Oracle9*i*

Sample Schemas

Release 1 (9.0.1)

June 2001

Part No. A90129-01

ORACLE

Oracle9*i* Sample Schemas, Release 1 (9.0.1)

Contributors: Alexander Hunold, Diana Lorentz, Neena Kochhar, Lex de Haan, Nancy Greenberg, Nagavalli Pataballa, Den Raphaely, David Austin, Bill Gietz, Hermann Baer, Shelley Higgins, Brajesh Goyal, Shailendra Mishra, Geoff Lee, Susan Mavris

Contents

Send Us Your Comments

Oracle9*i* Sample Schemas, Release 1 (9.0.1)

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us at:

- E-mail at `infodev_us@oracle.com`
- Postal service:
  Oracle Corporation
  Server Technologies Curriculum Development
  500 Oracle Parkway, Mailstop 4op1090
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Oracle has been using the schema SCOTT with its two prominent tables EMP and DEPT tables for a long time. With advances in Oracle's technology, these tables have become inadequate to show even the most basic features of the Oracle database and other Oracle products. As a result, many other schemas have been created over the years to suit the needs of product documentation, courseware, and software development and application demos.

This preface contains these topics:

Audience

This book is for all users of the seed database that is installed when you install Oracle.

About the Sample Schemas

The new Oracle9*i* Sample Schemas serve the purpose of providing a common platform for examples in Oracle9*i* and future releases. It is not possible to convert all examples throughout our documentation to this new environment at one time, but all examples will be converted as material is updated.

The new Oracle9*i* Sample Schemas are a set of interlinked schemas. This set of schemas is aimed at providing a layered approach to complexity:

- A simple schema (Human Resources, HR) for introducing basic topics. An extension to this schema supports Oracle Internet Directory demos.
- A second schema (Order Entry, OE) for dealing with matters of intermediate complexity. A multitude of datatypes is available in this schema.

  The Online Catalog (OC) subschema is a collection of object-relational database objects built inside the OE schema.
- A schema dedicated to multimedia datatypes (Product Media, PM)
- A set of schemas gathered under the main schema name QS (Queued Shipping) to demonstrate Oracle Advanced Queuing capabilities.
- A schema designed to allow for demos with larger amounts of data (Sales History, SH). An extension to this schema provides support for advanced analytic processing.

What Are the Customer Benefits?

- **Continuity of context**. When encountering the same set of tables everywhere, users, students, and developers spend less time with the schema and more time understanding or explaining the technical concepts.
- **Usability**. Customers can use these schemas in the seed database to run examples that are shown in Oracle documentation and training materials. This first-hand access to examples will facilitate both conceptual understanding and application development.

- **Quality**. Through central maintenance and testing of both the creation scripts that build the Oracle9*i* Sample Schemas and the examples that run against the schemas, the quality of Oracle documentation and training materials will be enhanced.

What are the Oracle9*i* Sample Schemas Design Principles?

- **Simplicity and Ease of Use**. The HR and OE schemas should not become overly complex by the addition of features, but rather should provide a graduated path from the simple to intermediate levels of database use.
- **Be fundamental**. The base schemas and the extensions should bring to the foreground the functionality that customers typically use. Only the most commonly used database objects are built automatically in the schemas, and the entire set of schemas provides a foundation upon which one can expand to illustrate additional functionality.
- **Extensibility**. The Oracle9*i* Sample Schemas provide a logical and physical foundation for adding objects to demonstrate functionality beyond the fundamental scope.
- **Relevance**. The Oracle9*i* Sample Schemas are designed to be applicable to e-business and other significant industry trends (for example, XML). When this goal conflicts with the goal of simplicity, schema extensions are used to showcase the trends in focus.

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle9i Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width font)`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.
`lowercase monospace (fixed-width font)`	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase monospace (fixed-width font) italic`	Lowercase monospace italic font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run `U`*`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`
. . .	Horizontal ellipsis points indicate either: ■ That we have omitted parts of the code that are not directly related to the example ■ That you can repeat a portion of the code	`CREATE TABLE ... AS `*`subquery`*`;` `SELECT `*`col1`*`, `*`col2`*`, ... , `*`coln`*` FROM employees;`
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	`acctbal NUMBER(11,2);` `acct CONSTANT NUMBER(4) := 3;`
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/`*`system_password`* `DB_NAME = `*`database_name`*

UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

Convention	Meaning	Example

`lowercase`	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.
	
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		
		`sqlplus hr/hr`
		
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Documentation Accessibility

Oracle's goal is to make our products, services, and supporting documentation accessible to the disabled community with good usability. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

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

1

Installation

When you do a complete installation of Oracle9*i*, the Sample Schemas are installed automatically with the seed database. If for some reason the seed database is removed from your system, you will need to reinstall the Sample Schemas before you can duplicate the examples you find in Oracle documentation and training materials.

This chapter describes how to install the Oracle9*i* Sample Schemas. It contains the following sections:

Caution: By installing any of the Oracle9*i* Sample Schemas, you will destroy any previously installed schemas that use any of the following user names:

- HR
- OE
- PM
- SH
- QS
- QS_ADM
- QS_WS
- QS_ES
- QS_OS
- QS_CBADM
- QS_CB
- QS_CS

Data contained in any of the above schemas will be lost by running any of the installation scripts described in this section. You should not use Oracle9*i* Sample Schemas for your personal or business data and applications. They are meant to be used for demonstration purposes only.

Using the Database Configuration Assistant

Using DBCA is by far the most intuitive and simple way to install the Sample Schemas. Step 4 of the database creation process lets you configure the Sample Schemas you wish to use in your database. The following dependencies are enforced by the Database Configuration Assistant:

- The checkbox "Example Schemas" needs to be checked for any Sample Schema to be created.
- "Oracle Spatial" needs to be selected to allow the Order Entry schema to be created.

- "Oracle JVM" and "Oracle Intermedia" need to be selected to allow for the creation of the Product Media schema. You can select these two options by clicking on the "Additional database configurations ..." button.
- The Order Entry schema option requires the Human Resources option to be selected.
- The Product Media schema option requires the Order Entry option to be selected.
- The Shipping schema option requires the Order Entry option to be selected.
- Selecting "Oracle OLAP Services" with the Sales History option selected will add OLAP server metadata to the Sales History schema.

Two of the three predefined database templates shipped with the Database Configuration Assistant contain the Sample Schemas:

- OLTP database
- DSS database

Manually Installing the Oracle9*i* Sample Schemas

Prerequisites

The Sample Schemas that are available to you depend on the edition of Oracle you install and its configuration. Please consult the following table to see which schemas you can install:

Schema	Oracle9*i* Personal Edition	Oracle9*i* Standard Edition	Oracle9*i* Enterprise Edition
HR	OK	OK	OK
OE	OK	OK	OK
PM	OK	OK	OK
QS	OK	OK	OK
SH	Not available	Not available	Needs Partitioning Option installed

Schema Dependencies

Various dependencies have been established among the schemas. Therefore, you must create the schemas in the following order: HR, OE, PM, QS, and SH.

> **Note:** To make it easier for you to remember, the Oracle9*i* Sample Schemas are ordered, both in complexity and dependencies, in alphabetical order.

Use this sequence to create the schemas:

1. Create the HR schema.
2. Create the OE schema: The HR schema must already be present, and you must know the password for the HR schema so that you can grant HR object privileges to OE. Some HR tables are visible to the OE user through the use of private synonyms. In addition, some OE tables have foreign key relationships to HR tables.

   > **Note:** The OE schema requires the database to be enabled for spatial data. You can accomplish this during installation or afterward using the Database Configuration Assistant.

3. Create the PM schema: Foreign key relationships require that the OE schema already exist when the PM schema is created. You need to know the password for OE to grant to PM the right to establish and use these foreign keys.

   > **Note:** The PM schema requires the database to be enabled for the Java Virtual Machine (JVM) and *inter*Media. You can accomplish this during installation or afterward using the Database Configuration Assistant.

4. Create the QS schema: The shipping schema QS is based on order entry data in OE. Again, foreign key relationships require that the OE schema already be present when the QS schema is created. You need to know the password for OE to grant to QS the right to establish and use these foreign keys.

5. Create the SH schema. The SH schema logically depends on the OE schema, although there is nothing that prevents you from creating this schema on its own, without the four other schemas.

Installing the Human Resources (HR) Schema

All scripts necessary to create this schema reside in $ORACLE_HOME/demo/schema/human_resources.

You need to call only one script, hr_main.sql, to create all objects and load the data. Running hr_main.sql accomplishes the following tasks:

1. Prompts for passwords and tablespace names used within the scripts
2. Erases any previously installed HR schema
3. Creates the user HR and grants the necessary privileges
4. Connects as HR
5. Calls the following scripts:
   - hr_cre.sql to create data objects
   - hr_popul.sql to populate data objects
   - hr_idx.sql to create indexes on data objects
   - hr_code.sql to create procedural objects
   - hr_comnt.sql to create comments on tables and columns
   - hr_analz.sql to gather schema statistics
6. [Optional] A pair of scripts, sh_dn_c.sql and sh_dn_d.sql are provided as schema extension. To prepare the Human Resources schema for use with the Directory capabilities of Oracle Internet Directory, run the sh_dn_c.sql create script. If you want to return to the initial setup of the HR schema, use the script sh_dn_d.sql to erase the effects of sh_dn_c.sql and erase the column added by this extension.

The file used to drop the HR schema is hr_drop.sql.

Installing the Order Entry (OE) Schema and its Online Catalog (OC) Subschema

All scripts necessary to create this schema reside in $ORACLE_HOME/demo/schema/order_entry.

You need to call only one script, oe_main.sql, to create all objects and load the data. Running oe_main.sql accomplishes the following tasks:

1. Prompts for passwords and tablespace names used within the scripts
2. Erases any previously installed OE schema
3. Creates the user OE and grants the necessary privileges
4. Connects as OE
5. Calls the following scripts:
   - oe_cre.sql to create data, procedural, and user defined objects
   - oe_oe_p_pi.sql to populate the PRODUCT_INFORMATION table
   - oe_p_whs.sql to populate the WAREHOUSES table
   - oe_p_cus.sql to populate the CUSTOMERS table
   - oe_p_ord.sql to populate the ORDERS table
   - oe_p_itm.sql to populate the ORDER_ITEMS table
   - oe_p_inv.sql to populate the INVENTORIES table
   - oe_views.sql to create table views
   - oe_idx.sql to create indexes on data objects
   - oe_comnt.sql to create comments on tables and columns
   - oc_main.sql to create the OC (Online catalog) object-oriented subschema within OE. The oc_main.sql script calls the following scripts:
   - oc_cre.sql to create a sequence of interrelated user defined objects, object tables and views
   - oc_popul.sql to populate object tables
   - oc_comnt.sql to create comments on tables and columns
   - oe_p_pd to populate the PRODUCT_DESCRIPTIONS table. Language-specific INSERT statements for product names and descriptions are stored in these files:
     * oe_p_us.sql
     * oe_p_ar.sql
     * oe_p_cs.sql

```
*  oe_p_d.sql
*  oe_p_dk.sql
*  oe_p_e.sql
*  oe_p_el.sql
*  oe_p_esa.sql
*  oe_p_f.sql
*  oe_p_frc.sql
*  oe_p_hu.sql
*  oe_p_i.sql
*  oe_p_iw.sql
*  oe_p_ja.sql
*  oe_p_ko.sql
*  oe_p_n.sql
*  oe_p_nl.sql
*  oe_p_pl.sql
*  oe_p_pt.sql
*  oe_p_ptb.sql
*  oe_p_ro.sql
*  oe_p_ru.sql
*  oe_p_s.sql
*  oe_p_sf.sql
*  oe_p_sk.sql
*  oe_p_th.sql
*  oe_p_tr.sql
*  oe_p_zhs.sql
*  oe_p_zht.sql
```

- `oe_analz` to gather schema statistics

The files used for dropping the `OE` schema and `OC` subschema are:

- oe_drop.sql
- oc_drop.sql

Installing The Product Media (PM) Schema

All files necessary to create this schema reside in $ORACLE_HOME/demo/schema/product_media.

You need to call only one script, pm_main.sql, to create all objects and load the data. Running pm_main.sql accomplishes the following tasks:

1. Prompts for passwords and tablespace names used within the scripts
2. Erases any previously installed PM schema
3. Creates the user PM and grants the necessary privileges
4. Connects as PM
5. Calls the following scripts:
   - pm_cre.sql

   The list of files used for populating the PM schema includes:
   - pm_p_lob.sql
   - pm_p_lob.ctl
   - pm_p_lob.dat

   Note: The SQL*Loader data file pm_p_lob.dat contains hard-coded absolute path names that have been set during installation. Before attempting to load the data in a different environment, you should first edit the path names in this file.

   - pm_p_ord.sql

The file used to drop the PM schema is pm_drop.sql.

Installing the Queued Shipping (QS) Schemas

All files necessary to create this schema reside in $ORACLE_HOME/demo/schema/shipping.

You need to call only one script, qs_main.sql, to create all objects and load the data. Running qs_main.sql accomplishes the following tasks:

1. Prompts for passwords and tablespace names used within the scripts
2. Erases any previously installed QS schema
3. Creates the user QS and grants the necessary privileges
4. Connects as QS
5. Calls the following scripts:
   - qs_adm.sql creates the Administrator schema
   - qs_cbadm.sql creates the Customer Billing Administration schema
   - qs_cre.sql creates queues, queue tables for the Queued Shipping schema
   - qs_cs.sql creates the Customer Service schema
   - qs_es.sql creates the Eastern Shipping schema
   - qs_os.sql creates the Overseas Shipping schema
   - qs_ws.sql creates the Western Shipping schema
   - qs_run.sql creates the demo application procedures and objects

The file used for dropping all queues in an orderly fashion is qs_drop.sql.

Installing the Sales History (SH) Schema

All files necessary to create this schema reside in $ORACLE_HOME/demo/schema/sales_history.

You need to call only one script, sh_main.sql, to create all objects and load the data. Running sh_main.sql accomplishes the following tasks:

1. Prompts for passwords and tablespace names used within the scripts
2. Erases any previously installed SH schema
3. Creates the user SH and grants the necessary privileges
4. Connects as SH
5. Calls the following scripts:
   - sh_cre.sql to create tables

- sh_pop1.sql to populate the dimension tables COUNTRIES and CHANNELS
- sh_pop2.sql to populate the dimension table TIMES
- sh_pop3.sql to populate the remaining tables. The dimension tables PROMOTIONS, CUSTOMERS, PRODUCTS and the fact table SALES are loaded by SQL*Loader. Then, two directory paths are created inside the database to point to the load and log file locations. This allows the loading of the table COSTS by defining the file sh_sales.dat as an external table.
- sh_idx.sql to create indexes on tables
- sh_cons.sql to add constraints to tables
- sh_hiera.sql to create dimensions and hierarchies
- sh_cremv.sql to create materialized views
- sh_comnt.sql to add comments for columns and tables
- sh_analz.sql to gather statistics

6. [Optional] A pair of scripts, sh_olp_c.sql and sh_olp_d.sql are provided as schema extension. To prepare the Sales History schema for use with the advanced analytic capabilities of OLAP Services, run the sh_olp_c.sql create script. If you want to return to the initial setup of the SH schema, use the script sh_olp_d.sql to erase the effects of sh_olp_c.sql and reinstate dimensions as they were before.

The file used to drop the SH schema is sh_drop.sql.

Resetting the Sample Schemas

In most situations, there is no difference between installing a particular Sample Schema for the first time or reinstalling it over a previously installed version. The *_main.sql scripts drop the schema users and all their objects.

In some cases, complex inter-object relationships in the OE or QS schemas prevent the DROP USER ... CASCADE operations from completing normally. In these rare cases, go through one of the following sequences.

For the OC catalog subschema of the OE schema:

1. Connect as the user OE.
2. Execute the script oc_drop.sql.

3. Connect as SYSTEM.
4. Make sure nobody is connected as OE:

```
SELECT username FROM v$session;
```

5. Drop the user:

```
DROP USER oe CASCADE;
```

For the QS schemas:

1. Connect as SYSTEM.
2. Make sure nobody is connected as a QS user:

```
SELECT username FROM v$session WHERE username like 'QS%';
```

3. Drop the schemas by executing the script qs_drop.sql. You will be prompted for the passwords for the individual users.

2

Rationale

The Oracle9*i* Sample Schemas are based on a fictitious company that sells goods through various channels. This chapter describes the fictitious company and contains these sections:

- Overall Description
- Human Resources (HR)
- Order Entry (OE)
- Product Media (PM)
- Queued Shipping (QS)
- Sales History (SH)

Overall Description

The sample company portrayed by the Oracle9*i* Sample Schemas operates worldwide to fill orders for several different products. The company has several divisions:

- The Human Resources division tracks information on the company's employees and facilities.
- The Order Entry division tracks product inventories and sales of the company's products through various channels.
- The Product Media division maintains descriptions and detailed information on each product sold by the company.
- The Shipping division manages the shipping of products to customer.
- The Sales History division tracks business statistics to facilitate business decisions.

Each of these divisions is represented by a schema.

Human Resources (HR)

In the company's human resource records, each employee has a unique identification number, email address, job identification number, salary, and manager. Some employees earn a commission in addition to their salary, which is also tracked. When an employee switches jobs, the company records the start date and end date of the former job, the job identification number, and department.

The company also tracks information about jobs within the organization. Each job has an identification number, job title, and a minimum and maximum salary range for the job. Some employees have been with the company for a long time and have held different jobs within the company. When an employee switches jobs, the company records the start date and end date of the former job, the job identification number, and the department.

The sample company is regionally diverse, so it tracks the locations of not only its warehouses but also of its departments. Each of the company's employees is assigned to a department. Each department is identified by a unique department code and a short name. Each department is associated with one location. Each location has a full address that includes the street address, postal code, city, state or province, and country code.

For each where it has facilities, the company records the country name, currency symbol, currency name and the region where the county resides geographically.

Order Entry (OE)

The company sells several categories of products, including computer hardware and software, music, clothing, and tools. The company maintains product information that includes product identification numbers, the category into which the product falls, the weight group (for shipping purposes), the warranty period if applicable, the supplier, the status of the product, a list price, a minimum price at which a product will be sold, and a URL address for manufacturer information. Inventory information is also recorded for all products, including the warehouse where the product is available and the quantity on hand. Because products are sold worldwide, the company maintains the names of the products and their descriptions in several different languages.

The company maintains warehouses in several locations to facilitate filling customer orders. Each warehouse has a warehouse identification number, name, and location identification number.

Customer information is tracked in some detail. Each customer is assigned an identification number. Customer records include name, street address, city or province, country, phone numbers (up to five phone numbers for each customer), and postal code. Some customers order through the Internet, so email addresses are also recorded. Because of language differences among customers, the company records the NLS language and territory of each customer.

The company places a credit limit on its customers to limit the amount they can purchase at one time. Some of customers have account managers, which we monitor. We keep track of a customer's phone numbers. In this day, we never know how many phone numbers a customer might have, but we try to keep track of all of them. Because of the language differences of our customers, we identify the language and territory of each customer.

When a customer places an order, the company tracks the date of the order, the mode of the order, status, shipping mode, total amount of the order, and the sales representative who helped place the order. This may be the same individual as the account manager for a customer, it may be different, or, in the case of an order over the Internet, the sales representative is not recorded. In addition to the order information, we also track the number of items ordered, the unit price, and the products ordered.

For each country in which it does business, the company records the country name, currency symbol, currency name, and the region where the county resides geographically. This data is useful customers living in different geographic regions around the world.

Online Catalog (OC) Description

The OC subschema of the OE schema addresses an online catalog merchandising scenario. The same customers and products are used as in the OE schema proper, but the OC subschema organizes the categories that the OE products belong to into a hierarchy of parent categories and subcategories. This hierarchy corresponds to the arrangement on an *e*-commerce portal site where the user navigates to specific products by drilling down through ever more specialized categories of products.

Product Media (PM)

The company stores multimedia and print information about its products in the database. Examples of such information are:

- Promotional audio and video clips
- Product images and thumbnails for web publishing
- Press release texts
- Print media ads
- Other promotion texts and translations

Queued Shipping (QS)

The sample company has decided to test the use of messaging to manage its proposed B2B applications. The plan calls for a small test that will allow a user from outside the firewall to place an order and track its status. The order needs to be booked into the main system. Then, depending on the location of the customer, the order is routed to the nearest region for shipping.

Eventually, the company intends to expand beyond its current in-house distribution system to a system that will allow other businesses to provide the shipping. Therefore, the messages sent between the businesses must also travel over HTTP and be in a self-contained format. XML is the perfect format for the message, and both the Advanced Queueing Servlet and Oracle Internet Directory provide the appropriate routing between the queues.

After the orders are either shipped or back ordered, a message needs to be sent back to appropriate employees to inform them of the order's status and to initiate the billing cycle. It is critical that the message be delivered only once and that there be a system for tracking and reviewing messages to facilitate resolution of any discrepancies with the order.

For the purpose of this test application, the company utilizes a single database server and a single application server. The application provides a mechanism for examining the XML messages as well as looking at the queues. To demonstrate connectivity from outside the firewall, both the generation of a new order and customer service reporting are performed using queues. The new order application directly enqueues a queue, while the customer service queries require XML messaging to dequeue a queue.

The users associated with this application are:

- QS (Queue Shipping)
- QS_ES (Eastern Shipping)
- QS_WS (Western Shipping)
- QS_OS (Overseas Shipping)
- QS_CB (Customer Billing)
- QS_CS (Customer Service)
- QS_ADM (Administration), and
- QS_CBADM (Customer Billing Administration)

Sales History (SH)

The sample company does a high volume of business, so it runs business statistics reports to aid in decision support. Many of these reports are time-based and non-volatile. That is, they analyze past data trends. The company loads data into its data warehouse regularly to gather statistics for these reports. Some examples of these reports include annual, quarterly, monthly, and weekly sales figures by product and annual, quarterly, monthly, and weekly sales figures by product.

The company also runs reports on distribution channels through which its sales are delivered. When the company runs special promotions on its products, it analyzes the impact of the promotions on sales. It also analyzes sales by geographical area.

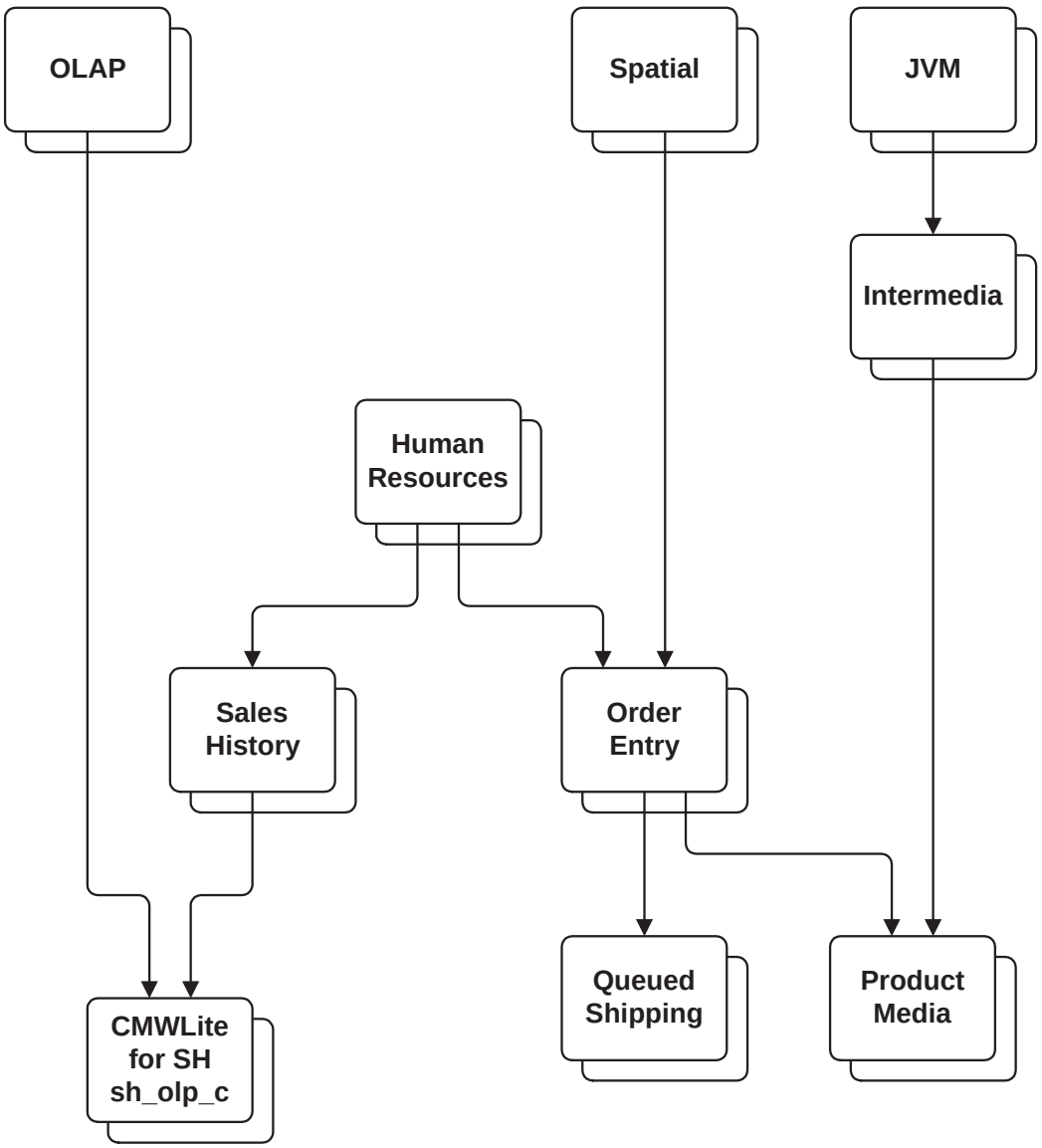
3

Diagrams

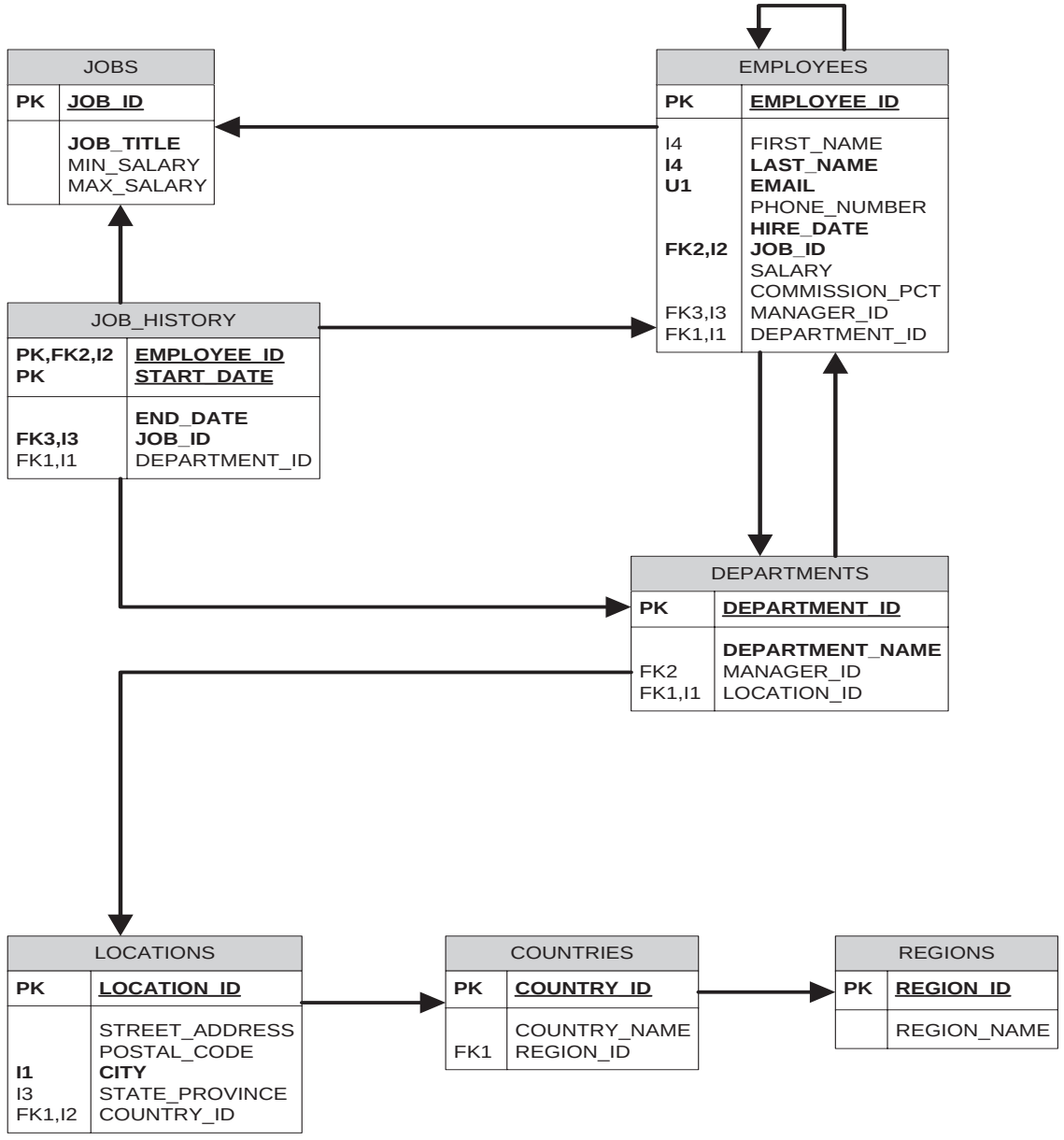
This chapter contains diagrams of the Sample Schemas. The first diagram shows the build order and prerequisites of the Sample Schemas. The remaining diagrams illustrate the configuration of the the various components of each schema.

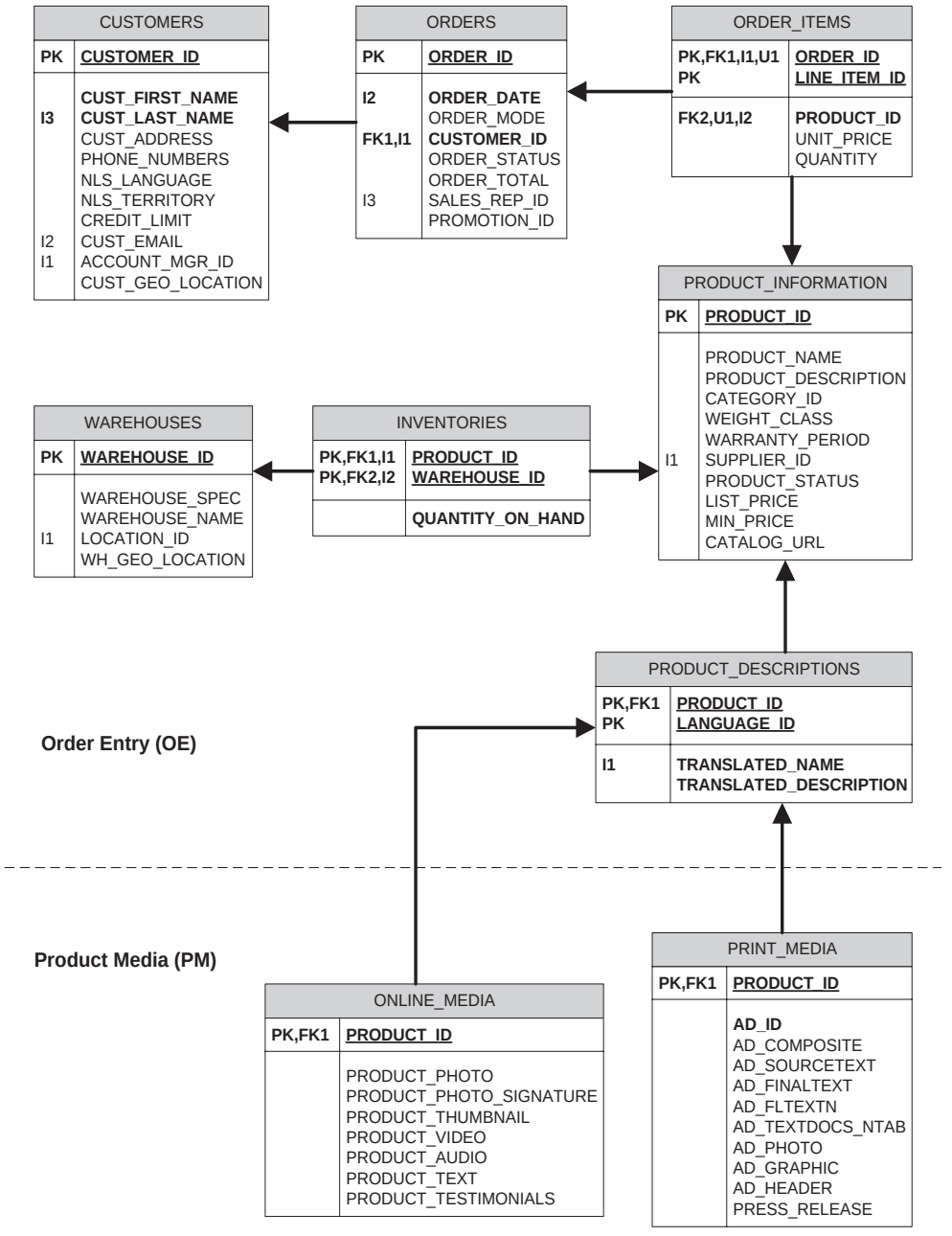
For more detailed information, and for a text description of each schema, please see the schema creation scripts in Chapter 4, "Oracle9i Sample Schema Scripts".

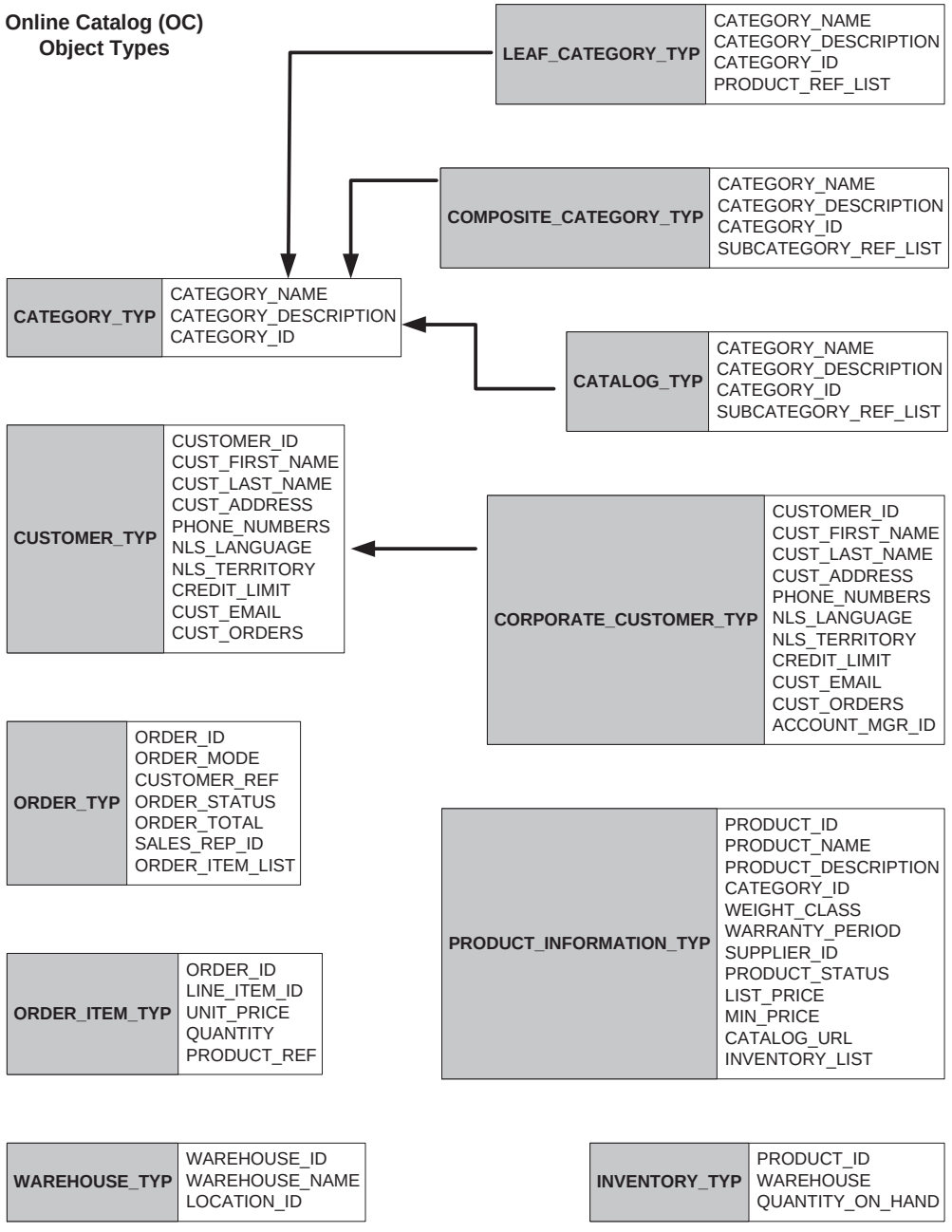
Build Order And Prerequisites For The Oracle9*i* Sample Schemas

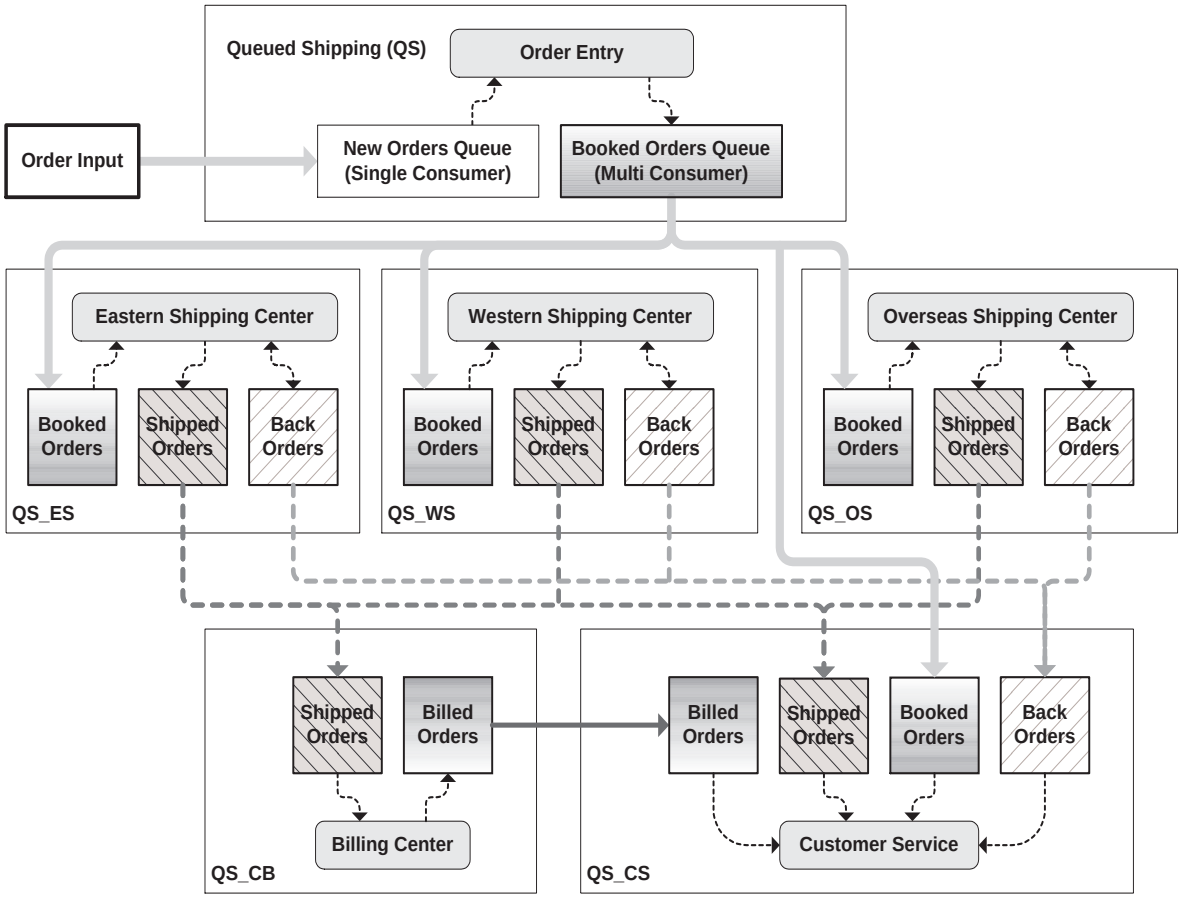


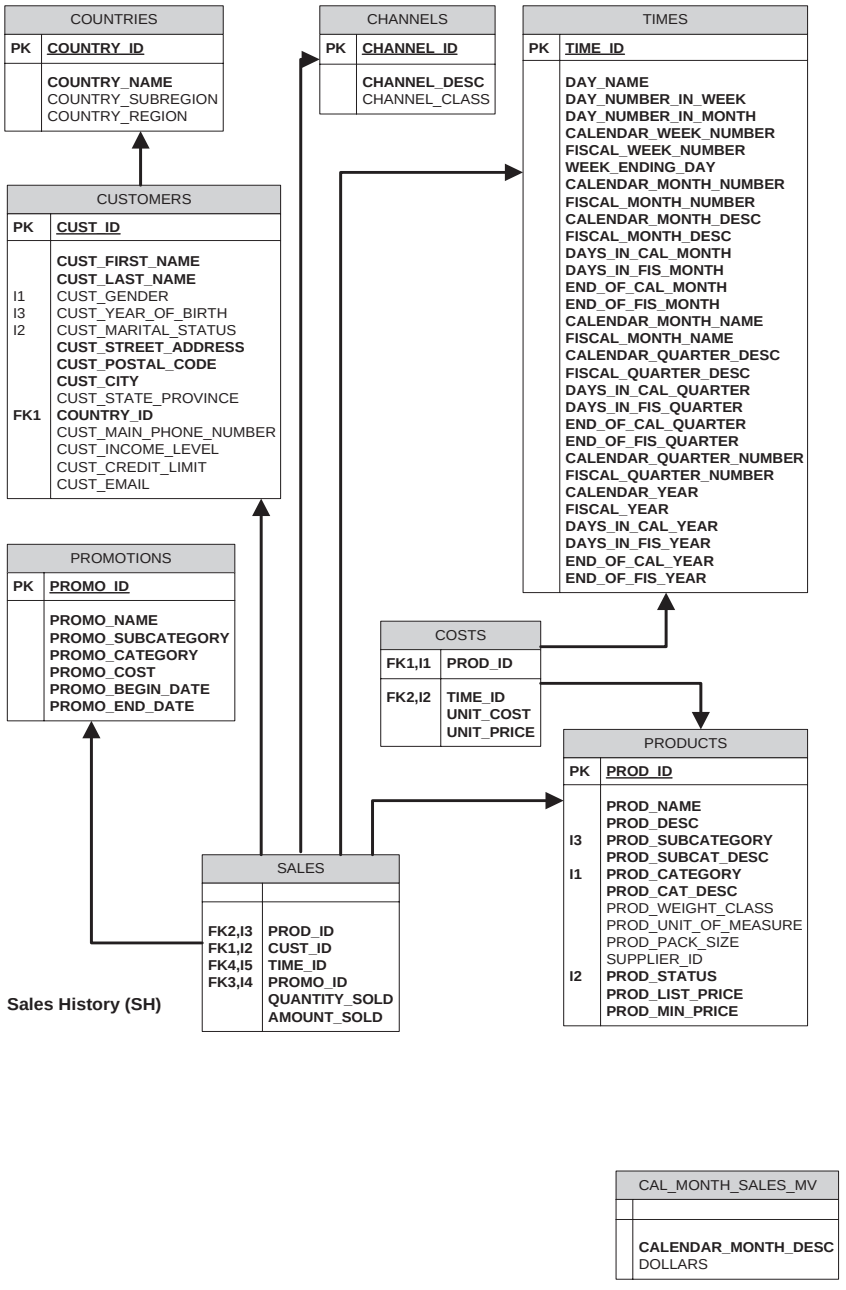
Human Resources (HR)




CUSTOMERS
PK CUSTOMER_ID
CUST_FIRST_NAME
I3 CUST_LAST_NAME
CUST_ADDRESS
PHONE_NUMBERS
NLS_LANGUAGE
NLS_TERRITORY
CREDIT_LIMIT
I2 CUST_EMAIL
I1 ACCOUNT_MGR_ID
CUST_GEO_LOCATION
ORDERS
PK ORDER_ID
I2 ORDER_DATE
ORDER_MODE
FK1,I1 CUSTOMER_ID
ORDER_STATUS
ORDER_TOTAL
I3 SALES_REP_ID
PROMOTION_ID
ORDER_ITEMS
PK,FK1,I1,U1 ORDER_ID
PK LINE_ITEM_ID
FK2,U1,I2 PRODUCT_ID
UNIT_PRICE
QUANTITY
PRODUCT_INFORMATION
PK PRODUCT_ID
PRODUCT_NAME
PRODUCT_DESCRIPTION
CATEGORY_ID
WEIGHT_CLASS
WARRANTY_PERIOD
I1 SUPPLIER_ID
PRODUCT_STATUS
LIST_PRICE
MIN_PRICE
CATALOG_URL
WAREHOUSES
PK WAREHOUSE_ID
WAREHOUSE_SPEC
WAREHOUSE_NAME
I1 LOCATION_ID
WH_GEO_LOCATION
INVENTORIES
PK,FK1,I1 PRODUCT_ID
PK,FK2,I2 WAREHOUSE_ID
QUANTITY_ON_HAND
PRODUCT_DESCRIPTIONS
PK,FK1 PRODUCT_ID
PK LANGUAGE_ID
I1 TRANSLATED_NAME
TRANSLATED_DESCRIPTION
Order Entry (OE)
Product Media (PM)
ONLINE_MEDIA
PK,FK1 PRODUCT_ID
PRODUCT_PHOTO
PRODUCT_PHOTO_SIGNATURE
PRODUCT_THUMBNAIL
PRODUCT_VIDEO
PRODUCT_AUDIO
PRODUCT_TEXT
PRODUCT_TESTIMONIALS
PRINT_MEDIA
PK,FK1 PRODUCT_ID
AD_ID
AD_COMPOSITE
AD_SOURCETEXT
AD_FINALTEXT
AD_FLTEXTN
AD_TEXTDOCS_NTAB
AD_PHOTO
AD_GRAPHIC
AD_HEADER
PRESS_RELEASE


Online Catalog (OC)
Object Types
LEAF_CATEGORY_TYP
CATEGORY_NAME
CATEGORY_DESCRIPTION
CATEGORY_ID
PRODUCT_REF_LIST
COMPOSITE_CATEGORY_TYP
CATEGORY_NAME
CATEGORY_DESCRIPTION
CATEGORY_ID
SUBCATEGORY_REF_LIST
CATEGORY_TYP
CATEGORY_NAME
CATEGORY_DESCRIPTION
CATEGORY_ID
CATALOG_TYP
CATEGORY_NAME
CATEGORY_DESCRIPTION
CATEGORY_ID
SUBCATEGORY_REF_LIST
CUSTOMER_TYP
CUSTOMER_ID
CUST_FIRST_NAME
CUST_LAST_NAME
CUST_ADDRESS
PHONE_NUMBERS
NLS_LANGUAGE
NLS_TERRITORY
CREDIT_LIMIT
CUST_EMAIL
CUST_ORDERS
CORPORATE_CUSTOMER_TYP
CUSTOMER_ID
CUST_FIRST_NAME
CUST_LAST_NAME
CUST_ADDRESS
PHONE_NUMBERS
NLS_LANGUAGE
NLS_TERRITORY
CREDIT_LIMIT
CUST_EMAIL
CUST_ORDERS
ACCOUNT_MGR_ID
ORDER_TYP
ORDER_ID
ORDER_MODE
CUSTOMER_REF
ORDER_STATUS
ORDER_TOTAL
SALES_REP_ID
ORDER_ITEM_LIST
PRODUCT_INFORMATION_TYP
PRODUCT_ID
PRODUCT_NAME
PRODUCT_DESCRIPTION
CATEGORY_ID
WEIGHT_CLASS
WARRANTY_PERIOD
SUPPLIER_ID
PRODUCT_STATUS
LIST_PRICE
MIN_PRICE
CATALOG_URL
INVENTORY_LIST
ORDER_ITEM_TYP
ORDER_ID
LINE_ITEM_ID
UNIT_PRICE
QUANTITY
PRODUCT_REF
WAREHOUSE_TYP
WAREHOUSE_ID
WAREHOUSE_NAME
LOCATION_ID
INVENTORY_TYP
PRODUCT_ID
WAREHOUSE
QUANTITY_ON_HAND


Queued Shipping (QS)
Order Entry
Order Input
New Orders Queue
(Single Consumer)
Booked Orders Queue
(Multi Consumer)
Eastern Shipping Center
Booked Orders
Shipped Orders
Back Orders
QS_ES
Western Shipping Center
Booked Orders
Shipped Orders
Back Orders
QS_WS
Overseas Shipping Center
Booked Orders
Shipped Orders
Back Orders
QS_OS
Shipped Orders
Billed Orders
Billing Center
QS_CB
Billed Orders
Shipped Orders
Booked Orders
Back Orders
Customer Service
QS_CS


COUNTRIES
PK COUNTRY_ID
COUNTRY_NAME
COUNTRY_SUBREGION
COUNTRY_REGION
CHANNELS
PK CHANNEL_ID
CHANNEL_DESC
CHANNEL_CLASS
TIMES
PK TIME_ID
DAY_NAME
DAY_NUMBER_IN_WEEK
DAY_NUMBER_IN_MONTH
CALENDAR_WEEK_NUMBER
FISCAL_WEEK_NUMBER
WEEK_ENDING_DAY
CALENDAR_MONTH_NUMBER
FISCAL_MONTH_NUMBER
CALENDAR_MONTH_DESC
FISCAL_MONTH_DESC
DAYS_IN_CAL_MONTH
DAYS_IN_FIS_MONTH
END_OF_CAL_MONTH
END_OF_FIS_MONTH
CALENDAR_MONTH_NAME
FISCAL_MONTH_NAME
CALENDAR_QUARTER_DESC
FISCAL_QUARTER_DESC
DAYS_IN_CAL_QUARTER
DAYS_IN_FIS_QUARTER
END_OF_CAL_QUARTER
END_OF_FIS_QUARTER
CALENDAR_QUARTER_NUMBER
FISCAL_QUARTER_NUMBER
CALENDAR_YEAR
FISCAL_YEAR
DAYS_IN_CAL_YEAR
DAYS_IN_FIS_YEAR
END_OF_CAL_YEAR
END_OF_FIS_YEAR
CUSTOMERS
PK CUST_ID
CUST_FIRST_NAME
CUST_LAST_NAME
I1 CUST_GENDER
I3 CUST_YEAR_OF_BIRTH
I2 CUST_MARITAL_STATUS
CUST_STREET_ADDRESS
CUST_POSTAL_CODE
CUST_CITY
CUST_STATE_PROVINCE
FK1 COUNTRY_ID
CUST_MAIN_PHONE_NUMBER
CUST_INCOME_LEVEL
CUST_CREDIT_LIMIT
CUST_EMAIL
PROMOTIONS
PK PROMO_ID
PROMO_NAME
PROMO_SUBCATEGORY
PROMO_CATEGORY
PROMO_COST
PROMO_BEGIN_DATE
PROMO_END_DATE
COSTS
FK1,I1 PROD_ID
FK2,I2 TIME_ID
UNIT_COST
UNIT_PRICE
PRODUCTS
PK PROD_ID
PROD_NAME
PROD_DESC
I3 PROD_SUBCATEGORY
PROD_SUBCAT_DESC
I1 PROD_CATEGORY
PROD_CAT_DESC
PROD_WEIGHT_CLASS
PROD_UNIT_OF_MEASURE
PROD_PACK_SIZE
SUPPLIER_ID
I2 PROD_STATUS
PROD_LIST_PRICE
PROD_MIN_PRICE
SALES
FK2,I3 PROD_ID
FK1,I2 CUST_ID
FK4,I5 TIME_ID
FK3,I4 PROMO_ID
QUANTITY_SOLD
AMOUNT_SOLD
Sales History (SH)
CAL_MONTH_SALES_MV
CALENDAR_MONTH_DESC
DOLLARS

4

Oracle9*i* Sample Schema Scripts

This chapter contains the scripts used to generate the Oracle9*i* sample schemas. Each section corresponds to a separate schema. This chapter contains these sections:

About the Scripts

There are two sets of scripts for each schema:

- One script that resets and creates all objects and data for a particular schema. This script is named `xx_main.sql`, where `xx` is the schema abbreviation. This main script calls all other scripts necessary to build and load the schema.
- One script that erases all objects from a particular schema, called `xx_drop.sql`, where `xx` is the schema abbreviation.

The Oracle9*i* Sample Schemas script directories are located in `$ORACLE_HOME/demo/schema`.

Note: This chapter does not include the scripts that populate the schemas, because they are very lengthy.

Master Script

The master script sets up the overall Sample Schema environment and creates all five schemas.

mksample.sql

```
Rem
Rem $Header: mksample.sql 12-apr-2001.21:20:07 ahunold Exp $
Rem
Rem mksample.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      mksample.sql - creates all 5 Sample Schemas
Rem
Rem    DESCRIPTION
Rem      Demo script that shows how to call the Sample Schema
Rem      creation scripts xx_main.sql
em      Installer variables present
Rem
Rem    NOTES
Rem      Always provide absolute pathnames.
Rem      Script will not run successfully unless the schema
```

```
Rem      creation prerequisites described in "Oracle9i Sample
rem      Schemas" are met.
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem      ahunold   04/13/01 - aaditional parameter (HR,OE,QS)
Rem      ahunold   04/04/01 - Installer variables
Rem      ahunold   04/03/01 - Merged ahunold_mkdir_log
Rem      ahunold   03/28/01 - Created
Rem

SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET ECHO ON

CONNECT system/&password_system

@?/demo/schema/human_resources/hr_main.sql &password_hr demo temp
&password_sys ?/demo/schema/human_resources/

CONNECT system/&password_system

@?/demo/schema/order_entry/oe_main.sql &password_oe demo temp
&password_hr &password_sys ?/demo/schema/order_entry/

CONNECT system/&password_system

@?/demo/schema/product_media/pm_main.sql &password_pm demo temp
&password_oe &password_sys %s_pmPath%

CONNECT system/&password_system

@?/demo/schema/sales_history/sh_main &password_sh demo temp
&password_sys %s_shPath% %s_logPath%

CONNECT system/&password_system

@?/demo/schema/shipping/qs_main.sql &passwords_qs demo temp
&password_system &password_oe &password_sys ?/demo/schema/shipping/
```

Human Resources (HR) Scripts

This section shows the HR schema scripts in alphabetical order.

hr_analz.sql

```
Rem
Rem $Header: hr_analz.sql 12-mar-2001.15:08:47 ahunold Exp $
Rem
Rem hr_analz.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      hr_analz.sql - Gathering statistics for HR schema
Rem
Rem    DESCRIPTION
Rem      Staistics are used by the cost based optimizer to
Rem      choose the best physical access strategy
Rem
Rem    NOTES
Rem      Results can be viewed in columns of DBA_TABLES,
Rem      DBA_TAB_COLUMNS and such
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     03/12/01 - cleanup b3
Rem    ahunold     03/07/01 - Merged ahunold_hr_analz
Rem    ahunold     03/07/01 - Created
Rem

SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET ECHO OFF

EXECUTE dbms_stats.gather_table_stats ('HR','COUNTRIES');
EXECUTE dbms_stats.gather_table_stats ('HR','DEPARTMENTS');
EXECUTE dbms_stats.gather_table_stats ('HR','EMPLOYEES');
EXECUTE dbms_stats.gather_table_stats ('HR','JOBS');
EXECUTE dbms_stats.gather_table_stats ('HR','JOB_HISTORY');
EXECUTE dbms_stats.gather_table_stats ('HR','LOCATIONS');
EXECUTE dbms_stats.gather_table_stats ('HR','REGIONS');
```

hr_code.sql

```
Rem
Rem $Header: hr_code.sql 03-mar-2001.10:05:12 ahunold Exp $
Rem
Rem hr_code.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      hr_code.sql - Create procedural objects for HR schema
Rem
Rem    DESCRIPTION
Rem      Create a statement level trigger on EMPLOYEES
Rem      to allow DML during business hours.
Rem      Create a row level trigger on the EMPLOYEES table,
Rem      after UPDATES on the department_id or job_id columns.
Rem      Create a stored procedure to insert a row into the
Rem      JOB_HISTORY table.  Have the above row level trigger
Rem      row level trigger call this stored procedure.
Rem
Rem    NOTES
Rem
Rem    CREATED by Nancy Greenberg - 06/01/00
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     03/03/01 - HR simplification, REGIONS table
Rem    ahunold     02/20/01 - Created
Rem

SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET ECHO OFF

REM
******************************************************************
******
```

```
REM procedure and statement trigger to allow dmls during business
hours:
CREATE OR REPLACE PROCEDURE secure_dml
IS
BEGIN
  IF TO_CHAR (SYSDATE, 'HH24:MI') NOT BETWEEN '08:00' AND '18:00'
        OR TO_CHAR (SYSDATE, 'DY') IN ('SAT', 'SUN') THEN
RAISE_APPLICATION_ERROR (-20205,
'You may only make changes during normal office hours');
  END IF;
END secure_dml;
/

CREATE OR REPLACE TRIGGER secure_employees
  BEFORE INSERT OR UPDATE OR DELETE ON employees
BEGIN
  secure_dml;
END secure_employees;
/

REM
*****************************************************************
******
REM procedure to add a row to the JOB_HISTORY table and row trigger
REM to call the procedure when data is updated in the job_id or
REM department_id columns in the EMPLOYEES table:

CREATE OR REPLACE PROCEDURE add_job_history
  (  p_emp_id          job_history.employee_id%type
   , p_start_date      job_history.start_date%type
   , p_end_date        job_history.end_date%type
   , p_job_id          job_history.job_id%type
   , p_department_id   job_history.department_id%type
   )
IS
BEGIN
  INSERT INTO job_history (employee_id, start_date, end_date,
                           job_id, department_id)
    VALUES(p_emp_id, p_start_date, p_end_date, p_job_id, p_
department_id);
END add_job_history;
/

CREATE OR REPLACE TRIGGER update_job_history
```

```
  AFTER UPDATE OF job_id, department_id ON employees
  FOR EACH ROW
BEGIN
  add_job_history(:old.employee_id, :old.hire_date, sysdate,
                  :old.job_id, :old.department_id);
END;
/

COMMIT;
```

hr_comnt.sql

```
Rem
Rem $Header: hr_comnt.sql 03-mar-2001.10:05:12 ahunold Exp $
Rem
Rem hr_comnt.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      hr_comnt.sql - Create comments for HR schema
Rem
Rem    DESCRIPTION
Rem
Rem
Rem    CREATED by Nancy Greenberg, Nagavalli Pataballa - 06/01/00
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     02/20/01 - New header
Rem    vpatabal    03/02/01 - Added comments for Regions table
Rem                         - Removed references to currency symbol
Rem                           and currency name columns of countries
Rem                         - Removed comments to DN column of
Rem                           employees and departments.
Rem    - Removed references to sequences

SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET ECHO OFF
```

```
COMMENT ON TABLE regions
IS 'Regions table that contains region numbers and names. Contains 4
rows; references with the Countries table.'

COMMENT ON COLUMN regions.region_id
IS 'Primary key of regions table.'

COMMENT ON COLUMN regions.region_name
IS 'Names of regions. Locations are in the countries of these
regions.'

COMMENT ON TABLE locations
IS 'Locations table that contains specific address of a specific
office,
warehouse, and/or production site of a company. Does not store
addresses /
locations of customers. Contains 23 rows; references with the
departments and countries tables. ';

COMMENT ON COLUMN locations.location_id
IS 'Primary key of locations table';

COMMENT ON COLUMN locations.street_address
IS 'Street address of an office, warehouse, or production site of a
company.
Contains building number and street name';

COMMENT ON COLUMN locations.postal_code
IS 'Postal code of the location of an office, warehouse, or
production site
of a company. ';

COMMENT ON COLUMN locations.city
IS 'A not null column that shows city where an office, warehouse, or
production site of a company is located. ';

COMMENT ON COLUMN locations.state_province
IS 'State or Province where an office, warehouse, or production site
of a
company is located.';

COMMENT ON COLUMN locations.country_id
IS 'Country where an office, warehouse, or production site of a
company is
located. Foreign key to country_id column of the countries table.';
```

```
REM ********************************************

COMMENT ON TABLE departments
IS 'Departments table that shows details of departments where
employees
work. Contains 27 rows; references with locations, employees, and
job_history tables.';

COMMENT ON COLUMN departments.department_id
IS 'Primary key column of departments table.';

COMMENT ON COLUMN departments.department_name
IS 'A not null column that shows name of a department.
Administration,
Marketing, Purchasing, Human Resources, Shipping, IT, Executive,
Public
Relations, Sales, Finance, and Accounting. ';

COMMENT ON COLUMN departments.manager_id
IS 'Manager_id of a department. Foreign key to employee_id column of
employees table. The manager_id column of the employee table
references this column.';

COMMENT ON COLUMN departments.location_id
IS 'Location id where a department is located. Foreign key to
location_id column of locations table.';


REM ********************************************

COMMENT ON TABLE job_history
IS 'Table that stores job history of the employees. If an employee
changes departments within the job or changes jobs within the
department,
new rows get inserted into this table with old job information of
the
employee. Contains a complex primary key: employee_id+start_date.
Contains 25 rows. References with jobs, employees, and departments
tables.';

COMMENT ON COLUMN job_history.employee_id
IS 'A not null column in the complex primary key employee_id+start_
date.
```

```
Foreign key to employee_id column of the employee table';

COMMENT ON COLUMN job_history.start_date
IS 'A not null column in the complex primary key employee_id+start_
date.
Must be less than the end_date of the job_history table. (enforced
by
constraint jhist_date_interval)';

COMMENT ON COLUMN job_history.end_date
IS 'Last day of the employee in this job role. A not null column.
Must be
greater than the start_date of the job_history table.
(enforced by constraint jhist_date_interval)';

COMMENT ON COLUMN job_history.job_id
IS 'Job role in which the employee worked in the past; foreign key
to
job_id column in the jobs table. A not null column.';

COMMENT ON COLUMN job_history.department_id
IS 'Department id in which the employee worked in the past; foreign
key to deparment_id column in the departments table';


REM ********************************************

COMMENT ON TABLE countries
IS 'country table. Contains 25 rows. References with locations
table.';

COMMENT ON COLUMN countries.country_id
IS 'Primary key of countries table.';

COMMENT ON COLUMN countries.country_name
IS 'Country name';

COMMENT ON COLUMN countries.region_id
IS 'Region ID for the country. Foreign key to region_id column in
the departments table.';

REM ********************************************

COMMENT ON TABLE jobs
IS 'jobs table with job titles and salary ranges. Contains 19 rows.
```

```
References with employees and job_history table.';

COMMENT ON COLUMN jobs.job_id
IS 'Primary key of jobs table.';

COMMENT ON COLUMN jobs.job_title
IS 'A not null column that shows job title, e.g. AD_VP, FI_
ACCOUNTANT';

COMMENT ON COLUMN jobs.min_salary
IS 'Minimum salary for a job title.';

COMMENT ON COLUMN jobs.max_salary
IS 'Maximum salary for a job title';

REM ********************************************

COMMENT ON TABLE employees
IS 'employees table. Contains 107 rows. References with departments,
jobs, job_history tables. Contains a self reference.';

COMMENT ON COLUMN employees.employee_id
IS 'Primary key of employees table.';

COMMENT ON COLUMN employees.first_name
IS 'First name of the employee. A not null column.';

COMMENT ON COLUMN employees.last_name
IS 'Last name of the employee. A not null column.';

COMMENT ON COLUMN employees.email
IS 'Email id of the employee';

COMMENT ON COLUMN employees.phone_number
IS 'Phone number of the employee; includes country code and area
code';

COMMENT ON COLUMN employees.hire_date
IS 'Date when the employee started on this job. A not null column.';

COMMENT ON COLUMN employees.job_id
IS 'Current job of the employee; foreign key to job_id column of the
jobs table. A not null column.';

COMMENT ON COLUMN employees.salary
```

```
IS 'Monthly salary of the employee. Must be greater
than zero (enforced by constraint emp_salary_min)';

COMMENT ON COLUMN employees.commission_pct
IS 'Commission percentage of the employee; Only employees in sales
department elgible for commission percentage';

COMMENT ON COLUMN employees.manager_id
IS 'Manager id of the employee; has same domain as manager_id in
departments table. Foreign key to employee_id column of employees
table.
(useful for reflexive joins and CONNECT BY query)';

COMMENT ON COLUMN employees.department_id
IS 'Department id where employee works; foreign key to department_id
column of the departments table';

COMMIT;
```

hr_cre.sql

Rem

```
Rem $Header: hr_cre.sql 03-mar-2001.10:05:13 ahunold Exp $
Rem
Rem hr_cre.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      hr_cre.sql - Create data objects for HR schema
Rem
Rem    DESCRIPTION
Rem      This script creates six tables, associated constraints
Rem      and indexes in the human resources (HR) schema.
Rem
Rem    NOTES
Rem
Rem    CREATED by Nancy Greenberg, Nagavalli Pataballa - 06/01/00
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     09/14/00 - Added emp_details_view
Rem    ahunold     02/20/01 - New header
Rem    vpatabal 03/02/01 - Added regions table, modified regions
```

```
Rem            column in countries table to NUMBER.
Rem            Added foreign key from countries table
Rem            to regions table on region_id.
Rem                    Removed currency name, currency symbol
Rem            columns from the countries table.
Rem                  Removed dn columns from employees and
Rem            departments tables.
Rem            Added sequences.
Rem            Removed not null constraint from
Rem             salary column of the employees table.

SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET ECHO OFF

REM
************************************************************************
REM Create the REGIONS table to hold region information for
locations
REM HR.LOCATIONS table has a foreign key to this table.

Prompt ******  Creating REGIONS table ....

CREATE TABLE regions
    ( region_id      NUMBER
       CONSTRAINT  region_id_nn NOT NULL
    , region_name    VARCHAR2(25)
    );

CREATE UNIQUE INDEX reg_id_pk
ON regions (region_id);

ALTER TABLE regions
ADD ( CONSTRAINT reg_id_pk
        PRIMARY KEY (region_id)
    ) ;

REM
************************************************************************
REM Create the COUNTRIES table to hold country information for
customers
```

```
REM and company locations.
REM OE.CUSTOMERS table and HR.LOCATIONS have a foreign key to this
table.

Prompt ******  Creating COUNTRIES table ....

CREATE TABLE countries
    ( country_id      CHAR(2)
       CONSTRAINT  country_id_nn NOT NULL
    , country_name    VARCHAR2(40)
    , region_id       NUMBER
    , CONSTRAINT     country_c_id_pk
             PRIMARY KEY (country_id)
    )
    ORGANIZATION INDEX;

ALTER TABLE countries
ADD ( CONSTRAINT countr_reg_fk
         FOREIGN KEY (region_id)
            REFERENCES regions(region_id)
    ) ;

REM
*****************************************************************
REM Create the LOCATIONS table to hold address information for
company departments.
REM HR.DEPARTMENTS has a foreign key to this table.

Prompt ******  Creating LOCATIONS table ....

CREATE TABLE locations
    ( location_id    NUMBER(4)
    , street_address VARCHAR2(40)
    , postal_code    VARCHAR2(12)
    , city       VARCHAR2(30)
CONSTRAINT     loc_city_nn  NOT NULL
    , state_province VARCHAR2(25)
    , country_id     CHAR(2)
    ) ;

CREATE UNIQUE INDEX loc_id_pk
ON locations (location_id) ;

ALTER TABLE locations
ADD ( CONSTRAINT loc_id_pk
```

```
        PRIMARY KEY (location_id)
    , CONSTRAINT loc_c_id_fk
        FOREIGN KEY (country_id)
          REFERENCES countries(country_id)
    ) ;

Rem Useful for any subsequent addition of rows to locations table
Rem Starts with 3300

CREATE SEQUENCE locations_seq
 START WITH     3300
 INCREMENT BY   100
 MAXVALUE       9900
 NOCACHE
 NOCYCLE;

REM
*********************************************************************
REM Create the DEPARTMENTS table to hold company department
information.
REM HR.EMPLOYEES and HR.JOB_HISTORY have a foreign key to this
table.

Prompt ******  Creating DEPARTMENTS table ....

CREATE TABLE departments
    ( department_id    NUMBER(4)
    , department_name  VARCHAR2(30)
CONSTRAINT  dept_name_nn  NOT NULL
    , manager_id       NUMBER(6)
    , location_id      NUMBER(4)
    ) ;

CREATE UNIQUE INDEX dept_id_pk
ON departments (department_id) ;

ALTER TABLE departments
ADD ( CONSTRAINT dept_id_pk
        PRIMARY KEY (department_id)
    , CONSTRAINT dept_loc_fk
        FOREIGN KEY (location_id)
          REFERENCES locations (location_id)
     ) ;

Rem Useful for any subsequent addition of rows to departments table
```

```
Rem Starts with 280

CREATE SEQUENCE departments_seq
 START WITH     280
 INCREMENT BY   10
 MAXVALUE       9990
 NOCACHE
 NOCYCLE;

REM
*************************************************************************
REM Create the JOBS table to hold the different names of job roles
within the company.
REM HR.EMPLOYEES has a foreign key to this table.

Prompt ******  Creating JOBS table ....

CREATE TABLE jobs
    ( job_id         VARCHAR2(10)
    , job_title      VARCHAR2(35)
CONSTRAINT     job_title_nn  NOT NULL
    , min_salary     NUMBER(6)
    , max_salary     NUMBER(6)
    ) ;

CREATE UNIQUE INDEX job_id_pk
ON jobs (job_id) ;

ALTER TABLE jobs
ADD ( CONSTRAINT job_id_pk
       PRIMARY KEY(job_id)
    ) ;

REM
*************************************************************************
REM Create the EMPLOYEES table to hold the employee personnel
REM information for the company.
REM HR.EMPLOYEES has a self referencing foreign key to this table.

Prompt ******  Creating EMPLOYEES table ....

CREATE TABLE employees
    ( employee_id    NUMBER(6)
    , first_name     VARCHAR2(20)
    , last_name      VARCHAR2(25)
```

```
 CONSTRAINT     emp_last_name_nn  NOT NULL
    , email          VARCHAR2(25)
CONSTRAINT     emp_email_nn  NOT NULL
    , phone_number   VARCHAR2(20)
    , hire_date      DATE
CONSTRAINT     emp_hire_date_nn  NOT NULL
    , job_id         VARCHAR2(10)
CONSTRAINT     emp_job_nn  NOT NULL
    , salary         NUMBER(8,2)
    , commission_pct NUMBER(2,2)
    , manager_id     NUMBER(6)
    , department_id  NUMBER(4)
    , CONSTRAINT     emp_salary_min
                     CHECK (salary > 0)
    , CONSTRAINT     emp_email_uk
                     UNIQUE (email)
    ) ;

CREATE UNIQUE INDEX emp_emp_id_pk
ON employees (employee_id) ;


ALTER TABLE employees
ADD ( CONSTRAINT     emp_emp_id_pk
                     PRIMARY KEY (employee_id)
    , CONSTRAINT     emp_dept_fk
                     FOREIGN KEY (department_id)
                      REFERENCES departments
    , CONSTRAINT     emp_job_fk
                     FOREIGN KEY (job_id)
                      REFERENCES jobs (job_id)
    , CONSTRAINT     emp_manager_fk
                     FOREIGN KEY (manager_id)
                      REFERENCES employees
    ) ;

ALTER TABLE departments
ADD ( CONSTRAINT dept_mgr_fk
       FOREIGN KEY (manager_id)
        REFERENCES employees (employee_id)
    ) ;


Rem Useful for any subsequent addition of rows to employees table
Rem Starts with 207
```

```
CREATE SEQUENCE employees_seq
 START WITH     207
 INCREMENT BY   1
 NOCACHE
 NOCYCLE;

REM
*****************************************************************
REM Create the JOB_HISTORY table to hold the history of jobs that
REM employees have held in the past.
REM HR.JOBS, HR_DEPARTMENTS, and HR.EMPLOYEES have a foreign key to
this table.

Prompt ******  Creating JOB_HISTORY table ....

CREATE TABLE job_history
    ( employee_id   NUMBER(6)
 CONSTRAINT    jhist_employee_nn  NOT NULL
    , start_date    DATE
CONSTRAINT    jhist_start_date_nn  NOT NULL
    , end_date      DATE
CONSTRAINT    jhist_end_date_nn  NOT NULL
    , job_id        VARCHAR2(10)
CONSTRAINT    jhist_job_nn  NOT NULL
    , department_id NUMBER(4)
    , CONSTRAINT    jhist_date_interval
                    CHECK (end_date > start_date)
    ) ;

CREATE UNIQUE INDEX jhist_emp_id_st_date_pk
ON job_history (employee_id, start_date) ;

ALTER TABLE job_history
ADD ( CONSTRAINT jhist_emp_id_st_date_pk
      PRIMARY KEY (employee_id, start_date)
    , CONSTRAINT     jhist_job_fk
                     FOREIGN KEY (job_id)
                     REFERENCES jobs
    , CONSTRAINT     jhist_emp_fk
                     FOREIGN KEY (employee_id)
                     REFERENCES employees
    , CONSTRAINT     jhist_dept_fk
                     FOREIGN KEY (department_id)
```

```
                        REFERENCES departments
    ) ;

REM
*****************************************************************
REM Create the EMP_DETAILS_VIEW that joins the employees, jobs,
REM departments, jobs, countries, and locations table to provide
details
REM about employees.

Prompt ******  Creating EMP_DETAILS_VIEW view ...

CREATE OR REPLACE VIEW emp_details_view
  (employee_id,
   job_id,
   manager_id,
   department_id,
   location_id,
   country_id,
   first_name,
   last_name,
   salary,
   commission_pct,
   department_name,
   job_title,
   city,
   state_province,
   country_name,
   region_name)
AS SELECT
  e.employee_id,
  e.job_id,
  e.manager_id,
  e.department_id,
  d.location_id,
  l.country_id,
  e.first_name,
  e.last_name,
  e.salary,
  e.commission_pct,
  d.department_name,
  j.job_title,
  l.city,
  l.state_province,
  c.country_name,
```

```
  r.region_name
FROM
  employees e,
  departments d,
  jobs j,
  locations l,
  countries c,
  regions r
WHERE e.department_id = d.department_id
  AND d.location_id = l.location_id
  AND l.country_id = c.country_id
  AND c.region_id = r.region_id
  AND j.job_id = e.job_id
WITH READ ONLY;

COMMIT;
```

hr_dn_c.sql

```
Rem
Rem $Header: hr_dn_c.sql 03-mar-2001.10:05:13 ahunold Exp $
Rem
Rem hr_dn_c.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      hr_dn_c.sql - Add DN column to HR.EMPLOYEES and DEPARTMENTS
Rem
Rem    DESCRIPTION
Rem      the DN (distinguished Name) column is used by OID.
Rem      This script adds the column to the HR schema. It is not
Rem      part of the default set of Sample Schemas, but shipped
Rem      as an extension script for demo purposes.
Rem
Rem    NOTES
Rem
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     02/20/01 - Created
Rem    vpatabal    03/02/01 - Modified dn for employee 178
Rem    ahunold     03/03/01 - employee 104, triggers

SET FEEDBACK 1
```

```
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET ECHO ON

DROP TRIGGER secure_employees;
DROP TRIGGER update_job_history;

ALTER TABLE departments
 ADD dn VARCHAR2(300);

COMMENT ON COLUMN departments.dn IS
'Distinguished name for each deparment.
e.g: "ou=Purchasing, o=IMC, c=US"';

ALTER TABLE employees
 ADD dn VARCHAR2(300);

COMMENT ON COLUMN employees.dn IS
'Distinguished name of the employee.
e.g. "cn=Lisa Ozer, ou=Sales, o=IMC, c=us"';

UPDATE departments SET
 dn='"ou=Administration, o=IMC, c=US"'
 WHERE department_id=10;

UPDATE departments SET
 dn='"ou=Mktg, o=IMC, c=US"'
 WHERE department_id=20;

UPDATE departments SET
 dn='"ou=Purchasing, o=IMC, c=US"'
 WHERE department_id=30;

UPDATE departments SET
 dn='"ou=HR, o=IMC, c=US"'
 WHERE department_id=40;

UPDATE departments SET
 dn='"ou=Shipping, o=IMC, c=US"'
 WHERE department_id=50;

UPDATE departments SET
```

```
 dn='"ou=IT, o=IMC, c=US"'
 WHERE department_id=60;

UPDATE departments SET
 dn='"ou=PR, o=IMC, c=US"'
 WHERE department_id=70;

UPDATE departments SET
 dn='"ou=Sales, o=IMC, c=US"'
 WHERE department_id=80;

UPDATE departments SET
 dn='"ou=Executive, o=IMC, c=US"'
 WHERE department_id=90;

UPDATE departments SET
 dn='"ou=Finance, ou=Fin-Accounting, o=IMC, c=US"'
 WHERE department_id=100;

UPDATE departments SET
 dn='"ou=Accounting, ou=Fin-Accounting, o=IMC, c=US"'
 WHERE department_id=110;

UPDATE departments SET
 dn='"ou=Treasury, ou=Fin-Accounting, ou=Europe, o=IMC, c=US"'
 WHERE department_id=120;

UPDATE departments SET
 dn='"ou=Corporate Tax, ou=Fin-Accounting, o=IMC, c=US"'
 WHERE department_id=130;

UPDATE departments SET
 dn='"ou=Control and Credit, ou=Fin-Accounting, o=IMC, c=US"'
 WHERE department_id=140;

UPDATE departments SET
 dn='"ou=Shareholder Services, ou=Fin-Accounting, ou=Europe, o=IMC,
c=US"'
 WHERE department_id=150;

UPDATE departments SET
 dn='"ou=Benefits, o=IMC, c=US"'
 WHERE department_id=160;

UPDATE departments SET
```

```
 dn='"ou=Manufacturing, o=IMC, c=US"'
 WHERE department_id=170;

UPDATE departments SET
 dn='"ou=Construction, ou=Manufacturing, o=IMC, c=US"'
 WHERE department_id=180;

UPDATE departments SET
 dn='"ou=Contracting, ou = Manufacturing, o=IMC, c=US"'
 WHERE department_id=190;

UPDATE departments SET
 dn='"ou=Operations, ou=Manufacturing, ou=Americas, o=IMC, c=US"'
 WHERE department_id=200;

UPDATE departments SET
 dn='"ou=Field Support, ou=IT, ou=Americas, o=IMC, c=US"'
 WHERE department_id=210;

UPDATE departments SET
 dn='"ou=Network Operations Center, ou=IT, ou=Europe, o=IMC, c=US"'
 WHERE department_id=220;

UPDATE departments SET
 dn='"ou=Help Desk, ou=IT, ou=Europe, o=IMC, c=US"'
 WHERE department_id=230;

UPDATE departments SET
 dn='"ou=Government, ou=Sales, ou=Americas, o=IMC, c=US"'
 WHERE department_id=240;

UPDATE departments SET
 dn='"ou=Retail, ou=Sales, ou=Europe, o=IMC, c=US"'
 WHERE department_id=250;

UPDATE departments SET
 dn='"ou=Recruiting, ou=HR, ou=Europe, o=IMC, c=US"'
 WHERE department_id=260;

UPDATE departments SET
 dn='"ou=Payroll, ou=HR, ou=Europe, o=IMC, c=US"'
 WHERE department_id=270;

UPDATE employees SET
 dn='"cn=Steven King, ou=Executive, o=IMC, c=us"'
```

```
 WHERE employee_id=100;

UPDATE employees SET
 dn='"cn=Neena Kochhar, ou=Executive, o=IMC, c=us"'
 WHERE employee_id=101;

UPDATE employees SET
 dn='"cn=Lex De Haan, ou=Executive, o=IMC, c=us"'
 WHERE employee_id=102;

UPDATE employees SET
 dn='"cn=Alexander Hunold, ou=IT, o=IMC, c=us"'
 WHERE employee_id=103;

UPDATE employees SET
 dn='"cn=Bruce Ernst, ou=IT, o=IMC, c=us"'
 WHERE employee_id=104;

UPDATE employees SET
 dn='"cn=David Austin, ou=IT, o=IMC, c=us"'
 WHERE employee_id=105;

UPDATE employees SET
 dn='"cn=Valli Pataballa, ou=IT, o=IMC, c=us"'
 WHERE employee_id=106;

UPDATE employees SET
 dn='"cn=Diana Lorentz, ou=IT, o=IMC, c=us"'
 WHERE employee_id=107;

UPDATE employees SET
 dn='"cn=Nancy Greenberg, ou=Accounting, o=IMC, c=us"'
 WHERE employee_id=108;

UPDATE employees SET
 dn='"cn=Daniel Faviet, ou=Accounting, o=IMC, c=us"'
 WHERE employee_id=109;

UPDATE employees SET
 dn='"cn=John Chen, ou=Accounting, o=IMC, c=us"'
 WHERE employee_id=110;

UPDATE employees SET
 dn='"cn=Ismael Sciarra, ou=Accounting, o=IMC, c=us"'
 WHERE employee_id=111;
```

```
UPDATE employees SET
 dn='"cn=Jose Manuel Urman, ou=Accounting, o=IMC, c=us"'
 WHERE employee_id=112;

UPDATE employees SET
 dn='"cn=Luis Popp, ou=Accounting, o=IMC, c=us"'
 WHERE employee_id=113;

UPDATE employees SET
 dn='"cn=Den Raphaely, ou=Purchasing, o=IMC, c=us"'
 WHERE employee_id=114;

UPDATE employees SET
 dn='"cn=Alexander Khoo, ou=Purchasing, o=IMC, c=us"'
 WHERE employee_id=115;

UPDATE employees SET
 dn='"cn=Shelli Baida, ou=Purchasing, o=IMC, c=us"'
 WHERE employee_id=116;

UPDATE employees SET
 dn='"cn=Sigal Tobias, ou=Purchasing, o=IMC, c=us"'
 WHERE employee_id=117;

UPDATE employees SET
 dn='"cn=Guy Himuro, ou=Purchasing, o=IMC, c=us"'
 WHERE employee_id=118;

UPDATE employees SET
 dn='"cn=Karen Colmenares, ou=Purchasing, o=IMC, c=us"'
 WHERE employee_id=119;

UPDATE employees SET
 dn='"cn=Matthew Weiss, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=120;

UPDATE employees SET
 dn='"cn=Adam Fripp, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=121;

UPDATE employees SET
 dn='"cn=Payam Kaufling, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=122;
```

```
UPDATE employees SET
 dn='"cn=Shanta Vollman, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=123;

UPDATE employees SET
 dn='"cn=Kevin Mourgos, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=124;

UPDATE employees SET
 dn='"cn=Julia Nayer, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=125;

UPDATE employees SET
 dn='"cn=Irene Mikkilineni, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=126;

UPDATE employees SET
 dn='"cn=James Landry, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=127;

UPDATE employees SET
 dn='"cn=Steven Markle, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=128;

UPDATE employees SET
 dn='"cn=Laura Bissot, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=129;

UPDATE employees SET
 dn='"cn=Mozhe Atkinson, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=130;

UPDATE employees SET
 dn='"cn=James Marlow, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=131;

UPDATE employees SET
 dn='"cn=TJ Olson, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=132;

UPDATE employees SET
 dn='"cn=Jason Mallin, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=133;

UPDATE employees SET
```

```
 dn='"cn=Michael Rogers, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=134;

UPDATE employees SET
 dn='"cn=Ki Gee, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=135;

UPDATE employees SET
 dn='"cn=Hazel Philtanker, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=136;

UPDATE employees SET
 dn='"cn=Renske Ladwig, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=137;

UPDATE employees SET
 dn='"cn=Stephen Stiles, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=138;

UPDATE employees SET
 dn='"cn=John Seo, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=139;

UPDATE employees SET
 dn='"cn=Joshua Patel, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=140;

UPDATE employees SET
 dn='"cn=Trenna Rajs, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=141;

UPDATE employees SET
 dn='"cn=Curtis Davies, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=142;

UPDATE employees SET
 dn='"cn=Randall Matos, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=143;

UPDATE employees SET
 dn='"cn=Peter Vargas, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=144;

UPDATE employees SET
 dn='"cn=John Russell, ou=Sales, o=IMC, c=us"'
```

```
 WHERE employee_id=145;

UPDATE employees SET
 dn='"cn=Karen Partners, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=146;

UPDATE employees SET
 dn='"cn=Alberto Errazuriz, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=147;

UPDATE employees SET
 dn='"cn=Gerald Cambrault, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=148;

UPDATE employees SET
 dn='"cn=Eleni Zlotkey, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=149;

UPDATE employees SET
 dn='"cn=Peter Tucker, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=150;

UPDATE employees SET
 dn='"cn=David Bernstein, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=151;

UPDATE employees SET
 dn='"cn=Peter Hall, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=152;

UPDATE employees SET
 dn='"cn=Christopher Olsen, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=153;

UPDATE employees SET
 dn='"cn=Nanette Cambrault, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=154;

UPDATE employees SET
 dn='"cn=Oliver Tuvault, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=155;

UPDATE employees SET
 dn='"cn=Janette King, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=156;
```

```
UPDATE employees SET
 dn='"cn=Patrick Sully, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=157;

UPDATE employees SET
 dn='"cn=Allan McEwen, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=158;

UPDATE employees SET
 dn='"cn=Lindsey Smith, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=159;

UPDATE employees SET
 dn='"cn=Louise Doran, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=160;

UPDATE employees SET
 dn='"cn=Sarath Sewall, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=161;

UPDATE employees SET
 dn='"cn=Clara Vishney, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=162;

UPDATE employees SET
 dn='"cn=Danielle Greene, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=163;

UPDATE employees SET
 dn='"cn=Mattea Marvins, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=164;

UPDATE employees SET
 dn='"cn=David Lee, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=165;

UPDATE employees SET
 dn='"cn=Sundar Ande, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=166;

UPDATE employees SET
 dn='"cn=Amit Banda, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=167;
```

```
UPDATE employees SET
 dn='"cn=Lisa Ozer, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=168;

UPDATE employees SET
 dn='"cn=Harrison Bloom, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=169;

UPDATE employees SET
 dn='"cn=Taylor Fox, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=170;

UPDATE employees SET
 dn='"cn=William Smith, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=171;

UPDATE employees SET
 dn='"cn=Elizabeth Bates, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=172;

UPDATE employees SET
 dn='"cn=Sundita Kumar, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=173;

UPDATE employees SET
 dn='"cn=Ellen Abel, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=174;

UPDATE employees SET
 dn='"cn=Alyssa Hutton, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=175;

UPDATE employees SET
 dn='"cn=Jonathod Taylor, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=176;

UPDATE employees SET
 dn='"cn=Jack Livingston, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=177;

UPDATE employees SET
 dn='"cn=Kimberely Grant, ou= , o=IMC, c=us"'
 WHERE employee_id=178;

UPDATE employees SET
```

```
 dn='"cn=Charles Johnson, ou=Sales, o=IMC, c=us"'
 WHERE employee_id=179;

UPDATE employees SET
 dn='"cn=Winston Taylor, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=180;

UPDATE employees SET
 dn='"cn=Jean Fleaur, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=181;

UPDATE employees SET
 dn='"cn=Martha Sullivan, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=182;

UPDATE employees SET
 dn='"cn=Girard Geoni, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=183;

UPDATE employees SET
 dn='"cn=Nandita Sarchand, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=184;

UPDATE employees SET
 dn='"cn=Alexis Bull, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=185;

UPDATE employees SET
 dn='"cn=Julia Dellinger, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=186;

UPDATE employees SET
 dn='"cn=Anthony Cabrio, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=187;

UPDATE employees SET
 dn='"cn=Kelly Chung, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=188;

UPDATE employees SET
 dn='"cn=Jennifer Dilly, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=189;

UPDATE employees SET
 dn='"cn=Timothy Gates, ou=Shipping, o=IMC, c=us"'
```

```
 WHERE employee_id=190;

UPDATE employees SET
 dn='"cn=Randall Perkins, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=191;

UPDATE employees SET
 dn='"cn=Sarah Bell, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=192;

UPDATE employees SET
 dn='"cn=Britney Everett, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=193;

UPDATE employees SET
 dn='"cn=Samuel McCain, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=194;

UPDATE employees SET
 dn='"cn=Vance Jones, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=195;

UPDATE employees SET
 dn='"cn=Alana Walsh, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=196;

UPDATE employees SET
 dn='"cn=Kevin Feeney, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=197;

UPDATE employees SET
 dn='"cn=Donald OConnell, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=198;

UPDATE employees SET
 dn='"cn=Douglas Grant, ou=Shipping, o=IMC, c=us"'
 WHERE employee_id=199;

UPDATE employees SET
 dn='"cn=Jennifer Whalen, ou=Administration, o=IMC, c=us"'
 WHERE employee_id=200;

UPDATE employees SET
 dn='"cn=Michael Hartstein, ou=Mktg, o=IMC, c=us"'
 WHERE employee_id=201;
```

```
UPDATE employees SET
 dn='"cn=Brajesh Goyal, ou=Mktg, o=IMC, c=us"'
 WHERE employee_id=202;

UPDATE employees SET
 dn='"cn=Susan Marvis, ou=HR, o=IMC, c=us"'
 WHERE employee_id=203;

UPDATE employees SET
 dn='"cn=Hermann Baer, ou=PR, o=IMC, c=us"'
 WHERE employee_id=204;

UPDATE employees SET
 dn='"cn=Shelley Higgens, ou=Accounting, o=IMC, c=us"'
 WHERE employee_id=205;

UPDATE employees SET
 dn='"cn=William Gietz, ou=Accounting, o=IMC, c=us"'
 WHERE employee_id=206;

REM
**********************************************************************
******

REM procedure and statement trigger to allow dmls during business
hours:
CREATE OR REPLACE PROCEDURE secure_dml
IS
BEGIN
  IF TO_CHAR (SYSDATE, 'HH24:MI') NOT BETWEEN '08:00' AND '18:00'
        OR TO_CHAR (SYSDATE, 'DY') IN ('SAT', 'SUN') THEN
RAISE_APPLICATION_ERROR (-20205,
'You may only make changes during normal office hours');
  END IF;
END secure_dml;
/

CREATE OR REPLACE TRIGGER secure_employees
  BEFORE INSERT OR UPDATE OR DELETE ON employees
BEGIN
  secure_dml;
END secure_employees;
/
```

```
Rem Recreating the triggers dropped above

REM
*****************************************************************
******
REM procedure to add a row to the JOB_HISTORY table and row trigger
REM to call the procedure when data is updated in the job_id or
REM department_id columns in the EMPLOYEES table:

CREATE OR REPLACE PROCEDURE add_job_history
  (  p_emp_id          job_history.employee_id%type
   , p_start_date      job_history.start_date%type
   , p_end_date        job_history.end_date%type
   , p_job_id          job_history.job_id%type
   , p_department_id   job_history.department_id%type
   )
IS
BEGIN
  INSERT INTO job_history (employee_id, start_date, end_date,
                           job_id, department_id)
    VALUES(p_emp_id, p_start_date, p_end_date, p_job_id, p_
department_id);
END add_job_history;
/

CREATE OR REPLACE TRIGGER update_job_history
  AFTER UPDATE OF job_id, department_id ON employees
  FOR EACH ROW
BEGIN
  add_job_history(:old.employee_id, :old.hire_date, sysdate,
                  :old.job_id, :old.department_id);
END;
/

COMMIT;
```

hr_dn_d.sql

```
Rem
Rem $Header: hr_dn_d.sql 03-mar-2001.10:05:14 ahunold Exp $
Rem
Rem hr_dn_d.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
```

```
Rem
Rem    NAME
Rem      hr_dn_d.sql - Drop DN column from EMPLOYEES and DEPARTMENTS
Rem
Rem    DESCRIPTION
Rem      the DN (distinguished Name) column is used by OID.
Rem      This script drops the column from the HR schema.
Rem
Rem    NOTES
Rem      Use this to undo changes made by hr_dn_c.sql
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     03/03/01 - HR simplification, REGIONS table
Rem    ahunold     02/20/01 - Merged ahunold_american
Rem    ahunold     02/20/01 - Created
Rem

SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET ECHO ON

ALTER TABLE departments
 DROP COLUMN dn ;

ALTER TABLE employees
 DROP COLUMN dn ;
```

hr_drop.sql

```
Rem
Rem $Header: hr_drop.sql 03-mar-2001.10:05:14 ahunold Exp $
Rem
Rem hr_drop.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      hr_drop.sql - Drop objects from HR schema
Rem
Rem    DESCRIPTION
```

```
Rem
Rem
Rem    NOTES
Rem
Rem    CREATED by Nancy Greenberg - 06/01/00
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     02/20/01 - New header, non-table objects
Rem    vpatabal    03/02/01 - DROP TABLE region

SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET ECHO OFF

CONNECT hr/&password_HR

DROP PROCEDURE add_job_history;
DROP PROCEDURE secure_dml;

DROP VIEW emp_details_view;

DROP SEQUENCE departments_seq;
DROP SEQUENCE employees_seq;
DROP SEQUENCE locations_seq;

DROP TABLE regions     CASCADE CONSTRAINTS;
DROP TABLE departments CASCADE CONSTRAINTS;
DROP TABLE locations   CASCADE CONSTRAINTS;
DROP TABLE jobs        CASCADE CONSTRAINTS;
DROP TABLE job_history CASCADE CONSTRAINTS;
DROP TABLE employees   CASCADE CONSTRAINTS;
DROP TABLE countries   CASCADE CONSTRAINTS;

COMMIT;
```

hr_idx.sql

```
Rem
Rem $Header: hr_idx.sql 03-mar-2001.10:05:15 ahunold Exp $
Rem
```

```
Rem hr_idx.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      hr_idx.sql - Create indexes for HR schema
Rem
Rem    DESCRIPTION
Rem
Rem
Rem    NOTES
Rem
Rem
Rem    CREATED by Nancy Greenberg - 06/01/00
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     02/20/01 - New header
Rem    vpatabal    03/02/01 - Removed DROP INDEX statements

SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET ECHO OFF

CREATE INDEX emp_department_ix
       ON employees (department_id);

CREATE INDEX emp_job_ix
       ON employees (job_id);

CREATE INDEX emp_manager_ix
       ON employees (manager_id);

CREATE INDEX emp_name_ix
       ON employees (last_name, first_name);

CREATE INDEX dept_location_ix
       ON departments (location_id);

CREATE INDEX jhist_job_ix
       ON job_history (job_id);

CREATE INDEX jhist_employee_ix
```

```
       ON job_history (employee_id);

CREATE INDEX jhist_department_ix
       ON job_history (department_id);

CREATE INDEX loc_city_ix
       ON locations (city);

CREATE INDEX loc_state_province_ix
       ON locations (state_province);

CREATE INDEX loc_country_ix
       ON locations (country_id);

COMMIT;
```

hr_main.sql

```
rem
rem Header: hr_main.sql 09-jan-01
rem
rem Copyright (c) 2001 Oracle Corporation.  All rights reserved.
rem
rem Owner  : ahunold
rem
rem NAME
rem   hr_main.sql - Main script for HR schema
rem
rem DESCRIPTON
rem   HR (Human Resources) is the smallest and most simple one
rem   of the Sample Schemas
rem
rem NOTES
rem   Run as SYS or SYSTEM
rem
rem MODIFIED   (MM/DD/YY)
rem   ahunold   04/13/01 - parameter 5, notes, spool
rem   ahunold   03/29/01 - spool
rem   ahunold   03/12/01 - prompts
rem   ahunold   03/07/01 - hr_analz.sql
rem   ahunold   03/03/01 - HR simplification, REGIONS table
rem   ngreenbe  06/01/00 - created
```

```
SET ECHO ON

ALTER SESSION SET NLS_LANGUAGE=American;

PROMPT
PROMPT specify password for HR as parameter 1:
define pass     = &1
PROMPT
PROMPT specify default tablespeace for HR as parameter 2:
define tbs      = &2
PROMPT
PROMPT specify temporary tablespace for HR as parameter 3:
define ttbs     = &3
PROMPT
PROMPT specify password for SYS as parameter 4:
define pass_sys = &4
PROMPT
PROMPT specify log path as parameter 5:
define log_path = &5
PROMPT

-- The first dot in the spool command below is
-- the SQL*Plus concatenation character

spool &log_path.hr_main.log

REM =======================================================
REM cleanup section
REM =======================================================

DROP USER hr CASCADE;

REM =======================================================
REM create user
REM three separate commands, so the create user command
REM will succeed regardless of the existence of the
REM DEMO and TEMP tablespaces
REM =======================================================

CREATE USER hr IDENTIFIED BY &pass;

ALTER USER hr DEFAULT TABLESPACE &tbs
              QUOTA UNLIMITED ON &tbs;

ALTER USER hr TEMPORARY TABLESPACE &ttbs;
```

```
GRANT create session
     , create table
     , create procedure
     , create sequence
     , create trigger
     , create view
     , create synonym
     , alter session
TO hr;

REM =======================================================
REM grants from sys schema
REM =======================================================

CONNECT sys/&pass_sys AS SYSDBA;
GRANT execute ON sys.dbms_stats TO hr;

REM =======================================================
REM create hr schema objects
REM =======================================================

CONNECT hr/&pass
@@hr_cre       -- create tables, sequences and constraints
@@hr_popul     -- populate tables
@@hr_idx       -- create indexes
@@hr_code      -- create procedural objects
@@hr_comnt     -- add comments to tables and columns
@@hr_analz     -- gather schema statistics
spool off
```

Order Entry (OE) Scripts

This section shows the OE schema scripts in alphabetical order.

Note: The scripts starting with "oc" deal with the object relational part of the OE schema, and are called from within the `oe_main.sql` script.

---

oc_comnt.sql

```
Rem
```

```
Rem $Header: oc_comnt.sql 05-mar-2001.15:51:26 ahunold Exp $
Rem
Rem oc_comnt.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      oc_comnt.sql - Comments for OC subschema
Rem
Rem    DESCRIPTION
Rem      The OC subschema (Online Catalog) exhibits objects and
rem object inheritance.
Rem
Rem    NOTES
Rem      Comments are added for tables, wherever possible.
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     03/05/01 - substituteable object table (WIP)
Rem    ahunold     01/29/01 - OC changes, including OC_COMNT.SQL
Rem    ahunold     01/29/01 - Created
Rem
```

oc_cre.sql

```
rem
rem Header: oc_cre.sql 09-jan-01
rem
rem Copyright (c) 2001 Oracle Corporation.  All rights reserved.
rem
rem Owner  : ahunold
rem
rem NAME
rem   oc_cre.sql - create OC subschema of OE Common Schmema
rem
rem DESCRIPTON
rem   Creates database objects. The script assumes that the OE
schema is present.
rem
rem MODIFIED   (MM/DD/YY)
rem   ahunold   04/10/01 - object methods
rem   ahunold   04/12/01 - change case, nested tables named
rem   gxlee     03/05/01 - substituteable object table
rem   ahunold   01/29/01 - typo
```

```
rem   ahunold   01/24/01 - Eliminate extra lines from last merge
rem   ahunold   01/09/01 - checkin ADE

--
==========================================================================
==
-- Type definitions
--
==========================================================================
==

CREATE TYPE warehouse_typ AS OBJECT
    ( warehouse_id       NUMBER(3)
    , warehouse_name     VARCHAR2(35)
    , location_id        NUMBER(4)
    ) ;
/
CREATE TYPE inventory_typ AS OBJECT
    ( product_id          NUMBER(6)
    , warehouse           warehouse_typ
    , quantity_on_hand    NUMBER(8)
    ) ;
/
CREATE TYPE inventory_list_typ AS TABLE OF inventory_typ;
/
CREATE TYPE product_information_typ AS OBJECT
    ( product_id           NUMBER(6)
    , product_name         VARCHAR2(50)
    , product_description  VARCHAR2(2000)
    , category_id          NUMBER(2)
    , weight_class         NUMBER(1)
    , warranty_period      INTERVAL YEAR(2) TO MONTH
    , supplier_id          NUMBER(6)
    , product_status       VARCHAR2(20)
    , list_price           NUMBER(8,2)
    , min_price            NUMBER(8,2)
    , catalog_url          VARCHAR2(50)
    , inventory_list       inventory_list_typ
    ) ;
/
CREATE TYPE order_item_typ AS OBJECT
    ( order_id           NUMBER(12)
    , line_item_id       NUMBER(3)
    , unit_price         NUMBER(8,2)
    , quantity           NUMBER(8)
```

```
    , product_ref  REF   product_information_typ
    ) ;
/
CREATE TYPE order_item_list_typ AS TABLE OF order_item_typ;
/
CREATE TYPE customer_typ;
/
CREATE TYPE order_typ AS OBJECT
    ( order_id           NUMBER(12)
    , order_mode         VARCHAR2(8)
    , customer_ref  REF  customer_typ
    , order_status       NUMBER(2)
    , order_total        NUMBER(8,2)
    , sales_rep_id       NUMBER(6)
    , order_item_list    order_item_list_typ
    ) ;
/
CREATE TYPE order_list_typ AS TABLE OF order_typ;
/
CREATE OR REPLACE TYPE customer_typ AS OBJECT
    ( customer_id        NUMBER(6)
    , cust_first_name    VARCHAR2(20)
    , cust_last_name     VARCHAR2(20)
    , cust_address       cust_address_typ
    , phone_numbers      phone_list_typ
    , nls_language       VARCHAR2(3)
    , nls_territory      VARCHAR2(30)
    , credit_limit       NUMBER(9,2)
    , cust_email         VARCHAR2(30)
    , cust_orders        order_list_typ
    )
NOT FINAL;
/
CREATE TYPE category_typ AS OBJECT
    ( category_name           VARCHAR2(50)
    , category_description    VARCHAR2(1000)
    , category_id             NUMBER(2)
    , NOT instantiable
      MEMBER FUNCTION category_describe RETURN VARCHAR2
      )
  NOT INSTANTIABLE NOT FINAL;
/
CREATE TYPE subcategory_ref_list_typ AS TABLE OF REF category_typ;
/
CREATE TYPE product_ref_list_typ AS TABLE OF number(6);
```

```
/
CREATE TYPE corporate_customer_typ UNDER customer_typ
      ( account_mgr_id     NUMBER(6)
      );
/
CREATE TYPE leaf_category_typ UNDER category_typ
    (
    product_ref_list    product_ref_list_typ
    , OVERRIDING MEMBER FUNCTION  category_describe RETURN VARCHAR2
    );
/
CREATE TYPE BODY leaf_category_typ AS
    OVERRIDING MEMBER FUNCTION  category_describe RETURN VARCHAR2 IS
    BEGIN
       RETURN  'leaf_category_typ';
    END;
   END;
/
CREATE TYPE composite_category_typ UNDER category_typ
      (
    subcategory_ref_list subcategory_ref_list_typ
      , OVERRIDING MEMBER FUNCTION  category_describe RETURN
VARCHAR2
      )
  NOT FINAL;
/
CREATE TYPE BODY composite_category_typ  AS
    OVERRIDING MEMBER FUNCTION category_describe RETURN VARCHAR2 IS
    BEGIN
      RETURN 'composite_category_typ';
    END;
   END;
/
CREATE TYPE catalog_typ UNDER composite_category_typ
      (
    MEMBER FUNCTION getCatalogName RETURN VARCHAR2
       , OVERRIDING MEMBER FUNCTION category_describe RETURN
VARCHAR2
      );
/
CREATE TYPE BODY catalog_typ AS
  OVERRIDING MEMBER FUNCTION category_describe RETURN varchar2 IS
  BEGIN
    RETURN 'catalog_typ';
  END;
```

```
  MEMBER FUNCTION getCatalogName RETURN varchar2 IS
  BEGIN
    -- Return the category name from the supertype
    RETURN self.category_name;
  END;
END;
/

--
==========================================================================
==
-- Table definitions
--
==========================================================================
==

CREATE TABLE categories_tab OF category_typ
    ( category_id PRIMARY KEY)
  NESTED TABLE TREAT
 (SYS_NC_ROWINFO$ AS leaf_category_typ).product_ref_list
    STORE AS product_ref_list_nestedtab
  NESTED TABLE TREAT
 (SYS_NC_ROWINFO$ AS composite_category_typ).subcategory_ref_list
    STORE AS subcategory_ref_list_nestedtab;

-- ============================================================
-- View definitions
-- ============================================================
--
-- oc_inventories

CREATE OR REPLACE VIEW oc_inventories OF inventory_typ
 WITH OBJECT OID (product_id)
 AS SELECT i.product_id,
           warehouse_typ(w.warehouse_id, w.warehouse_name,
w.location_id),
           i.quantity_on_hand
    FROM inventories i, warehouses w
    WHERE i.warehouse_id=w.warehouse_id;

-- oc_product_information

CREATE OR REPLACE VIEW oc_product_information OF product_
information_typ
 WITH OBJECT OID (product_id)
```

```
 AS SELECT p.product_id, p.product_name, p.product_description,
p.category_id,
           p.weight_class, p.warranty_period, p.supplier_id,
p.product_status,
           p.list_price, p.min_price, p.catalog_url,
           CAST(MULTISET(SELECT i.product_id,i.warehouse,i.quantity_
on_hand
                         FROM oc_inventories i
                         WHERE p.product_id=i.product_id)
                AS inventory_list_typ)
    FROM product_information p;

-- oc_customers: Multi-level collections
--
-- The view is created twice so that it can make a reference to
itself. The
-- first CREATE creates the view with a NULL in place of the
circular
-- reference. The second CREATE creates the view WITH the circular
reference,
-- which works this time because now the view already exists.

CREATE OR REPLACE VIEW oc_customers of customer_typ
 WITH OBJECT OID (customer_id)
 AS SELECT c.customer_id, c.cust_first_name, c.cust_last_name,
c.cust_address,
           c.phone_numbers,c.nls_language,c.nls_territory,c.credit_
limit,
           c.cust_email,
           CAST(MULTISET(SELECT o.order_id, o.order_mode,
                                NULL,
                                o.order_status,
                                o.order_total,o.sales_rep_id,
                                CAST(MULTISET(SELECT l.order_
id,l.line_item_id,
                                                      l.unit_
price,l.quantity,
                                                 make_ref(oc_product_
information,
                                                          l.product_id)
                                                 FROM order_items l
                                                 WHERE o.order_id =
l.order_id)
                                      AS order_item_list_typ)
                           FROM orders o
```

```
                        WHERE c.customer_id = o.customer_id)
                AS order_list_typ)
     FROM customers c;


CREATE OR REPLACE VIEW oc_customers OF customer_typ
 WITH OBJECT OID (customer_id)
 AS SELECT c.customer_id, c.cust_first_name, c.cust_last_name,
c.cust_address,
           c.phone_numbers,c.nls_language,c.nls_territory,c.credit_
limit,
           c.cust_email,
           CAST(MULTISET(SELECT o.order_id, o.order_mode,
                                MAKE_REF(oc_customers,o.customer_id),
                                o.order_status,
                                o.order_total,o.sales_rep_id,
                                CAST(MULTISET(SELECT l.order_
id,l.line_item_id,
                                                     l.unit_
price,l.quantity,
                                                MAKE_REF(oc_product_
information,
                                                         l.product_id)
                                                FROM order_items l
                                                WHERE o.order_id =
l.order_id)
                                     AS order_item_list_typ)
                           FROM orders o
                           WHERE c.customer_id = o.customer_id)
                AS order_list_typ)
     FROM customers c;

-- oc_corporate_customers

CREATE OR REPLACE VIEW oc_corporate_customers OF corporate_customer_
typ
  UNDER oc_customers
    AS SELECT c.customer_id, c.cust_first_name, c.cust_last_name,
              c.cust_address, c.phone_numbers,c.nls_language,c.nls_
territory,
              c.credit_limit, c.cust_email,
              CAST(MULTISET(SELECT o.order_id, o.order_mode,
                                MAKE_REF(oc_customers,o.customer_id),
                                o.order_status,
                                o.order_total,o.sales_rep_id,
```

```
                                  CAST(MULTISET(SELECT l.order_
id,l.line_item_id,
                                           l.unit_price,l.quantity,
                                           make_ref(oc_product_
information,
                                                    l.product_id)
                                               FROM order_items l
                                               WHERE o.order_id =
l.order_id)
                                        AS order_item_list_typ)
                               FROM orders o
                               WHERE c.customer_id = o.customer_id)
              AS order_list_typ), c.account_mgr_id
     FROM customers c;

-- oc_orders

CREATE OR REPLACE VIEW oc_orders OF order_typ WITH OBJECT OID
(order_id)
 AS SELECT o.order_id, o.order_mode,MAKE_REF(oc_
customers,o.customer_id),
        o.order_status,o.order_total,o.sales_rep_id,
       CAST(MULTISET(SELECT l.order_id,l.line_item_id,l.unit_
price,l.quantity,
                       make_ref(oc_product_information,l.product_id)
                     FROM order_items l
                     WHERE o.order_id = l.order_id)
            AS order_item_list_typ)
    FROM orders o;

--
==========================================================================
==
-- Instead-of triggers
--
==========================================================================
==

--
-- Create instead-of triggers
--
CREATE OR REPLACE TRIGGER orders_trg INSTEAD OF INSERT
 ON oc_orders FOR EACH ROW
BEGIN
   INSERT INTO ORDERS (order_id, order_mode, order_total,
```

```
                        sales_rep_id, order_status)
                VALUES (:NEW.order_id, :NEW.order_mode,
                        :NEW.order_total, :NEW.sales_rep_id,
                        :NEW.order_status);
END;
/

CREATE OR REPLACE TRIGGER orders_items_trg INSTEAD OF INSERT ON
NESTED
 TABLE order_item_list OF oc_orders FOR EACH ROW
DECLARE
    prod  product_information_typ;
BEGIN
    SELECT DEREF(:NEW.product_ref) INTO prod FROM DUAL;
    INSERT INTO order_items VALUES (prod.product_id, :NEW.order_id,
                                    :NEW.line_item_id, :NEW.unit_
price,
                                    :NEW.quantity);
END;
/

COMMIT;
```

oc_drop.sql

```
rem
rem $Header: oc_drop.sql 05-mar-2001.15:50:38 ahunold Exp $
rem
rem Copyright (c) 2001 Oracle Corporation.  All rights reserved.
rem
rem Owner  : ahunold
rem
rem NAME
rem   oc_drop.sql - drop OC subschema of OE Common Schema
rem
rem DESCRIPTON
rem   Drop all database objects
rem
rem MODIFIED   (MM/DD/YY)
rem   gxlee     03/05/01 - substituteable object table
rem   ahunold   01/29/01 - typo
rem   ahunold   01/09/01 - checkin ADE
```

```
drop table categories_tab                          cascade constraints
;

drop view oc_customers;
drop view oc_corporate_customers;
drop view oc_orders;
drop view oc_inventories;
drop view oc_product_information;

drop type order_list_typ force;
drop type product_ref_list_typ force;
drop type subcategory_ref_list_typ force;
drop type leaf_category_typ force;
drop type composite_category_typ force;
drop type catalog_typ force;
drop type category_typ force;

drop type customer_typ force;
drop type corporate_customer_typ force;
drop type warehouse_typ force;
drop type order_item_typ force;
drop type order_item_list_typ force;
drop type order_typ force;
drop type inventory_typ force;
drop type inventory_list_typ force;
drop type product_information_typ force;

commit;
```

oc_main.sql

```
rem
Rem $Header: oc_main.sql 13-apr-2001.12:12:39 ahunold Exp $
rem
rem Copyright (c) 2001 Oracle Corporation.  All rights reserved.
rem
rem Owner  : ahunold
rem
rem NAME
rem   oc_main.sql - create OC (Online Catalog) subschema in
rem                 OE (Order Entry) Common Schema
rem
rem DESCRIPTON
```

```
rem   Calls all other OC creation scripts
rem
rem MODIFIED   (MM/DD/YY)
rem   ahunold   01/29/01 - oc_comnt.sql added
rem   ahunold   01/09/01 - checkin ADE

SET ECHO ON

ALTER SESSION SET NLS_LANGUAGE=American;

prompt ...creating subschema OC in OE

REM =========================================================
REM create oc subschema (online catalog)
REM =========================================================

@@oc_cre
@@oc_popul
@@oc_comnt

spool off
```

oe_analz.sql

```
rem
Rem $Header: oe_analz.sql 06-feb-96.13:23:14 ahunold  Exp $
rem
rem Copyright (c) 2001 Oracle Corporation.  All rights reserved.
rem
rem Owner  : ahunold
rem
rem NAME
rem   oe_analz.sql - Gather statistics for OE Common Schema
rem
rem DESCRIPTON
rem
rem
rem MODIFIED   (MM/DD/YY)
rem   ahunold   01/29/01 - typos
rem   ahunold   01/09/01 - checkin ADE

EXECUTE dbms_stats.gather_table_stats ('OE', 'CUSTOMERS');

EXECUTE dbms_stats.gather_table_stats ('OE', 'ORDERS');
```

```
EXECUTE dbms_stats.gather_table_stats ('OE', 'ORDER_ITEMS');

EXECUTE dbms_stats.gather_table_stats ('OE', 'PRODUCT_INFORMATION');

EXECUTE dbms_stats.gather_table_stats ('OE', 'PRODUCT_
DESCRIPTIONS');

EXECUTE dbms_stats.gather_table_stats ('OE', 'WAREHOUSES');

EXECUTE dbms_stats.gather_table_stats ('OE', 'INVENTORIES');
```

oe_comnt.sql

```
rem
rem Header: oe_comnt.sql 09-jan-01
rem
rem Copyright (c) 2001 Oracle Corporation.  All rights reserved.
rem
rem Owner  : ahunold
rem
rem NAME
rem   oe_comnt.sql - create comments for OE Common Schema
rem
rem DESCRIPTON
rem
rem
rem MODIFIED   (MM/DD/YY)
rem   ahunold   01/30/01 - OE script headers
rem   ahunold   01/24/01 - Eliminate extra lines from last merge
rem   ahunold   01/09/01 - checkin ADE

COMMENT ON TABLE oe.customers IS
'Contains customers data either entered by an employee or by the
customer
him/herself over the Web.';

COMMENT ON COLUMN oe.customers.cust_address IS
'Object column of type address_typ.';

COMMENT ON COLUMN oe.customers.phone_numbers IS
'Varray column of type phone_list_typ';
.
```

```
COMMENT ON COLUMN oe.customers.cust_geo_location IS
'SDO (spatial) column.';

COMMENT ON COLUMN oe.customers.cust_first_name IS
'NOT NULL constraint.';

COMMENT ON COLUMN oe.customers.cust_last_name IS
'NOT NULL constraint.';

COMMENT ON COLUMN oe.customers.credit_limit IS
'Check constraint.';

COMMENT ON COLUMN oe.customers.customer_id IS
'Primary key column.';

COMMENT ON COLUMN oe.customers.account_mgr_id IS
'References hr.employees.employee_id.';

REM
================================================================================
=======

COMMENT ON TABLE oe.warehouses IS
'Warehouse data unspecific to any industry.';

COMMENT ON COLUMN oe.warehouses.wh_geo_location IS
'SDO (spatial) column.';

COMMENT ON COLUMN oe.warehouses.warehouse_id IS
'Primary key column.';

COMMENT ON COLUMN oe.warehouses.location_id IS
'Primary key column, references hr.locations.location_id.';

REM
================================================================================
=======

COMMENT ON TABLE oe.order_items IS
'Example of many-to-many resolution.';

COMMENT ON COLUMN oe.order_items.order_id IS
'Part of concatenated primary key, references orders.order_id.';

COMMENT ON COLUMN oe.order_items.product_id IS
```

```
'References product_information.product_id.';

COMMENT ON COLUMN oe.order_items.line_item_id IS
'Part of concatenated primary key.';

COMMENT ON COLUMN oe.orders.order_status IS
'0: Not fully entered, 1: Entered, 2: Canceled - bad credit, -
3: Canceled - by customer, 4: Shipped - whole order, -
5: Shipped - replacement items, 6: Shipped - backlog on items, -
7: Shipped - special delivery, 8: Shipped - billed, 9: Shipped -
payment plan,-
10: Shipped - paid';

REM
================================================================================
=======

COMMENT ON TABLE oe.orders IS
'Contains orders entered by a salesperson as well as over the Web.';

COMMENT ON COLUMN oe.orders.order_date IS
'TIMESTAMP WITH LOCAL TIME ZONE column, NOT NULL constraint.';

COMMENT ON COLUMN oe.orders.order_id IS
'PRIMARY KEY column.';

COMMENT ON COLUMN oe.orders.sales_rep_id IS
'References hr.employees.employee_id.';

COMMENT ON COLUMN oe.orders.promotion_id IS
'Sales promotion ID. Used in SH schema';

COMMENT ON COLUMN oe.orders.order_mode IS
'CHECK constraint.';

COMMENT ON COLUMN oe.orders.order_total IS
'CHECK constraint.';

REM
================================================================================
=======

COMMENT ON TABLE oe.inventories IS
'Tracks availability of products by product_it and warehouse_id.';
```

```
COMMENT ON COLUMN oe.inventories.product_id IS
'Part of concatenated primary key, references product_
information.product_id.';

COMMENT ON COLUMN oe.inventories.warehouse_id IS
'Part of concatenated primary key, references warehouses.warehouse_
id.';

REM
================================================================================
=======

COMMENT ON TABLE oe.product_information IS
'Non-industry-specific data in various categories.';

COMMENT ON COLUMN oe.product_information.product_id IS
'Primary key column.';

COMMENT ON COLUMN oe.product_information.product_description IS
'Primary language description corresponding to translated_
description in
oe.product_descriptions, added to provide non-NLS text columns for
OC views
to accss.';

COMMENT ON COLUMN oe.product_information.category_id IS
'Low cardinality column, can be used for bitmap index.
Schema SH uses it as foreign key';

COMMENT ON COLUMN oe.product_information.weight_class IS
'Low cardinality column, can be used for bitmap index.';

COMMENT ON COLUMN oe.product_information.warranty_period IS
'INTERVAL YEAER TO MONTH column, low cardinality, can be used for
bitmap
index.';

COMMENT ON COLUMN oe.product_information.supplier_id IS
'Offers possibility of extensions outside Common Schema.';

COMMENT ON COLUMN oe.product_information.product_status IS
'Check constraint. Appropriate for complex rules, such as "All
products in
status PRODUCTION must have at least one inventory entry." Also
appropriate
```

```
for a trigger auditing status change.';

REM
=================================================================
=======

COMMENT ON TABLE product_descriptions IS
'Non-industry-specific design, allows selection of
NLS-setting-specific data
derived at runtime, for example using the products view.';


COMMENT ON COLUMN product_descriptions.product_id IS
'Primary key column.';

COMMENT ON COLUMN product_descriptions.language_id IS
'Primary key column.';

REM Description of OE views
==================================================

COMMENT ON TABLE products IS
'This view joins product_information and product_descriptions, using
NLS
settings to pick the appropriate language-specific product
description.';

COMMENT ON TABLE bombay_inventory IS
'This view shows inventories at the Bombay warehouse.';

COMMENT ON TABLE sydney_inventory IS
'This view shows inventories at the Sydney warehouse.';

COMMENT ON TABLE toronto_inventory IS
'This view shows inventories at the Toronto warehouse.';
```

oe_cre.sql

```
rem
rem Header: oe_cre.sql 09-jan-01
rem
rem Copyright (c) 2001 Oracle Corporation.  All rights reserved.
rem
rem Owner  : ahunold
```

```
rem
rem NAME
rem   oe_cre.sql - create OE Common Schema
rem
rem DESCRIPTON
rem   Creates database objects. The script assumes that the HR
schema
rem   is present.
rem
rem MODIFIED   (MM/DD/YY)
rem   ahunold   03/02/01 - eliminating DROP SEQUENCE
rem   ahunold   01/30/01 - OE script headers
rem   ahunold   01/24/01 - Eliminate extra lines from last merge
rem   ahunold   01/05/01 - promo_id
rem   ahunold   01/05/01 - NN constraints in product_descriptions
rem   ahunold   01/09/01 - checkin ADE

--
==========================================================================
==
-- Type definitions
--
==========================================================================
==

CREATE TYPE cust_address_typ AS OBJECT
    ( street_address     VARCHAR2(40)
    , postal_code        VARCHAR2(10)
    , city               VARCHAR2(30)
    , state_province     VARCHAR2(10)
    , country_id         CHAR(2)
    );
/

REM
==========================================================================
=======
REM Create phone_list_typ varray to be varray column in customers
table.
REM
==========================================================================
=======

CREATE TYPE phone_list_typ AS VARRAY(5) OF VARCHAR2(25);
/
```

```
REM
================================================================================
=======
REM Create customers table.
REM The cust_geo_location column will become MDSYS.SDO_GEOMETRY
(spatial)
REM datatype when appropriate scripts and data are available.
REM
================================================================================
=======

CREATE TABLE customers
    ( customer_id        NUMBER(6)
    , cust_first_name    VARCHAR2(20) CONSTRAINT cust_fname_nn NOT
NULL
    , cust_last_name     VARCHAR2(20) CONSTRAINT cust_lname_nn NOT
NULL
    , cust_address       cust_address_typ
    , phone_numbers      phone_list_typ
    , nls_language       VARCHAR2(3)
    , nls_territory      VARCHAR2(30)
    , credit_limit       NUMBER(9,2)
    , cust_email         VARCHAR2(30)
    , account_mgr_id     NUMBER(6)
    , cust_geo_location  MDSYS.SDO_GEOMETRY
    , CONSTRAINT         customer_credit_limit_max
                         CHECK (credit_limit <= 5000)
    , CONSTRAINT         customer_id_min
                         CHECK (customer_id > 0)
    ) ;

CREATE UNIQUE INDEX customers_pk
   ON customers (customer_id) ;

REM Both table and indexes are analyzed using the oe_analz.sql
script.

ALTER TABLE customers
ADD ( CONSTRAINT customers_pk
      PRIMARY KEY (customer_id)
    ) ;

REM
================================================================================
```

```
=======
REM Create warehouses table;
REM  includes spatial data column wh_geo_location and
REM  XML type warehouse_spec (was bug b41)
REM
================================================================================
=======

CREATE TABLE warehouses
    ( warehouse_id       NUMBER(3)
    , warehouse_spec     SYS.XMLTYPE
    , warehouse_name     VARCHAR2(35)
    , location_id        NUMBER(4)
    , wh_geo_location    MDSYS.SDO_GEOMETRY
    ) ;

CREATE UNIQUE INDEX warehouses_pk
ON warehouses (warehouse_id) ;

ALTER TABLE warehouses
ADD (CONSTRAINT warehouses_pk PRIMARY KEY (warehouse_id)
    );

REM
================================================================================
=======
REM Create table order_items.
REM
================================================================================
=======

CREATE TABLE order_items
    ( order_id           NUMBER(12)
    , line_item_id       NUMBER(3)  NOT NULL
    , product_id         NUMBER(6)  NOT NULL
    , unit_price         NUMBER(8,2)
    , quantity           NUMBER(8)
    ) ;

CREATE UNIQUE INDEX order_items_pk
ON order_items (order_id, line_item_id) ;

CREATE UNIQUE INDEX order_items_uk
ON order_items (order_id, product_id) ;
```

```
ALTER TABLE order_items
ADD ( CONSTRAINT order_items_pk PRIMARY KEY (order_id, line_item_id)
    );

CREATE OR REPLACE TRIGGER insert_ord_line
  BEFORE INSERT ON order_items
  FOR EACH ROW
  DECLARE
    new_line number;
  BEGIN
    SELECT (NVL(MAX(line_item_id),0)+1) INTO new_line
      FROM order_items
      WHERE order_id = :new.order_id;
    :new.line_item_id := new_line;
  END;
/

REM
================================================================================
=======
REM Create table orders, which includes a TIMESTAMP column and a
check
REM constraint.
REM
================================================================================
=======

CREATE TABLE orders
    ( order_id           NUMBER(12)
    , order_date         TIMESTAMP WITH LOCAL TIME ZONE
CONSTRAINT order_date_nn NOT NULL
    , order_mode         VARCHAR2(8)
    , customer_id        NUMBER(6) CONSTRAINT order_customer_id_nn
NOT NULL
    , order_status       NUMBER(2)
    , order_total        NUMBER(8,2)
    , sales_rep_id       NUMBER(6)
    , promotion_id       NUMBER(6)
    , CONSTRAINT         order_mode_lov
                         CHECK (order_mode in ('direct','online'))
    , constraint         order_total_min
                         check (order_total >= 0)
    ) ;

CREATE UNIQUE INDEX order_pk
```

```
ON orders (order_id) ;

ALTER TABLE orders
ADD ( CONSTRAINT order_pk
      PRIMARY KEY (order_id)
    );
REM
================================================================================
=======
REM Create inventories table, which contains a concatenated primary
key.
REM
================================================================================
=======

CREATE TABLE inventories
  ( product_id         NUMBER(6)
  , warehouse_id       NUMBER(3) CONSTRAINT inventory_warehouse_id_
nn NOT NULL
  , quantity_on_hand   NUMBER(8)
CONSTRAINT inventory_qoh_nn NOT NULL
  , CONSTRAINT inventory_pk PRIMARY KEY (product_id, warehouse_id)
  ) ;

REM
================================================================================
=======
REM Create table product_information, which contains an INTERVAL
datatype and
REM a CHECK ... IN constraint.
REM
================================================================================
=======

CREATE TABLE product_information
    ( product_id          NUMBER(6)
    , product_name        VARCHAR2(50)
    , product_description VARCHAR2(2000)
    , category_id         NUMBER(2)
    , weight_class        NUMBER(1)
    , warranty_period     INTERVAL YEAR TO MONTH
    , supplier_id         NUMBER(6)
    , product_status      VARCHAR2(20)
    , list_price          NUMBER(8,2)
    , min_price           NUMBER(8,2)
```

```
    , catalog_url         VARCHAR2(50)
    , CONSTRAINT          product_status_lov
                          CHECK (product_status in ('orderable'
                                                    ,'planned'
                                                    ,'under
development'
                                                    ,'obsolete')
                                 )
    ) ;

ALTER TABLE product_information
ADD ( CONSTRAINT product_information_pk PRIMARY KEY (product_id)
    );

REM
================================================================
=======
REM Create table product_descriptions, which contains NVARCHAR2
columns for
REM NLS-language information.
REM
================================================================
=======

CREATE TABLE product_descriptions
    ( product_id             NUMBER(6)
    , language_id            VARCHAR2(3)
    , translated_name        NVARCHAR2(50)
CONSTRAINT translated_name_nn NOT NULL
    , translated_description NVARCHAR2(2000)
CONSTRAINT translated_desc_nn NOT NULL
    );

CREATE UNIQUE INDEX prd_desc_pk
ON product_descriptions(product_id,language_id) ;

ALTER TABLE product_descriptions
ADD ( CONSTRAINT product_descriptions_pk
PRIMARY KEY (product_id, language_id));

ALTER TABLE orders
ADD ( CONSTRAINT orders_sales_rep_fk
      FOREIGN KEY (sales_rep_id)
      REFERENCES hr.employees(employee_id)
      ON DELETE SET NULL
```

```
    ) ;

ALTER TABLE orders
ADD ( CONSTRAINT orders_customer_id_fk
      FOREIGN KEY (customer_id)
      REFERENCES customers(customer_id)
      ON DELETE SET NULL
    ) ;

ALTER TABLE warehouses
ADD ( CONSTRAINT warehouses_location_fk
      FOREIGN KEY (location_id)
      REFERENCES hr.locations(location_id)
      ON DELETE SET NULL
    ) ;

ALTER TABLE customers
ADD ( CONSTRAINT customers_account_manager_fk
      FOREIGN KEY (account_mgr_id)
      REFERENCES hr.employees(employee_id)
      ON DELETE SET NULL
    ) ;

ALTER TABLE inventories
ADD ( CONSTRAINT inventories_warehouses_fk
      FOREIGN KEY (warehouse_id)
      REFERENCES warehouses (warehouse_id)
      ENABLE NOVALIDATE
    ) ;

ALTER TABLE inventories
ADD ( CONSTRAINT inventories_product_id_fk
      FOREIGN KEY (product_id)
      REFERENCES product_information (product_id)
    ) ;

ALTER TABLE order_items
ADD ( CONSTRAINT order_items_order_id_fk
      FOREIGN KEY (order_id)
      REFERENCES orders(order_id)
      ON DELETE CASCADE
enable novalidate
    ) ;

ALTER TABLE order_items
```

```
ADD ( CONSTRAINT order_items_product_id_fk
      FOREIGN KEY (product_id)
      REFERENCES product_information(product_id)
    ) ;

REM
================================================================================
=======
REM Create cross-schema synonyms
REM
================================================================================
=======

CREATE SYNONYM countries FOR hr.countries;

CREATE SYNONYM locations FOR hr.locations;

CREATE SYNONYM departments FOR hr.departments;

CREATE SYNONYM jobs FOR hr.jobs;

CREATE SYNONYM employees FOR hr.employees;

CREATE SYNONYM job_history FOR hr.job_history;

REM
================================================================================
=======
REM Create sequences
REM
================================================================================
=======

CREATE SEQUENCE orders_seq
 START WITH     1000
 INCREMENT BY   1
 NOCACHE
 NOCYCLE;

REM
================================================================================
=======
REM Need commit for PO
REM
================================================================================
```

```
=======

COMMIT;
```

oe_drop.sql

```
rem
rem Header: oe_drop.sql 09-jan-01
rem
rem Copyright (c) 2001 Oracle Corporation.  All rights reserved.
rem
rem Owner  : ahunold
rem
rem NAME
rem   oe_drop.sql - drop OE Common Schema
rem
rem DESCRIPTON
rem   Deletes database objects.
rem
rem MODIFIED   (MM/DD/YY)
rem   ahunold   01/30/01 - OE script headers
rem   ahunold   01/09/01 - checkin ADE

CONNECT oe/&password_OE

DROP TABLE customers              CASCADE CONSTRAINTS ;
DROP TABLE warehouses             CASCADE CONSTRAINTS ;
DROP TABLE order_items            CASCADE CONSTRAINTS ;
DROP TABLE orders                 CASCADE CONSTRAINTS ;
DROP TABLE inventories            CASCADE CONSTRAINTS ;
DROP TABLE product_information    CASCADE CONSTRAINTS ;
DROP TABLE product_descriptions   CASCADE CONSTRAINTS ;
DROP TYPE cust_address_typ;
DROP TYPE phone_list_typ;
DROP SYNONYM countries;
DROP SYNONYM locations;
DROP SYNONYM departments;
DROP SYNONYM jobs;
DROP SYNONYM employees;
DROP SYNONYM job_history;
```

oe_idx.sql

```
rem
rem Header: oe_idx.sql 09-jan-01
rem
rem Copyright (c) 2001 Oracle Corporation.  All rights reserved.
rem
rem Owner  : ahunold
rem
rem NAME
rem   oe_idx.sql - create indexes for OE Common Schema
rem
rem DESCRIPTON
rem   Re-Creates indexes
rem
rem MODIFIED   (MM/DD/YY)
rem   ahunold   03/02/01 - eliminating DROP INDEX
rem   ahunold   01/30/01 - OE script headers
rem   ahunold   01/09/01 - checkin ADE

CREATE INDEX whs_location_ix
ON warehouses (location_id);

CREATE INDEX inv_product_ix
ON inventories (product_id);

CREATE INDEX inv_warehouse_ix
ON inventories (warehouse_id);

CREATE INDEX item_order_ix
ON order_items (order_id);

CREATE INDEX item_product_ix
ON order_items (product_id);

CREATE INDEX ord_sales_rep_ix
ON orders (sales_rep_id);

CREATE INDEX ord_customer_ix
ON orders (customer_id);

CREATE INDEX ord_order_date_ix
ON orders (order_date);

CREATE INDEX cust_account_manager_ix
```

```
ON customers (account_mgr_id);

CREATE INDEX cust_lname_ix
ON customers (cust_last_name);

CREATE INDEX cust_email_ix
ON customers (cust_email);

CREATE INDEX prod_name_ix
ON product_descriptions (translated_name);

CREATE INDEX prod_supplier_ix
ON product_information (supplier_id);

CREATE INDEX cust_upper_name_ix
ON customers (UPPER(cust_last_name), UPPER(cust_first_name));
```

oe_main.sql

```
rem
rem Header: oe_main.sql 09-jan-01
rem
rem Copyright (c) 2001 Oracle Corporation.  All rights reserved.
rem
rem Owner  : ahunold
rem
rem NAME
rem   oe_main.sql - Main script for OE schema, including OC
subschema
rem
rem DESCRIPTON
rem   Creates and populated the Order Entry (OE) and Online
rem   Catalog (OC) Sample Schema
rem
rem NOTES
rem   Run as SYS or SYSTEM
rem   Prerequisites:
rem     Tablespaces present
rem     Database enabled for Spatial and XML
rem
rem MODIFIED   (MM/DD/YY)
rem   ahunold   04/13/01 - spool, additional parameter
rem   ahunold   03/29/01 - spool
rem   ahunold   03/12/01 - prompts
```

```
rem   ahunold   03/02/01 - NLS_LANGUAGE
rem   ahunold   01/09/01 - checkin ADE

SET ECHO ON

ALTER SESSION SET NLS_LANGUAGE=American;

PROMPT
PROMPT specify password for OE as parameter 1:
define pass     = &1
PROMPT
PROMPT specify default tablespeace for OE as parameter 2:
define tbs      = &2
PROMPT
PROMPT specify temporary tablespace for OE as parameter 3:
define ttbs     = &3
PROMPT
PROMPT specify password for HR as parameter 4:
define passhr   = &4
PROMPT
PROMPT specify password for SYS as parameter 5:
define pass_sys = &5
PROMPT
PROMPT specify path for log files as parameter 6:
define log_path = &6
PROMPT

-- The first dot in the spool command below is
-- the SQL*Plus concatenation character

spool &log_path.oe_oc_main.log

-- Dropping the user with all its objects

DROP USER oe CASCADE;

REM =======================================================
REM create user
REM
REM The user is assigned tablespaces and quota in separate
REM ALTER USER statements so that the CREATE USER statement
REM will succeed even if the demo and temp tablespaces do
REM not exist.
REM =======================================================
```

```
CREATE USER oe IDENTIFIED BY &pass;

ALTER USER oe DEFAULT TABLESPACE &tbs QUOTA UNLIMITED ON &tbs;

ALTER USER oe TEMPORARY TABLESPACE &ttbs;


GRANT CREATE SESSION
,     CREATE TABLE
,     CREATE VIEW
,     CREATE SYNONYM
,     CREATE TYPE
,     CREATE MATERIALIZED VIEW
,     CREATE TRIGGER
,     CREATE SEQUENCE
,     QUERY REWRITE
TO    oe;

REM =======================================================
REM grants from sys schema
REM =======================================================

CONNECT sys/&pass_sys AS SYSDBA;
GRANT execute ON sys.dbms_stats TO oe;

REM =======================================================
REM grants from hr schema
REM =======================================================

CONNECT hr/&passhr;
GRANT REFERENCES, SELECT ON employees TO oe;
GRANT REFERENCES, SELECT ON countries TO oe;
GRANT REFERENCES, SELECT ON locations TO oe;
GRANT SELECT ON jobs TO oe;
GRANT SELECT ON job_history TO oe;
GRANT SELECT ON departments TO oe;


REM =======================================================
REM create oe schema (order entry)
REM =======================================================

CONNECT oe/&pass
@@oe_cre
REM Populate Product_Information, Product_Description, Warehouse
```

```
REM          Customer, Orders, Order_Items, and Inventory tables
@@oe_p_pi
@@oe_p_pd
@@oe_p_whs
@@oe_p_cus
@@oe_p_ord
@@oe_p_itm
@@oe_p_inv
@@oe_views
@@oe_comnt
@@oe_idx
@@oe_analz

@@oc_main

spool off
```

oe_views.sql

```
rem
rem Header: oe_views.sql 09-jan-01
rem
rem Copyright (c) 2001 Oracle Corporation.  All rights reserved.
rem
rem Owner  : ahunold
rem
rem NAME
rem   oe_views.sql - OE Common Schema
rem
rem DESCRIPTON
rem   Create all views
rem
rem MODIFIED   (MM/DD/YY)
rem   ahunold   01/09/01 - checkin ADE

CREATE OR REPLACE VIEW products
AS
SELECT i.product_id
,      d.language_id
,      CASE WHEN d.language_id IS NOT NULL
            THEN d.translated_name
            ELSE TRANSLATE(i.product_name USING NCHAR_CS)
       END    AS product_name
,      i.category_id
```

```
,      CASE WHEN d.language_id IS NOT NULL
            THEN d.translated_description
            ELSE TRANSLATE(i.product_description USING NCHAR_CS)
       END    AS product_description
,      i.weight_class
,      i.warranty_period
,      i.supplier_id
,      i.product_status
,      i.list_price
,      i.min_price
,      i.catalog_url
FROM   product_information  i
,      product_descriptions d
WHERE  d.product_id  (+) = i.product_id
AND    d.language_id (+) = sys_context('USERENV','LANG');

REM ======================================================
REM Create some inventory views
REM ======================================================

CREATE OR REPLACE VIEW sydney_inventory
AS
SELECT p.product_id
,      p.product_name
,      i.quantity_on_hand
FROM   inventories i
,      warehouses  w
,      products    p
WHERE  p.product_id = i.product_id
AND    i.warehouse_id = w.warehouse_id
AND    w.warehouse_name = 'Sydney';

CREATE OR REPLACE VIEW bombay_inventory
AS
SELECT p.product_id
,      p.product_name
,      i.quantity_on_hand
FROM   inventories i
,      warehouses  w
,      products    p
WHERE  p.product_id = i.product_id
AND    i.warehouse_id = w.warehouse_id
AND    w.warehouse_name = 'Bombay';

CREATE OR REPLACE VIEW toronto_inventory
```

```
AS
SELECT p.product_id
,      p.product_name
,      i.quantity_on_hand
FROM   inventories i
,      warehouses  w
,      products    p
WHERE  p.product_id = i.product_id
AND    i.warehouse_id = w.warehouse_id
AND    w.warehouse_name = 'Toronto';

REM ======================================================
REM Create product_prices view of product_information
REM columns to show view with a GROUP BY clause.
REM ======================================================

CREATE OR REPLACE VIEW product_prices
AS
SELECT category_id
,      COUNT(*)        as "#_OF_PRODUCTS"
,      MIN(list_price) as low_price
,      MAX(list_price) as high_price
FROM   product_information
GROUP BY category_id;
```

Product Media (PM) Scripts

This section shows the PM scripts in alphabetical order.

pm_analz.sql

```
Rem
Rem $Header: pm_analz.sql 07-mar-2001.14:29:47 ahunold Exp $
Rem
Rem pm_analz.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      pm_analz.sql - Gathering statistics for HR schema
Rem
Rem    DESCRIPTION
Rem      Staistics are used by the cost based optimizer to
```

```
Rem      choose the best physical access strategy
Rem
Rem    NOTES
Rem      Results can be viewed in columns of DBA_TABLES,
Rem      DBA_TAB_COLUMNS and such
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     03/07/01 - Merged ahunold_hr_analz
Rem    ahunold     03/07/01 - Created
Rem

SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET ECHO ON

EXECUTE dbms_stats.gather_table_stats ('PM','ONLINE_MEDIA');

EXECUTE dbms_stats.gather_table_stats ('PM','PRINT_MEDIA');
```

pm_cre.sql

```
Rem
Rem $Header: pm_cre.sql 09-feb-2001.13:09:54 ahunold Exp $
Rem
Rem pm_cre.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      pm_cre.sql - Table creation scripts
Rem
Rem    DESCRIPTION
Rem      PM is the Product Media schema of the Oracle 9i Sample
Rem        Schemas
Rem
Rem    NOTES
Rem
Rem
Rem    MODIFIED   (MM/DD/YY)
```

```
Rem    ahunold     02/09/01 - new load method
Rem    ahunold     02/05/01 - Created
Rem

REM
=====================================================================
REM Create TYPE adheader_typ to hold different headers used in
REM advertisements, the header name, date of creation, header text,
and
REM logo used. pm.print_media ad_header column has type adheader_
typ.

CREATE TYPE adheader_typ AS OBJECT
    ( header_name        VARCHAR2(256)
    , creation_date      DATE
    , header_text        VARCHAR2(1024)
    , logo               BLOB
    );
/

REM
=====================================================================
===
REM Create TYPE textdoc_tab as a nested table for
REM advertisements stored in different formats. Document type can be
pdf,
REM html,Word,Frame, ...
REM pm.print_media ad_textdocs_ntab column has type textdoc_tab.

CREATE TYPE textdoc_typ AS OBJECT
    ( document_typ      VARCHAR2(32)
    , formatted_doc     BLOB
    ) ;
/
CREATE TYPE textdoc_tab AS TABLE OF textdoc_typ;
/
REM
=====================================================================
==
REM Create table online_media to hold media for the online catalog
REM or other marketing/training needs.
REM pm.online_media has a foreign key on product_id that references
the
REM oe.product_information table. pm.online_media has a primary key
on
```

```
REM product_id.

CREATE TABLE online_media
   ( product_id         NUMBER(6)
   , product_photo      ORDSYS.ORDImage
   , product_photo_signature  ORDSYS.ORDImageSignature
   , product_thumbnail  ORDSYS.ORDImage
   , product_video      ORDSYS.ORDVideo
   , product_audio      ORDSYS.ORDAudio
   , product_text       CLOB
   , product_testimonialsORDSYS.ORDDoc
   ) ;

CREATE UNIQUE INDEX onlinemedia_pk
    ON online_media (product_id);

ALTER TABLE online_media
ADD ( CONSTRAINT onlinemedia__pk
      PRIMARY KEY (product_id)
    , CONSTRAINT loc_c_id_fk
                FOREIGN KEY (product_id)
                REFERENCES oe.product_information(product_id)
    ) ;

REM
================================================================================
REM Create table print_media to hold print advertising information.
REM pm.print_media has a foreign key on product_id that references
the
REM oe.product_information table. pm.print_media has a primary key
on
REM ad_id and product. pm.print_media references a nested table, ad_
textdoc_ntab, and
REM column object of type adheader_typ.

CREATE TABLE print_media
    ( product_id        NUMBER(6)
    , ad_id             NUMBER(6)
    , ad_composite      BLOB
    , ad_sourcetext     CLOB
    , ad_finaltext      CLOB
    , ad_fltextn        NCLOB
    , ad_textdocs_ntab  textdoc_tab
    , ad_photo          BLOB
```

```
    , ad_graphic        BFILE
    , ad_header         adheader_typ
    , press_release     LONG
    ) NESTED TABLE ad_textdocs_ntab STORE AS textdocs_nestedtab;

CREATE UNIQUE INDEX printmedia_pk
    ON print_media (product_id, ad_id);

ALTER TABLE print_media
ADD ( CONSTRAINT printmedia__pk
      PRIMARY KEY (product_id, ad_id)
    , CONSTRAINT printmedia_fk
                FOREIGN KEY (product_id)
                REFERENCES oe.product_information(product_id)
    ) ;

COMMIT;
```

pm_drop.sql

```
REM script name:     pm_drop.sql
REM purpose:         this script drops all tables from the pm schema
REM version:         8.2.0
REM release:         Sep 14, 2000
REM bugs fixed:      b7

REM
================================================================

DROP INDEX onlinemedia CASCADE CONSTRAINTS;
DROP INDEX printmedia CASCADE CONSTRAINTS;
DROP TABLE online_media CASCADE CONSTRAINTS;
DROP TABLE print_media CASCADE CONSTRAINTS;
DROP TYPE adheader_typ;
DROP TYPE textdoc_typ;
DROP TYPE testdoc_tab;

COMMIT;
```

pm_main.sql

```
Rem
Rem $Header: pm_main.sql 12-apr-2001.21:22:02 ahunold Exp $
Rem
```

```
Rem pm_main.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      pm_main.sql - Main schema creation and load script
Rem
Rem    DESCRIPTION
Rem      PM is the Product Media schema of the Oracle 9i Sample
Rem        Schemas
Rem
Rem    NOTES
Rem     1) use absolute pathnames as parameters 6.
Rem        UNIX: echo $ORACLE_HOME/demo/schema/product_media
Rem     2) there are hard-coded file names in the
Rem        data file pm_p_lob.dat. Should you want to create
Rem        and populate the PM Sample Schema from a location
Rem        other than the one chosen during installation, you
Rem        will have to edit this data file.
Rem     3) Run this as SYS or SYSTEM
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem     ahunold    04/13/01 - concatenation, no @@
Rem     ahunold    04/10/01 - added parameters 7 and 8
Rem     ahunold    03/29/01 - notes, spool
Rem     ahunold    03/20/01 - no ALTER USER
Rem     ahunold    03/12/01 - prompts & directory
Rem     ahunold    03/07/01 - pm_analz.sql.
Rem     ahunold    02/20/01 - removing pm_p_ini and pm_code
Rem     ahunold    02/09/01 - password passing for pm_p_lob
Rem     ahunold    02/05/01 - Created
Rem

SET ECHO ON

ALTER SESSION SET NLS_LANGUAGE=American;

PROMPT
PROMPT specify password for PM as parameter 1:
define pass     = &1
PROMPT
PROMPT specify default tablespeace for PM as parameter 2:
define tbs      = &2
PROMPT
PROMPT specify temporary tablespace for PM as parameter 3:
```

```
define ttbs     = &3
PROMPT
PROMPT specify password for OE as parameter 4:
define passoe   = &4
PROMPT
PROMPT specify password for SYS as parameter 5:
define pass_sys = &5
PROMPT
PROMPT specify directory path for the PM data files as parameter 6:
define data_path = &6
PROMPT
PROMPT specify directory path for the PM load log files as parameter
7:
define log_path = &7
PROMPT
PROMPT specify work directory path as parameter 8:
define work_path = &8
PROMPT

-- The first dot in the spool command below is
-- the SQL*Plus concatenation character

spool &log_path.pm_main.log

-- Dropping the user with all its objects

DROP USER pm CASCADE;

CREATE USER pm IDENTIFIED BY &pass;
ALTER USER pm DEFAULT TABLESPACE &tbs QUOTA UNLIMITED ON &tbs;
ALTER USER pm TEMPORARY TABLESPACE &ttbs;

GRANT CREATE SESSION
,     CREATE TABLE
,     CREATE TYPE
,     CREATE ANY INDEX
,     CREATE PROCEDURE
TO    pm;

REM =======================================================
REM grants from oe schema
REM =======================================================

CONNECT oe/&passoe
```

```
GRANT REFERENCES, SELECT ON product_information TO pm;
GRANT SELECT ON order_items TO pm;
GRANT SELECT ON orders TO pm;
GRANT SELECT ON product_descriptions TO pm;
GRANT SELECT ON inventories TO pm;
GRANT SELECT ON customers TO pm;
GRANT SELECT ON warehouses TO pm;


REM =========================================================
REM grants from sys schema
REM =========================================================

CONNECT sys/&pass_sys AS SYSDBA;

GRANT execute ON sys.dbms_stats TO pm;

CREATE OR REPLACE DIRECTORY media_dir AS '&data_path';

GRANT READ ON DIRECTORY media_dir TO PUBLIC WITH GRANT OPTION;

REM =========================================================
REM create pm schema (product media)
REM =========================================================

CONNECT pm/&pass

ALTER SESSION SET NLS_LANGUAGE=American;

@&data_path.pm_cre.sql   -- create objects
@&data_path.pm_p_ord.sq  -- load ORDSYS types

REM =========================================================
REM use sqlldr to populate PRINT_MEDIA and its nested table
REM =========================================================

@&data_path.pm_p_lob &pass &data_path &log_path &work_path

REM =========================================================
REM finish
REM =========================================================

@?/demo/schema/product_media/pm_analz -- gather statistics

spool off
```

Queued Shipping (QS) Scripts

This section shows the QS Scripts in alphabetical order.

qs_adm.sql

```
Rem
Rem $Header: qs_adm.sql 26-feb-2001.16:50:49 ahunold Exp $
Rem
Rem qs_adm.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      qs_adm.sql - Administration schema for QS schema group
Rem
Rem    DESCRIPTION
Rem      All object types are created in the qs_adm schema. All
Rem      application schemas that host any propagation source
Rem      queues are given the ENQUEUE_ANY system level privilege
Rem      allowing the application schemas to enqueue to the
Rem      destination queue.
Rem
Rem    NOTES
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     02/26/01 - Merged ahunold_qs_filenames
Rem    ahunold     02/26/01 - Created
Rem

CREATE OR REPLACE TYPE customer_typ AS OBJECT (
        customer_id     NUMBER,
        name            VARCHAR2(100),
        street          VARCHAR2(100),
        city            VARCHAR2(30),
        state           VARCHAR2(2),
        zip             NUMBER,
        country         VARCHAR2(100));
/


CREATE OR REPLACE TYPE orderitem_typ AS OBJECT (
```

```
line_item_idNUMBER,
        quantity        NUMBER,
unit_priceNUMBER,
product_idNUMBER,
        subtotal        NUMBER);
/

CREATE OR REPLACE TYPE orderitemlist_vartyp AS VARRAY (20) OF
ORDERITEM_TYP;
/

CREATE OR REPLACE TYPE order_typ AS OBJECT (
        orderno         NUMBER,
        status          VARCHAR2(30),
        ordertype       VARCHAR2(30),
        orderregion     VARCHAR2(30),
        customer        customer_typ,
        paymentmethod   VARCHAR2(30),
        items           orderitemlist_vartyp,
        total           NUMBER);
/

GRANT EXECUTE ON order_typ to QS;
GRANT EXECUTE ON orderitemlist_vartyp to QS;
GRANT EXECUTE ON orderitem_typ to QS;
GRANT EXECUTE ON customer_typ to QS;
execute dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','QS',FALSE);

GRANT EXECUTE ON order_typ to QS_WS;
GRANT EXECUTE ON orderitemlist_vartyp to QS_WS;
GRANT EXECUTE ON orderitem_typ to QS_WS;
GRANT EXECUTE ON customer_typ to QS_WS;
execute dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','QS_
WS',FALSE);

GRANT EXECUTE ON order_typ to QS_ES;
GRANT EXECUTE ON orderitemlist_vartyp to QS_ES;
GRANT EXECUTE ON orderitem_typ to QS_ES;
GRANT EXECUTE ON customer_typ to QS_ES;
execute dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','QS_
ES',FALSE);

GRANT EXECUTE ON order_typ to QS_OS;
GRANT EXECUTE ON orderitemlist_vartyp to QS_OS;
GRANT EXECUTE ON orderitem_typ to QS_OS;
```

```
GRANT EXECUTE ON customer_typ to QS_OS;
execute dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','QS_
OS',FALSE);

GRANT EXECUTE ON order_typ to qs_cbadm;
GRANT EXECUTE ON orderitemlist_vartyp to qs_cbadm;
GRANT EXECUTE ON orderitem_typ to qs_cbadm;
GRANT EXECUTE ON customer_typ to qs_cbadm;

GRANT EXECUTE ON order_typ to QS_CB;
GRANT EXECUTE ON orderitemlist_vartyp to QS_CB;
GRANT EXECUTE ON orderitem_typ to QS_CB;
GRANT EXECUTE ON customer_typ to QS_CB;

GRANT EXECUTE ON order_typ to QS_CS;
GRANT EXECUTE ON orderitemlist_vartyp to QS_CS;
GRANT EXECUTE ON orderitem_typ to QS_CS;
GRANT EXECUTE ON customer_typ to QS_CS;

COMMIT;
```

qs_cbadm.sql

```
Rem
Rem $Header: qs_cbadm.sql 26-feb-2001.16:50:50 ahunold Exp $
Rem
Rem qs_cbadm.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      qs_cbadm.sql - Customer Billing Administration schema
Rem
Rem    DESCRIPTION
Rem      The QS_CBADM schema belongs to the Queued Shipping
Rem      (QS) schema group of the Oracle9i Sample Schemas
Rem
Rem    NOTES
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     02/26/01 - Merged ahunold_qs_filenames
Rem    ahunold     02/26/01 - Created
Rem
```

```
REM =========================================================
REM create queue tables and queues
REM =========================================================
BEGIN
  dbms_aqadm.create_queue_table(
        queue_table => 'QS_CBADM_orders_sqtab',
        comment =>
          'Customer Billing Single Consumer Orders queue table',
        queue_payload_type => 'QS_ADM.order_typ',
        compatible => '8.1');
  dbms_aqadm.create_queue_table(
        queue_table => 'QS_CBADM_orders_mqtab',
        comment =>
          'Customer Billing Multi Consumer Service queue table',
        multiple_consumers => TRUE,
        queue_payload_type => 'QS_ADM.order_typ',
        compatible => '8.1');
  dbms_aqadm.create_queue (
        queue_name              => 'QS_CBADM_shippedorders_q',
        queue_table             => 'QS_CBADM_orders_sqtab');

END;
/

REM =========================================================
REM Grant dequeue privilege on the shopoeped orders queue to the
Customer Billing
Rem application.  The QS_CB application retrieves shipped orders
(not billed yet)
Rem from the shopoeped orders queue.
BEGIN
  dbms_aqadm.grant_queue_privilege(
    'DEQUEUE',
    'QS_CBADM_shippedorders_q',
    'QS_CB',
    FALSE);
END;
/

BEGIN
  dbms_aqadm.create_queue (
        queue_name              => 'QS_CBADM_billedorders_q',
        queue_table             => 'QS_CBADM_orders_mqtab');
END;
/
```

```
REM =====================================================
REM Grant enqueue privilege on the billed orders queue to Customer
Billing
Rem application.  The QS_CB application is allowed to put billed
orders into
Rem this queue.
BEGIN
  dbms_aqadm.grant_queue_privilege(
    'ENQUEUE',
    'QS_CBADM_billedorders_q',
    'QS_CB',
    FALSE);
END;
/

DECLARE
  subscriber     sys.aq$_agent;
BEGIN
  /* Subscribe to the BILLING billed orders queue */
  subscriber := sys.aq$_agent(
    'BILLED_ORDER',
    'QS_CS.QS_CS_billedorders_que',
    null);
  dbms_aqadm.add_subscriber(
    queue_name => 'QS_CBADM.QS_CBADM_billedorders_q',
    subscriber => subscriber);
END;
/

COMMIT;
```

qs_cre.sql

```
Rem
Rem $Header: qs_cre.sql 01-feb-2001.15:13:21 ahunold Exp $
Rem
Rem qs_cre.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      qs_cre.sql - Schema creation
Rem
```

```
Rem    DESCRIPTION
Rem      QS is the Queued Shipping schema of the Oracle 9i Sample
Rem        Schemas
Rem
Rem    NOTES
Rem
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     02/05/01 - Created
Rem

REM =========================================================
REM Create queue tables, queues for QS
REM =========================================================
BEGIN
  dbms_aqadm.create_queue_table(
        queue_table => 'QS_orders_sqtab',
        comment => 'Order Entry Single Consumer Orders queue table',
        queue_payload_type => 'QS_ADM.order_typ',
        message_grouping => DBMS_AQADM.TRANSACTIONAL,
        compatible => '8.1',
        primary_instance => 1,
        secondary_instance => 2);
END;
/

REM =========================================================
REM Create a priority queue table for QS
REM =========================================================
BEGIN
dbms_aqadm.create_queue_table(
        queue_table => 'QS_orders_pr_mqtab',
        sort_list =>'priority,enq_time',
        comment => 'Order Entry Priority MultiConsumer Orders queue
table',
        multiple_consumers => TRUE,
        queue_payload_type => 'QS_ADM.order_typ',
        compatible => '8.1',
        primary_instance => 2,
        secondary_instance => 1);
END;
/
REM =========================================================
REM Create a queue for New Orders for QS
REM =========================================================
```

```
BEGIN
dbms_aqadm.create_queue (
        queue_name              => 'QS_neworders_que',
        queue_table             => 'QS_orders_sqtab');
END;
/

REM =====================================================
REM Create a queue for booked orders for QS
REM =====================================================
BEGIN
dbms_aqadm.create_queue (
        queue_name              => 'QS_bookedorders_que',
        queue_table             => 'QS_orders_pr_mqtab');
END;
/

REM =====================================================
REM   Create the multiconsumer nonpersistent queue in QS schema
REM   This queue is used by the conenction dispatcher to count
REM   the number of user connections to the QS application
REM =====================================================
BEGIN
dbms_aqadm.create_np_queue(queue_name => 'LOGON_LOGOFF', multiple_
consumers => TRUE);
END;
/
```

qs_cs.sql

```
Rem
Rem $Header: qs_cs.sql 26-feb-2001.16:50:50 ahunold Exp $
Rem
Rem qs_cs.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      qs_cs.sql - Creates Customer Service Shipping schema
Rem
Rem    DESCRIPTION
Rem      The QS_CS schema belongs to the Queued Shipping
Rem      (QS) schema group of the Oracle9i Sample Schemas
```

```
Rem
Rem    NOTES
Rem     Customer support tracks the state of the customer request
Rem     in the system.
Rem     At any point, customer request can be in one of the
following states
Rem     A. BOOKED  B. SHIPPED  C. BACKED   D. BILLED
Rem     Given the order number the customer support will return the
state
Rem     the order is in. This state is maintained in the order_
status_table
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     02/26/01 - Merged ahunold_qs_filenames
Rem    ahunold     02/26/01 - Created from qs_cs_cre.sql
Rem

CREATE TABLE Order_Status_Table(customer_order       qs_adm.order_
typ,
                                status               varchar2(30));

Rem Create queue tables, queues for Customer Service

begin
dbms_aqadm.create_queue_table(
        queue_table => 'QS_CS_order_status_qt',
        comment => 'Customer Status multi consumer queue table',
        multiple_consumers => TRUE,
        queue_payload_type => 'QS_ADM.order_typ',
        compatible => '8.1');

dbms_aqadm.create_queue (
        queue_name              => 'QS_CS_bookedorders_que',
        queue_table             => 'QS_CS_order_status_qt');

dbms_aqadm.create_queue (
        queue_name              => 'QS_CS_backorders_que',
        queue_table             => 'QS_CS_order_status_qt');

dbms_aqadm.create_queue (
        queue_name              => 'QS_CS_shippedorders_que',
        queue_table             => 'QS_CS_order_status_qt');

dbms_aqadm.create_queue (
        queue_name              => 'QS_CS_billedorders_que',
```

```
        queue_table             => 'QS_CS_order_status_qt');

end;
/
```

qs_drop.sql

```
Rem
Rem $Header: qs_drop.sql 01-feb-2001.15:13:21 ahunold Exp $
Rem
Rem qs_drop.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      qs_drop.sql - Cleanup and drop all QS related schemas
Rem
Rem    DESCRIPTION
Rem      QS is the Queued Shipping schema of the Oracle 9i Sample
Rem        Schemas
Rem
Rem    NOTES
Rem
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     02/05/01 - Created
Rem

set echo on;
set serveroutput on;

CONNECT QS_ADM/&password_QS_ADM
execute dbms_aqadm.stop_queue(queue_name => 'QS.QS_neworders_que');
execute dbms_aqadm.stop_queue(queue_name => 'QS.QS_bookedorders_
que');
execute dbms_aqadm.stop_queue(queue_name => 'QS.logon_logoff');
execute dbms_aqadm.stop_queue(queue_name => 'QS_WS.QS_WS_
bookedorders_que');
execute dbms_aqadm.stop_queue(queue_name => 'QS_WS.QS_WS_
shippedorders_que');
execute dbms_aqadm.stop_queue(queue_name => 'QS_WS.QS_WS_backorders_
que');
execute dbms_aqadm.stop_queue(queue_name => 'QS_ES.QS_ES_
```

```
bookedorders_que');
execute dbms_aqadm.stop_queue(queue_name => 'QS_ES.QS_ES_
shippedorders_que');
execute dbms_aqadm.stop_queue(queue_name => 'QS_ES.QS_ES_backorders_
que');
execute dbms_aqadm.stop_queue(queue_name => 'QS_OS.QS_OS_
bookedorders_que');
execute dbms_aqadm.stop_queue(queue_name => 'QS_OS.QS_OS_
shippedorders_que');
execute dbms_aqadm.stop_queue(queue_name => 'QS_OS.QS_OS_backorders_
que');
execute dbms_aqadm.stop_queue(queue_name => 'QS_CS.QS_CS_
bookedorders_que');
execute dbms_aqadm.stop_queue(queue_name => 'QS_CS.QS_CS_backorders_
que');
execute dbms_aqadm.stop_queue(queue_name => 'QS_CS.QS_CS_
shippedorders_que');
execute dbms_aqadm.stop_queue(queue_name => 'QS_CS.QS_CS_
billedorders_que');

Rem Drop queue tables, queues for QS
Rem
CONNECT QS/&password_QS
begin
dbms_aqadm.drop_queue (
        queue_name              => 'QS_neworders_que');
end;
/

begin
dbms_aqadm.drop_queue (
        queue_name              => 'QS_bookedorders_que');
end;
/

begin
dbms_aqadm.drop_queue_table(
        queue_table => 'QS_orders_sqtab');
end;
/

Rem Create a priority queue table for QS
begin
dbms_aqadm.drop_queue_table(
        queue_table => 'QS_orders_pr_mqtab');
```

```
end;
/

CONNECT QS/&password_QS

Rem
Rem   Drop the multiconsumer nonpersistent queue in QS schema
Rem   This queue is used by the conenction dispatcher to count
Rem   the number of user connections to the QS application

execute dbms_aqadm.drop_queue(queue_name => 'LOGON_LOGOFF');

Rem Drop queue tables, queues for QS_WS Shipping
CONNECT QS_WS/&password_QS_WS

Rem Booked orders are stored in the priority queue table
begin
dbms_aqadm.drop_queue (
        queue_name              => 'QS_WS_bookedorders_que');
end;
/

Rem Shipped orders and back orders are stored in the FIFO queue
table
begin
dbms_aqadm.drop_queue (
        queue_name              => 'QS_WS_shippedorders_que');
end;
/

begin
dbms_aqadm.drop_queue (
        queue_name              => 'QS_WS_backorders_que');
end;
/

Rem Drop queue table for QS_WS shipping
begin
dbms_aqadm.drop_queue_table(
        queue_table => 'QS_WS_orders_pr_mqtab');
end;
/

Rem Drop queue tables for QS_WS shipping
begin
```

```
dbms_aqadm.drop_queue_table(
        queue_table => 'QS_WS_orders_mqtab');
end;
/

Rem Drop queue tables, queues for QS_ES Shipping
CONNECT QS_ES/&password_QS_ES

Rem Booked orders are stored in the priority queue table
begin
dbms_aqadm.drop_queue (
        queue_name              => 'QS_ES_bookedorders_que');
end;
/

Rem Shipped orders and back orders are stored in the FIFO queue
table
begin
dbms_aqadm.drop_queue (
        queue_name              => 'QS_ES_shippedorders_que');
end;
/

begin
dbms_aqadm.drop_queue (
        queue_name              => 'QS_ES_backorders_que');
end;
/

Rem Drop queue table for QS_ES shipping
begin
dbms_aqadm.drop_queue_table(
        queue_table => 'QS_ES_orders_mqtab');
end;
/

Rem Drop FIFO queue tables for QS_ES shipping
begin
dbms_aqadm.drop_queue_table(
        queue_table => 'QS_ES_orders_pr_mqtab');
end;
/


Rem Drop queue tables, queues for Overseas Shipping
```

```
CONNECT QS_OS/&password_QS_OS


Rem Booked orders are stored in the priority queue table
begin
dbms_aqadm.drop_queue (
        queue_name              => 'QS_OS_bookedorders_que');
end;
/

Rem Shipped orders and back orders are stored in the FIFO queue
table
begin
dbms_aqadm.drop_queue (
        queue_name              => 'QS_OS_shippedorders_que');
end;
/

begin
dbms_aqadm.drop_queue (
        queue_name              => 'QS_OS_backorders_que');
end;
/

Rem Create a priority queue table for QS_OS shipping
begin
dbms_aqadm.drop_queue_table(
        queue_table => 'QS_OS_orders_pr_mqtab');
end;
/

Rem Create a FIFO queue tables for QS_OS shipping
begin
dbms_aqadm.drop_queue_table(
        queue_table => 'QS_OS_orders_mqtab');
end;
/

Rem Create queue tables, queues for Customer Billing
CONNECT QS_CBADM/&password_QS_CBADM

begin
dbms_aqadm.drop_queue (
        queue_name              => 'QS_CBADM_shippedorders_q');
```

```
end;
/

begin
dbms_aqadm.drop_queue (
        queue_name              => 'QS_CBADM_billedorders_q');
end;
/

begin
dbms_aqadm.drop_queue_table(
        queue_table => 'QS_CBADM_orders_sqtab', force => true);

dbms_aqadm.drop_queue_table(
        queue_table => 'QS_CBADM_orders_mqtab', force => true);

end;
/

CONNECT QS_CS/&password_QS_CS

DROP TABLE Order_Status_Table;

Rem Drop queue tables, queues for Customer Service

begin
dbms_aqadm.drop_queue (
        queue_name              => 'QS_CS_bookedorders_que');

dbms_aqadm.drop_queue (
        queue_name              => 'QS_CS_backorders_que');

dbms_aqadm.drop_queue (
        queue_name              => 'QS_CS_shippedorders_que');

dbms_aqadm.drop_queue (
        queue_name              => 'QS_CS_billedorders_que');

end;
/

begin
dbms_aqadm.drop_queue_table(
        queue_table => 'QS_CS_order_status_qt', force => true);
end;
```

```
/

CONNECT QS_ADM/&password_QS_ADM

Rem drop objects types

drop type order_typ;
drop type orderitemlist_vartyp;
drop type orderitem_typ;
drop type customer_typ;


Rem drop queue admin account and individual accounts for each
application
Rem
CONNECT system/&password_SYSTEM
set serveroutput on;
set echo on;

drop user QS_ADM cascade;
drop user QS cascade;
drop user QS_WS cascade;
drop user QS_ES cascade;
drop user QS_OS cascade;
drop user QS_CBADM cascade;
drop user QS_CB cascade;
drop user QS_CS cascade;
```

qs_es.sql

```
Rem
Rem $Header: qs_es.sql 26-feb-2001.16:50:50 ahunold Exp $
Rem
Rem qs_es.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      qs_es.sql - Creates Eastern Shipping schema
Rem
Rem    DESCRIPTION
Rem      The QS_ES schema belongs to the Queued Shipping
Rem      (QS) schema group of the Oracle9i Sample Schemas
```

```
Rem
Rem    NOTES
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     02/26/01 - Merged ahunold_qs_filenames
Rem    ahunold     02/26/01 - Created from qs_es_cre.sql
Rem

REM ========================================================
REM Create a priority queue table for QS_ES shipping
REM ========================================================
BEGIN
  dbms_aqadm.create_queue_table(
        queue_table => 'QS_ES_orders_mqtab',
        comment =>
'East Shipping Multi Consumer Orders queue table',
        multiple_consumers => TRUE,
        queue_payload_type => 'QS_ADM.order_typ',
        compatible => '8.1');
END;
/

REM ========================================================
REM Create a FIFO queue tables for QS_ES shipping
REM ========================================================
BEGIN
   dbms_aqadm.create_queue_table(
        queue_table => 'QS_ES_orders_pr_mqtab',
        sort_list =>'priority,enq_time',
        comment =>
'East Shipping Priority Multi Consumer Orders queue table',
        multiple_consumers => TRUE,
        queue_payload_type => 'QS_ADM.order_typ',
        compatible => '8.1');
END;
/

REM ========================================================
REM Booked orders are stored in the priority queue table
REM ========================================================
BEGIN
   dbms_aqadm.create_queue (
        queue_name              => 'QS_ES_bookedorders_que',
        queue_table             => 'QS_ES_orders_pr_mqtab');
END;
```

```
/

REM =======================================================
REM Shipped orders and back orders are stored in the FIFO
REM queue table
REM =======================================================
BEGIN
   dbms_aqadm.create_queue (
        queue_name              => 'QS_ES_shippedorders_que',
        queue_table             => 'QS_ES_orders_mqtab');
END;
/

BEGIN
   dbms_aqadm.create_queue (
        queue_name              => 'QS_ES_backorders_que',
        queue_table             => 'QS_ES_orders_mqtab');
END;
/

COMMIT;
```

qs_main.sql

```
Rem
Rem $Header: qs_main.sql 13-apr-2001.12:14:38 ahunold Exp $
Rem
Rem qs_main.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      qs_main.sql - Main schema creation script
Rem
Rem    DESCRIPTION
Rem      QS is the Queued Shipping schema of the Oracle 9i Sample
Rem        Schemas
Rem
Rem    NOTES
Rem      Run as SYS or SYSTEM
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     04/13/01 - spool, additional parameter
```

```
Rem    ahunold     03/12/01 - prompts
Rem    ahunold     02/26/01 - 8 char filenames
Rem    ahunold     02/05/01 - Created
Rem

SET ECHO ON

ALTER SESSION SET NLS_LANGUAGE=American;

PROMPT
PROMPT specify one password for the users QS,QS_ADM,QS_CBADM,
PROMPT QS_WS,QS_ES,QS_OS,QS_CS and QS_CB as parameter 1:
define pass     = &1
PROMPT
PROMPT specify default tablespeace for QS as parameter 2:
define tbs      = &2
PROMPT
PROMPT specify temporary tablespace for QS as parameter 3:
define ttbs     = &3
PROMPT
PROMPT specify password for SYSTEM as parameter 4:
define master_pass = &4
PROMPT
PROMPT specify password for OE as parameter 5:
define passoe   = &5
PROMPT
PROMPT specify password for SYS as parameter 6:
define pass_sys = &6
PROMPT
PROMPT specify log directory path as parameter 7:
define log_path = &7
PROMPT

spool &log_path.qs_main.log

REM =========================================================
REM cleanup section
REM =========================================================

DROP USER qs_adm CASCADE;
DROP USER qs CASCADE;
DROP USER qs_ws CASCADE;
DROP USER qs_es CASCADE;
DROP USER qs_os CASCADE;
DROP USER qs_cbadm CASCADE;
```

```
DROP USER qs_cb CASCADE;
DROP USER qs_cs CASCADE;

REM =========================================================
REM Start job_queue_processes to handle AQ propagation
REM =========================================================

alter system set job_queue_processes=4;

REM =========================================================
REM CREATE USERs
REM The user is assigned tablespaces and quota in separate
REM ALTER USER statements so that the CREATE USER statement
REM will succeed even if the &tbs and temp tablespaces do
REM not exist.
REM =========================================================

REM =========================================================
REM Create a common admin account for all Queued Shipping
REM applications
REM =========================================================

CREATE USER qs_adm IDENTIFIED BY &pass;
ALTER USER qs_adm DEFAULT TABLESPACE &tbs QUOTA UNLIMITED ON &tbs;
ALTER USER qs_adm TEMPORARY TABLESPACE &ttbs;

REM ALTER USER qs_adm DEFAULT TABLESPACE &tbs QUOTA ON &tbs
UNLIMITED;
REM ALTER USER qs_adm TEMPORARY TABLESPACE &ttbs;

GRANT ALTER SESSION TO qs_adm;
GRANT CREATE CLUSTER TO qs_adm;
GRANT CREATE DATABASE LINK TO qs_adm;
GRANT CREATE SEQUENCE TO qs_adm;
GRANT CREATE SESSION TO qs_adm;
GRANT CREATE SYNONYM TO qs_adm;
GRANT CREATE TABLE TO qs_adm;
GRANT CREATE VIEW TO qs_adm;
GRANT CREATE CLUSTER TO qs_adm;
GRANT CREATE INDEXTYPE TO qs_adm;
GRANT CREATE OPERATOR TO qs_adm;
GRANT CREATE PROCEDURE TO qs_adm;
GRANT CREATE SEQUENCE TO qs_adm;
GRANT CREATE TABLE TO qs_adm;
GRANT CREATE TRIGGER TO qs_adm;
```

```
GRANT CREATE TYPE TO qs_adm;
GRANT aq_administrator_role TO qs_adm;
GRANT EXECUTE ON dbms_aq TO qs_adm;
GRANT EXECUTE ON dbms_aqadm TO qs_adm;

REM  ================================================
REM  connected as sys to grant execute on dbms_lock
REM  and connected again as system
REM  ================================================

CONNECT sys/&pass_sys AS SYSDBA;
GRANT execute ON sys.dbms_stats TO qs_adm;
GRANT execute ON dbms_lock to qs_adm;

CONNECT system/&master_pass

execute dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','qs_
adm',FALSE);
execute dbms_aqadm.grant_system_privilege('DEQUEUE_ANY','qs_
adm',FALSE);

REM =======================================================
REM Create the application schemas and grant appropriate
REM permission to all schemas
REM =======================================================

REM =======================================================
REM Create Queued Shipping schemas
REM =======================================================

CREATE USER qs IDENTIFIED BY &pass;
ALTER USER qs DEFAULT TABLESPACE &tbs QUOTA UNLIMITED ON &tbs;

ALTER USER qs TEMPORARY TABLESPACE &ttbs;

GRANT ALTER SESSION TO qs;
GRANT CREATE CLUSTER TO qs;
GRANT CREATE DATABASE LINK TO qs;
GRANT CREATE SEQUENCE TO qs;
GRANT CREATE SESSION TO qs;
GRANT CREATE SYNONYM TO qs;
GRANT CREATE TABLE TO qs;
GRANT CREATE VIEW TO qs;
GRANT CREATE CLUSTER TO qs;
GRANT CREATE INDEXTYPE TO qs;
```

```
GRANT CREATE OPERATOR TO qs;
GRANT CREATE PROCEDURE TO qs;
GRANT CREATE SEQUENCE TO qs;
GRANT CREATE TABLE TO qs;
GRANT CREATE TRIGGER TO qs;
GRANT CREATE TYPE TO qs;
GRANT EXECUTE ON dbms_aq to qs;
GRANT EXECUTE ON dbms_aqadm to qs;

REM =======================================================
REM Create an account for Western Region Shipping
REM =======================================================

CREATE USER qs_ws IDENTIFIED BY &pass;
ALTER USER qs_ws DEFAULT TABLESPACE &tbs QUOTA UNLIMITED ON &tbs;

ALTER USER qs_ws TEMPORARY TABLESPACE &ttbs;


GRANT ALTER SESSION TO qs_ws;
GRANT CREATE CLUSTER TO qs_ws;
GRANT CREATE DATABASE LINK TO qs_ws;
GRANT CREATE SEQUENCE TO qs_ws;
GRANT CREATE SESSION TO qs_ws;
GRANT CREATE SYNONYM TO qs_ws;
GRANT CREATE TABLE TO qs_ws;
GRANT CREATE VIEW TO qs_ws;
GRANT CREATE CLUSTER TO qs_ws;
GRANT CREATE INDEXTYPE TO qs_ws;
GRANT CREATE OPERATOR TO qs_ws;
GRANT CREATE PROCEDURE TO qs_ws;
GRANT CREATE SEQUENCE TO qs_ws;
GRANT CREATE TABLE TO qs_ws;
GRANT CREATE TRIGGER TO qs_ws;
GRANT CREATE TYPE TO qs_ws;
GRANT EXECUTE ON dbms_aq to qs_ws;
GRANT EXECUTE ON dbms_aqadm to qs_ws;

REM =======================================================
REM Create an account for Eastern Region Shipping
REM =======================================================

CREATE USER qs_es IDENTIFIED BY &pass;
ALTER USER qs_es DEFAULT TABLESPACE &tbs QUOTA UNLIMITED ON &tbs;
```

```
ALTER USER qs_es TEMPORARY TABLESPACE &ttbs;


GRANT ALTER SESSION TO qs_es;
GRANT CREATE CLUSTER TO qs_es;
GRANT CREATE DATABASE LINK TO qs_es;
GRANT CREATE SEQUENCE TO qs_es;
GRANT CREATE SESSION TO qs_es;
GRANT CREATE SYNONYM TO qs_es;
GRANT CREATE TABLE TO qs_es;
GRANT CREATE VIEW TO qs_es;
GRANT CREATE CLUSTER TO qs_es;
GRANT CREATE INDEXTYPE TO qs_es;
GRANT CREATE OPERATOR TO qs_es;
GRANT CREATE PROCEDURE TO qs_es;
GRANT CREATE SEQUENCE TO qs_es;
GRANT CREATE TABLE TO qs_es;
GRANT CREATE TRIGGER TO qs_es;
GRANT CREATE TYPE TO qs_es;
GRANT EXECUTE ON dbms_aq TO qs_es;
GRANT EXECUTE ON dbms_aqadm TO qs_es;

REM =======================================================
REM Create an account for Overseas Shipping
REM =======================================================

CREATE USER qs_os IDENTIFIED BY &pass;
ALTER USER qs_os DEFAULT TABLESPACE &tbs QUOTA UNLIMITED ON &tbs;

ALTER USER qs_os TEMPORARY TABLESPACE &ttbs;


GRANT ALTER SESSION TO qs_os;
GRANT CREATE CLUSTER TO qs_os;
GRANT CREATE DATABASE LINK TO qs_os;
GRANT CREATE SEQUENCE TO qs_os;
GRANT CREATE SESSION TO qs_os;
GRANT CREATE SYNONYM TO qs_os;
GRANT CREATE TABLE TO qs_os;
GRANT CREATE VIEW TO qs_os;
GRANT CREATE CLUSTER TO qs_os;
GRANT CREATE INDEXTYPE TO qs_os;
GRANT CREATE OPERATOR TO qs_os;
GRANT CREATE PROCEDURE TO qs_os;
GRANT CREATE SEQUENCE TO qs_os;
```

```
GRANT CREATE TABLE TO qs_os;
GRANT CREATE TRIGGER TO qs_os;
GRANT CREATE TYPE TO qs_os;
GRANT EXECUTE ON dbms_aq TO qs_os;
GRANT EXECUTE ON dbms_aqadm TO qs_os;

REM =======================================================
REM Customer Billing, for security reason, has an admin
REM schema that hosts all the queue tables and an
REM application schema from where the application runs.
REM =======================================================

CREATE USER qs_cbadm IDENTIFIED BY &pass;
ALTER USER qs_cbadm DEFAULT TABLESPACE &tbs QUOTA UNLIMITED ON &tbs;

ALTER USER qs_cbadm TEMPORARY TABLESPACE &ttbs;


GRANT ALTER SESSION TO qs_cbadm;
GRANT CREATE CLUSTER TO qs_cbadm;
GRANT CREATE DATABASE LINK TO qs_cbadm;
GRANT CREATE SEQUENCE TO qs_cbadm;
GRANT CREATE SESSION TO qs_cbadm;
GRANT CREATE SYNONYM TO qs_cbadm;
GRANT CREATE TABLE TO qs_cbadm;
GRANT CREATE VIEW TO qs_cbadm;
GRANT CREATE CLUSTER TO qs_cbadm;
GRANT CREATE INDEXTYPE TO qs_cbadm;
GRANT CREATE OPERATOR TO qs_cbadm;
GRANT CREATE PROCEDURE TO qs_cbadm;
GRANT CREATE SEQUENCE TO qs_cbadm;
GRANT CREATE TABLE TO qs_cbadm;
GRANT CREATE TRIGGER TO qs_cbadm;
GRANT CREATE TYPE TO qs_cbadm;
GRANT EXECUTE ON dbms_aq to qs_cbadm;
GRANT EXECUTE ON dbms_aqadm to qs_cbadm;

REM =======================================================
REM Create an account for Customer Billing
REM =======================================================

CREATE USER qs_cb IDENTIFIED BY &pass;
ALTER USER qs_cb DEFAULT TABLESPACE &tbs QUOTA UNLIMITED ON &tbs;

ALTER USER qs_cb TEMPORARY TABLESPACE &ttbs;
```

```
GRANT ALTER SESSION TO qs_cb;
GRANT CREATE CLUSTER TO qs_cb;
GRANT CREATE DATABASE LINK TO qs_cb;
GRANT CREATE SEQUENCE TO qs_cb;
GRANT CREATE SESSION TO qs_cb;
GRANT CREATE SYNONYM TO qs_cb;
GRANT CREATE TABLE TO qs_cb;
GRANT CREATE VIEW TO qs_cb;
GRANT CREATE CLUSTER TO qs_cb;
GRANT CREATE INDEXTYPE TO qs_cb;
GRANT CREATE OPERATOR TO qs_cb;
GRANT CREATE PROCEDURE TO qs_cb;
GRANT CREATE SEQUENCE TO qs_cb;
GRANT CREATE TABLE TO qs_cb;
GRANT CREATE TRIGGER TO qs_cb;
GRANT CREATE TYPE TO qs_cb;
GRANT EXECUTE ON dbms_aq TO qs_cb;
GRANT EXECUTE ON dbms_aqadm TO qs_cb;

REM =======================================================
REM Create an account for Customer Service
REM =======================================================

CREATE USER qs_cs IDENTIFIED BY &pass;
ALTER USER qs_cs DEFAULT TABLESPACE &tbs QUOTA UNLIMITED ON &tbs;

ALTER USER qs_cs TEMPORARY TABLESPACE &ttbs;


GRANT ALTER SESSION TO qs_cs;
GRANT CREATE CLUSTER TO qs_cs;
GRANT CREATE DATABASE LINK TO qs_cs;
GRANT CREATE SEQUENCE TO qs_cs;
GRANT CREATE SESSION TO qs_cs;
GRANT CREATE SYNONYM TO qs_cs;
GRANT CREATE TABLE TO qs_cs;
GRANT CREATE VIEW TO qs_cs;
GRANT CREATE CLUSTER TO qs_cs;
GRANT CREATE INDEXTYPE TO qs_cs;
GRANT CREATE OPERATOR TO qs_cs;
GRANT CREATE PROCEDURE TO qs_cs;
GRANT CREATE SEQUENCE TO qs_cs;
GRANT CREATE TABLE TO qs_cs;
```

```
GRANT CREATE TRIGGER TO qs_cs;
GRANT CREATE TYPE TO qs_cs;
GRANT EXECUTE ON dbms_aq TO qs_cs;
GRANT EXECUTE ON dbms_aqadm TO qs_cs;


REM =====================================================
REM Create objects
REM =====================================================

REM =====================================================
REM grants from oe schema to user qs_adm
REM =====================================================

CONNECT oe/&passoe
GRANT REFERENCES, SELECT ON customers TO qs_adm;
GRANT REFERENCES, SELECT ON product_information TO qs_adm;

PROMPT calling qs_adm.sql ...
CONNECT qs_adm/&pass
@@qs_adm

PROMPT calling qs_cre.sql ...
CONNECT qs/&pass;
@@qs_cre

PROMPT calling qs_es.sql ...
CONNECT qs_es/&pass
@@qs_es

PROMPT calling qs_ws.sql ...
CONNECT qs_ws/&pass
@@qs_ws

PROMPT calling qs_os.sql ...
CONNECT qs_os/&pass
@@qs_os

PROMPT calling qs_cbadm.sql ...
CONNECT qs_cbadm/&pass
@@qs_cbadm

PROMPT calling qs_cs.sql ...
CONNECT qs_cs/&pass
@@qs_cs
```

```
PROMPT calling qs_run.sql ...
CONNECT qs_adm/&pass
@@qs_run

spool off
```

qs_os.sql

```
Rem
Rem $Header: qs_os.sql 26-feb-2001.16:50:51 ahunold Exp $
Rem
Rem qs_os.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      qs_os.sql - Creates Overseas Shipping schema
Rem
Rem    DESCRIPTION
Rem      The QS_OS schema belongs to the Queued Shipping
Rem      (QS) schema group of the Oracle9i Sample Schemas
Rem
Rem    NOTES
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     02/26/01 - Merged ahunold_qs_filenames
Rem    ahunold     02/26/01 - Created from qs_os_cre.sql
Rem

REM ========================================================
REM Create a priority queue table for QS_OS shipping
REM ========================================================
BEGIN
  dbms_aqadm.create_queue_table(
        queue_table => 'QS_OS_orders_pr_mqtab',
        sort_list =>'priority,enq_time',
        comment =>
          'Overseas Shipping Priority MultiConsumer Orders queue
table',
        multiple_consumers => TRUE,
        queue_payload_type => 'QS_ADM.order_typ',
        compatible => '8.1');
END;
```

```
/

REM =====================================================
REM Create a FIFO queue tables for QS_OS shipping
REM =====================================================
BEGIN
  dbms_aqadm.create_queue_table(
        queue_table => 'QS_OS_orders_mqtab',
        comment =>
          'Overseas Shipping Multi Consumer Orders queue table',
        multiple_consumers => TRUE,
        queue_payload_type => 'QS_ADM.order_typ',
        compatible => '8.1');
END;
/

REM =====================================================
REM Booked orders are stored in the priority queue table
REM =====================================================
BEGIN
  dbms_aqadm.create_queue (
        queue_name              => 'QS_OS_bookedorders_que',
        queue_table             => 'QS_OS_orders_pr_mqtab');
END;
/

REM =====================================================
REM Shipped orders and back orders are stored in the FIFO queue
table
REM =====================================================
BEGIN
  dbms_aqadm.create_queue (
        queue_name              => 'QS_OS_shippedorders_que',
        queue_table             => 'QS_OS_orders_mqtab');
END;
/

BEGIN
  dbms_aqadm.create_queue (
        queue_name              => 'QS_OS_backorders_que',
        queue_table             => 'QS_OS_orders_mqtab');
END;
/

COMMIT;
```

qs_run.sql

```
Rem
Rem $Header: qs_run.sql 01-feb-2001.15:13:21 ahunold Exp $
Rem
Rem qs_run.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      qs_run.sql - Create the application
Rem
Rem    DESCRIPTION
Rem      QS is the Queued Shipping schema of the Oracle 9i Sample
Rem        Schemas
Rem
Rem    NOTES
Rem
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     02/05/01 - Created
Rem

CREATE OR REPLACE TYPE simpleorder_typ AS OBJECT (
        orderno        NUMBER,
statusVARCHAR2(30),
ordertypeVARCHAR2(30),
orderregionVARCHAR2(30),
paymentmethodVARCHAR2(30),
totalNUMBER);
/


CREATE OR REPLACE PACKAGE QS_Applications AS
    -- this procedure is called from the java front end to enqueue
    -- new orders
    PROCEDURE new_order_enq(simpleOrder IN simpleorder_typ,
    customer    IN customer_typ,
    items       IN orderitemlist_vartyp);

    PROCEDURE get_ship_notification(
                    orderid IN number,
```

```
                    status  OUT number,
                    tracking_id OUT varchar2);

    -- move new orders from qs_neworders_que to qs_bookedorders_que.
    -- sets the shipping region
    PROCEDURE qs_move_orders;

    -- Each shipping region calls this shipping_app by providing the
    -- name of the consumer as an IN Parameter. This application
movQS_ES
    -- the messages from bookedorder queuQS_ES to either
shippedorder queue
    -- or backedorder queue based on the inventory information
    PROCEDURE shipping_app( consumer IN VARCHAR2);


    -- Move shipped orders from the shipped order queue to the
billed
    -- order queue in the billing area
    PROCEDURE billing_app;

    PROCEDURE new_order_driver(QS_OStart IN NUMBER, QS_OStop IN
NUMBER);

END QS_Applications;
/
show errors


CREATE OR REPLACE PACKAGE BODY QS_Applications AS

PROCEDURE new_order_enq(simpleOrder IN simpleorder_typ,
    customer IN customer_typ,
    items IN orderitemlist_vartyp) IS

    qs_enq_order_data        qs_adm.order_typ;
    enqopt                   dbms_aq.enqueue_options_t;
    msgprop                  dbms_aq.message_properties_t;
    enq_msgid                raw(16);
    itemlist     orderitemlist_vartyp;
    item     orderitem_typ;

BEGIN

    -- form the book items object from items
```

```
    msgprop.correlation := simpleOrder.ordertype;

    qs_enq_order_data := qs_adm.order_typ(
simpleOrder.orderno,
simpleOrder.status,
simpleOrder.ordertype,
simpleOrder.orderregion,
customer,
simpleOrder.paymentmethod,
itemlist, simpleOrder.total);

    -- put the shipping priority into the message property
    -- before enqueuing the message.
    if (simpleOrder.ordertype = 'RUSH') then
msgprop.priority := 1;
    else
msgprop.priority := 2;
    end if;


    dbms_aq.enqueue('qs.qs_neworders_que', enqopt, msgprop,
    qs_enq_order_data, enq_msgid);

--    dbms_output.put_line('One order enqueued successfully');
    commit;


END new_order_enq;

PROCEDURE get_ship_notification(
                     orderid IN number,
                     status  OUT number,
                     tracking_id OUT varchar2) IS
   deqopt           dbms_aq.dequeue_options_t;
   mprop            dbms_aq.message_properties_t;
   deq_msgid        RAW(16);
   msg_data         RAW(80);
   no_messages      exception;
   pragma exception_init (no_messages, -25228);

BEGIN
   status := 0;

   deqopt.navigation := dbms_aq.FIRST_messAGE;
```

```
   deqopt.wait := 10;
   deqopt.correlation := to_char(orderid);
   deqopt.consumer_name := 'ORDER_ENTRY';


   BEGIN
     dbms_aq.dequeue(
                queue_name => 'qs.qs_notification_que',
                dequeue_options => deqopt,
                message_properties => mprop,
                payload => msg_data,
                msgid => deq_msgid);
     status := 1;
     tracking_id := rawtohex(deq_msgid);
     commit;
   EXCEPTION
     WHEN no_messages THEN
        status := 0;
     WHEN OTHERS THEN
        RAISE;
   END;

END get_ship_notification;


PROCEDURE qs_move_orders IS

    new_orders      BOOLEAN := TRUE;
    dopt                    dbms_aq.dequeue_options_t;
    enqopt                  dbms_aq.enqueue_options_t;
    mprop                   dbms_aq.message_properties_t;
    qs_deq_order_data       qs_adm.order_typ;
    deq_msgid               RAW(16);
    no_messages             exception;
    pragma exception_init   (no_messages, -25228);

begin

    --dopt.wait := DBMS_AQ.NO_WAIT;
    dopt.navigation := dbms_aq.FIRST_messAGE;

    --while (new_orders) LOOP
    LOOP
BEGIN
    dbms_aq.dequeue(
```

```
queue_name => 'qs.qs_neworders_que',
dequeue_options => dopt,
message_properties => mprop,
payload => qs_deq_order_data,
msgid => deq_msgid);

    -- assign the shipping region
    if (qs_deq_order_data.customer.country NOT IN ('USA', 'US'))
then
--dbms_output.put_line('International shipment ... ');
qs_deq_order_data.orderregion := 'INTERNATIONAL';
    else
if (qs_deq_order_data.customer.state IN ('TX', 'Texas',
    'CA', 'California',
    'NV', 'Nevada')) then

    qs_deq_order_data.orderregion := 'WESTERN';
--dbms_output.put_line('western shipment');
else
    qs_deq_order_data.orderregion := 'EASTERN';
--dbms_output.put_line('eastern shipment');
end if;
--dbms_output.put_line('Dequeuing a message ...');
--dbms_output.put_line('Region is ' || qs_deq_order_
data.orderregion);
    end if;


            -- change the order status to booked
            qs_deq_order_data.status := 'BOOKED';

    -- enqueue into booked orders queue
    dbms_aq.enqueue(
queue_name => 'qs.qs_bookedorders_que',
enqueue_options => enqopt,
message_properties => mprop,
payload => qs_deq_order_data,
msgid => deq_msgid);

    commit;

    --        dopt.navigation := dbms_aq.NEXT_messAGE;
EXCEPTION
            WHEN no_messages THEN
                 new_orders := FALSE;
```

```
END;
    END LOOP;


END qs_move_orders;

PROCEDURE billing_app IS
    new_orders              BOOLEAN := TRUE;
    dopt                    dbms_aq.dequeue_options_t;
    enqopt                  dbms_aq.enqueue_options_t;
    mprop                   dbms_aq.message_properties_t;
    deq_order_data        qs_adm.order_typ;
    deq_msgid               RAW(16);
    no_messages             exception;
    pragma exception_init   (no_messages, -25228);

begin

    --dopt.wait := DBMS_AQ.NO_WAIT;
    dopt.navigation := dbms_aq.FIRST_messAGE;
    dopt.consumer_name := 'CUSTOMER_BILLING';

    --while (new_orders) LOOP
    LOOP
        BEGIN
            dbms_aq.dequeue(
                queue_name => 'QS_CBADM.QS_CBADM_shippedorders_que',
                dequeue_options => dopt,
                message_properties => mprop,
                payload => deq_order_data,
                msgid => deq_msgid);


            -- change the order status to billed
            deq_order_data.status := 'BILLED';


            -- enqueue into booked orders queue
            dbms_aq.enqueue(
                queue_name => 'QS_CBADM.QS_CBADM_billedorders_que',
                enqueue_options => enqopt,
                message_properties => mprop,
                payload => deq_order_data,
                msgid => deq_msgid);
```

```
            commit;

    --        dopt.navigation := dbms_aq.NEXT_messAGE;
        EXCEPTION
            WHEN no_messages THEN
                 new_orders := FALSE;
        END;
    END LOOP;

END billing_app;



PROCEDURE shipping_app( consumer IN VARCHAR2) IS

    deq_msgid               RAW(16);
    dopt                    dbms_aq.dequeue_options_t;
    enqopt                  dbms_aq.enqueue_options_t;
    mprop                   dbms_aq.message_properties_t;
    deq_order_data          qs_adm.order_typ;
    qname                   varchar2(30);
    shipqname               varchar2(30);
    bookqname               varchar2(30);
    backqname               varchar2(30);
    notqname                varchar2(30);
    no_messages             exception;
    pragma exception_init   (no_messages, -25228);
    new_orders      BOOLEAN := TRUE;
    is_backed      BOOLEAN := FALSE;
    is_avail     int;
    region                  varchar2(30);
    notify                  BOOLEAN := FALSE;
    not_enqopt              dbms_aq.enqueue_options_t;
    not_mprop               dbms_aq.message_properties_t;
    not_msg_data            RAW(80);
    not_msgid               RAW(16);
    ship_orderno            number;

begin


    dopt.consumer_name := consumer;
    --dopt.wait := DBMS_AQ.NO_WAIT;
    dopt.navigation := dbms_aq.FIRST_messAGE;
```

```
    IF (consumer = 'West_Shipping') THEN
    qname     := 'QS_WS.QS_WS_bookedorders_que';
    shipqname := 'QS_WS.QS_WS_shippedorders_que';
    backqname := 'QS_WS.QS_WS_backorders_que';
            notqname  := 'QS_WS.QS_WS_notification_que';
            region := 'WESTERN';
            notify := TRUE;
    ELSIF (consumer = 'East_Shipping') THEN
    qname     := 'QS_ES.QS_ES_bookedorders_que';
    shipqname := 'QS_ES.QS_ES_shippedorders_que';
    backqname := 'QS_ES.QS_ES_backorders_que';
            notqname  := 'QS_ES.QS_ES_notification_que';
            region := 'EASTERN';
            notify := TRUE;
    ELSE
    qname     := 'QS_OS.QS_OS_bookedorders_que';
    shipqname := 'QS_OS.QS_OS_shippedorders_que';
    backqname := 'QS_OS.QS_OS_backorders_que';
            region := 'INTERNATIONAL';
    END IF;

    --WHILE (new_orders) LOOP
    LOOP
BEGIN
            is_backed := FALSE;
    dbms_aq.dequeue(
queue_name => qname,
dequeue_options => dopt,
message_properties => mprop,
payload => deq_order_data,
msgid => deq_msgid);

            ship_orderno := deq_order_data.orderno;
            IF (notify = TRUE) THEN

              not_mprop.correlation := TO_CHAR(ship_orderno);
              not_msg_data := hextoraw(to_char(ship_orderno));
        dbms_aq.enqueue(
   queue_name => notqname,
   enqueue_options => not_enqopt,
   message_properties => not_mprop,
   payload => not_msg_data,
   msgid => not_msgid);

            END IF;
```

```
            deq_order_data.orderregion := region;

    -- check if all books in an order are availabe

    if (is_backed = FALSE) then
-- change the status of the order to SHIPPED order
deq_order_data.status := 'SHIPPED';
dbms_aq.enqueue(
    queue_name => shipqname,
    enqueue_options => enqopt,
    message_properties => mprop,
    payload => deq_order_data,
    msgid => deq_msgid);
    end if;

    commit;
EXCEPTION
    WHEN no_messages THEN
new_orders := FALSE;
END;
    END LOOP;
END shipping_app;

PROCEDURE new_order_driver(QS_OStart IN NUMBER, QS_OStop IN NUMBER)
IS
    neworder                simpleorder_typ;
    customer                customer_typ;
    item                    orderitem_typ;
    items     orderitemlist_vartyp;
    itc                     number;
    iid                     number;
    iprice                  number;
    iquantity               number;
    ordertype               varchar2(30);
    order_date              date;
    custno                  number;
    custid                  number;
    custname                varchar2(100);
    cstreet                  varchar2(100);
    ccity                    varchar2(30);
    cstate                   varchar2(2);
    czip                     number;
    ccountry                 varchar2(100);
```

```
BEGIN

    for i in QS_OStart .. QS_OStop loop

        if ((i MOD 20) = 0) THEN
          ordertype := 'RUSH';
        ELSE
          ordertype := 'NORMAL';
        end if;

--        neworder.paymentmethod := 'MASTERCARD';

        select to_char(sysdate) into order_date from sys.dual;

        custid := i MOD 10;

        select cust_first_name, c.cust_address.street_address,
c.cust_address.city, c.cust_address.state_province, c.cust_
address.postal_code, c.cust_address.country_id into
               custname, cstreet, ccity, cstate,
               czip, ccountry
        from oe.customers c where customer_id = custid;

select product_id, list_price into iid, iprice from oe.product_
information where product_id = i;

item := orderitem_typ (1, 1, iprice, iid, iprice);
items(1) := item;
customer := Customer_typ(custid, custname, cstreet, ccity, cstate,
                   czip, ccountry);
neworder := simpleorder_typ(i, NULL, ordertype, NULL, 'MASTERCARD',
iprice);
        new_order_enq(neworder, customer, items);
        dbms_output.put_line('order processed' || neworder.orderno);

dbms_lock.sleep(10 - (i MOD 10));
    end loop;
END new_order_driver;


END QS_Applications;
/
show errors

grant execute on QS_Applications to qs;
```

```
grant execute on QS_Applications to QS_WS;
grant execute on QS_Applications to QS_ES;
grant execute on QS_Applications to QS_OS;
grant execute on QS_Applications to QS_CB;
grant execute on QS_Applications to QS_CBADM;
```

qs_ws.sql

```
Rem
Rem $Header: qs_ws.sql 26-feb-2001.16:50:51 ahunold Exp $
Rem
Rem qs_ws.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      qs_ws.sql - Creates Western Shipping schema
Rem
Rem    DESCRIPTION
Rem      The QS_WS schema belongs to the Queued Shipping
Rem      (QS) schema group of the Oracle9i Sample Schemas
Rem
Rem    NOTES
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold     02/26/01 - Merged ahunold_qs_filenames
Rem    ahunold     02/26/01 - Created from qs_ws_cre.sql
Rem

REM ========================================================
REM Create a priority queue table for QS_WS shipping
REM ========================================================
BEGIN
  dbms_aqadm.create_queue_table(
        queue_table => 'QS_WS_orders_pr_mqtab',
        sort_list =>'priority,enq_time',
        comment => 'West Shipping Priority MultiConsumer Orders
queue table',
        multiple_consumers => TRUE,
        queue_payload_type => 'QS_ADM.order_typ',
        compatible => '8.1');
END;
/
```

```
REM =====================================================
REM Create a FIFO queue tables for QS_WS shipping
REM =====================================================
BEGIN
  dbms_aqadm.create_queue_table(
        queue_table => 'QS_WS_orders_mqtab',
        comment => 'West Shipping Multi Consumer Orders queue
table',
        multiple_consumers => TRUE,
        queue_payload_type => 'QS_ADM.order_typ',
        compatible => '8.1');
END;
/

REM =====================================================
REM Booked orders are stored in the priority queue table
REM =====================================================
BEGIN
  dbms_aqadm.create_queue (
        queue_name              => 'QS_WS_bookedorders_que',
        queue_table             => 'QS_WS_orders_pr_mqtab');
END;
/

REM =====================================================
REM Shipped orders and back orders are stored in the FIFO
REM queue table
REM =====================================================
BEGIN
  dbms_aqadm.create_queue (
        queue_name              => 'QS_WS_shippedorders_que',
        queue_table             => 'QS_WS_orders_mqtab');
END;
/

BEGIN
dbms_aqadm.create_queue (
        queue_name              => 'QS_WS_backorders_que',
        queue_table             => 'QS_WS_orders_mqtab');
END;
/

REM =====================================================
REM In order to test history, set retention to 1 DAY for
REM the queues in QS_WS
```

```
REM =========================================================

BEGIN
  dbms_aqadm.alter_queue(
         queue_name => 'QS_WS_bookedorders_que',
         retention_time => 86400);
END;
/

BEGIN
  dbms_aqadm.alter_queue(
         queue_name => 'QS_WS_shippedorders_que',
         retention_time => 86400);
END;
/

BEGIN
  dbms_aqadm.alter_queue(
         queue_name => 'QS_WS_backorders_que',
         retention_time => 86400);
END;
/

REM =========================================================
REM Create the queue subscribers
REM =========================================================
DECLARE
  subscriber     sys.aq$_agent;
BEGIN
  /* Subscribe to the QS_WS back orders queue */
  subscriber := sys.aq$_agent(
         'BACK_ORDER',
         'QS_CS.QS_CS_backorders_que',
         null);
  dbms_aqadm.add_subscriber(
         queue_name => 'QS_WS.QS_WS_backorders_que',
         subscriber => subscriber);
END;
/

DECLARE
  subscriber     sys.aq$_agent;
BEGIN
  /* Subscribe to the QS_WS shipped orders queue */
  subscriber := sys.aq$_agent(
```

```
         'SHIPPED_ORDER',
         'QS_CS.QS_CS_shippedorders_que',
         null);
  dbms_aqadm.add_subscriber(
         queue_name => 'QS_WS.QS_WS_shippedorders_que',
         subscriber => subscriber);
END;
/

COMMIT;
```

Sales History (SH) Scripts

This section shows the SH scripts in alphabetical order.

sh_analz.sql

```
Rem
Rem $Header: sh_analz.sql 01-feb-2001.15:13:21 ahunold Exp $
Rem
Rem sh_analz.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      sh_analz.sql - Gather statistics for SH schema
Rem
Rem    DESCRIPTION
Rem      SH is the Sales History schema of the Oracle 9i Sample
Rem   Schemas
Rem
Rem    NOTES
Rem
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    hbaer      01/29/01 - Created
Rem

EXECUTE dbms_stats.gather_schema_stats(-
'SH',-
granularity => 'ALL',-
cascade => TRUE,-
```

```
estimate_percent => 20,-
block_sample => TRUE);
```

sh_comnt.sql

```
Rem
Rem $Header: sh_comnt.sql 01-feb-2001.15:13:21 ahunold Exp $
Rem
Rem sh_comnt.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      sh_comnt.sql - Populates the countries and channel table
Rem
Rem    DESCRIPTION
Rem      SH is the Sales History schema of the Oracle 9i Sample
Rem   Schemas
Rem
Rem    NOTES
Rem
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    hbaer      01/29/01 - Created
Rem

COMMENT ON TABLE times
IS 'Time dimension table to support multiple hierarchies and
materialized views';

COMMENT ON COLUMN times.time_id
IS 'primary key; day date, finest granularity, CORRECT ORDER';

COMMENT ON COLUMN times.day_name
IS 'Monday to Sunday, repeating';

COMMENT ON COLUMN times.day_number_in_week
IS '1 to 7, repeating';

COMMENT ON COLUMN times.day_number_in_month
IS '1 to 31, repeating';

COMMENT ON COLUMN times.calendar_week_number
```

```
IS '1 to 53, repeating';

COMMENT ON COLUMN times.fiscal_week_number
IS '1 to 53, repeating';

COMMENT ON COLUMN times.week_ending_day
IS 'date of last day in week, CORRECT ORDER';

COMMENT ON COLUMN times.calendar_month_number
IS '1 to 12, repeating';

COMMENT ON COLUMN times.fiscal_month_number
IS '1 to 12, repeating';

COMMENT ON COLUMN times.calendar_month_desc
IS 'e.g. 1998-01, CORRECT ORDER';

COMMENT ON COLUMN times.fiscal_month_desc
IS 'e.g. 1998-01, CORRECT ORDER';

COMMENT ON COLUMN times.calendar_month_name
IS 'January to December, repeating';

COMMENT ON COLUMN times.fiscal_month_name
IS 'January to December, repeating';

COMMENT ON COLUMN times.calendar_quarter_desc
IS 'e.g. 1998-Q1, CORRECT ORDER';

COMMENT ON COLUMN times.fiscal_quarter_desc
IS 'e.g. 1999-Q3, CORRECT ORDER';

COMMENT ON COLUMN times.calendar_quarter_number
IS '1 to 4, repeating';

COMMENT ON COLUMN times.fiscal_quarter_number
IS '1 to 4, repeating';

COMMENT ON COLUMN times.calendar_year
IS 'e.g. 1999, CORRECT ORDER';

COMMENT ON COLUMN times.fiscal_year
IS 'e.g. 1999, CORRECT ORDER';

COMMENT ON COLUMN times.days_in_cal_month
```

```
IS 'e.g. 28,31, repeating';

COMMENT ON COLUMN times.days_in_fis_month
IS 'e.g. 25,32, repeating';

COMMENT ON COLUMN times.days_in_cal_quarter
IS 'e.g. 88,90, repeating';

COMMENT ON COLUMN times.days_in_fis_quarter
IS 'e.g. 88,90, repeating';

COMMENT ON COLUMN times.days_in_cal_year
IS '365,366 repeating';

COMMENT ON COLUMN times.days_in_fis_year
IS 'e.g. 355,364, repeating';

COMMENT ON COLUMN times.end_of_cal_month
IS 'last day of calendar month';

COMMENT ON COLUMN times.end_of_fis_month
IS 'last day of fiscal month';

COMMENT ON COLUMN times.end_of_cal_quarter
IS 'last day of calendar quarter';

COMMENT ON COLUMN times.end_of_fis_quarter
IS 'last day of fiscal quarter';

COMMENT ON COLUMN times.end_of_cal_year
IS 'last day of cal year';

COMMENT ON COLUMN times.end_of_fis_year
IS 'last day of fiscal year';

rem ============================================

COMMENT ON TABLE channels
IS 'small dimension table';

COMMENT ON COLUMN channels.channel_id
IS 'primary key column';

COMMENT ON COLUMN channels.channel_desc
IS 'e.g. telesales, internet, catalog';
```

```
COMMENT ON COLUMN channels.channel_class
IS 'e.g. direct, indirect';

rem ============================================

COMMENT ON TABLE promotions
IS 'dimension table without a PK-FK relationship with the facts
table, to show outer join functionality';

COMMENT ON COLUMN promotions.promo_id
IS 'primary key column';

COMMENT ON COLUMN promotions.promo_name
IS 'promotion description';

COMMENT ON COLUMN promotions.promo_subcategory
IS 'enables to investigate promotion hierarchies';

COMMENT ON COLUMN promotions.promo_category
IS 'promotion category';

COMMENT ON COLUMN promotions.promo_cost
IS 'promotion cost, to do promotion effect calculations';

COMMENT ON COLUMN promotions.promo_begin_date
IS 'promotion begin day';

COMMENT ON COLUMN promotions.promo_end_date
IS 'promotion end day';

rem ============================================

COMMENT ON TABLE countries
IS 'country dimension table (snowflake)';

COMMENT ON COLUMN countries.country_id
IS 'primary key';

COMMENT ON COLUMN countries.country_name
IS 'country name';

COMMENT ON COLUMN countries.country_subregion
IS 'e.g. Western Europe, to allow hierarchies';
```

```
COMMENT ON COLUMN countries.country_region
IS 'e.g. Europe, Asia';

rem ============================================

COMMENT ON TABLE products
IS 'dimension table';

COMMENT ON COLUMN products.prod_id
IS 'primary key';

COMMENT ON COLUMN products.prod_name
IS 'product name';

COMMENT ON COLUMN products.prod_desc
IS 'product description';

COMMENT ON COLUMN products.prod_subcategory
IS 'product subcategory';

COMMENT ON COLUMN products.prod_subcat_desc
IS 'product subcategory description';

COMMENT ON COLUMN products.prod_category
IS 'product category';

COMMENT ON COLUMN products.prod_cat_desc
IS 'product category description';

COMMENT ON COLUMN products.prod_weight_class
IS 'product weight class';

COMMENT ON COLUMN products.prod_unit_of_measure
IS 'product unit of measure';

COMMENT ON COLUMN products.prod_pack_size
IS 'product package size';

COMMENT ON COLUMN products.supplier_id
IS 'this column';

COMMENT ON COLUMN products.prod_status
IS 'product status';

COMMENT ON COLUMN products.prod_list_price
```

```
IS 'product list price';

COMMENT ON COLUMN products.prod_min_price
IS 'product minimum price';

rem ============================================

COMMENT ON TABLE customers
IS 'dimension table';

COMMENT ON COLUMN customers.cust_id
IS 'primary key';

COMMENT ON COLUMN customers.cust_first_name
IS 'first name of the customer';

COMMENT ON COLUMN customers.cust_last_name
IS 'last name of the customer';

COMMENT ON COLUMN customers.cust_gender
IS 'gender; low cardinality attribute';

COMMENT ON COLUMN customers.cust_year_of_birth
IS 'customer year of birth';

COMMENT ON COLUMN customers.cust_marital_status
IS 'customer marital status; low cardinality attribute';

COMMENT ON COLUMN customers.cust_street_address
IS 'customer street address';

COMMENT ON COLUMN customers.cust_postal_code
IS 'postal code of the customer';

COMMENT ON COLUMN customers.cust_city
IS 'city where the customer lives';

COMMENT ON COLUMN customers.cust_state_province
IS 'customer geography: state or province';

COMMENT ON COLUMN customers.cust_main_phone_number
IS 'customer main phone number';

COMMENT ON COLUMN customers.cust_income_level
IS 'customer income level';
```

```
COMMENT ON COLUMN customers.cust_credit_limit
IS 'customer credit limit';

COMMENT ON COLUMN customers.cust_email
IS 'customer email id';

COMMENT ON COLUMN customers.country_id
IS 'foreign key to the countries table (snowflake)';

rem ============================================

COMMENT ON TABLE sales
IS 'facts table, without a primary key; all rows are uniquely
identified by the combination of all foreign keys';

COMMENT ON COLUMN sales.prod_id
IS 'FK to the products dimension table';

COMMENT ON COLUMN sales.cust_id
IS 'FK to the customers dimension table';

COMMENT ON COLUMN sales.time_id
IS 'FK to the times dimension table';

COMMENT ON COLUMN sales.channel_id
IS 'FK to the channels dimension table';

COMMENT ON COLUMN sales.promo_id
IS 'promotion identifier, without FK constraint (intentionally) to
show outer join optimization';

COMMENT ON COLUMN sales.quantity_sold
IS 'product quantity sold with the transaction';

COMMENT ON COLUMN sales.amount_sold
IS 'invoiced amount to the customer';
```

sh_cons.sql

```
Rem
Rem $Header: sh_cons.sql 01-feb-2001.15:13:21 ahunold Exp $
Rem
Rem sh_cons.sql
```

```
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      sh_cons.sql - Define constraints
Rem
Rem    DESCRIPTION
Rem      SH is the Sales History schema of the Oracle 9i Sample
Rem   Schemas
Rem
Rem    NOTES
Rem
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    hbaer      01/29/01 - Created
Rem

ALTER TABLE sales MODIFY CONSTRAINT sales_product_fk  ENABLE
NOVALIDATE;
ALTER TABLE sales MODIFY CONSTRAINT sales_customer_fk ENABLE
NOVALIDATE;
ALTER TABLE sales MODIFY CONSTRAINT sales_time_fk     ENABLE
NOVALIDATE;
ALTER TABLE sales MODIFY CONSTRAINT sales_channel_fk  ENABLE
NOVALIDATE;
ALTER TABLE sales MODIFY CONSTRAINT sales_promo_fk    ENABLE
NOVALIDATE;
ALTER TABLE costs MODIFY CONSTRAINT costs_time_fk     ENABLE
NOVALIDATE;
ALTER TABLE costs MODIFY CONSTRAINT costs_product_fk  ENABLE
NOVALIDATE;
```

sh_cre.sql

```
Rem
Rem $Header: sh_cre.sql 01-feb-2001.15:13:21 ahunold Exp $
Rem
Rem sh_cre.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      sh_cre.sql - Create database objects
```

```
Rem
Rem    DESCRIPTION
Rem      SH is the Sales History schema of the Oracle 9i Sample
Rem   Schemas
Rem
Rem    NOTES
Rem      Prerequisite: Enterprise Edition with Partitioning Option
Rem      installed
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    hbaer      01/29/01 - Created
Rem

REM TABLE TIMES attribute definitions and examples
REM since most of the attributes are CHARACTER values, a correct
time based
REM order CANNOT be guaranteed for all of them. The ones were this
is guaranteed
REM are marked accordingly
REM for correct time based ordering the VARCHAR2() attributes have
to be converted
REM with the appropriate TO_DATE() function
REM     time_id                     /* day date, finest granularity,
CORRECT ORDER */
REM     day_name                    /* Monday to Sunday, repeating
*/
REM     day_number_in_week          /* 1 to 7, repeating */
REM     day_number_in_month         /* 1 to 31, repeating */
REM     calendar_week_number        /* 1 to 53, repeating */
REM     fiscal_week_number          /* 1 to 53, repeating */
REM     week_ending_day             /* date of last day in week,
CORRECT ORDER */
REM     calendar_month_number       /* 1 to 12, repeating */
REM     fiscal_month_number         /* 1 to 12, repeating */
REM     calendar_month_desc         /* e.g. 1998-01, CORRECT ORDER
*/
REM     fiscal_month_desc           /* e.g. 1998-01, CORRECT ORDER
*/
REM     calendar_month_name         /* January to December,
repeating */
REM     fiscal_month_name           /* January to December,
repeating */
REM     calendar_quarter_desc       /* e.g. 1998-Q1, CORRECT ORDER
*/
REM     fiscal_quarter_desc         /* e.g. 1999-Q3, CORRECT ORDER
```

```
*/
REM     calendar_quarter_number     /* 1 to 4, repeating */
REM     fiscal_quarter_number       /* 1 to 4, repeating */
REM     calendar_year               /* e.g. 1999, CORRECT ORDER */
REM     fiscal_year                 /* e.g. 1999, CORRECT ORDER */
REM     days_in_cal_month           /* e.g. 28,31, repeating */
REM     days_in_fis_month           /* e.g. 25,32, repeating */
REM     days_in_cal_quarter         /* e.g. 88,90, repeating */
REM     days_in_fis_quarter         /* e.g. 88,90, repeating */
REM     days_in_cal_year            /* 365,366 repeating */
REM     days_in_fis_year            /* e.g. 355,364, repeating */
REM     end_of_cal_month            /* last day of cal month */
REM     end_of_fis_month            /* last day of fiscal month */
REM     end_of_cal_quarte           /* last day of cal quarter */
REM     end_of_fis_quarter          /* last day of fiscal quarter */
REM     end_of_cal_year             /* last day of cal year */
REM     end_of_fis_year             /* last day of fiscal year */

REM creation of dimension table TIMES ...
CREATE TABLE times
    (
      time_id                 DATE
    , day_name                VARCHAR2(9)
      CONSTRAINT       tim_day_name_nn           NOT NULL
    , day_number_in_week      NUMBER(1)
      CONSTRAINT       tim_day_in_week_nn        NOT NULL
    , day_number_in_month     NUMBER(2)
      CONSTRAINT       tim_day_in_month_nn       NOT NULL
    , calendar_week_number    NUMBER(2)
      CONSTRAINT       tim_cal_week_nn           NOT NULL
    , fiscal_week_number      NUMBER(2)
      CONSTRAINT       tim_fis_week_nn           NOT NULL
    , week_ending_day         DATE
      CONSTRAINT       tim_week_ending_day_nn    NOT NULL
    , calendar_month_number   NUMBER(2)
      CONSTRAINT       tim_cal_month_number_nn   NOT NULL
    , fiscal_month_number     NUMBER(2)
      CONSTRAINT       tim_fis_month_number_nn   NOT NULL
    , calendar_month_desc     VARCHAR2(8)
      CONSTRAINT       tim_cal_month_desc_nn     NOT NULL
    , fiscal_month_desc       VARCHAR2(8)
      CONSTRAINT       tim_fis_month_desc_nn     NOT NULL
    , days_in_cal_month       NUMBER
      CONSTRAINT       tim_days_cal_month_nn     NOT NULL
    , days_in_fis_month       NUMBER
```

```
      CONSTRAINT      tim_days_fis_month_nn     NOT NULL
    , end_of_cal_month       DATE
      CONSTRAINT      tim_end_of_cal_month_nn   NOT NULL
    , end_of_fis_month       DATE
      CONSTRAINT      tim_end_of_fis_month_nn   NOT NULL
    , calendar_month_name     VARCHAR2(9)
      CONSTRAINT       tim_cal_month_name_nn     NOT NULL
    , fiscal_month_name       VARCHAR2(9)
      CONSTRAINT       tim_fis_month_name_nn     NOT NULL
    , calendar_quarter_desc   CHAR(7)
      CONSTRAINT       tim_cal_quarter_desc_nn   NOT NULL
    , fiscal_quarter_desc     CHAR(7)
      CONSTRAINT            tim_fis_quarter_desc_nn   NOT NULL
    , days_in_cal_quarter     NUMBER
      CONSTRAINT            tim_days_cal_quarter_nn   NOT NULL
    , days_in_fis_quarter     NUMBER
      CONSTRAINT            tim_days_fis_quarter_nn   NOT NULL
    , end_of_cal_quarter      DATE
      CONSTRAINT      tim_end_of_cal_quarter_nn NOT NULL
    , end_of_fis_quarter      DATE
      CONSTRAINT      tim_end_of_fis_quarter_nn NOT NULL
    , calendar_quarter_number NUMBER(1)
      CONSTRAINT            tim_cal_quarter_number_nn NOT NULL
    , fiscal_quarter_number   NUMBER(1)
      CONSTRAINT            tim_fis_quarter_number_nn NOT NULL
    , calendar_year           NUMBER(4)
      CONSTRAINT            tim_cal_year_nn           NOT NULL
    , fiscal_year             NUMBER(4)
      CONSTRAINT            tim_fis_year_nn           NOT NULL
    , days_in_cal_year        NUMBER
      CONSTRAINT            tim_days_cal_year_nn      NOT NULL
    , days_in_fis_year        NUMBER
      CONSTRAINT            tim_days_fis_year_nn      NOT NULL
    , end_of_cal_year         DATE
      CONSTRAINT      tim_end_of_cal_year_nn    NOT NULL
    , end_of_fis_year         DATE
      CONSTRAINT      tim_end_of_fis_year_nn    NOT NULL
    )
PCTFREE 5;

CREATE UNIQUE INDEX time_pk
ON times (time_id) ;

ALTER TABLE times
ADD ( CONSTRAINT time_pk
```

```
      PRIMARY KEY (time_id) RELY ENABLE VALIDATE
    ) ;

REM creation of dimension table CHANNELS ...
CREATE TABLE channels
    ( channel_id       CHAR(1)
    , channel_desc     VARCHAR2(20)
        CONSTRAINT     chan_desc_nn NOT NULL
    , channel_class    VARCHAR2(20)
    )
PCTFREE 5;

CREATE UNIQUE INDEX chan_pk
ON channels (channel_id) ;

ALTER TABLE channels
ADD ( CONSTRAINT chan_pk
      PRIMARY KEY (channel_id) RELY ENABLE VALIDATE
    ) ;

REM creation of dimension table PROMOTIONS ...
CREATE TABLE promotions
    ( promo_id           NUMBER(6)
    , promo_name         VARCHAR2(20)
        CONSTRAINT  promo_name_nn       NOT NULL
    , promo_subcategory  VARCHAR2(30)
        CONSTRAINT  promo_subcat_nn     NOT NULL
    , promo_category     VARCHAR2(30)
        CONSTRAINT  promo_cat_nn        NOT NULL
    , promo_cost         NUMBER(10,2)
        CONSTRAINT  promo_cost_nn       NOT NULL
    , promo_begin_date   DATE
        CONSTRAINT  promo_begin_date_nn NOT NULL
    , promo_end_date     DATE
        CONSTRAINT  promo_end_date_nn   NOT NULL
    )
PCTFREE 5;


CREATE UNIQUE INDEX promo_pk
ON promotions (promo_id) ;

ALTER TABLE promotions
ADD ( CONSTRAINT promo_pk
      PRIMARY KEY (promo_id) RELY ENABLE VALIDATE
```

```
    ) ;

REM creation of dimension table COUNTRIES ...
CREATE TABLE countries
    ( country_id        CHAR(2)
    , country_name      VARCHAR2(40)
        CONSTRAINT      country_country_name_nn NOT NULL
    , country_subregion VARCHAR2(30)
    , country_region    VARCHAR2(20)
    )
PCTFREE 5;

ALTER TABLE countries
ADD ( CONSTRAINT country_pk
      PRIMARY KEY (country_id) RELY ENABLE VALIDATE
    ) ;

REM creation of dimension table CUSTOMERS ...
CREATE TABLE customers
    ( cust_id                NUMBER
    , cust_first_name        VARCHAR2(20)
        CONSTRAINT           customer_fname_nn NOT NULL
    , cust_last_name         VARCHAR2(40)
        CONSTRAINT           customer_lname_nn NOT NULL
    , cust_gender     CHAR(1)
    , cust_year_of_birth     NUMBER(4)
    , cust_marital_status    VARCHAR2(20)
    , cust_street_address    VARCHAR2(40)
        CONSTRAINT           customer_st_addr_nn NOT NULL
    , cust_postal_code       VARCHAR2(10)
        CONSTRAINT           customer_pcode_nn NOT NULL
    , cust_city              VARCHAR2(30)
        CONSTRAINT           customer_city_nn NOT NULL
    , cust_state_province    VARCHAR2(40)
    , country_id             CHAR(2)
        CONSTRAINT           customer_country_id_nn NOT NULL
    , cust_main_phone_number VARCHAR2(25)
    , cust_income_level      VARCHAR2(30)
    , cust_credit_limit      NUMBER
    , cust_email             VARCHAR2(30)
    )
PCTFREE 5;

CREATE UNIQUE INDEX customers_pk
   ON customers (cust_id) ;
```

```
ALTER TABLE customers
ADD ( CONSTRAINT customers_pk
      PRIMARY KEY (cust_id) RELY ENABLE VALIDATE
    ) ;

ALTER TABLE customers
ADD ( CONSTRAINT customers_country_fk
      FOREIGN KEY (country_id) REFERENCES countries(country_id)
      RELY ENABLE VALIDATE);

REM creation of dimension table PRODUCTS ...
CREATE TABLE products
    ( prod_id              NUMBER(6)
    , prod_name            VARCHAR2(50)
CONSTRAINT    products_prod_name_nn NOT NULL
    , prod_desc            VARCHAR2(4000)
CONSTRAINT    products_prod_desc_nn NOT NULL
    , prod_subcategory     VARCHAR2(50)
CONSTRAINT   products_prod_subcat_nn NOT NULL
    , prod_subcat_desc     VARCHAR2(2000)
CONSTRAINT   products_prod_subcatd_nn NOT NULL
    , prod_category        VARCHAR2(50)
CONSTRAINT   products_prod_cat_nn NOT NULL
    , prod_cat_desc        VARCHAR2(2000)
CONSTRAINT   products_prod_catd_nn NOT NULL
    , prod_weight_class    NUMBER(2)
    , prod_unit_of_measure VARCHAR2(20)
    , prod_pack_size       VARCHAR2(30)
    , supplier_id          NUMBER(6)
    , prod_status          VARCHAR2(20)
CONSTRAINT   products_prod_stat_nn NOT NULL
    , prod_list_price      NUMBER(8,2)
CONSTRAINT   products_prod_list_price_nn NOT NULL
    , prod_min_price       NUMBER(8,2)
CONSTRAINT   products_prod_min_price_nn NOT NULL
     )
PCTFREE 5;

CREATE UNIQUE INDEX products_pk
   ON products (prod_id) ;

ALTER TABLE products
ADD ( CONSTRAINT products_pk
      PRIMARY KEY (prod_id) RELY ENABLE VALIDATE
```

```
    ) ;

REM creation of fact table SALES ...

CREATE TABLE sales
    ( prod_id        NUMBER(6)
        CONSTRAINT   sales_product_nn     NOT NULL
    , cust_id        NUMBER
        CONSTRAINT   sales_customer_nn    NOT NULL
    , time_id        DATE
        CONSTRAINT   sales_time_nn        NOT NULL
    , channel_id     CHAR(1)
        CONSTRAINT   sales_channel_nn     NOT NULL
    , promo_id       NUMBER(6)
        CONSTRAINT   sales_promo_nn       NOT NULL
    , quantity_sold  NUMBER(3)
        CONSTRAINT   sales_quantity_nn    NOT NULL
    , amount_sold         NUMBER(10,2)
        CONSTRAINT   sales_amount_nn      NOT NULL
    )PCTFREE 5 NOLOGGING
        PARTITION BY RANGE (time_id)
        (PARTITION SALES_Q1_1998 VALUES LESS THAN (TO_
DATE('01-APR-1998','DD-MON-YYYY')),
         PARTITION SALES_Q2_1998 VALUES LESS THAN (TO_
DATE('01-JUL-1998','DD-MON-YYYY')),
         PARTITION SALES_Q3_1998 VALUES LESS THAN (TO_
DATE('01-OCT-1998','DD-MON-YYYY')),
         PARTITION SALES_Q4_1998 VALUES LESS THAN (TO_
DATE('01-JAN-1999','DD-MON-YYYY')),
         PARTITION SALES_Q1_1999 VALUES LESS THAN (TO_
DATE('01-APR-1999','DD-MON-YYYY')),
         PARTITION SALES_Q2_1999 VALUES LESS THAN (TO_
DATE('01-JUL-1999','DD-MON-YYYY')),
         PARTITION SALES_Q3_1999 VALUES LESS THAN (TO_
DATE('01-OCT-1999','DD-MON-YYYY')),
         PARTITION SALES_Q4_1999 VALUES LESS THAN (TO_
DATE('01-JAN-2000','DD-MON-YYYY')),
         PARTITION SALES_Q1_2000 VALUES LESS THAN (TO_
DATE('01-APR-2000','DD-MON-YYYY')),
         PARTITION SALES_Q2_2000 VALUES LESS THAN (TO_
DATE('01-JUL-2000','DD-MON-YYYY')),
         PARTITION SALES_Q3_2000 VALUES LESS THAN (TO_
DATE('01-OCT-2000','DD-MON-YYYY')),
         PARTITION SALES_Q4_2000 VALUES LESS THAN (MAXVALUE))
;
```

```
REM creation of second fact table COSTS ...

CREATE TABLE costs
    ( prod_id        NUMBER(6)
        CONSTRAINT   costs_product_nn     NOT NULL
    , time_id        DATE
        CONSTRAINT   costs_time_nn        NOT NULL
    , unit_cost      NUMBER(10,2)
        CONSTRAINT   costs_unit_cost_nn    NOT NULL
    , unit_price     NUMBER(10,2)
        CONSTRAINT   costs_unit_price_nn      NOT NULL
    )PCTFREE 5 NOLOGGING
PARTITION BY RANGE (time_id)
(PARTITION COSTS_Q1_1998 VALUES LESS THAN (TO_
DATE('01-APR-1998','DD-MON-YYYY')),
 PARTITION COSTS_Q2_1998 VALUES LESS THAN (TO_
DATE('01-JUL-1998','DD-MON-YYYY')),
 PARTITION COSTS_Q3_1998 VALUES LESS THAN (TO_
DATE('01-OCT-1998','DD-MON-YYYY')),
 PARTITION COSTS_Q4_1998 VALUES LESS THAN (TO_
DATE('01-JAN-1999','DD-MON-YYYY')),
 PARTITION COSTS_Q1_1999 VALUES LESS THAN (TO_
DATE('01-APR-1999','DD-MON-YYYY')),
 PARTITION COSTS_Q2_1999 VALUES LESS THAN (TO_
DATE('01-JUL-1999','DD-MON-YYYY')),
 PARTITION COSTS_Q3_1999 VALUES LESS THAN (TO_
DATE('01-OCT-1999','DD-MON-YYYY')),
 PARTITION COSTS_Q4_1999 VALUES LESS THAN (TO_
DATE('01-JAN-2000','DD-MON-YYYY')),
 PARTITION COSTS_Q1_2000 VALUES LESS THAN (TO_
DATE('01-APR-2000','DD-MON-YYYY')),
 PARTITION COSTS_Q2_2000 VALUES LESS THAN (TO_
DATE('01-JUL-2000','DD-MON-YYYY')),
 PARTITION COSTS_Q3_2000 VALUES LESS THAN (TO_
DATE('01-OCT-2000','DD-MON-YYYY')),
 PARTITION COSTS_Q4_2000 VALUES LESS THAN (MAXVALUE))
;

REM establish foreign keys to ALL dimension tables

ALTER TABLE sales
ADD ( CONSTRAINT sales_product_fk
      FOREIGN KEY (prod_id)
      REFERENCES products RELY ENABLE VALIDATE
```

```
    , CONSTRAINT sales_customer_fk
      FOREIGN KEY (cust_id)
      REFERENCES customers RELY ENABLE VALIDATE
    , CONSTRAINT sales_time_fk
      FOREIGN KEY (time_id)
      REFERENCES times RELY ENABLE VALIDATE
    , CONSTRAINT sales_channel_fk
      FOREIGN KEY (channel_id)
      REFERENCES channels RELY ENABLE VALIDATE
    , CONSTRAINT sales_promo_fk
      FOREIGN KEY (promo_id)
      REFERENCES promotions RELY ENABLE VALIDATE
    ) ;

ALTER TABLE costs
ADD ( CONSTRAINT costs_product_fk
      FOREIGN KEY (prod_id)
      REFERENCES products RELY ENABLE VALIDATE
    , CONSTRAINT costs_time_fk
      FOREIGN KEY (time_id)
      REFERENCES times RELY ENABLE VALIDATE
    ) ;

COMMIT;
```

sh_cremv.sql

```
Rem
Rem $Header: sh_cremv.sql 01-feb-2001.15:13:21 ahunold Exp $
Rem
Rem sh_cremv.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      sh_cremv.sql - Create materialized views
Rem
Rem    DESCRIPTION
Rem      SH is the Sales History schema of the Oracle 9i Sample
Rem   Schemas
Rem
Rem    NOTES
```

```
Rem
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    hbaer      01/29/01 - Created
Rem    ahunold    03/05/01 - no DROPs needed, part of creation
script


Rem first materialized view; simple aggregate join MV
Rem equivalent to example 1 in MV chapter DWG, page 8-11

CREATE MATERIALIZED VIEW cal_month_sales_mv
PCTFREE 5
BUILD IMMEDIATE
REFRESH FORCE
ENABLE QUERY REWRITE
AS
SELECT   t.calendar_month_desc
,        sum(s.amount_sold) AS dollars
FROM     sales s
,        times t
WHERE    s.time_id = t.time_id
GROUP BY t.calendar_month_desc;


Rem more complex mv with additional key columns to join to other
dimensions;

CREATE MATERIALIZED VIEW fweek_pscat_sales_mv
PCTFREE 5
BUILD IMMEDIATE
REFRESH COMPLETE
ENABLE QUERY REWRITE
AS
SELECT   t.week_ending_day
,        p.prod_subcategory
,        sum(s.amount_sold) AS dollars
,        s.channel_id
,        s.promo_id
FROM     sales s
,        times t
,        products p
WHERE    s.time_id = t.time_id
AND      s.prod_id = p.prod_id
GROUP BY t.week_ending_day
```

```
,        p.prod_subcategory
,        s.channel_id
,        s.promo_id;

CREATE BITMAP INDEX FW_PSC_S_MV_SUBCAT_BIX
ON fweek_pscat_sales_mv(prod_subcategory);

CREATE BITMAP INDEX FW_PSC_S_MV_CHAN_BIX
ON fweek_pscat_sales_mv(channel_id);

CREATE BITMAP INDEX FW_PSC_S_MV_PROMO_BIX
ON fweek_pscat_sales_mv(promo_id);

CREATE BITMAP INDEX FW_PSC_S_MV_WD_BIX
ON fweek_pscat_sales_mv(week_ending_day);
```

sh_drop.sql

```
Rem
Rem $Header: sh_drop.sql 01-feb-2001.15:13:21 ahunold Exp $
Rem
Rem sh_drop.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      sh_drop.sql - Drop database objects
Rem
Rem    DESCRIPTION
Rem      SH is the Sales History schema of the Oracle 9i Sample
Rem   Schemas
Rem
Rem    NOTES
Rem
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem     hbaer      01/29/01 - Created
Rem

REM drop all tables of schema

DROP TABLE sales      CASCADE CONSTRAINTS ;
DROP TABLE costs      CASCADE CONSTRAINTS ;
DROP TABLE times      CASCADE CONSTRAINTS ;
```

```
DROP TABLE channels   CASCADE CONSTRAINTS ;
DROP TABLE promotions CASCADE CONSTRAINTS ;
DROP TABLE customers  CASCADE CONSTRAINTS ;
DROP TABLE countries  CASCADE CONSTRAINTS ;
DROP TABLE products   CASCADE CONSTRAINTS ;

REM automatically generated by dbms_olap package

DROP TABLE mview$_exceptions;

REM drop all dimensions

DROP DIMENSION customers_dim;
DROP DIMENSION times_dim;
DROP DIMENSION products_dim;
DROP DIMENSION promotions_dim;
DROP DIMENSION channels_dim;

REM drop materialized views

DROP MATERIALIZED VIEW cal_month_sales_mv;
DROP MATERIALIZED VIEW fweek_pscat_sales_mv;

COMMIT;
```

sh_hiera.sql

```
Rem
Rem $Header: sh_hiera.sql 01-feb-2001.15:13:21 ahunold Exp $
Rem
Rem sh_hiera.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      sh_hiera.sql - Create dimensions and hierarchies
Rem
Rem    DESCRIPTION
Rem      SH is the Sales History schema of the Oracle 9i Sample
Rem   Schemas
Rem
Rem    NOTES
Rem
```

```
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    hbaer      01/29/01 - Created
Rem

CREATE DIMENSION times_dim
   LEVEL day         IS TIMES.TIME_ID
   LEVEL month       IS TIMES.CALENDAR_MONTH_DESC
   LEVEL quarter     IS TIMES.CALENDAR_QUARTER_DESC
   LEVEL year        IS TIMES.CALENDAR_YEAR
   LEVEL fis_week    IS TIMES.WEEK_ENDING_DAY
   LEVEL fis_month   IS TIMES.FISCAL_MONTH_DESC
   LEVEL fis_quarter IS TIMES.FISCAL_QUARTER_DESC
   LEVEL fis_year    IS TIMES.FISCAL_YEAR
   HIERARCHY cal_rollup    (
             day     CHILD OF
             month   CHILD OF
             quarter CHILD OF
             year
   )
   HIERARCHY fis_rollup    (
             day         CHILD OF
             fis_week    CHILD OF
             fis_month CHILD OF
             fis_quarter CHILD OF
             fis_year
   )
   ATTRIBUTE day DETERMINES
(day_number_in_week, day_name, day_number_in_month,
         calendar_week_number)
   ATTRIBUTE month DETERMINES
(calendar_month_desc,
         calendar_month_number, calendar_month_name,
         days_in_cal_month, end_of_cal_month)
   ATTRIBUTE quarter DETERMINES
(calendar_quarter_desc,
         calendar_quarter_number,days_in_cal_quarter,
 end_of_cal_quarter)
   ATTRIBUTE year DETERMINES
(calendar_year,
         days_in_cal_year, end_of_cal_year)
   ATTRIBUTE fis_week DETERMINES
(week_ending_day,
         fiscal_week_number)
   ATTRIBUTE fis_month DETERMINES
```

```
(fiscal_month_desc, fiscal_month_number, fiscal_month_name,
 days_in_fis_month, end_of_fis_month)
   ATTRIBUTE fis_quarter DETERMINES
(fiscal_quarter_desc,
         fiscal_quarter_number, days_in_fis_quarter,
 end_of_fis_quarter)
   ATTRIBUTE fis_year DETERMINES
(fiscal_year,
         days_in_fis_year, end_of_fis_year)
;

execute dbms_olap.validate_dimension('times_dim','sh',false,true)
SELECT COUNT(*) FROM mview$_exceptions;

CREATE DIMENSION customers_dim
LEVEL customerIS (customers.cust_id)
LEVEL city IS (customers.cust_city)
LEVEL state IS (customers.cust_state_province)
LEVEL country IS (countries.country_id)
LEVEL subregion IS (countries.country_subregion)
LEVEL region IS (countries.country_region)
HIERARCHY geog_rollup (
customerCHILD OF
city CHILD OF
state CHILD OF
country CHILD OF
subregion CHILD OF
region
JOIN KEY (customers.country_id) REFERENCES country
)
ATTRIBUTE customer DETERMINES
(cust_first_name, cust_last_name, cust_gender,
 cust_marital_status, cust_year_of_birth,
 cust_income_level, cust_credit_limit,
         cust_street_address, cust_postal_code,
         cust_main_phone_number, cust_email)
        ATTRIBUTE city DETERMINES (cust_city)
        ATTRIBUTE state DETERMINES (cust_state_province)
ATTRIBUTE country DETERMINES (countries.country_name)
        ATTRIBUTE subregion DETERMINES (countries.country_subregion)
        ATTRIBUTE region DETERMINES (countries.country_region)
;

execute dbms_olap.validate_dimension('customers_
dim','sh',false,true)
```

```
SELECT COUNT(*) FROM mview$_exceptions;

CREATE DIMENSION products_dim
LEVEL product IS (products.prod_id)
 LEVEL subcategory IS (products.prod_subcategory)
LEVEL categoryIS (products.prod_category)
HIERARCHY prod_rollup (
productCHILD OF
subcategory CHILD OF
category
)
ATTRIBUTE product DETERMINES
        (products.prod_name, products.prod_desc,
         prod_weight_class, prod_unit_of_measure,
         prod_pack_size,prod_status, prod_list_price, prod_min_
price)
ATTRIBUTE subcategory DETERMINES
        (prod_subcategory, prod_subcat_desc)
ATTRIBUTE category DETERMINES
        (prod_category, prod_cat_desc)
;

execute dbms_olap.validate_dimension('products_dim','sh',false,true)
SELECT COUNT(*) FROM mview$_exceptions;

CREATE DIMENSION promotions_dim
LEVEL promo   IS (promotions.promo_id)
LEVEL subcategory IS (promotions.promo_subcategory)
LEVEL category   IS (promotions.promo_category)
HIERARCHY promo_rollup (
promo CHILD OF
subcategory CHILD OF
category
)
ATTRIBUTE promo DETERMINES
        (promo_name, promo_cost,
         promo_begin_date, promo_end_date)
        ATTRIBUTE subcategory DETERMINES (promo_subcategory)
        ATTRIBUTE category DETERMINES (promo_category)
;

execute dbms_olap.validate_dimension('promotions_
dim','sh',false,true)
SELECT COUNT(*) FROM mview$_exceptions;
```

```
CREATE DIMENSION channels_dim
LEVEL channel    IS (channels.channel_id)
LEVEL channel_class IS (channels.channel_class)
HIERARCHY channel_rollup (
channelCHILD OF
channel_class
)
        ATTRIBUTE channel DETERMINES (channel_desc)
        ATTRIBUTE channel_class DETERMINES (channel_class)
;

execute dbms_olap.validate_dimension('channels_dim','sh',false,true)
SELECT COUNT(*) FROM mview$_exceptions;

COMMIT;
```

sh_idx.sql

```
Rem
Rem $Header: sh_idx.sql 01-feb-2001.15:13:21 ahunold Exp $
Rem
Rem sh_idx.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      sh_idx.sql - Create database objects
Rem
Rem    DESCRIPTION
Rem      SH is the Sales History schema of the Oracle 9i Sample
Rem   Schemas
Rem
Rem    NOTES
Rem
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem     hbaer      01/29/01 - Created
Rem     ahunold    03/05/01 - no DROPs needed, part of creation
suite

REM some indexes on fact table SALES
```

```
CREATE BITMAP INDEX sales_prod_bix
       ON sales (prod_id)
       LOCAL NOLOGGING COMPUTE STATISTICS ;

CREATE BITMAP INDEX sales_cust_bix
       ON sales (cust_id)
       LOCAL NOLOGGING COMPUTE STATISTICS ;

CREATE BITMAP INDEX sales_time_bix
       ON sales (time_id)
       LOCAL NOLOGGING COMPUTE STATISTICS ;

CREATE BITMAP INDEX sales_channel_bix
       ON sales (channel_id)
       LOCAL NOLOGGING COMPUTE STATISTICS ;

CREATE BITMAP INDEX sales_promo_bix
       ON sales (promo_id)
       LOCAL NOLOGGING COMPUTE STATISTICS ;

REM some indexes on fact table COSTS

CREATE BITMAP INDEX costs_prod_bix
       ON costs (prod_id)
       LOCAL NOLOGGING COMPUTE STATISTICS ;

CREATE BITMAP INDEX costs_time_bix
       ON costs (time_id)
       LOCAL NOLOGGING COMPUTE STATISTICS ;

REM some indexes on dimension tables

CREATE BITMAP INDEX products_prod_status_bix
ON products(prod_status)
        NOLOGGING COMPUTE STATISTICS ;

CREATE INDEX products_prod_subcat_ix
ON products(prod_subcategory)
        NOLOGGING COMPUTE STATISTICS ;

CREATE INDEX products_prod_cat_ix
ON products(prod_category)
        NOLOGGING COMPUTE STATISTICS ;

CREATE BITMAP INDEX customers_gender_bix
```

```
ON customers(cust_gender)
        NOLOGGING COMPUTE STATISTICS ;

CREATE BITMAP INDEX customers_marital_bix
ON customers(cust_marital_status)
        NOLOGGING COMPUTE STATISTICS ;

CREATE BITMAP INDEX customers_yob_bix
ON customers(cust_year_of_birth)
        NOLOGGING COMPUTE STATISTICS ;

COMMIT;
```

sh_main.sql

```
Rem
Rem $Header: sh_main.sql 13-apr-2001.11:36:13 ahunold Exp $
Rem
Rem sh_main.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      sh_main.sql - Main schema creation and load script
Rem
Rem    DESCRIPTION
Rem      SH is the Sales History schema of the Oracle 9i Sample
Rem   Schemas
Rem
Rem    NOTES
Rem     CAUTION: use absolute pathnames as parameters 5 and 6.
Rem    Example (UNIX) echo $ORACLE_HOME/demo/schema/sales_history
Rem     Please make sure that parameters 5 and 6 are specified
Rem     INCLUDING the trailing directory delimiter, since the
Rem     directory parameters and the filenames are concatenated
Rem     without adding any delimiters.
Rem     Run this as SYS or SYSTEM
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem     ahunold    04/13/01 - spool, notes
Rem     ahunold    04/10/01 - flexible log and data paths
Rem     ahunold    03/28/01 - spool
Rem     ahunold    03/23/01 - absolute path names
Rem     ahunold    03/14/01 - prompts
```

```
Rem     ahunold    03/09/01 - privileges
Rem     hbaer      03/01/01 - changed loading from COSTS table from
Rem      SQL*Loader to external table with GROUP BY
Rem      Added also CREATE DIRECTORY privilege
Rem

SET ECHO ON

ALTER SESSION SET NLS_LANGUAGE=American;

PROMPT
PROMPT specify password for SH as parameter 1:
define pass     = &1
PROMPT
PROMPT specify default tablespeace for SH as parameter 2:
define tbs      = &2
PROMPT
PROMPT specify temporary tablespace for SH as parameter 3:
define ttbs     = &3
PROMPT
PROMPT specify password for SYS as parameter 4:
define pass_sys = &4
PROMPT
PROMPT specify directory path for the data files as parameter 5:
define data_dir = &5
PROMPT
PROMPT writeable directory path for the log files as parameter 6:
define log_dir = &6
PROMPT

ALTER SESSION SET NLS_LANGUAGE='American';

-- The first dot in the spool command below is
-- the SQL*Plus concatenation character

spool &log_dir.sh_main.log

-- Dropping the user with all its objects

DROP USER sh CASCADE;

REM =======================================================
REM create user
REM THIS WILL ONLY WORK IF APPROPRIATE TS ARE PRESENT
REM =======================================================
```

```
CREATE USER sh IDENTIFIED BY &pass;

ALTER USER sh DEFAULT TABLESPACE &tbs
 QUOTA UNLIMITED ON &tbs;
ALTER USER sh TEMPORARY TABLESPACE &ttbs;

GRANT CREATE SESSION
,     ALTER SESSION
,     CREATE TABLE
,     CREATE DIMENSION
,     CREATE MATERIALIZED VIEW
,     QUERY REWRITE
,     CREATE ANY DIRECTORY
,     DROP ANY DIRECTORY
TO    sh;

ALTER USER sh DEFAULT ROLE ALL;

rem   ALTER USER sh GRANT CONNECT THROUGH olapsvr;

GRANT select_catalog_role TO sh;

REM =========================================================
REM grants for sys schema
REM =========================================================

CONNECT sys/&pass_sys AS SYSDBA;
GRANT execute ON sys.dbms_stats TO sh;

REM =========================================================
REM create sh schema objects (sales history - star schema)
REM =========================================================

CONNECT sh/&pass
PROMPT creating tables ...
@&data_dir.sh_cre.sql

PROMPT inserting rows tables ...
@&data_dir.sh_pop1.sql
@&data_dir.sh_pop2.sql

PROMPT loading data ...
@&data_dir.sh_pop3.sql &pass &data_dir &log_dir
```

```
PROMPT creating indexes ...
@&data_dir.sh_idx.sql

PROMPT adding constraints ...
@&data_dir.sh_cons.sql

PROMPT creating dimensions and hierarchies ...
@&data_dir.sh_hiera.sql

PROMPT creating materialized views ...
@&data_dir.sh_cremv.sql

PROMPT gathering statistics ...
@&data_dir.sh_analz.sql

PROMPT adding comments ...
@&data_dir.sh_comnt.sql

PROMPT creating PLAN_TABLE ...
@?/rdbms/admin/utlxplan.sql

PROMPT creating REWRITE_TABLE ...
@?/rdbms/admin/utlxrw.sql

PROMPT creating MV_CAPABILITIES_TABLE ...
@?/rdbms/admin/utlxmv.sql

COMMIT;

spool off
```

sh_olp_c.sql

```
Rem
Rem $Header: sh_olp_c.sql 05-mar-2001.13:53:14 ahunold Exp $
Rem
Rem sh_olp_c.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
Rem
Rem    NAME
Rem      sh_olp_c.sql - Create columns used by OLAP Server
Rem
Rem    DESCRIPTION
```

```
Rem      SH is the Sales History schema of the Oracle 9i Sample
Rem   Schemas
Rem
Rem    NOTES
Rem
Rem
Rem    MODIFIED   (MM/DD/YY)
rem      pfay      04/10/01 - change case
Rem      ahunold   04/05/01 - dimension names
Rem      ahunold   03/05/01 - external table, no DROPs
Rem      ahunold   02/07/01 - CMWLite
Rem      ahunold   02/01/01 - Merged ahunold_two_facts
Rem      hbaer     01/29/01 - Created
Rem

ALTER TABLE products
ADD prod_total VARCHAR2(13)
DEFAULT 'Product total';

ALTER TABLE customers
ADD cust_total VARCHAR2(14)
DEFAULT 'Customer total';

ALTER TABLE promotions
ADD promo_total VARCHAR2(15)
DEFAULT 'Promotion total';

ALTER TABLE channels
ADD channel_total VARCHAR2(13)
DEFAULT 'Channel total';

ALTER TABLE countries
ADD country_total VARCHAR2(11)
DEFAULT 'World total';

COMMIT;

Rem modified dimension definition to include new total column

DROP DIMENSION times_dim;

CREATE DIMENSION times_dim
   LEVEL day         IS TIMES.TIME_ID
   LEVEL month       IS TIMES.CALENDAR_MONTH_DESC
   LEVEL quarter     IS TIMES.CALENDAR_QUARTER_DESC
```

```
   LEVEL year        IS TIMES.CALENDAR_YEAR
   LEVEL fis_week    IS TIMES.WEEK_ENDING_DAY
   LEVEL fis_month   IS TIMES.FISCAL_MONTH_DESC
   LEVEL fis_quarter IS TIMES.FISCAL_QUARTER_DESC
   LEVEL fis_year    IS TIMES.FISCAL_YEAR
   HIERARCHY cal_rollup    (
             day     CHILD OF
             month   CHILD OF
             quarter CHILD OF
             year
   )
   HIERARCHY fis_rollup    (
             day         CHILD OF
             fis_week    CHILD OF
             fis_month CHILD OF
             fis_quarter CHILD OF
             fis_year
   )
   ATTRIBUTE day DETERMINES
(day_number_in_week, day_name, day_number_in_month,
         calendar_week_number)
   ATTRIBUTE month DETERMINES
(calendar_month_desc,
         calendar_month_number, calendar_month_name,
         days_in_cal_month, end_of_cal_month)
   ATTRIBUTE quarter DETERMINES
(calendar_quarter_desc,
         calendar_quarter_number,days_in_cal_quarter,
 end_of_cal_quarter)
   ATTRIBUTE year DETERMINES
(calendar_year,
         days_in_cal_year, end_of_cal_year)
   ATTRIBUTE fis_week DETERMINES
(week_ending_day,
         fiscal_week_number)
   ATTRIBUTE fis_month DETERMINES
(fiscal_month_desc, fiscal_month_number, fiscal_month_name,
 days_in_fis_month, end_of_fis_month)
   ATTRIBUTE fis_quarter DETERMINES
(fiscal_quarter_desc,
         fiscal_quarter_number, days_in_fis_quarter,
 end_of_fis_quarter)
   ATTRIBUTE fis_year DETERMINES
(fiscal_year,
         days_in_fis_year, end_of_fis_year)
```

```
;

execute dbms_olap.validate_dimension('times_dim','sh',false,true)
SELECT COUNT(*) FROM mview$_exceptions;

DROP DIMENSION customers_dim;

CREATE DIMENSION customers_dim
LEVEL customerIS (customers.cust_id)
LEVEL city IS (customers.cust_city)
LEVEL state IS (customers.cust_state_province)
LEVEL country IS (countries.country_id)
LEVEL subregion IS (countries.country_subregion)
LEVEL region IS (countries.country_region)
LEVEL geog_total IS (countries.country_total)
LEVEL cust_total IS (customers.cust_total)
HIERARCHY cust_rollup (
customerCHILD OF
city CHILD OF
state CHILD OF
               cust_total
)
HIERARCHY geog_rollup (
customerCHILD OF
city CHILD OF
state CHILD OF
country CHILD OF
subregion CHILD OF
region         CHILD OF
               geog_total
JOIN KEY (customers.country_id) REFERENCES country
)
ATTRIBUTE customer DETERMINES
(cust_first_name, cust_last_name, cust_gender,
 cust_marital_status, cust_year_of_birth,
 cust_income_level, cust_credit_limit,
         cust_street_address, cust_postal_code,
         cust_main_phone_number, cust_email)
        ATTRIBUTE city DETERMINES (cust_city)
        ATTRIBUTE state DETERMINES (cust_state_province)
ATTRIBUTE country DETERMINES (countries.country_name)
        ATTRIBUTE subregion DETERMINES (countries.country_subregion)
        ATTRIBUTE region DETERMINES (countries.country_region)
        ATTRIBUTE geog_total DETERMINES (countries.country_total)
        ATTRIBUTE cust_total DETERMINES (customers.cust_total)
```

```
;

execute dbms_olap.validate_dimension('customers_
dim','sh',false,true)
SELECT COUNT(*) FROM mview$_exceptions;

DROP DIMENSION products_dim;

CREATE DIMENSION products_dim
LEVEL product IS (products.prod_id)
 LEVEL subcategory IS (products.prod_subcategory)
LEVEL categoryIS (products.prod_category)
LEVEL prod_totalIS (products.prod_total)
HIERARCHY prod_rollup (
productCHILD OF
subcategory CHILD OF
category        CHILD OF
   prod_total
)
ATTRIBUTE product DETERMINES
        (products.prod_name, products.prod_desc,
         prod_weight_class, prod_unit_of_measure,
         prod_pack_size,prod_status, prod_list_price, prod_min_
price)
ATTRIBUTE subcategory DETERMINES
        (prod_subcategory, prod_subcat_desc)
ATTRIBUTE category DETERMINES
        (prod_category, prod_cat_desc)
ATTRIBUTE prod_total DETERMINES
        (prod_total)
;

execute dbms_olap.validate_dimension('products_dim','sh',false,true)
SELECT COUNT(*) FROM mview$_exceptions;

DROP DIMENSION promotions_dim;

CREATE DIMENSION promotions_dim
LEVEL promo   IS (promotions.promo_id)
LEVEL subcategory IS (promotions.promo_subcategory)
LEVEL category   IS (promotions.promo_category)
LEVEL promo_total IS (promotions.promo_total)
HIERARCHY promo_rollup (
promo CHILD OF
subcategory CHILD OF
```

```
categoryCHILD OF
promo_total
)
ATTRIBUTE promo DETERMINES
        (promo_name, promo_cost,
         promo_begin_date, promo_end_date)
        ATTRIBUTE subcategory DETERMINES (promo_subcategory)
        ATTRIBUTE category DETERMINES (promo_category)
        ATTRIBUTE promo_total DETERMINES (promo_total)
;

execute dbms_olap.validate_dimension('promotions_
dim','sh',false,true)
SELECT COUNT(*) FROM mview$_exceptions;

DROP DIMENSION channels_dim;

CREATE DIMENSION channels_dim
LEVEL channel    IS (channels.channel_id)
LEVEL channel_class IS (channels.channel_class)
LEVEL channel_total IS (channels.channel_total)
HIERARCHY channel_rollup (
channelCHILD OF
channel_classCHILD OF
channel_total
)
        ATTRIBUTE channel DETERMINES (channel_desc)
        ATTRIBUTE channel_class DETERMINES (channel_class)
        ATTRIBUTE channel_total DETERMINES (channel_total)
;

execute dbms_olap.validate_dimension('channels_dim','sh',false,true)
SELECT COUNT(*) FROM mview$_exceptions;

rem ------------------------------------------------------------------
rem    CMWLite
rem ------------------------------------------------------------------

set serveroutput on size 99999

declare
  CUBE_TYPE constant varchar2(30) := 'CUBE';
  MEASURE_TYPE constant varchar2(30) := 'MEASURE';
  DIMENSION_TYPE constant varchar2(30) := 'DIMENSION';
  HIERARCHY_TYPE constant varchar2(30) := 'HIERARCHY';
```

```
  LEVEL_TYPE constant varchar2(30) := 'LEVEL';
  DIMENSION_ATTRIBUTE_TYPE constant varchar2(30) := 'DIMENSION
ATTRIBUTE';
  LEVEL_ATTRIBUTE_TYPE constant varchar2(30) := 'LEVEL ATTRIBUTE';
  TABLE_TYPE constant varchar2(30) := 'TABLE';
  COLUMN_TYPE constant varchar2(30) := 'COLUMN';
  FOREIGN_KEY_TYPE constant varchar2(30) := 'FOREIGN KEY';
  FUNCTION_TYPE constant varchar2(30) := 'FUNCTION';
  PARAMETER_TYPE constant varchar2(30) := 'PARAMETER';
  CATALOG_TYPE constant varchar2(30) := 'CATALOG';
  DESCRIPTOR_TYPE constant varchar2(30) := 'DESCRIPTOR';
  INSTANCE_TYPE CONSTANT VARCHAR2(30) := 'INSTANCE';

  sh_products_dim number;
  sh_customers_dim number;
  sh_times_dim number;
  sh_channels_dim number;
  sh_promotions_dim number;
  time_desc_id number;
  time_span_id number;
  end_date_id number;
  long_desc_id number;
  short_desc_id number;
  desc_id number;
  name_id number;
  sh_catId number;
  tmp number;
  errtxt varchar(60);


begin
 dbms_output.put_line
('<<<<< CREATE CWMLite Metadata for the Sales History Schema
>>>>>');
 dbms_output.put_line('-');
 dbms_output.put_line
('<<<<< CREATE CATALOG sh_cat for Sales History >>>>>');
begin
   select catalog_id into sh_catId
     from all_olap_catalogs
     where catalog_name = 'SH_CAT';
   cwm_classify.drop_catalog(sh_catId, true);
   dbms_output.put_line('Catalog Dropped');
 exception
   when no_data_found then
```

```
      dbms_output.put_line('No catalog to drop');
    when cwm_exceptions.catalog_not_found then
      dbms_output.put_line('No catalog to drop');
 end;
 sh_catId := cwm_classify.create_catalog('SH_CAT', 'Sales History
CWM Business Area');
 dbms_output.put_line('CWM Collect Garbage');
 cwm_utility.collect_garbage;


dbms_output.put_line('-');
dbms_output.put_line
 ('<<<<< CREATE the Sales CUBE >>>>>');
dbms_output.put_line
 ('Sales amount, Sales quantity
<TIMES CHANNELS PRODUCTS CUSTOMERS PROMOTIONS >');
begin
   dbms_output.put_line('Drop SALES_CUBE prior to recreation');
   cwm_olap_cube.drop_cube(USER, 'SALES_CUBE');
   dbms_output.put_line('Cube Dropped');
 exception
   when cwm_exceptions.cube_not_found then
     dbms_output.put_line('No cube to drop');
 end;

CWM_OLAP_CUBE.Create_Cube(USER, 'SALES_CUBE' , 'Sales Analysis',
'Sales amount, Sales quantity <TIMES CHANNELS PRODUCTS CUSTOMERS
PROMOTIONS >');

dbms_output.put_line
('Add dimensions -
 to SALES_CUBE and map the foreign keys');

-- The level name in the map_cube parameter list names
--the lowest level of aggregation.  It must be the
--lowest level in the dimension that contains data

sh_times_dim := CWM_OLAP_CUBE.Add_Dimension(USER, 'SALES_CUBE' ,
USER, 'TIMES_DIM', 'TIMES_DIM');
CWM_OLAP_CUBE.Map_Cube(USER, 'SALES_CUBE' , USER, 'SALES', 'SALES_
TIME_FK', 'DAY', USER, 'TIMES_DIM', 'TIMES_DIM');

sh_channels_dim := CWM_OLAP_CUBE.Add_Dimension(USER, 'SALES_CUBE' ,
USER, 'CHANNELS_DIM', 'CHANNELS_DIM');
CWM_OLAP_CUBE.Map_Cube(USER, 'SALES_CUBE' , USER, 'SALES', 'SALES_
```

```
CHANNEL_FK', 'CHANNEL', USER, 'CHANNELS_DIM', 'CHANNELS_DIM');

sh_products_dim := CWM_OLAP_CUBE.Add_Dimension(USER, 'SALES_CUBE' ,
USER, 'PRODUCTS_DIM', 'PRODUCTS_DIM');
CWM_OLAP_CUBE.Map_Cube(USER, 'SALES_CUBE' , USER, 'SALES', 'SALES_
PRODUCT_FK', 'PRODUCT', USER, 'PRODUCTS_DIM', 'PRODUCTS_DIM');

sh_customers_dim := CWM_OLAP_CUBE.Add_Dimension(USER, 'SALES_CUBE' ,
USER, 'CUSTOMERS_DIM', 'CUSTOMERS_DIM');
CWM_OLAP_CUBE.Map_Cube(USER, 'SALES_CUBE' , USER, 'SALES', 'SALES_
CUSTOMER_FK', 'CUSTOMER', USER, 'CUSTOMERS_DIM', 'CUSTOMERS_DIM');

sh_promotions_dim := CWM_OLAP_CUBE.Add_Dimension(USER, 'SALES_CUBE'
, USER, 'PROMOTIONS_DIM', 'PROMOTIONS_DIM');
CWM_OLAP_CUBE.Map_Cube(USER, 'SALES_CUBE' , USER, 'SALES', 'SALES_
PROMO_FK', 'PROMO', USER, 'PROMOTIONS_DIM', 'PROMOTIONS_DIM');


dbms_output.put_line
('Create measures -
 for SALES_CUBE and map to columns in the fact table');

CWM_OLAP_MEASURE.Create_Measure
(USER, 'SALES_CUBE' , 'SALES_AMOUNT', 'Sales', 'Dollar Sales');
CWM_OLAP_MEASURE.Set_Column_Map
(USER, 'SALES_CUBE' , 'SALES_AMOUNT', USER, 'SALES', 'AMOUNT_SOLD');

CWM_OLAP_MEASURE.Create_Measure
(USER, 'SALES_CUBE' , 'SALES_QUANTITY', 'Quantity', 'Quantity
Sold');
CWM_OLAP_MEASURE.Set_Column_Map
(USER, 'SALES_CUBE' , 'SALES_QUANTITY', USER, 'SALES', 'QUANTITY_
SOLD');

dbms_output.put_line
('Set default aggregation method -
 to SUM for all measures over TIME');
 tmp:= cwm_utility.create_function_usage('SUM');
 cwm_olap_measure.set_default_aggregation_method
(USER, 'SALES_CUBE', 'SALES_AMOUNT', tmp, USER, 'TIMES_DIM', 'TIMES_
DIM');
 tmp:= cwm_utility.create_function_usage('SUM');
 cwm_olap_measure.set_default_aggregation_method
(USER, 'SALES_CUBE', 'SALES_QUANTITY', tmp, USER, 'TIMES_DIM',
'TIMES_DIM');
```

```
dbms_output.put_line('Add SALES_CUBE to the catalog');
 begin
   select catalog_id into sh_catId
     from all_olap_catalogs
     where catalog_name = 'SH_CAT';
   cwm_classify.add_catalog_entity(sh_catID, USER, 'SALES_CUBE',
'SALES_AMOUNT');
   cwm_classify.add_catalog_entity(sh_catID, USER, 'SALES_CUBE',
'SALES_QUANTITY');
   dbms_output.put_line('SALES_CUBE successfully added to sh_cat');
 exception
   when no_data_found then
     dbms_output.put_line('        No sh_cat catalog to add sales_
cube to');
 end;


dbms_output.put_line('-');
dbms_output.put_line
('<<<<< CREATE the Cost CUBE >>>>>');
dbms_output.put_line
 ('Unit Cost, Unit Price < TIMES PRODUCTS >');
begin
   dbms_output.put_line('Drop COST_CUBE prior to recreation');
   cwm_olap_cube.drop_cube(USER, 'COST_CUBE');
   dbms_output.put_line('Cube Dropped');
 exception
   when cwm_exceptions.cube_not_found then
     dbms_output.put_line('      No cube to drop');
 end;

CWM_OLAP_CUBE.Create_Cube(USER, 'COST_CUBE' , 'Cost Analysis', 'Unit
Cost, Unit Price < TIMES PRODUCTS >');


dbms_output.put_line
('Add dimensions -
 to COST_CUBE and map the foreign keys');

-- The level name in the map_cube parameter list names
--the lowest level of aggregation.  It must be the
--lowest level in the dimension that contains data

sh_times_dim := CWM_OLAP_CUBE.Add_Dimension(USER, 'COST_CUBE' ,
```

```
USER, 'TIMES_DIM', 'TIMES_DIM');
CWM_OLAP_CUBE.Map_Cube(USER, 'COST_CUBE' , USER, 'COSTS', 'COSTS_
TIME_FK', 'DAY', USER, 'TIMES_DIM', 'TIMES_DIM');

sh_products_dim := CWM_OLAP_CUBE.Add_Dimension(USER, 'COST_CUBE' ,
USER, 'PRODUCTS_DIM', 'PRODUCTS_DIM');
CWM_OLAP_CUBE.Map_Cube(USER, 'COST_CUBE' , USER, 'COSTS', 'COSTS_
PRODUCT_FK', 'PRODUCT', USER, 'PRODUCTS_DIM', 'PRODUCTS_DIM');


dbms_output.put_line
('Create measures -
 for COST_CUBE and map to columns in the fact table');

CWM_OLAP_MEASURE.Create_Measure(USER, 'COST_CUBE' , 'UNIT_COST',
'Cost', 'Unit Cost Amount');
CWM_OLAP_MEASURE.Set_Column_Map(USER, 'COST_CUBE' , 'UNIT_COST',
USER, 'COSTS', 'UNIT_COST');

CWM_OLAP_MEASURE.Create_Measure(USER, 'COST_CUBE' , 'UNIT_PRICE',
'Price', 'Unit Price Amount');
CWM_OLAP_MEASURE.Set_Column_Map(USER, 'COST_CUBE' , 'UNIT_PRICE',
USER, 'COSTS', 'UNIT_PRICE');


dbms_output.put_line
('Set default aggregation method -
 to SUM for all measures over TIME');
 tmp:= cwm_utility.create_function_usage('SUM');
 cwm_olap_measure.set_default_aggregation_method
(USER, 'COST_CUBE', 'UNIT_COST', tmp, USER, 'TIMES_DIM', 'TIMES_
DIM');
 tmp:= cwm_utility.create_function_usage('SUM');
 cwm_olap_measure.set_default_aggregation_method
(USER, 'COST_CUBE', 'UNIT_PRICE', tmp, USER, 'TIMES_DIM', 'TIMES_
DIM');


dbms_output.put_line('Add COST_CUBE to the catalog');
 begin
   select catalog_id into sh_catId
     from all_olap_catalogs
     where catalog_name = 'SH_CAT';
   cwm_classify.add_catalog_entity(sh_catID, USER, 'COST_CUBE',
'UNIT_COST');
```

```
   cwm_classify.add_catalog_entity(sh_catID, USER, 'COST_CUBE',
'UNIT_PRICE');
   dbms_output.put_line('COST_CUBE successfully added to sh_cat');
   dbms_output.put_line(' ');
 exception
   when no_data_found then
     dbms_output.put_line('      No sh_cat catalog to add COST_CUBE
to');
     dbms_output.put_line(' ');
 end;



dbms_output.put_line('-');
dbms_output.put_line('<<<<< TIME DIMENSION >>>>>');

dbms_output.put_line
('Dimension - display name, description and plural name');

CWM_OLAP_DIMENSION.set_display_name(USER, 'TIMES_DIM', 'Time');
CWM_OLAP_DIMENSION.set_description(USER, 'TIMES_DIM', 'Time
Dimension Values');
CWM_OLAP_DIMENSION.set_plural_name(USER, 'TIMES_DIM', 'Times');

dbms_output.put_line
('Level - display name and description');

cwm_olap_level.set_display_name(USER, 'TIMES_DIM', 'DAY', 'Day');
cwm_olap_level.set_description(USER, 'TIMES_DIM', 'DAY', 'Day level
of the Calendar hierarchy');

cwm_olap_level.set_display_name(USER, 'TIMES_DIM', 'MONTH',
'Month');
cwm_olap_level.set_description(USER, 'TIMES_DIM', 'MONTH', 'Month
level of the Calendar hierarchy');

cwm_olap_level.set_display_name(USER, 'TIMES_DIM', 'QUARTER',
'Quarter');
cwm_olap_level.set_description(USER, 'TIMES_DIM', 'QUARTER',
'Quarter level of the Calendar hierarchy');

cwm_olap_level.set_display_name(USER, 'TIMES_DIM', 'YEAR', 'Year');
cwm_olap_level.set_description(USER, 'TIMES_DIM', 'YEAR', 'Year
level of the Calendar hierarchy');
```

```
cwm_olap_level.set_display_name(USER, 'TIMES_DIM', 'FIS_WEEK',
'Fiscal Week');
cwm_olap_level.set_description(USER, 'TIMES_DIM',  'FIS_WEEK', 'Week
level of the Fiscal hierarchy');

cwm_olap_level.set_display_name(USER, 'TIMES_DIM', 'FIS_MONTH',
'Fiscal Month');
cwm_olap_level.set_description(USER, 'TIMES_DIM', 'FIS_MONTH',
'Month level of the Fiscal hierarchy');

cwm_olap_level.set_display_name(USER, 'TIMES_DIM', 'FIS_QUARTER',
'Fiscal Quarter');
cwm_olap_level.set_description(USER, 'TIMES_DIM', 'FIS_QUARTER',
'Quarter level of the Fiscal hierarchy');

cwm_olap_level.set_display_name(USER, 'TIMES_DIM', 'FIS_YEAR',
'Fiscal Year');
cwm_olap_level.set_description(USER, 'TIMES_DIM', 'FIS_YEAR', 'Year
level of the Fiscal hierarchy');


dbms_output.put_line
('Hierarchy - display name and description');

cwm_olap_hierarchy.set_display_name(USER, 'TIMES_DIM', 'CAL_ROLLUP',
'Calendar');
cwm_olap_hierarchy.set_description(USER, 'TIMES_DIM', 'CAL_ROLLUP',
'Standard Calendar hierarchy');

cwm_olap_hierarchy.set_display_name(USER, 'TIMES_DIM', 'FIS_ROLLUP',
'Fiscal');
cwm_olap_hierarchy.set_description(USER, 'TIMES_DIM', 'FIS_ROLLUP',
'Fiscal hierarchy');


dbms_output.put_line('- default calculation hierarchy');
cwm_olap_cube.set_default_calc_hierarchy(USER,'SALES_CUBE', 'CAL_
ROLLUP', USER, 'TIMES_DIM', 'TIMES_DIM');
cwm_olap_cube.set_default_calc_hierarchy(USER,'COST_CUBE', 'CAL_
ROLLUP', USER, 'TIMES_DIM', 'TIMES_DIM');


dbms_output.put_line('- default display hierarchy');
cwm_olap_dimension.set_default_display_hierarchy(USER, 'TIMES_DIM',
'CAL_ROLLUP');
```

```
dbms_output.put_line
('Level Attributes - name, display name, description');

--Level: DAY
cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'DAY', 'DAY_
NUMBER_IN_WEEK', 'DAY_NUMBER_IN_WEEK');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'DAY',
'DAY_NUMBER_IN_WEEK',  'Day Number in Week');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'DAY',
'DAY_NUMBER_IN_WEEK',  'Day Number in Week where Monday is day
number 1');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'DAY', 'DAY_
NAME', 'DAY_NAME');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'DAY',
'DAY_NAME',  'Day Name');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'DAY',
'DAY_NAME',  'Name of the Day of the Week');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'DAY', 'DAY_
NUMBER_IN_MONTH', 'DAY_NUMBER_IN_MONTH');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'DAY',
'DAY_NUMBER_IN_MONTH',  'Day Number in Month');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'DAY',
'DAY_NUMBER_IN_MONTH',  'Day number in month');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'DAY',
'CALENDAR_WEEK_NUMBER', 'CALENDAR_WEEK_NUMBER');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'DAY',
'CALENDAR_WEEK_NUMBER',  'Calendar Week Number');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'DAY',
'CALENDAR_WEEK_NUMBER',  'Calendar Week Number');

--Level: MONTH
cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'MONTH',
'CALENDAR_MONTH_DESC', 'CALENDAR_MONTH_DESC');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM',
'MONTH', 'CALENDAR_MONTH_DESC',  'Calendar Month');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'MONTH',
'CALENDAR_MONTH_DESC',  'Calendar Month Description');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'MONTH',
'CALENDAR_MONTH_NUMBER', 'CALENDAR_MONTH_NUMBER');
```

```
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM',
'MONTH', 'CALENDAR_MONTH_NUMBER',  'Calendar Month Number');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'MONTH',
'CALENDAR_MONTH_NUMBER',  'Month Number in Calendar year where
January is month number 1');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'MONTH',
'CALENDAR_MONTH_NAME', 'CALENDAR_MONTH_NAME');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM',
'MONTH', 'CALENDAR_MONTH_NAME',  'Calendar Month Name');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'MONTH',
'CALENDAR_MONTH_NAME',  'Name of the Month');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'MONTH', 'DAYS_
IN_CAL_MONTH', 'DAYS_IN_CAL_MONTH');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM',
'MONTH', 'DAYS_IN_CAL_MONTH',  'Days in Calendar Month');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'MONTH',
'DAYS_IN_CAL_MONTH',  'Number of Days in Calendar Month');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'MONTH', 'END_
OF_CAL_MONTH', 'END_OF_CAL_MONTH');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM',
'MONTH', 'END_OF_CAL_MONTH',  'End of Calendar Month');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'MONTH',
'END_OF_CAL_MONTH',  'Last Day of the Calendar Month');

--Level: QUARTER
cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'QUARTER',
'CALENDAR_QUARTER_DESC', 'CALENDAR_QUARTER_DESC');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM',
'QUARTER', 'CALENDAR_QUARTER_DESC',  'Calendar Quarter');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM',
'QUARTER', 'CALENDAR_QUARTER_DESC',  'Calendar Quarter
Description');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'QUARTER',
'CALENDAR_QUARTER_NUMBER', 'CALENDAR_QUARTER_NUMBER');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM',
'QUARTER', 'CALENDAR_QUARTER_NUMBER',  'Calendar Quarter Number');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM',
'QUARTER', 'CALENDAR_QUARTER_NUMBER',  'Calendar Quarter Number');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'QUARTER',
'DAYS_IN_CAL_QUARTER', 'DAYS_IN_CAL_QUARTER');
```

```
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM',
'QUARTER', 'DAYS_IN_CAL_QUARTER',  'Days in Calendar Quarter');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM',
'QUARTER', 'DAYS_IN_CAL_QUARTER',  'Number of Days in Calendar
Quarter');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'QUARTER',
'END_OF_CAL_QUARTER', 'END_OF_CAL_QUARTER');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM',
'QUARTER', 'END_OF_CAL_QUARTER',  'End of Calendar Quarter');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM',
'QUARTER', 'END_OF_CAL_QUARTER',  'Last Day of the Calendar
Quarter');

--Level: YEAR
cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'YEAR',
'CALENDAR_YEAR', 'CALENDAR_YEAR');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'YEAR',
'CALENDAR_YEAR',  'Calendar Year');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'YEAR',
'CALENDAR_YEAR',  'Calendar Year');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'YEAR', 'DAYS_
IN_CAL_YEAR', 'DAYS_IN_CAL_YEAR');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'YEAR',
'DAYS_IN_CAL_YEAR',  'Days in Calendar Year');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'YEAR',
'DAYS_IN_CAL_YEAR',  'Number of Days in Calendar Year');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'YEAR', 'END_
OF_CAL_YEAR', 'END_OF_CAL_YEAR');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'YEAR',
'END_OF_CAL_YEAR',  'End of Calendar Year');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'YEAR',
'END_OF_CAL_YEAR',  'Last Day of the Calendar Year');

--Level: FISCAL WEEK
cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'FIS_WEEK',
'FISCAL_WEEK_NUMBER', 'FISCAL_WEEK_NUMBER');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'FIS_
WEEK', 'FISCAL_WEEK_NUMBER',  'Fiscal Week Number');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'FIS_
WEEK', 'FISCAL_WEEK_NUMBER',  'Fiscal Week Number');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'FIS_WEEK',
```

```
'WEEK_ENDING_DAY', 'WEEK_ENDING_DAY');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'FIS_
WEEK', 'WEEK_ENDING_DAY',  'Fiscal Week Ending Day');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'FIS_
WEEK', 'WEEK_ENDING_DAY',  'Fiscal Week Ending Day');

--Level: FISCAL MONTH
cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'FIS_MONTH',
'FISCAL_MONTH_DESC', 'FISCAL_MONTH_DESC');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'FIS_
MONTH', 'FISCAL_MONTH_DESC',  'Fiscal Month');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'FIS_
MONTH', 'FISCAL_MONTH_DESC',  'Fiscal Month Description');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'FIS_MONTH',
'FISCAL_MONTH_NUMBER', 'FISCAL_MONTH_NUMBER');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'FIS_
MONTH', 'FISCAL_MONTH_NUMBER',  'Fiscal Month Number');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'FIS_
MONTH', 'FISCAL_MONTH_NUMBER',  'Fiscal Month Number');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'FIS_MONTH',
'FISCAL_MONTH_NAME', 'FISCAL_MONTH_NAME');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'FIS_
MONTH', 'FISCAL_MONTH_NAME',  'Fiscal Month Name');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'FIS_
MONTH', 'FISCAL_MONTH_NAME',  'Fiscal Month Name');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'FIS_MONTH',
'DAYS_IN_FIS_MONTH', 'DAYS_IN_FIS_MONTH');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'FIS_
MONTH', 'DAYS_IN_FIS_MONTH',  'DAYS_IN_FIS_MONTH');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'FIS_
MONTH', 'DAYS_IN_FIS_MONTH',  'Number of Days in Fiscal Month');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'FIS_MONTH',
'END_OF_FIS_MONTH', 'END_OF_FIS_MONTH');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'FIS_
MONTH', 'END_OF_FIS_MONTH',  'End of Fiscal Month');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'FIS_
MONTH', 'END_OF_FIS_MONTH',  'Last Day of the Fiscal Month');

--Level: FISCAL QUARTER
cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'FIS_QUARTER',
'FISCAL_QUARTER_NUMBER', 'FISCAL_QUARTER_NUMBER');
```

```
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'FIS_
QUARTER', 'FISCAL_QUARTER_NUMBER',  'Fiscal Quarter Number');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'FIS_
QUARTER', 'FISCAL_QUARTER_NUMBER',  'Fiscal Quarter Number');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'FIS_QUARTER',
'DAYS_IN_FIS_QUARTER', 'DAYS_IN_FIS_QUARTER');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'FIS_
QUARTER', 'DAYS_IN_FIS_QUARTER',  'Days in Fiscal Quarter');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'FIS_
QUARTER', 'DAYS_IN_FIS_QUARTER',  'Number of Days in Fiscal
Quarter');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'FIS_QUARTER',
'END_OF_FIS_QUARTER', 'END_OF_FIS_QUARTER');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'FIS_
QUARTER', 'END_OF_FIS_QUARTER',  'End of Fiscal Quarter');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'FIS_
QUARTER', 'END_OF_FIS_QUARTER',  'Last Day of the Fiscal Quarter');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'FIS_QUARTER',
'FISCAL_QUARTER_DESC', 'FISCAL_QUARTER_DESC');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'FIS_
QUARTER', 'FISCAL_QUARTER_DESC',  'Fiscal Quarter Description');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'FIS_
QUARTER', 'FISCAL_QUARTER_DESC',  'Fiscal Quarter Description');

--Level: FISCAL YEAR
cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'FIS_YEAR',
'DAYS_IN_FIS_YEAR', 'DAYS_IN_FIS_YEAR');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'FIS_
YEAR', 'DAYS_IN_FIS_YEAR',  'Days in Fiscal Year');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'FIS_
YEAR', 'DAYS_IN_FIS_YEAR',  'Number of Days in Fiscal Year');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'FIS_YEAR',
'END_OF_FIS_YEAR', 'END_OF_FIS_YEAR');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'FIS_
YEAR', 'END_OF_FIS_YEAR',  'End of Fiscal Year');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'FIS_
YEAR', 'END_OF_FIS_YEAR',  'Last Day of the Fiscal Year');

cwm_olap_level_attribute.set_name(USER, 'TIMES_DIM', 'FIS_YEAR',
'FISCAL_YEAR', 'FISCAL_YEAR');
cwm_olap_level_attribute.set_display_name(USER, 'TIMES_DIM', 'FIS_
```

```
YEAR', 'FISCAL_YEAR',  'Fiscal Year');
cwm_olap_level_attribute.set_description(USER, 'TIMES_DIM', 'FIS_
YEAR', 'FISCAL_YEAR',  'Fiscal Year');


dbms_output.put_line
('Drop dimension attributes prior to re-creation');

 begin
    cwm_olap_dim_attribute.drop_dimension_attribute
(USER, 'TIMES_DIM', 'Long Description');
    dbms_output.put_line('- Long Description dropped');
 exception
    when cwm_exceptions.attribute_not_found then
      null;
 end;
  begin
    cwm_olap_dim_attribute.drop_dimension_attribute
(USER, 'TIMES_DIM', 'Short Description');
    dbms_output.put_line('- Short Description dropped');
 exception
    when cwm_exceptions.attribute_not_found then
      null;
 end;
begin
    cwm_olap_dim_attribute.drop_dimension_attribute
(USER, 'TIMES_DIM', 'Period Number');
    dbms_output.put_line('- Period Number dropped');
 exception
    when cwm_exceptions.attribute_not_found then
      null;
 end;
begin
    cwm_olap_dim_attribute.drop_dimension_attribute
(USER, 'TIMES_DIM', 'Period Number of Days');
    dbms_output.put_line('- Period Number of Days dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('      No attribute to drop');
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute
(USER, 'TIMES_DIM', 'Period End Date');
    dbms_output.put_line('- Period End Date dropped');
 exception
```

```
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('      No attribute to drop');
 end;


dbms_output.put_line
('Create dimension attributes and add their level attributes');

--Level attributes must be associated with a Dimension attribute
--SQL does not create Dimension attributes, so we do it here

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute
(USER, 'TIMES_DIM', 'Long Description', 'Long Time Period Names',
'Full name of time periods');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Long Description', 'DAY', 'DAY_NAME');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Long Description', 'MONTH', 'CALENDAR_MONTH_DESC');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Long Description', 'FIS_MONTH', 'FISCAL_MONTH_DESC');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Long Description', 'QUARTER', 'CALENDAR_QUARTER_DESC');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Long Description', 'FIS_QUARTER', 'FISCAL_QUARTER_DESC');
dbms_output.put_line('- Long Description created');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute
(USER, 'TIMES_DIM', 'Short Description', 'Short Time Period Names',
'Short name of time periods');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Short Description', 'DAY', 'DAY_NAME');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Short Description', 'MONTH', 'CALENDAR_MONTH_DESC');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Short Description', 'FIS_MONTH', 'FISCAL_MONTH_DESC');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Short Description', 'QUARTER', 'CALENDAR_QUARTER_DESC');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Short Description', 'FIS_QUARTER', 'FISCAL_QUARTER_DESC');
dbms_output.put_line('- Short Description created');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'TIMES_DIM',
'Period Number', 'Period Number', 'Number of the Time Period');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period Number', 'DAY', 'DAY_NUMBER_IN_WEEK');
```

```
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period Number', 'DAY', 'DAY_NUMBER_IN_MONTH');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period Number', 'MONTH', 'CALENDAR_MONTH_NUMBER');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period Number', 'QUARTER', 'CALENDAR_QUARTER_NUMBER');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period Number', 'FIS_WEEK', 'FISCAL_WEEK_NUMBER');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period Number', 'FIS_MONTH', 'FISCAL_MONTH_NUMBER');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period Number', 'FIS_QUARTER', 'FISCAL_QUARTER_NUMBER');
dbms_output.put_line('- Period Number created');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'TIMES_DIM',
'Period Number of Days', 'Period Number of Days', 'Number of Days in
Time Period');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period Number of Days', 'MONTH', 'DAYS_IN_CAL_MONTH');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period Number of Days', 'QUARTER', 'DAYS_IN_CAL_QUARTER');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period Number of Days', 'YEAR', 'DAYS_IN_CAL_YEAR');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period Number of Days', 'FIS_MONTH', 'DAYS_IN_FIS_MONTH');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period Number of Days', 'FIS_QUARTER', 'DAYS_IN_FIS_QUARTER');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period Number of Days', 'FIS_YEAR', 'DAYS_IN_FIS_YEAR');
dbms_output.put_line('- Period Number of Days created');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'TIMES_DIM',
'Period End Date', 'Period End Date', 'Last Day in Time Period');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period End Date', 'MONTH', 'END_OF_CAL_MONTH');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period End Date', 'QUARTER', 'END_OF_CAL_QUARTER');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period End Date', 'YEAR', 'END_OF_CAL_YEAR');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period End Date', 'FIS_MONTH', 'END_OF_FIS_MONTH');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period End Date', 'FIS_QUARTER', 'END_OF_FIS_QUARTER');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'TIMES_DIM',
'Period End Date', 'FIS_YEAR', 'END_OF_FIS_YEAR');
```

```
dbms_output.put_line('- Period End Day created');

dbms_output.put_line
('Classify entity descriptor use');
 begin
 SELECT descriptor_id INTO time_desc_id
       FROM all_olap_descriptors
       WHERE descriptor_value = 'Time'
       AND descriptor_type = 'Dimension Type';
       begin
           cwm_classify.add_entity_descriptor_use(time_desc_id,
'DIMENSION', USER, 'TIMES_DIM', 'TIMES');
           dbms_output.put_line('- Time dimension');
         exception
           when cwm_exceptions.element_already_exists
              then null;
       end;
 end;

--In this case it is the dimension attribute descriptors that are
being classified
 begin
       SELECT descriptor_id INTO long_desc_id
       FROM all_olap_descriptors
       WHERE descriptor_value = 'Long Description'
       AND descriptor_type = 'Dimensional Attribute Descriptor';
       begin
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
DIMENSION_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'Long Description');
     dbms_output.put_line('- Long description');
           exception
             when cwm_exceptions.element_already_exists
                then null;
           end;
           begin
             cwm_classify.add_entity_descriptor_use(long_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'DAY', 'DAY_NAME');
     dbms_output.put_line('- Day name');
           exception
             when cwm_exceptions.element_already_exists
                then null;
           end;
           begin
             cwm_classify.add_entity_descriptor_use(long_desc_id,
```

```
LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'MONTH', 'CALENDAR_MONTH_
DESC');
   dbms_output.put_line('- Calendar month description');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'QUARTER', 'CALENDAR_
QUARTER_DESC');
   dbms_output.put_line('- Calendar quarter description');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
    LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'FIS_MONTH', 'FISCAL_
MONTH_DESC');
    dbms_output.put_line('- Fiscal month description');
    exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
    LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'FIS_QUARTER', 'FISCAL_
QUARTER_DESC');
    dbms_output.put_line('- Fiscal quarter description');
    exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
       end;
 end;

dbms_output.put_line('- Short Description');
 begin
       SELECT descriptor_id INTO short_desc_id
       FROM all_olap_descriptors
       WHERE descriptor_value = 'Short Description'
       AND descriptor_type = 'Dimensional Attribute Descriptor';
       begin
         begin
```

```
          cwm_classify.add_entity_descriptor_use(short_desc_id,
DIMENSION_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'Short Description');
        exception
          when cwm_exceptions.element_already_exists
             then null;
        end;
        begin
          cwm_classify.add_entity_descriptor_use(short_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'DAY', 'DAY_NAME');
     dbms_output.put_line('- Day name');
        exception
          when cwm_exceptions.element_already_exists
             then null;
        end;
        begin
          cwm_classify.add_entity_descriptor_use(short_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'MONTH', 'CALENDAR_MONTH_
DESC');
   dbms_output.put_line('- Calendar month description');
        exception
          when cwm_exceptions.element_already_exists
             then null;
        end;
        begin
          cwm_classify.add_entity_descriptor_use(short_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'QUARTER', 'CALENDAR_
QUARTER_DESC');
   dbms_output.put_line('- Calendar quarter description');
        exception
          when cwm_exceptions.element_already_exists
             then null;
        end;
        begin
          cwm_classify.add_entity_descriptor_use(short_desc_id,
    LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'FIS_MONTH', 'FISCAL_
MONTH_DESC');
    dbms_output.put_line('- Fiscal month description');
    exception
          when cwm_exceptions.element_already_exists
             then null;
        end;
        begin
          cwm_classify.add_entity_descriptor_use(short_desc_id,
    LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'FIS_QUARTER', 'FISCAL_
QUARTER_DESC');
```

```
    dbms_output.put_line('- Fiscal quarter description');
    exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
       end;
 end;

--- ------------------- Process the CUSTOMERS Dimension ------------

dbms_output.put_line('-');
dbms_output.put_line
('<<<<< CUSTOMERS DIMENSION >>>>>');
dbms_output.put_line
('Dimension - display name, description and plural name');

CWM_OLAP_DIMENSION.set_display_name(USER, 'CUSTOMERS_DIM',
'Customer');
CWM_OLAP_DIMENSION.set_description(USER, 'CUSTOMERS_DIM', 'Customer
Dimension Values');
CWM_OLAP_DIMENSION.set_plural_name(USER, 'CUSTOMERS_DIM',
'Customers');


dbms_output.put_line
('Level - display name and description');

cwm_olap_level.set_display_name(USER, 'CUSTOMERS_DIM', 'CUSTOMER',
'Customer');
cwm_olap_level.set_description(USER, 'CUSTOMERS_DIM', 'CUSTOMER',
'Customer level of standard CUSTOMER hierarchy');

cwm_olap_level.set_display_name(USER, 'CUSTOMERS_DIM', 'CITY',
'City');
cwm_olap_level.set_description(USER, 'CUSTOMERS_DIM', 'CITY', 'City
level of the standard CUSTOMER hierarchy');

cwm_olap_level.set_display_name(USER, 'CUSTOMERS_DIM', 'STATE',
'State');
cwm_olap_level.set_description(USER, 'CUSTOMERS_DIM', 'STATE',
'State level of the standard CUSTOMER hierarchy');

cwm_olap_level.set_display_name(USER, 'CUSTOMERS_DIM', 'COUNTRY',
'Country');
cwm_olap_level.set_description(USER, 'CUSTOMERS_DIM', 'COUNTRY',
```

```
'Country level of the standard CUSTOMER hierarchy');

cwm_olap_level.set_display_name(USER, 'CUSTOMERS_DIM', 'SUBREGION',
'Subregion');
cwm_olap_level.set_description(USER, 'CUSTOMERS_DIM', 'SUBREGION',
'Subregion level of the standard CUSTOMER hierarchy');

cwm_olap_level.set_display_name(USER, 'CUSTOMERS_DIM', 'REGION',
'Region');
cwm_olap_level.set_description(USER, 'CUSTOMERS_DIM', 'REGION',
'Region level of the standard CUSTOMER hierarchy');

cwm_olap_level.set_display_name(USER, 'CUSTOMERS_DIM', 'GEOG_TOTAL',
'Geography Total');
cwm_olap_level.set_description(USER, 'CUSTOMERS_DIM', 'GEOG_TOTAL',
'Geography Total for the standard CUSTOMER hierarchy');

cwm_olap_level.set_display_name(USER, 'CUSTOMERS_DIM', 'CUST_TOTAL',
'Customer Total');
cwm_olap_level.set_description(USER, 'CUSTOMERS_DIM', 'CUST_TOTAL',
'Customer Total for the standard CUSTOMER hierarchy');


dbms_output.put_line
('Hierarchy - display name and description');

cwm_olap_hierarchy.set_display_name(USER, 'CUSTOMERS_DIM', 'GEOG_
ROLLUP', 'Standard');
cwm_olap_hierarchy.set_description(USER, 'CUSTOMERS_DIM', 'GEOG_
ROLLUP', 'Standard GEOGRAPHY hierarchy');

cwm_olap_hierarchy.set_display_name(USER, 'CUSTOMERS_DIM', 'CUST_
ROLLUP', 'Standard');
cwm_olap_hierarchy.set_description(USER, 'CUSTOMERS_DIM', 'CUST_
ROLLUP', 'Standard CUSTOMER hierarchy');


dbms_output.put_line('- default calculation hierarchy');
cwm_olap_cube.set_default_calc_hierarchy(USER,'SALES_CUBE', 'GEOG_
ROLLUP', USER, 'CUSTOMERS_DIM', 'CUSTOMERS_DIM');


dbms_output.put_line('- default display hierarchy');
cwm_olap_dimension.set_default_display_hierarchy(USER, 'CUSTOMERS_
DIM', 'GEOG_ROLLUP');
```

```
dbms_output.put_line
('Level Attributes - name, display name, description');

--Level: CUSTOMER

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'CUSTOMER',
'CUST_FIRST_NAME', 'CUST_FIRST_NAME');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_FIRST_NAME', 'First Name');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_FIRST_NAME', 'Customer First Name');

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'CUSTOMER',
'CUST_LAST_NAME', 'CUST_LAST_NAME');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_LAST_NAME', 'Last Name');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_LAST_NAME', 'Customer Last Name');

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'CUSTOMER',
'CUST_GENDER', 'CUST_GENDER');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_GENDER', 'Gender');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_GENDER', 'Customer Gender');

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'CUSTOMER',
'CUST_MARITAL_STATUS', 'CUST_MARITAL_STATUS');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_MARITAL_STATUS', 'Marital Status');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_MARITAL_STATUS', 'Customer Marital Status');

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'CUSTOMER',
'CUST_YEAR_OF_BIRTH', 'CUST_YEAR_OF_BIRTH');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_YEAR_OF_BIRTH', 'Year of Birth');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_YEAR_OF_BIRTH', 'Customer Year of Birth');

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'CUSTOMER',
'CUST_INCOME_LEVEL', 'CUST_INCOME_LEVEL');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
```

```
'CUSTOMER', 'CUST_INCOME_LEVEL', 'Income Level');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_INCOME_LEVEL', 'Customer Income Level');

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'CUSTOMER',
'CUST_CREDIT_LIMIT', 'CUST_CREDIT_LIMIT');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_CREDIT_LIMIT', 'Credit Limit');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_CREDIT_LIMIT', 'Customer Credit Limit');

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'CUSTOMER',
'CUST_STREET_ADDRESS', 'CUST_STREET_ADDRESS');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_STREET_ADDRESS', 'Street Address');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_STREET_ADDRESS', 'Customer Street Address');

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'CUSTOMER',
'CUST_POSTAL_CODE', 'CUST_POSTAL_CODE');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_POSTAL_CODE', 'Postal Code');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_POSTAL_CODE', 'Customer Postal Code');

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'CUSTOMER',
'CUST_MAIN_PHONE_NUMBER', 'CUST_MAIN_PHONE_NUMBER');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_MAIN_PHONE_NUMBER', 'Main Phone Number');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_MAIN_PHONE_NUMBER', 'Customer Main Phone Number');

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'CUSTOMER',
'CUST_EMAIL', 'CUST_EMAIL');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_EMAIL', 'E-mail');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'CUSTOMER', 'CUST_EMAIL', 'Customer E-mail');

--Level: CITY

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'CITY',
'CUST_CITY', 'CUST_CITY');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'CITY', 'CUST_CITY', 'City');
```

```
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'CITY', 'CUST_CITY', 'City Name');

--Level: STATE

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'STATE',
'CUST_STATE_PROVINCE', 'CUST_STATE_PROVINCE');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'STATE', 'CUST_STATE_PROVINCE', 'State/Province');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'STATE', 'CUST_STATE_PROVINCE', 'State/Province Name');

--Level: SUBREGION

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM',
'SUBREGION', 'COUNTRY_SUBREGION', 'COUNTRY_SUBREGION');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'SUBREGION', 'COUNTRY_SUBREGION', 'Subregion');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'SUBREGION', 'COUNTRY_SUBREGION', 'Subregion Name');

--Level: REGION

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'REGION',
'COUNTRY_REGION', 'COUNTRY_REGION');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'REGION', 'COUNTRY_REGION', 'Region');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'REGION', 'COUNTRY_REGION', 'Region Name');

--Level: COUNTRY

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'COUNTRY',
'COUNTRY_NAME', 'COUNTRY_NAME');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'COUNTRY', 'COUNTRY_NAME', 'Country Name');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'COUNTRY', 'COUNTRY_NAME', 'Country Name');

--Level: GEOGRAPHY TOTAL

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'GEOG_
TOTAL', 'COUNTRY_TOTAL', 'COUNTRY_TOTAL');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'GEOG_TOTAL', 'COUNTRY_TOTAL', 'Country Total');
```

```
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'GEOG_TOTAL', 'COUNTRY_TOTAL', 'Country Total');

--Level: CUSTOMER TOTAL

cwm_olap_level_attribute.set_name(USER, 'CUSTOMERS_DIM', 'CUST_
TOTAL', 'CUST_TOTAL', 'CUST_TOTAL');
cwm_olap_level_attribute.set_display_name(USER, 'CUSTOMERS_DIM',
'CUST_TOTAL', 'CUST_TOTAL', 'Customer Total');
cwm_olap_level_attribute.set_description(USER, 'CUSTOMERS_DIM',
'CUST_TOTAL', 'CUST_TOTAL', 'Customer Total');


dbms_output.put_line
('Drop dimension attributes prior to re-creation');

 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER,
'CUSTOMERS_DIM', 'Long Description');
    dbms_output.put_line('- Long Description dropped');
 exception
    when cwm_exceptions.attribute_not_found then
      null;
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER,
'CUSTOMERS_DIM', 'Short Description' );
    dbms_output.put_line('- Short Description dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('      No attribute to drop');
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER,
'CUSTOMERS_DIM', 'First Name');
    dbms_output.put_line('- First Name dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('      No attribute to drop');
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER,
'CUSTOMERS_DIM', 'Last Name');
    dbms_output.put_line('- Last Name dropped');
 exception
```

```
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('      No attribute to drop');
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER,
'CUSTOMERS_DIM', 'Gender');
    dbms_output.put_line('- Gender dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('      No attribute to drop');
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER,
'CUSTOMERS_DIM', 'Marital Status');
    dbms_output.put_line('- Marital Status dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('      No attribute to drop');
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER,
'CUSTOMERS_DIM', 'Year of Birth');
    dbms_output.put_line('- Year of Birth dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('      No attribute to drop');
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER,
'CUSTOMERS_DIM', 'Income Level');
    dbms_output.put_line('- Income Level dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('      No attribute to drop');
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER,
'CUSTOMERS_DIM', 'Credit Limit');
    dbms_output.put_line('- Credit Limit dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('No attribute to drop');
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER,
```

```
'CUSTOMERS_DIM', 'Street Address');
    dbms_output.put_line('- Street Address dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('      No attribute to drop');
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER,
'CUSTOMERS_DIM', 'Postal Code');
    dbms_output.put_line('- Postal Code dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('      No attribute to drop');
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER,
'CUSTOMERS_DIM', 'Phone Number');
    dbms_output.put_line('- Phone Number dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('No attribute to drop');
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER,
'CUSTOMERS_DIM', 'E-mail');
    dbms_output.put_line('- E-mail dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('No attribute to drop');
 end;

dbms_output.put_line
('Create dimension attributes and add their level attributes');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'CUSTOMERS_
DIM', 'Long Description', 'Customer Information', 'Long Description
of Customer Information');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Long Description', 'CUSTOMER', 'CUST_LAST_NAME');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Long Description', 'CITY', 'CUST_CITY');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Long Description', 'STATE', 'CUST_STATE_PROVINCE');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Long Description', 'COUNTRY', 'COUNTRY_NAME');
```

```
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Long Description', 'SUBREGION', 'COUNTRY_SUBREGION');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Long Description', 'REGION', 'COUNTRY_REGION');
dbms_output.put_line('- Long Description created');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'CUSTOMERS_
DIM', 'Short Description', 'Customer Information', 'Short
Description of Customer Information');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Short Description', 'CUSTOMER', 'CUST_LAST_NAME');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Short Description', 'CITY', 'CUST_CITY');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Short Description', 'STATE', 'CUST_STATE_PROVINCE');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Short Description', 'COUNTRY', 'COUNTRY_NAME');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Short Description', 'SUBREGION', 'COUNTRY_SUBREGION');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Short Description', 'REGION', 'COUNTRY_REGION');
dbms_output.put_line('- Short Description created');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'CUSTOMERS_
DIM', 'First Name', 'First Name', 'First Name');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'First Name', 'CUSTOMER', 'CUST_FIRST_NAME');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'CUSTOMERS_
DIM', 'Last Name', 'Last Name', 'Last Name');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Last Name', 'CUSTOMER', 'CUST_LAST_NAME');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'CUSTOMERS_
DIM', 'Gender', 'Gender', 'Gender');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Gender', 'CUSTOMER', 'CUST_GENDER');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'CUSTOMERS_
DIM', 'Marital Status', 'Marital Status', 'Marital Status');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Marital Status', 'CUSTOMER', 'CUST_MARITAL_STATUS');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'CUSTOMERS_
DIM', 'Year of Birth', 'Year of Birth', 'Year of Birth');
```

```
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Year of Birth', 'CUSTOMER', 'CUST_YEAR_OF_BIRTH');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'CUSTOMERS_
DIM', 'Income Level', 'Income Level', 'Income Level');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Income Level', 'CUSTOMER', 'CUST_INCOME_LEVEL');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'CUSTOMERS_
DIM', 'Credit Limit', 'Credit Limit', 'Credit Limit');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Credit Limit', 'CUSTOMER', 'CUST_CREDIT_LIMIT');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'CUSTOMERS_
DIM', 'Street Address', 'Street Address', 'Street Address');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Street Address', 'CUSTOMER', 'CUST_STREET_ADDRESS');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'CUSTOMERS_
DIM', 'Postal Code', 'Postal Code', 'Postal Code');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Postal Code', 'CUSTOMER', 'CUST_POSTAL_CODE');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'CUSTOMERS_
DIM', 'Phone Number', 'Phone Number', 'Phone Number');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'Phone Number', 'CUSTOMER', 'CUST_MAIN_PHONE_NUMBER');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'CUSTOMERS_
DIM', 'E-mail', 'E-mail', 'E-mail');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CUSTOMERS_DIM',
'E-mail', 'CUSTOMER', 'CUST_EMAIL');
dbms_output.put_line('- Other Customer Information created');


dbms_output.put_line
('Classify entity descriptor use');
 begin
       SELECT descriptor_id INTO long_desc_id
       FROM all_olap_descriptors
       WHERE descriptor_value = 'Long Description'
       AND descriptor_type = 'Dimensional Attribute Descriptor';
       begin
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
```

```
DIMENSION_ATTRIBUTE_TYPE, USER, 'CUSTOMERS_DIM', 'Long
Description');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'CUSTOMERS_DIM', 'CUSTOMER', 'CUST_LAST_
NAME');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'CUSTOMERS_DIM', 'CITY', 'CUST_CITY');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'CUSTOMERS_DIM', 'STATE', 'CUST_STATE_
PROVINCE');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'CUSTOMERS_DIM', 'COUNTRY', 'COUNTRY_
NAME');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'CUSTOMERS_DIM', 'SUBREGION', 'COUNTRY_
SUBREGION');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
```

```
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'CUSTOMERS_DIM', 'REGION', 'COUNTRY_
REGION');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
       end;
 dbms_output.put_line('- Long Description');
 end;

begin
       SELECT descriptor_id INTO short_desc_id
       FROM all_olap_descriptors
       WHERE descriptor_value = 'Short Description'
       AND descriptor_type = 'Dimensional Attribute Descriptor';
       begin
         begin
           cwm_classify.add_entity_descriptor_use(short_desc_id,
DIMENSION_ATTRIBUTE_TYPE, USER, 'CUSTOMERS_DIM', 'Short
Description');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(short_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'CUSTOMERS_DIM', 'CUSTOMER', 'CUST_LAST_
NAME');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(short_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'CUSTOMERS_DIM', 'CITY', 'CUST_CITY');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(short_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'CUSTOMERS_DIM', 'STATE', 'CUST_STATE_
PROVINCE');
```

```
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(short_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'CUSTOMERS_DIM', 'COUNTRY', 'COUNTRY_
NAME');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(short_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'CUSTOMERS_DIM', 'SUBREGION', 'COUNTRY_
SUBREGION');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(short_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'CUSTOMERS_DIM', 'REGION', 'COUNTRY_
REGION');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
       end;
 dbms_output.put_line('- Short Description');
 end;

-- ------------------- Process the PRODUCT Dimension --------------

dbms_output.put_line('-');
dbms_output.put_line
('<<<<< PRODUCTS DIMENSION >>>>>');
dbms_output.put_line
('Dimension - display name, description and plural name');

CWM_OLAP_DIMENSION.set_display_name(USER, 'PRODUCTS_DIM',
'Product');
CWM_OLAP_DIMENSION.set_description(USER, 'PRODUCTS_DIM', 'Product
Dimension Values');
CWM_OLAP_DIMENSION.set_plural_name(USER, 'PRODUCTS_DIM',
```

```
'Products');


dbms_output.put_line
('Level - display name and description');

cwm_olap_level.set_display_name(USER, 'PRODUCTS_DIM', 'PRODUCT',
'Products');
cwm_olap_level.set_description(USER, 'PRODUCTS_DIM', 'PRODUCT',
'Product level of standard PRODUCT hierarchy');

cwm_olap_level.set_display_name(USER, 'PRODUCTS_DIM', 'SUBCATEGORY',
'Sub-categories');
cwm_olap_level.set_description(USER, 'PRODUCTS_DIM', 'SUBCATEGORY',
'Sub-category level of standard PRODUCT hierarchy');

cwm_olap_level.set_display_name(USER, 'PRODUCTS_DIM', 'CATEGORY',
'Categories');
cwm_olap_level.set_description(USER, 'PRODUCTS_DIM', 'CATEGORY',
'Category level of standard PRODUCT hierarchy');

cwm_olap_level.set_display_name(USER, 'PRODUCTS_DIM', 'PROD_TOTAL',
'Product Total');
cwm_olap_level.set_description(USER, 'PRODUCTS_DIM', 'PROD_TOTAL',
'Product Total for the standard PRODUCT hierarchy');


dbms_output.put_line
('Hierarchy - display name and description');

cwm_olap_hierarchy.set_display_name(USER, 'PRODUCTS_DIM', 'PROD_
ROLLUP', 'Standard');
cwm_olap_hierarchy.set_description(USER, 'PRODUCTS_DIM', 'PROD_
ROLLUP', 'Standard Product hierarchy');


dbms_output.put_line('- default calculation hierarchy');
cwm_olap_cube.set_default_calc_hierarchy(USER,'SALES_CUBE', 'PROD_
ROLLUP', USER, 'PRODUCTS_DIM', 'PRODUCTS_DIM');
cwm_olap_cube.set_default_calc_hierarchy(USER,'COST_CUBE', 'PROD_
ROLLUP', USER, 'PRODUCTS_DIM', 'PRODUCTS_DIM');


dbms_output.put_line('- default display hierarchy');
cwm_olap_dimension.set_default_display_hierarchy(USER, 'PRODUCTS_
```

```
DIM', 'PROD_ROLLUP');


dbms_output.put_line
('Level Attributes - name, display name, description');

--Level: PRODUCT
cwm_olap_level_attribute.set_name(USER, 'PRODUCTS_DIM', 'PRODUCT',
'PROD_NAME', 'PROD_NAME');
cwm_olap_level_attribute.set_display_name(USER, 'PRODUCTS_DIM',
'PRODUCT', 'PROD_NAME',  'Product Name(s)');
cwm_olap_level_attribute.set_description(USER, 'PRODUCTS_DIM',
'PRODUCT', 'PROD_NAME',  'Names for Product values of the Standard
Product hierarchy');

cwm_olap_level_attribute.set_name(USER, 'PRODUCTS_DIM', 'PRODUCT',
'PROD_DESC', 'PROD_DESC');
cwm_olap_level_attribute.set_display_name(USER, 'PRODUCTS_DIM',
'PRODUCT', 'PROD_DESC',  'Product Description');
cwm_olap_level_attribute.set_description(USER, 'PRODUCTS_DIM',
'PRODUCT', 'PROD_DESC',  'Product Description including
characteristics of the product');

cwm_olap_level_attribute.set_name(USER, 'PRODUCTS_DIM', 'PRODUCT',
'PROD_WEIGHT_CLASS', 'PROD_WEIGHT_CLASS');
cwm_olap_level_attribute.set_display_name(USER, 'PRODUCTS_DIM',
'PRODUCT', 'PROD_WEIGHT_CLASS',  'Weight Class');
cwm_olap_level_attribute.set_description(USER, 'PRODUCTS_DIM',
'PRODUCT', 'PROD_WEIGHT_CLASS',  'Product Weight Class');

cwm_olap_level_attribute.set_name(USER, 'PRODUCTS_DIM', 'PRODUCT',
'PROD_UNIT_OF_MEASURE', 'PROD_UNIT_OF_MEASURE');
cwm_olap_level_attribute.set_display_name(USER, 'PRODUCTS_DIM',
'PRODUCT', 'PROD_UNIT_OF_MEASURE',  'Unit of Measure');
cwm_olap_level_attribute.set_description(USER, 'PRODUCTS_DIM',
'PRODUCT', 'PROD_UNIT_OF_MEASURE',  'Product Unit of Measure');

--Level: SUBCATEGORY
cwm_olap_level_attribute.set_name(USER, 'PRODUCTS_DIM',
'SUBCATEGORY', 'PROD_SUBCATEGORY', 'PROD_SUBCATEGORY');
cwm_olap_level_attribute.set_display_name(USER, 'PRODUCTS_DIM',
'SUBCATEGORY', 'PROD_SUBCATEGORY',  'Sub-category');
cwm_olap_level_attribute.set_description(USER, 'PRODUCTS_DIM',
'SUBCATEGORY', 'PROD_SUBCATEGORY',  'Product Sub-category');
```

```
cwm_olap_level_attribute.set_name(USER, 'PRODUCTS_DIM',
'SUBCATEGORY', 'PROD_SUBCAT_DESC', 'PROD_SUBCAT_DESC');
cwm_olap_level_attribute.set_display_name(USER, 'PRODUCTS_DIM',
'SUBCATEGORY', 'PROD_SUBCAT_DESC',  'Sub-category Description');
cwm_olap_level_attribute.set_description(USER, 'PRODUCTS_DIM',
'SUBCATEGORY', 'PROD_SUBCAT_DESC',  'Product Sub-category
Description');

--Level: CATEGORY
cwm_olap_level_attribute.set_name(USER, 'PRODUCTS_DIM', 'CATEGORY',
'PROD_CATEGORY', 'PROD_CATEGORY');
cwm_olap_level_attribute.set_display_name(USER, 'PRODUCTS_DIM',
'CATEGORY', 'PROD_CATEGORY',  'Category');
cwm_olap_level_attribute.set_description(USER, 'PRODUCTS_DIM',
'CATEGORY', 'PROD_CATEGORY',  'Product category');

cwm_olap_level_attribute.set_name(USER, 'PRODUCTS_DIM', 'CATEGORY',
'PROD_CAT_DESC', 'PROD_CAT_DESC');
cwm_olap_level_attribute.set_display_name(USER, 'PRODUCTS_DIM',
'CATEGORY', 'PROD_CAT_DESC',  'Category Description');
cwm_olap_level_attribute.set_description(USER, 'PRODUCTS_DIM',
'CATEGORY', 'PROD_CAT_DESC',  'Product Category Description');

--Level: PRODUCT TOTAL
cwm_olap_level_attribute.set_name(USER, 'PRODUCTS_DIM', 'PROD_
TOTAL', 'PROD_TOTAL', 'PROD_TOTAL');
cwm_olap_level_attribute.set_display_name(USER, 'PRODUCTS_DIM',
'PROD_TOTAL', 'PROD_TOTAL',  'Product Total');
cwm_olap_level_attribute.set_description(USER, 'PRODUCTS_DIM',
'PROD_TOTAL', 'PROD_TOTAL',  'Product Total');


dbms_output.put_line
('Drop dimension attributes prior to re-creation');

 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER, 'PRODUCTS_
DIM', 'Long Description');
    dbms_output.put_line('- Long Description dropped');
 exception
    when cwm_exceptions.attribute_not_found then
      null;
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER, 'PRODUCTS_
```

```
DIM', 'Short Description' );
    dbms_output.put_line('- Short Description dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('No attribute to drop');
 end;


dbms_output.put_line
('Create dimension attributes and add their level attributes');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'PRODUCTS_
DIM', 'Long Description', 'Long Product Description', 'Full
Description of Products');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'PRODUCTS_DIM',
'Long Description', 'PRODUCT', 'PROD_DESC');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'PRODUCTS_DIM',
'Long Description', 'SUBCATEGORY', 'PROD_SUBCAT_DESC');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'PRODUCTS_DIM',
'Long Description', 'CATEGORY', 'PROD_CAT_DESC');
dbms_output.put_line('- Long Description created');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'PRODUCTS_
DIM', 'Short Description', 'Short Product Names', 'Short name of
Products');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'PRODUCTS_DIM',
'Short Description', 'PRODUCT', 'PROD_NAME');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'PRODUCTS_DIM',
'Short Description', 'SUBCATEGORY', 'PROD_SUBCAT_DESC');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'PRODUCTS_DIM',
'Short Description', 'CATEGORY', 'PROD_CAT_DESC');
dbms_output.put_line('- Short Description created');


dbms_output.put_line
('Classify entity descriptor use');

 begin
       SELECT descriptor_id INTO long_desc_id
       FROM all_olap_descriptors
       WHERE descriptor_value = 'Long Description'
       AND descriptor_type = 'Dimensional Attribute Descriptor';
       begin
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
```

```
DIMENSION_ATTRIBUTE_TYPE, USER, 'PRODUCTS_DIM', 'Long Description');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'PRODUCTS_DIM', 'PRODUCT', 'PROD_DESC');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'PRODUCTS_DIM', 'SUBCATEGORY', 'PROD_
SUBCAT_DESC');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'PRODUCTS_DIM', 'CATEGORY', 'PROD_CAT_
DESC');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
       end;
   dbms_output.put_line('- Long Description');
 end;

 begin
       SELECT descriptor_id INTO short_desc_id
       FROM all_olap_descriptors
       WHERE descriptor_value = 'Short Description'
       AND descriptor_type = 'Dimensional Attribute Descriptor';
       begin
         begin
           cwm_classify.add_entity_descriptor_use(short_desc_id,
DIMENSION_ATTRIBUTE_TYPE, USER, 'PRODUCTS_DIM', 'Short
Description');
         exception
           when cwm_exceptions.element_already_exists
              then null;
```

```
         end;
         begin
           cwm_classify.add_entity_descriptor_use(short_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'PRODUCTS_DIM', 'PRODUCT', 'PROD_DESC');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(short_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'PRODUCTS_DIM', 'SUBCATEGORY', 'PROD_
SUBCAT_DESC');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(short_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'PRODUCTS_DIM', 'CATEGORY', 'PROD_CAT_
DESC');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
       end;
 dbms_output.put_line('- Short Description');
 end;

-- ------------------- Process the PROMOTIONS Dimension
--------------

dbms_output.put_line('-');
dbms_output.put_line
('<<<<< PROMOTIONS DIMENSION >>>>>');
dbms_output.put_line
('Dimension - display name, description and plural name');

CWM_OLAP_DIMENSION.set_display_name(USER, 'PROMOTIONS_DIM',
'Promotions');
CWM_OLAP_DIMENSION.set_description(USER, 'PROMOTIONS_DIM',
'Promotion Values');
CWM_OLAP_DIMENSION.set_plural_name(USER, 'PROMOTIONS_DIM',
'Promotions');
```

```
dbms_output.put_line
('Level - display name and description');

cwm_olap_level.set_display_name(USER, 'PROMOTIONS_DIM', 'PROMO',
'Promotions');
cwm_olap_level.set_description(USER, 'PROMOTIONS_DIM', 'PROMO',
'Promotion level of the standard PROMOTION hierarchy');

cwm_olap_level.set_display_name(USER, 'PROMOTIONS_DIM',
'SUBCATEGORY', 'Promotions Sub-categories');
cwm_olap_level.set_description(USER, 'PROMOTIONS_DIM',
'SUBCATEGORY', 'Sub-category level of the standard PROMOTION
hierarchy');

cwm_olap_level.set_display_name(USER, 'PROMOTIONS_DIM', 'CATEGORY',
'Promotions Categories');
cwm_olap_level.set_description(USER, 'PROMOTIONS_DIM', 'CATEGORY',
'Category level of the standard PROMOTION hierarchy');

cwm_olap_level.set_display_name(USER, 'PROMOTIONS_DIM', 'PROMO_
TOTAL', 'Promotions Total');
cwm_olap_level.set_description(USER, 'PROMOTIONS_DIM', 'PROMO_
TOTAL', 'Promotions Total for the standard PROMOTION hierarchy');


dbms_output.put_line
('Hierarchy - display name and description');

cwm_olap_hierarchy.set_display_name(USER, 'PROMOTIONS_DIM', 'PROMO_
ROLLUP', 'Standard Promotions');
cwm_olap_hierarchy.set_description(USER, 'PROMOTIONS_DIM', 'PROMO_
ROLLUP', 'Standard Promotions hierarchy');


dbms_output.put_line('- default calculation hierarchy');
cwm_olap_cube.set_default_calc_hierarchy(USER,'SALES_CUBE', 'PROMO_
ROLLUP', USER, 'PROMOTIONS_DIM', 'PROMOTIONS_DIM');


dbms_output.put_line('- default display hierarchy');
cwm_olap_dimension.set_default_display_hierarchy(USER, 'PROMOTIONS_
DIM', 'PROMO_ROLLUP');


dbms_output.put_line
```

```
('Level Attributes - name, display name, description');

--Level: PROMO
cwm_olap_level_attribute.set_name(USER, 'PROMOTIONS_DIM', 'PROMO',
'PROMO_NAME', 'PROMO_NAME');
cwm_olap_level_attribute.set_display_name(USER, 'PROMOTIONS_DIM',
'PROMO', 'PROMO_NAME', 'Promotion Name(s)');
cwm_olap_level_attribute.set_description(USER, 'PROMOTIONS_DIM',
'PROMO', 'PROMO_NAME', 'Names for the Promotions in the Standard
Promotions hierarchy');

cwm_olap_level_attribute.set_name(USER, 'PROMOTIONS_DIM', 'PROMO',
'PROMO_COST', 'PROMO_COST');
cwm_olap_level_attribute.set_display_name(USER, 'PROMOTIONS_DIM',
'PROMO', 'PROMO_COST', 'Promotion costs');
cwm_olap_level_attribute.set_description(USER, 'PROMOTIONS_DIM',
'PROMO', 'PROMO_COST', 'Promotion costs');

cwm_olap_level_attribute.set_name(USER, 'PROMOTIONS_DIM', 'PROMO',
'PROMO_BEGIN_DATE', 'PROMO_BEGIN_DATE');
cwm_olap_level_attribute.set_display_name(USER, 'PROMOTIONS_DIM',
'PROMO', 'PROMO_BEGIN_DATE', 'Begin date');
cwm_olap_level_attribute.set_description(USER, 'PROMOTIONS_DIM',
'PROMO', 'PROMO_BEGIN_DATE', 'Promotion Begin Date');

cwm_olap_level_attribute.set_name(USER, 'PROMOTIONS_DIM', 'PROMO',
'PROMO_END_DATE', 'PROMO_END_DATE');
cwm_olap_level_attribute.set_display_name(USER, 'PROMOTIONS_DIM',
'PROMO', 'PROMO_END_DATE', 'End date');
cwm_olap_level_attribute.set_description(USER, 'PROMOTIONS_DIM',
'PROMO', 'PROMO_END_DATE', 'Promotion End Date');

--Level: SUBCATEGORY
cwm_olap_level_attribute.set_name(USER, 'PROMOTIONS_DIM',
'SUBCATEGORY', 'PROMO_SUBCATEGORY', 'PROMO_SUBCATEGORY');
cwm_olap_level_attribute.set_display_name(USER, 'PROMOTIONS_DIM',
'SUBCATEGORY', 'PROMO_SUBCATEGORY', 'Sub-Category');
cwm_olap_level_attribute.set_description(USER, 'PROMOTIONS_DIM',
'SUBCATEGORY', 'PROMO_SUBCATEGORY', 'Promotion Sub-Category');

--Level: CATEGORY
cwm_olap_level_attribute.set_name(USER, 'PROMOTIONS_DIM',
'CATEGORY', 'PROMO_CATEGORY', 'PROMO_CATEGORY');
cwm_olap_level_attribute.set_display_name(USER, 'PROMOTIONS_DIM',
'CATEGORY', 'PROMO_CATEGORY', 'Category');
```

```
cwm_olap_level_attribute.set_description(USER, 'PROMOTIONS_DIM',
'CATEGORY', 'PROMO_CATEGORY', 'Promotion Category');

--Level: PROMOTIONS TOTAL
cwm_olap_level_attribute.set_name(USER, 'PROMOTIONS_DIM', 'PROMO_
TOTAL', 'PROMO_TOTAL', 'PROMO_TOTAL');
cwm_olap_level_attribute.set_display_name(USER, 'PROMOTIONS_DIM',
'PROMO_TOTAL', 'PROMO_TOTAL', 'Promotions Total');
cwm_olap_level_attribute.set_description(USER, 'PROMOTIONS_DIM',
'PROMO_TOTAL', 'PROMO_TOTAL', 'Promotions Total');


dbms_output.put_line
('Drop dimension attributes prior to re-creation');

 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER,
'PROMOTIONS_DIM', 'Long Description');
    dbms_output.put_line('- Long Description dropped');
 exception
    when cwm_exceptions.attribute_not_found then
      null;
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER,
'PROMOTIONS_DIM', 'Short Description' );
    dbms_output.put_line('- Short Description dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('No attribute to drop');
 end;


dbms_output.put_line
('Create dimension attributes and add their level attributes');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'PROMOTIONS_
DIM', 'Long Description', 'Long Description of Promotions', 'Long
Description of Promotions');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'PROMOTIONS_DIM',
'Long Description', 'PROMO', 'PROMO_NAME');
dbms_output.put_line('- Long Description created');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'PROMOTIONS_
DIM', 'Short Description', 'ShortDescription of Promotions', 'Short
```

```
Description of Promotions');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'PROMOTIONS_DIM',
'Short Description', 'PROMO', 'PROMO_NAME');
dbms_output.put_line('- Short Description created');


dbms_output.put_line
('Classify entity descriptor use');

 begin
       SELECT descriptor_id INTO long_desc_id
       FROM all_olap_descriptors
       WHERE descriptor_value = 'Long Description'
       AND descriptor_type = 'Dimensional Attribute Descriptor';
       begin
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
DIMENSION_ATTRIBUTE_TYPE, USER, 'PROMOTIONS_DIM', 'Long
Description');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'PROMOTIONS_DIM', 'PROMO', 'PROMO_
NAME');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
       end;
   dbms_output.put_line('- Long Description');
 end;

 begin
       SELECT descriptor_id INTO short_desc_id
       FROM all_olap_descriptors
       WHERE descriptor_value = 'Short Description'
       AND descriptor_type = 'Dimensional Attribute Descriptor';
       begin
         begin
           cwm_classify.add_entity_descriptor_use(short_desc_id,
DIMENSION_ATTRIBUTE_TYPE, USER, 'PROMOTIONS_DIM', 'Short
Description');
```

```
          exception
            when cwm_exceptions.element_already_exists
               then null;
          end;
          begin
            cwm_classify.add_entity_descriptor_use(short_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'PROMOTIONS_DIM', 'PROMO', 'PROMO_
NAME');
          exception
            when cwm_exceptions.element_already_exists
               then null;
          end;
        end;
 dbms_output.put_line('- Short Description');
 end;


-- ------------------- Process the CHANNELS Dimension --------------

dbms_output.put_line('-');
dbms_output.put_line
('<<<<< CHANNELS DIMENSION >>>>>');
dbms_output.put_line
('Dimension - display name, description and plural name');

CWM_OLAP_DIMENSION.set_display_name(USER, 'CHANNELS_DIM',
'Channel');
CWM_OLAP_DIMENSION.set_description(USER, 'CHANNELS_DIM', 'Channel
Values');
CWM_OLAP_DIMENSION.set_plural_name(USER, 'CHANNELS_DIM',
'Channels');


dbms_output.put_line
('Level - display name and description');

cwm_olap_level.set_display_name(USER, 'CHANNELS_DIM', 'CHANNEL',
'Channel');
cwm_olap_level.set_description(USER, 'CHANNELS_DIM', 'CHANNEL',
'Channel level of the standard hierarchy');

cwm_olap_level.set_display_name(USER, 'CHANNELS_DIM', 'CHANNEL_
CLASS', 'Channel Class');
cwm_olap_level.set_description(USER, 'CHANNELS_DIM', 'CHANNEL_
CLASS', 'Channel Class level of the standard hierarchy');
```

```
cwm_olap_level.set_display_name(USER, 'CHANNELS_DIM', 'CHANNEL_
TOTAL', 'Channel Total');
cwm_olap_level.set_description(USER, 'CHANNELS_DIM', 'CHANNEL_
TOTAL', 'Channel Total for the standard hierarchy');


dbms_output.put_line
('Hierarchy - display name and description');

cwm_olap_hierarchy.set_display_name(USER, 'CHANNELS_DIM', 'CHANNEL_
ROLLUP', 'Standard Channels');
cwm_olap_hierarchy.set_description(USER, 'CHANNELS_DIM', 'CHANNEL_
ROLLUP', 'Standard Channels hierarchy');


dbms_output.put_line('- default calculation hierarchy');
cwm_olap_cube.set_default_calc_hierarchy(USER,'SALES_CUBE',
'CHANNEL_ROLLUP', USER, 'CHANNELS_DIM', 'CHANNELS_DIM');


dbms_output.put_line('- default display hierarchy');
cwm_olap_dimension.set_default_display_hierarchy(USER, 'CHANNELS_
DIM', 'CHANNEL_ROLLUP');


dbms_output.put_line
('Level Attributes - name, display name, description');

--Level: CHANNEL
cwm_olap_level_attribute.set_name(USER, 'CHANNELS_DIM', 'CHANNEL',
'CHANNEL_DESC', 'CHANNEL_DESC');
cwm_olap_level_attribute.set_display_name(USER, 'CHANNELS_DIM',
'CHANNEL', 'CHANNEL_DESC', 'Channel');
cwm_olap_level_attribute.set_description(USER, 'CHANNELS_DIM',
'CHANNEL', 'CHANNEL_DESC', 'Channel Description');

--Level: CHANNEL CLASS
cwm_olap_level_attribute.set_name(USER, 'CHANNELS_DIM', 'CHANNEL_
CLASS', 'CHANNEL_CLASS', 'CHANNEL_CLASS');
cwm_olap_level_attribute.set_display_name(USER, 'CHANNELS_DIM',
'CHANNEL_CLASS', 'CHANNEL_CLASS', 'Channel Class');
cwm_olap_level_attribute.set_description(USER, 'CHANNELS_DIM',
'CHANNEL_CLASS', 'CHANNEL_CLASS', 'Channel Class Identifier');
```

```
--Level: CHANNEL TOTAL
cwm_olap_level_attribute.set_name(USER, 'CHANNELS_DIM', 'CHANNEL_
TOTAL', 'CHANNEL_TOTAL', 'CHANNEL_TOTAL');
cwm_olap_level_attribute.set_display_name(USER, 'CHANNELS_DIM',
'CHANNEL_TOTAL', 'CHANNEL_TOTAL', 'Channel Total');
cwm_olap_level_attribute.set_description(USER, 'CHANNELS_DIM',
'CHANNEL_TOTAL', 'CHANNEL_TOTAL', 'Channel Total');


dbms_output.put_line
('Drop dimension attributes prior to re-creation');

 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER, 'CHANNELS_
DIM', 'Long Description');
    dbms_output.put_line('- Long Description dropped');
 exception
    when cwm_exceptions.attribute_not_found then
      null;
 end;
 begin
    cwm_olap_dim_attribute.drop_dimension_attribute(USER, 'CHANNELS_
DIM', 'Short Description' );
    dbms_output.put_line('- Short Description dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       dbms_output.put_line('No attribute to drop');
 end;


dbms_output.put_line
('Create dimension attributes and add their level attributes');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'CHANNELS_
DIM', 'Long Description', 'Long Description of Channels', 'Long
Description of Channels');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CHANNELS_DIM',
'Long Description', 'CHANNEL', 'CHANNEL_DESC');
dbms_output.put_line('- Long Description created');

CWM_OLAP_DIM_ATTRIBUTE.create_dimension_attribute(USER, 'CHANNELS_
DIM', 'Short Description', 'Short Description of Channels', 'Short
Description of Channels');
  CWM_OLAP_DIM_ATTRIBUTE.add_level_attribute(USER, 'CHANNELS_DIM',
'Short Description', 'CHANNEL', 'CHANNEL_DESC');
```

```
dbms_output.put_line('- Short Description created');


dbms_output.put_line
('Classify entity descriptor use');

 begin
       SELECT descriptor_id INTO long_desc_id
       FROM all_olap_descriptors
       WHERE descriptor_value = 'Long Description'
       AND descriptor_type = 'Dimensional Attribute Descriptor';
       begin
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
DIMENSION_ATTRIBUTE_TYPE, USER, 'CHANNELS_DIM', 'Long Description');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
         begin
           cwm_classify.add_entity_descriptor_use(long_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'CHANNELS_DIM', 'CHANNEL', 'CHANNEL_
DESC');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
       end;
   dbms_output.put_line('- Long Description');
 end;

 begin
       SELECT descriptor_id INTO short_desc_id
       FROM all_olap_descriptors
       WHERE descriptor_value = 'Short Description'
       AND descriptor_type = 'Dimensional Attribute Descriptor';
       begin
         begin
           cwm_classify.add_entity_descriptor_use(short_desc_id,
DIMENSION_ATTRIBUTE_TYPE, USER, 'CHANNELS_DIM', 'Short
Description');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
```

```
         begin
           cwm_classify.add_entity_descriptor_use(short_desc_id,
LEVEL_ATTRIBUTE_TYPE, USER, 'CHANNELS_DIM', 'CHANNEL', 'CHANNEL_
DESC');
         exception
           when cwm_exceptions.element_already_exists
              then null;
         end;
       end;
 dbms_output.put_line('- Short Description');
 end;


-- ------------------- Final Processing
-------------------------------

dbms_output.put_line('-');
dbms_output.put_line
('<<<<< FINAL PROCESSING >>>>>');
commit;
dbms_output.put_line
('- Changes have been committed');
exception
  when others then
    cwm_utility.dump_error;
    errtxt := cwm_utility.get_last_error_description;
    dbms_output.put_line('ERROR: ' || errtxt);
    rollback;
    raise;
end;
.
/

COMMIT;
```

sh_olp_d.sql

```
Rem
Rem $Header: sh_olp_d.sql 05-mar-2001.13:53:15 ahunold Exp $
Rem
Rem sh_olp_d.sql
Rem
Rem  Copyright (c) Oracle Corporation 2001. All Rights Reserved.
```

```
Rem
Rem    NAME
Rem      sh_olp_d.sql - Drop columns used by OLAP Server
Rem
Rem    DESCRIPTION
Rem      SH is the Sales History schema of the Oracle 9i Sample
Rem   Schemas
Rem
Rem    NOTES
Rem
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    ahunold    04/05/01 - dimension names
Rem    ahunold    03/05/01 - external table, no DROPs
Rem    ahunold    02/07/01 - CMWLite
Rem    ahunold    02/01/01 - Merged ahunold_two_facts
Rem    hbaer      01/29/01 - Created
Rem

ALTER TABLE products
DROP COLUMN prod_total;

ALTER TABLE customers
DROP COLUMN cust_total;

ALTER TABLE promotions
DROP COLUMN promo_total;

ALTER TABLE channels
DROP COLUMN channel_total;

ALTER TABLE countries
DROP COLUMN country_total;

COMMIT;

REM redefinition of original dimensions

DROP DIMENSION times_dim;

DROP DIMENSION customers_dim;

DROP DIMENSION products_dim;

DROP DIMENSION promotions_dim;
```

```
DROP DIMENSION promotions_dim;

DROP DIMENSION channels_dim;
DROP DIMENSION channels_dim;

@@sh_hiera
```