PRONTUARIO DE ORACLE

Facultad de Informática
Universidad Complutense de Madrid

Contenido

1. Introducción

Oracle es un sistema gestor de bases de datos que implementa el modelo relacional y una versión del modelo relacional orientado a objetos. Es un sistema multiplataforma (PC, estaciones de trabajo, mainframes, ...) que admite diferentes modelos de ejecución:

- Cliente/servidor. El servidor realiza gran parte de las tareas de acceso a los datos en respuesta a la solicitud del cliente, que puede ser una plataforma diferente. En el laboratorio se utilizará este modelo (un servidor PC y clientes Windows) mediante comunicación soportada con TCP/IP.
- Centralizado o basada en mainframe. El servidor realiza todas las tareas de acceso a los datos e incluso devuelve los datos con formato para la presentación en informes o formularios. El cliente no realiza prácticamente ninguna función. Se usa en grandes ordenadores centrales (mainframes) con terminales soportados directamente por su propio sistema de comunicaciones (por ejemplo, el protocolo SNA y el método de acceso VTAM en el sistema operativo MVS de IBM).
- Procesamiento distribuido. La base de datos está repartida en diferentes servidores y los usuarios no son conscientes de la ubicación física de los datos. El servidor distribuido maneja los datos de manera transparente al usuario.
- Procesamiento paralelo. La base de datos está en una única plataforma con varios procesadores y las consultas se pueden ejecutar en paralelo.

Las limitaciones de Oracle están determinadas por la plataforma en la que se ejecute más que por el propio sistema gestor de bases de datos. Así, un servidor UNIX no podrá dar servicio con eficacia a más de dos o tres mil usuarios. Sin embargo, en grandes sistemas centralizados con capacidad de teleprocesamiento importante, este número no es una limitación.

Este SGBD está orientado a medianas y grandes demandas, por lo que dispone de todas las características que se requieren de un SGBD en estos entornos, entre otras:

- Mecanismos de seguridad. Dispone de un lenguaje de control de que permite definir derechos de consulta, modificación y creación de datos a los usuarios según el nombre con el que se inicie la sesión.
- Copias de seguridad y recuperación. Permite realizarlas con el servidor parado o funcionando (copias en caliente) por lo que es posible un servicio ininterrumpido.
- Conectividad abierta. Permite acceder a datos de otros SGBD.
- Herramientas de desarrollo. Generación de informes, formularios, ...

2. Consolas básicas de trabajo

Hay dos consolas básicas para el trabajo con el sistema gestor de bases de datos Oracle: SQL*Plus y SQL Plus Worksheet. Oracle SQL*Plus es una consola básica al estilo de las interfaces de comandos básicas de los sistemas operativos, mientras que SQL Plus Worksheet es una consola que aporta más facilidades para manejar el servidor de bases de datos. La primera consola tiene el inductor de comandos SQL>. La segunda está dividida en dos paneles: el superior permite la escritura de comandos y programas y la inferior muestra la salida del sistema. Se destacan en color las palabras clave predefinidas (aunque no todas, como **`outer`**).

En ambas consolas se pueden utilizar tres tipos de comandos:

- Instrucciones SQL. Estas instrucciones SQL incluyen las partes DML, ADL y DDL de SQL estándar además de otras que no forman parte de él.
- Bloques PL/SQL. Se usan para agrupar instrucciones SQL y de PL/SQL (incluyendo estructuras de control y condicionales). Por ejemplo: **`CREATE function`** o **`CREATE procedure`**.
- Comandos SQL*Plus. Se usan para opciones específicas de las dos consolas interactivas SQL*Plus y SQL Worksheet.

En la consola SQL*Plus las instrucciones y comandos pueden ocupar más de una línea que se van numerando. Las líneas terminan con el carácter ";" para indicar su término. Para que se ejecute es necesario finalizarla con el carácter "/" y pulsar Intro. Si sólo se presiona Intro, la instrucción no se ejecuta, se queda en el buffer y puede ser ejecutada más tarde.

En la consola SQL Worksheet las instrucciones y comandos se escriben terminadas en punto y coma y se ejecutan pulsando el símbolo Ejecutar  que aparece en la parte superior izquierda o con la tecla F5.

2.1.1. Comandos SQL*Plus más comunes

Permiten manipular comandos SQL, dar formato a los resultados y establecer opciones. Deben ocupar una única línea en SQL*Plus. Para comandos más largos se puede continuar en la línea siguiente usando el guión – antes del salto de línea. No es necesario terminar con ;. Los comandos SQL*Plus terminan con RETURN.

A continuación se describen algunos de los comandos más comunes.

- **`COLUMN`** da formato una columna en la consulta actual.

  `COLUMN precio FORMAT $99.99 HEADING “Precio de Venta”`

- **`DESCRIBE`** lista la definición de columnas para una tabla de la base de datos. Para cada columna se muestra el nombre (Name), si admite nulos (Null?) y el tipo de datos (Type).

  `DESCRIBE` Clientes;

Lista la estructura de la tabla clientes. El resultado será el siguiente:

Name	Null?	Type
NUMERO	NOT NULL	NUMBER(38)
FECHA	NOT NULL	DATE
NOMBRE	NOT NULL	CHAR(30)
TELEFONO	NOT NULL	CHAR(20)
DIRECCION		CHAR(100)
ANOTACION		LONG

- **`EXECUTE`**: ejecuta una instrucción PL/SQL

  `EXECUTE` *instrucción*

- **`HELP`**: muestra la ayuda ON-line para los comando SQL*Plus.

  `HELP COLUMN`

  Para obtener una lista con todos los comandos SQL*Plus se usa

  `HELP COMMANDS`

- **`HOST`**: ejecuta un comando del sistema operativo sin salir de SQL*Plus.

  `SQL>HOST LS *.SQL`

  También se puede usar el símbolo (**`!`**) en vez de **`HOST`**.

  Poniendo host sin ningún comando lanzamos una consola del sistema operativo.

- **`REMARK`**: se usa para comentar. Cualquier línea que empiece por remark, rem o dos guiones – es ignorada.

- **`RUN`**: ejecuta la instrucción SQL que esté en el buffer.

- **SET**: establece valores para las variables de sistema de SQL*Plus. Por ejemplo:
  SET AUTOCOMMIT ON

  Indica a Oracle que debe comprometer los cambios inmediatamente después de la instrucción SQL que ha provocado ese cambio.

  SET TIMING ON

  Hace que SQL*Plus muestre el tiempo transcurrido al ejecutar un comando.

  SET PAUSE ON

  Hace que SQL*Plus pare al principio de cada página. Sólo se pasa a la siguiente página cuando el usuario pulsa RETURN.

  SET ECHO ON

  SQL*Plus mostrará cada uno de los comandos de un fichero ejecutado con **START**.

  Se puede obtener ayuda del resto de las variables usando **HELP SET**.

- **SPOOL**: almacena los resultados de las consultas en un archivo del sistema operativo.

  SPOOL nombre_del_fichero (se añade automáticamente la extensión **LST**).

  Para dejar de enviar al fichero:

  SPOOL OFF

  Para mostrar el estado actual (abierto o cerrado) del fichero de salida se usa:

  SHOW SPOOL

  Para cerrar el fichero e imprimirlo:

  SPOOL OUT

  SQL*Plus Worksheet tiene una opción de menú en Archivo que permite almacenar en fichero tanto la salida como la entrada.

- **START**: ejecuta comandos almacenados en un archivo (también se puede usar **@**). Es un comando muy útil y es preferible escribir comandos en un fichero que directamente en el inductor de SQL.

  START nombre_de_archivo

  @ nombre_de_archivo

  No es necesario especificar la extensión .SQL

2.1.2. Formato de la salida

Se puede cambiar el tamaño de la línea para permitir la visualización de líneas largas. Para ello, el sistema trunca las columnas del resultado.

- **SET LINESIZE** tamaño establece la longitud de las líneas en caracteres
- **SHOW LINESIZE** muestra el tamaño actual de las líneas
- **SET PAGESIZE** tamaño establece el número de líneas por página
- **SHOW PAGESIZE** muestra el número de líneas por página
- **TTITLE [RIGHT] '**texto**[|**texto**]' [LEFT]** establece el título que aparece en la página
- **TTITLE OFF** elimina el título de la página
- **SHOW TTITLE** muestra el título de la página
- **BTITLE [RIGHT] '**texto**[|**texto**]' [LEFT]** establece el pie de página
- **BTITLE OFF** elimina el pie de página

- **SHOW BTITLE** muestra el pie de página

2.1.3. Formato del resultado de una consulta

El comando **COLUMN** se puede usar para cambiar el nombre de la columna en la salida (heading) y para dar un nuevo formato al contenido de la columna en el resultado de una consulta. Sintaxis:

COLUMN <nombre_de_columna**> HEADING <**nombre_de_cabecera**> FORMAT <**formato**>**

Ejemplos:

```
COLUMN sid HEADING "Identificador de estudiante" FORMAT 99999
COLUMN lname HEADING "Apellido" FORMAT A15
COLUMN Precio FORMAT $9,99.99
```

2.1.4. Cálculo de subtotales

Si en la consulta se ha usado **order by** para ordenar los resultados, se puede usar el comando **BREAK** para crear subconjuntos de filas en el resultado de la consulta. Cada subconjunto se corresponde con un valor de la columna por la que ordenamos. Para cada subconjunto se pueden calcular subtotales.

BREAK ON nombre_columna permite la ruptura de secuencias

BREAK ON [REPORT SKIP M**] ON** nombre_columna_ruptura **SKIP** N permite la ruptura de secuencias y salto

COMPUTE SUM OF nombre_columna **ON** nombre_columna_ruptura permite el cálculo de la suma de una columna en la ruptura de secuencia

COMPUTE SUM OF nombre_columna **ON REPORT** permite el cálculo de la suma de una columna en la ruptura de secuencia

La cláusula **SKIP** permite añadir espacios después de cada subconjunto.

2.1.5. Comandos para manipular el buffer de la consola SQL*Plus

Los comandos que se explican en este apartado se aplican sólo a la consola SQL*Plus, no a la consola SQL Worksheet. Estos comandos no tienen utilidad en esta última porque una instrucción SQL se puede escribir en varias líneas en el panel de entrada, estando a la vista todas ellas. Además, se tiene acceso directo a los últimos comandos introducidos mediante los botones  y  (ALTERnativamente con las teclas CTRL+T y CTRL+N).

En la consola SQL*Plus se almacena en un buffer el último comando introducido en el intérprete. Es posible acceder, cambiar, guardar y guardar el contenido del buffer utilizando los siguientes comandos. Todos los comandos (menos **list**) se refieren a la línea actual que puede cambiarse usando **list** número_de_línea.

- **APPEND** (Abreviado **A**) TEXT: añade el texto al final de una línea.
- **CHANGE /OLD/NEW** : cambia OLD por NEW en una línea.

  Abreviado **C /OLD/NEW**

  Si no se pone el texto NEW el texto OLD se borra de la línea.
- **CLEAR BUFFER** o **CL BUFFER** borra todas las líneas.
- **DEL** borra una línea.
- **GET** fichero, carga el contenido del fichero en el buffer.
- **INPUT** (abreviado **I**) añade una o más líneas.
- **INPUT TEXTO** añade una línea con el texto **TEXTO**.

- **LIST** (abreviado **L**) lista todas las líneas del buffer.
- **LIST** n, lista una línea y hace que sea la línea actual. También se puede usar **L** n o simplemente n, siendo n el número de línea.
- **LIST** *, lista la línea actual.
- **LIST LAST**, lista la última línea del buffer.
- **LIST M N**, lista desde la línea m a la n.
- **SAVE** fichero, guarda el contenido del buffer a un fichero llamado fichero.

También se puede editar el contenido del buffer usando el comando EDIT, que invoca al editor predeterminado del sistema operativo y carga el contenido del buffer en ese editor. Después de editarlo, se puede salvar el contenido del buffer.

Se puede cambiar el editor predeterminado usando:
```
SQL> DEFINE _EDITOR=vi
```

3. Diccionario de datos

Toda la información de las tablas está registrada en el diccionario del sistema (*Data Dictionary*), que está formado por tablas especiales que se crean en la instalación de Oracle (que son administradas por el sistema).

Las tablas que constituyen el diccionario de datos están accesibles a través de la vista (del sistema) *dictionary* o su sinónimo *dict*.

El diccionario de datos lo actualiza Oracle y puede ser consultado (total o parcialmente según los permisos) por los usuarios. Las consultas se hacen usando las vistas del diccionario de datos, que son de tres tipos:

- Las vistas cuyo nombre comienza por **ALL_** , pueden ser consultadas por todos los usuarios y ofrecen información sobre todos los objetos del sistema.

  Ejemplo: **ALL_TABLES**, **ALL_SEQUENCES**, **ALL_VIEWS**, ...

- Las vistas cuyo nombre comienza por **USER_** , ofrecen información para listar los objetos propios.

  Ejemplo: **USER_TABLES**, **USER_SEQUENCES**, ..

- Las vistas cuyo nombre comienza por **DBA_** , sólo son accesibles para tareas de administración.

Para consultar la lista de tablas que componen el diccionario se escribe:
```
HELP DATA DICT
```

Que muestra una lista con la siguiente información:

Nombre de la tabla	Descripción
ACCESSIBLE_COLUMNS	columns of all tables, views, AND clusters
ACCESSIBLE_TABLES	tables AND views accessible TO the USER
AUDIT_ACTIONS	maps action type numbers TO action type names
ALL_INDEXES	Descriptions of indexes ON accessible
ALL_SEQUENCES	Descriptions of the USER's own sequences
ALL_TABLES	description of tables accessible TO the USER
.....	
USER_TABLES	Descriptions of the USER's own tables
USER_TAB_COLUMNS	columns of the USER's tables, views, AND clusters

USER_TAB_GRANTS	grants ON objects WHERE the USER is the owner, grantor, OR grantee

También se puede ver la estructura de una tabla del diccionario como se muestra a continuación:

```
DESCRIBE ALL_TABLES;
DESCRIBE ALL_INDEXES;
DESCRIBE ALL_SEQUENCES;
```

Listar las tablas, índices y secuencias definidas por el usuario EIDOS.

Para las tablas:

```
SELECT TABLE_NAME "tabla"
      FROM ALL_TABLES
      WHERE OWNER='inf01';
```

Para los índices:

```
SELECT TABLE_NAME, INDEX_NAME
      FROM ALL_INDEXES
      WHERE OWNER='inf01';
```

Para las secuencias:

```
SELECT SEQUENCE_NAME
      FROM ALL_SEQUENCES
      WHERE SEQUENCE_OWNER='inf01';
```

3.1. Tabla USER_CATALOG

Contiene información sobre las tablas y vistas definidas por un usuario. El esquema de esta tabla es:

```
USER_CATALOG(TABLE_NAME, TABLE_TYPE);
```

También podemos referirnos a ella usando su sinónimo público **CAT**.

La siguiente instrucción SQL muestra todas las tablas y vistas definidas por el usuario actual:

```
SELECT * FROM CAT;
```

3.2. Tabla USER_OBJECTS

Contiene información sobre todos los objetos definidos por el usuario actual. Además de la información accesible a través de USER_CAT, usando USER_OBJECTS tendremos acceso a las vistas, funciones, procedimientos, índices, sinónimos, disparadores, etc.

El esquema de la tabla es el siguiente:

```
USER_OBJECTS (OBJECT_NAME, OBJECT_ID, OBJECT_TYPE, CREATED,
LAST_DDL_TIME, TIMESTAMP, STATUS)
```

Donde:

CREATED indica cuándo fue creado el objeto.

TIMESTAMP es lo mismo que **CREATED** pero en formato **STRING**.

LAST_DDL_TIME indica el último acceso por una instrucción DDL.

STATUS indica si el objeto es válido o no.

3.3. Tabla USER_TABLES

Si queremos obtener más información sobre las tablas que su nombre no basta con usar **USER_CAT**. Hay que usar **USER_TABLES** o su sinónimo **TABS**.

Podemos obtener su esquema utilizando **DESCRIBE USER_TABLES;**

3.4. Tabla USER_tab_columns

Almacena información detallada sobre las columnas de las tablas. Su sinónimo público es **COLS**.

3.5. Tabla USER_views

Almacena información sobre las vistas definidas por un usuario. El esquema de esta tabla es:

```
USER_VIEWS(VIEW_NAME, TEXT_LENGTH, TEXT)
```

Donde **TEXT** es el texto de la vista y **TEXT_LENGTH** la longitud del mismo.

3.6. Tabla USER_CONSTRAINTS

Almacena información sobre las restricciones definidas por el usuario sobre las tablas.

Ej:
```
SELECT TABLE_NAME, CONSTRAINT_TYPE, CONSTRAINT_NAME
      FROM USER_CONSTRAINTS;
```

Da como salida:
```
TABLE_NAME                     C CONSTRAINT_NAME
------------------------------ - ------------------------------
Códigos postales               C SYS_C003451
Códigos postales               C SYS_C003452
Códigos postales               P SYS_C003453
DOMICILIOS                     P SYS_C003454
DOMICILIOS                     R SYS_C003455
DOMICILIOS                     R SYS_C003456
EMPLEADOS                      C SYS_C003446
EMPLEADOS                      C SYS_C003447
EMPLEADOS                      P SYS_C003448
EMPLEADOS                      C SUELDO_ADMISIBLE
Empleados modificados          P SYS_C003499
TELÉFONOS                      P SYS_C003449
TELÉFONOS                      R SYS_C003450

13 filas seleccionadas.
```

Donde el tipo de restricción es:

- C para indicar NOT NULL
- P para indicar PRIMARY KEY
- R para indicar REFERENCES

Los nombres de las restricciones son los proporcionados por el usuario o por el sistema cuando aquél no dio ninguno.

3.7. Tabla ALL_CONS_COLUMNS

Almacena información sobre las columnas a las que se refiere una restricción de clave primaria o de integridad referencial.

Ej:
```
SELECT COLUMN_NAME
FROM ALL_CONS_COLUMNS
WHERE CONSTRAINT_NAME='SYS_C003454';
```

Da como salida las columnas que forman la clave primaria:
```
COLUMN_NAME
-----------
```

```
DNI
CALLE
Código postal

3 filas seleccionadas.
```

3.8. Tabla USER_TRIGGERS

Almacena información sobre los disparadores definidos sobre las tablas de usuario.

Ej:

```
SELECT TRIGGER_TYPE, TABLE_NAME, TRIGGERING_EVENT
FROM USER_TRIGGERS
WHERE TRIGGER_NAME='ActualizaEstadísticas';
```

Da como salida:

```
TRIGGER TYPE     TABLE_NAME   TRIGGERING_EVENT
---------------  -----------  -------------------------
AFTER STATEMENT  Estudiantes  INSERT OR UPDATE OR DELETE

1 filas seleccionadas.
```

4. Instrucciones DDL básicas de SQL

4.1. Eliminación de tablas

DROP TABLE nombre_de_la_tabla**;**

Elimina la tabla del esquema relacional. Se eliminan también todos los índices de la tabla. Si hay claves externas en otras tablas que se refieran a alguna clave (candidata o primaria) de la tabla que estamos borrando, se produce un error de consistencia y la tabla no será borrada.

Para forzar el borrado de la tabla aunque haya otras tablas que hacen referencia a sus claves, se usa la siguiente versión:

DROP TABLE nombre_de_la_tabla **CASCADE CONSTRAINTS;**

que borra las restricciones de integridad referencial que se refieran a las claves de la tabla borrada y borra la tabla de la base de datos.

4.2. Creación de tablas

CREATE TABLE nombre_de_la_tabla

(definición_de_columna**, ... ,** definición_de_columna**,**

restricción_de_tabla**, ... ,** restricción_de_tabla**);**

Nota: también hay otras propiedades, como el espacio de tablas (**tablespace**) en el que se va a colocar la tabla, el espacio que va a ocupar, ...

Como máximo pueden tener 254 columnas. El nombre de la tabla puede tener de 1 a 30 caracteres. Deben empezar con un carácter entre A y Z. Dependiendo del sistema, se hace distinción entre mayúsculas y minúsculas. También es posible, dependiendo del sistema, incluir caracteres del conjunto extendido ASCII (eñes, acentos) y espacios (para los espacios es preciso encerrar el nombre entre comillas dobles). No se pueden crear tablas con el mismo nombre dentro de la misma base de datos. La tabla tiene como propietario al usuario que las crea. Otro usuario que desee usar nuestras tablas debe tener autorización para ello (véase el apartado 5) y hacer referencia a la tabla como (*propietario.tabla*).

definición_de_columna: <nombre_de_columna> <tipo_datos> [(<tamaño>)] [**DEFAULT** <expr>] [<restricción_de_columna>]

Donde:

- <nombre_de_columna>: Nombre de la columna. Cada columna o campo puede tener un nombre de 1 a 30 caracteres. Se permite el mismo nombre de campo en tablas distintas de una misma base de datos, pero no en la misma tabla.
- <tipo_datos> y <tamaño>: Tipo de datos de la columna. Algunos tipos de datos habituales son:

  CHAR(N): cadena de longitud fija de N caracteres

  NUMBER(P[,D]): número decimal con P dígitos de los cuales D son los decimales. Es el único tipo de datos numérico de Oracle, pero también acepta los nombres de los tipos de datos de otros sistemas, como por ejemplo:

  INTEGER: número entero, que se representa en Oracle como number(38).
- **DEFAULT** Valor predeterminado: se usa para fijar un valor predeterminado cuando se inserta una fila sin especificar ningún valor en esta columna. Ejemplo: beneficios **INTEGER DEFAULT = 10000**
- <restricción_de_columna>: Restricción a aplicar sobre la columna. Sintaxis:

  [CONSTRAINT <nombre**>]**

  [NOT] NULL |

  CHECK (<condición**>) |**

  UNIQUE |

  PRIMARY KEY |

  REFERENCES <nombre_tabla**> [(<**nombre_columna**>)] [ON DELETE CASCADE]**

  Cada restricción puede tener un nombre opcional. La definición de restricciones en las columnas permite establecer reglas de validación de datos (restricciones de dominio), así como los controles necesarios para mantener la integridad referencial (mediante **REFERENCES**) entre tablas a través de las columnas claves. Los tipos de claves que se pueden especificar son:

  - *Clave primaria*: columna que identifica de forma única al registro, es un valor único y no nulo (**NOT NULL**). Por ejemplo: el código del cliente es una clave primaria que identifica de forma única e irrepetible a cada cliente.
  - *Clave candidata:* **UNIQUE** fuerza a que no pueda haber dos filas con el mismo valor en esta columna.
  - *Clave externa*: Columna de la tabla que hace referencia a un valor que tiene que estar registrado en otra tabla.

    La cláusula **ON DELETE CASCADE** es opcional. Si se especifica, Oracle borrará automáticamente todas las filas dependientes cuando se borra la fila referenciada (la que contiene la clave primaria o candidata).

4.2.1. Restricciones de tabla

Se definen después de describir todas las columnas de una tabla para describir las restricciones que afectan a toda la tabla, no a campos individuales. Por ejemplo, cuando una clave primaria está formada por más de una columna. Las más comunes son clave primaria, externa y candidata.

[CONSTRAINT <nombre**>]**

UNIQUE (<columna**< {, <**columna**>}) |**

PRIMARY KEY (<columna**< {, <**columna**>})|**

FOREIGN KEY (<columna**< {, <**columna**>})**

REFERENCES <nombre_tabla**> [(<**nombre_columna**>)] [ON DELETE CASCADE]**

4.3. Modificación de tablas

ALTER TABLE

Permite modificar la estructura de las tablas después de su creación. Permite:

- Añadir nuevas columnas. (add)
- Añadir restricciones a una columna.
- Modificar el ancho de la columna.
- Modificar el tipo de datos de la columna sólo si la columna no contiene datos o está vacía.

RENAME

Permite renombrar tablas: **RENAME** nombre_antiguo **TO** nombre_nuevo**;**

4.4. Índices

4.4.1. Creación de índices

CREATE INDEX nombre_del_índice

A partir de la versión 7 de Oracle no hay que crear índices explícitamente para cada clave; para cada clave se crea automáticamente un índice. Se pueden crear otros índices para optimizar ciertos accesos a la base de datos.

4.4.2. Eliminación de índices

Para eliminar un índice se usa **DROP INDEX** nombre_del_índice.

5. Instrucciones ADL básicas de SQL

El lenguaje de acceso a datos (ADL, Access Data Language) permite definir usuarios, permisos y roles sobre permisos que se pueden adjuntar a los usuarios. En realidad se trata de la definición de *seguridad* para la base de datos con los mecanismos internos que proporciona. En este caso se realiza mediante SQL.

5.1. Creación de usuarios

CREATE USER

5.2. Concesión de privilegios

Sintaxis general:

GRANT *Privilegio* **ON** *Recurso* **TO** *Usuarios* [**WITH GRANT OPTION**]

Concede el privilegio *Privilegio* sobre el recurso *Recurso* a los usuarios *Usuarios*, permitiendo a su vez que los usuarios *Usuarios* puedan propagarlo a otros.

En lugar de *Privilegio* se puede especificar **ALL PRIVILEGES**.

Ej:

GRANT SELECT ON Distribución **TO** usuario_invitado.

GRANT *Privilegio* **ON** *Recurso* **TO** *Usuarios* [**WITH GRANT OPTION**]

5.3. Revocación de privilegios

Sintaxis general:

REVOKE *Privilegio* **ON** *Recurso* **FROM** *Usuarios* [**RESTRICT|CASCADE**]

Privilegio puede ser cualquiera de los listados anteriormente más **GRANT OPTION**.

RESTRICT|CASCADE se refieren a la revocación de los permisos propagados a otros usuarios.

Algunos privilegios son:

CONNECT (inicio de sesión), **CREATE SYNONYM**, **CREATE PUBLIC SYNONYM**, **CREATE PROCEDURE**, **CREATE VIEW**, **CREATE TABLE**, **CREATE TRIGGER**, **INSERT**, **UPDATE**, **DELETE**, **SELECT**, **REFERENCES** (Ej: Integridad referencial), **USAGE** (Aplicado sobre los dominios, permiten usar dominios en la definición del esquema de una tabla).

5.4. Creación de roles (papeles):

Un papel es una colección de permisos. Los papeles se asignan a usuarios.

Sintaxis general:

CREATE ROLE NombreRol

Pasos en la creación de roles:

1. Crear el rol.
2. Conceder privilegios al rol con GRANT.
3. Conceder el rol creado a los usuarios.

Ejemplo:

CREATE ROLE nombre_papel

GRANT SELECT ON Distribución **TO** nombre_papel

GRANT nombre_papel **TO** usuario_invitado

6. Notas sobre el DML de SQL

6.1. Reuniones (joins)

Hasta la versión 8i de Oracle incluida no hay una sintaxis específica para las reuniones (tanto internas como externas) como la que se recoge en el estándar SQL:1999.

6.1.1. Reuniones internas

La reunión zeta interna a $\bowtie_\theta$ b se expresa con la instrucción

```
SELECT a.*,b.* FROM a,b WHERE θ
```

La sintaxis ANSI, que adopta la versión 9i de Oracle, permite:

```
SELECT a.*,b.* FROM a INNER JOIN b ON θ
```

Las equirreuniones se expresan como:

```
SELECT a.*,b.* FROM a INNER JOIN b using(columna_1,...,columna_n)
```

más compacta que usar

```
SELECT a.*,b.* FROM a INNER JOIN b ON a.columna_1=b.columna_1 AND ...
AND a.columna_n=b.columna_n
```

6.1.2. Reuniones externas

La expresión de las reuniones externas hasta la versión 8i de Oracle se realiza en la condición de la selección. Es necesario escribir un símbolo más encerrado entre paréntesis (+) a continuación de la

columna de la tabla en la que no se puede encontrar correspondencia en la reunión con la otra tabla. La versión 9 admite la sintaxis ANSI.

- Reunión externa por la izquierda.

  La reunión externa por la izquierda a $=\bowtie_\theta$ b, que incluye en el resultado todas las filas de a aunque no encuentren correspondencia con ninguna fila de b, se expresa:

  ```
  SELECT a.*,b.* FROM a,b WHERE a.columna_1 = a.columna_1 (+) AND ...
  AND a.columna_n = b.columna_n (+)
  ```

  Con la sintaxis ANSI disponible en Oracle 9i:

  ```
  SELECT a.*,b.* FROM a LEFT OUTER JOIN b ON a.columna_1 =
  b.columna_1 AND ... AND a.columna_n = b.columna_n
  ```

  que, si se trata de una equirreunión, se puede simplificar a:

  ```
  SELECT a.*,b.* FROM a LEFT OUTER JOIN b using (columna_1, ...,
  columna_n)
  ```

  Si se trata de una reunión natural, se puede simplificar a:

  ```
  SELECT a.*,b.* FROM a natural LEFT OUTER JOIN b
  ```

- Reunión externa por la derecha.

  La reunión externa por la derecha a $\bowtie=_\theta$ b, que incluye en el resultado todas las filas de b aunque no encuentren correspondencia con ninguna fila de a, se expresa:

  ```
  SELECT a.*,b.* FROM a,b WHERE a.columna_1 (+) = a.columna_1 AND ...
  AND a.columna_n (+) = b.columna_n
  ```

  Con la sintaxis ANSI disponible en Oracle 9i:

  ```
  SELECT a.*,b.* FROM a RIGHT OUTER JOIN b ON a.columna_1 =
  b.columna_1 AND ... AND a.columna_n = b.columna_n
  ```

  que, si se trata de una equirreunión, se puede simplificar a:

  ```
  SELECT a.*,b.* FROM a RIGHT OUTER JOIN b using (columna_1, ...,
  columna_n)
  ```

  Si se trata de una reunión natural, se puede simplificar a:

  ```
  SELECT a.*,b.* FROM a natural RIGHT OUTER JOIN b
  ```

- Reunión externa completa.

  La reunión externa completa a $=\bowtie=_\theta$ b, que incluye en el resultado todas las filas de a y b aunque no encuentren correspondencia con ninguna fila de b o de a, respectivamente, no se puede expresar con una única instrucción **SELECT**. Se puede expresar:

  ```
  SELECT a.*,b.* FROM a,b WHERE a.columna_1 (+) = a.columna_1 (+) AND
  ... AND a.columna_n (+) = b.columna_n (+)
  ```

  Con la sintaxis ANSI disponible en Oracle 9i:

  ```
  SELECT a.*,b.* FROM a full OUTER JOIN b ON a.columna_1 =
  b.columna_1 AND ... AND a.columna_n = b.columna_n
  ```

  que, si se trata de una equirreunión, se puede simplificar a:

  ```
  SELECT a.*,b.* FROM a RIGHT OUTER JOIN b using (columna_1, ...,
  columna_n)
  ```

  Si se trata de una reunión natural, se puede simplificar a:

  ```
  SELECT a.*,b.* FROM a natural full OUTER JOIN b
  ```

Restricciones sobre las reuniones externas según la sintaxis de Oracle 8.

- El operador (+) sólo puede aparecer en la cláusula WHERE, no en la lista de proyección, y sólo se puede aplicar a una columna de una tabla o vista.
- Si A y B se reúnen con varias condiciones de reunión, el operador (+) se debe usar en todas estas condiciones.
- El operador (+) se puede aplicar sólo a una columna, no a una expresión arbitraria. Sin embargo, una expresión arbitraria puede contener una columna marcada con el operador (+).
- Una condición que contenga el operador (+) no se puede combinar con otra condición usando el operador lógico OR.
- Una condición no puede usar el operador IN para comparar con otra expresión con una columna marcada con (+).
- Una condición no puede comparar con una subconsulta con cualquier columna marcada con (+).
- En una consulta de reunión sobre más de dos tablas no se puede expresar más de una tabla que aporte nulos que se reúna con otra dada.

6.2. Algunas funciones útiles

- SYSDATE. Obtiene la fecha y la hora del sistema como número decimal cuya parte entera representa la fecha y la parte fraccionaria la hora. Se puede aplicar la función TO_CHAR para extraer la información de la fecha y la hora como una cadena de caracteres con el formato que se desee. Por ejemplo:

```
SELECT TO_CHAR(sysdate,'hh:mi:ss') FROM DUAL;
TO_CHAR(
--------
10:35:11
1 fila seleccionada.
```

Además de este formato se pueden aplicar también **dd**, **mm** e **yyyy** para la fecha, entre otros.

6.3. La tabla DUAL

La tabla **dual** se usa para obtener datos como pseudocolumnas. La tabla en sí no existe, sólo se usa en instrucciones SQL que no toman datos de ninguna tabla.

Ejemplos:

```
SELECT POWER(4,3) FROM DUAL;
SELECT SYSDATE FROM DUAL;
SELECT USER, SYSDATE, nombre_secuencia.CURRVAL FROM DUAL;
```

Donde **USER** es el nombre del usuario, **SYSDATE** es la fecha del sistema y codigo_cliente.**CURRVAL** es el último valor asignado a la secuencia.

6.4. Secuencias

Las secuencias son series de números que el sistema genera automáticamente. Estos números se usan como valores para las claves primarias, de forma que cada fila de una tabla se pueda identificar unívocamente, incluso en un entorno multiusuario en el que varios usuarios puedan estar generando valores de una misma secuencia, como en el caso de la adición concurrente de filas a una misma tabla.

Su sintaxis es:

```
CREATE SEQUENCE nombre_secuencia
[INCREMENT BY número_entero]
[START WITH número_entero]
[MAXVALUE número_entero |NOMAXVALUE]
```

[MINVALUE número_entero **|NOMINVALUE]**

[CYCLE|NOCYCLE]

Para acceder a las secuencias se usan las funciones **NEXTVAL** y **CURRVAL**. La función **NEXTVAL** realiza una petición al sistema para que genere el siguiente valor de la secuencia. En el ejemplo, nombre_secuencia.**NEXTVAL**, nos dará el siguiente valor que le corresponde a la secuencia. Para conocer el valor actual de la secuencia, o sea, el último código asignado, se usa la columna nombre_secuencia.**CURRVAL**, desde la tabla **DUAL** del sistema. Antes de poder usar **CURRVAL** hay que usar **NEXTVAL** para iniciar la secuencia.

7. La consola de administración Enterprise Manager

7.1. Introducción a la arquitectura

El SGBD Oracle denomina servicios a las bases de datos. Un SGBD Oracle puede tener definidas múltiples bases de datos y, para cada una de ellas, todos los objetos necesarios, como:

- Usuarios. Usuarios del SGBD definidos en el propio SGBD que presentan una serie de privilegios concedidos por el usuario del sistema (**system**) o por otro usuario con suficientes privilegios.
- Espacios de tablas. Proporcionan una forma de organizar lógicamente las tablas. Hay que conceder permiso explícito para que los usuarios puedan acceder a ellas (con **CREATE USER** o **ALTER USER**, al indicar la cuota permitida sobre los espacios de tablas, bien sea el predeterminado (**DEFAULT**) u otro).

Los usuarios que crean objetos (tablas, vistas, índices, ...) son los propietarios de éstos y pueden conceder acceso a otros usuarios.

7.2. La consola de administración

Oracle Enterprise Manager es una consola para la administración del servidor. Mediante una interfaz gráfica se pueden administrar el esquema de las bases de datos, la seguridad, las sesiones, el almacenamiento y otros aspectos como los almacenes de datos y las bases de datos XML. Esta consola se ejecuta desde el acceso directo Enterprise Manager Console.

La consola se articula en torno a una jerarquía de objetos del servidor. En el nodo Bases de datos se encuentran todos los servicios que ofrece uno o más servidores, que se han definido previamente con la herramienta Net Configuration Assitant o manualmente en los ficheros **.ora**. Cada servicio o base de datos contiene, entre otros:

- El esquema que lo define. Este esquema está organizado por usuarios y, por cada usuario, los objetos que puede manejar (tablas, vistas, índices, etc.).
- Información de seguridad. Usuarios, roles y perfiles.

Con esta herramienta se pueden crear, borrar y modificar los mismos objetos y sus parámetros que con los lenguajes DDL y ACL. Las acciones de administración puntuales, como ALTERar una tabla, conceder permisos, se realizan de forma muy cómoda con esta herramienta.

8. Utilidades

En este apartado se describen brevemente algunas de las utilidades más prácticas y usadas de Oracle. Para obtener más ayuda sobre ellas hay que consultar el libro Utilities de la documentación de Oracle.

8.1. SQL*Loader

SQL*Loader es un programa que permite la carga de datos en tablas de Oracle a partir de ficheros de texto. Es útil cuando se importan datos de fuentes que no son una base de datos Oracle. Es una situación común cuando se migran datos de unos sistemas a otros y es necesario vigilar la correcta conversión de los datos entre los dos sistemas. Cuando los datos no cumplen las restricciones de integridad impuestas

sobre las tablas de la base de datos Oracle, SQL*Loader informa de ello tanto en pantalla como en un fichero de registro (log) que genera. También puede generar un fichero con todos los registros que no se han cargado satisfactoriamente.

Este programa se llama con el comando sqlldr desde una **interfaz de comandos del sistema operativo**, ***no*** desde la consola SQL*Plus o SQL Worksheet.

Este comando admite una serie de parámetros que guían su funcionamiento; en este apartado se repasarán más adelante los más comunes.

La forma más común de ejecutar SQL*Loader es:

sqlldr USERid=*usuario@servicio/contraseña*
control=*archivo_de_control* log=*archivo_de_informe*.

Donde ***usuario*** es el nombre de usuario definido en la base de datos bajo el cual se van a cargar los datos, ***servicio*** es el nombre del servicio (gestor de bases de datos) mediante el que se va a realizar la conexión, ***contraseña*** es la palabra clave del usuario de la base de datos, ***archivo_de_control*** es el nombre del archivo de control en donde se definen parámetros de funcionamiento y ***archivo_de_informe*** es el nombre del archivo que genera el programa para indicar el resultado de la operación. Los nombres de archivo pueden incorporar la ruta completa o usar nombres de archivo con rutas relativas. Lo más habitual es llamar al programa desde el directorio en el que se tengan los archivos de control y datos y especificar sólo los nombres de estos archivos.

El archivo de control puede tener la siguiente estructura:

```
LOAD DATA
[INFILE *|Fichero
   [BADFILE Fichero]
   [DISCARDFILE Fichero]
…]
INTO TABLE NombreTabla
[REPLACE|APPEND|]
[FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"']
( CAMPO1    [POSITION(CarIni:CarFin)]
  [, CAMPO2 [POSITION(CarIni:CarFin)]
)
[BEGINDATA
LíneaDatos1
LíneaDatos1
.
.
.
LíneaDatosn
]
```

El fichero de entrada puede ser el nombre del fichero o bien * que significa que los datos se incorporan al propio fichero de control.

Si el fichero de entrada es de formato variable se le indica el separador de campo.

Si el fichero de entrada es de formato fijo se le indica en cada campo las posiciones que ocupa.

Opcionalmente se pueden añadir los datos directamente en el fichero de control.

Las opciones más importantes son:

```
LOAD DATA
INFILE NombreFichero
INTO TABLE NombreTabla
APPEND
FIELDS TERMINATED BY ','
(Campo1, ..., CampoN)
```

Que hace referencia al fichero ***NombreFichero*** que contiene los datos de entrada estructurados como se indica a continuación:

```
valor11,...,valor1N,
valor21,...,valor2N,
...
valorM1,...,valorMN
```

Existen muchas más opciones como, por ejemplo, cargar en varias tablas en función del valor de un campo. Dos utilidades más son la forma de cargar fechas y aplicar funciones a los datos de entrada.

```
LOAD DATA
```

```
INFILE *
INTO TABLE Pacientes
REPLACE
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
( NOMBRE "upper(:nombre)",
  APELLIDOS "upper(:apellidos)",
  FEC_NACIMIENTO DATE 'DD/MM/YYYY HH24:MI:SS',
)
BEGINDATA
Pedro, Fernández Martín, 23/04/1956 13:24:23
```

Para cargar fechas se indica que el campo es de fecha poniendo después del nombre del campo la palabra DATE, un espacio y, entre comillas simples, el formato (por ejemplo: YYYYMMDD y DD-MM-YYYY).

Para aplicar una función SQL tenemos que tener en cuenta que cada campo posee una variable asociada que se llama :nombre_campo. Después del nombre del campo y entre comillas dobles se escriben las funciones a aplicar.

8.2. Import/Export

Las utilidades Import/Export son unas de las más antiguas de Oracle. Son herramientas basadas en línea de comandos que permiten extraer y cargar tablas, esquemas, objetos o la base de datos completa. A diferencia de SQL*Loader, la transferencia de datos se realiza entre instancias de Oracle. Son herramientas realmente útiles y de uso bastante sencillo. La exportación de los datos se realiza con el comando **`EXP`** y la importación con **`IMP`**.

8.2.1. Export

Su sintaxis es:

EXP parámetro=valor ó parámetro=(valor1,valor2, ...valorn)

Algunos de sus parámetros son:

Parámetro	**Valor predeterminado**	**Descripción**
BUFFER	Depende del SO	Tamaño del buffer para el array de comunicación con el servidor. Tamaños grandes de array mejoran la eficiencia.
COMPRESS	Y	Este parámetro es engañoso, no comprime los datos, sino que prepara la creación de la tabla a importar para que ésta se cree con una sola extensión (más óptimo).
CONSISTENT	N	Si se desea que la exportación se haga en una transacción de solo lectura
FILE	N/A	Nombre del archivo en el que se exportan los datos
FILESIZE	0	Si vale distinto de cero el tamaño máximo de fichero de exportación creado
FULL	N	Hacer una exportación completa de la base de datos
OWNER	N/A	Lista de esquemas a exportar de la lista de usuarios indicados
PARFILE	N/A	Nombre del fichero de parámetros (en vez de introducir éstos vía comando)
QUERY	N/A	Permite definir una cláusula WHERE, que se aplicará a todas las

		tablas, y sólo las filas que la cumplan se exportarán.
ROWS	Y	Si hay que exportar los datos o bien sólo se desea la estructura
TABLES	N/A	Lista de tablas a exportar encerradas entre paréntesis y separadas por comas.
TABLESPACES	N/A	Lista de tablespaces a transportar
TRANSPORT_ TABLESPACE	N	Si exportar metadatos para los tablespaces transportables
USERID	N/A	Nombre, contraseña y servicio bajo los que se realiza la exportación

Especificando **`exp help=y`** se muestra la ayuda.

Una de las aplicaciones más inmediatas es realizar una copia de seguridad del trabajo realizado en el laboratorio. La exportación de todos los objetos de un usuario se realiza con:
`exp USERid=`*`usuario/contraseña@servicio`*` owner=(`*`usuario`*`)`
`file=`*`nombre_archivo`*

Donde ***`usuario`*** es el usuario bajo el que se realiza la exportación, ***`contraseña`*** es su contraseña, ***`servicio`*** es el servicio en el que se ubica el usuario y ***`nombre_archivo`*** es el nombre de archivo en donde se alojarán los datos.

8.2.2. Import

Su sintaxis es:

IMP parámetro=valor ó parámetro=(valor1,valor2, ...valorn)

Algunos de sus parámetros son:

Parámetro	**Valor defecto**	**Descripción**
COMMIT	N	Si se hace commit después de cada array host o bien al finalizar la importación de la tabla.
FILE	N/A	Nombre del archivo desde el que se importan los datos
FROMUSER	N/A	Lista de usuarios a importar
IGNORE	N	Ignora la mayoría de los errores de creación de objetos. Muy útil si el esquema ya está creado en la BD.
INDEXFILE	N/A	Se especifica se importarán todos los CREATE INDEX y otros comandos DDL.
SHOW	N	Muestra las acciones que haría para importar, pero no almacena ninguno .
TABLES	N/A	Lista de tablas a importar encerradas entre paréntesis y separadas por comas.
TOUSER	N/A	Usuario donde se importan los datos.
TTS_OWNERS	N/A	Lista de los propietarios de objetos en los tablespaces transportables.
USERID	N/A	Nombre, contraseña y servicio bajo los que se realiza la importación

Especificando **IMP HELP=Y** se muestra la ayuda.

Siguiendo el ejemplo de la copia de seguridad del trabajo realizado en el laboratorio, la importación de todos los objetos del usuario a partir del archivo exportado anteriormente se realiza con:
IMP USERID=*usuario*/*contraseña*@*servicio* TOUSER=(*usuario*) FILE=*nombre_archivo*

Donde ***usuario*** es el usuario bajo el que se realiza la importación, ***contraseña*** es su contraseña, ***servicio*** es el servicio en el que se ubica el usuario y ***nombre_archivo*** es el nombre de archivo desde el que se leerán los datos.