Alquiler de Vehículos

Sean las tablas:

1. *Precio_Combustible (Tipo_Combustible, precio_por_litro*) Es una tabla de dos filas una con el precio actual de la gasolina y otra con el del diésel.
2. *Modelos (id_modelo (PK), nombre, precio_cada_dia, capacidad_deposito, tipo_combustible (FK → Precio_Combustible)*
3. *Vehículos (matricula (PK), id_modelo (FK → Modelos), color, alquiladoA (FK → Clientes)* El campo alquilado puede estar a nulo si el vehículo está disponible.
4. *Clientes (NIF (PK), nombre, ape1, ape2, dirección)*
5. *Reservas (idReserva (PK), cliente (FK → Clientes), matricula (FK → Vehículos), fecha_ini, fecha_fin)*
6. *Facturas (nroFactura (PK), importe, cliente (FK → Clientes))*.
7. *Lineas_Factura (NroFactura (FK → Facturas), concepto, importe) [PK compuesta NroFactura + Concepto]*

Además existen las secuencias de la figura:

Nombre de la secuencia	**Alimenta los valores de**
`seq_modelos`	La clave primaria de la tabla modelos: *id_modelo*
`seq_num_fact`	La clave primaria de la tabla de facturas: *nroFactura*
`seq_reservas`	La clave primaria de la tabla de reservas: *idReserva*

Se te provee ya de un script que crea las tablas **`AlquilerCoches_Enun.sql`** que además contiene:

1. El procedimiento almacenado **`reset_seq( p_seq_name varchar )`** que sirve para resetear las secuencias utilizado en prácticas anteriores
2. Un procedimiento almacenado **`inicializa_test`** que reinicia el contenido de la base de datos y la provee de unas filas de prueba (**`inicializa_test`** llama a **`reset_seq`**).
3. Un procedimiento almacenado con los tests automáticos **`test_alquila_coches`** que llama a **`incializa_test`** cada vez que prueba un caso.

Se pide: Implementar la transacción:

```
create or replace procedure alquilar( arg_NIF_cliente varchar,
 arg_matricula varchar, arg_fecha_ini date, arg_fecha_fin date) is
```

que:

1. Comprueba si la fecha de inicio pasada como argumento no es posterior a la fecha fin pasada como argumento. En caso contrario devolverá el error -20003 con el mensaje '*El numero de dias sera mayor que cero*.'
2. En caso contrario utiliza una SELECT con un par de *joins* para saber el valor del modelo del vehículo pasado como argumento, el prcio de alquilarlo diariamente, la capacidad de su depósito de combustible, el tipo de combustible que utiliza y el precio por litro del mismo.

   Del resultado de esta SELECT deberías ser capaz de deducir si el vehículo existe. Si no existiese has de devolver el error -20002 con el mensaje '*Vehiculo inexistente.*'.

   Se recomienda utilizar esta SELECT para bloquear el vehículo y que nadie pueda insertar reservas que se solapen con la que estoy insertando antes de que finalice mi transacción

(lee el siguiente artículo, sobretodo la parte final https://asktom.oracle.com/pls/asktom/f?p=100:11:::::P11_QUESTION_ID:42171194352295).

Asegurate que lo que estás bloqueando en ese *join* es el vehículo y no el modelo o el tipo de combustible, (aquí tienes alguna idea: https://asktom.oracle.com/pls/apex/asktom.search?tag=for-update-with-a-join).

3. Siguiendo la misma estrategia del artículo anteriormente referenciado utilizar una SELECT para saber si en el intervalo entre `arg_fecha_ini` y `arg_fecha_fin` no existe ya alguna reserva solapada en la tabla de reservas.

   Si existiese habría que devolver la excepción -20004, con el mensaje *'El vehiculo no esta disponible.'*.

4. Insertamos una fila en la tabla de reservas para el cliente, vehículo e intervalo de fechas pasado como argumento. En esta operación deberíamos ser capaces de detectar si el cliente no existe, en cuyo caso lanzaremos la excepción -20001, con el mensaje '*Cliente inexistente*'.

   Piensa; el resultado de la SELECT del paso anterior ¿sigue siendo fiable en este paso?:

   1. **En este paso, la ejecución concurrente del mismo procedimiento ALQUILA con, quizás otros o los mimos argumentos, ¿podría habernos añadido una reserva no recogida en esa SELECT que fuese incompatible con nuestra reserva?, ¿por qué?.**

   2. **En este paso otra transacción concurrente cualquiera ¿podría hacer INSERT o UPDATE sobre reservas y habernos añadido una reserva no recogida en esa SELECT que fuese incompatible con nuestra reserva?, ¿por qué?.**

   Explica los por qués en un comentario sobre el script.

5. Se crea una factura correspondiente a alquilar al cliente con ese NIF el vehículo con esa matrícula durante los días transcurridos: *n_dias = fecha_fin – fecha_ini*.

   El campo importe de la factura se rellena con la suma de los importes de las líneas de factura, que se crearán como se indica más adelante. Conviene que tengas esa suma hallada en este paso para no tener que hacer un UPDATE innecesario sobre la factura.

   Piensa por qué en este paso ninguna transacción concurrente podría habernos borrado el cliente o haber cambiado su NIF. Explica los por qués en un comentario sobre el script.

   Si fecha_fin es *null*, se entiende que la fecha final no está cerrada.

   La transacción creará además dos líneas de factura:

   1. Correspondiente a lo que cuesta alquilar el vehículo *n_días*, para ello en la primera SELECT tiene el precio del alquiler diario del vehículo que le multiplicará por *n_días*. El concepto de la línea será "*tantos* días de alquiler vehículo modelo *el-que-sea*". Donde *tantos* es lo que valga *n_días* y *el-que-sea* será el modelo del vehículo, que se puede saber también con esa primera SELECT.

      Si la fecha final no está cerrada se le cobra 4 días.

   2. Otra línea de factura correspondiente a un depósito lleno de gasolina de ese vehículo. Para ello, la primera SELECT ya nos da la capacidad en litros de ese modelo, y el precio del combustible de ese modelo, los cuales multiplicaremos. En el concepto figurará "Deposito lleno (X litros de Y)", donde X es el número de litros e Y el tipo de combustible).

   (Utiliza CURRVAL para rellenar el campo NroFactura de la tabla de *Lineas de Factura*).

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.