 (page 5-7)

- Example 5-4 (page 5-7)

- Example 5-5 (page 5-7)

5.6 Writing As Is and As Was Queries

The common query techniques are "as is" and "as was" queries.

Characteristics of an As Is Query (page 5-8)
> An As Is query has the following characteristics:

Characteristics of an As Was Query (page 5-8)
> An As Was query (also known as point-in-time analysis) has the following characteristics:

Examples: As Is and As Was Queries (page 5-9)
> The following examples illustrate the characteristics of As Is and As Was queries:

5.6.1 Characteristics of an As Is Query

An As Is query has the following characteristics:

- The resulting report shows the data as it happened.

- The snowflake dimension tables are also joined using the surrogate key columns (that is the primary key and foreign key columns).

- The fact table is joined with the dimension tables (at leaf level) using the surrogate key column.

- Slowly-changing data in the dimensions are joined with their corresponding fact records and are presented individually.

- It is possible to add up the components if the different versions share similar characteristics.

5.6.2 Characteristics of an As Was Query

An As Was query (also known as point-in-time analysis) has the following characteristics:

- The resulting report shows the data that results from freezing the dimensions and dimension hierarchy at a specific point in time.

- Each snowflake table is initially filtered by applying a point-in-time date filter which selects the records or versions which are valid as of the analysis date. This structure is called the point-in-time version of the snowflake.

- The filtered snowflake is joined with an unfiltered version of itself by using the natural key. All of the snowflake attributes are taken from the point-in-time version alias. The resulting structure is called the composite snowflake.

- A composite dimension is formed by joining the individual snowflakes on the surrogate key.

- The fact table is joined with the composite dimension table at the leaf level using the surrogate key column.

- The point-in-time version is super-imposed on all other possible SCD versions of the same business entity -- both backward as well as forward in time. Joining in this fashion gives the impression that the dimension is composed of only the specific point-in-time records.

- All of the fact components for various versions add up correctly due to the super-imposition of point-in-time attributes within the dimensions.

5.6.3 Examples: As Is and As Was Queries

The following examples illustrate the characteristics of As Is and As Was queries:

Data used for the examples

Assume that your data warehouse has a `Customer` table, a `County`, and a `TaxPaid` fact table. As of January 1, 2011, these tables include the values shown below.

Customer Table

Cust Id	Cust Cd	Cust Nm	Gender	M Status	County Id	County Cd	Country Nm	...	Eff Frm	Eff To
101	JoD	John Doe	Male	Single	5001	SV	Sunnyvale	...	1-Jan-11	31-Dec-99
102	JaD	Jane Doe	Female	Single	5001	SV	Sunnyvale	...	1-Jan-11	31-Dec-99
103	JiD	Jim Doe	Male	Married	5002	CU	Cupertino	...	1-Jan-11	31-Dec-99

County Table

County Id	County CD	County Nm	Population	...	Eff Frm	Eff To
5001	SV	Sunnyvale	Very High	...	1-Jan-11	31-Dec-99

County Id	County CD	County Nm	Population	...	Eff Frm	Eff To
5002	CU	Cupertino	High	...	1-Jan-11	31-Dec-99

TaxPaid Table

Cust Id	Day	Tax Type	Tax
101	1-Jan-11	Professional Tax	100
102	1-Jan-11	Professional Tax	100
103	1-Jan-11	Professional Tax	100

Assume that the following events occurred in January 2011:

- On January 20, 2011, Jane Doe marries.

- On Jan 29, 2011, John Doe moves from Sunnyvale to Cupertino.

Consequently, as shown below, on February 1, 2011, the `Customer` and `TaxPaid` tables have new data while the values in the `County` table stay the same.

Customer table

Cust Id	Cust Cd	Cust Nm	Gender	M Status	County Id	County Cd	Country Nm	...	Eff Frm	Eff To
101	JoD	John Doe	Male	Single	5001	SV	Sunnyvale	...	1-Jan-11	29-Jan-11
102	JaD	Jane Doe	Female	Single	5001	SV	Sunnyvale	...	1-Jan-11	20-Jan-11
103	JiD	Jim Doe	Male	Married	5002	CU	Cupertino	...	1-Jan-11	31-Dec-99
104	JaD	Jane Doe	Female	Married	5001	SV	Sunnyvale	...	21-Jan-11	31-Dec-99
105	JoD	John Doe	Male	Single	5002	CD	Cupertino	...	30-Jan-11	31-Dec-99

County table

County Id	County CD	County Nm	Population	...	Eff Frm	Eff To
5001	SV	Sunnyvale	Very High	...	1-Jan-11	31-Dec-99
5002	CU	Cupertino	High	...	1-Jan-11	31-Dec-99

TaxPaid Table

Cust Id	Day	Tax Type	Tax
101	1-Jan-11	Professional Tax	100
102	1-Jan-11	Professional Tax	100
103	1-Jan-11	Professional Tax	100
105	1-Feb-11	Professional Tax	100
104	1-Feb-11	Professional Tax	100
103	1-Feb-11	Professional Tax	100

Example 5-6   As Is Query for Tax Collection Split by Marital Status

To show the tax collection data split by martial status, the following SQL statement that joins the `TaxPaid` fact table and the `Customer` dimension table on the `cust_id` surrogate key and the `Customer` and `County` snowflakes on the `cnty_id` surrogate key:

```
SELECT cust.cust_nm, cust.m_status, SUM(fct.tx)
FROM taxpaid fct, customer cust, county cnty
WHERE fct.cust_id = cust.cust_id
AND cust.cnty_id = cnt.cnt_id
GROUP BY cust.cust_nm, cust.m_status
ORDER BY 1,2,3;
```

The results of this query are shown below. There are two rows for Jane Doe; one row for a marital status of Married and another for a marital status of Single.

Cust Nm	M Status	Tax
Jane Doe	Married	100
Jane Doe	Single	100
Jim Doe	Married	200
John Doe	Single	200

Example 5-7   As Was Queries for Tax Collection Split by Marital Status

Assuming the Data used for the examples (page 5-9), issue the following SQL statement to show the tax collection data split by marital status using an analysis date of January 15, 2011:

```
select
   cust.cust_nm, cust.m_status, sum(fct.tax)
from
   TaxPaid fct,
   (
     select
        cust_act.cust_id, cust_pit.cust_cd, cust_pit.cust_nm,
        cust_pit.m_status, cust_pit.gender,
        cust_pit.cnty_id, cust_pit.cnty_cd, cust_pit.cnty_nm
     from Customer cust_act
     inner join (
```

```
        select
            cust_id, cust_cd, cust_nm,
            m_status, gender,
            cnty_id, cnty_cd, cnty_nm
        from Customer cust_all
        where to_date('15-JAN-2011', 'DD-MON-YYYY') between eff_from and eff_to
      ) cust_pit
      on (cust_act.cust_cd = cust_pit.cust_cd)
  ) cust,
  (
    select
        cnty_act.cnty_id, cnty_pit.cnty_cd, cnty_pit.cnty_nm
    from County cnty_act
    inner join (
        select
            cnty_id, cnty_cd, cnty_nm
        from County cnty_all
        where to_date('15-JAN-2011', 'DD-MON-YYYY') between eff_from and eff_to
    ) cnty_pit
    on (cnty_act.cnty_cd = cnty_pit.cnty_cd)
  ) cnty
where fct.cust_id = cust.cust_id
and cust.cnty_id = cnty.cnty_id
GROUP BY cust.cust_nm, cust.m_status
order by 1,2,3;
```

The results of this query are shown below. Since Jane Doe was single on January 15, 2011 (the analysis date), all tax for Jane Doe is accounted under her Single status.

Cust Nm	M Status	Tax
Jane Doe	Single	200
Jim Doe	Married	200
John Doe	Single	200

Assume instead that you issued the exact same query except that for the `to_date` phrase you specify `09-FEB-2011` rather than `15-JAN-2011`. Since Jane Doe was married on February 9, 2011, then, as shown below all tax for Jane Doe would be accounted under her Married status.

Cust Nm	M Status	Tax
Jane Doe	Married	200
Jim Doe	Married	200
John Doe	Single	200

Example 5-8   As Is Query for Tax Collection Data Split by County

Assuming the Data used for the examples (page 5-9), issue the following SQL statement to show the tax collection data split by county:

```
SELECT cust.cust_nm, cnty.cnty_nm, SUM(fct.tax)
FROM TaxPaid fct, customer cust, county cnty
WHERE fct.cust_id = cust.cust_id
```

```
AND cust.cnty_id = cnty.cnty_ID
GROUP BY cut.cust_nm, cnty.cnty_nm
ORDER BY 1,2,3;
```

The results of this query are shown below. Because John Doe lived in two different counties, there are two rows of data for John Doe.

Cust Nm	County Nm	Tax
Jane Doe	Sunnyvale	200
Jim Doe	Cupertino	200
John Doe	Cupertino	100
John Doe	Sunnyvale	100

Example 5-9   As Was Queries for Tax Collection Data Split by County

Assuming the Data used for the examples (page 5-9), issue the following SQL statement to show the tax collection data split by county using an analysis date of January 15, 2011:

```
select
   cust.cust_nm, cnty.cnty_nm, sum(fct.tax)
from
  TaxPaid fct,
  (
     select
        cust_act.cust_id, cust_pit.cust_cd, cust_pit.cust_nm,
        cust_pit.m_status, cust_pit.gender,
        cust_pit.cnty_id, cust_pit.cnty_cd, cust_pit.cnty_nm
     from Customer cust_act
     inner join (
        select
           cust_id, cust_cd, cust_nm,
           m_status, gender,
           cnty_id, cnty_cd, cnty_nm
        from Customer cust_all
        where to_date('15-JAN-2011', 'DD-MON-YYYY') between eff_from and eff_to
     ) cust_pit
     on (cust_act.cust_cd = cust_pit.cust_cd
  ) cust,
  (
     select
        cnty_act.cnty_id, cnty_pit.cnty_cd, cnty_pit.cnty_nm
     from County cnty_act
     inner join (
        select
           cnty_id, cnty_cd, cnty_nm
        from County cnty_all
        where to_date('15-JAN-2011', 'DD-MON-YYYY') between eff_from and eff_to
     ) cnty_pit
     on (cnty_act.cnty_cd = cnty_pit.cnty_cd)
  ) cnty
where fct.cust_id = cust.cust_id
and cust.cnty_id = cnty.cnty_id
GROUP BY cust.cust_nm, cnty.cnty_nm
order by 1,2,3;
```

The results of this query are shown below. Because John Doe was in Sunnyvale as of the analysis date of January 15, 2011, all tax for John Doe is accounted for under the Sunnyvale county.

Cust Nm	County Nm	Tax
Jane Doe	Sunnyvale	200
Jim Doe	Cupertino	200
John Doe	Sunnyvale	200

Assume instead that you issued the exact same query except that for the to_date phrase you specify 09-FEB-2011 rather than 15-JAN-2011. Since John Doe lived in Cupertino on February 9, 2011, then, as shown below all tax for John Doe would be accounted under Cupertino.

Cust Nm	County Nm	Tax
Jane Doe	Sunnyvale	200
Jim Doe	Cupertino	200
John Doe	Cupertino	200

5.7 Tutorial: Creating a New Oracle Airlines Data Model Dashboard

Explains how to create a dashboard based on dashboards in the Oracle Airlines Data Model webcat included with the sample Oracle Business Intelligence Suite Extended Edition reports delivered with Oracle Airlines Data Model.

In this example assume that you want to create a dashboard named "Agent Revenue and Booking Analysis", and put both "Agent Revenue Analysis in USD" and "Agent Performance Analysis PCT CANCEL" into this new dashboard.

To create a dashboard, perform the following steps:

1. In the browser, open the login page at http://*servername*:9704/analytics where *servername* is the server on which the webcat is installed.

   Login with the username of oadm, and provide the password.

2. Select **New**, and then select **Dashboard** to create an Oracle Business Intelligence Suite Extended Edition dashboard.
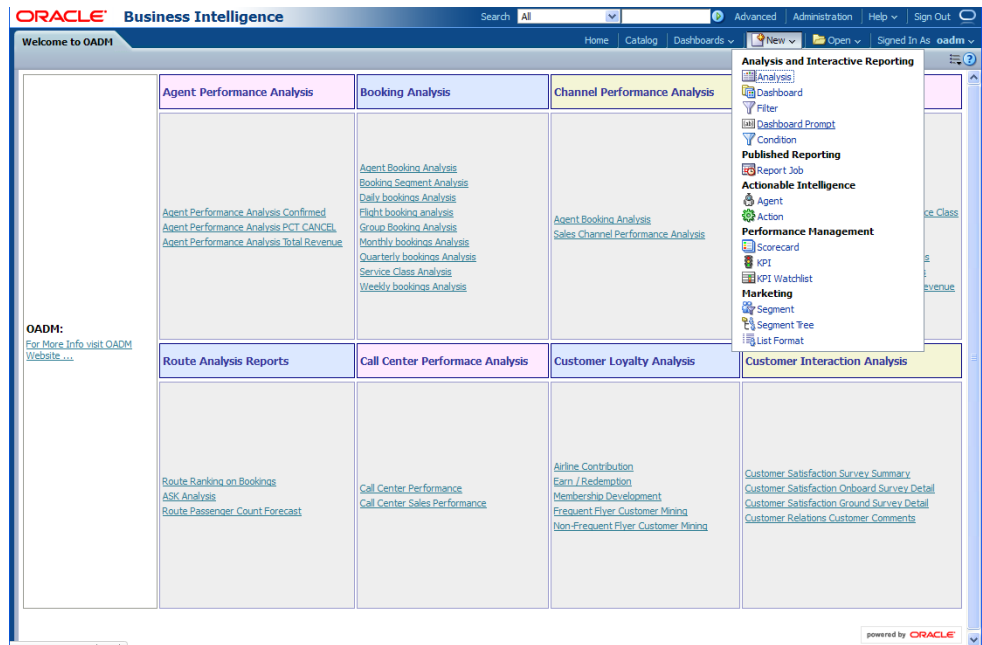
3. Enter a name and description. Then save the dashboard to the Dashboards folder. Click **OK**.
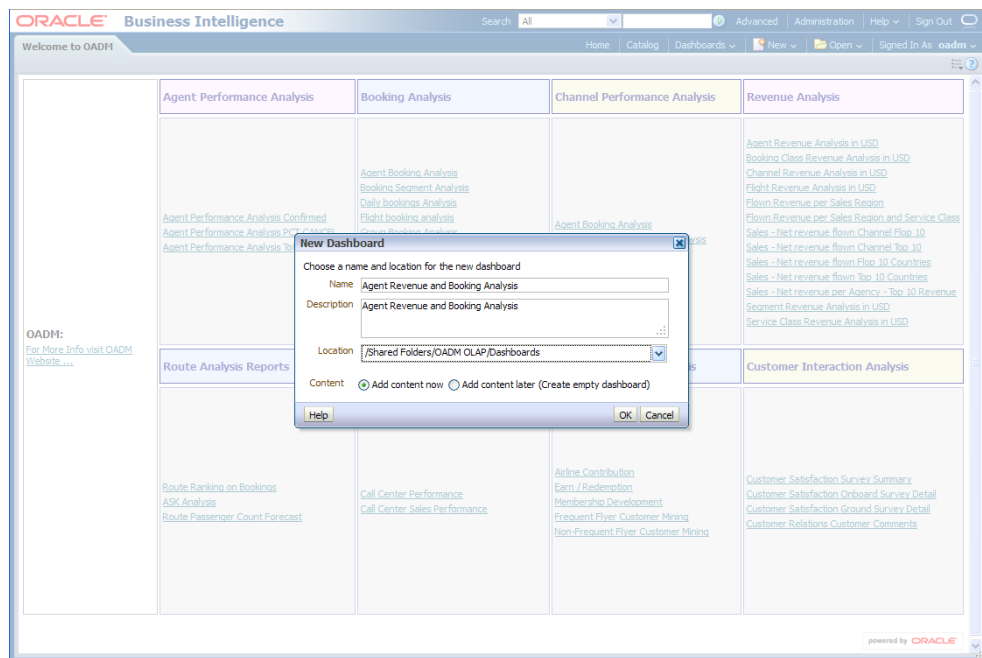
4. In the Catalog view, expand the Revenue Analysis folder. You can see the Agent Revenue Analysis in USD Report, drag it from the catalog view into the right panel.
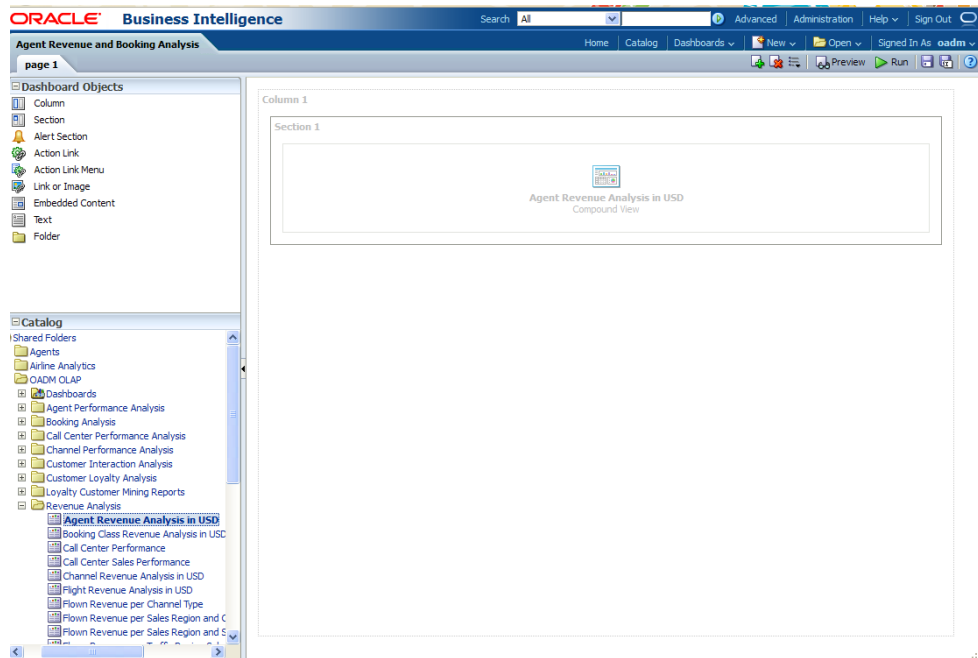
5. In the Catalog view, expand the Revenue Analysis folder. You can see the Agent Performance Analysis PCT CANCEL Report. Drag that report from the Catalog view into the right panel.
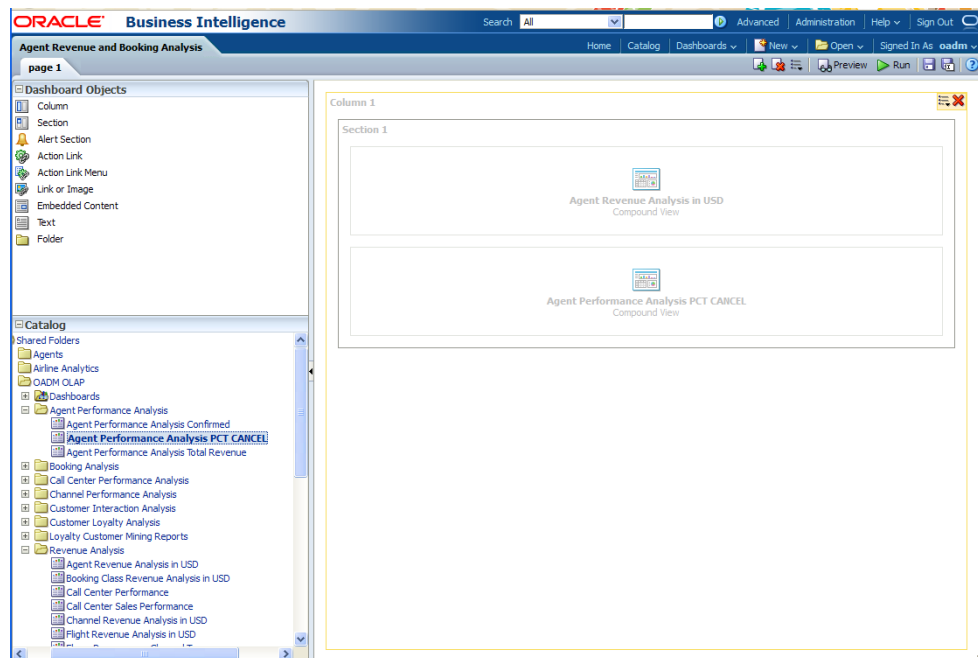
6. You can change the layout of this section to organize the two reports by horizontal or vertical.
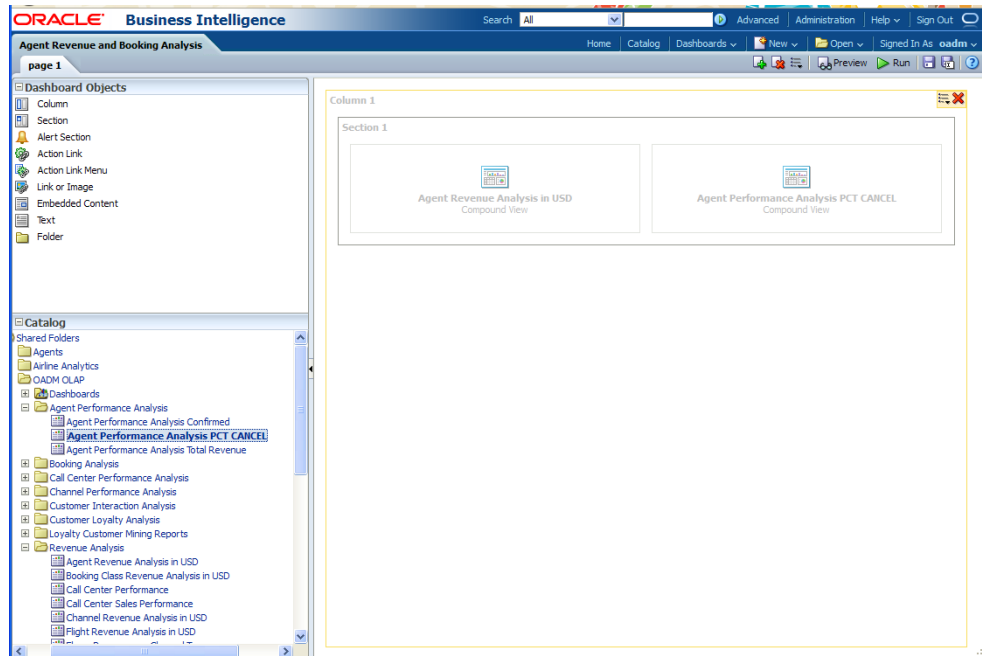
Note that the page name is still "Page1" so you must change it.

7. To change the page name:

a. Select the Dashboard.
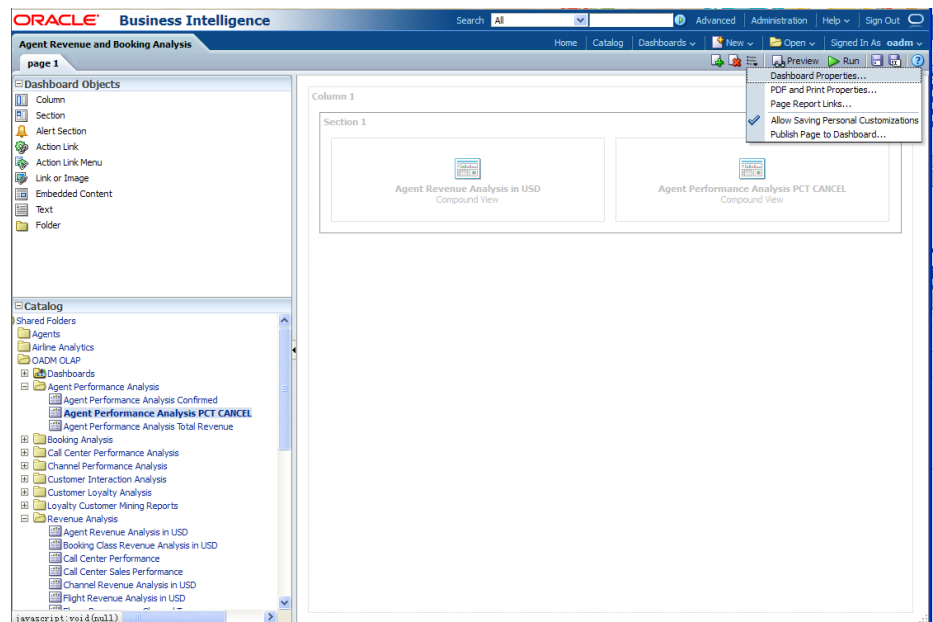
b. In Dashboard Properties window, click **Change Name**.
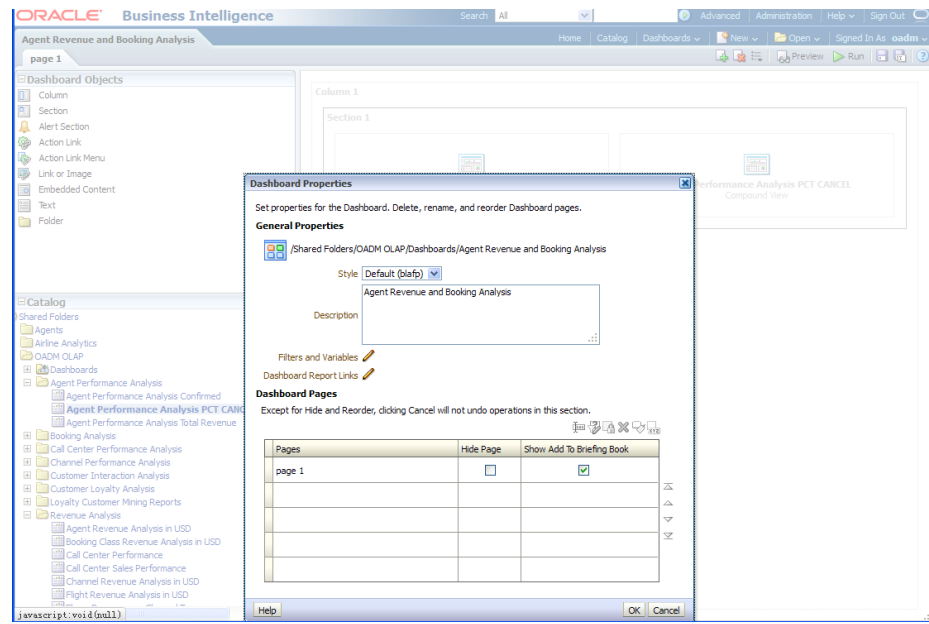
c. Change the name to "Agent Revenue and Booking Analysis", then click **OK**.
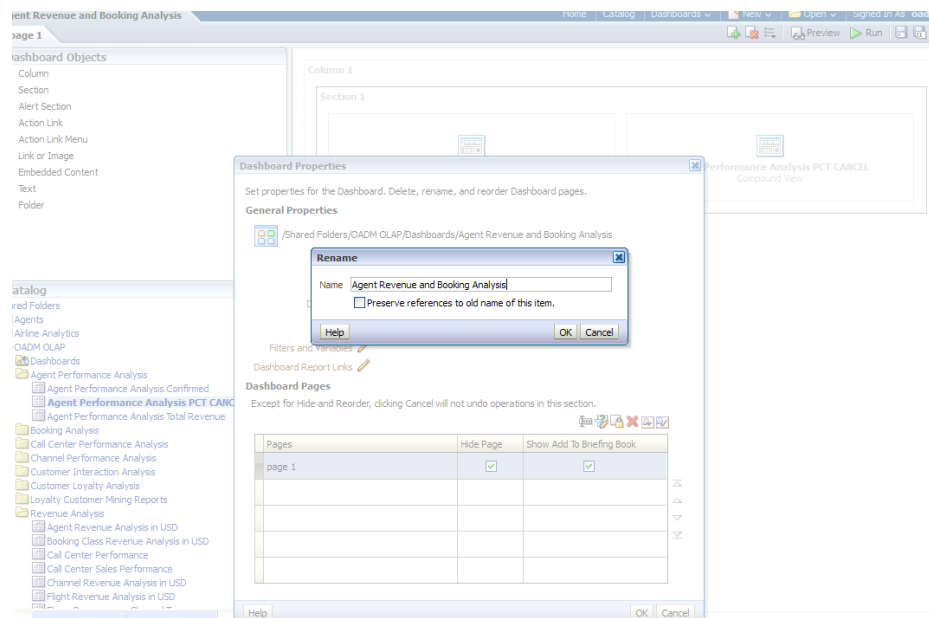
8. Click **Save** on the top of the dashboard. Now you have a new dashboard.
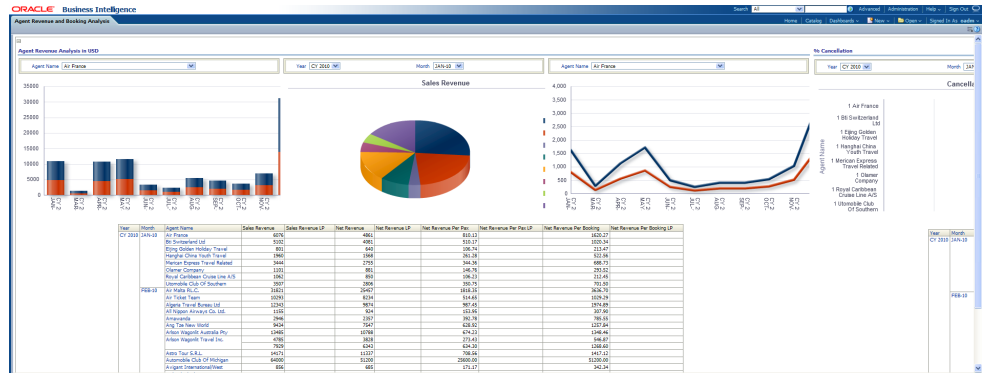
Note:

For more information on creating a report, see the "Creating Analyses and Dashboards 11*g*" OBE tutorial.

To access the tutorial, open the Oracle Learning Library in your browser by following the instructions in "Oracle Technology Network (page xiv)"; and, then, search for the tutorial by name.

5.8 Tutorial: Creating a New Oracle Airlines Data Model Report

Explains how to create a report based on the Oracle Airlines Data Model webcat included with the sample Oracle Business Intelligence Suite Extended Edition reports delivered with Oracle Airlines Data Model.

In this example, assume that you want to create a report named "Agent Booking and Flown Revenue" to put both booking count and flown revenue in one report.

To create a this new report, perform the following steps:

1. In the browser, open the login page at `http://servername:9704/analytics` where `servername` is the server on which the webcat is installed.

   Login with the username of `oadm`, and provide the password.

2. Select **New**, and then select **Analysis** to create an Oracle Business Intelligence Suite Extended Edition report.
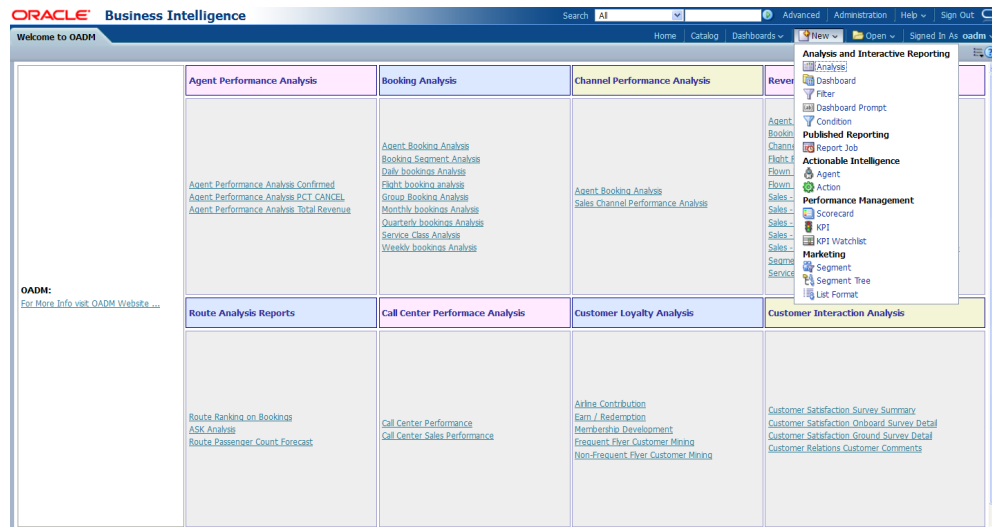
3. Select **Subject Area**, then select **OADM_OLAP** to create a relational report.
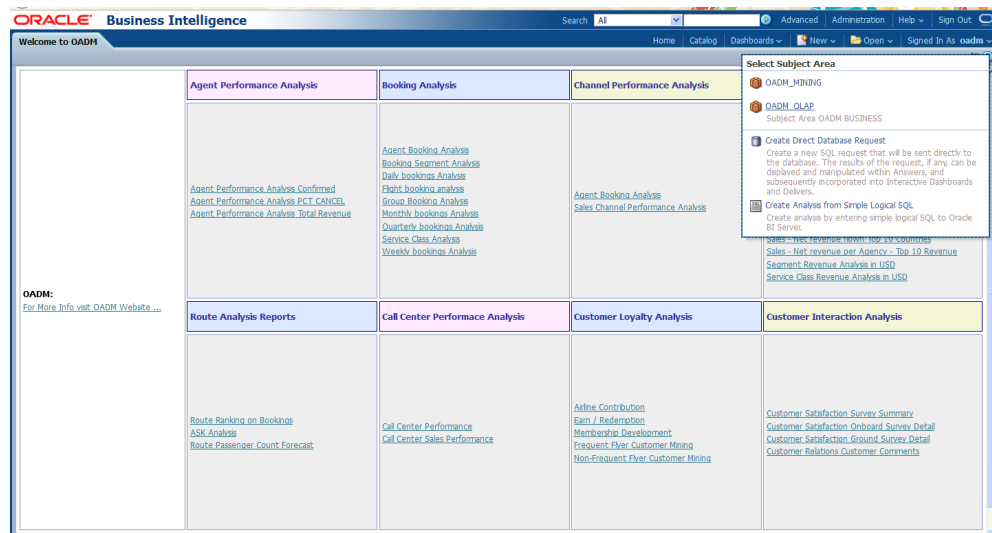
4. Drag and put the dimension and fact columns into the Select Columns panel.
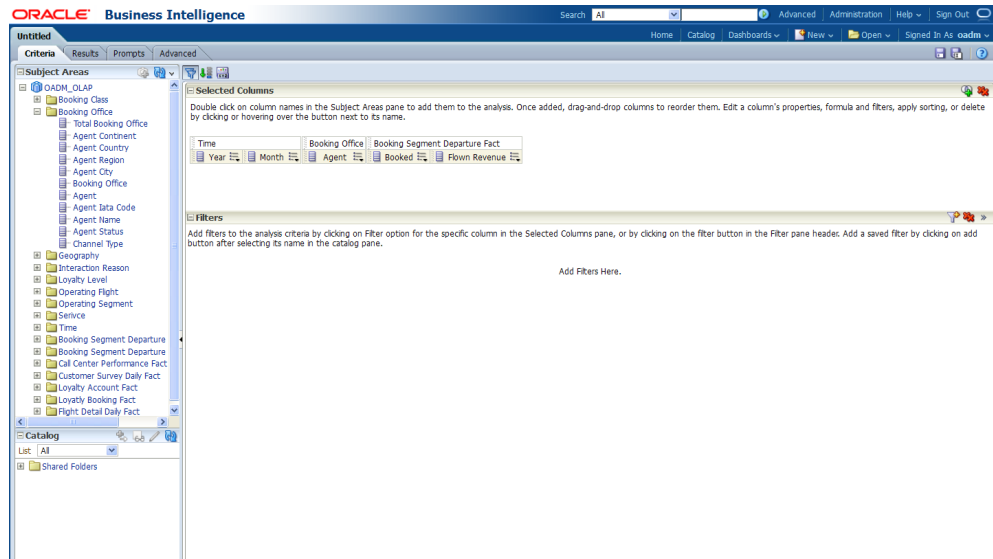
5. Select the **Results** tab to view the report.
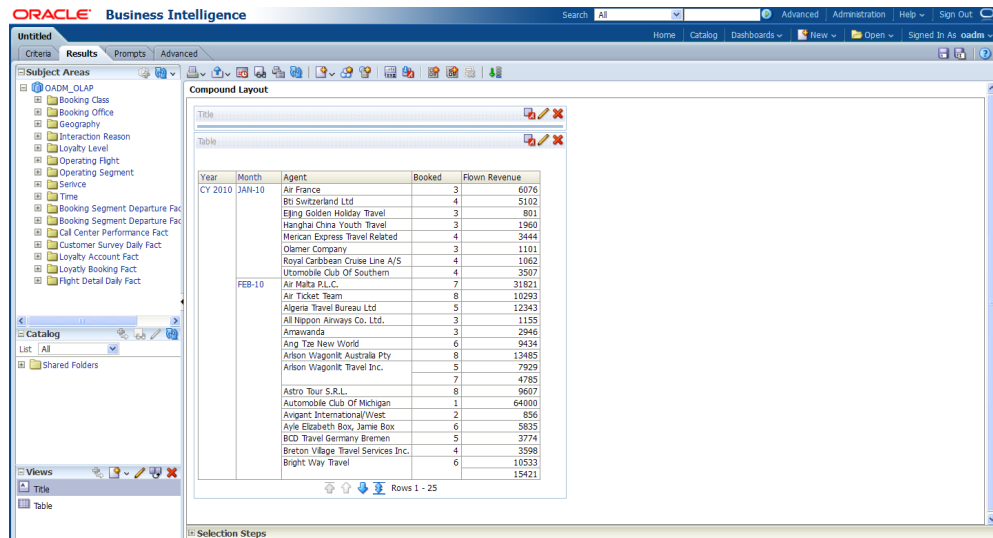
6. Select **New View** to add a chart into report.
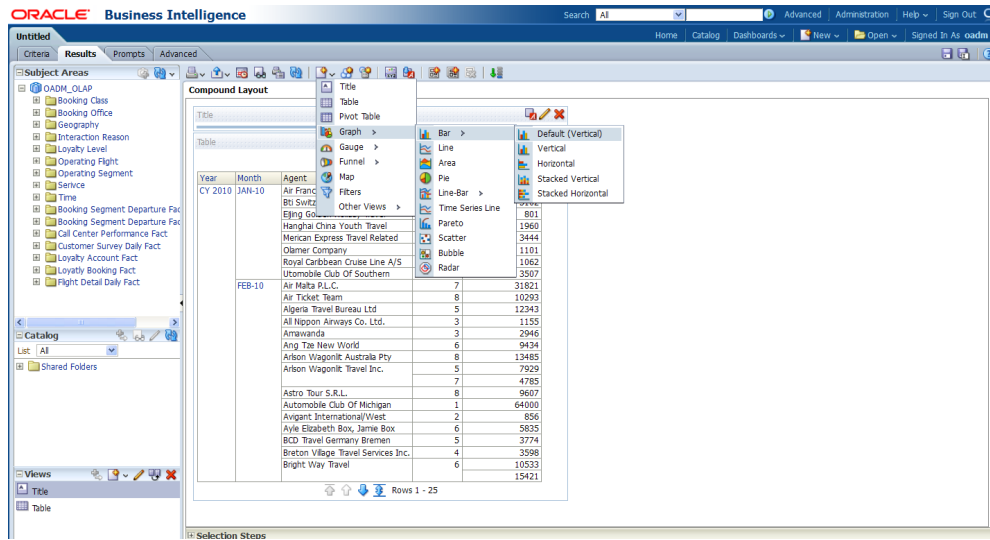
7. Select **Save** to save this report to the Network Health Analysis folder.
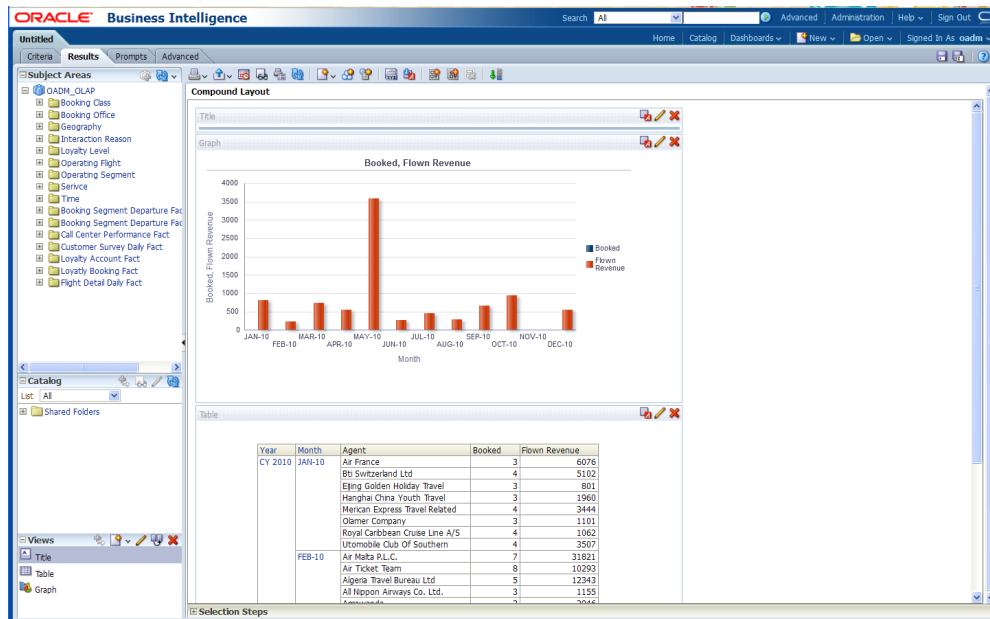
Note:

For more information on creating a report, see the "Creating Analyses and Dashboards 11*g*" OBE tutorial.

To access the tutorial, open the Oracle Learning Library in your browser by following the instructions in "Oracle Technology Network (page xiv)"; and, then, search for the tutorial by name.

6

Oracle Airlines Data Model Security

Provides information on Oracle Airlines Data Model security.

6.1 Oracle Airlines Data Model Security Overview

Provides an overview of Oracle Airlines Data Model security. When dealing with security, Oracle Airlines Data Model does not provide specific security features but assumes that you leverage the security features of its dependent Oracle products, including Oracle Database and WebLogic Server.

Oracle Airlines Data Model Security

Oracle Airlines Data Model has no specific security features enabled by default.

Oracle Airlines Data Model is a "normal" data warehouse implemented on top of an Oracle Database (although the data warehouse may only include industry information). Oracle Airlines Data Model is an addition to the Oracle Database and includes all the Oracle Database standard security features, or in some cases Oracle Advanced Security features that should be leveraged (as for any data warehouse containing sensitive data). Oracle Database provides privileged user controls, multi-factor authorization, transparent data encryption, auditing, configuration scanning, and SQL monitoring, and other industry leading security controls.

Read the following documents and apply the security solutions where relevant:

- *Oracle Database Security Guide*

- *Oracle Database Advanced Security Guide*

- *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server* (especially relevant when either WebLogic only or full OBIEE suite is used in combination with Oracle Airlines Data Model).

- *Oracle Application Server Security Guide* (only if used)

- *Oracle Application Server Administrator's Guide* (only if used)

All Oracle documentation is available from Oracle Help Center:

http://docs.oracle.com

There are several areas in Oracle Airlines Data Model where you must take specific care and note the security considerations, including:

- When the sensitivity of the data Oracle Airlines Data Model deals with is an important consideration.

- When the sensitivity of the access granted for some (power) users is an important consideration. For example, for anyone accessing critical application areas (for example campaign management).

Oracle Airlines Data Model Basic Security Considerations

The following principles are fundamental to using any application securely:

- **Keep software up to date.** This includes using the latest product release and any patches that apply to it.

- **Limit privileges as much as possible.** Users should be given only the access necessary to perform their work. User privileges should be reviewed periodically to determine relevance to current work requirements. This also means that no user should have direct access to OADM_SYS (at least for production).

- **Monitor system activity.** Establish who should access which system components, and how often, and monitor those components. This means that each user should have their own user ID (no generic login). It also means that roles and data access should be restricted and monitored, especially when accessing sensitive or very sensitive data.

- **Install software securely.** For example, use firewalls, secure protocols using TLS (SSL), and secure passwords.

- **Learn about and use the Oracle Airlines Data Model security features.**

  > **Note:**
  >
  > Using Oracle Airlines Data Model the Oracle Database DBA will leverage the standard or advanced security database options to protect Oracle Airlines Data Model content.

- **Use secure development practices.** For example, take advantage of existing database security functionality instead of creating your own application security.

- **Keep up to date on security information.** Oracle regularly issues security-related patch updates and security alerts. You must install all security patches as soon as possible. See the "Critical Patch Updates and Security Alerts" Web site:

  http://www.oracle.com/technetwork/topics/security/alerts-086861.html

Understanding the Oracle Airlines Data Model Environment

When planning your Oracle Airlines Data Model implementation, consider the following:

- **Which resources must be protected?**

  – You must protect employee and customer personal data, such as credit-card numbers, social security numbers and login, PIN, and so on.

  – You must protect internal data, such as proprietary source code, custom development, and so on. This is not specific to Oracle Airlines Data Model, but you must make sure that any party able to access Oracle Airlines Data Model and its customer paying documentation should have an NDA in place, to avoid being liable for dispatch or reuse of information by 3rd party.

  – You must protect system components from being disabled by external attacks or intentional system overloads. This is not specific to Oracle Airlines Data Model.

- **Who Are You Protecting Data From?**

  The list of sources of possible threats includes a wide range of people and organizations. Threats depend on the data stored. Individuals include employees or an external hacker.

For example, you must protect your subscribers' data from other subscribers, but someone in your organization might need to access that data to manage it. You should analyze your workflows to determine who needs access to the data and what for; it is possible that a system administrator can manage your system components without needing to access the system data. Also, hackers continue to target privileged accounts inside and outside of databases.

The following list provides some examples of threats and the possible financial impact:

- **Insider**: typically an employee or a contractor, who tries to abuse the normal rights for "revenge" or for their own benefit. In this case the possible damage is highly dependent on the breadth of access the person has and how malevolent the actions are. The damage can range from self adjustment (of one's own bills) up to destruction or re-selling of data.

- **Individual**: individual persons outside the organization, not including employees or contractors, that try to get into the organization's network to test themselves or to test the security for "fun". These threats can cause significant damage.

- **Criminal organizations**: such organizations target the organization for subscriber data, in particular bank account and credit card data. They may also target the network to take control and blackmail the organization.

- **State-like organizations**: their aim is usually to gather information on their own target(s). organization data is a treasure if they want this data and they do not or cannot go through a standard judicial process to obtain the data.

- **Research organizations**: these organizations try to test security and look for breaches. Their goal is often to publish their finding but such organizations sometimes notify the organization and provide the opportunity (time) to change the system before they publish security vulnerabilities. Such threats, when identified may have a small financial impact if the vulnerability stays under the organization's control.

The organization has to protect Oracle Airlines Data Model with the highest security standard possible with respect to the possible wide range of threats.

- **What happens if protections on a strategic resource fail?**

  In some cases, a fault in your security scheme is nothing more than an inconvenience. In other cases, a fault might cause great damage to the organization and to your customers. Understanding the security ramifications of each resource allows you to properly protect it.

Oracle Airlines Data Model is an option of the Oracle Database and because it is a physicalized Enterprise Data Warehouse, the security principles of Oracle Airlines Data Model follow the standard security principles valid for any Oracle Database. Additionally, you must apply the security policies associated with the applicable reporting tools. By default, Oracle Airlines Data Model uses the repository and reports with Oracle Business Intelligence Suite Extended Edition(OBIEE), which leverages WebLogic server. Hence, the security standards applied to the Oracle Airlines Data Model reporting layer should be set up as security for WebLogic server (and OBIEE).

Also, security features must protect the servers and their operating systems on which the database for Oracle Airlines Data Model runs, as well as any interfaces and configurations for supplying and accessing the systems or for reports. In particular:

- ETL/ELT servers and ETL GUI (staging area and more)

- Direct feed to Oracle Airlines Data Model (whether with GUI or not)

- Access to the database management (DBA GUI)

- Access to/from Oracle Airlines Data Model

  - For Development Tools: SQL developer, SQL Data Modeler

  - For research and discovery: Oracle Data Mining GUI, Oracle Data Miner (including SNA), Oracle R

  - For cubes management: Oracle Analytical Workspace Manager

  - External feeds to other applications (from Oracle Airlines Data Model directly, for example through a trigger)

  - Configuration Files

- Oracle Airlines Data Model Content

  - Users Traceability and Audit-ability (especially for sensitive data)

  - Encryption and masking (especially for sensitive data and the backups for sensitive data)

  - User access rights (through roles)

- Backup and Disaster Recovery Systems (and high availability architecture).

- BI systems and applications on top of Oracle Airlines Data Model, leveraging its data

  - OBIEE

  - Any other system (for example a campaign management tool)

Depending on which part of Oracle Airlines Data Model has been implemented and filled with data, some or all of the security steps should be applied. Additionally, any customization must be protected as part of the global security processes.

Recommended Deployment Configurations

Typically, as for any data warehouse, there are several environments on which you install Oracle Airlines Data Model:

- **The Test and Development Environment (R&D)**: this area is for testing, development, research, and unit testing with a subset of data. The test and development environment is typically where most customization takes place before being moved to other environments. The test and development area usually contains a full end-to-end implementation (from staging area up to BI reporting). Despite the test and development apparently temporary presence, this environment should be kept available for debugging and for work on extensions to Oracle Airlines Data Model (this area could be reduced when there are no planned project extensions).

> **Note:**
>
> The staging, data warehouse, and BI server do not need to be installed on the same platform and they can have their own test or pre-production and production environment.

- **The Pre-production Environment**: this area is typically provided for scalability testing (load and performance) and global integration testing. The pre-production environment should contain a representative subset of the data set (for example, containing a few months of history), for a relevant test and for final user acceptance.

- **The Production Environment**: this area is the final end user environment, providing the required performance and capacity to store the complete data set and any required history.

All these environments must be protected equally since they likely will contain copies of similar sensitive data, whatever subset of data you use for the different testing and development or pre-production environments. Additionally, the backup of each of these environments, if any, should be protected as if it was production.

6.2 Architecture Analysis for Security

Describes areas in Oracle Airlines Data Model that are potential security risks, to first be aware of them and second, to secure them.

Security Boundaries of Trust

Within each boundary shown in the following figure, assume that whoever accesses the data can be trusted (with "recognized" users or roles). Outside of those boundaries, anyone is a potential threat (this is critical from a security analysis perspective).

The following figure identifies points of interactions at risk, trust boundaries, through which an attack could occur.

Figure 6-1   Security Boundaries of Trust
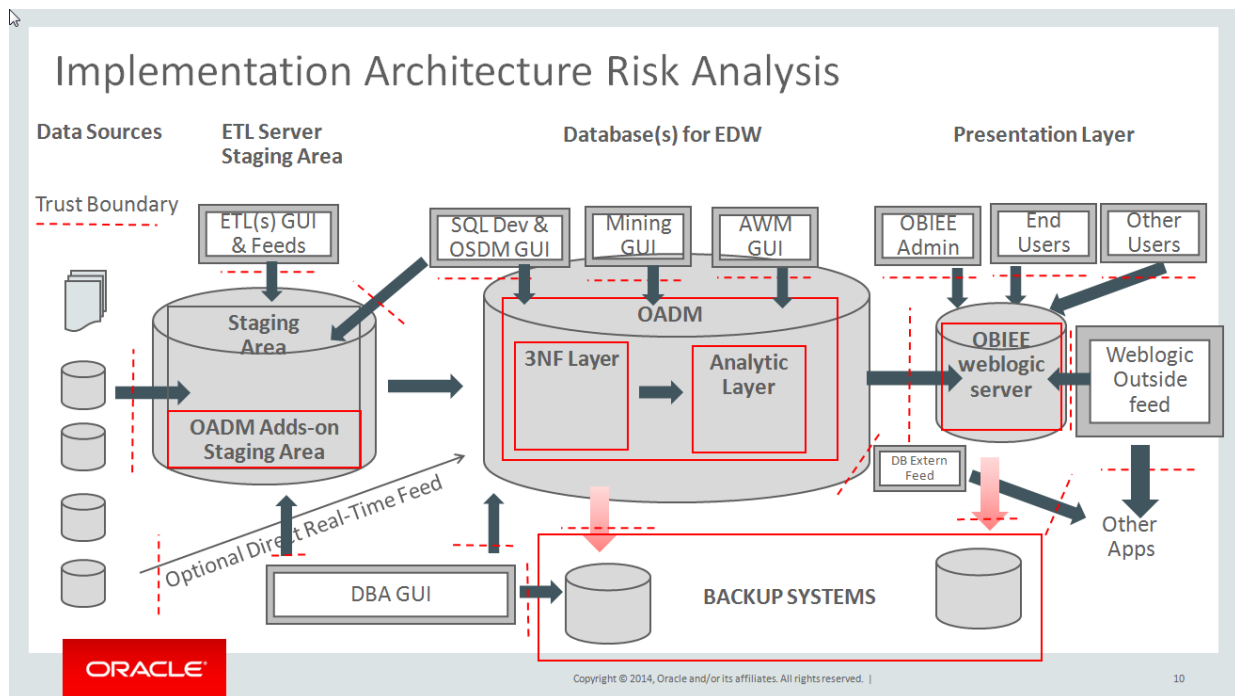
1. **ETL GUI:** Whatever tools and GUI you use for the Extraction, Transform, and Load tools that manage the data movement and transformation from the source to staging area, there is clearly a risk of attack:

    *   Because the tool has access to the raw data source as well as all its transformation rules (Database login data, data source visibility, metadata and transformation rules, mapping).

    *   Because someone could change the rules (and place unexpected code into Oracle Airlines Data Model).

    Note:

    If you use Big Data this is also considered part of the "Staging area" even if it also performs operational analysis and reporting. Adding a Big Data environment into the architecture thus includes additional interfaces, and additional boundaries of trust that need to be considered for security and analyzed.

    ---

2. **SQL Developer and Oracle Data Modeler GUI:** (or any GUI that can directly access the database of either the staging area or the Oracle Airlines Data Model area – hence two accesses at risk). The access at this boundary must to be restricted (leveraging roles) and monitored (traceability). Additionally, the access string must be protected (do not allow the `tnsnames.ora` to be saved with an unencryption schema password) and the schema password should never be saved with the applications.

3. **Data Source Feed:** When some data from the source system could cause a security breach (through unexpected code or corrupted data, intentional or not, that could break the process). The best way to protect from such an attack is to set up a data profiling check (for example with ODI Data Profiling option or manually by

verifying that the input data have the content you expect). Additionally, the schema name and password for the connection should be protected as for 1 and 2 (if not done in the respective GUI).

4. **Direct Data Source Feed:** This case is similar to case 3, but a direct data source feed assumes a real-time or near real-time direct feed to Oracle Airlines Data Model (to the foundation or analytic layer). The protection should however be similar to 3.

5. **DBA GUI Access:** Due to its rights, the DBA is clearly a role with risk, whatever GUI this person uses (Oracle Enterprise Manager, or any other tools of choice),

> **See Also:**
>
> Leverage the suggestions in *Oracle Database Security Guide* to split the various DBA roles and data access

6. **DBA Backup Process and Access GUI:** This is similar to 5, but for the backup process. Refer to the *Oracle Database Security Guide* for this area, in particular for backup encryption and splitting the backup role from the standard DBA role.

7. **Backup Process:** The backup process must be as secure as the Oracle database it is supposed to backup. Configuration files must be encrypted to avoid malevolent use (for example running an unwanted "Restore" or changing the target for a backup). Refer to the security guide of the backup software for such risk analysis.

8. **Oracle Advanced Analytics Mining GUI:** (or any other mining tools GUI): Power users that have access to mining normally also have access to most of the data in clear text (to be able to run the mining and to build the models). They should be treated correspondingly from a data access perspective with much care, as their power, unless explicitly limited within their role, is similar to the DBA. Such power users may also use significant resources (for example CPUs) to run their models, possibly unwillingly deploying a self-developed model to the full customer base on a production system at an unplanned time, making the data warehouse slow down such that it becomes unusable. On top of a secure access to the database (with hopefully encrypted configuration files), the GUI for mining users should be protected (and access traced), as for the DBA GUI (5).

9. **Oracle Analytics Workspace Manager GUI:** (or any GUI accessing the database OLAP cubes or creating M/R/HOLAP cubes in Oracle Airlines Data Model or accessing data from Oracle Airlines Data Model): Security risk and mitigations should be as for the mining GUI (7).

10. **WebLogic or BI Server Access:** (to Oracle Airlines Data Model): The BI Server that serves as "BI windows" to Oracle Airlines Data Model end users is a boundary of trust, independent of the BI tool used (including WebLogic or other tools) The configuration file and the connection to access Oracle Airlines Data Model should be protected according to the security guidelines for the corresponding tool.

> **See Also:**
>
> For more information for WebLogic guidelines see *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server*

11. **External Data Feed:** (outgoing): When information is directly picked up from Oracle Airlines Data Model database by an external application (campaign management or monitoring tools for example, to obtain the churn probability and to generate the best offer for a customer) or when Oracle Airlines Data Model itself is setup to export on a regular basis directly to a file, using a database trigger or Spool scripts for example, the data exported should be correspondingly:

   - Anonymized (as part of the export process) or encrypted (before being transmitted) according to the sensitivity of the content.

   - Stored in a place that is only accessible for the purpose required, and not changeable, possibly encrypted or unreadable except by the targeted application.

   - In a format and with the data expected (quality check).

   - Removed after use by the target application.

   Check the appropriate database security guide to protect the external feed and the security guide of the application when direct load or query to the database.

12. **OBIEE Administrator GUI:** The administrator is usually accessing the BI tools and manages the access rights and data accessed as well as the way they are organized and presented to the end-users. With OBIEE, the repository (rpd – default in Oracle Airlines Data Model is `oadm.rpd`) and the webcat files are the most important to protect.

   The OBIEE admin should remove or restrict access to OADM_SAMPLE after it is no longer required, change the default password of OADM RPD, and define all the users (individually), associated with database users. Additionally, the OBIEE administrator should define various groups with limited rights (at OBIEE level) and associate each user with one of those groups. It is important to implement a 2 level security:.

   - Database level (I can only access the data I am entitled to – independently of what rights I am given at BI level)

   - OBIEE level (I can only access the reports and create new reports or leverage OBIEE features on data I am entitled to see and/or use).

   > **See Also:**
   >
   > For more information for WebLogic guidelines see *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server*

13. **OBIEE End-Users:** The end-users usually access their data through their browser on their mobile device or from their desktop.

   > **See Also:**
   >
   > For more information for WebLogic guidelines see *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server*

14. **WebLogic Outside or Other Incoming Feed:** This corresponds to the loading of external data to the WebLogic server to combine them with Oracle Airlines Data

Model output. It is proper to the WebLogic server and should be handled through *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server* instructions. An example of load is an end-users loading an Excel file with 3rd party data (weather data, market share data, and so on) to enrich its own reports. If this file is corrupted or prepared with a malevolent intention, it can represent a risk for the complete architecture.

15. **WebLogic Outside or Other (outgoing) Feeds:** This is exported from WebLogic server to the external world (as MS object or flat file, SMS or email, web agent, and so on). This can become particularly sensitive depending on the information sent or its form of management. Such external feeds must be handled as referred to in *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server*.

16. **BI Server Backup**: This is similar to the Oracle Airlines Data Model backup, the BI server backup is to be secured in a similar way as for the Database.

See Also:

For more information for WebLogic guidelines see *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server* to securely handle backups and see the *Oracle Database Security Guide*

---

17. **Application GUI:** (that manages the access and data loaded directly from the Oracle Airlines Data Model database): Such applications should be similarly protected (encrypted where possible), monitored and limited (from a data access) as for a normal end-user. Avoid general application (generic) user accounts unless the information is not accessible by anyone else but the application itself (no external communications – internal processes only). Always use the "user ID" that logs into the application and do not use a generic user account.

6.3 Security by Component

Outlines security by component. To obtain the highest level of security you must leverage all the possible security options you can within your budget and resources. The default security provided is better than no security, but you should provide enough resources to obtain the highest possible security standard.

Security Component by Component

- Operating System Security

  This area is not Oracle Airlines Data Model specific. For example, if the Linux OS is used, see the following documents:

  – Guide to the Secure Configuration of Red Hat Enterprise Linux 5

  – Hardening Tips for the Red Hat Enterprise Linux 5

- Oracle Database Security and WebLogic Server Security

  Not Oracle Airlines Data Model specific.

6.4 Performing a Secure Oracle Airlines Data Model Installation

Presents information to plan for a secure Oracle Airlines Data Model installation. Outlines the steps required to perform a secure installation.

Pre-Installation Configuration

On the Oracle Database, after installation of partitioning, Database OLAP, and mining, the simplest recommendation is first to change all the default passwords (in particular of SYS and SYSTEM) and lock all accounts and schema that are not used.

Installing Oracle Airlines Data Model Securely

You can perform a custom installation or a typical installation. Perform a custom installation to avoid installing options and products you do not need. If you perform a typical installation, remove or disable features that you do not need after the installation.

On the Oracle Database, during the installation process, first unlock and set passwords for OADM_SYS (and OADM_SAMPLE if installed). For example, use ALTER *USER* IDENTIFIED BY SQL command.

The following are areas with default passwords:

- OADM_SYS

- OADM_SAMPLE (to be removed at least for pre-production and production).

- OADM_USER

- RPD Admin default password

- OADM _REPORT user OBIEE default password

Define the list of people and applications that must access Oracle Airlines Data Model, then determine the data they require the constraints to their access (the complete data set or only a subset), and for each type of profile define a role at the database level with corresponding privileges. (CREATE ROLE command, and GRANT *ROLE, PRIVILEGEs* TO *ROLE / USER* command).

Create as many schemas as you need for each end-user that accesses Oracle Airlines Data Model and specify the correct roles to limit access as required.

When you use OBIEE for reporting, change the default passwords of the RPD (see OBIEE Admin Guide for this) and change also the user password. Create a user for each end-user, with groups, rights, and roles as you did for the Database but at the OBIEE level.

Post-Installation Configuration

There is nothing specific for this area except that you must encrypt any custom scripts that you create and use with Oracle Airlines Data Model.

6.5 Implementing Data Model Security

Explains the Oracle Airlines Data Model security features for tables and entities. This topic lists security sensitivity at various levels for Oracle Airlines Data Model entities.

Implementing Data Model Security

Oracle Airlines Data Model is an option of the database and it does not contain product specific security features. However to secure Oracle Airlines Data Model you must apply the rules of a secured data warehouse as explained in the *Oracle Database Security Guide*. The security rules must be applied and enforced.

The Oracle Airlines Data Model includes entities and some entities must be protected at various levels of security based on the sensitivity of the data they store.

The range for secure entities in the listing ranges from 1 (very sensitive) to 5 (public data).

Table 6-1   Entity Sensitivity Levels

Level	Type of data	Actions recommended
1 Highly sensitive	payment data, and so on.	Encrypt by default (at least the sensitive fields). Audit, do not allow backup (without encryption), limit roles accessing them
2 Sensitive	address, age, resource, and so on.	Mask by default (at least sensitive fields), audit from time to time, limit roles accessing them
3 Somewhat sensitive	No value	Limit access
4 Not sensitive	No value	No value
5 Public	Publicly available data (weather, and so on.)	No value

Note:

By default, any information in Oracle Airlines Data Model is somewhat sensitive. Even if no data is stored in an entity, the entity should be protected from access.

6.6 Security Maintenance, Monitoring and Control

Provides information on controls for maintaining security for Oracle Business Intelligence Suite Extended Edition (OBIEE) and the Oracle Database. You must ensure that you remove or at least lock all unused users at the OBIEE and database level, and perform regular monitoring of user permissions and usage.

Users and Schema Management at Database and OBIEE Level

Remove (or at least lock) all unused users from both the OBIEE and the database level.

Monitor and automatically lock users without activity for the last *N* or 60 days (on OBIEE and on Oracle Airlines Data Model). Providing a regular schedule for locking or resetting for the password for accounts every 120 days, to force a password change, is also a good practice.

6.7 Security Considerations for Developers

Provides information for developers about how to create secure applications for Oracle Airlines Data Model, and how to extend Oracle Airlines Data Model without compromising security.

Security Considerations for Developers

Oracle Airlines Data Model developers work on the following levels:

- **Database level**: In Oracle Airlines Data Model itself, typically for extensions and Customization, from tables to Intra-ETLs up to OLAP cubes or Mining models.

- **ETL level**: In the staging area typically, to move data from source to Oracle Airlines Data Model, while taking care of quality (correctness of content and quantity, timely arrival, meaningfulness for the scope), coherence, latency (or time coherence) and correctness (of mapping to Oracle Airlines Data Model).

- **OBIEE level**: Development (extensions, customizations) of dashboards, reports, alerts, web agents, guided analytics, users and user interfaces.

Each developer should work only on the test environment (except those who test performance in a pre-production environment). Developers should have access to a limited extract of the data that only concerns their scope of work. If possible, provide a dummy set of sample data, as similar to a real data set as possible, for unit testing before testing the system with real data.

If it is not possible to use sample data, in most cases, you must encrypt sensitive data on your site to meet a regulatory compliance for customer and payment information. Also, consider encrypting the source files or input files if sensitive information is stored there.

6.8 Dealing with Data Privacy, Data Retention, and other Data Related Rules and Laws

Describes regulations and covers dealing with data privacy, other data related rules and laws.

Dealing with Data Privacy, Data Retention, and other Data Related Rules and Laws

Each country has its own regulations with respect to data privacy, data allowed and forbidden to be stored and used, and the time for which it must be stored and available for criminal or state inquiry.

For example, in Europe, you must consider the laws at least for data retention, that includes:

- **DIRECTIVE 2006/24/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL - 15 March 2006,** which amended the **Directive 2002/58/EC** for Data privacy of Electronic communications.

These items specify rules for data for quality, security, and confidentiality.

Another concrete example: some fields need to be removed from Oracle Airlines Data Model(or not used at least) in Europe: ETHNICITY field in PARTY is not allowed. The Opt-in and Opt out option of customers must be leveraged by campaign management systems. The Op-in and Opt out options are available in the Social Network Analytics option of Oracle Airlines Data Model.

Because it is locally specific, you should review the Oracle Airlines Data Model content for the project and data you are considering using with your data protection staff (or review the law currently applicable).

6.9 Database Vault on Oracle Airlines Data Model

Describes how to use Database Vault with Oracle Airlines Data Model.

DBA Privileges

It is known that the user who has DBA privilege can see data in OADM_SYS schema, Oracle Database Vault provides powerful security controls to help protect application data from unauthorized access, and comply with privacy and regulatory requirements.

Describes how to register Oracle Database Vault from SQL*Plus in a non-multitenant environment.

You can query the V$OPTION dynamic view to verify if Oracle Database Vault is configured and enabled.

Describes how to secure the OADM_SYS Schema.

6.9.1 Registering Oracle Database Vault with an Oracle Database

Describes how to register Oracle Database Vault from SQL*Plus in a non-multitenant environment.

Registering Oracle Database Vault with an Oracle Database

You can register Oracle Database Vault from SQL*Plus in a non-multitenant environment.

1. Log into the database instance as user SYS with the SYSDBA administrative privilege.

```
sqlplus sys as sysdba
Enter password: password
```

2. As user SYS with the SYSDBA administrative privilege, configure the Database Vault user accounts.

```
SQL>CREATE USER dbv_owner IDENTIFIED BY dbv_owner;
SQL>CREATE USER dbv_acctmgr IDENTIFIED BY dbv_acctmgr;
SQL>GRANT CREATE SESSION TO dbv_owner, dbv_acctmgr;
SQL>BEGIN
DVSYS.CONFIGURE_DV (
dvowner_uname => 'dbv_owner',
dvacctmgr_uname => 'dbv_acctmgr');
END;
/
```

3. Run the `utlrp.sql` script to recompile invalidated objects. If the script gives you any instructions, then follow them, and then run the script again. If the script terminates abnormally without giving any instructions, then run the script again.

    SQL>@$ORACLE_HOME/rdbms/admin/utlrp.sql

4. Connect as the Database Vault Owner user that you just configured.

    ```
    SQ>CONNECT dbv_owner
    ```

    Enter password: password

5. Enable Oracle Database Vault.

    SQL>EXEC DBMS_MACADM.ENABLE_DV;

6. Connect with the SYSDBA administrative privilege.

    SQ>CONNECT / AS SYSDBA

7. Restart the database.

```
SQL>SHUTDOWN IMMEDIATE
SQL>STARTUP
```

6.9.2 Verifying That Oracle Database Vault Is Configured and Enabled

You can query the V$OPTION dynamic view to verify if Oracle Database Vault is configured and enabled.

To find if Oracle Database Vault is configured and enabled, run the following query, which should show the VALUE setting as TRUE.

1. Query the V$OPTION dynamic view as follows:

```
SQL>SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Oracle Database Vault';
```

2. To check if Oracle Label Security is enabled, query V$OPTION as follows:

```
SQL>SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Oracle Label Security';
```

6.9.3 Logging into Oracle Database Vault

From Oracle Enterprise Manager Cloud Control (Cloud Control), you can use the Oracle Database Vault pages to administer and monitor Database Vault-protected databases from a centralized console, automate alerts, view Database Vault reports, and propagate Database Vault policies to other Database Vault-protected databases.

1. Ensure that you have configured the Cloud Control target databases that you plan to use with Database Vault.

2. If necessary, register Oracle Database Vault.

3. Start Cloud Control.

   For example:

   ```
   https://myserver.example.com:7799/em
   ```

4. Log into Cloud Control as user SYSMAN.

5. In the Cloud Control home page, from the **Targets** menu, select **Databases**.

6. In the Databases page, select the link for the Oracle Database Vault-protected database to which you want to connect. The Database home page appears.

7. From the **Security** menu, select **Database Vault**.The Database Login page appears.

8. Enter the following information:

   Username: Enter the name of a of a user who has been granted the appropriate Oracle Database Vault role(dbv_owner).

   The Database Vault home page appears.

6.9.4 Securing OADM_SYS Schema from DBA Access

Describes how to secure the OADM_SYS Schema.

1. Log On as SYSTEM to access the OADM_SYS Schema.

2. Create a Realm.

a. Log into Oracle Database Vault Administrator from Cloud Control as a user who has been granted the DV_OWNERor DV_ADMIN role and the SELECT ANY DICTIONARY privilege.

"Logging into Oracle Database Vault" on page 3-7 explains how to log in.

b. In the Administration page, under Database Vault Components, click **Realms**. (It should be selected by default.)

c. In the Realms page of Oracle Database Vault Administrator, click **Create**.

d. In the Create Realm page, under General, enter OADM Apps after Name.

e. In the Description field, enter Realm to protect the HR schema.

f. After Status, ensure that Enabled is selected so that the realm can be used.

g. Under Audit Options, ensure that Audit On Failure is selected so that you can create an audit trial later on.

h. Click **Next** to display the Realm secured objects page.

i. Click the Add button and in the Add Secured Object dialog box, enter the following information:

Owner: Enter OADM_SYS to select the OADM_SYS schema.

Object Type: Enter TABLE.

Object Name: Enter DWB_ACCT_PYMT.

j. Click **OK**.

The OADM_SYS.DWB_ACCT_PYMT table is added to the Create Realm : Realm Secured Objects page

k. Click **Done**, and then click **Finish**.

3. Create the OADM Manager User Account.

At this stage, there are no database accounts or roles authorized to access or otherwise manipulate the database objects the realm protects. So, the next step is to authorize database accounts or database roles so that they can have access to the schemas within the realm. Create the OADM_MGR user account.

a. .In SQL*Plus, connect as the Database Vault Account Manager, who has the DV_ACCTMGR role, and create the local user OADM_MGR.

```
SQL>conn dbv_acctmgr/dbv_acctmgr;
SQL>GRANT CREATE SESSION TO OADM_MGR identified by OADM_MGR;
```

b. Connect as SYS with the SYSDBA privilege, and then grant SEBASTIAN the following additional privilege:

```
SQL>conn / as sysdba;
SQL>GRANT READ ANY TABLE TO OADM_MGR;
```

4. Create an Authorization for the Realm.

At this stage, even though OADM_MGR has the SELECT ANY TABLE privilege, he cannot select from the OADM_SYS.DWB_ACCT_PYMT table because it is protected by a realm.

Next, authorize user OADM_MGR to have access to the OADM Apps realm as follows:

a. In the Realms page of Database Vault Administrator, select OADM Apps in the list of realms, and then click **Edit**.

b. Click **Next** until you reach the Realm authorizations page.

c. Click **Add**and then enter the following information in the Add Authorizations dialog box:

   • Realm Authorization Grantee: Enter OADM_MGR.

   • Realm Authorization Type: Select Participant from the list.

   • Realm Authorization Ruleset: Leave this field blank.

d. Click **OK**.

e. Click **Done**, and then **Finish**.

5. Test the Realm.

To test the realm, access the DWB_ACCT_PYMT table as a user other than OADM_SYS. The SYSTEM account normally has access to all objects in the OADM schema, but now that you have safeguarded the DWB_ACCT_PYMT table with Oracle Database Vault, this is no longer the case. Connect as SYSTEM, and then access the account balance information in the DWB_ACCT_PYMT table again:

```
SQL>CONNECT SYSTEM
Enter password: password
```

The following output should appear:

Error at line 1:

ORA-01031: insufficient privileges

SQL>CONNECT OADM_MGR

Enter password: password

Query data in DWB_ACCT_PYMT,data appears

6.10 Data Masking on Oracle Airlines Data Model

Describes how to use Data Masking with Oracle Airlines Data Model.

Data Masking Overview

When performing real-world testing, there is the risk of exposing sensitive data to non-production users in a test environment. The Data Masking feature of the Enterprise Manager for Oracle Database Plug-in enables you to securely manage test data.

Describes steps for using data masking with Oracle Airlines Data Model.

6.10.1 Using Data Masking

Describes steps for using data masking with Oracle Airlines Data Model.

Assume there is sensitive data showing bank card number information where masking is required before sending the data to a test environment.

1. Go to EM cloud control, Security->Application Data Model.

2. Create Application Data Model.

3. Enter Application Data Model name and select source database.

4. Enter the database credentials and select OADM_SYS schema.

5. Enter the job name.

6. Job submitted successfully.

7. Job done.

8. Edit the Application Data Model.

9. Add sensitive column.

10. Select column BNK_CARD_NBR in table DWR_BNK_CARD.

11. Sensitive column added.

12. Set sensitive column type.

13. Set sensitive column type to 'CREDIT_CARD_NUMBER'.

14. Sensitive column type updated.

15. Go to Security->Data Masking and Subsetting->Data Masking Definitions.

16. Data masking definition.

17. Create data masking definition.

18. Add column BNK_CARD_NBR.

19. Set format entry to random number.

20. Set the random number from 100000000 to 999999999.

21. Data masking definition created.

22. Generate data masking script.

23. Schedule data masking script job.

24. Generating masking script.

25. Masking script generating job succeeded.

26. Masking script generated.

27. Schedule data masking job.

28. Data masking job submitted successfully.

29. Data masking job succeeded.

30. Now, check BNK_CARD_NBR, it is masked.

6.11 Transparent Data Encryption in Oracle Airlines Data Model

Describes how to configure Transparent Data Encryption (TDE), and demonstrates using TDE (making one encrypted column and one encrypted tablespace).

Transparent Data Encryption (TDE) stops would-be attackers from bypassing the database and reading sensitive information from storage by enforcing data-at-rest encryption in the database layer. Applications and users authenticated to the database continue to have access to application data transparently (no application code or configuration changes are required), while attacks from OS users attempting to read sensitive data from tablespace files and attacks from thieves attempting to read information from acquired disks or backups are denied access to the clear text data.

Configuring a Software Keystore (page 6-19)
> Describes the steps required to configure a Software Keystore.

Demonstration of Oracle Airlines Data Model Working with TDE (page 6-19)
> Demonstrates TDE with one IETL package (PKG_DWD_BKG_FCT) as a TDE example.

6.11.1 Configuring a Software Keystore

Describes the steps required to configure a Software Keystore.

Steps show how to configure TDE, and demonstrate after using TDE, making one encrypted column and one encrypted tablespace, how Oracle Airlines Data Model can work transparently.

1. Set the Software Keystore Location in the `sqlnet.ora` file.

```
NAMES.DIRECTORY_PATH (TNSNAME, EXCONNECT)
  ERYPTION_WALLET_LOCATION =
    (SOURCE=(METHOD=FILE)(MTHOD_DATA=
      (DIRECTORY=/biaora/home/app/biaora/admin/orcl12102/wallet)))
```

2. Create the Software Keystore:

```
SQL>ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/biaora/home/app/biaora/admin/
orcl12102/wallet' IDENTIFIED BY "password";

keystore altered
```

3. Open the Software Keystore:

```
SQL>ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "password";

keystore altered

SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "password" with backup;
```

6.11.2 Demonstration of Oracle Airlines Data Model Working with TDE

Demonstrates TDE with one IETL package (PKG_DWD_BKG_FCT) as a TDE example.

1. Query that shows dependent tables for PKG_DWD_BKG_FCT.

Figure 6-2   Query Listing Tables for PCK_DWD_BKG_FCT

2. Query shows part of data in table DWB_BKG. including the column FLT_NBR that is encrypted in a later step.

Figure 6-3   Shows Query for DWB_BKG with FLT_NBR Column

3. Query shows data and current tablespace of table DWR_FLT, later we will move it to one encrypted tablespace.

Figure 6-4   Data and Table for DWR_FLT

4. Query shows encrypted columns.

Figure 6-5   Query to Show Encrypted Columns

5. Showing current encryption property for each tablespace.

Figure 6-6   Showing Encryption Property for Tablespaces

6. Encrypt column FLT_NBR.

Figure 6-7   Alter DWB_BKG and Encrypt Column FLT_NBR
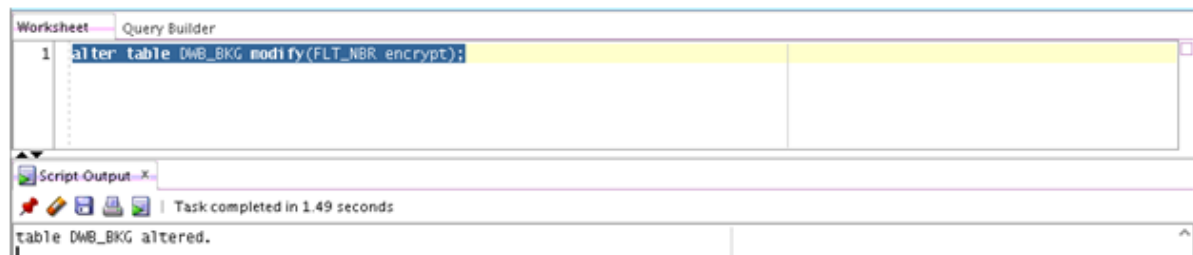
7. Now, column FLT_NBR is encrypted.

Figure 6-8   List the FLT_NBR Column in DWB_BKG Encrypted
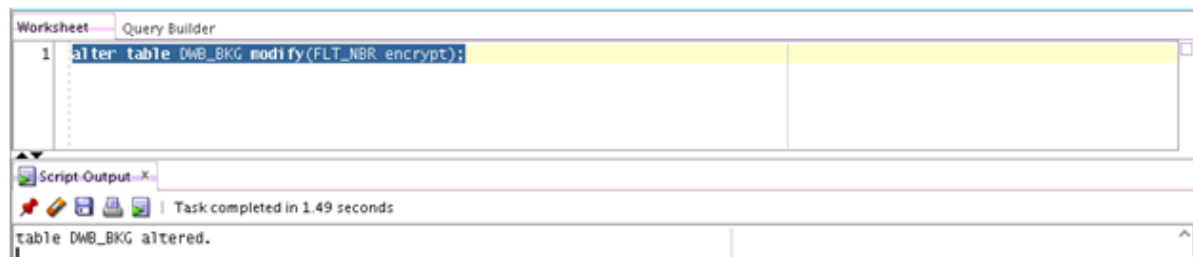
8. Shows the command to create the encrypted tablespace: TBS_REFERENCE_TDE:

    ```
    CREATE SMALLFILE TABLESPACE "TBS_REFERENCE_TDE" DATAFILE '/biaora/home/app/biaora/
    oradata/orcl12102/TBS_REFERENCE_TDE' SIZE 64M LOGGING EXTENT MANAGEMENT LOCAL
    SEGMENT SPACE MANAGEMENT AUTO DEFAULT STORAGE(ENCRYPT) ENCRYPTION USING 'AES256';
    ```

9. List the encrypted value for tablespaces, including: TBS_REFERENCE_TDE:

    ```
    Select tablespace_name,encrypted from sys.user_tablespaces where tablespace_name
    like 'TBS_%'
    ```

Figure 6-9   Show the TBS_REFERENCE_TDE Tablespace
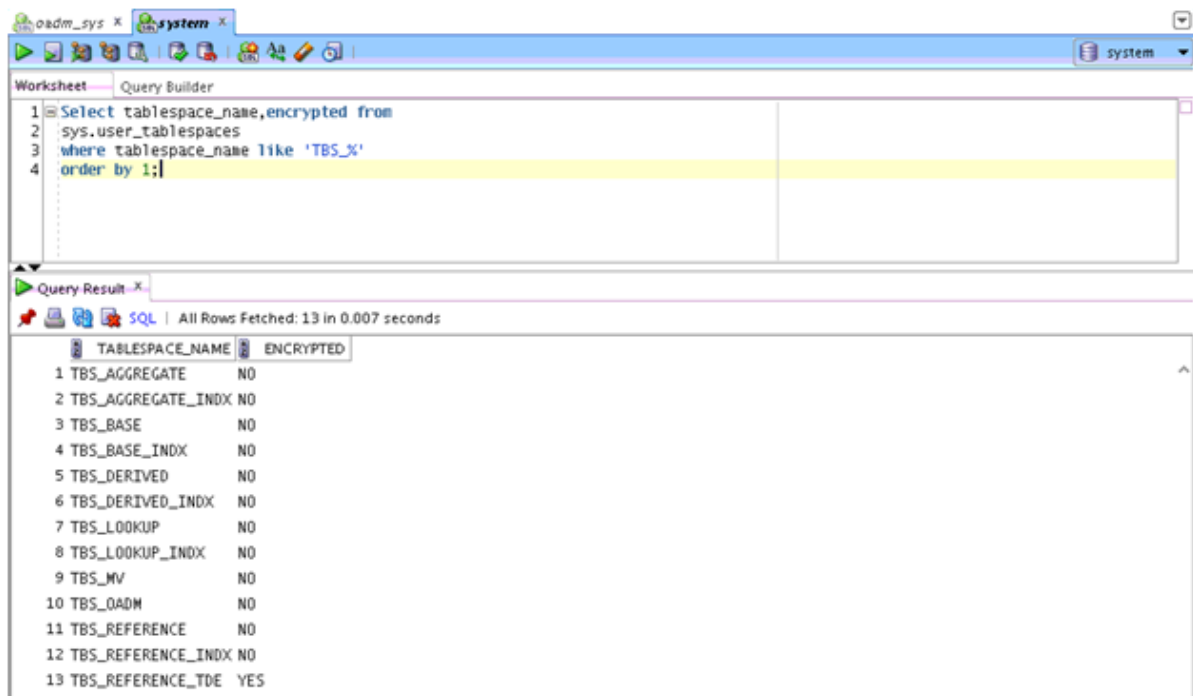
10. Shows the commands to grant privileges on the new TBS_REFERENCE_TDE tablespace:

```
SQ> ALTER USER OADM_SYS  QUOTA UNLIMITED ON TBS_REFERNCE_TDE;

user OADM_SYS altered.
```

11. Shows the commands to move the table DWR_FLT to the encrypted tablespace:

```
SQL>ALTER TABLE OADM_SYS.DWR_FLT MOVE TABLESPACE "TBS_REFERENCE_TDE";

SQL>BEGIN
DBMS_STATS.GATHER_TABLE_STATS('OADM_SYS','DWR_FLT',estimate_percent=>NULL,cascade=
>TRUE);

END;
/
```

12. Shows the query that list the table DWR_FLT in the encrypted TBS_REFERENCE_TDE tablespace.

Figure 6-10   Shows the Table Name and Tablespace for DWR_FLT
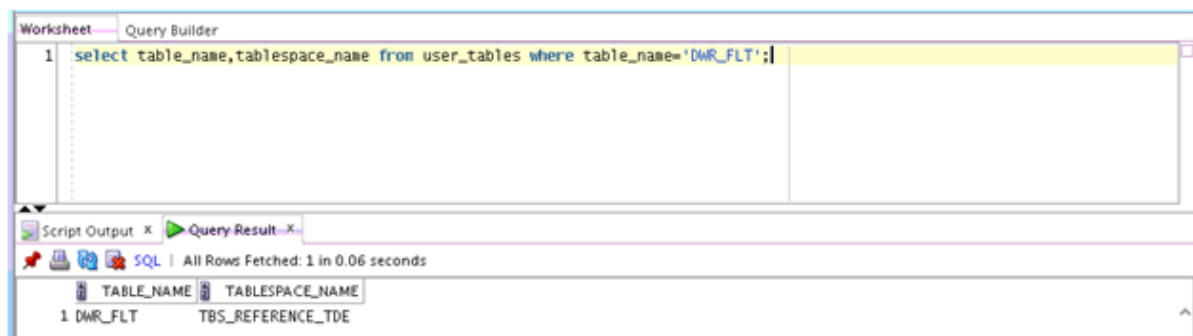
13. Shows the content of the table DWD_BKG_FCT. Before running the IETL, the target table DWD_BKG_FCT is empty:

Figure 6-11   Shows the Empty DWD_BKG_FCT Count
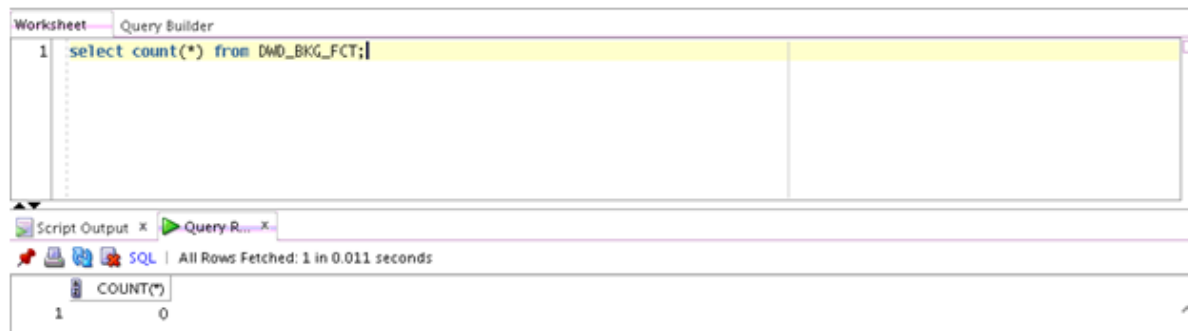
14. Run the IETL process to populate the DWD_BKG_FCT table:

Figure 6-12   Shows a Run of the IETL Process
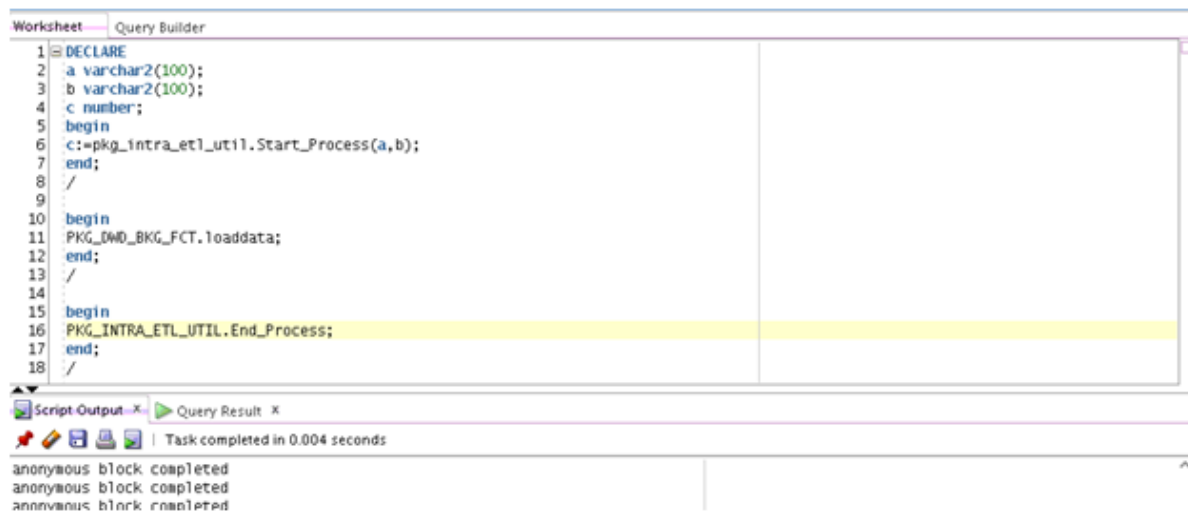
15. Now the target table has data.

Figure 6-13   Shows a Nonzero Count for DWD_BKG_FCT
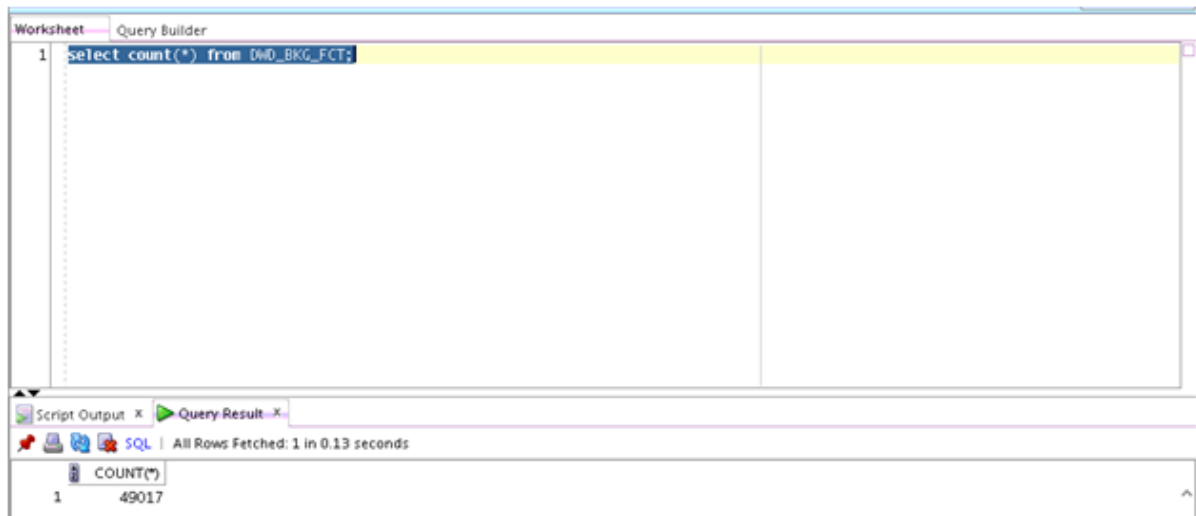
Although the column DWB_BKG.FLT_NBR is encrypted and the table DWR_FLT is moved to the encrypted tablespace, this is transparent to Oracle Airlines Data Model.

A

Sizing and Configuring an Oracle Airlines Data Model Warehouse

Provides information about sizing and configuring an Oracle Airlines Data Model warehouse.

Sizing an Oracle Airlines Data Model Warehouse (page A-1)
> Businesses now demand more information sooner and are delivering analytics from their Enterprise Data Warehouse (EDW) to an ever-widening set of users and applications. To keep up with this increase in demand the EDW must now be near real-time and be highly available. Regardless of the design or implementation of a data warehouse, the initial key to good performance lies in the hardware configuration used.

Configuring a Balanced System for Oracle Airlines Data Model (page A-3)
> Many data warehouse operations are based upon large table scans and other I/O-intensive operations, which perform vast quantities of random I/Os. To achieve optimal performance the hardware configuration must be sized end to end to sustain this level of throughput.

A.1 Sizing an Oracle Airlines Data Model Warehouse

Businesses now demand more information sooner and are delivering analytics from their Enterprise Data Warehouse (EDW) to an ever-widening set of users and applications. To keep up with this increase in demand the EDW must now be near real-time and be highly available. Regardless of the design or implementation of a data warehouse, the initial key to good performance lies in the hardware configuration used.

But how do you go about sizing such a system? You must first understand how much throughput capacity is required for your system and how much throughput each individual CPU or core in your configuration can drive, thus the number one task is to calculate the database space requirement in your data warehouse.

The data warehouse environment has the following data volume estimate resources:

- The estimated raw data extract from source systems. This estimate affects the ETL system configuration and the stage layer database space in data warehouse system. Because this value is determined by your unique OLTP system, you must calculate this information yourself.

- The space needed for data stored to support the objects defined in the default Oracle Airlines Data Model schema. This appendix provides information you can use to make this calculation.

Calculation Factors When Making a Data Volume Calculation for an Oracle Airlines Data Model Warehouse

Consider the following calculation factors when making a data volume calculation:

- Reference and lookup tables data. Assume this data is permanently stored.

- Base tables data (transaction data). Assume that this data is stored within its life cycle.

- Star schema (derived and summary). Assume that this data is stored within its life cycle.

- Calculate each type of data retention.

- Define how many months or years of each type of tables to retain.

- Calculate data growth.

- Assume that annual growth rate: applies to both transaction and reference data and data in the star schema.

- Assume that annual change rate applies only to reference data.

- Calculate Staging Area data requirements, if proposed.

  > **Tip:**
  >
  > Multiply ETL volume by day by number of days held for problem resolution and re-run of transform with new extract from source systems.

- Calculate data volume for indexes, temporary tables, and transaction logs.

- Calculate the space requirement for business intelligence tools, such as cubes, and data mining.

- Consider the redo log and Oracle ASM space requirement.

- Consider the RAID architecture [RAID 1, 0+1, 5]

- Consider the backup strategy.

- Consider the compress factor if applied.

- Consider the operating system and file system disk space requirements.

Formula to Determine Minimum Disk Space Requirements for an Oracle Airlines Data Model Warehouse

Use the following formula to determine the minimum disk space requirements for an Oracle Airlines Data Model warehouse.

```
Disk Space Minimum Requirements = Raw data size * Database space
factor * (1+GrthperY)nY*OS and File system factor * Compress
Factor * Storage Redundant factor
```

where:

- ```
  Raw data size = (reference and lookup data per year + base/
  transaction data per year + derived and summary data per year
  +staging data +other data(OLAP/Data Mining))
  ```

- `Database space factor = Indexes + Temporary Tables + Logs]`

- `GrthperY` = growth rate per year

- `OS and File system factor` is the install and configuration and maintain space for OS and Database

- `Redundant factor= ASM disk space and RAID factor. [RAID 1=2, RAID 5=1.25 or 1.33]`

- `Compress factor` depends how you apply the compress function. If you using HCC, it has a huge savings in disk space by using compression.

A.2 Configuring a Balanced System for Oracle Airlines Data Model

Many data warehouse operations are based upon large table scans and other I/O-intensive operations, which perform vast quantities of random I/Os. To achieve optimal performance the hardware configuration must be sized end to end to sustain this level of throughput.

This type of hardware configuration is called a balanced system. In a balanced system all components - from the CPU to the disks - are orchestrated to work together to guarantee the maximum possible I/O throughput. I/O performance is always a key consideration for data warehouse designers and administrators. The typical workload in a data warehouse is especially I/O intensive, with operations such as large data loads and index builds, creation of materialized views, and queries over large volumes of data. Design the underlying I/O system for a data warehouse to meet these heavy requirements.

To create a balanced system, answer the following questions:

- How many CPUs are required? What speed is required?

- What amount of memory is required? Data warehouse do not have the same memory requirements as mission-critical OLTP applications?

- How many I/O bandwidth components are required? What is the desired I/O speed?

Each component must be able to provide sufficient I/O bandwidth to ensure a well-balanced I/O system.

The following topics provide more information about configuring a balanced system for Oracle Airlines Data Model:

Maintaining High Throughput in an Oracle Airlines Data Model Warehouse (page A-4)
> The hardware configuration and data throughput requirements for a data warehouse are unique mainly because of the sheer size and volume of data. Before you begin sizing the hardware configuration for your data warehouse, estimate the highest throughput requirement to determine whether current or proposed hardware configuration can deliver the necessary performance.

Configuring I/O in an Oracle Airlines Data Model for Bandwidth not Capacity (page A-4)
> Based on the data volume calculated and the highest throughput requirement, you can estimate the I/O throughput along with back-end ETL process and front end business intelligence applications by time unit. Typically, a value of approximately 200MB per second I/O

throughput per core is a good planning number for designing a balanced system.

Planning for Growth of Your Oracle Airlines Data Model (page A-5)
A data warehouse designer plans for future growth of a data warehouse. There are several approaches to handling the growth in a system, and the key consideration is to be able to grow the I/O system without compromising on the I/O bandwidth.

Testing the I/O System Before Building Oracle Airlines Data Model Warehouse (page A-5)
When creating a data warehouse on a new system, test the I/O bandwidth before creating all of the database data files to validate that the expected I/O levels are being achieved.

Balanced Hardware Configuration Guidelines for Oracle Airlines Data Model (page A-5)
You can reference the follow tips for a balanced hardware configuration.

A.2.1 Maintaining High Throughput in an Oracle Airlines Data Model Warehouse

The hardware configuration and data throughput requirements for a data warehouse are unique mainly because of the sheer size and volume of data. Before you begin sizing the hardware configuration for your data warehouse, estimate the highest throughput requirement to determine whether current or proposed hardware configuration can deliver the necessary performance.

When estimating throughput, use the following criteria:

- The amount of data accessed by queries during peak time, and the acceptable response time

- The amount of data that is loaded within a window of time

A.2.2 Configuring I/O in an Oracle Airlines Data Model for Bandwidth not Capacity

Based on the data volume calculated and the highest throughput requirement, you can estimate the I/O throughput along with back-end ETL process and front end business intelligence applications by time unit. Typically, a value of approximately 200MB per second I/O throughput per core is a good planning number for designing a balanced system.

All subsequent critical components on the I/O path - the Host Bus Adapters, fiber channel connections, the switch, the controller, and the disks - have to be sized appropriately.

When running a data warehouse on an Oracle Real Application Cluster (Oracle RAC) it is just as important to size the cluster interconnect with the same care and caution you would use for the I/O subsystem throughput.

When configuring the storage subsystem for a data warehouse, it should be simple, efficient, highly available, and very scalable. An easy way to achieve this is to apply the S.A.M.E. methodology (Stripe and Mirror Everything). S.A.M.E. can be implemented at the hardware level or by using Oracle ASM (Automatic Storage Management) or by using a combination of both. There are many variables in sizing the I/O systems, but one basic rule of thumb is that the data warehouse system has multiple disks for each CPU (at least two disks for each CPU at a bare minimum) to achieve optimal performance.

A.2.3 Planning for Growth of Your Oracle Airlines Data Model

A data warehouse designer plans for future growth of a data warehouse. There are several approaches to handling the growth in a system, and the key consideration is to be able to grow the I/O system without compromising on the I/O bandwidth.

You cannot, for example, add four disks to an existing system of 20 disks, and grow the database by adding a new tablespace striped across only the four new disks. A better solution is to add new tablespaces striped across all 24 disks, and over time also convert the existing tablespaces striped across 20 disks to be striped across all 24 disks.

A.2.4 Testing the I/O System Before Building Oracle Airlines Data Model Warehouse

When creating a data warehouse on a new system, test the I/O bandwidth before creating all of the database data files to validate that the expected I/O levels are being achieved.

On most operating systems, you can perform the test using simple scripts to measure the performance of reading and writing large test files.

A.2.5 Balanced Hardware Configuration Guidelines for Oracle Airlines Data Model

You can reference the follow tips for a balanced hardware configuration.

- Total throughput = `#cores X 100-200MB` (depends on the chip set)

- Total host bus adaptor (HBA) throughput = Total core throughput

> **Note:**
>
> If the total core throughput is 1.6 GB, then you need four 4 GB HBAs.

- Use one disk controller per HBA port (throughput capacity must be equal).

- Switches must have the capacity as HBAs and disk controllers.

- Use a maximum of ten physical disk per controller (that is, use smaller drives: 146 or 300 GB).

- Use a minimum of 4 GB of memory per core (8 GB if using compress).

- Interconnect bandwidth equals I/O bandwidth (InfiniBand).

Oracle has a series of Optimized Warehouse Reference configurations that help customers take the risk out of designing and deploying Oracle data warehouses. Using extensive field experience and technical knowledge, Oracle and its hardware partners have developed a choice of data warehouse reference configurations that can support various sizes, user populations, and workloads. These configurations are fast, reliable, and can easily scale from 500 GB to over 100 TB on single and clustered servers to support tens to thousands of users.

Index