

Comenzado el jueves, 4 de diciembre de 2025, 09:47

Estado Finalizado

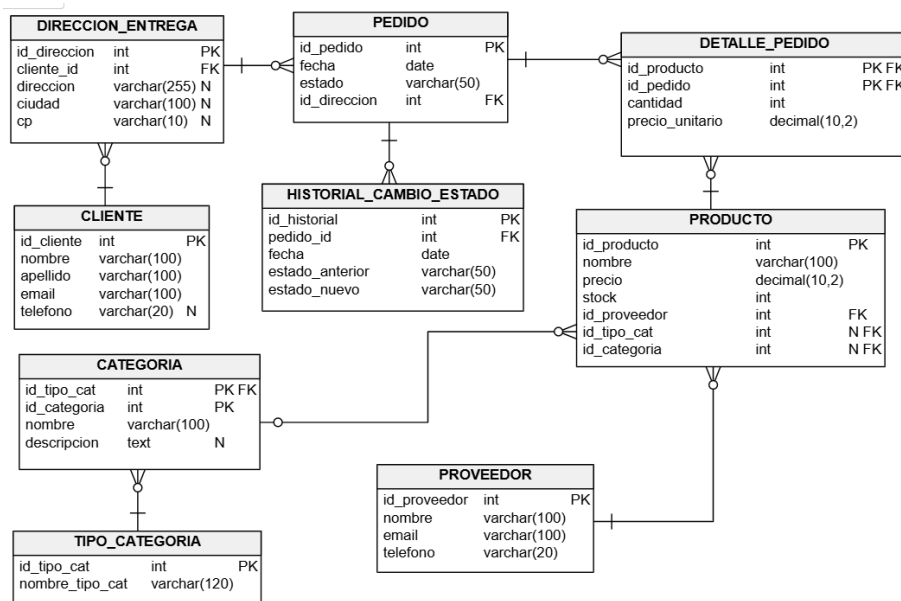
Finalizado en jueves, 4 de diciembre de 2025, 11:40

Tiempo empleado 1 hora 52 minutos

Calificación Sin calificar aún

Información

Dado el siguiente esquema de bases de datos cuyo script de creación se encuentra en el siguiente [link](#) resuelva los ejercicios planteados a continuación:



Pregunta 1

Finalizado

Se puntúa 1,00 sobre 1,00

En el esquema dado se requiere incorporar la siguiente restricción según SQL estándar utilizando el recurso declarativo más restrictivo posible (a nivel de atributo, de tupla, de tabla o general) y utilizando sólo las tablas/atributos necesarios.

- Verificar que cada pedido incluya al menos un producto

Seleccione la/las opción/es que considera correcta/s, de acuerdo a lo solicitado y justifique claramente por qué la/s considera correcta/s

Nota: Las opciones incorrectas pueden restar el puntaje

- ☐ a. ALTER TABLE detalle_pedido
ADD CONSTRAINT check_pedido_producto
CHECK (NOT EXISTS (
SELECT 1 FROM detalle_pedido p
GROUP BY id_pedido HAVING COUNT(*) < 1));
- ☐ b. CREATE ASSERTION check_pedido_producto CHECK (EXISTS (
SELECT 1 FROM producto p JOIN detalle_pedido d USING (id_pedido)
GROUP BY p.id_pedido HAVING COUNT(id_producto) < 1));
- ☐ c. CREATE ASSERTION check_pedido_producto CHECK (NOT EXISTS (
SELECT 1 FROM pedido p LEFT JOIN detalle_pedido d USING (id_pedido)
LEFT JOIN producto pr USING (id_producto)
GROUP BY p.id_pedido HAVING COUNT(id_producto) < 1));
- ☒ d. CREATE ASSERTION check_pedido_producto CHECK (NOT EXISTS (
SELECT 1 FROM pedido p LEFT JOIN detalle_pedido USING (id_pedido)
GROUP BY p.id_pedido HAVING COUNT(id_producto) < 1));

Respuesta correcta

La respuesta correcta es:

```
CREATE ASSERTION check_pedido_producto CHECK ( NOT EXISTS (  
SELECT 1 FROM pedido p LEFT JOIN detalle_pedido USING (id_pedido)  
GROUP BY p.id_pedido HAVING COUNT(id_producto) < 1 ));
```

Pregunta **2**

Finalizado

Se puntúa 1,00 sobre 1,00

Justifique claramente (sin describir el/los query/s) su/s elección/es del punto anterior

La respuesta correcta es la d. ya que necesito hacer un chequeo general porque utilizo los atributos id_pedido e id_producto de dos tablas diferentes (PEDIDO y DETALLE_PEDIDO).

Si bien DETALLE_PEDIDO tiene ambos atributos, necesito usar id_pedido desde la tabla PEDIDO para poder hacer un LEFT JOIN con DETALLE_PEDIDO y obtener una nueva tabla.

Con el LEFT JOIN figuraría NULL en id_producto en la nueva tabla creada, si no hubiera id_producto para un id_pedido específico en DETALLE_PEDIDO con lo cual lograría así contar los productos existentes para cada pedido.

Finalmente logro chequear que no existan pedidos que no incluyan productos.

Comentario:

Bien

Pregunta **3**

Finalizado

Se puntúa 1,00 sobre 1,00

En el esquema dado se requiere incorporar la siguiente restricción según SQL estándar utilizando el recurso declarativo más restrictivo posible (a nivel de atributo, de tupla, de tabla o general) y utilizando solo las tablas/atributos necesarios.

Las fechas en el historial correspondientes a cada pedido deben ser posteriores o iguales a la fecha del pedido.

Resuelva según lo solicitado y justifique el tipo de chequeo utilizado.

```
CREATE ASSERTION ej3 CHECK (  
  NOT EXISTS (  
    SELECT 1  
    FROM PEDIDO p  
    JOIN HISTORIAL_CAMBIO_ESTADO h  
    ON p.id_pedido = h.pedido_id  
    WHERE h.fecha < p.fecha  
  ));
```

Realizo un chequeo general porque necesito comparar el atributo fecha de dos tablas diferentes (PEDIDO e HISTORIAL_CAMBIO_ESTADO).

Comentario:

Bien

Pregunta **4**

Finalizado

Se puntúa 0,00 sobre 1,00

Sobre el esquema dado se requiere definir la siguiente vista, de manera que resulte automáticamente actualizable en PostgreSQL, siempre que sea posible. Si la vista es actualizable garantice que no haya migración de tuplas

- VI: que contenga todos los datos de los productos que aún no han sido incluidos en ningún pedido

```
CREATE OR REPLACE VIEW v4 AS
SELECT p.*
FROM PRODUCTO p
NOT IN (
SELECT 1
FROM DETALLE_PEDIDO dp
WHERE p.id_producto = dp.id_producto
)
WITH CHECK OPTION;
```

Comentario:

Mal la subconsulta si hay algún producto que no tiene pedidos al menos retorna un 1. Mal uso del NOT IN "que"?

Pregunta **5**

Finalizado

Se puntúa 0,50 sobre 1,00

Explique que significa que una operación de actualización sobre una vista produce una migración de tuplas. Ejemplifique sobre el esquema dado.

Que una operación de actualización sobre una vista produzca una migración de tuplas quiere decir que al actualizar una nueva fila no se chequee que se pueda ver en la vista. Así se podrían crear filas que no serían visibles desde la vista.

Al garantizar que no haya migración de tuplas, primero se chequea poder ver la fila actualizada en la vista, si no, se rechaza la actualización.

Si quisiera hacer:

```
INSERT INTO v4 VALUES (1, 'nombreProducto', 100.50, 20, 2, 3, 4);
```

Teniendo en cuenta que todos los datos son válidos, si el id_producto = 1 existe en la tabla DETALLE_PEDIDO, entonces la operación sería rechazada ya que no cumple con lo especificado en la vista, ese producto está incluido en un pedido.

Comentario:

El ejemplo no es adecuado porque si el id_producto = 1 existe en la tabla DETALLE_PEDIDO el insert no procede por violación de PK

Pregunta **6**

Finalizado

Se puntúa como 0 sobre 1,00

Para el esquema dado, es necesario consultar **los pedidos de un producto realizados después de una determinada fecha**, mostrando: el nombre y apellido del cliente, el id del pedido, la fecha del pedido, cantidad y el precio unitario; el id del producto y la fecha son datos que se aportan. Resuelva con el recurso que considere más conveniente.

```
CREATE OR REPLACE FUNCTION fn_ej6(id_producto_buscado int, fecha_buscada date)
RETURNS TABLE (
  nombre_cliente CLIENTE.nombre%type,
  apellido_cliente CLIENTE.apellido%type,
  id_pedido        PEDIDO.id_pedido%type,
  fecha_pedido     PEDIDO.fecha%type,
  cantidad         DETALLE_PEDIDO.cantidad%type,
  precio_unitario  DETALLE_PEDIDO.precio_unitario%type) AS
$$
BEGIN

RETURN QUERY SELECT
  c.nombre,
  c.apellido,
  p.id_pedido,
```

Comentario:

Casi Bien

- Está mal lanzar una excepción si no hay datos, una excepción es encontrar un error inesperado. Que no hayan datos NO es un error.

Pregunta **7**

Finalizado

Se puntúa como 0 sobre 1,00

Es posible plantear con una sentencia declarativa un control que no permita agregar registros al histórico de los pedidos que estan enviados? Si su respuesta es positiva plantee el control; caso contrario justifique porque.

No puedo porque necesito primero asegurarme de que el nuevo registro que quiero agregar tiene un pedido_id cuyo id_pedido correspondiente en PEDIDO tiene un estado diferente a DEVUELTO. Voy a tener que hacer directamente un trigger con su función correspondiente.

Comentario:

Si bien no se puede, su explicación no termina de convencer de que sabe porque.

Podría ser mas claro y expresar mejor las ideas.

Pregunta 8

Finalizado

Se puntúa 0,00 sobre 1,00

En caso de que sea posible plantee de manera procedural lo requerido en el punto anterior

```
CREATE OR REPLACE TRIGGER tr_ej8
BEFORE INSERT ON HISTORIAL_CAMBIO_ESTADO
FOR EACH ROW EXECUTE FUNCTION fn_ej8();

CREATE OR REPLACE FUNCTION fn_ej8() RETURNS TRIGGER AS
$$

DECLARE
v_estado_pedido PEDIDO.estado%type;

BEGIN

INSERT p.estado
INTO v_estado_pedido
FROM PEDIDO p
```

Comentario:

Mal

- No tiene que insertar nada en pedido.

Actividad previa

◀ Avisos

Ir a...

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](tel:+5402494385650) Conmutador: int. 2000

✉ moodle@exa.unicen.edu.ar



📱 Descargar la app para dispositivos móviles

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

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