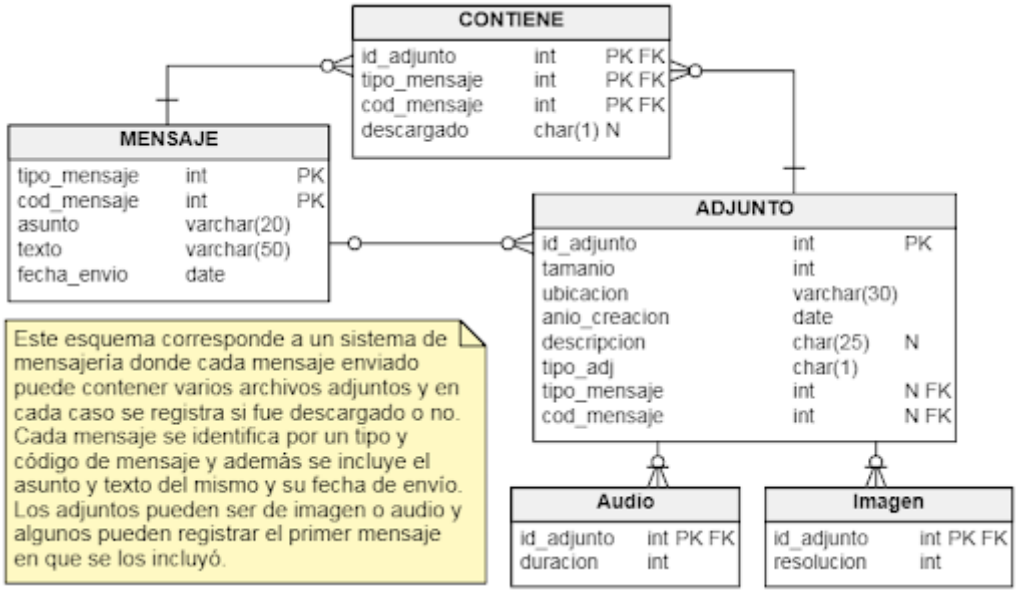
Comenzado el	Wednesday, 4 de November de 2020, 10:15
Estado	Finalizado
Finalizado en	Wednesday, 4 de November de 2020, 12:21
Tiempo empleado	2 horas 5 minutos

Pregunta 1

Finalizado

Puntúa como 1,00

Considerando el siguiente esquema de BD (script de creación):



Provea las sentencias de creación de las siguientes vistas en PosgreSQL, teniendo en cuenta de construirlas de manera que resulten automáticamente actualizables, siempre que sea posible:

a) VISTA_1, con el identificador de los mensajes, junto con el asunto, texto y fecha de envío, si es que contienen a lo sumo 3 adjuntos creados en el corriente año con tamaño menor a 50 Mb

b) VISTA_2, que contiene todos los datos comunes de adjunto más un atributo indicando la duración o la resolución, según sea el adjunto de tipo audio o imagen

Determine en cada caso si la vista resulta automáticamente actualizable o no, justificando su respuesta.

a)

```
create view VISTA_1 AS
    SELECT *
FROM MENSAJE m
join (

    select a.tipo_mensaje,a.cod_mensaje
    from adjunto
    where tamanio < 50
    group by a.tipo_mensaje, a.cod_mensaje
    having count(*) < 4 ) x
on x.tipo_mensaje = m.tipo_mensaje and x.cod_mensaje = m.cod_mensa

En este caso la vista no es actualizable debido a que tenemos mas
```

/

Pregunta 2

Finalizado

Puntúa como 1,00

b)

```
create view VISTA_2 AS
select a.*,au.duracion
from adjunto a
join audio au on a.id_adjunto = au.id_adjunto
    union
select a.*,i.resolucion
from adjunto a2
full join imagen i on i.id_adjunto = a2.id_adjunto;

En este caso la vista no es actualizado porque la definicion de la
```

Pregunta 3

Finalizado

Puntúa como 1,00

Considere la VISTA_2 definida en el Ej.1 y, mediante el procedimiento que crea más adecuado, implemente la propagación de cualquier operación (insert, update y delete) sobre dicha vista a las tablas base que conforman la misma. Explique el comportamiento que tendrán las actualizaciones sobre la vista, a partir de un ejemplo para cada operación.

```
create or replace procedure insert())
language plpgsql
    as $$
    begin
          if not exist (select 1 from adjunto where id_adjunto =
              --no esta ingresado en el sistema
              if tipo = 'I' then
                                    insert into imagen (id_adjun
ELSE
                insert into audio (id_adjunto, duracion) VALUES
          end if;
    end; $$

Si no tiene el adjunto en la tabla de arriba de la jerarquia ent
```

/

Pregunta 4

Finalizado

Puntúa como 1,00

Dadas las siguientes definiciones de vistas:

```
CREATE VIEW MovimientoUSDT
AS SELECT  id_usuario, moneda, fecha, tipo, comision, valor
FROM Movimiento
WHERE moneda LIKE '%USDT%';

CREATE VIEW MovUSDTValor
AS SELECT *
FROM MovimientoUSDT
WHERE valor < 1200
WITH LOCAL CHECK OPTION;

CREATE VIEW MovUSDTValorComi
AS SELECT *
FROM MovUSDTValor
WHERE comision < 25
WITH CASCADED CHECK OPTION;
```

Para las siguientes sentencias ejecutadas en el orden dado (acumulativas), considerando la tabla Movimiento del esquema del TPE, suponiendo que está inicialmente vacía y que los datos que se pretende insertar satisfacen las restricciones de integridad referencial definidas

Indique si cada operación procede o no.

1) INSERT INTO MovimientoUSDT (id_usuario, moneda, fecha, tipo, comision, valor)
VALUES ('1', 'EURO', to_date('2020-01-01','yyyy-MM-dd'), 'E', 5, 1500)

Procede

2) INSERT INTO MovUSDTValorComi (id_usuario, moneda, fecha, tipo, comision, valor)
VALUES ('2', 'EURO', to_date('2020-02-02','yyyy-MM-dd'), 'E', 20, 1000)

No Procede

3) INSERT INTO MovUSDTValor (id_usuario, moneda, fecha, tipo, comision, valor)
VALUES ('3', 'BITCOIN', to_date('2020-03-03','yyyy-MM-dd'), 'S', 30, 700)

Procede

/

4) INSERT INTO MovUSDTValor (id_usuario, moneda, fecha, tipo, comision, valor)

VALUES (‘4’, 'USDT', to_date('2020-04-04','yyyy-MM-dd'), 'E', 40, 1700)

No Procede

5) INSERT INTO MovUSDTValor (id_usuario, moneda, fecha, tipo, comision, valor)

VALUES ('5', 'USDT', to_date('2020-07-07','yyyy-MM-dd'), 'S', 33, 1200)

No Procede

6) INSERT INTO MovimientoUSDT (id_usuario, moneda, fecha, tipo, comision, valor)

VALUES ('3', 'BITCOIN', to_date('2020-03-03','yyyy-MM-dd'), 'E', 20, 1100)

Procede

Pregunta 5

Finalizado

Puntúa como 1,00

a) Justifique en forma clara y concisa su decisión para cada una de las operaciones (indique su número)

```
1. Procede porque mas alla de que cumpla la restriccion o no la vi
2. No procede porque la vista tiene chequeo en cascada, cumple par
3. Procede porque cumple con el chequeo local, ya que solo chequea
4. No procede porque al tener chequeo local verifica la restriccio
5. No procede porque al tener chequeo local verifica la restriccio
6. Procede porque no especifica ningun tipo de chequeo

Todas las inserciones anteriores ademas proceden porque los campos
```

Pregunta 6

Finalizado

Puntúa como 1,00

b) Indique el contenido de cada objeto (tabla y vistas) luego de finalizar la última operación.

```
Movimiento
   ('1', 'EURO', to_date('2020-01-01','yyyy-MM-dd'), 'E', 5, 1500)
   (‘3’, 'BITCOIN', to_date('2020-03-03','yyyy-MM-dd'), 'S', 30, 7
   ('3', 'BITCOIN', to_date('2020-03-03','yyyy-MM-dd'), 'E', 20, 1

Billetera

(usuario 1) En este caso supongo que es una extraccion pero no con
(usuario 3) No se que significa la letra S, pero supongo que es un

Las tres vistas estan vacias porque como ninguna tupla es visualiz
```

Pregunta 7

Finalizado

Puntúa como 1,00

La siguiente consulta sobre el esquema de Películas debería listar “los datos completos de las películas de idioma Español entregadas por un distribuidor internacional del país GL”

```
SELECT *
FROM unc_esq_peliculas.pelicula p
   JOIN unc_esq_peliculas.renglon_entrega re ON (p.codigo_pelicula = re.codigo_pelicula)
   JOIN unc_esq_peliculas.entrega e  ON (re.nro_entrega = e.nro_entrega)
   JOIN unc_esq_peliculas.distribuidor d  ON (e.id_distribuidor = d.id_distribuidor)
  JOIN unc_esq_peliculas.internacional i  ON (e.id_distribuidor = i.id_distribuidor)
WHERE p.idioma LIKE 'Español%' AND i.codigo_pais = ‘GL’;
```

a) ¿La consulta responde a lo solicitado? Justifique brevemente

b) ¿Representa una consulta optimizada? Si su respuesta es SÍ, justifique por qué. Si su respuesta es NO, plantee un SQL optimizado y explique la/s modificación/es introducida/s

a)

```
No cumple porque brinda todos los datos del ensamble, cuadno solo
Cumple con lo solicitado pero por la consulta por español podria t
 i.codigo_pais = ‘GL’; error sintáctico
```

Pregunta 8

Finalizado

Puntúa como 1,00

b)

```
--TRAIGO SOLO LOS DATOS DE PELICULA YA QUE LO PIDE EL ENUNCIADO
SELECT p.*
FROM unc_esq_peliculas.pelicula p
-- EL SIGUIENTE JOIN SOLAMENTE TRAE LOS DATOS QUE ME INTERESAN PAR
    JOIN (select nro_entrega,codigo_pelicula from unc_esq_pelicula
-- EL SIGUIENTE JOIN SOLAMENTE TRAE LOS DATOS QUE ME INTERESAN PAR
    JOIN (select nro_entrega,id_distribuidor from unc_esq_pelicula
--asumo ya que la consigna no aclara que se precisa idioma 'Españo
--EN EL WHERE AGREGO QUE EL ID DISTRIBUIDOR SE ENCUENTRE ENTRE LOS
WHERE p.idioma = 'Español' and e.id_distribuidor in (
    select i.id_distribuidor
    from unc_esq_peliculas.internacional i
    where i.codigo_pais = 'GL');
```

Pregunta 9

Finalizado

Puntúa como 1,00

Considere el esquema de Películas (unc_esq_peliculas). Se requiere llenar una tabla **RepEntrega** con el registro de la cantidad de entregas realizadas por cada uno de los distribuidores de un cierto tipo, dado por el usuario como parámetro.

Implemente el método más adecuado en PostgreSQL y dé una sentencia de ejecución del mismo

Nota: considere la tabla **RepEntrega** creada como sigue:

Create table RepEntrega (

id_distribuidor numeric(5,0) not null,

nombre_distribuidor varchar(80) not null,

cant_entregas int not null)

```
Si el usuario da un tipo por parametro entiendo que esta consulta

create or replace procedure llenado_tabla(tipo_p varchar(1))
language plpgsql
    as $$
    begin
        delete from RepEntrega; //Borrado total para que no se sup
        insert into RepEntrega (select d.id_distribuidor,d.nombre,
            from unc_esq_peliculas.distribuidor d
                join entrega e on d.id_distribuidor = e.id_distrib
            where tipo = tipo_p
            group by d.id_distribuidor,d.nombre);
    end; $$

execute procedure llenado_tabla(PARAMETRO);
```