README - Cómo ejecutar las pruebas (Windows / cmd.exe)

Propósito

Este documento explica cómo ejecutar automáticamente el DDL (`modelo_logico.sql`) y los tests (`tests/test_data.sql`) en un entorno Windows (cmd.exe). Incluye las comprobaciones clave que los scripts realizan y la salida esperada para validar el comportamiento de triggers relacionados con ventas, facturación y compras/inventario.

Requisitos previos

- PostgreSQL servidor instalado y accesible.
- Cliente `psql` en PATH (para ejecutar desde cmd.exe).
- Una cuenta de PostgreSQL con permisos para crear bases de datos y objetos.
- Versión recomendada: PostgreSQL >= 12 (por el uso de columnas GENERATED).

Archivos relevantes

- `c:\Users\vegas\OneDrive\Desktop\Model_logic\modelo_logico.sql` (DDL)
- `c:\Users\vegas\OneDrive\Desktop\Model_logic\tests\test_data.sql` (tests automatizados)
- `c:\Users\vegas\OneDrive\Desktop\Model_logic\tests\run_tests.cmd` (wrapper Windows)

Pasos rápidos (ejecutar en cmd.exe)

1. Abre una ventana de cmd.exe como el usuario de Windows habitual.
2. Edita `c:\Users\vegas\OneDrive\Desktop\Model_logic\tests\run_tests.cmd` y modifica las variables `USER` y `DB` a los valores correctos para tu entorno (por ejemplo: `SET USER=postgres` y `SET DB=tienda_dev`).
3. Ejecuta el wrapper (desde cualquier carpeta):

```
c:\Users\vegas\OneDrive\Desktop\Model_logic\tests\run_tests.cmd
```

Qué hace el wrapper

- Verifica que `psql` esté disponible en PATH.
- Intenta crear la base de datos (si ya existe se ignora el error).
- Aplica `modelo_logico.sql` (crea tablas, triggers y funciones).
- Ejecuta `tests/test_data.sql`, que inserta datos de ejemplo y ejecuta SELECTs para validar efectos de triggers.

Salidas esperadas y comprobaciones clave

Los `SELECT` en `tests/test_data.sql` muestran la salida. Aquí las comprobaciones más importantes y lo que deberías ver:

1. Venta - monto_total recalculado

- Tras insertar líneas en `detalles_venta`, el trigger debe recalcular `venta.monto_total`.
- Ejemplo esperado (valores aproximados según los INSERTs del test):
  - `monto_total` = 599.97 (2 * 199.99 + 1 * 199.99) — revisa la fila correspondiente en `venta`.

2. Facturación

- Se inserta una fila en `facturacion` referenciando la venta; el `total` debe coincidir con `venta.monto_total`.

3. Compras e inventario

- El test inserta una `compra` y dos `compra_producto` con cantidades 10 y 5.
- Antes de marcar `recibida = TRUE`, `inventario` no debe reflejar aún esas cantidades (puede no existir la fila o mostrar la cantidad previa).
- Al ejecutar `UPDATE compra SET recibida = TRUE ...`, los triggers deben aplicar las líneas al inventario.
- Resultado esperado: la fila `inventario` para `id_tienda = 1` e `id_producto = 1` mostrará `cantidad` incrementada en 15 (10 + 5).
- El campo `compra.total_compra` debe ser 1475.00 (10*100* + *5*95) según las líneas de ejemplo.

4. Idempotencia

- Repetir `UPDATE compra SET recibida = TRUE` sobre la misma compra no debe duplicar la cantidad en `inventario` (la columna `aplicada` evita re-aplicaciones).

Salida de ejemplo (resumen)

- SELECTs mostrarán filas con `venta.monto_total` (ej. 599.97), `facturacion.total` igual a ese monto, y `compra.total_compra` = 1475.00.
- SELECT de `inventario` debe mostrar cantidad incrementada a 15 para el par tienda/producto usado en la prueba.

Errores comunes y soluciones

- "psql no encontrado": añade la carpeta `bin` de PostgreSQL a la variable de entorno PATH.
- Permiso denegado al crear base de datos: usa un usuario con privilegios de creación o crea la BD manualmente antes.
- Columnas GENERATED no soportadas (Postgres < 12): convierte las columnas `GENERATED` en triggers o actualiza el servidor.
- Triggers no ejecutan efectos: revisa que las funciones y triggers se hayan creado correctamente en la BD (busca en `pg_trigger`, `pg_proc`).

Comprobaciones adicionales (manuales)

Puedes ejecutar manualmente estos SELECTs para validar resultados específicos:

```
-- Venta recien creada
SELECT v.id_venta, v.monto_total FROM venta v ORDER BY id_venta DESC LIMIT 1;

-- Detalles de venta
```

```
SELECT * FROM detalles_venta WHERE id_venta = (SELECT id_venta FROM venta ORDER BY
id_venta DESC LIMIT 1);

-- Facturacion asociada
SELECT * FROM facturacion WHERE id_venta = (SELECT id_venta FROM venta ORDER BY
id_venta DESC LIMIT 1);

-- Compra recien creada
SELECT id_compra, total_compra, recibida, aplicada, fecha_recepcion FROM compra
ORDER BY id_compra DESC LIMIT 1;

-- Inventario para tienda 1, producto 1
SELECT * FROM inventario WHERE id_tienda = 1 AND id_producto = 1;
```