Carga de Datos y Estrategias de Destino - Día 4

Change Status

in-progress 40 min

Learning Objectives

1. Entender estrategias de carga de datos
2. Aprender carga a diferentes tipos de bases de datos
3. Comprender diferencias entre carga completa e incremental
4. Conocer optimizaciones de performance en carga

Practical exercise to apply the concepts learned.

Ejercicio: Implementar diferentes estrategias de carga

Preparar datos de ejemplo:

```
import pandas as pd
import numpy as np
from datetime import datetime

# Generar datos de ventas
np.random.seed(42)
ventas = pd.DataFrame({
    'venta_id': range(1, 1001),
    'cliente_id': np.random.randint(1, 101, 1000),
    'producto_id': np.random.randint(1, 51, 1000),
    'cantidad': np.random.randint(1, 11, 1000),
    'precio_unitario': np.round(np.random.uniform(10, 500, 1000), 2),
    'fecha_venta': pd.date_range('2024-01-01', periods=1000, freq='1H'),
    'updated_at': datetime.now()
})

ventas['total'] = ventas['cantidad'] * ventas['precio_unitario']

print(f"Generados {len(ventas)} registros de ventas")
print(ventas.head())
```

Carga completa (full load):

```
import sqlite3

def carga_completa_sqlite(df, tabla):
    conn = sqlite3.connect(':memory:')

    # Crear tabla
    conn.execute(f'''
        CREATE TABLE {tabla} (
            venta_id INTEGER PRIMARY KEY,
            cliente_id INTEGER,
            producto_id INTEGER,
            cantidad INTEGER,
            precio_unitario REAL,
            total REAL,
            fecha_venta TEXT,
            updated_at TEXT
        )
    ''')

    # Insertar datos
    df.to_sql(tabla, conn, if_exists='replace', index=False)

    # Verificar
    cursor = conn.execute(f"SELECT COUNT(*) FROM {tabla}")
    count = cursor.fetchone()[0]

    conn.close()
    return count

registros_cargados = carga_completa_sqlite(ventas, 'ventas_completas')
print(f"Carga completa: {registros_cargados} registros")
```

Carga incremental (simulada):

```python
def carga_incremental(df, archivo_parquet, ultimo_id=0):
    # Simular carga incremental: solo registros nuevos
    nuevos_registros = df[df['venta_id'] > ultimo_id]

    if len(nuevos_registros) > 0:
        try:
            # En producción, leer archivo existente y append
            nuevos_registros.to_parquet(
                archivo_parquet,
                engine='pyarrow',
                index=False
            )
            print(f"Carga incremental: {len(nuevos_registros)} nuevos registros")
            return len(nuevos_registros)
        except Exception as e:
            print(f"Error en carga incremental: {e}")
            return 0
    else:
        print("No hay nuevos registros para cargar")
        return 0

nuevos_cargados = carga_incremental(ventas, 'ventas_incremental.parquet', ultimo_id=500)
print(f"Registros nuevos agregados: {nuevos_cargados}")
```

Comparar estrategias:

```python
import time

def comparar_estrategias_carga():
    estrategias = {}

    # Medir carga completa
    start = time.time()
    carga_completa_sqlite(ventas, 'ventas_test')
    estrategias['completa'] = time.time() - start

    # Medir carga incremental (simulada)
    start = time.time()
    carga_incremental(ventas, 'ventas_inc_test.parquet', ultimo_id=800)
    estrategias['incremental'] = time.time() - start

    print("Comparación de estrategias:")
    print(".2f")
    print(".2f")

    return estrategias

resultados = comparar_estrategias_carga()
```

Verificación: ¿En qué situaciones usarías carga completa vs incremental? ¿Qué factores influyen en el tamaño óptimo de batch para carga de datos?

Requerimientos:

Pandas y SQLAlchemy
Conocimiento de bases de datos SQL
Familiaridad con formatos de archivo analíticos