Práctica 1 PL/SQL:

Reserva de Pistas de Tenis

En esta práctica vamos a trabajar con 2 tablas que son:

1. ***Pistas (nro)*** cuya PK es *nro* y que representa las pistas de un club de tenis.
2. ***Reservas (Fecha, Hora, Pista, Socio)*** cuya PK es la compuesta por *Fecha*, *Hora* y *Pista*, y que mantiene una FK hacia *pistas* mediante el campo Pista que representa el *nro* de pista. *Fecha* es un campo *date* y *Hora* es un campo *integer* que toma valores de 0 a 23. *Socio* es un *VARCHAR(30)* con el nombre del socio.

Junto con el enunciado se provee el script **Práctica1_Tenis_Sol.sql** que:

1. Borra las tablas, las crea de nuevo y las rellena con varias filas de ejemplo.
2. En él se implementan las siguientes transacciones:

```
create or replace function
 anularReserva(   p_socio varchar, p_fecha date,
                  p_hora integer, p_pista integer )
return integer is
```

que busca una reserva del socio del primer parámetro de la pista, fecha y hora de los tres últimos parámetros, y devuelve verdadero si la anulación tiene éxito, falso en caso contrario.

y

```
create or replace function
     reservarPista( p_socio varchar, p_fecha date, p_hora integer )
return integer
```

que busca una pista libre para esa hora y día, si la encuentra inserta la reserva a nombre del socio socio. El método devuelve verdadero si la reserva tiene éxito, falso en caso contrario.

Se pide:

Paso 1: Leer el código fuente de ambas funciones investigando las siguientes cuestiones:

1. ¿Por qué en las comparaciones de fecha en Oracle conviene utilizar la función *trunc*?
2. ¿Qué es sql%rowcount y cómo funciona?
3. ¿Qué es una variable de tipo cursor?. ¿Qué variable de tipo cursor hay en la segunda función? ¿Qué efecto tienen las operaciones open, fetch y close?. ¿Qué valores toman las propiedades de cursor FOUND y NOTFOUND y en qué caso?
4. En la función anularReserva discute si da lo mismo sustituir el rollback por un commit y por qué
5. En la función reservarPista investiga si la transacción se puede quedar abierta en algún caso. Haz el arreglo correspondiente para que esto no ocurra.

Paso 2: Añade al final del script:

1. Un bloque anónimo PL/SQL que:
   a) Invoque a la función para reservar pista, haciendo 3 nuevas reservas válidas de las pistas existentes:

```
reservarPista( 'Socio 1', CURRENT_DATE, 12 );
reservarPista( 'Socio 2', CURRENT_DATE, 12 );
reservarPista( 'Socio 3', CURRENT_DATE, 12 );
```

   y una cuarta reserva no válida por no haber ya hueco:

```
reservarPista( 'Socio 4', CURRENT_DATE, 12 );
```

   b) Invoque a la función para anular reservas, primero de una de las reservas anteriores:

```
anularreserva( 'Socio 1', CURRENT_DATE, 12, 1);
```

   luego de una reserva inexistente

```
anularreserva( 'Socio 1', date '1920-1-1', 12, 1);
```

2. Una SELECT que permita verificar visualmente que todo ha funcionado bien.

Paso 3: Estudia en el vídeo de ubuVirtual "Depuración en SQLDeveloper" cómo manejar el debugger de SQLDeveloper.

Convierte el bloque anónimo en un `PROCEDURE TEST_FUNCIONES_TENIS` sin argumentos. Prueba a ejecutarlo con un bloque anónimo y con exec. Utiliza el debugger si algo sale mal.

Paso 4: Experimenta el *debugger* invocando una llamada a cada una de las 2 funciones. En esa ocasión, haz:

1. El script a ejecutar como administrador que de los privilegios necesarios al usuario hr para poder utilizar el *debugger*. Ejecútalo.
2. Verifica que funciona bien ejecutando paso a paso una nueva llamada que haga una otra reserva correcta (no olvides compilar la función "para depuración").

Paso 5: Crea un nuevo script similar al anterior titulado script_tenis_procedures.sql. Reprograma en el mismo:

1. Las 2 funciones como procedures pReservarPista y pAnularReserva respectivamente.

   Nota que ya no hay sentencias RETURN. En el caso de que haya algún problema, en lugar de utilizar `RETURN 0 como antes, ahora` debes de lanzar una excepción apropiada. Declara para ello las siguientes excepciones:

   En pAnularReserva:

```
raise_application_error(-20000, 'Reserva inexistente');
```

   En pReservarPista:

```
raise_application_error( -20001, 'No quedan pistas libres en esa
fecha y hora');
```

2. Se provee además al alumno del procedure TEST_PROCEDURES_TENIS que prueba los 2 procedimientos anteriores. Trata de comprenderlo con ayuda del profesor
   a) ¿Por qué el Test5 pasa satisfactoriamente aun cuando no hay nada programado?
   b) ¿Qué te parece más sencillo? ¿utilizar cursores explícitos o implícitos para detectar que las reservas y anulaciones han modificado correctamente la BD? ¿por qué?
   c) ¿Por qué ponemos rollback en los tests que no devuelven excepciones?, ¿qué estamos intentando detectar?

Paso 6: Crea un nuevo script similar al anterior titulado *test_mejorado.sql*. Reprograma en el mismo el procedure `test_reservar_pista` a partir de las ideas que saques de `test_anular_reserva`

Paso 7: Discute si ocurre algún error lógico cuando 2 transacciones `pReservarPista` que se solapan en el tiempo quieren ambas reservar la última pista que queda libre en una determinada fecha y hora y si sugieres algún arreglo al respecto. Implementa el arreglo en tal caso.

Entregable

La entrega es voluntaria y

- No cuenta para nota (simplemente es para recibir una realimentación de si lo estás haciendo bien o en qué fallas por parte del profesor)
- El número de miembros de cada grupo de trabajo no está limitado.
- Hay que entregar el *script* **hasta donde hayas podido completarlo** (actividades de los pasos 2, 3, y

No es necesario que lo entregues al completo si no te ha dado tiempo, pero cuanto más completes, más practicas.