Rodríguez López, Alejandro. UO281827

**Cuestiones:**

1. **Crear un trigger que cada vez que se inserte una fila en la tabla**

**‘estudiante\_grado\_modulo’ se presente el mensaje ‘Se ha realizado una inserción’.**

create or replace function presenta\_insercion() returns trigger as $$

begin

if tg\_op = 'INSERT' then

raise notice 'Se ha realizado una insercion';

return new;

end if;

return null;

end;

$$ language plpgsql;

create trigger tg\_presenta\_insercion after

insert on estudiante\_grado\_modulo

for each row execute procedure presenta\_insercion();

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)**

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

begin

if tg\_op = 'INSERT' then

raise notice 'Se ha realizado una insercion';

return new;

elsif tg\_op = 'DELETE' then

raise notice 'Se ha realizado un borrado';

return new;

elsif tg\_op = 'UPDATE' then

raise notice 'Se ha realizado una actualizacion';

return new;

end if;

return null;

end;

$$ language plpgsql;

create trigger tg\_presenta\_InsertDeleteUpdateOperacion after

insert or delete or update

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)**

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** |

CREATE TABLE operacion\_notas\_log (

operacion char(1) not NULL,

hora timestamp not null,

estudianteId int not null,

moduloId int not null,

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).**

create or replace function operacion\_log() 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;

end if;

return null;

end;

$$ language plpgsql;

create trigger tg\_operacion\_log after

update on estudiante\_grado\_modulo

for each row execute procedure operacion\_log();

* 1. **Realizar otras modificaciones y comprueba los cambios en la tabla ‘operacion\_notas\_log’. (REALIZAR POR LOS ESTUDIANTES)**

update estudiante\_grado\_modulo

set nota = 5

where estudiante\_id=1 and modulo\_id=4;

INSERT INTO estudiante\_grado\_modulo

VALUES (2, 4, 7);

UPDATE estudiante\_grado\_modulo

SET nota=6

WHERE estudiante\_id = 2 AND modulo\_id = 4;

1. **Eliminar operacion\_notas\_log y su trigger.**

DROP TABLE operacion\_notas\_log;

DROP TRIGGER tg\_operacion\_log ON estudiante\_grado\_modulo;

1. **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)**

create or replace function operacion\_log() 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 = 'INSERT' THEN

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

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

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;

end if;

return null;

end;

$$ language plpgsql;

create trigger tg\_operacion\_log after

update or insert or delete

on estudiante\_grado\_modulo

for each row execute procedure operacion\_log();

INSERT INTO estudiante\_grado\_modulo

VALUES (1, 1, 1);

UPDATE estudiante\_grado\_modulo

SET nota = 3

WHERE estudiante\_id=1 AND modulo\_id=1 AND grado\_id=1;

DELETE

FROM estudiante\_grado\_modulo

WHERE estudiante\_id=1 AND grado\_id=1 AND modulo\_id=1;

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)**

DROP TABLE operacion\_notas\_log;

DROP TRIGGER tg\_operacion\_log 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)** |

**CREATE TABLE operacion\_calificacion\_log(**

**operacion char(1) not NULL,**

**hora timestamp not null,**

**estudianteId int not null,**

**moduloId int not null,**

**nota int, calificacion varchar(10));**

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’** |

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

declare

calif varchar(10);

begin

-- calif

if new.nota < 4 THEN

calif = 'Pobre';

elsif new.nota < 5 THEN

calif = 'No buena';

elsif new.nota < 7 THEN

calif = 'Buena';

elsif new.nota < 9 THEN

calif = 'Muy buena';

else

calif = 'Excelente';

end if;

-- insert

if tg\_op = 'UPDATE' then

insert into operacion\_calificacion\_log values ('U', now(), new.estudiante\_id, new.modulo\_id, new.nota, calif);

return new;

elsif tg\_op = 'INSERT' THEN

insert into operacion\_calificacion\_log values ('I', now(), new.estudiante\_id, new.modulo\_id, new.nota, calif);

elsif tg\_op = 'DELETE' THEN

insert into operacion\_calificacion\_log values ('D', now(), old.estudiante\_id, old.modulo\_id, old.nota, calif);

return old;

end if;

return null;

end;

$$ language plpgsql;

create trigger tg\_calificacion\_log after

update

or

insert

or

delete on estudiante\_grado\_modulo

for each row execute procedure calificacion\_log();