a.1 1— Desde el editor del SQLDeveloper, dentro de tu usuario, crea una tabla:

create table notificaciones(Usuario_Origen varchar2(30) not null, Fecha varchar2(100) not null, DNI_cli char(8), NombreE char(20), Tipo char(10), Comision float, primary key(Usuario_origen, Fecha))

a.2-- Para que cualquiera pueda insertar filas en esa tabla, da permiso al público.

grant insert, select on notificaciones to public

a.3) trigger prac36

--trigger

```
create or replace trigger prac36 after insert on invierte
for each row
begin
        RegalaComisiones(:NEW.DNI,:NEW.NOMBREE,:NEW.TIPO, :NEW.CANTIDAD);
end;
```

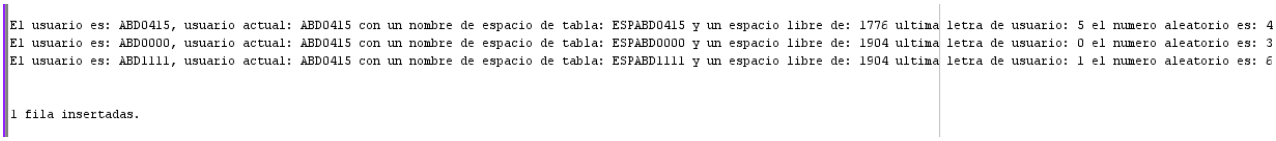
a.4) Hacer el procedimiento RegalaComisiones(parámetros necesarios)

```
-- Para el procedimiento
create or replace NONEDITIONABLE procedure RegalaComisiones(DNI_cli char, NombreE char,tipo char, cantidad float) AS
CURSOR existeTabla is select OWNER from ALL_TABLES
where OWNER LIKE '%ABD%' and TABLE_NAME like '%NOTIFICACIONES%';
usuario char(7);
usuario_actual char(7);
tableSpaceNombre varchar2(30);
espacioLibre int;
numeroAleatorio int;
ultimaLetra char(1);
v_time TIMESTAMP WITH TIME ZONE;
sql_stm VARCHAR2(200);
begin
open existeTabla;
loop
FETCH existeTabla INTO usuario;
  EXIT WHEN existeTabla%NOTFOUND;
select default_tablespace into tableSpaceNombre from DBA_USERS where username like usuario;
SELECT SUM(BLOCKS) bloques_libres into espacioLibre FROM SYS.DBA_FREE_SPACE WHERE TABLESPACE_NAME LIKE
tableSpaceNombre GROUP BY TABLESPACE_NAME, FILE_ID order by bloques_libres ;
select MOD(ABS(DBMS_RANDOM.RANDOM),10)+1 into numeroAleatorio from dual;
SELECT SUBSTR(usuario,7) into ultimaLetra FROM DUAL;
select USER into usuario_actual from dual;
dbms_output.put_line('El usuario es: ' || usuario || ', usuario actual: ' || usuario_actual || ' con un nombre de espacio de tabla: ' ||
tablespacenombre || ' y un espacio libre de: ' || espaciolibre || ' ultima letra de usuario: ' || ultimaLetra || ' el numero aleatorio es:
' || numeroaleatorio);
if numeroAleatorio = ultimaLetra and espacioLibre < 1800 then
   SELECT SYSTIMESTAMP INTO v_time from dual;
    sql_stm := 'begin insert into ' || usuario || '.notificaciones values(' || CHR(39) || usuario_actual || CHR(39) || ', ' || CHR(39) ||
v_time || CHR(39) || ', ' || CHR(39) || DNI_cli || CHR(39) || ', ' || CHR(39) || NombreE || CHR(39) || ', ' || CHR(39) || tipo ||
CHR(39) || ', ' || CHR(39) || 0.2*cantidad || CHR(39) || '); end;';
  pone_notificacion (sql_stm); -- execute immediate sql_stm
end if;
end loop;
end;
```

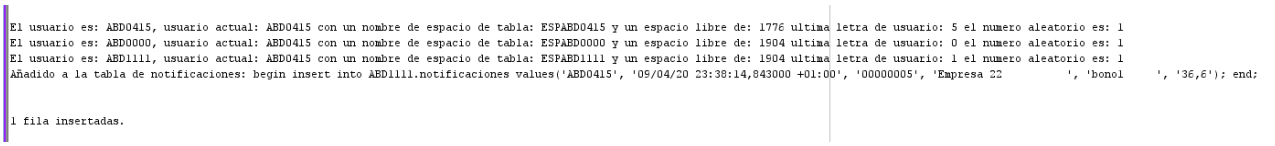
--para probar

insert into invierte values('00000005', 'Empresa 22',183,'bono2');

Ejemplo al añadir y no coincidir el número aleatorio:



Ejemplo al añadir y si coincidir el número aleatorio:



b) ¿Cómo funcionan los Checkpoints, Rollback y Pragma autonomous Transactions?

apartado f)

Trans. Principal Empieza: 3.27.902

se ha dormido -> antes: 1 despues: 1 Num.Trans.Secun: 1.31.910

se ha dormido -> antes: 0 despues: 0 Num.Trans.Secun: 4.23.910

se ha dormido -> antes: 1 despues: 1 Num.Trans.Secun: 9.0.921

se ha dormido -> antes: 0 despues: 0 Num.Trans.Secun: 4.15.911

se ha dormido -> antes: 1 despues: 1 Num.Trans.Secun: 4.5.911

Trans. Principal TERMINA: 3.27.902

-- pone notificacion

```
create or replace PROCEDURE pone_notificacion (sql_stm varchar2) as
suma_total float;
 PRAGMA AUTONOMOUS_TRANSACTION;
begin
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
   NAME 'Trans-Principal';
EXECUTE IMMEDIATE sql_stm;
select SUM(comision) into suma_total
from notificaciones;
DBMS_OUTPUT.put_line(suma_total);
if suma_total > 100 then
   DBMS_OUTPUT.put_line('se ha rechazado la operacón');
            rollback;
else
   DBMS_OUTPUT.put_line('se ha finalizado correctamente');
            commit;
end if;
end;
```

Si intentamos añadir: insert into invierte values('00000005', 'Empresa 22',183,'bono2');
nos dará error porque en la tabla de ABD1111 de notificaciones tienes un suma total de comisiones de 73,6 si añadimos una cantidad de 183 -> comisión = 0,2*183 = 36,6 la suma total ascendería a 110,2 sobrepasando los 100 euros por lo que hacemos rollback de la insercción.
Resultado de la ejecución:

```
El usuario es: ABD0000, usuario actual: ABD0415 con un nombre de espacio de tabla: ESPABD0000 y un espacio libre de: 1904 ultima letra de usuario: 0 el numero aleatorio es: 1
El usuario es: ABD1111, usuario actual: ABD0415 con un nombre de espacio de tabla: ESPABD1111 y un espacio libre de: 1904 ultima letra de usuario: 1 el numero aleatorio es: 1
begin insert into ABD1111.notificaciones values('ABD0415', '10/04/20 12:26:00,192000 +01:00', '00000005', 'Empresa 22          ', 'bono2      ', '36,6'); end;
110,2
se ha rechazado la operacón


1 fila insertadas.
```

	USUARIO_ORIGEN	FECHA	DNI_CLI	NOMBREE	TIPO	COMISION
1	ABD1111	07/04/20 21:40:04,018000 +01:00	12312312	sdas	sdasdqw	0,4
2	ABD0415	09/04/20 23:36:37,727000 +01:00	00000005	Empresa 22	bono1	36,6
3	ABD0415	09/04/20 23:38:14,843000 +01:00	00000005	Empresa 22	bono1	36,6

En cambio sí bajamos la cantidad: insert into invierte values('00000005', 'Empresa 22',4,'bono1');
No habrá ningún problema y el resultado será:

	USUARIO_ORIGEN	FECHA	DNI_CLI	NOMBREE	TIPO	COMISION
1	ABD0415	10/04/20 12:29:20,760000 +01:00	00000005	Empresa 22	bono1	0,8
2	ABD1111	07/04/20 21:40:04,018000 +01:00	12312312	sdas	sdasdqw	0,4
3	ABD0415	09/04/20 23:36:37,727000 +01:00	00000005	Empresa 22	bono1	36,6
4	ABD0415	09/04/20 23:38:14,843000 +01:00	00000005	Empresa 22	bono1	36,6

c)¿Cómo funcionan los Rollback?

-- añadimos suma_total

```
create or replace PROCEDURE pone_notificacion(DNI_cli char, NombreE char,tipo char, cantidad float) as
CURSOR existeTabla is select OWNER from ALL_TABLES where OWNER LIKE '%ABD%' and TABLE_NAME like '%NOTIFICACIONES%';
usuario char(7);
usuario_actual char(7);
tableSpaceNombre varchar2(30);
espacioLibre int;
numeroAleatorio int;
suma_Final float;
ultimaLetra char(1);
v_time TIMESTAMP WITH TIME ZONE;
sql_stm VARCHAR2(200);
cont int;
 PRAGMA AUTONOMOUS_TRANSACTION;
begin
SET TRANSACTION ISOLATION LEVEL READ COMMITTED NAME 'Trans-Principal';
suma_Final := 0;
cont := 0;
loop
DBMS_OUTPUT.put_line('este es el bucle: ' || cont);
 if cont = 3 then exit;
 end if;
open existeTabla;
loop
```

```
FETCH existeTabla INTO usuario;
  EXIT WHEN existeTabla%NOTFOUND;
  select default_tablespace into tableSpaceNombre from DBA_USERS where username like usuario;
SELECT SUM(BLOCKS) bloques_libres into espacioLibre FROM SYS.DBA_FREE_SPACE WHERE TABLESPACE_NAME LIKE
tableSpaceNombre GROUP BY TABLESPACE_NAME, FILE_ID order by bloques_libres ;
select MOD(ABS(DBMS_RANDOM.RANDOM),10)+1 into numeroAleatorio from dual;
SELECT SUBSTR(usuario,7) into ultimaLetra FROM DUAL;
select USER into usuario_actual from dual;
dbms_output.put_line('El usuario es: ' || usuario || ', usuario actual: ' || usuario_actual || ' con un nombre de espacio de tabla: ' ||
tablespacenombre || ' y un espacio libre de: ' || espaciolibre || ' ultima letra de usuario: ' || ultimaLetra || ' el numero aleatorio es:
' || numeroaleatorio);
if numeroAleatorio = ultimaLetra and espacioLibre < 1800 then
    SELECT SYSTIMESTAMP INTO v_time from dual;
    sql_stm := 'begin insert into ' || usuario || '.notificaciones values(' || CHR(39) || usuario_actual || CHR(39) || ', ' || CHR(39) ||
v_time || CHR(39) || ', ' || CHR(39) || DNI_cli || CHR(39) || ', ' || CHR(39) || NombreE || CHR(39) || ', ' || CHR(39) || tipo ||
CHR(39) || ', ' || CHR(39) || 0.2*cantidad || CHR(39) || '); end;';
   EXECUTE IMMEDIATE sql_stm;
   DBMS_OUTPUT.put_line(sql_stm);
   suma_Final := 0.2*cantidad;
end if;
end loop;
close existeTabla;
if suma_Final > 1000 then
    cont := cont + 1;
    suma_Final := 0;
    rollback;
else
    cont := 3;
    commit;
end if;
end loop;
end;
create or replace NONEDITIONABLE procedure RegalaComisiones(DNI_cli char, NombreE char,tipo char, cantidad float) AS
begin
 pone_notificacion(DNI_cli, NombreE, tipo, cantidad);
end;
```

d)¿Cómo funcionan los bloqueos explícitos de tabla?

-- añadimos lock table notificaciones al usuario en cuestión

```
create or replace PROCEDURE pone_notificacion(DNI_cli char, NombreE char,tipo char, cantidad float) as
CURSOR existeTabla is select OWNER from ALL_TABLES where OWNER LIKE '%ABD%' and TABLE_NAME like '%NOTIFICACIONES%';
usuario char(7);
usuario_actual char(7);
tableSpaceNombre varchar2(30);
espacioLibre int;
numeroAleatorio int;
s varchar2(100);
suma_Final float;
ultimaLetra char(1);
v_time TIMESTAMP WITH TIME ZONE;
sql_stm VARCHAR2(200);
cont int;
 PRAGMA AUTONOMOUS_TRANSACTION;
begin
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
    NAME 'Trans-Principal';
```

```
suma_Final := 0;
cont := 0;
loop
if cont = 3 then exit;
 end if;
open existeTabla;
loop
FETCH existeTabla INTO usuario;
  EXIT WHEN existeTabla%NOTFOUND;
  s := 'begin lock table ' || usuario || '.notificaciones in EXCLUSIVE MODE; end;';
  EXECUTE IMMEDIATE s;
  select default_tablespace into tableSpaceNombre from DBA_USERS where username like usuario;
SELECT SUM(BLOCKS) bloques_libres into espacioLibre FROM SYS.DBA_FREE_SPACE WHERE TABLESPACE_NAME LIKE
tableSpaceNombre GROUP BY TABLESPACE_NAME, FILE_ID order by bloques_libres ;
select MOD(ABS(DBMS_RANDOM.RANDOM),10)+1 into numeroAleatorio from dual;
SELECT SUBSTR(usuario,7) into ultimaLetra FROM DUAL;
select USER into usuario_actual from dual;
dbms_output.put_line('El usuario es: ' || usuario || ', usuario actual: ' || usuario_actual || ' con un nombre de espacio de tabla: ' ||
tablespacenombre || ' y un espacio libre de: ' || espaciolibre || ' ultima letra de usuario: ' || ultimaLetra || ' el numero aleatorio es:
' || numeroaleatorio);
if numeroAleatorio = ultimaLetra and espacioLibre < 1800 then
   SELECT SYSTIMESTAMP INTO v_time from dual;
   sql_stm := 'begin insert into ' || usuario || '.notificaciones values(' || CHR(39) || usuario_actual || CHR(39) || ', ' || CHR(39) ||
v_time || CHR(39) || ', ' || CHR(39) || DNI_cli || CHR(39) || ', ' || CHR(39) || NombreE || CHR(39) || ', ' || CHR(39) || tipo ||
CHR(39) || ', ' || CHR(39) || 0.2*cantidad || CHR(39) || '); end;';
   EXECUTE IMMEDIATE sql_stm;
   DBMS_OUTPUT.put_line(sql_stm);
   suma_Final := 0.2*cantidad;
end if;
end loop;
close existeTabla;
if suma_Final > 1000 then
   cont := cont + 1;
   suma_Final := 0;
   DBMS_OUTPUT.put_line('rollback de todo');
   rollback;
else
   cont := 3;
   DBMS_OUTPUT.put_line('todo perfecto');
   commit;
end if;
end loop;
end;
```

Ejemplo de ejecución, si intentamos añadir en la tabla notificaciones de ABD1111 mientras está siendo bloqueda se quedará esperando hasta que deje de estarlo, ejemplo de como se queda esperando:

e.- ¿Cuándo terminan los bloqueos de tabla?

Usando lock table notificaciones share mode, así permitiremos a varios usuarios leer la tabla a la vez, en cambio el otro método sólo nos permitía leerlo por un usuario y para los demás la tabla se bloqueaba.

f.- Para entender: ¿cómo funcionan las transacciones autónomas?

Trans. Principal Empieza: 3.27.902

se ha dormido -> antes: 1 despues: 1 Num.Trans.Secun: 1.31.910

se ha dormido -> antes: 0 despues: 0 Num.Trans.Secun: 4.23.910

se ha dormido -> antes: 1 despues: 1 Num.Trans.Secun: 9.0.921

se ha dormido -> antes: 0 despues: 0 Num.Trans.Secun: 4.15.911

se ha dormido -> antes: 1 despues: 1 Num.Trans.Secun: 4.5.911

Trans. Principal TERMINA: 3.27.902

g) ¿ Cómo funcionan las transacciones autónomas?

cogemos el procedimiento del principio

g.1)

```
create or replace NONEDITIONABLE trigger prac36 after insert on invierte
for each row
DECLARE
  PRAGMA AUTONOMOUS_TRANSACTION;
begin
        RegalaComisiones(:NEW.DNI,:NEW.NOMBREE,:NEW.TIPO, :NEW.CANTIDAD);
   commit;
end;
```

g.2)

podríamos poner que el tr_anterior se ejecute antes de la inserción, es decir before insert on invierte.

g.3)

se mantiene lo que haya en Prueba_Autonoma ya que es una transacción independiente a las demás, por lo tanto se puede realizar commits dentro de esta.

h)Qué características de una instalación con varias transacciones hacen que una transacción deba ser serializable

Grandes BDs con T's cortas y actualizan pocas filas.

Protege mejor de lecturas fantasma e irrepetibles cuando una T lee dos veces la misma fila.

i)Qué características de una instalación con varias transacciones hacen hacen que una transacción deba ser read commited

Respuesta más rápida pero más insegura

j)¿Qué nivel de aislamiento u otro recurso aplicarías para estas situaciones?:

j.1.- Unas transacciones actualizan la tabla ValoresBolsa. Qué nivel de aislamiento se necesita en otra transacción para que vea cuanto antes los últimos cambios? ¿Harías algo en esas transacciones para que se vean los cambios cuanto antes? . Explica el porqué.

read commited porque tiene una respuesta más rápida.

j.2.- Unas transacciones actualizan la tabla ControlPuerta, que puede estar abierta, cerrada o bloqueada. ¿Qué harías para que cada transacción vea el estado inmediatamente después de la actualización de la otra transacción? Explica el porqué.

read commited porque se actualiza una vez actualizado, en cambio serializable lo bloquearía para que no haya lectura repetible.

j.3.- Unas transacciones actualizan la tabla MovtosVisa. Nuestra transacción hace el cierre fin de mes a las doce de la noche del último día, contabiliza y marca cada movimiento como procesado. Y al final suma el total de los movimientos marcados. Qué nivel de aislamiento necesitamos? Explica el porqué.

Serializable ya que puede ser que otra transacción nos borre alguno de los marcados y estemos operando con valores incorrectos.

Serializable no permite que esto pase.

j.4.- Unas transacciones actualizan la tabla MovtosVisa. Nuestra transacción solo lista los movimientos entre dos fechas. Qué tipo de transacción definiremos para agilizar el proceso? Indica la instrucción.

read commited para que pueda a ver más de una en funcionamiento.

k.- ¿Qué resultado dará cada situación descrita?:

k.1.- En este trozo de procedimiento: ¿qué sucede con el segundo rollback?

Savepoint SP1;

Proc-update1();

Savepoint SP2;

Proc-update2();

IF fallo1 then rollback savepoint SP1 end if;

IF fallo2 then rollback savepoint SP2 end if;

si falla el primero el segundo no se produciría pq no tendría fallo ya que no existiría el update2, porque se ha hecho rollaback

si no falla el primero se haría simplemente un update de 1 y el dos se eliminaria

k.2.- En este trozo de procedimiento: ¿qué sucede con el segundo rollback?

Savepoint SP1;

Update . . . .

Rollback;

Update. . . . .

Rollback to savepoint SP1;

Da error, porque el primer rollback elimina el savepoint creado y por lo tanto el segundo está haciendo rollaback a un savepoint que no existe.