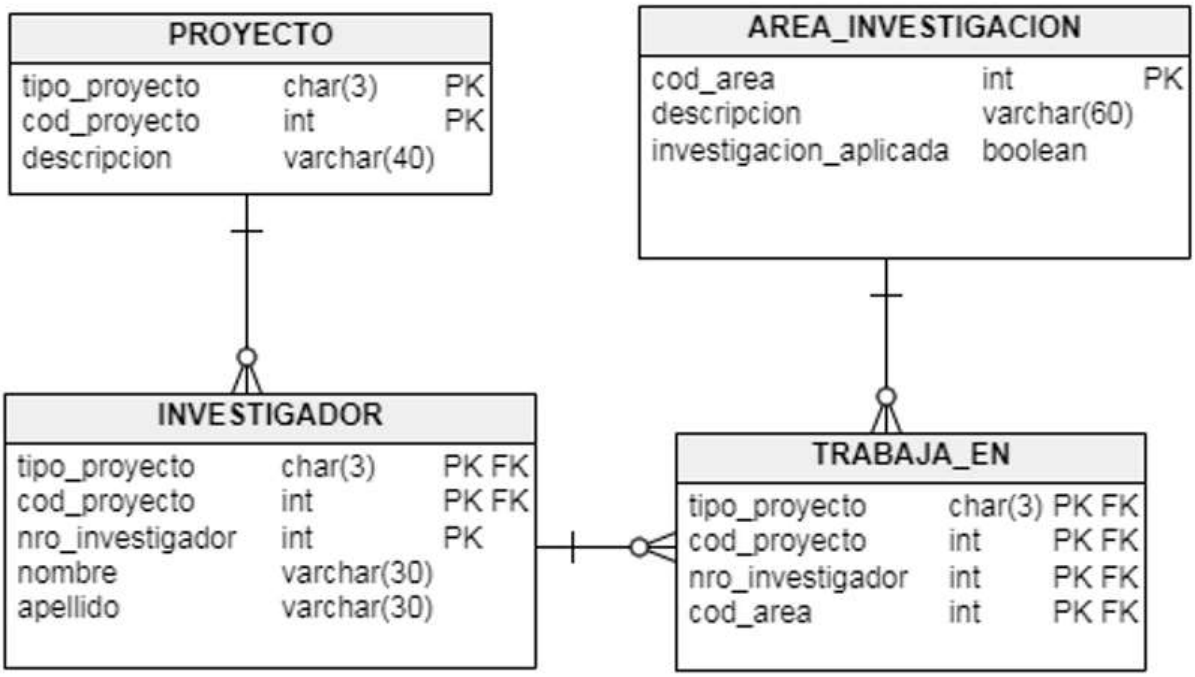
Comenzado el	lunes, 3 de julio de 2023, 18:02
Estado	Finalizado
Finalizado en	lunes, 3 de julio de 2023, 19:42
Tiempo empleado	1 hora 40 minutos

Pregunta 1

Finalizado

Puntúa como 1,00

Utilice el recurso más conveniente para retornar todos los datos de las áreas de investigación que no posean investigadores de un proyecto (campo descripción de proyecto) dada por parámetro.



```
CREATE OR REPLACE FUNCTION obtener_areas_investigacion_sin_investigadores(descripcion_proyecto
VARCHAR)
  RETURNS TABLE (cod_area INT, descripcion VARCHAR, investigacion_aplicada BOOLEAN)
AS $$
BEGIN
  RETURN QUERY
    SELECT a.cod_area, a.descripcion, a.investigacion_aplicada
    FROM area_investigacion a
    WHERE NOT EXISTS (
      SELECT 1
      FROM trabaja_en t
      INNER JOIN proyecto p ON t.tipo_proyecto = p.tipo_proyecto AND t.cod_proyecto = p.cod_proyecto
      INNER JOIN investigador i ON t.tipo_proyecto = i.tipo_proyecto AND t.cod_proyecto = i.cod_proyecto AND
t.nro_investigador = i.nro_investigador
      WHERE p.descripcion = descripcion_proyecto
        AND t.cod_area = a.cod_area
    );
END;
$$ LANGUAGE plpgsql;
```

```
-- Soluciones incorrectas:
    --  Que no implementen correctamente la estructura de la funcion
    --  Que lo relsuevan con cursores.
    --  Que realizan mal los ensambles entre tablas (PK-FK)
    --  Que proyecten las columnas requeridas
    --  Que NO resuelvan correctamente la consulta requerida
    --  Que NO utilicen la descripcion de la tabla PROYECTO

CREATE OR REPLACE FUNCTION fn_area_investigacion(proy varchar(40))
RETURNS TABLE (cod_area int, descripcion varchar(60), investigacion_aplicada boolean) AS
$$
BEGIN
 RETURN QUERY
                SELECT a.cod_area, a.descripcion, a.investigacion_aplicada
                FROM AREA_INVESTIGACION a
                WHERE NOT EXISTS (
                        SELECT 1
                        FROM TRABAJA_EN t
                             JOIN INVESTIGADOR i ON (
                                    t.cod_proyecto =  i.cod_proyecto AND
                                    t.tipo_proyecto =  i.tipo_proyecto AND
                                    t.nro_investigador =  i.nro_investigador
                             )
                             JOIN PROYECTO p ON (
                                    p.cod_proyecto =  i.cod_proyecto AND
                                    p.tipo_proyecto =  i.tipo_proyecto
                            )
                        WHERE   a.cod_area = t.cod_area
                        AND     p.descripcion = proy
                    );
END
$$
language 'plpgsql';
```

Información

Debe resolver el siguiente ejercicio teniendo en cuenta lo siguiente:

- Que debe reemplazar en TODO el script donde dice **unc_xxxxx** por su usuario (**unc_123456** por ejemplo). Consejo: baje el script, ábralo con un editor de texto (notepad++ o equivalente) y haga un reemplazar todo de **unc_xxxxx** por **unc_123456** (siendo **123456** su número de legajo)
- Después de hacer el reemplazo, debe ejecutar el script completo (no debe dar error)
- La creación de las tablas y sus datos DEBE formar parte de su solución (debe entregarlo, al principio de su entrega SIN haber modificado el orden de creación de las tablas, orden de creación de las tablasclaves extranjeras, NI orden de creación de las tablas los datos como así los datos en si, sólo debe reemplazar lo que se pide)
- Que la cátedra realizará un Copy & Paste del texto de su respuesta y si Ud. deja una aclaración que no esté como comentario de SQL dentro del texto de la misma, ésto dará error.
- Que los objetos que Ud. declare se denominen en el caso de los trigger **tg_[leg]_[nombre]**, las funciones **fn_[leg]_[nombre]** y los procedimientos **pr_[leg]_[nombre]**;
  - por ejemplo si hiciera una función, un posible nombre sería **fn_123456_manejo_punto_1**
  - **[leg]** es su número de legajo
  - **[nombre]** es el nombre que Ud. crea conveniente para el trigger o función.
- Que el código que Ud. entregue NO debe hacer referencia a un esquema determinado, ni tampoco en la declaración de los objetos (triggers o funciones) suponga que al principio de su código hay un set search_path = ...
- NO debe tener un search_path.

Criterios por los cuales el ejercicio no será evaluado (y por lo tanto tendrá 0 punto):

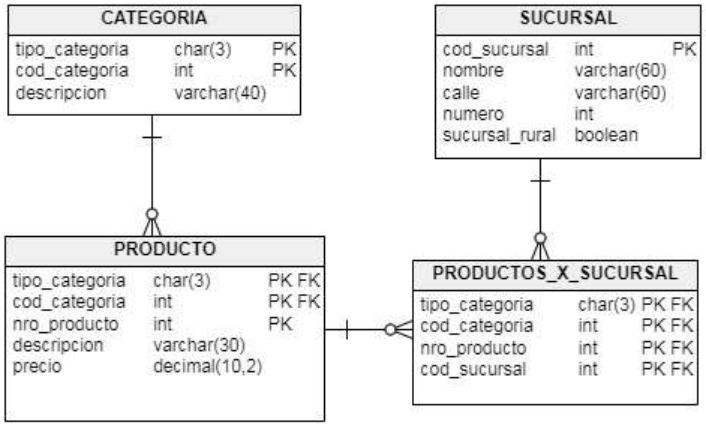
- Modifica el orden de creación y/o contenido de las tablas/datos
- Si el trigger o la función no compila (no importa qué tipo de error sea ni donde de el error)
- Si llegara a dar error en la ejecución (no compilación)
- Si su solución llega a ser muy ineficiente, tener en cuenta por ejemplo NO realizar triggers por ROW y Funciones por Statement
- Si hace un mal uso de las FK y/o PK

Pregunta 2

Finalizado

Puntúa como 1,00

Considerando el siguiente esquema - (recuerde que las tablas serán unc_xxxxxx_**tabla** siendo tabla en este caso, CATEGORIA, PRODUCTO, etc.)



a) Se necesita controlar que un Producto (sin importar su Categoria) no pueda estar en mas de 3 sucursales.

b) Se necesita controlar que si el nombre de la sucursal contiene la frase "***rural***" (sin importar si está en mayúsculas o mínusculas) el atributo **sucursal_rural** debe ser **true**.

Las tablas y datos se encuentran con éste link (recuerde que todo lo que Ud. dé como respuesta será ejecutado, realice lo pedido en el orden adecuado)

1. Modifique y ejecute el script dado (creación de tablas y datos, reemplazar unc_xxxxx, etc.)
2. De la sentencia declarativa que controlaría cada requerimiento dado (Si la sentencia declarativa NO puede ser implementada declarativamente en Postgres, colóquela en la siguiente pregunta)
3. Si la restricción declarativa no es soportada por el DBMS, provea el/los trigger/s junto con sus funcion/es (en el orden correcto para que no de error al compilar) utilizando los parámetros adecuados para que se comporten lo más eficientemente para que éstos este requerimiento sea controlado

```
A)
CREATE OR REPLACE FUNCTION check_product_branch_limit()
RETURNS TRIGGER AS $$
DECLARE
    product_count INTEGER;
BEGIN
    SELECT COUNT(*)
    INTO product_count
    FROM unc_250791_PRODUCTOS_X_SUCURSAL
    WHERE nro_producto = NEW.nro_producto;

    IF product_count >= 3 THEN
        RAISE EXCEPTION 'Un producto no puede estar en más de 3 sucursales';
    END IF;

    RETURN NEW;
END;
$$
 LANGUAGE plpgsql;

-----------------------------------------

CREATE TRIGGER trg_product_branch_limit
BEFORE INSERT ON unc_250791_PRODUCTOS_X_SUCURSAL
FOR EACH ROW
EXECUTE FUNCTION check_product_branch_limit()


-----------------------------------
PARA CORROBORAR QUE TODO FUNCIONE
INSERT INTO unc_250791_PRODUCTOS_X_SUCURSAL (tipo_categoria, cod_categoria, nro_producto, c
VALUES ('K0W', 49, 9, 1);
INSERT INTO unc_250791_PRODUCTOS_X_SUCURSAL (tipo_categoria, cod_categoria, nro_producto, c
VALUES ('M5G', 92, 14, 10), ('M5G', 92, 14, 11), ('M5G', 92, 14, 12), ('M5G', 92, 14, 13);




////////////////////////////////////////////////////////////////////////////////
```

1) Debían realizar es reemplazar el xxxxxx por su número de libreta en script dado

2) Ese Script lo debian entregar, SIN NINGUN tipo de modificación, tal como se aclaraba

3) Luego de ese script debería estar su solución

-- a continuación se da una posible solución

-- Primero los Create table (no se coloca el código completo simplemente para claridad, pero queda claro que debería estar el script dado con el reemplazo pedido)

Create table ...

Create table ...

Create table ...

Create table ...

Alter table ....

Alter table ....

Alter table ...

insert into ....

insert into ....

insert into ....

insert into ....

-- El punto 2 aclara que tienen que dar la sentencia declarativa, salvo que ésta NO pudiera ser implementada en Postgres, si no podía ser, debería ir en le cuadro de texto siguiente (si la colocaba comentada, no habia problema)

-- Lógicamente se pide esto porque si NO puede ser implementada, al correr el script daría error, ya que justamente NO puede ser implementada

---

-- a) Se necesita controlar que un Producto (sin importar su categoria) no pueda atender en mas de 3 sucursales.

-- la restricción que controla esto es del tipo de TABLA ya que sólo necesito la tabla PRODUCTOS_X_SUCURSAL

-- al especificar "Sin importar la categoría" debería utilizar sólo el atributo nro_producto

-- una posible solución es la siguiente

-- a) Restricción declarativa es de Tabla, va en el siguiente cuadro porque si la colocan acá daría error

```
CREATE OR REPLACE FUNCTION fn_xxxxxx_a() RETURNS TRIGGER AS $$
BEGIN
IF ((SELECT count(distinct cod_sucursal)
      from unc_xxxxxx_PRODUCTOS_X_SUCURSAL where nro_producto = NEW.nro_producto) > 2) THEN
        RAISE EXCEPTION 'Un Producto no puede estar en mas de 3 sucursales';
END IF;
RETURN NEW;
END$$ LANGUAGE 'plpgsql';

CREATE OR REPLACE TRIGGER tg_xxxxxx_punto1
BEFORE INSERT OR UPDATE OF cod_sucursal, nro_producto
ON unc_xxxxxx_PRODUCTOS_X_SUCURSAL
FOR EACH ROW
EXECUTE PROCEDURE fn_xxxxxx_a();
```

-- b) Se necesita controlar que si el nombre de la sucursal contiene la frase "rural" (sin importar si está en mayúsculas o mínusculas) el atributo sucursal_rural debe ser true

-- Restricción de FILA, perfectamente implementable en Postgres (por ende debía estar en este script)

alter table unc_xxxxxx_sucursal add constraint chk_xxxxxx_b check (

(nombre ilike '%rural%' and sucursal_rural = true) or (nombre not ilike '%rural%'));

-- Se usó ILIKE que ya me ahorra el tema de tratar las mayúsculas y minísculas, pero si hacían un LOWER o UPPER y utilizaban el LIKE, era exactamente lo mismo.

-- Aclaraciones Generales sobre éste ejercicio

-- Si no contaban los Distintos, no invalidaba todo el ejercicio, pero demuestran que no saben lo que cuentan, restaba muchos puntos.

-- Si colocaban la restricción a) de tabla SIN comentar, invalidaba todo el ejercicio (ya que daba error al ejecutar)

-- Si utilizaban la CATEGORIA estaba mal, se pedía explicitamente que no lo hicieran

-- El Update NO tenía que ser por toda la FILA, sólo tenía que ser por los campos involucrados (en este caso cod_sucursal y nro_producto)

-- Si para la restricción b, hacían triggers, estaba mal, se aclaró muchas veces que SIEMPRE tienen que usar lo mas restrictivo

-- Si modificaban el script entregado (mas allá de lo que se pedía o si no estaba) esaba mal.

-- Si ensamblaban tablas en el a), estaba mal, no tiene sentido hacerlo, de hecho al hacerlo pasaría de ser una restricción de tabla a general.

-- Esta solución es con un trigger BEFORE, perfectamente podría ser AFTER y en vez de ser >2 ser >3, cualquiera de las dos era equivalente

-- Si el trigger era BEFORE, pero usaban >3 en vez de >2 o >=3 (que obviamente es lo mismo si son enteros) bajaba muchos puntos porque denota que NO sabe como funciona un trigger BEFORE

-- Si implementaban algún tipo de CONTADOR, estaba mal, en ningún lado se pide que lo hagan.

Comentario:

Criterios por los cuales el ejercicio no será evaluado (y por lo tanto tendrá 0 punto):

- Modifica el orden de creación y/o contenido de las tablas/datos

No están las tablas dadas para modificar

Pregunta 3

Finalizado

Puntúa como 1,00

En caso de Ud. consiere de que la o las sentencias declarativas NO pueden ser implementadas en Postgres, colóquelas acá aclarando a que inciso hace referencia

```
-- Acá sólo debería haber colocado la restricción DECLARATIVA del punto a)
-- Se colocan todas las soluciones ya que como funciona moodle no puedo discriminar que le tocó a cada uno.
alter table unc_xxxxxx_atiende add constraint chk_xxxxxx_a check (not exists (
   select 1 from unc_xxxxxx_atiende group by nro_matricula having count(distinct cod_centro) > 3 ) );

alter table unc_xxxxxx_trabaja_en add constraint chk_xxxxxx_a check (not exists (
   select 1 from unc_xxxxxx_trabaja_en group by nro_investigador having count(distinct cod_area) > 3 ) );

alter table unc_xxxxxx_productos_x_sucursal add constraint chk_xxxxxx_a check (not exists (
   select 1 from unc_xxxxxx_productos_x_sucursal group by nro_producto having count(distinct cod_sucursal) > 3 ) );
```

```
-- Esta restricción NO puede ser implementada en Postgres ya que es de Tabla
-- Si hacían un assertion estaba mal
-- Si hacían triggers, estaba mal
```

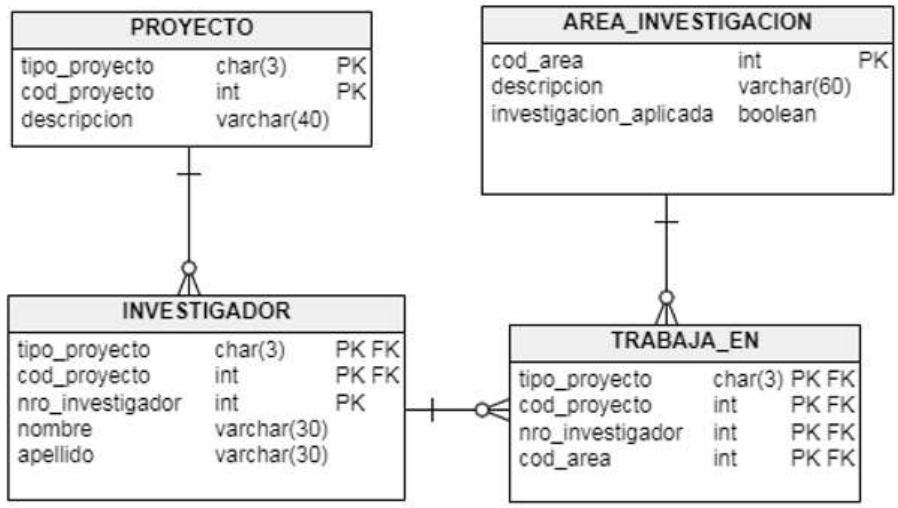
Comentario:
Debería estar la restricción del punto a)

Pregunta **4**

Finalizado

Puntúa como 1,00

Utilizando el siguiente esquema. Cree una vista **automáticamente actualizable en PostgreSQL, optimizada bajo reglas de equivalencia (nombre cuales de ellas) y que las tuplas no puedan migrarse** que seleccione el nombre y apellido de aquellos investigadores que trabajen en menos de 5 áreas de investigación aplicada y que además represente un proyecto con descripción ROBOTICS. (la vista debe comenzar con v_xxxxxx_nombre, siendo xxxxxx su número de legajo y nombre el nombre que Ud. le quiera dar)




```
la vista debe cumplir los siguientes requisitos para ser actualizable
La consulta definitoria de la vista debe tener exactamente una entrada en la cláusula FROM, q
La consulta de definición no debe contener una de las siguientes cláusulas en el nivel superi
La lista de selección no debe contener ninguna función de ventana, ninguna función de devoluc

CREATE OR REPLACE VIEW v_250791_areas AS
SELECT i.nombre, i.apellido
FROM investigador i
JOIN trabaja_en t ON i.tipo_proyecto = t.tipo_proyecto AND i.cod_proyecto = t.cod_proyecto AN
JOIN area_investigacion a ON t.cod_area = a.cod_area
JOIN proyecto p ON i.tipo_proyecto = p.tipo_proyecto AND i.cod_proyecto = p.cod_producto
WHERE a.investigacion_aplicada = true AND p.descripcion = 'ROBOTICS'
GROUP BY i.nro_investigador, i.nombre, i.apellido
HAVING COUNT(DISTINCT t.cod_area) < 5;
```

```
-- Análisis:
--   1) Se solicita: "cree una vista automáticamente actualizable en PostgreSQL.... donde se selecione el nombre y apellido ...."
--      Teniendo en cuenta los conceptos teóricos de "vista actulalizabe", NO es posible crear una vista actualizable debido a que no se preserva la clave (KEY PRESERVED).
--      Si se puede realizar migraciones de las tuplas mediante TRIGGERS INSTEAD OF.
--   2) Se solicita: "...las tuplas no puedan migrarse..."
--      No es necesario aplicar ninguna clausula o estrategia.
--   3) Se solicita: "...optimizada bajo reglas de equivalencia (nombre cuales de ellas)..."
--      La vista debe estar optimizada y nombrar cuales se aplica
--   4) Se solicita: "Utilizando el siguiente esquema. Cree una vista automáticamente actualizable en PostgreSQL, optimizada bajo reglas de equivalencia (nombre cuales de ellas) y que las tuplas no puedan migrarse que seleccione el nombre y apellido de aquellos investigadores que trabajen en menos de 5 áreas de investigación aplicada y que además represente un proyecto con descripción ROBOTICS"
--      La vista debe cumplir el requerimiento
--   5) Se solicita que la vista contenga un nombre especifico (v_xxxxxx_nombre) para poder ejecutarla.

-- Soluciones incorrectas:
    -- * Que NO comprendan el concepto de actualización de vistas.
    -- * Que la vista NO este optimizada bajo reglas de equivalencia.
    -- * Que NO se especificara cuales reglas de equivalencia se aplicaron
    -- * Que la vista NO cumpla la consulta como requerimiento. Posibles errores:
    --      * Que realizan mal los ensambles entre tablas (PK-FK)
    --      * Que la vista no proyecte las columnas requeridas
    --      * Que realizan agrupación por atributos incorrectos
    --      * Que NO utilicen el campo descripción de la tabla PROYECTO

-- Posible Solución aplicando las regla de equivalencia:
-- (1) enable por join,
-- (2) Filtar o seleccionar antes de ensamblar
-- (3) Proyectar solo los atributos necesarios antes de ensamblar
CREATE OR REPLACE VIEW Vista_xxxxxx_Proyecto_Robotics
AS
    SELECT i.nombre, i.apellido
    FROM INVESTIGADOR i
    WHERE (i.tipo_proyecto, i.cod_proyecto)
              IN (
                    SELECT p.tipo_proyecto, p.cod_proyecto
                    FROM proyecto p
                    WHERE p.descripcion LIKE 'ROBOTICS'
                )
    AND (i.tipo_proyecto, i.cod_proyecto, i.nro_investigador)
            IN (
                    SELECT t.tipo_proyecto, t.cod_proyecto, t.nro_investigador
                    FROM TRABAJA_EN t
                    WHERE t.cod_area IN (
                                            SELECT a.cod_area
                                            FROM AREA_INVESTIGACION a
                                            WHERE s.investigacion_aplicada = TRUE
                                        )
                    GROUP BY t.tipo_proyecto, t.cod_proyecto, t.nro_investigador
                    HAVING COUNT(*) < 5
            )
;
```

Actividad previa

◄ Recuperatorio del 1er Parcial

Ir a...

Siguiente actividad

2do Parcial ▶

Mantente en contacto

Facultad, Pabellón Central Paraje Arroyo Seco. Campus Universitario. (B7001BBO) Tandil. Buenos Aires, Argentina

https://exa.unicen.edu.ar/

(+54) (0249) 438-5650 Conmutador: int. 2000

moodle@exa.unicen.edu.ar

Descargar la app para dispositivos móviles