**Cuestiones:**

1. **Crear un trigger llamado ‘presenta\_InsertDeleteUpdateOperacion’, donde cada vez que se realiza una operación de inserción (‘INSERT’), un borrado (‘DELETE’) o una actualización (‘UPDATE’) se presente un mensaje. (REALIZAR POR LOS ESTUDIANTES)**

*-- Crear función similar a la del apartado anterior que tenga en cuenta el resto de operaciones.*

create or replace function presenta\_InsertDeleteUpdateOperacion() returns trigger as

$$

begin

if tg\_op = 'INSERT' then

raise notice 'Se ha realizado una inserción.';

return new;

elsif tg\_op = 'DELETE' then

raise notice 'Se ha realizado una eliminación.';

return old;

elsif tg\_op = 'UPDATE' then

raise notice 'Se ha realizado una modificación.';

return new;

end if;

return null;

end;

$$ language plpgsql;

*-- separador --*

create or replace trigger tg\_presenta\_multi

after insert or update or delete on estudiante\_grado\_modulo

for each row execute procedure presenta\_InsertDeleteUpdateOperacion();

1. **Añadir dos columnas a la table ‘estudiante\_grado\_modulo’ llamadas ‘nota’ (int) y ‘grado’ (varchar(10)), con un valor por defecto de 0 e ‘indefinido’ respectivamente. (REALIZAR POR LOS ESTUDIANTES)**

*-- Añadir dos columnas a la table ‘estudiante\_grado\_modulo’ llamdas ‘nota’ (int) y*

*-- ‘grado’ (varchar(10)), con un valor por defecto de 0 e ‘indefinido’ respectivamente.*

alter table estudiante\_grado\_modulo add column nota int default 0;

alter table estudiante\_grado\_modulo add column grado varchar(10) default 'indefinido';

1. **Cada vez que se realiza una inserción, actualización o borrado en la tabla ‘estudiante\_grado\_modulo’ se necesita registrar que dicha operación ha sido ejecutada. Para tal fin, se creará una tabla denominada ‘operacion\_notas\_log’ que deberá contener los siguientes campos: (REALIZAR POR LOS ESTUDIANTES)**

|  |  |
| --- | --- |
| **Tabla:**  **operacion\_notas\_log** | |
| **Atributo** | **Tipo** |
| **operacion** | **char(1)** |
| **hora** | **timestamp** |
| **estudiantetid** | **int** |
| **moduloid** | **int** |
| **nota** | **int** |

drop table operacion\_notas\_log;

create table operacion\_notas\_log(

operacion char(1),

hora timestamp,

estudianteid int,

moduloid int,

nota int

);

1. **Crea un trigger para que cada vez que se actualice una nota en la tabla ‘estudiante\_grado\_modulo’ se añada un registro en la tabla ‘operacion\_notas\_log’ con la siguiente información: (‘U’, hora actual, estudiante id, modulo id, nota).** 
   1. **Realizar otras modificaciones y comprueba los cambios en la tabla ‘operacion\_notas\_log’. (REALIZAR POR LOS ESTUDIANTES)**

Después de hacer algunos cambios aleatorios a la base de datos ‘estudiante\_grado\_modulo’, todas las operaciones quedan guardadas correctamente en la nueva tabla:

Escala de tiempo

Descripción generada automáticamente

**7. Crear un trigger similar al anterior pero que también registre las operaciones de inserción (‘I’) y borrado (‘D’). Usa la estructura del condicional anidada. Escribe al menos un ejemplo para comprobar el funcionamiento de cada uno de los dos tipos de operaciones añadidas. (REALIZAR POR LOS ESTUDIANTES)**

*-- Crear un trigger similar al anterior que atienda a todas las modificaciones en*

*-- en la tabla ‘estudiante\_grado\_modulo’.*

create or replace function operacion\_log\_multi() returns trigger as

$$

begin

if tg\_op = 'UPDATE' then

insert into operacion\_notas\_log(operacion, hora, estudianteid, moduloid,

nota) values ('U', now(), new.estudiante\_id, new.modulo\_id, new.nota);

return new;

elsif tg\_op = 'DELETE' then

insert into operacion\_notas\_log(operacion, hora, estudianteid, moduloid,

nota) values ('D', now(), old.estudiante\_id, old.modulo\_id, old.nota);

return old;

elsif tg\_op = 'INSERT' then

insert into operacion\_notas\_log(operacion, hora, estudianteid, moduloid,

nota) values ('I', now(), new.estudiante\_id, new.modulo\_id, new.nota);

return new;

end if;

return null;

end;

$$ language plpgsql;

*-- separador --*

create or replace trigger tg\_operacion\_log\_multi

after update or delete or insert on estudiante\_grado\_modulo

for each row execute procedure operacion\_log\_multi();

****

1. **Borrar la tabla ‘operacion\_notas\_log’.** **También será necesario borrar el trigger tg\_operacion\_log que insertaba en la tabla borrada. (REALIZAR POR LOS ESTUDIANTES)**

*-- Borrar la tabla 'operacion\_notas\_log' y el trigger 'tg\_operacion\_log\_multi':*

drop table operacion\_notas\_log;

drop trigger tg\_operacion\_log\_multi on estudiante\_grado\_modulo;

1. **Crear una tabla denominada ‘operacion\_calificacion\_log’ similar a la del apartado 4, pero añadiendo el atributo ‘calificacion’. (REALIZAR POR LOS ESTUDIANTES)**

|  |  |
| --- | --- |
| **Tabla:**  **operacion\_calificacion\_log** | |
| **Atributo** | **Tipo** |
| **operacion** | **char(1)** |
| **stamp** | **timestamp** |
| **estudiantetid** | **int** |
| **moduloid** | **int** |
| **nota** | **int** |
| **calificacion** | **varchar(10)** |

*-- Crear una tabla denominada ‘operacion\_calificacion\_log’ similar a la del apartado 4,*

*-- pero añadiendo el atributo ‘calificacion’.*

create table operacion\_calificacion\_log(

operacion char(1),

hora timestamp,

estudianteid int,

moduloid int,

calificacion int

);

1. **Crear un trigger denominado ‘calificacion\_log’ similar al del punto 5, pero antes de que la nota sea insertada o actualizada en la tabla ‘estudiante\_grado\_modulo’ se calculará una calificación no numérica de acuerdo al baremo indicado en la siguiente tabla:**

|  |  |
| --- | --- |
| **nota** | **calificación** |
| **< 4** | **‘Pobre’** |
| **4 <= x < 5** | **‘No buena’** |
| **5 <= x < 7** | **‘Buena’** |
| **7 <= x < 9** | **‘Muy buena’** |
| **9 <= x < = 10** | **‘Excelente’** |

*-- Crear un trigger similar al del ejercicio 5, pero antes de que la nota sea insertada o actualizada,*

*-- se deberá calcular la calificación no numérica de acuerdo al siguiente baremo:*

*-- <4: Pobre, 4-5: No buena, 5-7: Buena, 7-9: Muy buena, 10: Excelente.*

create or replace function calificacion\_log\_connota() returns trigger as

$$

begin

if tg\_op = 'UPDATE' then

if new.nota < 4 then

insert into operacion\_calificacion\_log(operacion, hora, estudianteid, moduloid,

calificacion) values ('U', now(), new.estudiante\_id, new.modulo\_id, 'Pobre');

elsif new.nota >= 4 and new.nota < 5 then

insert into operacion\_calificacion\_log(operacion, hora, estudianteid, moduloid,

calificacion) values ('U', now(), new.estudiante\_id, new.modulo\_id, 'No buena');

elsif new.nota >= 5 and new.nota < 7 then

insert into operacion\_calificacion\_log(operacion, hora, estudianteid, moduloid,

calificacion) values ('U', now(), new.estudiante\_id, new.modulo\_id, 'Buena');

elsif new.nota >= 7 and new.nota < 9 then

insert into operacion\_calificacion\_log(operacion, hora, estudianteid, moduloid,

calificacion) values ('U', now(), new.estudiante\_id, new.modulo\_id, 'Muy buena');

elsif new.nota >= 9 and new.nota <= 10 then

insert into operacion\_calificacion\_log(operacion, hora, estudianteid, moduloid,

calificacion) values ('U', now(), new.estudiante\_id, new.modulo\_id, 'Excelente');

end if;

return new;

end if;

return null;

end;

$$ language plpgsql;

*-- separador --*

create or replace trigger tg\_calificacion\_log\_connota

before update on estudiante\_grado\_modulo

for each row execute procedure calificacion\_log\_connota();