



[Área personal](#)

[Mis cursos](#)

[BD-TUDAI-Tandil-IC-2023](#)

[Exámenes](#)

[Recuperatorio del 2do Parcial](#)

Comenzado el lunes, 3 de julio de 2023, 19:32

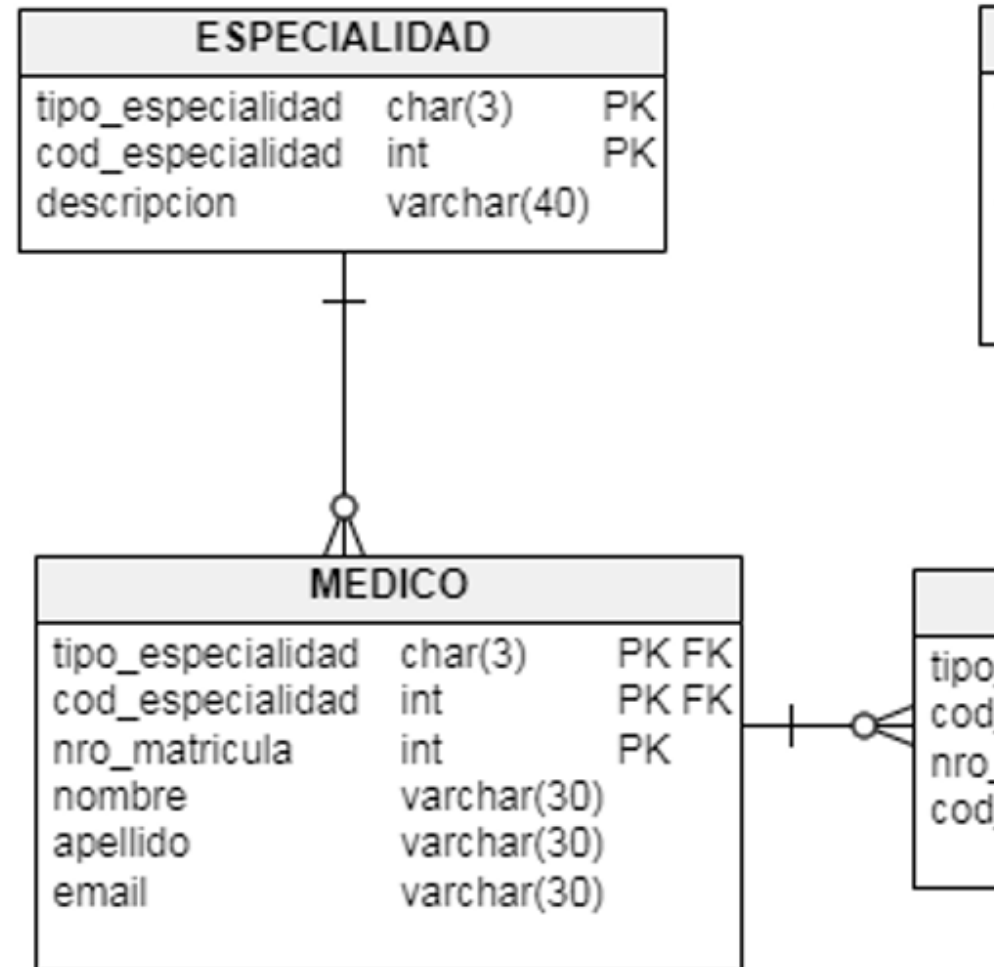
Estado Finalizado

Finalizado en lunes, 3 de julio de 2023, 20:31

**Tiempo
empleado** 59 minutos 23 segundos

Pregunta 1
Finalizado
Puntúa como 1,00

Utilice el recurso más conveniente para retornar todos los datos de los centros de salud que no posean médicos de una especialidad (campo descripción de especialidad) dada por parámetro.





```
CREATE OR REPLACE FUNCTION
sin_esta_especialidad(desc_especialidad varchar(40))
RETURNS TABLE(cod_centro int,nombre varchar(60), calle
varchar(60),numero int,sala_atencion boolean) AS $$
BEGIN
RETURN QUERY
SELECT cs.cod_centro,cs.nombre,cs.calle,cs,sala_atencion
FROM CENTRO_SALUD cs
JOIN ATIENDE a ON cs.cod_centro = a.cod_centro
JOIN MEDICO m ON
a.tipo_especialidad=m.tipo_especialidad AND
ON a.cod_especialidad = m.cod_especialidad AND
ON a.nro_matricula = m.nro_matricula AND
JOIN ESPECIALIDAD e ON m.tipo_especialidad =
e.tipo_especialidad AND
ON m.cod_especialidad = e.cod_especialidad
WHERE desc_especialidad != e.descripcion;

END
$$
LANGUAGE plpgsql;
```



```
-- Soluciones incorrectas:
-- Que no implementen correctamente la estructura de la
función
-- Que lo resuelvan 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 descripción de la tabla ESPECIALIDAD

CREATE OR REPLACE FUNCTION fn_centro_salud(especialidad
varchar(40))
RETURNS TABLE (cod_centro int, nombre varchar(60), calle
varchar(60), numero int, sala_atencion boolean) AS
$$
BEGIN
    RETURN QUERY
        SELECT c.cod_centro, c.nombre, c.calle, c.numero,
c.sala_atencion
        FROM CENTRO_SALUD c
        WHERE NOT EXISTS (
            SELECT 1
            FROM ATIENDE a
            JOIN MEDICO m ON (
                a.cod_especialidad =
```



```
m.cod_especialidad AND                                a.tipo_especialidad =  
m.tipo_especialidad AND                               a.nro_matricula =  
m.nro_matricula  
  
                )  
            JOIN ESPECIALIDAD e ON (  
                m.cod_especialidad AND                 e.cod_especialidad =  
                m.tipo_especialidad AND                 e.tipo_especialidad =  
                m.nro_matricula  
            )  
        WHERE    c.cod_centro = a.cod_centro  
        AND      e.descripcion = especialidad  
    );  
  
END  
$$  
language 'plpgsql';
```



Debe resolver el siguiente ejercicio teniendo en cuenta lo siguiente:

- Que debe reemplazar en TODO el script donde dice `unc_xxxxxx` 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_xxxxxx` 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 tablas claves extranjeras, NI orden de creación de las tablas los datos como así los datos en sí, 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.



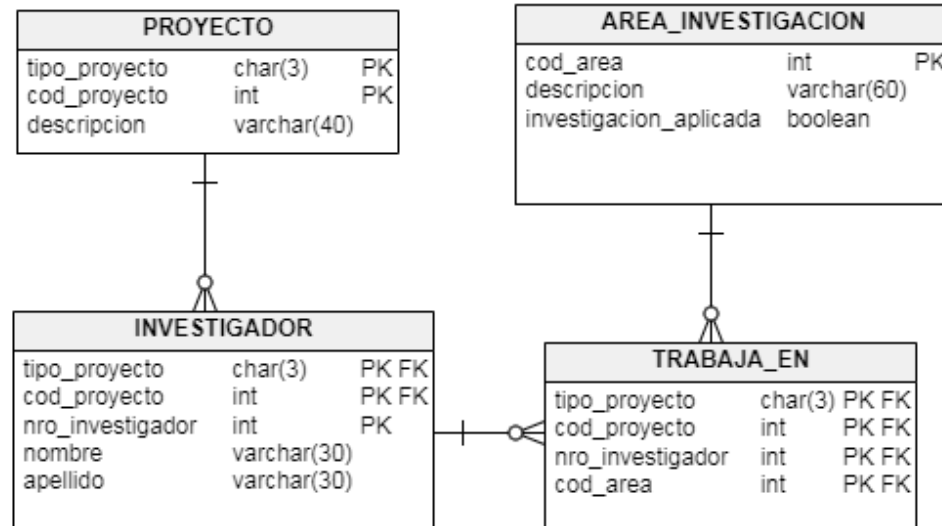
- Que el código que Ud. entregue NO debe hacer referencia a un esquema determinado, ni tampoco en la declaración de los objetos (triggers o funciones) suponga que al principio de su código hay un set search_path =
- NO debe tener un search_path.

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
Sin contestar
Puntúa como
1,00

Considerando el siguiente esquema - (recuerde que las tablas serán unc_XXXXXX_tabla siendo tabla en este caso, PROYECTO, INVESTIGADOR, etc.)



- a) Se necesita controlar que un Investigador(sin importar su proyecto) no pueda trabajar en mas de 3 áreas de Investigación.
- b) Se necesita controlar que si la descripcion del área de Investigación contiene la frase "*aplicada*" (sin importar si está en mayúsculas o minúsculas) el atributo `investigación_aplicada` 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





- 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 (mas allá de las pedidas obviamente), 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 investigador (sin importar su proyecto) no pueda atender en mas de 3 areas de invetigación.

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

-- al especificar "Sin importar el proyecto" debería utilizar sólo el

atributo nro_investigador

-- 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_area)
```

```
    from unc_xxxxxx_TRABAJA_EN where nro_investigador =  
NEW.nro_investigador) > 2) THEN
```

```
    RAISE EXCEPTION 'Un Investigador no puede atender en mas  
de 3 áreas';
```

```
END IF;
```

```
RETURN NEW;
```

```
END$$ LANGUAGE 'plpgsql';
```

```
CREATE OR REPLACE TRIGGER tg_xxxxxx_punto1
```

```
BEFORE INSERT OR UPDATE OF cod_area, nro_investigador
```

```
ON unc_xxxxxx_TRABAJA_EN
```

```
FOR EACH ROW
```

```
EXECUTE PROCEDURE fn_xxxxxx_a();
```





-- b) Se necesita controlar que si el nombre del área de Investigación contiene la frase "aplicada" (sin importar si está en mayúsculas o minúsculas) el atributo investigación_aplicada debe ser true

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

```
alter table unc_XXXXXX_area_investigacion add constraint  
chk_XXXXXX_b check (
```

```
(descripcion ilike '%aplicada%' and investigacion_aplicada = true)  
or (descripcion not ilike '%aplicada%'));
```

-- acá se usó ILIKE que ya me ahorra el tema de tratar las mayúsculas y minúsculas, 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 el proyecto estaba mal, se pedía explícitamente 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 `cos_area` y `nro_investigador`)

-- 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 (obviamente mas allá de lo que se pedía, o si no estaba) estaba 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.



Pregunta **3**
Sin contestar
Puntúa como
1,00

En caso de Ud. considere 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 ) );
```

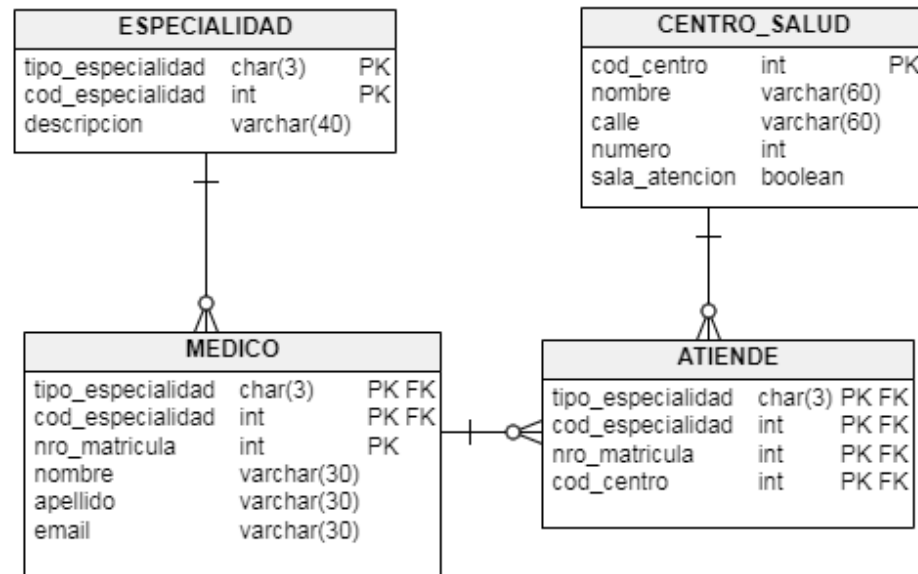
-- Esta restricción NO puede ser implementada en Postgres ya
que es de Tabla

-- Si hacían un assertion estaba mal

-- Si hacían triggers, estaba mal

Pregunta 4
Sin contestar
Puntúa como
1,00

Ante el siguiente modelo. Cree una vista **automáticamente actualizable** en PostgreSQL, optimizada bajo reglas de **equivalencia** (nombre cuales de ellas) y que las tuplas no puedan migrarse donde se seleccione el nombre y apellido de aquellos médicos que atienden en más de 3 centros de salud con sala de atención y que además tengan especialidad PEDIATRIA. (la vista debe comenzar con v_xxxxxx_nombre, siendo xxxxxx su número de legajo y nombre el nombre que Ud. le quiera dar)







```
-- Análisis:
-- 1) Se solicita: "cree una vista automáticamente actualizable
en PostgreSQL.... donde se seleccione el nombre y apellido ...."
-- Teniendo en cuenta los conceptos teóricos de "vista
actualizabe", 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: "nombre y apellido de aquellos médicos que
atienden en más de 3 centros de salud con sala de atención y que
además tengan especialidad PEDIATRIA"
-- 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
ESPECIALIDAD

-- 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_Medicos_Pediatricos
AS
    SELECT m.nombre, m.apellido
    FROM MEDICO m
    WHERE (m.tipo_especialidad, m.cod_especialidad)
           IN (
               SELECT e.tipo_especialidad,
e.cod_especialidad
               FROM ESPECIALIDAD e
               WHERE e.descripcion LIKE 'PEDIATRIA'
           )
    AND (m.tipo_especialidad, m.cod_especialidad,
m.nro_matricula)
           IN (
               SELECT a.tipo_especialidad,
a.cod_especialidad, a.nro_matricula
               FROM ATIENDE a
               WHERE a.cod_centro IN (
                                   SELECT cs.cod_centro
                                   FROM CENTRO_SALUD cs
                                   WHERE
cs.sala_atencion = TRUE
                                   )
               GROUP BY a.tipo_especialidad,
a.cod_especialidad, a.nro_matricula
               HAVING COUNT(*) > 3
           )
    ;
```

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

📱 Descargar la app para dispositivos móviles

🌐 <https://exa.unicen.edu.ar/>

☎ [\(+54\) \(0249\) 438-5650](tel:+5402494385650) Conmutador: int. 2000

✉ moodle@exa.unicen.edu.ar

[Facultad de Ciencias Exactas](#) – [UNICEN](#)

Contacto administradores plataforma: E-mail moodle@exa.unicen.edu.ar – Tel. [+54 0249 4385650](tel:+5402494385650) int. 2098