Generador de Inserciones SQL desde Facturas de Supermercado

Módulo profesional optativo II

Autor: Manuel Gutiérrez

Índice de Contenidos

1. Introducción y Objetivos ----------1
2. Diseño del Modelo de Datos----------1
3. Arquitectura Técnica ----------1
4. Gestión de Duplicados e Integridad de Datos ----------1
5. Validaciones y Manejo de Errores ----------1
6. Resultados de Ejecución ----------1

1. Introducción y Objetivos

El objetivo principal de este proyecto es desarrollar una herramienta de minería de datos (ETL - Extract, Transform, Load) capaz de procesar información no estructurada proveniente de archivos de texto plano (simulación de tickets de supermercado) y transformarla en un script SQL estructurado para poblar una base de datos relacional.

El sistema está diseñado en Python y automatiza las siguientes tareas:

- Lectura masiva de archivos `.txt` desde un directorio específico.
- Extracción de información relevante (fechas, cajeros, productos, importes) mediante patrones de texto.
- Normalización de datos maestros para evitar redundancias.
- Generación de un archivo `.sql` con las sentencias `INSERT` necesarias.

2. Diseño del Modelo de Datos

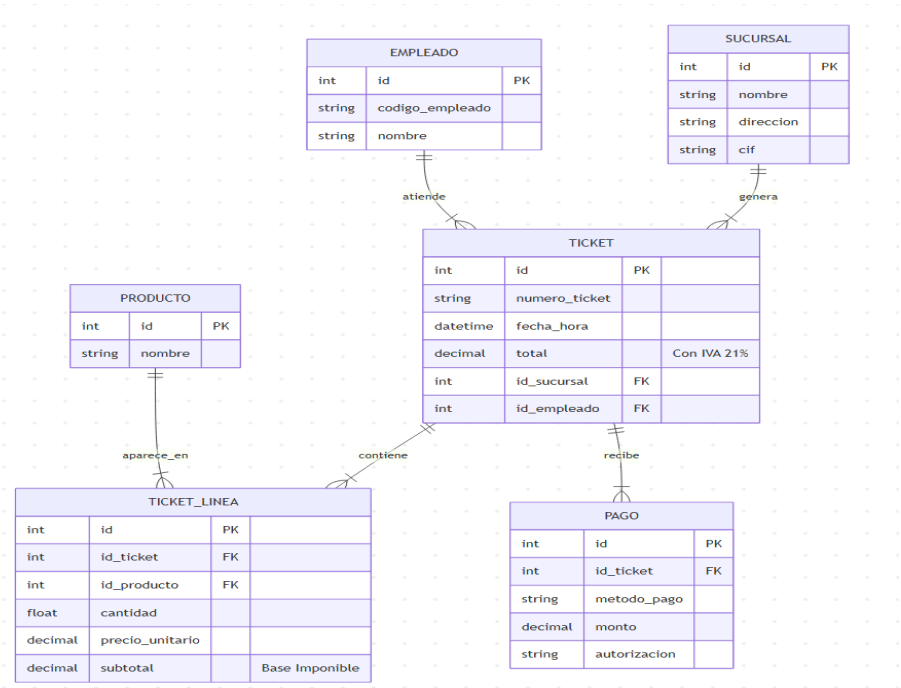
Para el almacenamiento eficiente de la información, se ha diseñado un modelo relacional normalizado en esquema de estrella, optimizado para evitar la redundancia de datos y asegurar la integridad referencial.

Tablas de Dimensiones (Catálogos):

- **sucursal:** Almacena la información de la tienda (Nombre, Dirección, CIF).
- **empleado:** Registro único de los cajeros identificados por su código interno.
- **producto:** Catálogo de artículos vendidos. Se garantiza que un mismo producto no se duplique, asignándole un ID único interno independientemente de cuántas veces se venda.

Tablas de Hechos (Transacciones):

- **ticket:** Cabecera de la compra. Actúa como entidad central relacionando la sucursal y el empleado mediante claves foráneas. Almacena el total final con impuestos.
- **ticket_linea:** Detalle de la compra. Vincula cada ticket con los productos específicos adquiridos, su cantidad y el precio unitario calculado.
- **pago:** Registra la transacción financiera asociada al ticket (método de pago y autorización).



3. Arquitectura Técnica

El script `build_insert_from_tickets.py` ha sido desarrollado utilizando un enfoque modular y orientado a objetos para la gestión de estados. Las decisiones técnicas clave incluyen:

Clase `IdRegistry` (Patrón Singleton en memoria) Se ha implementado una clase específica para actuar como memoria centralizada. Antes de asignar un ID a un empleado, sucursal o producto, el sistema consulta si la clave única (ej. nombre del producto) ya existe en su diccionario interno.

- Si existe: Devuelve el ID previamente asignado.
- Si no existe: Genera un nuevo ID secuencial y lo almacena. Esto garantiza que, al procesar 20 facturas con productos repetidos, la tabla de productos solo contenga registros únicos.

Librería `re` (Expresiones Regulares) Para el parsing de los archivos de texto, se utilizan expresiones regulares estrictas. Esto permite extraer datos como fechas, horas y precios independientemente de pequeños cambios en el espaciado o formato del ticket, haciendo el sistema mucho más robusto que una lectura por posición fija o partición de cadenas (`split`).

Librería `glob` y `pathlib` Estas librerías permiten la lectura automática, iterativa y agnóstica del sistema operativo de todos los archivos con extensión `.txt` en la carpeta objetivo.

4. Gestión de Duplicados e Integridad de Datos

Uno de los desafíos principales en la minería de datos de facturación es asegurar la coherencia numérica. Durante el análisis, se detectó que confiar ciegamente en el texto "TOTAL" impreso en el archivo podía inducir a errores si el archivo original contenía erratas o inconsistencias.

Solución de Integridad Matemática En lugar de leer el total del texto, el script realiza el cálculo programáticamente:

1. Extrae y suma los subtotales de cada línea de producto .
2. Aplica el impuesto (IVA 21%) al final mediante la fórmula: `total_con_iva = round(base_imponible * 1.21, 2)`

Esta decisión de diseño asegura que la base de datos sea matemáticamente consistente: la suma de las líneas del ticket más los impuestos coincide exactamente con el total registrado y el pago realizado.

5. Validaciones y Manejo de Errores

El sistema implementa varias capas de seguridad para garantizar una ejecución limpia:

1. **Validación de Datos Críticos:** Si un archivo no contiene una fecha válida o un número de ticket legible, el sistema descarta el archivo automáticamente y notifica el error en la consola, evitando la inserción de registros "huérfanos" o incompletos en la base de datos.
2. **Algoritmo de Reparación de Líneas :** Se detectaron casos donde la impresora del ticket generaba un salto de línea incorrecto, separando la descripción del producto de su precio.
   El script implementa un algoritmo que "mira adelante": si una línea no termina en precio y la siguiente sí, las fusiona antes de procesarlas.
3. **Control de Excepciones:** El uso de bloques `try-except` encapsulados asegura que si un archivo específico está corrupto, el programa no se detenga, permitiendo procesar el resto del lote correctamente.

6. Resultados de Ejecución

A continuación, se muestra el resultado de la ejecución del script procesando el lote de 20 facturas.

Captura de la consola de ejecución :

```
PS C:\Users\Hades\Documents\Clase\DAW2\Python\PythonPrimerTrimestre\InsercionesSQL> python .\build_insert_from_tickets.py
Encontrados 20 archivos. Procesando...
Tickets procesados.
SQL generado en: C:\Users\Hades\Documents\Clase\DAW2\Python\PythonPrimerTrimestre\InsercionesSQL\InsertUnderlineTicket.sql
```

Muestra del archivo SQL generado :

En la captura adjunta se visualiza un fragmento del código SQL generado, mostrando las tablas correspondiente de empleado y producto. Asimismo, en el margen derecho del editor, se puede constatar la longitud total del archivo resultante, evidenciando el volumen de instrucciones generadas.

```
InsercionesSQL > InsertUnderlineTicket.sql
INSERT INTO sucursal (id, nombre, direccion, cif) VALUES (1, 'SUPERMERCADOS EL AHORRO', '', 'B12345678');
INSERT INTO empleado (id, codigo_empleado, nombre) VALUES (1, '015', 'Juan Pérez');
INSERT INTO empleado (id, codigo_empleado, nombre) VALUES (2, '033', 'Laura García');
INSERT INTO empleado (id, codigo_empleado, nombre) VALUES (3, '011', 'Carlos Ruiz');
INSERT INTO empleado (id, codigo_empleado, nombre) VALUES (4, '028', 'Ana Fernández');
INSERT INTO empleado (id, codigo_empleado, nombre) VALUES (5, '019', 'Sofía Martínez');
INSERT INTO empleado (id, codigo_empleado, nombre) VALUES (6, '042', 'Diego Gómez');
INSERT INTO empleado (id, codigo_empleado, nombre) VALUES (7, '024', 'Marta López');
INSERT INTO producto (id, nombre) VALUES (1, 'Pasta Spaghetti 500g');
INSERT INTO producto (id, nombre) VALUES (2, 'Café Molido 250g');
INSERT INTO producto (id, nombre) VALUES (3, 'Arroz Redondo 1kg');
INSERT INTO producto (id, nombre) VALUES (4, 'Aceite de Oliva 1L');
INSERT INTO producto (id, nombre) VALUES (5, 'Sal Fina 1kg');
INSERT INTO producto (id, nombre) VALUES (6, 'Manzana Golden (kg)');
INSERT INTO producto (id, nombre) VALUES (7, 'Pechuga de Pollo (kg)');
INSERT INTO producto (id, nombre) VALUES (8, 'Cebolla (kg)');
INSERT INTO producto (id, nombre) VALUES (9, 'Huevos Camperos 12u');
INSERT INTO producto (id, nombre) VALUES (10, 'Pimiento Rojo (kg)');
INSERT INTO producto (id, nombre) VALUES (11, 'Queso Manchego 250g');
INSERT INTO producto (id, nombre) VALUES (12, 'Leche Entera 1L');
INSERT INTO producto (id, nombre) VALUES (13, 'Tomate Triturado 400g');
INSERT INTO producto (id, nombre) VALUES (14, 'Galletas María 200g');
INSERT INTO producto (id, nombre) VALUES (15, 'Banana (kg)');
INSERT INTO producto (id, nombre) VALUES (16, 'Azúcar 1kg');
INSERT INTO producto (id, nombre) VALUES (17, 'Pan Baguette 250g');
INSERT INTO producto (id, nombre) VALUES (18, 'Mantequilla 250g');
INSERT INTO producto (id, nombre) VALUES (19, 'Yogur Natural 125g');
INSERT INTO producto (id, nombre) VALUES (20, 'Refresco Cola 2L');
```