Práctica 1: Reserva de Pistas de Tenis

Sean las tablas:

1. *Pistas (nro)* cuya PK es *nro* que representa las pistas de un club de tenis
2. y *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.

Se provee el script **`Practica_PL_SQL_Tenis.sql`** que:

1. Borra las tablas, las crea de nuevo y las rellena con varias filas de ejemplo.
2. En el 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 comparacionesde 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:
    1. 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 );
```

2. 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. 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 4: Convierte el bloque anónimo en un `PROCEDURE TEST_FUNCIONES_TENIS` sin argumentos. Prueba a ejecutarlo con un bloque anónimos y con exec. Utiliza el debugger si algo sale mal.

Deja como un comentario el bloque anónimo obtenido el el paso 2 para que lo pruebe el profesor.

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

1. Las 2 funciones como procedures `pReservarPista` y `pAnularReserva` respectivamente.
2. El procedimiento `TEST_FUNCIONES_TENIS` como otra versión `TEST_PROCEDURES_TENIS` que pruebe los 2 procedimientos anteriores (vete haciéndolo a la par que vas generando los procedimientos)

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

3. En `pAnularReserva`:
   ```
   raise_application_error(-20000, 'Reserva inexistente');
   ```
4. En `pReservarPista`:
   ```
   raise_application_error(
   -20001, 'No quedan pistas libres en esa fecha y hora');
   ```

Paso 6: Discute en un comentario de este último *script* si 2 transacciones ReservarPista se solapan en el tiempo queriendo 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

- Esta entrega NO es obligatoria
- No cuenta para nota (simplemente es para recibir una realimentación de si lo estás haciendo bien o en qué fallas por parte del profesor)
- Se **recomienda** el uso de un **repositorio GIT** en la carpeta entregada, y deberá tener varias confirmaciones (commits) a medida que el proyecto se va desarrollando. Te recomiendo un commit por pregunta resuelta. En futuras entregas será un requisito, por lo que se recomienda su uso

La entrega constará de **un único fichero comprimido** que alojará el contenido de la carpeta en la que se ha desarrollado el proyecto. Contendrá todos los *scripts* PL/SQL (formato SQL) generados para resolver el enunciado y que respondan a TODAS las cuestiones realizadas. Las respuestas prosaicas se añadirán como comentarios en el propio *script*.