

INSTRUCCIONES

La entrega de las soluciones será en formato PDF con nombre de fichero:

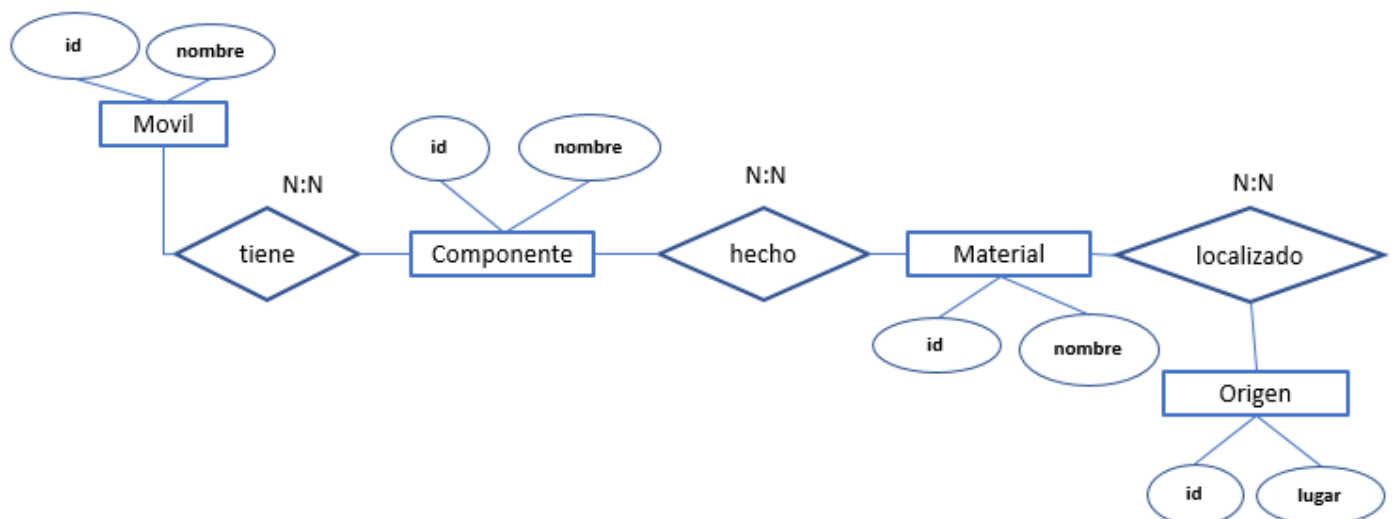
Ex2_UOXXXXXX_Nombre_Apellido1.pdf

No incluir signos de puntuación en el nombre del fichero tales como acentos o diéresis.

Indicar en el apartado **RESPUESTA** la solución a cada ejercicio. En algunos casos se proporciona la salida esperada como resultado. **Importante:** Obtener la misma salida no es indicativo de obtener la máxima puntuación del ejercicio. Los nombres indicados en negrita de tablas, funciones, procedimientos, triggers, columnas, tuplas, etc. deben respetarse en la solución propuesta o bien son explicativos (en el caso de los parámetros formales de funciones y procedimientos se podrá elegir la denominación que se desee).

Nota: los ejercicios 1 y 2 deberían abordarse en secuencia pues son dependientes.

BD: **movil**

MODELO E-R de la BD movil

Ejercicio 1 (1,5 puntos)

Crear una tabla aislada del resto de la base de datos **movil** llamada **estandar**, popularizarla y codificar una función que opere sobre ella, todo conforme al siguiente detalle:

- Columnas: **especificacion** (varchar(10) Primary Key); **velocidad** (integer Not Null)
- Popularizar con las tuplas: ('GPRS',50); ('EDGE',350); ('HSPA',21000); ('LTE',60000); ('LTEA',100000); ('NR',1000000)
- Crear la función **alo_sumo_21000()** que proyecte la tabla **estandar** por la columna **especificacion** incluyendo solo aquellas especificaciones que a lo sumo ofrezcan una velocidad de 21000.

RESPUESTA:

```
-- 1.
create table estandar(
    especificacion varchar(10),
    velocidad integer not null,
    primary key(especificacion)
);
--
insert into estandar values('GPRS', 50);
insert into estandar values('EDGE', 350);
insert into estandar values('HSPA', 21000);
insert into estandar values('LTE', 60000);
insert into estandar values('LTEA', 100000);
insert into estandar values('NR', 1000000);
--
create or replace function alo_sumo_21000() returns table
(spec varchar(10)) as $$
begin
    return query
        select especificacion
        from estandar
        where(velocidad <= 21000);
end;
$$ language plpgsql;
```

Ejercicio 2 (2 puntos)

La tabla **estandar** está aislada en la base de datos **movil** porque la utilizamos como referencia documental acumulada, supongamos que queremos bloquear temporalmente las operaciones de cambio en la citada tabla y procedemos como sigue: codificar el trigger **cambio_estandar_trigger** y la función asociada **cambio_estandar()** que operen antes de un update, delete o insert en la tabla **estandar** impidiendo llevar a término cualquiera de esas operaciones e imprimiendo por consola el mensaje 'tabla bloqueada: operación no realizada por falta de autorización'.

RESPUESTA:

```
-- 2.
create or replace function cambio_estandar() returns trigger as $$
begin
    raise notice 'Tabla bloqueada: operacion no realizada
por falta de autorizacion.';
    if(TG_OP = 'INSERT') then
        return old;
    elsif (TG_OP = 'UPDATE') then
        return old;
    else -- si llega a este punto, TG_OP = 'DELETE'.
        return new;
    end if;
end;
$$ language plpgsql;
--
create trigger cambio_estandar_trigger
before insert or update or delete on estandar
for each row execute procedure cambio_estandar();
```

Ejercicio 3 (2 puntos)

Codificar el trigger **cambios_movil_trigger** y la función asociada **cambios_movil()** que operen como sigue: tras una operación de inserción, borrado o actualización en la tabla **movil** se imprimirá por consola un mensaje específico que indique el tipo de operación que se ha realizado.

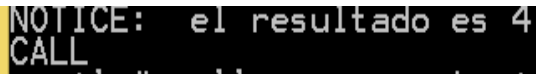
RESPUESTA:

```
create or replace function cambios_movil() returns trigger as $$
begin
    raise notice 'Operacion % realizada en la tabla "movil".', TG_OP;
    return null;
end;
$$ language plpgsql;
--
create trigger cambios_movil_trigger
after insert or update or delete on movil
for each statement execute procedure cambios_movil();
```

Ejercicio 4 (2 puntos)

Codificar un procedimiento con un parámetro **numero_materiales_componentel(x)** que opere como sigue: informará del número total de materiales que se asocian con un determinado componente, la invocación del procedimiento se corresponderá, considerando su único parámetro que además será sólo de entrada, con el **id** del componente que se esté consultando. El procedimiento imprimirá por consola el resultado junto con algún texto explicativo.

La siguiente captura se corresponde a la salida de la invocación **numero_materiales_componentel(103)**



```
NOTICE: el resultado es 4
CALL
```

RESPUESTA:

```
create or replace procedure numero_materiales_componentel
(id integer) as $$
declare i integer default 0;
begin
    select count(*) into i
    from componentematerial as cm
    where id_componente = id
    group by (id_componente);

    raise notice 'El resultado es %.', i;
end;
$$ language plpgsql;
```

Ejercicio 5 (2,5 puntos)

Codificar una función con un parámetro **material_todos_origenes(x)** que opere el siguiente resultado: invocándola con el **nombre** de un **material** listará el **id** y **nombre** de todos los orígenes que se asocian con el material en cuestión, lo hará en orden descendente según el **id** de **origen**, finalmente, en el caso de que el material no exista en la base de datos, en relación con la información que se solicita, se imprimirá por consola un mensaje notificándolo.

La siguiente captura se corresponde a la invocación **material_todos_origenes('coltan')**

```
material_todos_origenes
-----
(304,Tailandia)
(303,Brasil)
(302,Australia)
(301,Congo)
(4 filas)
```

RESPUESTA:

```
create or replace function material_todos_origenes(x varchar(50)) returns table(id integer, nombre varchar(100)) as $$
begin
    return query
        select o.id, o.lugar
        from origen as o
        inner join materialorigen as mo on (mo.id_origen
= o.id)
        inner join material as m on (mo.id_material = m.i
d)
        where lower(m.nombre) = lower(x)
        order by o.id desc;
    if not found then
        raise exception 'El material no existe en la base
.';
    end if;
end;
$$ language plpgsql;
```