Práctica 2. Tipos de datos y tablas

0. Introducción

Con esta práctica se inicia el aprendizaje del lenguaje SQL. En primer lugar se revisarán los tipos de datos soportados por Oracle y después se estudiará una primera aproximación de las sentencias que permiten la creación y el mantenimiento de los objetos Tabla.

1. Tipos de datos en Oracle

1.1 Carácter

Se puede representar con varios tipos:

- CHAR
- VARCHAR2
- LONG
- CLOB
- BLOB

Los tipos CHAR, VARCHAR2 y CLOB admiten cualquier tipo de carácter ASCII o EBCDIC, (dependiendo de la máquina con la que se trabaje). En cada caso puede definirse la longitud máxima entre paréntesis.

El tipo **CHAR** permite hasta un máximo de 2000 bytes, es de ***longitud fija*** y debe especificarse su tamaño en la definición. Si se inserta un valor de menor tamaño en un campo CHAR, se rellena a blancos; si el valor es de mayor tamaño manda un mensaje de error.

El tipo **VARCHAR2** permite un tamaño máximo de 4000 bytes, y ocupa exactamente el espacio necesitado. Opcionalmente se puede especificar el tamaño máximo del campo. Por tanto, se utilizará para cadenas de caracteres de ***longitud variable*** (sólo ocupan espacio los caracteres introducidos y no la longitud total definida para el campo). VARCHAR es un sinónimo de VARCHAR2.

Los tipos **CLOB** y **BLOB** se utilizan para almacenar grandes bloques de información no estructurada (texto, imágenes, videos, sonidos, etc.). Tiene una longitud máxima de 8 terabytes para los CLOB que almacenan caracteres y 128 terabytes para los BLOB que almacenan datos binarios. El tipo LONG tiene una longitud máxima de 2 GB y se mantiene por compatibilidad con versiones anteriores. Se recomienda utilizar el tipo CLOB en vez de LONG.

Las constantes de tipo cadena se encierran entre comillas simples.

1.2 Numérico

Se definen con la palabra reservada NUMBER. Sintaxis:

NUMBER [(PRECISION [, ESCALA])]

- **PRECISION**: Número de dígitos del número, contándose los decimales. Su rango de valores es de 1 a 38 (por defecto la precisión es 38). Si se utiliza este parámetro, el número se tratará como un número decimal de coma fija con la precisión dada, y si no se especifica será de coma flotante con precisión 38.
- **ESCALA**: Número de dígitos decimales. Su rango es de -84 a 127.

Ejemplos:

VALOR	DECLARACIÓN	ALMACENADO
7.456.123,89	NUMBER	7456123,89
7.456.123,89	NUMBER(9)	7456124 (1)
7.456.123,89	NUMBER(9,2)	7456123,89
7.456.123,89	NUMBER(9,1)	7456123,9 (2)
7.456.123,89	NUMBER(6)	Exceso precisión (3)
7.456.123,8	NUMBER(15,1)	7456123,8
7.456.123,89	NUMBER(7,-2)	7.456.100 (4)
7.456.123,89	NUMBER(7,2)	exceso precisión

Notas:

(1) Si se declara un número entero y se le pasa un valor decimal, redondea dicho valor.
(2) Si existe un mayor número de dígitos decimales que los que se han declarado en la escala, se redondea a partir de los valores sobrantes.
(3) Si el número total de dígitos de la parte entera es mayor que la diferencia entre la precisión y la escala declarada, se produce un error de exceso de precisión.
(4) Si la escala es negativa se redondea el número a la izquierda del punto decimal tantas posiciones como indique la escala.

Se utiliza por defecto como separador decimal el punto.

1.3 Fecha

Se define con la palabra reservada **DATE**, que almacena la **información del año, mes, día, hora, minutos y segundos**. Sin, embargo, cuando se hace una consulta sobre un campo de tipo DATE, Oracle, por defecto, aplica una máscara y solo muestra el día, mes y año. Si queremos mostrar la información completa debería aplicarse la función TO_CHAR, indicando con qué formato queremos obtener la salida:

TO_CHAR (fecha, 'DD-MM-YYYY HH24:MI:SS') obtendría el resultado 05-07-2013 23:37:19
TO_CHAR (fecha, 'DD-MON-YY HH:MI:SS AM') obtendría 05-JUL-13 11:38:16 PM

También podemos obtener parte de una fecha (el año, mes o día) con la función EXTRACT. Por ejemplo EXTRACT (YEAR FROM SYSDATE) obtendría el año de la fecha actual, ya que la función del sistema **SYSDATE devuelve el valor de la fecha actual**.

Las operaciones "+" y "-" con fechas obtienen valores en formato de días con fracción de días. También podemos sumar a una fecha un número de días y obtendríamos una nueva fecha.

Si se desea almacenar fracciones de segundo se utiliza el tipo TIMESTAMP (n) donde n es el número de dígitos en la parte fraccional de segundos hasta un máximo de 9. TIMESTAMP también permite indicar el desplazamiento de la fecha en función de la zona horaria añadiéndole la cláusula WITH LOCAL TIMEZONE.

Las constantes de tipo fecha van encerradas entre comillas simples.

1.4 Otros Tipos

- BFILE. Actúa como un puntero a ficheros de datos en disco. El puntero está formado por dos partes: el nombre del fichero y el id del directorio donde se encuentra almacenado. Proporciona acceso de sólo lectura a los datos. Es un tipo de datos LOB que contiene una referencia a datos binarios de un tamaño máximo de 4GB.

- XML. Permite almacenar datos XML de hasta 4GB.

1.5 Tipos especiales del sistema

Existen una serie de tipos utilizados internamente por el sistema, para representar distintos datos manejados por ORACLE y que pueden ser accedidos por el usuario, como son: ROWID, ROWNUM.

1.5.1 ROWID

El más importante de estos tipos es el tipo **ROWID**, utilizado para almacenar la dirección física de cada tupla de la base de datos.

Ejemplo: Un valor de ROWID podría ser:

0000000F.0000.0002 = id_bloque.id_tupla.id_fichero

donde:

- ***id_bloque*** identifica el bloque de datos dentro del fichero de datos que contiene la tupla (su tamaño depende del sistema operativo).
- ***id_tupla*** identifica la tupla dentro del bloque (la 1ª tupla tiene el número cero).
- ***id_fichero*** identifica el fichero de base de datos que contiene la tupla (empezando por uno).

1.5.2 ROWNUM

Para cada tupla devuelta por la consulta, la pseudocolumna ROWNUM devuelve un número indicando el orden en que ORACLE selecciona la tupla desde una tabla o desde el resultado de un join de varias tablas.

2. Valores nulos

NULL es un valor predefinido en SQL, para indicar que el valor del atributo de la tupla correspondiente es desconocido. Cualquier atributo de una tabla puede contener valores nulos excepto si ha sido declarado como NOT NULL o forma parte de una clave primaria.

Un valor nulo:

- Es apropiado cuando el actual valor es desconocido.
- No es equivalente a cero.

Se puede usar la función NVL en Oracle para convertir temporalmente un valor nulo a otro valor. Por ejemplo NVL (cantidad,0) retornará 0 si cantidad es nulo; en otro caso retornará el valor de cantidad. Las funciones de grupos ignoran los valores nulos.

Ejemplo:

```
AVG (1000,NULL,NULL,NULL,2000)=3000/2=1500
```

El único operador de comparación que se puede usar con valores nulos es **IS NULL** o **IS NOT NULL**. Si se usa otro comparador el resultado será siempre desconocido. ORACLE trata las condiciones evaluadas como desconocidas como FALSE. ORACLE no devuelve un error en estas situaciones.

3. Tablas

3.1 Definición

Una tabla es un objeto de la base de datos que almacena los datos. La información sobre la definición de cada tabla y sus características se almacena en el diccionario de datos.

3.2 Creación de una tabla

Sintaxis Oracle:

```
CREATE TABLE nombre_tabla
({nombre_atributo tipo [DEFAULT valor][NOT NULL]{restricción_atributo},
      ...}
 { restricción_de_tabla ,}
);
```

Ejemplo:

```
CREATE TABLE ordenadores
(     identificador     NUMBER(6) NOT NULL PRIMARY KEY,
      numero_serie      VARCHAR2(15) NOT NULL UNIQUE,
      fecha_adquisicion DATE DEFAULT sysdate,
      procesador        VARCHAR2(10),
      memoria           NUMBER(5) CHECK (memoria >=1024),
      disco_duro        NUMBER(4),
      usuario           VARCHAR2(10)
);
```

Los tipos válidos son los definidos en el apartado anterior.

La opción DEFAULT permite asignar un valor por defecto, cuando no se proporcione al insertar una nueva tupla en la tabla.

La opción NOT NULL obliga a que exista un valor asignado al atributo para cada tupla.

La restricción UNIQUE en *numero_serie* hace que el valor de este atributo no pueda repetirse para distintas filas.

La restricción CHECK en el atributo *memoria* obliga a que todas las filas de la tabla tengan un valor en memoria mayor o igual a 1024.

La opción PRIMARY KEY designa al atributo o conjunto de atributos que son clave primaria, lo que significa que los valores del atributo (o la combinación de los valores de los atributos, en caso de que la clave primaria esté formada por varios) no pueden repetirse para las distintas filas. Esta opción puede especificarse como restricción de atributo o como restricción de tabla, pero es obligatoria su especificación a nivel de tabla si la clave primaria está formada por varios atributos.

La definición completa de restricciones a nivel de atributo y de tabla se estudiará en prácticas posteriores.

3.3 Borrado de una tabla

Sintaxis Oracle:

```
DROP TABLE nombre_tabla [CASCADE CONSTRAINT];
```

CASCADE CONSTRAINT: Borra todas las restricciones de integridad referencial que hacen referencia a las claves primaria o única de la tabla a borrar (en otras tablas). Se verá mas adelante.

Ejemplo: Borrado de la tabla de prueba

```
DROP TABLE ordenadores;
```

3.4 Modificación de la estructura de una tabla

Se permiten las siguientes modificaciones sobre la tabla:

- Añadir un atributo
- Añadir una restricción de tabla o de atributo

- Redefinir un atributo (tipo, tamaño, valor por defecto)
- Modificar los parámetros de almacenamiento de la tabla.

Sintaxis:

```
ALTER TABLE nombre_tabla [ADD | MODIFY]
[ ( { nom_atributo [DEFAULT valor]  restricciones_atributo , }
    { restricciones_tabla }
  )
] ;
```

Ejemplo: Añadir un atributo a la tabla de ordenadores que se denomine precio de compra y que puedan contener hasta un importe máximo de 9000 euros.

```
ALTER TABLE ordenadores
      ADD (precio_compra NUMBER(6,2));
```

3.5 Cambio de nombre de una tabla

Sintaxis:

```
RENAME nombre_antiguo TO nombre_nuevo;
```

3.6 Visualización de la estructura de una tabla

Nos muestra por cada atributo de una tabla:

- Nombre
- Tipo
- Restricción de valores nulos
-

Sintaxis:

```
DESC nombre_tabla
```

Anexo I: Ejercicios

El objetivo de estos ejercicios es crear una tabla denominada Pelicula, usando las diferentes formas de interacción con la base de datos estudiadas.

Descripción de la tabla de películas.

Se pretende crear una tabla para almacenar la información de las películas que tiene un conjunto de tiendas de una cadena de videoclubs. Después de un análisis, la información que se debe almacenar es la siguiente:

- *Código de Película*. Su valor podrá estar comprendido entre 1 y 99999. Es la clave primaria.
- *Título.* Su valor será alfanumérico variable con hasta 255 caracteres.
- *Sinopsis*. Su valor será alfanumérico variable con hasta 2000 caracteres.
- *Año.* Año de estreno de la película. Valor entre 1900 y 2050
- *Periodo de alquiler.* Número de días de alquiler. Valor entre 1 y 30. Por defecto, si no se indica ninguna al dar de alta la película, será 3.
- *Precio de alquiler.* Valor entre 0.5 y 99 euros. Por defecto, si no se indica ninguna al dar de alta la película, será 3.
- *Duración*. Duración de la película en minutos. Valor máximo 900.
- *Fecha de alta*. Fecha en la que se ha dado de alta la película. Si no se especifica un valor, se incluirá la fecha del sistema.

Debe analizarse qué información podría admitirse que no tuviese valor asignado y cuál debe estar siempre rellena y, de acuerdo a ello, definir las restricciones correspondientes.

Ejercicio 1

Conéctate a la base de datos mediante el **SQL Developer**. Examina los objetos de que dispones para confirmar que no existe la tabla Película.

Abre una hoja de trabajo y escribe una sentencia para crear la tabla película con la descripción dada anteriormente. Ejecuta la sentencia y guárdala en un fichero llamado *crea_pelicula.sql.*

Comprueba que ahora ya dispones de la tabla de pelicula entre tus objetos (puede ser necesario refrescar el SQL Developer). Examina la definición de las columnas y ejecuta una consulta que muestre el contenido de la tabla. Cierra la conexión.

Ejercicio 2

En este caso en lugar de ejecutar la sentencia para la creación de la tabla Película de manera interactiva, se desea que se haga mediante un fichero por lotes (.bat) que llame a un fichero SQL. A continuación se se dan las indicaciones oportunas:

- Usa el fichero *crea_pelicula.sql* que has creado anteriormente y añade las siguientes operaciones:
    - Antes de la creación de la tabla:
        - Activa un spool pelicula.txt
        - Borra la tabla de pelicula
    - Después de la creación de la tabla
        - Muestra la estructura de la tabla creada
        - Cierra el spool
        - Finaliza la sesión sql
- Crea un fichero de lotes .BAT de MS-DOS que ejecute el fichero SQL anterior.
- Ejecuta el fichero .BAT y comprueba que la sentencia se ha ejecutado correctamente examinando el contenido del fichero de spool.