Oracle9*i*

サンプル・スキーマ

リリース 2（9.2）

2002 年 7 月

部品番号：J06302-01

Oracle9*i* サンプル・スキーマ , リリース 2（9.2）

部品番号：J06302-01

原本名：Oracle9*i* Sample Schemas, Release 2 (9.2)

原本部品番号：A96539-01

原本協力者：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, and Susan Mavris

Printed in Japan.

制限付権利の説明

プログラム（ソフトウェアおよびドキュメントを含む）の使用、複製または開示は、オラクル社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されております。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

* オラクル社とは、Oracle Corporation（米国オラクル）または日本オラクル株式会社（日本オラクル）を指します。

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておりません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である Oracle Corporation（米国オラクル）およびその関連会社は一切責任を負いかねます。当プログラムを米国国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Notice が適用されます。

Restricted Rights Notice

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

目次

4 Oracle9*i* サンプル・スキーマのスクリプト

はじめに

Oracle では、スキーマ `SCOTT` を 2 つの重要な表 `EMP` 表および `DEPT` 表とともに長い間使用しています。Oracle のテクノロジの進歩につれて、これらの表は、Oracle データベースおよび他の Oracle 製品の最も基本的な機能を表すのに不十分になってきました。その結果、製品マニュアル、教育用ソフトウェア、ソフトウェア開発およびアプリケーションのデモの要件を満たすために、他の多くのスキーマが長年にわたって作成されてきました。

ここで説明する項目は、次のとおりです。

- 対象読者
- サンプル・スキーマについて
- サンプル・スキーマにより提供される利点
- Oracle9i サンプル・スキーマの設計方針
- このマニュアルの構成
- 関連文書
- 表記規則

対象読者

このマニュアルは、Oracle をインストールするとインストールされるシード・データベースを使用する、すべてのユーザーを対象としています。

サンプル・スキーマについて

新しい Oracle9*i* サンプル・スキーマは、Oracle9*i* および将来のリリースにおけるサンプルを共通化することを目的としています。オラクル社マニュアルのすべてのサンプルを一度にこの新しい環境に変換することはできませんが、資料の更新時にすべてのサンプルを変換していきます。

新しい Oracle9*i* サンプル・スキーマは、連結されたスキーマのセットです。このスキーマのセットは、複雑さに対して次のように階層化されたアプローチを提供することを目的としています。

- 基本的な項目を説明するための単純なスキーマ（Human Resources、`HR`）。このスキーマの拡張機能では、Oracle Internet Directory のデモをサポートしています。
- 中間レベルの複雑な状態を扱うための 2 つ目のスキーマ（Order Entry、`OE`）。このスキーマでは、数多くのデータ型を使用できます。

  Online Catalog（`OC`）サブスキーマは、`OE` スキーマ内部に作成されたオブジェクト・リレーショナル・データベースのオブジェクトの集合です。
- マルチメディア・データ型専用のスキーマ（Product Media、`PM`）。
- Oracle Advanced Queuing 機能のデモのために主スキーマ名 `QS`（Queued Shipping）の下に収集されたスキーマのセット。
- より大量のデータのデモを可能にするために設計されたスキーマ（Sales History、`SH`）。このスキーマの拡張機能では、拡張分析処理をサポートしています。

サンプル・スキーマにより提供される利点

- **コンテキストの一貫性**。同じ表セットがいたるところにあれば、ユーザー、研修受講者および開発者は、スキーマに時間をかけることなく、技術的な概念を理解することに、より多くの時間を費やすことができます。
- **有用性**。これらのシード・データベースのスキーマを使用して、Oracle マニュアルおよび研修資料に記載されたサンプルを実行できます。サンプルを直接使用することにより、概念の理解とアプリケーション開発が容易になります。
- **品質**。Oracle9*i* サンプル・スキーマを構築する作成スクリプトとスキーマに対して実行されるサンプルの両方を集中的に管理しテストすることによって、Oracle マニュアルと研修資料の品質が向上します。

Oracle9*i* サンプル・スキーマの設計方針

- **簡単な使用方法**。`HR` および `OE` スキーマは、機能の追加によって複雑になりすぎないようにする必要があります。データベースの簡単な使用方法から中級の使用方法まで、段階分けされた使用方法を提供する必要があります。
- **基本的であること**。基本的なスキーマと拡張機能では、通常使用する機能を最優先する必要があります。最も一般的に使用されるデータベース・オブジェクトのみがスキーマ内で自動的に作成され、スキーマ全体のセットでは、追加機能を示すために拡張可能な基盤が提供されています。
- **拡張性**。Oracle9*i* サンプル・スキーマでは、基本的な範囲を超えた機能のデモ用のオブジェクトを追加するための論理的および物理的な基盤が提供されています。
- **関連性**。Oracle9*i* サンプル・スキーマは、E-Business やその他の重要な業界動向（XML など）に適用できるように設計されています。これによって使用方法が複雑になる場合は、スキーマ拡張機能を使用して、業界動向に焦点を合せることができます。

このマニュアルの構成

このマニュアルは、次の章で構成されています。

第 1 章「インストール」

この章では、Oracle9*i* サンプル・スキーマをインストールする方法について説明します。

第 2 章「サンプル・スキーマの詳細」

この章では、サンプル・スキーマのベースとなる架空の会社について説明します。

第 3 章「サンプル・スキーマ図」

この章には、サンプル・スキーマの図が記載されています。

第 4 章「Oracle9i サンプル・スキーマのスクリプト」

この章では、サンプル・スキーマの生成に使用するスクリプトを記載しています。

関連文書

リリース・ノート、インストレーション・マニュアル、ホワイト・ペーパー、またはその他の関連文書は、OTN-J（Oracle Technology Network Japan）に接続すれば、無償でダウンロードできます。OTN-J を使用するには、オンラインでの登録が必要です。次の URL で登録できます。

`http://otn.oracle.co.jp/membership/`

OTN-J のユーザー名とパスワードを取得済みであれば、次の OTN-J Web サイトの文書セクションに直接接続できます。

`http://otn.oracle.co.jp/document/`

表記規則

このマニュアル・セットの本文とコード例に使用されている表記規則について説明します。

- 本文中の表記規則
- コード例の表記規則

本文中の表記規則

本文中には、特別な用語が一目でわかるように様々な表記規則が使用されています。次の表は、本文の表記規則と使用例を示しています。

表記規則	**意味**	**例**
太字	太字は、本文中に定義されている用語または用語集に含まれている用語、あるいはその両方を示します。	この句を指定する場合は、**索引構成表**を作成します。
固定幅フォントの大文字	固定幅フォントの大文字は、システムにより指定される要素を示します。この要素には、パラメータ、権限、データ型、Recovery Manager キーワード、SQL キーワード、SQL*Plus またはユーティリティ・コマンド、パッケージとメソッドの他、システム指定の列名、データベース・オブジェクトと構造体、ユーザー名、およびロールがあります。	この句は、`NUMBER` 列に対してのみ指定できます。 `BACKUP` コマンドを使用すると、データベースのバックアップを作成できます。 `USER_TABLES` データ・ディクショナリ・ビューの `TABLE_NAME` 列を問い合せます。 `DBMS_STATS.GENERATE_STATS` プロシージャを使用します。

表記規則	意味	例
固定幅フォントの小文字	固定幅フォントの小文字は、実行可能ファイル、ファイル名、ディレクトリ名およびサンプルのユーザー指定要素を示します。この要素には、コンピュータ名とデータベース名、ネット・サービス名、接続識別子の他、ユーザー指定のデータベース・オブジェクトと構造体、列名、パッケージとクラス、ユーザー名とロール、プログラム・ユニット、およびパラメータ値があります。 **注意 :** 一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は、記載されているとおりに入力してください。	`sqlplus` と入力して SQL*Plus をオープンします。 パスワードは `orapwd` ファイルに指定されています。 データ・ファイルと制御ファイルのバックアップを `/disk1/oracle/dbs` ディレクトリに作成します。 `department_id`、`department_name` および `location_id` の各列は、`hr.departments` 表にあります。 初期化パラメータ `QUERY_REWRITE_ENABLED` を `true` に設定します。 `oe` ユーザーで接続します。 これらのメソッドは `JRepUtil` クラスに実装されます。
固定幅フォントの小文字のイタリック	固定幅フォントの小文字のイタリックは、プレースホルダまたは変数を示します。	*`parallel_clause`* を指定できます。 `U`*`old_release`*`.SQL` を実行します。*`old_release`* は、アップグレード前にインストールしたリリースです。

コード例の表記規則

コード例は、SQL、PL/SQL、SQL*Plus またはその他のコマンドラインを示します。次のように、固定幅フォントで、通常の本文とは区別して記載されています。

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

次の表は、コード例の記載上の表記規則と使用例を示しています。

表記規則	**意味**	**例**
[ ]	大カッコで囲まれている項目は、1つ以上のオプション項目を示します。大カッコ自体は入力しないでください。	DECIMAL (*digits* [, *precision*])
{ }	中カッコで囲まれている項目は、そのうちの1つのみが必要であることを示します。中カッコ自体は入力しないでください。	{ENABLE | DISABLE}
|	縦線は、大カッコまたは中カッコ内の複数の選択肢を区切るために使用します。オプションのうち1つを入力します。縦線自体は入力しないでください。	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	水平の省略記号は、次のどちらかを示します。	
	■ 例に直接関係のないコード部分が省略されていること。	CREATE TABLE ...AS *subquery*;
	■ コードの一部が繰返し可能であること。	SELECT *col1*, *col2*, ..., *coln* FROM employees;
. . .	垂直の省略記号は、例に直接関係のない数行のコードが省略されていることを示します。	SQL> SELECT NAME FROM V$DATAFILE; NAME ------------------------------------ /fsl/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.

その他の表記	大カッコ、中カッコ、縦線および省略記号以外の記号は、示されているとおりに入力してください。	acctbal NUMBER(11,2);
		acct CONSTANT NUMBER(4) := 3;

表記規則	意味	例

イタリック	イタリックの文字は、特定の値を指定する必要のあるプレースホルダまたは変数を示します。	`CONNECT SYSTEM/`*`system_password`*
		`DB_NAME = `*`database_name`*

大文字	大文字は、システムにより指定される要素を示します。これらの用語は、ユーザー定義用語と区別するために大文字で記載されています。大カッコで囲まれている場合を除き、記載されているとおりの順序とスペルで入力してください。ただし、この種の用語は大 / 小文字区別がないため、小文字でも入力できます。	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

小文字	小文字は、ユーザー指定のプログラム要素を示します。たとえば、表名、列名またはファイル名を示します。
	注意 : 一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は、記載されているとおりに入力してください。
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

1

インストール

Oracle9*i* の完全インストールを実行すると、自動的にサンプル・スキーマがシード・データベースにインストールされます。なんらかの理由でシード・データベースをシステムから削除した場合は、サンプル・スキーマを再インストールしてから、Oracle マニュアルおよび研修資料内のサンプルをコピーする必要があります。

この章では、Oracle9*i* サンプル・スキーマをインストールする方法について説明します。この章は次の項で構成されています。

- Database Configuration Assistant の使用
- Oracle9i サンプル・スキーマの手動インストール
- サンプル・スキーマのリセット

注意： Oracle9*i* サンプル・スキーマをインストールすると、次のユーザー名を使用するインストール済みのスキーマが破棄されます。

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

この項で説明するインストール・スクリプトを実行すると、前述のスキーマのいずれかに含まれたデータが失われます。個人用や仕事用のデータやアプリケーションには、Oracle9*i* サンプル・スキーマを使用しないでください。サンプル・スキーマは、デモや学習の目的でのみ使用します。

Database Configuration Assistant の使用

DBCA の使用は、サンプル・スキーマをインストールする最も直観的で簡単な方法です。データベース作成処理の手順 4 を実行すると、データベースに使用するサンプル・スキーマを構成できます。Database Configuration Assistant の要件は次のとおりです。

- サンプル・スキーマを作成するには、チェックボックス「Example Schemas」をオンにする必要があります。
- Order Entry スキーマを作成するには、「Oracle Spatial」を選択する必要があります。
- Product Media スキーマを作成するには、「Oracle interMedia」を選択する必要があります。このオプションは、「データベース内のデータベース・オプションの構成」ボタンをクリックすると選択できます。
- マテリアライズド・ビューとディメンションを評価するには、「Oracle JVM」を選択する必要があります。これらの機能を使用する場合は、「データベース内のデータベース・オプションの構成」ボタンをクリックして、このオプションを選択します。

- Order Entry スキーマ・オプションでは、「Human Resources」オプションが選択されている必要があります。
- Product Media スキーマ・オプションでは、「Order Entry」オプションが選択されている必要があります。
- Shipping スキーマ・オプションでは、「Order Entry」オプションが選択されている必要があります。
- 「Sales History」オプションを選択して、「Oracle OLAP Services」を選択すると、OLAP サーバー・メタデータが Sales History スキーマに追加されます。

Database Configuration Assistant に添付されている事前定義されたデータベースの 3 つのテンプレートのうち、次の 2 つにはサンプル・スキーマが含まれています。

- OLTP データベース
- DSS データベース

Oracle9*i* サンプル・スキーマの手動インストール

前提条件

使用可能なサンプル・スキーマは、インストールする Oracle の Edition とその構成によって異なります。次の表を参照して、インストールできるスキーマを確認してください。

スキーマ	**Oracle9*i* Personal Edition**	**Oracle9*i* Standard Edition**	**Oracle9*i* Enterprise Edition**
HR	使用可能	使用可能	使用可能
OE	使用可能	使用可能	使用可能
PM	使用可能	使用可能	使用可能
QS	使用可能	使用可能	使用可能
SH	使用不能	使用不能	Partitioning Option のインストールが必要

スキーマ依存性

スキーマ間には、様々な依存性が確立しています。そのため、`HR`、`OE`、`PM`、`QS`、`SH` の順にスキーマを作成する必要があります。

注意： Oracle9*i* サンプル・スキーマの複雑さと依存性の順序は、アルファベット順です。

次の順序でスキーマを作成します。

1. `HR` スキーマを作成します。
2. `OE` スキーマを作成します。`HR` オブジェクト権限を `OE` に付与するには、`HR` スキーマがすでに存在している必要があり、`HR` スキーマ用のパスワードを知っている必要があります。`HR` 表の中には、プライベート・シノニムを使用することで `OE` ユーザーが参照できるものもあります。また、`OE` 表の中には、`HR` 表に対する外部キー関係を持つものもあります。

注意： `OE` スキーマでは、データベースが空間データ用に使用できることが必要です。これは、インストール中または Database Configuration Assistant の使用後に可能になります。

---

3. `PM` スキーマを作成します。外部キー関係では、`PM` スキーマの作成時に `OE` スキーマがすでに存在していることが必要です。これらの外部キーを設定して使用する権限を `PM` に付与するには、`OE` 用のパスワードを知っている必要があります。

注意： `PM` スキーマでは、データベースが Java Virtual Machine（Java VM）および *inter*Media で使用できることが必要です。これは、インストール中または Database Configuration Assistant の使用後に可能になります。

---

4. `QS` スキーマを作成します。出荷スキーマ `QS` は、`OE` 内の受注データに基づいています。また、外部キー関係では、`QS` スキーマの作成時に `OE` スキーマがすでに存在していることが必要です。これらの外部キーを設定して使用する権限を `QS` に付与するには、`OE` 用のパスワードを知っている必要があります。
5. `SH` スキーマを作成します。他の 4 つのスキーマがない場合でも、このスキーマを単独で作成できますが、`SH` スキーマは論理的に `OE` スキーマに依存しています。

Human Resources（HR）スキーマのインストール

このスキーマの作成に必要なスクリプトは、すべて `$ORACLE_HOME/demo/schema/human_resources` にあります。

すべてのオブジェクトの作成およびデータのロードに必要なスクリプトは、`hr_main.sql` のみです。`hr_main.sql` を実行すると、次の作業が実行されます。

1. パスワード、およびスクリプト内で使用する表領域名の入力を求めるプロンプトが表示されます。
2. インストール済みの `HR` スキーマがすべて消去されます。
3. ユーザー `HR` を作成して、必要な権限を付与します。
4. `HR` として接続します。
5. 次のスクリプトを実行します。
   - データ・オブジェクト作成用の `hr_cre.sql`
   - データ・オブジェクト移入用の `hr_popul.sql`
   - データ・オブジェクトの索引作成用の `hr_idx.sql`
   - プロシージャ・オブジェクト作成用の `hr_code.sql`
   - 表および列コメント作成用の `hr_comnt.sql`
   - スキーマ統計収集用の `hr_analz.sql`
6. [オプション] スキーマ拡張機能として、`sh_dn_c.sql` および `sh_dn_d.sql` の 2 つのスクリプトが提供されています。Human Resources スキーマで Oracle Internet Directory のディレクトリ機能を使用できるようにするには、`sh_dn_c.sql` 作成スクリプトを実行します。`HR` スキーマの初期設定に戻る場合は、スクリプト `sh_dn_d.sql` を使用して `sh_dn_c.sql` の結果を削除し、この拡張機能によって追加された列を削除します。

`HR` スキーマの削除に使用するファイルは `hr_drop.sql` です。

Order Entry（OE）スキーマとその Online Catalog（OC）サブスキーマのインストール

このスキーマの作成に必要なスクリプトは、すべて `$ORACLE_HOME/demo/schema/order_entry` にあります。

すべてのオブジェクトの作成およびデータのロードに必要なスクリプトは、`oe_main.sql` のみです。`oe_main.sql` を実行すると、次の作業が実行されます。

1. パスワード、およびスクリプト内で使用する表領域名の入力を求めるプロンプトが表示されます。
2. インストール済みの `OE` スキーマがすべて消去されます。
3. ユーザー `OE` を作成して、必要な権限を付与します。
4. `OE` として接続します。
5. 次のスクリプトを実行します。
   - データ・オブジェクト、プロシージャ・オブジェクトおよびユーザー定義オブジェクト作成用の `oe_cre.sql`
   - `PRODUCT_INFORMATION` 表の移入用の `oe_oe_p_pi.sql`
   - `WAREHOUSES` 表の移入用の `oe_p_whs.sql`
   - `CUSTOMERS` 表の移入用の `oe_p_cus.sql`
   - `ORDERS` 表の移入用の `oe_p_ord.sql`
   - `ORDER_ITEMS` 表の移入用の `oe_p_itm.sql`
   - `INVENTORIES` 表の移入用の `oe_p_inv.sql`
   - 表ビュー作成用の `oe_views.sql`
   - データ・オブジェクトの索引作成用の `oe_idx.sql`
   - 表および列コメント作成用の `oe_comnt.sql`
   - `OE` 内に `OC`（オンライン・カタログ）オブジェクト指向サブスキーマを作成する `oc_main.sql` スクリプトにより、次のスクリプトが実行されます。
   - 相関関係のある一連のユーザー定義オブジェクト、オブジェクト表およびビューを作成する `oc_cre.sql`
   - オブジェクト表の移入用の `oc_popul.sql`
   - 表および列コメント作成用の `oc_comnt.sql`
   - `PRODUCT_DESCRIPTIONS` 表の移入用の `oe_p_pd.sql`。製品名と説明のための言語固有の `INSERT` 文は、次のファイルに格納されています。
     - `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`
  - スキーマ統計収集用の `oe_analz.sql`

`OE` スキーマおよび `OC` サブスキーマの削除に使用するファイルは、次のとおりです。

- `oe_drop.sql`
- `oc_drop.sql`

Product Media（PM）スキーマのインストール

このスキーマの作成に必要なスクリプトは、すべて `$ORACLE_HOME/demo/schema/product_media` にあります。

すべてのオブジェクトの作成およびデータのロードに必要なスクリプトは、`pm_main.sql` のみです。`pm_main.sql` を実行すると、次の作業が実行されます。

1. パスワード、およびスクリプト内で使用する表領域名の入力を求めるプロンプトが表示されます。
2. インストール済みの `PM` スキーマがすべて消去されます。
3. ユーザー `PM` を作成して、必要な権限を付与します。
4. `PM` として接続します。
5. 次のスクリプトを実行します。
   - `pm_cre.sql`

   `PM` スキーマの移入に使用するファイルは、次のとおりです。

   - `pm_p_lob.sql`
   - `pm_p_lob.ctl`
   - `pm_p_lob.dat`

   注意： SQL*Loader のデータ・ファイル `pm_p_lob.dat` では、インストール中に設定された絶対パス名がハードコードされています。データを異なる環境にロードする場合は、最初にこのファイル内のパス名を編集する必要があります。

   - `pm_p_ord.sql`

`PM` スキーマの削除に使用するファイルは、`pm_drop.sql` です。

Queued Shipping（QS）スキーマのインストール

このスキーマの作成に必要なスクリプトは、すべて `$ORACLE_HOME/demo/schema/shipping` にあります。

すべてのオブジェクトの作成およびデータのロードに必要なスクリプトは、`qs_main.sql` のみです。`qs_main.sql` を実行すると、次の作業が実行されます。

1. パスワード、およびスクリプト内で使用する表領域名の入力を求めるプロンプトが表示されます。
2. インストール済みの `QS` スキーマがすべて消去されます。
3. ユーザー `QS` を作成して、必要な権限を付与します。
4. `QS` として接続します。
5. 次のスクリプトを実行します。
   - Administrator スキーマ作成用の `qs_adm.sql`
   - Customer Billing Administration スキーマ作成用の `qs_cbadm.sql`
   - Queued Shipping スキーマ用のキューおよびキュー表を作成する `qs_cre.sql`
   - Customer Service スキーマ作成用の `qs_cs.sql`
   - Eastern Shipping スキーマ作成用の `qs_es.sql`
   - Overseas Shipping スキーマ作成用の `qs_os.sql`
   - Western Shipping スキーマ作成用の `qs_ws.sql`
   - デモ・アプリケーション用のプロシージャおよびオブジェクトを作成する `qs_run.sql`

すべてのキューを正しい順序で削除するために使用するファイルは、`qs_drop.sql` です。

Sales History（SH）スキーマのインストール

このスキーマの作成に必要なスクリプトは、すべて `$ORACLE_HOME/demo/schema/sales_history` にあります。

すべてのオブジェクトの作成およびデータのロードに必要なスクリプトは、`sh_main.sql` のみです。`sh_main.sql` を実行すると、次の作業が実行されます。

1. パスワード、およびスクリプト内で使用する表領域名の入力を求めるプロンプトが表示されます。
2. インストール済みの `SH` スキーマがすべて消去されます。
3. ユーザー `SH` を作成して、必要な権限を付与します。
4. `SH` として接続します。

5. 次のスクリプトを実行します。
   - 表作成用の `sh_cre.sql`
   - ディメンション表 `COUNTRIES` および `CHANNELS` の移入用の `sh_pop1.sql`
   - ディメンション表 `TIMES` の移入用の `sh_pop2.sql`
   - 残りの表を移入する `sh_pop3.sql`。ディメンション表 `PROMOTIONS`、`CUSTOMERS`、`PRODUCTS` とファクト表 `SALES` は、SQL*Loader によってロードされます。それから、ファイルをロードしログを格納する位置を示すために、データベース内に 2 つのディレクトリ・パスが作成されます。この結果、ファイル `sh_sales.dat` を外部表として定義することによって、表 `COSTS` のロードが可能になります。
   - 表の索引作成用の `sh_idx.sql`
   - 表に制約を追加する `sh_cons.sql`
   - ディメンションおよび階層を作成する `sh_hiera.sql`
   - マテリアライズド・ビュー作成用の `sh_cremv.sql`
   - 列および表コメント追加用の `sh_comnt.sql`
   - 統計収集用の `sh_analz.sql`
6. [オプション] スキーマ拡張機能として、`sh_olp_c.sql` および `sh_olp_d.sql` の 2 つのスクリプトが提供されています。Sales History スキーマで OLAP サービスの拡張分析機能を使用できるようにするには、`sh_olp_c.sql` 作成スクリプトを実行します。`SH` スキーマの初期設定に戻る場合は、スクリプト `sh_olp_d.sql` を使用して `sh_olp_c.sql` の結果を削除し、ディメンションを元の状態に戻します。

`SH` スキーマの削除に使用するファイルは、`sh_drop.sql` です。

サンプル・スキーマのリセット

サンプル・スキーマを初期状態にリセットするには、SQL*Plus のコマンドライン・インタフェースから次の構文を使用します。

```
@?/demo/schema/mksample systempwd syspwd hrpwd oepwd pmpwd qspwd shpwd
```

パラメータ *systempwd*、*syspwd*、*hrpwd*、*oepwd*、*pmpwd*、*qspwd* および *shpwd* には、`SYSTEM` と `SYS` および `HR`、`OE`、`PM`、`QS` の各スキーマのパスワードを指定します。

`mksample` スクリプトによって、ディレクトリ `$ORACLE_HOME/demo/schema/log/` に、次のような複数のログ・ファイルが生成されます。

- `mkverify.log` ―サンプル・スキーマ作成ログ・ファイル
- `hr_main.log` ― `HR` スキーマ作成ログ・ファイル
- `oe_oc_main.log` ― `OE` スキーマ作成ログ・ファイル
- `pm_main.log` ― `PM` スキーマ作成ログ・ファイル
- `pm_p_lob.log` ― `PM.PRINT_MEDIA` のロードによって生成される SQL*Loader ログ・ファイル
- `qs_main.log` ― `QS` スキーマ作成ログ・ファイル
- `sh_main.log` ― `SH` スキーマ作成ログ・ファイル
- `sh_cust.log` ― `SH.CUSTOMERS` のロードによって生成される SQL*Loader ログ・ファイル
- `sh_prod.log` ― `SH.PRODUCTS` のロードによって生成される SQL*Loader ログ・ファイル
- `sh_promo.log` ― `SH.PROMOTIONS` のロードによって生成される SQL*Loader ログ・ファイル
- `sh_sales.log` ― `SH.SALES` のロードによって生成される SQL*Loader ログ・ファイル
- `sh_sales_ext.log` ― `SH.COSTS` のロードによって生成される外部表ログ・ファイル

> **関連項目：** `mksample` スクリプトのコピーは、第 4 章「Oracle9i サンプル・スキーマのスクリプト」を参照してください。

多くの場合、あるサンプル・スキーマを初めてインストールすることと、インストール済みのバージョンを上書きして再インストールすることは、同じ作業です。`*_main.sql` スクリプトによって、スキーマ・ユーザーとそのすべてのオブジェクトが削除されます。

OE または QS スキーマ内のオブジェクト間の関係が複雑であるために、DROP USER ... CASCADE 操作が正常に終了しない場合があります。これはまれなケースですが、このような場合、次のいずれかの手順を実行します。

OE スキーマの OC カタログ・サブスキーマの場合の手順は、次のとおりです。

1. OE として接続します。
2. スクリプト `oc_drop.sql` を実行します。
3. SYSTEM として接続します。
4. OE として接続しているユーザーがいないことを確認します。

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

5. OE ユーザーを削除します。

   ```
   DROP USER oe CASCADE;
   ```

QS スキーマの場合の手順は、次のとおりです。

1. SYSTEM として接続します。
2. QS として接続しているユーザーがいないことを確認します。

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

3. スクリプト `qs_drop.sql` を実行して、スキーマを削除します。個々のユーザーに対するパスワードの入力を求めるプロンプトが表示されます。

2

サンプル・スキーマの詳細

Oracle9*i* サンプル・スキーマは、様々なチャネルを介して商品を販売する架空の会社をベースにしています。この章では、この架空の会社について説明します。この章は次の項で構成されています。

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

概要

Oracle9*i* サンプル・スキーマで仮設定している会社は、全世界的に事業を展開しており、いくつかの異なる製品を受注しています。この会社には、次の部門があります。

- 人事（Human Resources）部門では、会社の従業員と施設に関する情報を管理しています。
- 受注（Order Entry）部門では、様々な販売チャネルにわたる、会社の製品の在庫と販売を管理しています。
- 製品メディア（Product Media）部門では、会社が販売する各製品に関する説明書と詳細情報を管理しています。
- 出荷（Shipping）部門では、顧客への製品の出荷を管理しています。
- 販売（Sales）部門では、ビジネス上の判断に役立つ事業統計を管理しています。

これらの部門は、それぞれスキーマで表現されます。

Human Resources （HR）

会社の人事レコードには、従業員ごとに一意の識別番号、電子メール・アドレス、職種識別番号、給料および管理者が含まれています。給料に加えて歩合給を受け取る従業員については、そのことも記録されています。

また、会社は組織内での職種についての情報も記録しています。各職種には、識別番号、役職、その職種の給料の上限と下限があります。長期間勤務している従業員の中には、複数の職種に属している人もいます。従業員が職種を変更する場合、会社は前の職種の開始日と終了日、職種識別番号および部署を記録します。

サンプルになっている会社は様々な地域に分かれているため、倉庫の所在地のみでなく部署の所在地も記録しています。会社の従業員は、各部署に配属されています。各部署は、一意の部署コードや短縮名で識別されます。各部署は1つの所在地に関連付けられています。それぞれの所在地には、番地、郵便番号、都市、州または県、国コードを含む完全な住所があります。

施設の所在地に対しては、会社は国名、通貨記号、通貨名および地理的に位置する地域を記録します。

Order Entry（OE）

この会社は、コンピュータのハードウェアとソフトウェア、音楽、衣料および工具などいくつかの分野に分類される製品を販売しています。会社は、製品識別番号、その製品の分類、重量によるグループ（出荷のため）、ある場合は保証期間、サプライヤ、製品のステータス、表示価格、最低販売価格、製造会社の URL アドレスを含む製品情報を管理しています。また、在庫がある倉庫、在庫数などすべての製品に対する在庫情報も記録されます。世界中で製品が販売されているため、この会社はいくつかの異なる言語で製品名と製品についての説明書を管理しています。

会社は、顧客の注文に対応するために、いくつかの場所に倉庫を設置しています。各倉庫には、倉庫識別番号、倉庫名および地域識別番号があります。

顧客情報は詳細に記録されています。顧客ごとに識別番号が割り当てられています。顧客レコードには、名前、番地、都市または県、国、電話番号（顧客ごとに最大 5 つまで）および郵便番号があります。インターネット経由で注文する顧客もいるため、電子メール・アドレスも記録されています。顧客は様々な言語を使用しているため、会社は顧客ごとに使用するネイティブ言語と地域を記録します。

また、顧客には一度に購入できる金額を制限する与信限度額が設定されています。顧客には、会社が監視するために顧客管理者が任命されている場合もあります。顧客の電話番号も記録されています。その時点で、1 人の顧客が持っている電話番号の数は不明ですが、すべての番号を記録するようにしています。顧客の使用する様々な言語に対応するため、顧客ごとに言語と地域を記録します。

顧客が発注すると、会社は受注日、受注モード、ステータス、出荷モード、受注数量および営業担当者を記録します。営業担当者は、顧客の顧客管理者と同一人物とは限りません。また、インターネット経由で受注した場合、営業担当者は記録されません。受注情報の他にも、受注した品目の数量、単価および製品を記録します。

事業を展開している国ごとに、国名、通貨記号、通貨名および地理的に位置する地域を記録します。顧客は世界中の様々な地域に居住しているため、このデータは便利です。

Online Catalog（OC）の説明

`OE` スキーマの `OC` サブスキーマは、オンライン・カタログの販売シナリオに沿って対処します。`OE` スキーマの場合と同じ顧客と製品が適切に使用されますが、`OC` サブスキーマでは、`OE` 製品が属するカテゴリを、親カテゴリとサブカテゴリの階層に編成します。この階層は E-Commerce のポータル・サイトでの配置に対応しており、ユーザーは、より特化した製品カテゴリにドリルダウンすることで特定の製品へ到達できます。

Product Media（PM）

この会社には、データベース内の製品に関するマルチメディア情報および印刷された情報が保管されています。それらの情報の例は、次のとおりです。

- 販売促進用ビデオ
- Web 公開用の製品画像とサムネイル
- PR 用の文書
- 印刷媒体の広告
- 他の販売促進用の文書と翻訳

Queued Shipping（QS）

サンプルの会社では、メッセージ機能を使用して、計画されている B2B アプリケーションの使用をテストすることに決定しました。この計画には、ユーザーがファイアウォールの外部から発注してそのステータスを追跡できる小規模のテストが必要です。注文は主システムに登録される必要があります。その後、顧客の住所に応じて、出荷に最も近い地域に注文が、転送されます。

会社は、現行の社内分散システムを、最終的に他の企業が出荷できるシステムに展開することを考えています。そのため、企業間で送信されるメッセージは、HTTP を使用して伝送可能な自己完結型の形式である必要があります。XML はメッセージにとって完全な形式であり、Advanced Queueing Servlet と Oracle Internet Directory によって、キュー間のルーティングが提供されます。

受注した商品が出荷されるか、入荷待ちになった後で、メッセージを担当社員に送り返して受注のステータスを通知し、支払請求周期を開始する必要があります。メッセージが 1 回しか送信されないことと、メッセージを追跡して調査するシステムがあることは、受注情報に不一致が生じた場合に役立ちます。

このテスト用アプリケーションのために、会社はデータベース・サーバーとアプリケーション・サーバーを 1 台ずつ使用します。アプリケーションには、XML メッセージおよびキューを調査するメカニズムがあります。ファイアウォール外からの接続性をテストするために、新規注文の作成と顧客サービスの報告を、キューを使用して実行します。新規注文のアプリケーションではキューが直接エンキューされますが、顧客サービスの問合せでは、XML メッセージ機能によってキューがデキューされる必要があります。

このアプリケーションに関連するユーザーは、次のとおりです。

- `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）
- `QS_CBADM`（Customer Billing Administration）

Sales History（SH）

サンプルの会社では、大規模な事業を行っているため、意思決定支援のために事業統計レポートを作成します。これらのレポートの多くは、時間ベースで作成され蓄積されます。つまり、過去のデータ傾向を分析できます。データをデータ・ウェアハウスにロードして、これらのレポート用の統計を定期的に収集します。これらのレポートでは、年次、四半期、月次および週次の製品ごとの販売数や額などを表示します。

また、販売が行われる流通チャネルのレポートも出力します。製品に対して特別な販売促進を実施する場合は、販売促進の効果を分析できます。地域によって販売を分析することもできます。

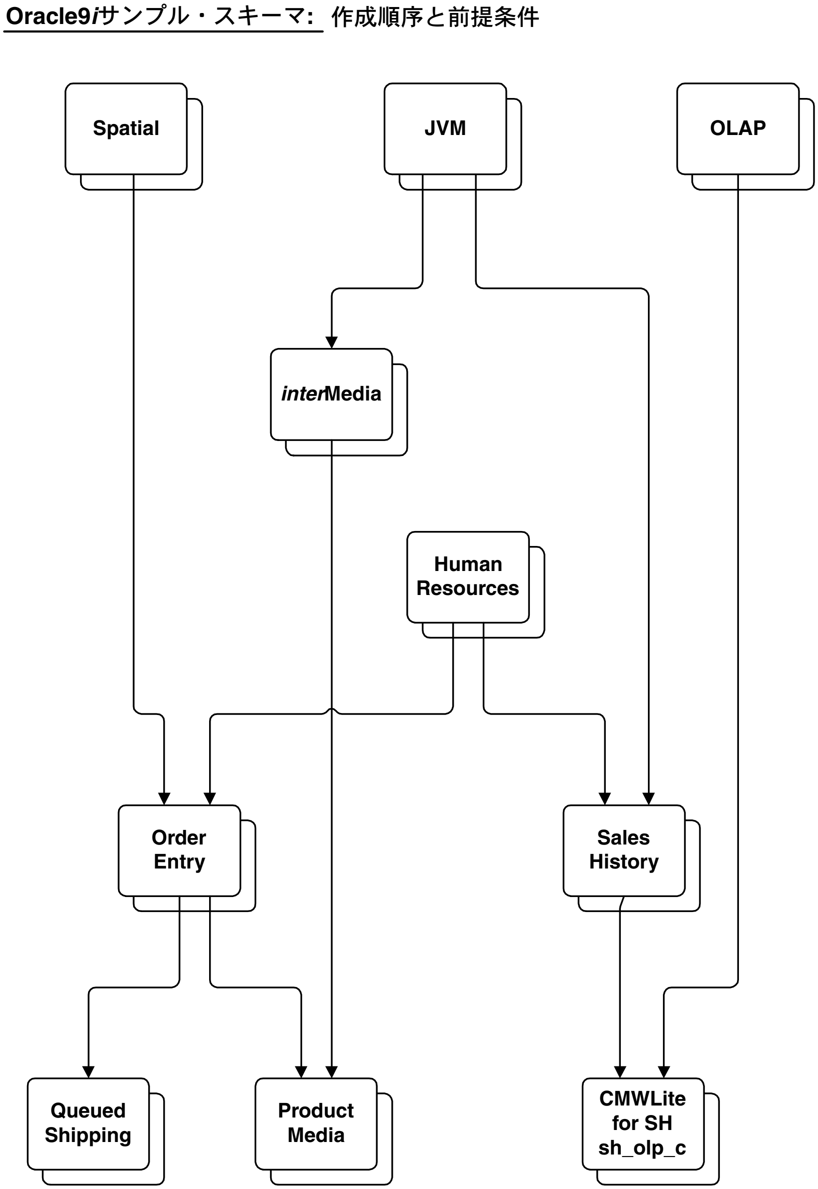
3

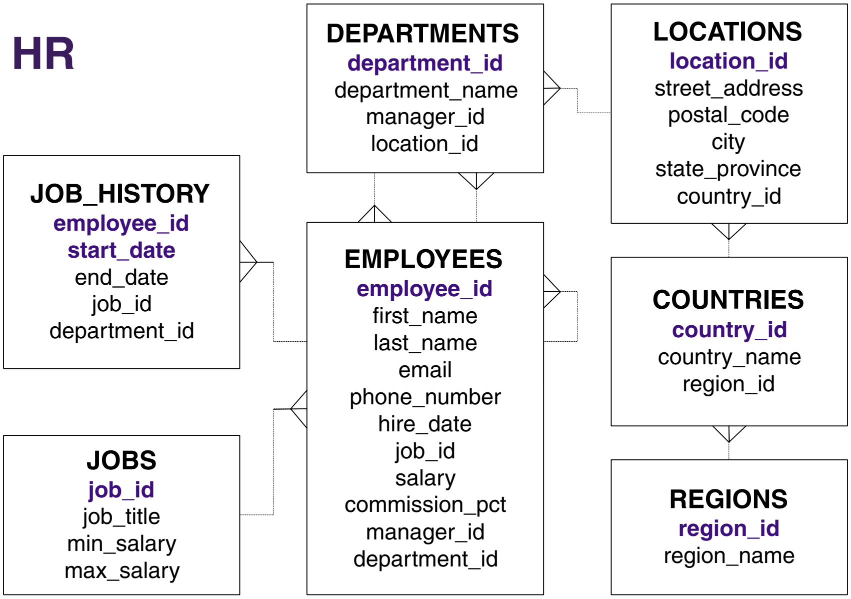
サンプル・スキーマ図

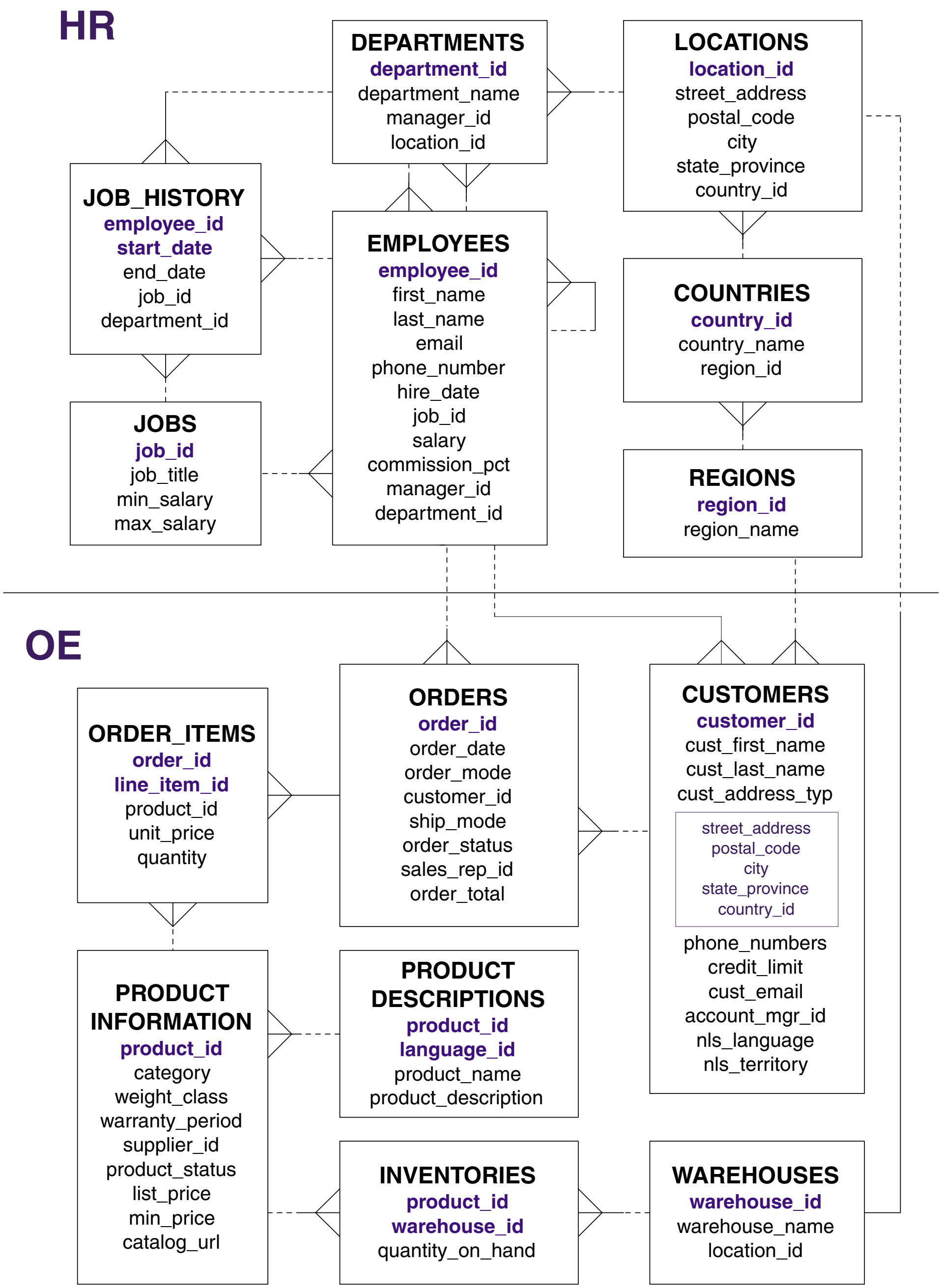
この章には、サンプル・スキーマの図が記載されています。最初の図は、サンプル・スキーマの作成順序と前提条件を示しています。残りの図では、各スキーマの様々なコンポーネントの構成を示します。

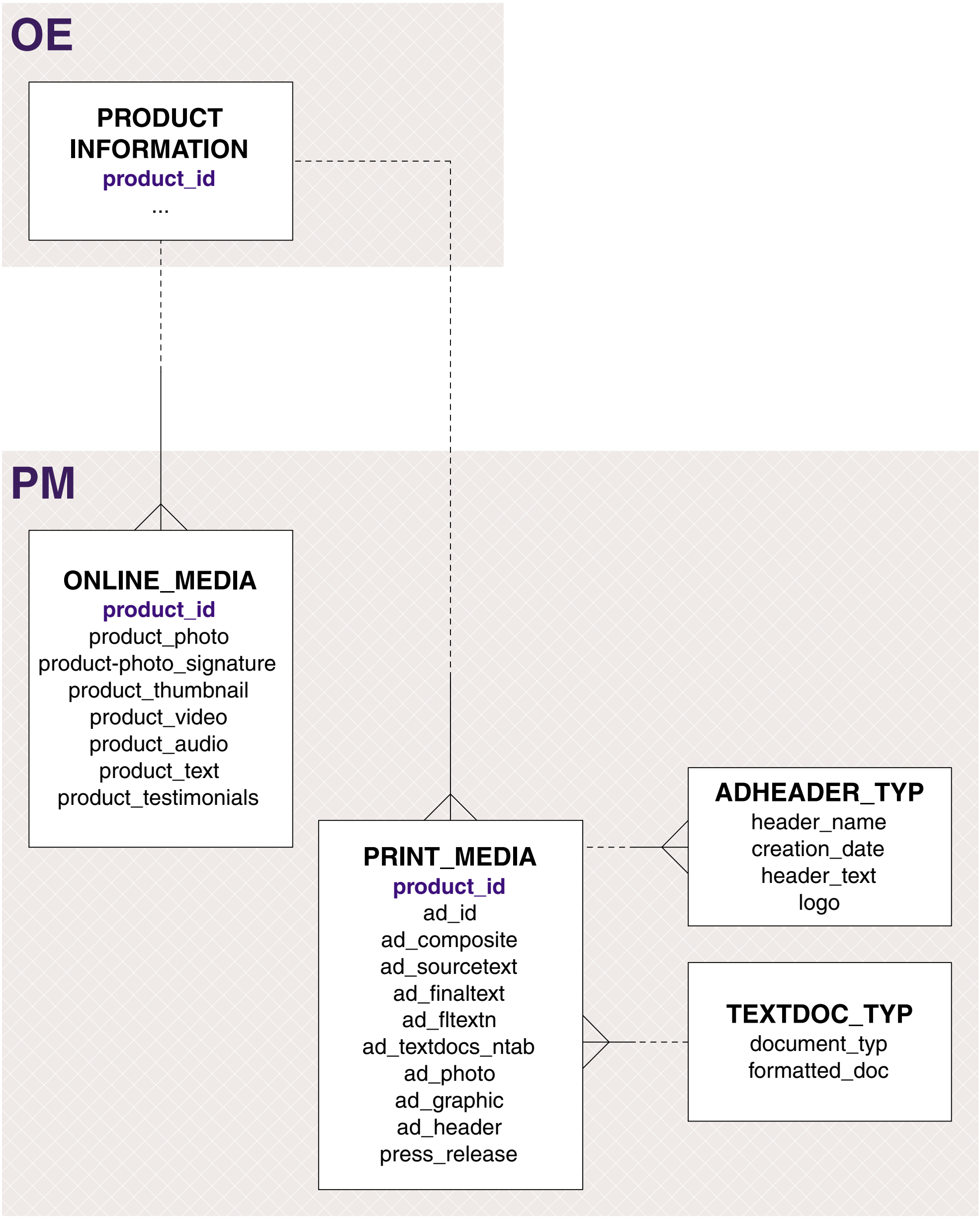
詳細情報と各スキーマの説明は、第4章「Oracle9i サンプル・スキーマのスクリプト」内の作成スクリプトを参照してください。

サンプル・スキーマ図

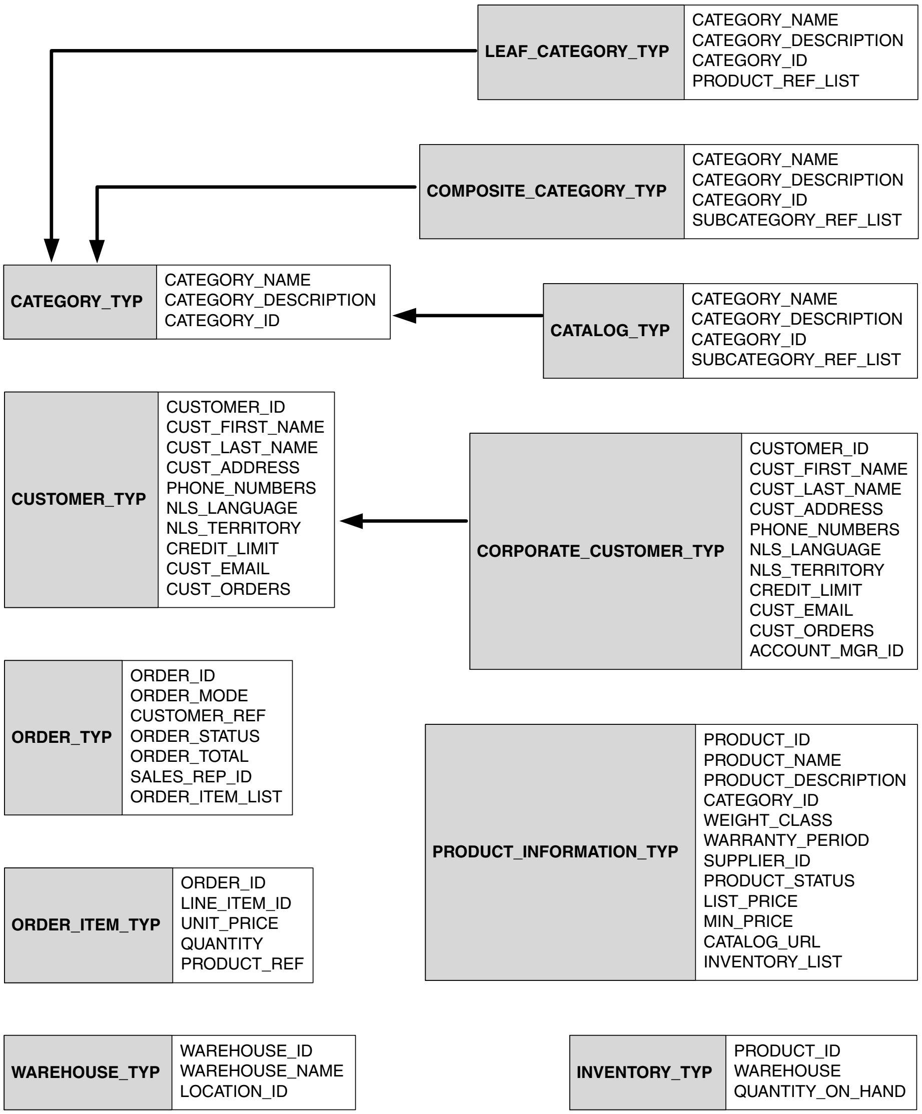

Oracle9*i*サンプル・スキーマ: 作成順序と前提条件
Spatial
JVM
OLAP
*inter*Media
Human Resources
Order Entry
Sales History
Queued Shipping
Product Media
CMWLite for SH sh_olp_c

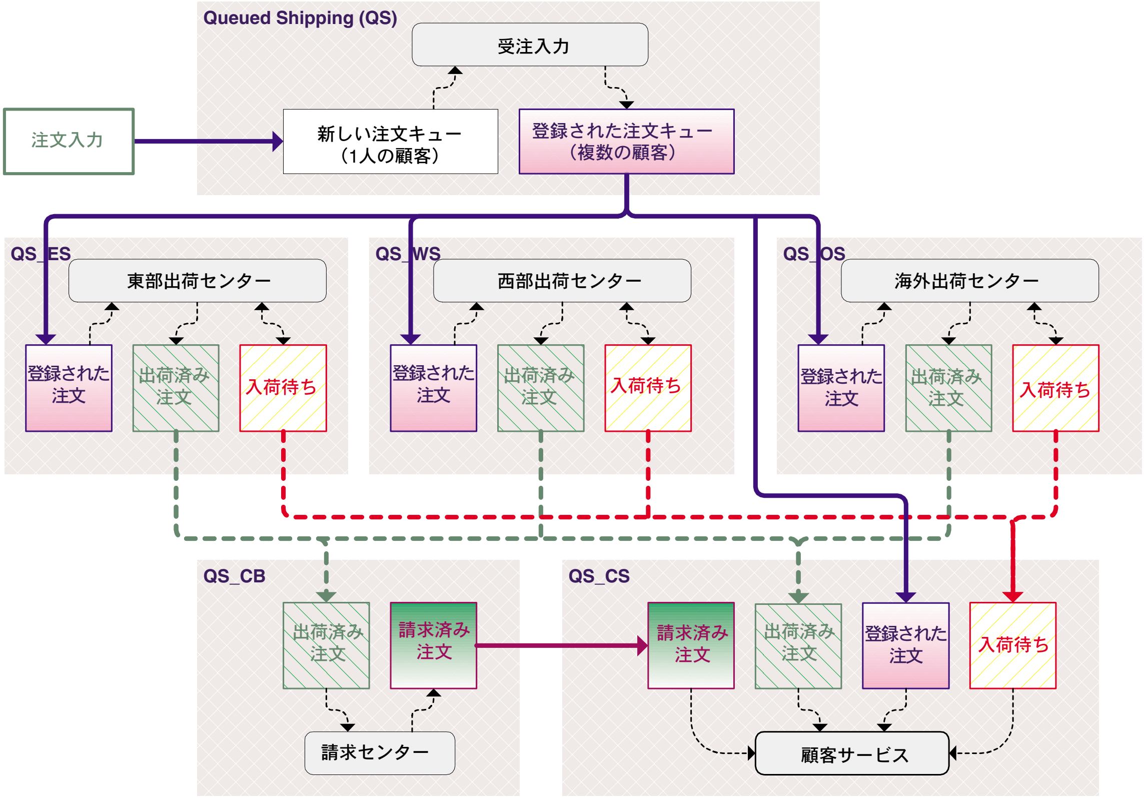

HR
DEPARTMENTS
department_id
department_name
manager_id
location_id
LOCATIONS
location_id
street_address
postal_code
city
state_province
country_id
JOB_HISTORY
employee_id
start_date
end_date
job_id
department_id
EMPLOYEES
employee_id
first_name
last_name
email
phone_number
hire_date
job_id
salary
commission_pct
manager_id
department_id
COUNTRIES
country_id
country_name
region_id
JOBS
job_id
job_title
min_salary
max_salary
REGIONS
region_id
region_name

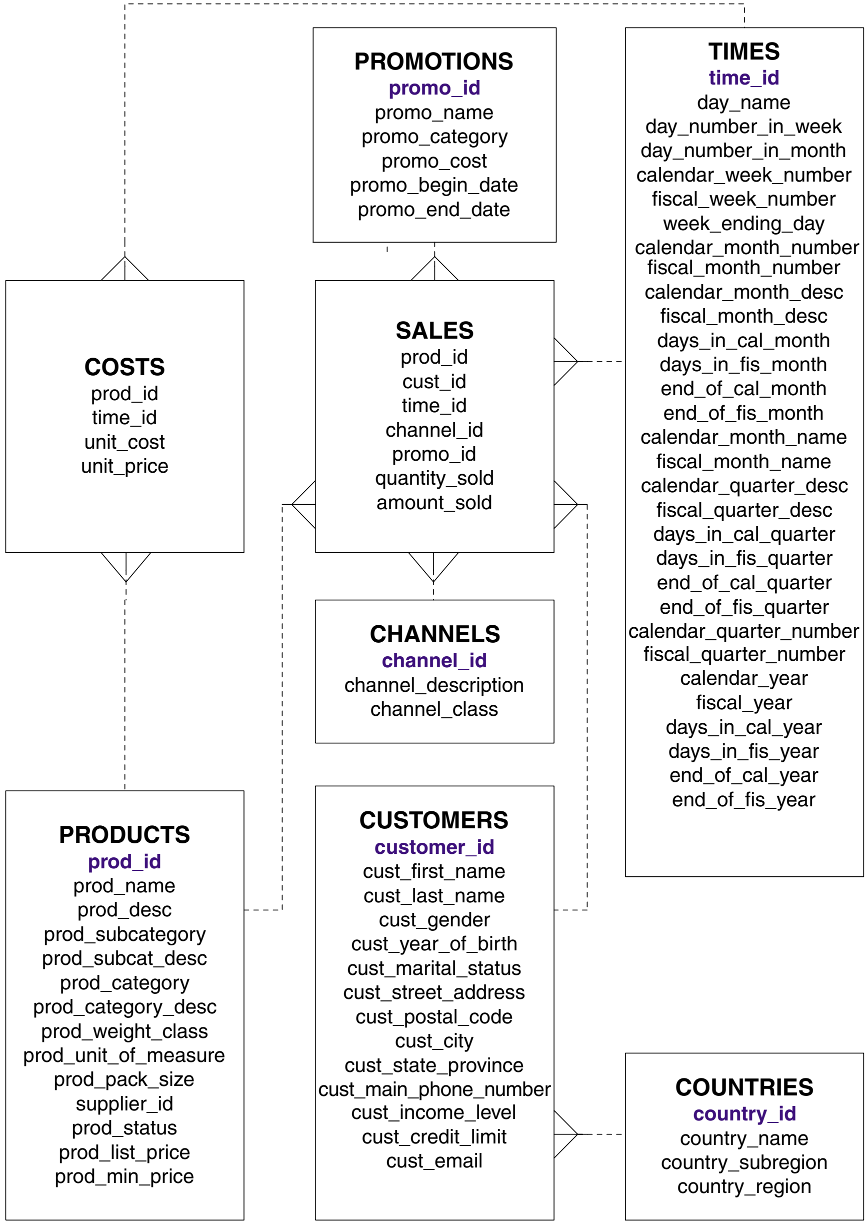

HR
DEPARTMENTS
department_id
department_name
manager_id
location_id
LOCATIONS
location_id
street_address
postal_code
city
state_province
country_id
JOB_HISTORY
employee_id
start_date
end_date
job_id
department_id
EMPLOYEES
employee_id
first_name
last_name
email
phone_number
hire_date
job_id
salary
commission_pct
manager_id
department_id
COUNTRIES
country_id
country_name
region_id
JOBS
job_id
job_title
min_salary
max_salary
REGIONS
region_id
region_name
OE
ORDER_ITEMS
order_id
line_item_id
product_id
unit_price
quantity
ORDERS
order_id
order_date
order_mode
customer_id
ship_mode
order_status
sales_rep_id
order_total
CUSTOMERS
customer_id
cust_first_name
cust_last_name
cust_address_typ
street_address
postal_code
city
state_province
country_id
phone_numbers
credit_limit
cust_email
account_mgr_id
nls_language
nls_territory
PRODUCT INFORMATION
product_id
category
weight_class
warranty_period
supplier_id
product_status
list_price
min_price
catalog_url
PRODUCT DESCRIPTIONS
product_id
language_id
product_name
product_description
INVENTORIES
product_id
warehouse_id
quantity_on_hand
WAREHOUSES
warehouse_id
warehouse_name
location_id


OE
PRODUCT INFORMATION
product_id
...
PM
ONLINE_MEDIA
product_id
product_photo
product-photo_signature
product_thumbnail
product_video
product_audio
product_text
product_testimonials
PRINT_MEDIA
product_id
ad_id
ad_composite
ad_sourcetext
ad_finaltext
ad_fltextn
ad_textdocs_ntab
ad_photo
ad_graphic
ad_header
press_release
ADHEADER_TYP
header_name
creation_date
header_text
logo
TEXTDOC_TYP
document_typ
formatted_doc

Online Catalog（OC）サブスキーマ: オブジェクト型の図




Queued Shipping (QS)
受注入力
注文入力
新しい注文キュー
（1人の顧客）
登録された注文キュー
（複数の顧客）
QS_ES
東部出荷センター
登録された注文
出荷済み注文
入荷待ち
QS_WS
西部出荷センター
登録された注文
出荷済み注文
入荷待ち
QS_OS
海外出荷センター
登録された注文
出荷済み注文
入荷待ち
QS_CB
出荷済み注文
請求済み注文
請求センター
QS_CS
請求済み注文
出荷済み注文
登録された注文
入荷待ち
顧客サービス


PROMOTIONS
promo_id
promo_name
promo_category
promo_cost
promo_begin_date
promo_end_date
TIMES
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
COSTS
prod_id
time_id
unit_cost
unit_price
SALES
prod_id
cust_id
time_id
channel_id
promo_id
quantity_sold
amount_sold
CHANNELS
channel_id
channel_description
channel_class
PRODUCTS
prod_id
prod_name
prod_desc
prod_subcategory
prod_subcat_desc
prod_category
prod_category_desc
prod_weight_class
prod_unit_of_measure
prod_pack_size
supplier_id
prod_status
prod_list_price
prod_min_price
CUSTOMERS
customer_id
cust_first_name
cust_last_name
cust_gender
cust_year_of_birth
cust_marital_status
cust_street_address
cust_postal_code
cust_city
cust_state_province
cust_main_phone_number
cust_income_level
cust_credit_limit
cust_email
COUNTRIES
country_id
country_name
country_subregion
country_region

4

Oracle9*i* サンプル・スキーマのスクリプト

この章では、Oracle9*i* サンプル・スキーマの生成に使用するスクリプトを記載しています。各項は、個々のスキーマに対応しています。この章は次の項で構成されています。

- スクリプトについて
- マスター・スクリプト
- Human Resources（HR）スキーマのスクリプト
- Order Entry（OE）スキーマのスクリプト
- Product Media（PM）スキーマのスクリプト
- Queued Shipping（QS）スキーマのスクリプト
- Sales History（SH）スキーマのスクリプト

スクリプトについて

各スキーマには2つのスクリプトがあります。

- あるスキーマのすべてのオブジェクトとデータを再設定して作成する1つのスクリプト。このスクリプトの名前は、*xx*`_main.sql` です。*xx* は、スキーマの略称です。この主となるスクリプトは、他のすべてのスクリプトを実行して、スキーマを作成およびロードします。
- あるスキーマからすべてのオブジェクトを消去する *xx*`_drop.sql` と呼ばれるスクリプト。*xx* は、スキーマの略称です。

Oracle9*i* サンプル・スキーマ・スクリプトのディレクトリは、`$ORACLE_HOME/demo/schema` です。

注意： スキーマを移入するスクリプトは非常に長いため、この章には記載されていません。

マスター・スクリプト

マスター・スクリプトによって、サンプル・スキーマ環境が設定され、5つのスキーマがすべて作成されます。

注意： 後述のマスター・スクリプト（`mksample.sql`）では、`%s_pmPath%`、`%s_logPath%` および `%s_shPath%.` などの変数が使用されています。これらの変数は、インストール時に設定されます。

---

mksample.sql

```
Rem
Rem $Header: mksample.sql 05-dec-2001.16:41:15 ahunold Exp $
Rem
Rem mksample.sql
Rem
Rem Copyright (c) 2001, Oracle Corporation.  All rights reserved.
Rem
Rem    NAME
Rem      mksample.sql - creates all 5 Sample Schemas
Rem
Rem    DESCRIPTION
Rem      This script rees and creates all Schemas belonging
Rem      to the Oracle9i Sample Schemas.
Rem      If you are unsure about the prerequisites for the Sample Schemas,
Rem      please use the Database Configuration Assistant DBCA to
Rem      configure the Sample Schemas.
Rem
Rem    NOTES
Rem      - This script is edited during installation to match
Rem        the directory structur on you system
Rem      - CAUTION: This script will erase the following schemas:
Rem        - HR
Rem        - OE
Rem        - PM
Rem        - SH
Rem        - QS, QS_ADM, QS_CB, QS_CBADM, QS_CS, QS_ES, QS_OS, QS_WS
Rem      - CAUTION: Never use the above mentioned Sample Schemas for
Rem        anything other than demos and examples
Rem      - USAGE: To return the Sample Schemas to their initial
Rem        state, you can call this script and pass the passwords
Rem        for SYS, SYSTEM and the schemas as parameters.
Rem        Example: @?/demo/schema/mksample mgr secure h1 o2 p3 q4 s5
Rem        (please choose your own passwords for security purposes)
Rem      - LOG FILES: The SQL*Plus and SQL*Loader log files are written
Rem        to the equivalent of $ORACLE_HOME/demo/schema/log
Rem        If you edit the log file location further down in this
Rem        script, use absolute pathnames
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem      ahunold   12/05/01 - added parameters
Rem      ahunold   05/03/01 - dupl lines
Rem      ahunold   04/23/01 - Verification, parameters for pm_main.
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 999
SET ECHO OFF
SET CONCAT '.'

PROMPT
PROMPT specify password for SYSTEM as parameter 1:
DEFINE password_system     = &1
PROMPT
```

```
PROMPT specify password for SYS as parameter 2:
DEFINE password_sys        = &2
PROMPT
PROMPT specify password for HR as parameter 3:
DEFINE password_hr         = &3
PROMPT
PROMPT specify password for OE as parameter 4:
DEFINE password_oe         = &4
PROMPT
PROMPT specify password for PM as parameter 5:
DEFINE password_pm         = &5
PROMPT
PROMPT specify password for all QS schemas as parameter 6:
DEFINE password_qs         = &6
PROMPT
PROMPT specify password for  SH as parameter 7:
DEFINE password_sh         = &7
PROMPT
PROMPT Sample Schema creating will take about 40 minutes to complete...
PROMPT

CONNECT system/&&password_system

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

CONNECT system/&&password_system

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

CONNECT system/&&password_system

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

CONNECT system/&&password_system

@?/demo/schema/shipping/qs_main.sql &&password_qs example temp &&password_system
&&password_oe &&password_sys ?/demo/schema/log/

CONNECT system/&&password_system

@?/demo/schema/sales_history/sh_main &&password_sh example temp &&password_sys %s_
shPath% %s_logPath%

CONNECT system/&&password_system
```

```
SPOOL OFF

SPOOL ?/demo/schema/log/mkverify.log

SELECT owner, object_type, object_name, subobject_name, status
FROM dba_objects
WHERE ( owner in ('HR','OE','SH','PM') OR owner like 'QS%' )
AND object_name NOT LIKE 'SYS%'
ORDER BY  1,2,3,4;

SELECT  owner, object_type, status, count(*)
FROM dba_objects
WHERE ( owner in ('HR','OE','SH','PM') OR owner like 'QS%' )
AND object_name LIKE 'SYS%'
GROUP BY owner, object_type, status;

SELECT          owner, table_name, num_rows
 FROM           dba_tables
 WHERE  (       owner in ('HR','OE','SH','PM')
    OR          owner like 'QS%' )
 ORDER BY       1,2,3;

SPOOL OFF
```

Human Resources（HR）スキーマのスクリプト

この項では、HR スキーマのスクリプトをアルファベット順に記載しています。

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 11-may-2001.09:49:06 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     05/11/01 - disable
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;
/

ALTER TRIGGER secure_employees DISABLE;

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   08/28/01 - roles
rem   ahunold   07/13/01 - NLS Territory
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 OFF

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

DEFINE spool_file = &log_path.hr_main.log
SPOOL &spool_file

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 CONNECT TO hr;
GRANT RESOURCE 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
ALTER SESSION SET NLS_LANGUAGE=American;
ALTER SESSION SET NLS_TERRITORY=America;

--
-- create tables, sequences and constraint
--

@?/demo/schema/human_resources/hr_cre

--
-- populate tables
--

@?/demo/schema/human_resources/hr_popul

--
-- create indexes
--

@?/demo/schema/human_resources/hr_idx

--
-- create procedural objects
--
```

```
@?/demo/schema/human_resources/hr_code

--
-- add comments to tables and columns
--

@?/demo/schema/human_resources/hr_comnt

--
-- gather schema statistics
--

@?/demo/schema/human_resources/hr_analz

spool off
```

Order Entry（OE）スキーマのスクリプト

この項では、OE スキーマのスクリプトをアルファベット順に記載しています。

注意： oc で始まるスクリプトは、OE スキーマのオブジェクト・リレーショナル部分を扱っており、`oe_main.sql` スクリプト内部からコールされます。

---

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
rem   is present.
rem
rem NOTES
rem   The OIDs assigned for the object types are used to
rem   simplify the setup of Replication demos and are not needed
rem   in most unreplicated environments.
rem
rem MODIFIED   (MM/DD/YY)
rem   ahunold   04/25/01 - OID
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
 OID '82A4AF6A4CD3656DE034080020E0EE3D'
 AS OBJECT
    ( warehouse_id       NUMBER(3)
    , warehouse_name     VARCHAR2(35)
    , location_id        NUMBER(4)
    ) ;
/
CREATE TYPE inventory_typ
 OID '82A4AF6A4CD4656DE034080020E0EE3D'
 AS OBJECT
    ( product_id          NUMBER(6)
    , warehouse           warehouse_typ
    , quantity_on_hand    NUMBER(8)
    ) ;
/
CREATE TYPE inventory_list_typ
 OID '82A4AF6A4CD5656DE034080020E0EE3D'
 AS TABLE OF inventory_typ;
/
CREATE TYPE product_information_typ
 OID '82A4AF6A4CD6656DE034080020E0EE3D'
 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
 OID '82A4AF6A4CD7656DE034080020E0EE3D'
 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
 OID '82A4AF6A4CD8656DE034080020E0EE3D'
 AS TABLE OF order_item_typ;
/
CREATE TYPE customer_typ
 OID '82A4AF6A4CD9656DE034080020E0EE3D';
```

```
/
CREATE TYPE order_typ
 OID '82A4AF6A4CDA656DE034080020E0EE3D'
 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
 OID '82A4AF6A4CDB656DE034080020E0EE3D'
 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
 OID '82A4AF6A4CDC656DE034080020E0EE3D'
 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
 OID '82A4AF6A4CDD656DE034080020E0EE3D'
 AS TABLE OF REF category_typ;
/
```

```
CREATE TYPE product_ref_list_typ
 OID '82A4AF6A4CDE656DE034080020E0EE3D'
 AS TABLE OF number(6);
/
CREATE TYPE corporate_customer_typ
 OID '82A4AF6A4CDF656DE034080020E0EE3D'
 UNDER customer_typ
      ( account_mgr_id     NUMBER(6)
      );
/
CREATE TYPE leaf_category_typ
 OID '82A4AF6A4CE0656DE034080020E0EE3D'
 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
 OID '82A4AF6A4CE1656DE034080020E0EE3D'
 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
 OID '82A4AF6A4CE2656DE034080020E0EE3D'
 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 01-feb-2002.13:19:06 ahunold Exp $
rem
rem Copyright (c) 2001, 2002, 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   ahunold   02/01/02 - bug2205388
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 29-aug-2001.10:44:11 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

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
```

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 NOTES
rem   The OIDs assigned for the object types are used to
rem   simplify the setup of Replication demos and are not needed
rem   in most unreplicated environments.
rem
rem MODIFIED   (MM/DD/YY)
rem   ahunold   09/17/01 - FK in PRODUCT_DESCRIPTIONS
rem   ahunold   04/25/01 - OID
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
  OID '82A4AF6A4CD1656DE034080020E0EE3D'
  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
  OID '82A4AF6A4CD2656DE034080020E0EE3D'
  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)
    ) ;

ALTER TABLE product_descriptions
ADD ( CONSTRAINT pd_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, 2002, 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   02/01/02 - bug2205388
rem   ahunold   01/30/01 - OE script headers
rem   ahunold   01/09/01 - checkin ADE

rem
rem  First drop the Online Catalog (OC) subschema objects
rem

@?/demo/schema/order_entry/oc_drop.sql

DROP TABLE        customers             CASCADE CONSTRAINTS;
DROP TABLE        inventories           CASCADE CONSTRAINTS;
DROP TABLE        order_items           CASCADE CONSTRAINTS;
DROP TABLE        orders                CASCADE CONSTRAINTS;
DROP TABLE        product_descriptions  CASCADE CONSTRAINTS;
DROP TABLE        product_information   CASCADE CONSTRAINTS;
DROP TABLE        warehouses            CASCADE CONSTRAINTS;

DROP TYPE         cust_address_typ;
DROP TYPE         phone_list_typ;

DROP SEQUENCE     orders_seq;

DROP SYNONYM      countries;
DROP SYNONYM      departments;
DROP SYNONYM      employees;
DROP SYNONYM      job_history;
DROP SYNONYM      jobs;
DROP SYNONYM      locations;

DROP VIEW         bombay_inventory;
DROP VIEW         product_prices;
DROP VIEW         products;
DROP VIEW         sydney_inventory;
DROP VIEW         toronto_inventory;

COMMIT;
```

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   08/28/01 - roles
rem   ahunold   07/13/01 - NLS Territory.
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 OFF

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

DEFINE spool_file = &log_path.oe_oc_main.log
SPOOL &spool_file

-- 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 CONNECT TO oe;
GRANT RESOURCE TO oe;
GRANT CREATE MATERIALIZED VIEW  TO oe;
GRANT 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
ALTER SESSION SET NLS_LANGUAGE=American;
ALTER SESSION SET NLS_TERRITORY=America;
@?/demo/schema/order_entry/oe_cre
@?/demo/schema/order_entry/oe_p_pi
@?/demo/schema/order_entry/oe_p_pd
@?/demo/schema/order_entry/oe_p_whs
@?/demo/schema/order_entry/oe_p_cus
```

```
@?/demo/schema/order_entry/oe_p_ord
@?/demo/schema/order_entry/oe_p_itm
@?/demo/schema/order_entry/oe_p_inv
@?/demo/schema/order_entry/oe_views
@?/demo/schema/order_entry/oe_comnt
@?/demo/schema/order_entry/oe_idx
@?/demo/schema/order_entry/oe_analz

@?/demo/schema/order_entry/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）スキーマのスクリプト

この項では、PM スキーマのスクリプトをアルファベット順に記載しています。

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      The OIDs assigned for the object types are used to
Rem      simplify the setup of Replication demos and are not needed
Rem      in most unreplicated environments.
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem      ahunold   04/25/01 - OID
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
```

```
  OID '82A4AF6A4CCE656DE034080020E0EE3D'
  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
  OID '82A4AF6A4CCF656DE034080020E0EE3D'
  AS OBJECT
    ( document_typ      VARCHAR2(32)
    , formatted_doc     BLOB
    ) ;
/
CREATE TYPE textdoc_tab
  OID '82A4AF6A4CD0656DE034080020E0EE3D'
  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
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     ahunold    02/01/02 - bug 2205497
Rem     ahunold    09/14/00 - Created
Rem

REM drop all tables of schema

DROP TABLE online_media  CASCADE CONSTRAINTS;
DROP TABLE print_media   CASCADE CONSTRAINTS;

DROP TYPE textdoc_tab;

DROP TYPE adheader_typ;
DROP TYPE textdoc_typ;

COMMIT;
```

pm_main.sql

```
Rem
Rem $Header: pm_main.sql 29-aug-2001.09:13:23 ahunold Exp $
Rem
Rem pm_main.sql
Rem
Rem Copyright (c) 2001, Oracle Corporation.  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    08/28/01 - roles
Rem     ahunold    07/13/01 - NLS Territory
Rem     ahunold    04/23/01 - typo
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 OFF

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

DEFINE spool_file = &log_path.pm_main.log
SPOOL &spool_file

-- 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 CONNECT TO pm;
GRANT RESOURCE TO pm;
GRANT CREATE ANY DIRECTORY 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;
ALTER SESSION SET NLS_TERRITORY=America;

@&data_path.pm_cre.sql   -- create objects
@&data_path.pm_p_ord.sql -- 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）スキーマのスクリプト

この項では、QS スキーマのスクリプトをアルファベット順に記載しています。

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 29-aug-2001.10:44:11 ahunold Exp $
Rem
Rem qs_main.sql
Rem
Rem Copyright (c) 2001, Oracle Corporation.  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     08/28/01 - roles
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 OFF

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

DEFINE spool_file = &log_path.qs_main.log
SPOOL &spool_file

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 CONNECT, RESOURCE 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 CONNECT, RESOURCE 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 CONNECT, RESOURCE 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 CONNECT, RESOURCE 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 CONNECT, RESOURCE 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 CONNECT, RESOURCE 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 CONNECT, RESOURCE 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 CONNECT, RESOURCE 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
@?/demo/schema/shipping/qs_adm

PROMPT calling qs_cre.sql ...
CONNECT qs/&pass;
@?/demo/schema/shipping/qs_cre
```

```
PROMPT calling qs_es.sql ...
CONNECT qs_es/&pass
@?/demo/schema/shipping/qs_es

PROMPT calling qs_ws.sql ...
CONNECT qs_ws/&pass
@?/demo/schema/shipping/qs_ws

PROMPT calling qs_os.sql ...
CONNECT qs_os/&pass
@?/demo/schema/shipping/qs_os

PROMPT calling qs_cbadm.sql ...
CONNECT qs_cbadm/&pass
@?/demo/schema/shipping/qs_cbadm

PROMPT calling qs_cs.sql ...
CONNECT qs_cs/&pass
@?/demo/schema/shipping/qs_cs

PROMPT calling qs_run.sql ...
CONNECT qs_adm/&pass
@?/demo/schema/shipping/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）スキーマのスクリプト

この項では、SH スキーマのスクリプトをアルファベット順に記載しています。

sh_analz.sql

```
Rem
Rem $Header: sh_analz.sql 27-apr-2001.13:56:20 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      To avoid regression test differences, COMPUTE
Rem      statistics are gathered.
Rem
Rem      It is not recommended to use the estimate_percent
Rem      parameter for larger data volumes. For example:
Rem      EXECUTE dbms_stats.gather_schema_stats( -
Rem          'SH'                           ,       -
Rem          granularity => 'ALL'           ,       -
Rem          cascade => TRUE                ,       -
Rem          estimate_percent => 20         ,       -
Rem          block_sample => TRUE           );
Rem
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem      ahunold   04/27/01 - COMPUTE
Rem      hbaer     01/29/01 - Created
Rem

EXECUTE dbms_stats.gather_schema_stats(-
'SH',-
granularity => 'ALL',-
cascade => TRUE,-
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 04-sep-2001.09:40:37 ahunold Exp $
REM
REM sh_cre.sql
REM
REM Copyright (c) 2001, Oracle Corporation.  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      ahunold   09/04/01 - .
REM      ahunold   08/16/01 - added partitions
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_1995 VALUES LESS THAN
(TO_DATE('01-JAN-1996','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
 PARTITION SALES_1996 VALUES LESS THAN
(TO_DATE('01-JAN-1997','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
 PARTITION SALES_H1_1997 VALUES LESS THAN
(TO_DATE('01-JUL-1997','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
 PARTITION SALES_H2_1997 VALUES LESS THAN
(TO_DATE('01-JAN-1998','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
 PARTITION SALES_Q1_1998 VALUES LESS THAN
(TO_DATE('01-APR-1998','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
         PARTITION SALES_Q2_1998 VALUES LESS THAN
(TO_DATE('01-JUL-1998','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
         PARTITION SALES_Q3_1998 VALUES LESS THAN
(TO_DATE('01-OCT-1998','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
         PARTITION SALES_Q4_1998 VALUES LESS THAN
(TO_DATE('01-JAN-1999','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
         PARTITION SALES_Q1_1999 VALUES LESS THAN
(TO_DATE('01-APR-1999','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
         PARTITION SALES_Q2_1999 VALUES LESS THAN
(TO_DATE('01-JUL-1999','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
         PARTITION SALES_Q3_1999 VALUES LESS THAN
(TO_DATE('01-OCT-1999','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
         PARTITION SALES_Q4_1999 VALUES LESS THAN
(TO_DATE('01-JAN-2000','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
         PARTITION SALES_Q1_2000 VALUES LESS THAN
(TO_DATE('01-APR-2000','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
         PARTITION SALES_Q2_2000 VALUES LESS THAN
(TO_DATE('01-JUL-2000','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
         PARTITION SALES_Q3_2000 VALUES LESS THAN
(TO_DATE('01-OCT-2000','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
         PARTITION SALES_Q4_2000 VALUES LESS THAN
(TO_DATE('01-JAN-2001','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')))
;

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','NLS_DATE_LANGUAGE = American')),
 PARTITION COSTS_Q2_1998 VALUES LESS THAN
(TO_DATE('01-JUL-1998','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
 PARTITION COSTS_Q3_1998 VALUES LESS THAN
(TO_DATE('01-OCT-1998','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
 PARTITION COSTS_Q4_1998 VALUES LESS THAN
(TO_DATE('01-JAN-1999','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
 PARTITION COSTS_Q1_1999 VALUES LESS THAN
(TO_DATE('01-APR-1999','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
 PARTITION COSTS_Q2_1999 VALUES LESS THAN
(TO_DATE('01-JUL-1999','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
 PARTITION COSTS_Q3_1999 VALUES LESS THAN
(TO_DATE('01-OCT-1999','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
 PARTITION COSTS_Q4_1999 VALUES LESS THAN
(TO_DATE('01-JAN-2000','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
 PARTITION COSTS_Q1_2000 VALUES LESS THAN
(TO_DATE('01-APR-2000','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
 PARTITION COSTS_Q2_2000 VALUES LESS THAN
(TO_DATE('01-JUL-2000','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
 PARTITION COSTS_Q3_2000 VALUES LESS THAN
(TO_DATE('01-OCT-2000','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')),
 PARTITION COSTS_Q4_2000 VALUES LESS THAN
(TO_DATE('01-JAN-2001','DD-MON-YYYY','NLS_DATE_LANGUAGE = American')))
;

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-2002.12:36:00 ahunold Exp $
Rem
Rem sh_drop.sql
Rem
Rem Copyright (c) 2001, 2002, Oracle Corporation.  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     ahunold    02/01/02 - bug2206757
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 ;

DROP TABLE mv_capabilities_table  CASCADE CONSTRAINTS ;
DROP TABLE plan_table             CASCADE CONSTRAINTS ;
DROP TABLE rewrite_table          CASCADE CONSTRAINTS ;
DROP TABLE sales_transactions_ext 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 29-aug-2001.09:10:41 ahunold Exp $
Rem
Rem sh_main.sql
Rem
Rem Copyright (c) 2001, Oracle Corporation.  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    08/28/01 - roles
Rem     ahunold    07/13/01 - NLS Territory
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 OFF

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

DEFINE spool_file = &log_dir.sh_main.log
SPOOL &spool_file

-- 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;

CREATE ROLE sales_history_role;

GRANT CREATE ANY DIRECTORY     TO sales_history_role;
GRANT DROP ANY DIRECTORY       TO sales_history_role;
GRANT CREATE DIMENSION         TO sales_history_role;
GRANT QUERY REWRITE            TO sales_history_role;
GRANT CREATE MATERIALIZED VIEW TO sales_history_role;

GRANT CONNECT               TO sh;
GRANT RESOURCE              TO sh;
GRANT sales_history_role    TO sh;
GRANT select_catalog_role   TO sh;

ALTER USER sh DEFAULT ROLE ALL;

rem   ALTER USER sh GRANT CONNECT THROUGH olapsvr;

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

ALTER SESSION SET NLS_LANGUAGE=American;
ALTER SESSION SET NLS_TERRITORY=America;

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 17-sep-2001.15:57:34 ahunold Exp $
Rem
Rem sh_olp_c.sql
Rem
Rem Copyright (c) 2001, Oracle Corporation.  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      ahunold   09/17/01 - sh_analz.sql
rem      ahunold   05/10/01 - Time dimension attributes
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 of Days');
    dbms_output.put_line('- Period Number of Days dropped');
 exception
     when cwm_exceptions.attribute_not_found then
       null;
 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
       null;
 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
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 Date 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;


   dbms_output.put_line('- Time Span');
    begin
          SELECT descriptor_id INTO time_span_id
          FROM all_olap_descriptors
          WHERE descriptor_value = 'Time Span'
          AND descriptor_type = 'Time Dimension Attribute Type';
          begin
            begin
              cwm_classify.add_entity_descriptor_use(time_span_id,
   DIMENSION_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'Period Number of Days');
            exception
              when cwm_exceptions.element_already_exists
                 then null;
            end;
            begin
              cwm_classify.add_entity_descriptor_use(time_span_id,
   LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'MONTH', 'DAYS_IN_CAL_MONTH');
        dbms_output.put_line('- Days in calendar month');
            exception
              when cwm_exceptions.element_already_exists
                 then null;
            end;
            begin
              cwm_classify.add_entity_descriptor_use(time_span_id,
   LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'QUARTER', 'DAYS_IN_CAL_QUARTER');
      dbms_output.put_line('- Days in calendar quarter');
            exception
              when cwm_exceptions.element_already_exists
                 then null;
            end;
            begin
              cwm_classify.add_entity_descriptor_use(time_span_id,
   LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'YEAR', 'DAYS_IN_CAL_YEAR');
      dbms_output.put_line('- Days in calendar year');
            exception
              when cwm_exceptions.element_already_exists
                 then null;
            end;
    begin
      cwm_classify.add_entity_descriptor_use(time_span_id,
   LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'FIS_MONTH', 'DAYS_IN_FIS_MONTH');
      dbms_output.put_line('- Days in fiscal month');
    exception
      when cwm_exceptions.element_already_exists
         then null;
    end;
    begin
      cwm_classify.add_entity_descriptor_use(time_span_id,
   LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'FIS_QUARTER', 'DAYS_IN_FIS_QUARTER');
      dbms_output.put_line('- Days in fiscal quarter');
    exception
      when cwm_exceptions.element_already_exists
         then null;
    end;
    begin
      cwm_classify.add_entity_descriptor_use(time_span_id,
   LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'FIS_YEAR', 'DAYS_IN_FIS_YEAR');
      dbms_output.put_line('- Days in fiscal year');
    exception
      when cwm_exceptions.element_already_exists
         then null;
            end;
          end;
    end;


    dbms_output.put_line('- End Date');
     begin
           SELECT descriptor_id INTO end_date_id
           FROM all_olap_descriptors
           WHERE descriptor_value = 'End Date'
           AND descriptor_type = 'Time Dimension Attribute Type';
           begin
             begin
               cwm_classify.add_entity_descriptor_use(end_date_id,
    DIMENSION_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'Period End Date');
             exception
               when cwm_exceptions.element_already_exists
                  then null;
             end;
             begin
               cwm_classify.add_entity_descriptor_use(end_date_id,
    LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'MONTH', 'END_OF_CAL_MONTH');
         dbms_output.put_line('- End of calendar month');
             exception
               when cwm_exceptions.element_already_exists
                  then null;
             end;
             begin
               cwm_classify.add_entity_descriptor_use(end_date_id,
    LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'QUARTER', 'END_OF_CAL_QUARTER');
       dbms_output.put_line('- End of calendar quarter');
             exception
               when cwm_exceptions.element_already_exists
                  then null;
             end;
             begin
               cwm_classify.add_entity_descriptor_use(end_date_id,
    LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'YEAR', 'END_OF_CAL_YEAR');
       dbms_output.put_line('- End of calendar year');
             exception
               when cwm_exceptions.element_already_exists
                  then null;
             end;
     begin
       cwm_classify.add_entity_descriptor_use(end_date_id,
    LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'FIS_MONTH', 'END_OF_FIS_MONTH');
       dbms_output.put_line('- End of fiscal month');
     exception
       when cwm_exceptions.element_already_exists
          then null;
     end;
     begin
       cwm_classify.add_entity_descriptor_use(end_date_id,
    LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'FIS_QUARTER', 'END_OF_FIS_QUARTER');
       dbms_output.put_line('- End of fiscal quarter');
     exception
       when cwm_exceptions.element_already_exists
          then null;
     end;
     begin
       cwm_classify.add_entity_descriptor_use(end_date_id,
    LEVEL_ATTRIBUTE_TYPE, USER, 'TIMES_DIM', 'FIS_YEAR', 'END_OF_FIS_YEAR');
       dbms_output.put_line('- End of fiscal year');
     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;

-- ------------------- Statistics ---------------------------

@?/demo/schema/sales_history/sh_analz.sql
```

sh_olp_d.sql

```
Rem
Rem $Header: sh_olp_d.sql 17-sep-2001.15:57:34 ahunold Exp $
Rem
Rem sh_olp_d.sql
Rem
Rem Copyright (c) 2001, Oracle Corporation.  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    09/17/01 - sh_analz.sql
Rem    ahunold    04/23/01 - duplicate lines
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 channels_dim;

@@sh_hiera
@@sh_analz
```