Estructura de Tablas

Dada la tabla de recorridos:

> *Recorridos (idRecorrido(PK), estacionOrigen, estacionDestino, kms, horaSalida, horaLlegada, precio)*
>
> Que representa un recorrido del autocar que podría repetrise en varias fechas, caracterizado por una estación origen otra destino, una hora de salida y otra de llegada.
>
> Como en Oracle no hay campos TIME las horas de salida y llegada se han representado mediante campos TIMESTAMP en los que despreciaremos la fecha.

la tabla de viajes:

> *Viajes (idViaje(PK), idAutocar, idRecorrido --> FK a recorridos, fecha, nPlazasLibres, realizado, idConductor)*
>
> que representa el viajes que hace un autocar en un determinado recorrido y en una determinada fecha.
>
> ***`NplazasLibres`*** representa el número de plazas libres que quedan en el autocar.
>
> ***`realizado`*** es un booleano que indica si el viaje se ha realizado o no, y el ***`idConductor`*** identifica al conductor (no intervienen en este problema).
>
> Como en Oracle no hay campos BOOLEAN se representa mediante un dígito que toma los valores 0/1.

la tabla de tickets

> *Tickets (idTicket(PK), idViaje (FK ---> Viajes, fechaCompra, cantida, precio)*
>
> Que representa un ticket o un conjunto de varios tickets (por eso hay un campo ***`cantidad`*** que indica cuantos) correspondientes a un viaje.
>
> El campo ***`precio`*** contiene la multiplicación del campo ***`cantidad`*** por el campo ***`recorrido.precio`***, correspondiente al recorrido de ese viaje.

El script que crea las tablas está en `ComprarBilletesBus_Enun.sql`. El resto de tablas del script no intervienen en este problema.

Descripción General

Se pide implementar el procedimiento almacenado:

```
create or replace procedure comprarBillete(
    arg_hora recorridos.horaSalida%type,
    arg_fecha viajes.fecha%type,
    arg_origen recorridos.estacionorigen%type,
    arg_destino recorridos.estaciondestino%type,
    arg_nroPlazas viajes.nplazaslibres%type

)
```

que dada una hora, una fecha, una ciudad origen y una ciudad destino que sirven para caracterizar un viaje, inserta una fila en la tabla de *tickets* por la compra de `arg_nroPlazas`, decrementando el numero de plazas libres de ese viaje.

Algoritmo

Para ello, se sugiere seguir estos pasos (pero se puede hacer de otras formas):

1. Consultar el ***`id_viaje`*** correspondiente a la fecha `arg_fecha`, origen `arg_origen`, destino `arg_destino` y hora de salida `arg_hora` pasados por parámetro. Es recomendable que en la misma consulta obtengasel ***`precio`*** que tiene el recorrido de ese viaje.

2. Decrementar en la tabla de viajes el número de plazas libres ***`NplazasLibres`*** según el parámetro `arg_nroPlazas`.
3. Insertar el ticket correspondiente, teniendo en cuenta que el ***`idTicket`*** sale de la secuencia `seq_tickets` y que la fecha de compra es la del sistema.

Excepciones

El procedimiento debe poder lanzar las siguientes excepciones:

1. La primera registrará el problema de que **no queden plazas suficientes para ese viaje**. Su código será el "-20.001" y el mensaje de error `"Plazas insuficientes. Se solicitaron '||arg_nroPlazas||' y solo quedan '||v_numPlazasLibres"`. (Nota que tienes que consultar cuantas plazas libres quedan y asignárselas a la variable `numPlazasLibres`, para lo cual puedes aprovechar la misma consulta del paso 1 del Algoritmo.
2. La segunda registrará el problema de que **no exista un viaje que corresponda con ese origen, destino, fecha y hora de salida**. Su código será el "-20.002" y el mensaje de error `"No existe ese viaje para esa fecha, hora, origen y destino."`.

Toda excepción sea del tipo que sea, retrocederá la transacción. Las excepciones **que no sean** las dos anteriores se relanzarán con su código y mensaje de error.

Pruebas automáticas

Se pide realizar otro procedimiento almacenado que realice pruebas automáticas del `procedure` **`comprarBillete`**. Esas pruebas harían 3 cosas

1. Comprar un billete para un viaje inexistente
   ```
   comprarBillete( '1/1/1 12:00:00', '15/04/2010', 'Burgos', 'Madrid', 3);
   --Pongo la fecha 1/1/1 para acomodar un timestamp en lo que debiera ser un time, pero valdria cualquier otra fecha
   ```
   - Caso de no saltar ninguna excepción mostrará el mensaje `Mal no detecta NO_EXISTE_VIAJE`.
   - Caso de saltar una excepción que no sea la -20.002 mostrará `'Mal no detecta NO_EXISTE_VIAJE:'` junto con mensaje de erro de la excepción que realmente se ha producido.
2. Comprar un billete de 50 plazas en un vieje que no tiene tantas plazas libres.
   ```
   comprarBillete( '1/1/1 8:30:00', '22/01/2009', 'Burgos', 'Madrid', 50);
   ```
   - Caso de no saltar ninguna excepción mostrará el mensaje `Mal no detecta NO_PLAZAS`.
   - Caso de saltar una excepción que no sea la -20.002 mostrará `'Mal no detecta NO_PLAZAS:'` junto con mensaje de erro de la excepción que realmente se ha producido.
3. Finalmente hacemos una compra de un billete de 5 plazas sin problemas.
   ```
   comprarBillete( '1/1/1 8:30:00', '22/01/2009', 'Burgos', 'Madrid', 5);
   ```
   Comprobamos que hay un único ticket con la fecha de hoy[1] y es el número `idticket` igual a 3 (debido al nextval del la secuencia) que genera la cadena `'11122/01/0925113550"` al concatenar los campos del join de viajes con tickets en este orden `IDVIAJE||IDAUTOCAR||IDRECORRIDO||FECHA||NPLAZASLIBRES||REALIZADO||IDCONDUCTOR||IDTICKET||CANTIDAD||PRECIO`

1 Oracle internamente guarda TIMESTAMPs en los campos DATE. Para extraer la fecha sin la hora usa TRUNC(*expresión de tipo fecha*)

- Si el valor que vevuelve la consulta no coincide con el esperado, se mostrará '`Mal: no modifica bien la BD.`', junto con la cadena que devuelve la consulta y la esperada.
- Si coinciden los dos valores se mostrará '`BIEN: si modifica bien la BD.`'
- Y si surgiera alguna excepción que la relance (es decir, no hace falta hacer tratamiento de la excepción)

El mensaje que ha de mostrarse cuando todo funciona debiera ser el siguiente:

```
-- Mensajes de creación de tablas, procedmientos, inserción de filas etc ...
-- que omitimos

Detecta OK NO_EXISTE_VIAJE: ORA-20002: No existe ese viaje para esa fecha, hora,
origen y destino.
Detecta OK NO_PLAZAS: ORA-20001: Plazas insuficientes. Se solicitaron 50 y solo
quedan 30
BIEN: si modifica bien la BD.
```

Entregable

- La entrega es obligatoria
- Cuenta para nota en el bloque de Prácticas PL/SQL
- Es un trabajo individual

Hay que entregar 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*.

Recuerda que **se requiere** el uso de un **repositorio GIT** en la carpeta entregada, y deberá tener varias confirmaciones (commits) a medida que el proyecto se va desarrollando.

Fecha límite de entrega: **26/06/2024 23:55**