Extract-Transfom-Load (ETL) en la gestión de datos geoespaciales: prácticas, herramientas y casos de uso

Autores: Salinas Carlos Alberto [1], Scantamburlo Javier [2] , Quaino Fabio Matías José [2]

[1] IDECOR - Infraestructura de Datos Espaciales de Córdoba - Rivera Indarte 748, C.P. 5000, Córdoba - (0351) 4286048
[2] AMDG S.A.S. - General Paz 554, Piso 5, C.P. 5900, Villa María, Córdoba - (0353) 6570273

Contacto: carlosalberto.salinas@gmail.com, javier.scantamburlo@holon.com.ar, fabiomjq@gmail.com

Resumen: En este artículo se analiza la importancia de los Procesos de Extracción, Transformación y Carga (ETL) en la gestión efectiva de datos geoespaciales. Se examinan herramientas de software, junto con prácticas recomendadas para mejorar la calidad de los datos geoespaciales mediante la validación y limpieza. Además, se presentan casos de uso ilustrativos que destacan la versatilidad y la aplicabilidad de los ETL en escenarios geoespaciales diversos.

Palabras clave: ETL, Interoperabilidad, Apache Airflow, Talend Studio, Información geoespacial, Calidad de datos, Integración de datos.

---

1. INTRODUCCIÓN

La gestión eficiente de datos geoespaciales constituye un aspecto fundamental en numerosos ámbitos, como la planificación, la toma de decisiones, la optimización de recursos y la gestión del territorio. En este contexto, los Procesos de Extracción, Transformación y Carga (ETL) se destacan como herramientas esenciales para manejar y analizar estos datos, facilitando la transferencia desde diversas fuentes hacia sistemas de destino donde pueden ser almacenados y utilizados efectivamente.

Este documento se propone explorar el papel crucial de los procesos ETL en la gestión de datos geoespaciales, enfocándose en su aplicación práctica, las mejores prácticas asociadas y los desafíos específicos que surgen en este contexto. A través de una revisión detallada de herramientas, prácticas y casos de uso relevantes, se pretende proporcionar una visión integral y actualizada de la

implementación de procesos ETL en el ámbito de la información geoespacial.

2. ETL EN EL CONTEXTO DE INFORMACIÓN GEOESPACIAL

Los Procesos de Extracción, Transformación y Carga (ETL) desempeñan un papel crucial en la gestión y análisis de datos geoespaciales. Se requiere un enfoque cuidadoso en cada etapa del proceso, desde la extracción hasta la carga en sistemas de destino.

Extracción (Extract): En la etapa de extracción, se recopilan datos desde diversas fuentes, como bases de datos espaciales, archivos geográficos y servicios web de mapas. Estos datos pueden incluir información sobre ubicaciones geográficas, coordenadas y atributos asociados. Es crucial garantizar la precisión y completitud de la extracción para mantener la integridad de los datos.

Transformación (Transform): Una vez extraídos, los datos geoespaciales se someten a procesos de transformación para prepararlos para su análisis y uso posterior. Esto puede implicar la limpieza y normalización de datos, la conversión entre sistemas de coordenadas y la generación de nuevas variables derivadas. La complejidad de la transformación aumenta debido a la naturaleza multidimensional de la información espacial y la variedad de formatos en los que se presentan los datos.

Carga (Load): En la etapa de carga, los datos transformados se cargan en un sistema de destino donde estarán disponibles para su análisis y visualización. Esto puede ser una base de datos geoespacial, un sistema de información geográfica (GIS) o una aplicación de visualización de mapas. La eficiencia en la carga es esencial para garantizar la accesibilidad y disponibilidad de los datos para los usuarios finales.

3. FORMATO GEOJSON

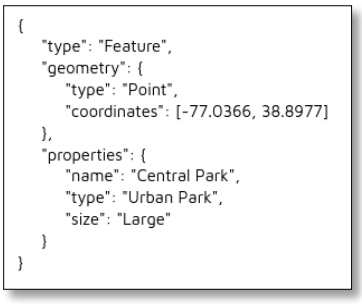


Figura 1. GeoJSON

El formato GeoJSON emerge como una herramienta fundamental en la representación de datos geoespaciales debido a su simplicidad, versatilidad y compatibilidad con una amplia gama de aplicaciones y sistemas. Diseñado como una extensión del formato JSON (JavaScript Object Notation), GeoJSON proporciona una estructura de texto legible que describe objetos geoespaciales y sus atributos asociados.

GeoJSON se compone principalmente de dos elementos:

- Geometría: Define la forma geográfica representada, que puede ser un punto, una línea o un polígono. Esta geometría incluye coordenadas que indican su ubicación en la Tierra.
- Propiedades: Proporciona información adicional sobre la geometría, como su nombre, tipo o cualquier otro atributo relevante.

Las figuras básicas que pueden representarse mediante GeoJSON incluyen:

- Punto: Representa un único punto en el espacio.
- Línea: Describe una secuencia ordenada de segmentos de línea.
- Polígono: Define un área cerrada mediante una secuencia de coordenadas que delimitan sus límites.

4. USO DE GEOJSON EN EL CONTEXTO DE ETL

GeoJSON se ha consolidado como un formato popular y versátil para la representación de datos geoespaciales, siendo ampliamente utilizado en aplicaciones web y sistemas de información geográfica (GIS). Su simplicidad y compatibilidad con una amplia variedad de herramientas hacen de GeoJSON una opción ideal en el campo de la geoinformática.

En los procesos ETL geoespaciales, el uso de GeoJSON facilita la interoperabilidad entre diferentes sistemas y aplicaciones, simplifica la integración de datos geográficos y estandariza la representación de la información. Además, su formato ligero y legible facilita el intercambio de datos entre plataformas y aplicaciones.

El formato GeoJSON juega un papel esencial en cada etapa del proceso ETL. Durante la extracción, GeoJSON facilita la recopilación de datos geoespaciales desde diversas fuentes. En la etapa de transformación, GeoJSON estandariza la representación de la información geográfica, facilitando su procesamiento y análisis. Finalmente, en la etapa de carga, GeoJSON asegura la interoperabilidad al cargar los datos transformados en sistemas de destino compatibles.

5. CALIDAD DE DATOS GEOESPACIALES: VALIDACIÓN Y LIMPIEZA

La calidad de los datos geoespaciales es esencial para asegurar la precisión y fiabilidad de los análisis y decisiones derivadas de ellos. La validación y limpieza de datos geoespaciales se lleva a cabo principalmente en la etapa de Transformación de un proceso ETL. En esta etapa, los datos extraídos de diversas fuentes se someten a una serie de operaciones para prepararlos y estructurarlos de manera adecuada antes de cargarlos en el sistema de destino.

Algunas de las tareas más comunes de validación y limpieza de datos geoespaciales son:

- **Verificación de la topología**: Asegura la corrección de las geometrías y evita errores topológicos como solapamientos o brechas.
- **Control de la integridad de los datos**: Verifica que los datos estén completos y no falten valores clave, como coordenadas geográficas o atributos esenciales.
- **Comprobación de la precisión espacial**: Evalúa la exactitud de las geometrías en comparación con fuentes de referencia conocidas, como mapas oficiales o imágenes satelitales.
- **Normalización de atributos**: Estandariza los valores de los atributos para garantizar consistencia y coherencia en los datos, como la conversión de unidades de medida a un sistema común.

6. CALIDAD DE DATOS CON FUNCIONES POSTGIS

PostGIS ofrece una variedad de funciones de utilidad para manipular datos en bases de datos mediante SQL y asegurar la calidad de los mismos. A continuación, se describen algunas de estas funciones clave:

- **ST_MakeValid**: Transforma geometrías inválidas en geometrías válidas según los estándares del modelo de datos geoespaciales. Corrige problemas de topología como vértices superpuestos o duplicados en polígonos, líneas que se cruzan a sí mismas, y polígonos con agujeros mal definidos.
- **ST_Dump**: Descompone geometrías complejas, como GeometryCollection o MultiPolygon, en una serie de geometrías más simples, como Point o Polygon. Esto facilita la manipulación y análisis de cada geometría individual
- **ST_Union**: Combina múltiples geometrías en una sola, eliminando fronteras internas en el caso de polígonos adyacentes.
- **ST_Force2D**: Convierte una geometría tridimensional a bidimensional,

eliminando la dimensión Z (altura) y manteniendo solo las coordenadas X e Y.

- **ST_CollectionExtract**: Permite extraer geometrías de un tipo específico (como por ejemplo Polygon) de una geometría de colección (GeometryCollection).
- **ST_Multi**: Convierte una geometría simple en su equivalente múltiple, por ejemplo, de Polygon a MultiPolygon.
- **ST_Transform**: Esta función transforma una geometría de un sistema de coordenadas a otro.

A continuación, se detallarán algunos ejemplos de uso en SQL:

6.1. Ver detalle de geometría no válida

```
SELECT
    *, ST_IsValidDetail(geom) AS geom_validity_detail
FROM
    my_table
WHERE
    (SELECT valid FROM ST_IsValidDetail(geom)) != TRUE;
```

Figura 2. SQL detalle geometría

Esta instrucción SQL consulta una tabla y selecciona todas las filas de un registro cuya geometría no sea válida, además agrega una columna adicional que contiene el detalle sobre el error de validez de la geometría.

6.2. Hacer válidas las geometrías de una tabla y actualizarlas

```
UPDATE my_table
SET geom = ST_MakeValid(geom)
WHERE
    NOT ST_IsValid(geom);
```

Figura 3. SQL validar geometría

Este código SQL actualiza las geometrías de una tabla corrigiendo aquellas que son inválidas. Utiliza la función ST_MakeValid para hacer válidas las geometrías (geom) que no cumplen con los estándares de validez (NOT ST_IsValid).

6.3. Normalizar tipos de geometría (punto, multipunto; línea, multilínea; polígonos, multipolígonos)

```
SELECT
    entityid,
    ST_Union(geom) AS normalized_geom
FROM
    (
        SELECT
            entityid,
             ST_Multi(ST_CollectionExtract((ST_Dump(ST_MakeValid(geom))).geom, 3))
AS geom
        FROM
            my_table
        WHERE
            geom IS NOT NULL
    ) AS extracted_geometries
GROUP BY
    entityid;
```

Figura 4. SQL normalizar geometría

Este ejemplo SQL normaliza tipos de geometría de una tabla PostGIS. Primero, selecciona y valida las geometrías no nulas, luego descompone geometrías compuestas en partes individuales y extrae sólo los polígonos. Convierte estas geometrías a su forma múltiple (MultiPolygon) y finalmente, combina todas las geometrías para cada entityid en una sola geometría.

7. MATRIZ DE HERRAMIENTAS ETL

Esta sección presenta un análisis comparativo de diversas herramientas ETL, resaltando sus ventajas y desventajas en el contexto de la gestión de datos geoespaciales.

Software	Puntos positivos	Puntos negativos
Apache Airflow	Escalabilidad sencilla. Arquitectura basada en código. Amplia variedad de herramientas y librerías. Compatible con programación dirigida por eventos. Posibilidad de implementar IA gracias a su base en Python.	Curva de aprendizaje pronunciada. Configuración inicial compleja. No es la opción óptima para procesamiento en tiempo real.
Talend Studio	Interfaz visual intuitiva. Admite tanto ETLs como servicios web o APIs.	Descontinuación de la versión gratuita. Insuficiente soporte para

	Gran diversidad de componentes. Funciones avanzadas de gestión de datos.	programas de escritorio. Problemas en configuración y despliegues.
Node-RED	Interfaz visual intuitiva. Ideal para procesamiento en tiempo real. Fácil instalación.	No apto para flujos complejos. Escalabilidad limitada. Diseñado principalmente para sistemas dinámicos en lugar de grandes volúmenes de datos.
SSIS	Interfaz visual intuitiva. Amplia variedad de componentes.	Limitado a entornos Windows y SQL Server. Rendimiento inferior con tecnologías no relacionadas con Microsoft.
Plataformas de integración en la nube (Airbyte, Dataddo)	Facilidad de uso. Amplia gama de conectores con distintas herramientas. Escalabilidad sencilla. No requiere instalación obligatoria.	Limitaciones en la personalización. Dependencia de conectores externos. Complejidad en la gestión de errores. Ausencia de componentes geoespaciales.

Tabla 1. Comparativa de herramientas ETL

Se han seleccionado Apache Airflow y Talend Studio como softwares representativos para ilustrar las opciones de desarrollo en el contexto de procesos ETL geoespaciales. Airflow se destaca por su escalabilidad, arquitectura basada en código y capacidad para implementar inteligencia artificial debido a su base en Python. Talend ofrece una interfaz visual intuitiva, una amplia gama de componentes y funciones avanzadas de gestión de datos, además de la posibilidad de exponer los datos mediante web services o APIs REST.

8. BIBLIOTECAS EN PYTHON PARA MANIPULACIÓN DE DATOS

El ecosistema de Python ofrece una variedad de bibliotecas poderosas para el análisis y procesamiento de datos geoespaciales, que pueden integrarse con Apache Airflow para su utilización en tareas de un DAG.

8.1. Pandas: análisis de datos estructurados

Pandas es una biblioteca ampliamente utilizada en el análisis de datos estructurados y series temporales. Aunque su principal enfoque no está en datos geoespaciales, es fundamental en la carga de datos, limpieza, preprocesamiento, filtrado y agregación de datos, lo que resulta crucial para preparar conjuntos de datos antes de aplicar técnicas de análisis espacial.

8.2. Geopandas: análisis geoespacial integrado

GeoPandas es una extensión de Pandas que agrega capacidades geoespaciales, convirtiéndola en una herramienta poderosa para el análisis y manipulación de datos geoespaciales. Permite la creación y manipulación de objetos geoespaciales como puntos, líneas y polígonos, y ofrece operaciones espaciales avanzadas como unión y disolución de geometrías, cálculo de áreas y distancias, y consultas espaciales.

8.2. GDAL: manipulación avanzada de datos geoespaciales

GDAL es una biblioteca que proporciona una amplia gama de herramientas para leer, escribir y manipular datos raster y vectoriales en diversos formatos. Es especialmente útil para tareas avanzadas de procesamiento de imágenes, como reproyección de datos, conversión entre formatos raster, y aplicaciones de operaciones matemáticas y estadísticas. Además, GDAL ofrece herramientas para el procesamiento de datos vectoriales, como simplificación de geometrías, transformación de coordenadas y extracción de atributos.

9. CASOS DE USO CON APACHE AIRFLOW

En esta sección, se presentarán cinco casos de uso ilustrativos que demuestran la eficacia y versatilidad de Airflow ETL en diversos escenarios.

9.1. Conversión de archivos “.shp” a “.geojson”

Apache Airflow se emplea para automatizar la conversión de datos geoespaciales entre diferentes formatos. Esta funcionalidad permite gestionar eficientemente la transformación de datos, extrayendo información de un formato como .shp y convirtiéndola en GeoJSON de manera automatizada.

Figura 5. Airflow caso 9.1.

9.2. Migración de datos de MySQL a PostGIS

Airflow facilita la migración de datos geoespaciales entre diferentes sistemas de bases de datos, como MySQL y PostGIS. Esto posibilita el traslado de datos a un entorno más propicio para el análisis espacial, con la capacidad de extraer, transformar y cargar los datos de forma automatizada.

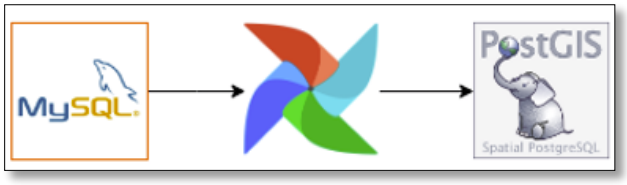


Figura 6. Airflow caso 9.2.

9.3. Invocar servicio web y almacenar en PostGIS

Mediante la integración con servicios web externos, Apache Airflow obtiene datos geográficos actualizados y los almacena en bases de datos como PostGIS. Esta funcionalidad asegura que los datos geográficos se mantengan actualizados y disponibles para su análisis y visualización de manera automatizada.

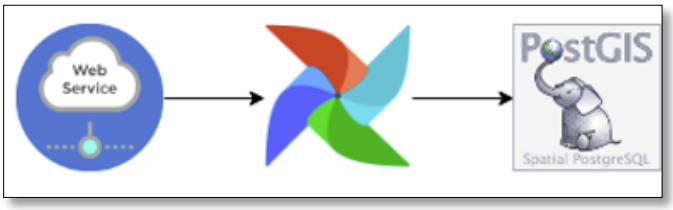


Figura 7. Airflow caso 9.3.

9.4. Completar información en base a un servicio externo

Se utiliza Apache Airflow para enriquecer los datos geográficos utilizando servicios externos de geocodificación. Esto permite completar la información geográfica de los registros existentes, mejorando así la calidad y utilidad de los datos geoespaciales.

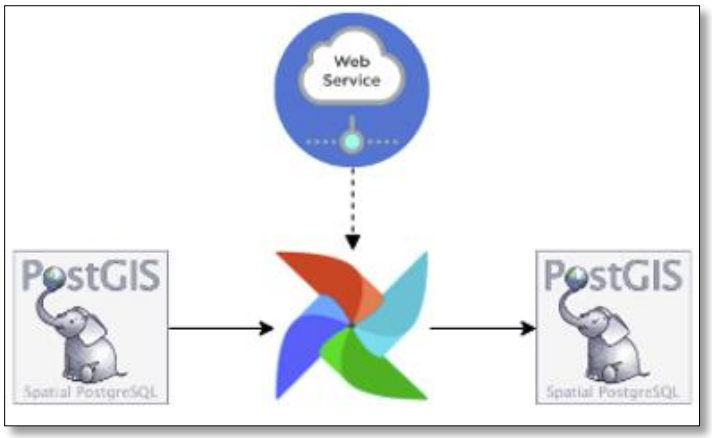


Figura 8. Airflow caso 9.4.

9.5. Uso de Algoritmos Intermedios para Calidad de Datos

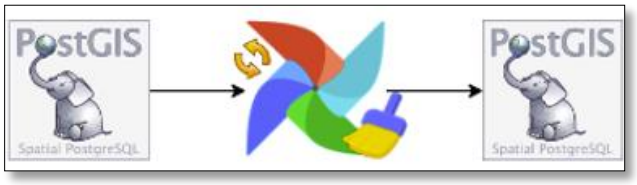


Figura 9. Airflow caso 9.5.

10. CASOS DE USO CON TALEND STUDIO

En esta sección se presentan casos de uso que ilustran la eficacia de Talend Studio en la realización de tareas específicas en el contexto de Extracción, Transformación y Carga de datos.

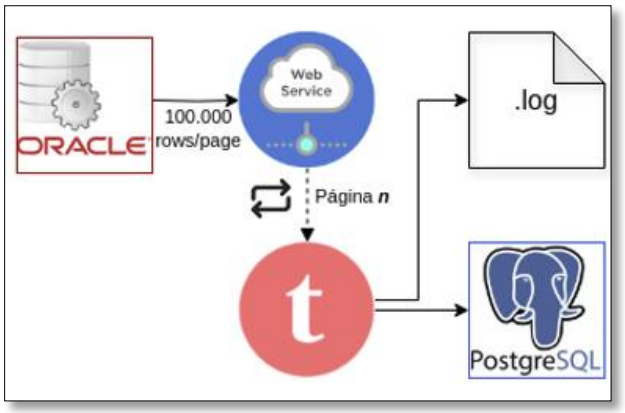


Figura 10. Talend caso 10.1.

10.1. Migración de registros masiva

Talend Studio permite realizar migraciones masivas de datos de manera eficiente. En este escenario específico, se emplea un servicio para extraer millones de registros de una base de datos de forma paginada. El proceso ETL se encarga de invocar este servicio página por página hasta que se complete la extracción, almacenando los registros obtenidos en la nueva base de datos. En caso de que se produzca algún error durante la ejecución del proceso, se registra la página en la que ocurrió el incidente dentro del archivo de log

10.2. Generación de nuevas tablas y vistas

Talend Studio destaca por su versatilidad en la integración de datos procedentes de diversas fuentes de manera eficiente. Esta herramienta permite extraer registros de múltiples tablas o vistas mediante operaciones JOIN, lo que facilita la creación de nuevas tablas o vistas que contengan los datos procesados de las fuentes originales. Además, la comunidad ha desarrollado un plugin denominado "Talend Spatial" para la versión 7 de Talend, el cual amplía las funcionalidades del software base al ofrecer herramientas para la curación de polígonos y operaciones de agregación destinadas a realizar cálculos geoespaciales dentro de los procesos ETL.

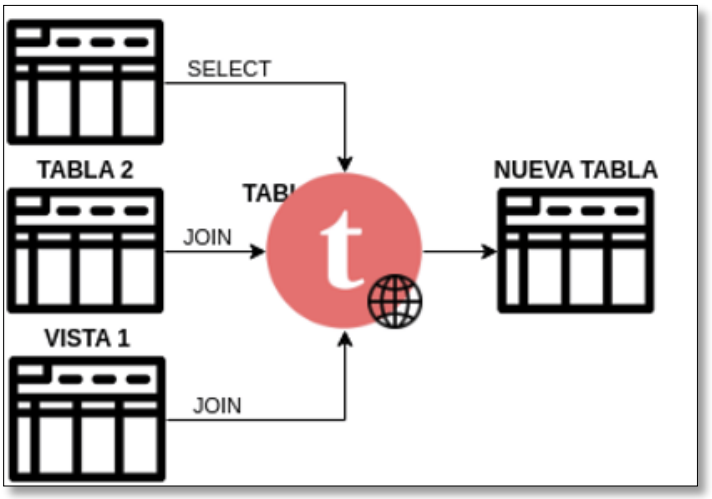


Figura 11. Talend caso 10.2.

11. CASO DE ÉXITO EN IDECOR

IDECOR ha implementado Talend Studio como herramienta para realizar migración de datos entre sistemas, integrando ETLs y servicios web. A continuación, se presenta el contexto del caso y su solución técnica.

11.1. Contexto

1. Origen de datos: Una base de datos Oracle Spatial que contiene datos geoespaciales y una base de datos Oracle on-premise con datos transaccionales.
2. Destino de datos: Una base de datos PostGIS en la nube.

11.2. Diseño técnico de la solución

- Creación de Stored Procedures en Oracle para la extracción de novedades según un rango de fechas específicas.

- Desarrollo de servicios SOAP para la publicación de los datos expuestos por los stored procedures. Los datos geoespaciales se serializan como GeoJSON, quedando de este modo establecido un protocolo y un formato estándar de interoperabilidad, que puede ser publicado en Internet de manera segura.
- Implementación de paginado para gestionar grandes volúmenes de datos de manera eficiente, permitiendo su transferencia sin saturar el sistema
- Desarrollo de ETL en IDECOR para la extracción de datos a partir de los servicios SOAP (desarrollados anteriormente) y su posterior inserción en la base de datos PostGIS que se encuentra en la nube.
- Desarrollo de resiliencia a fallos de comunicación mediante mecanismos de reintento y gestión de errores. Se lleva un registro detallado de logs para acceder y analizar el historial de ejecuciones.
- Ajuste de geometrías utilizando funciones de PostGIS para asegurar la consistencia y calidad de los datos
- Orquestación del ETL para asegurar un flujo controlado de ejecución. Se programaron ejecuciones periódicas según días y horarios específicos mediante Crontab.
- Implementación de paneles de observabilidad y alertas para monitorear el estado de las ejecuciones.

11.3. Principales agregados

- Actualización solo de novedades, evitando la redundancia y optimizando el rendimiento.
- Intercambio y serialización de datos de manera efectiva a través Internet.

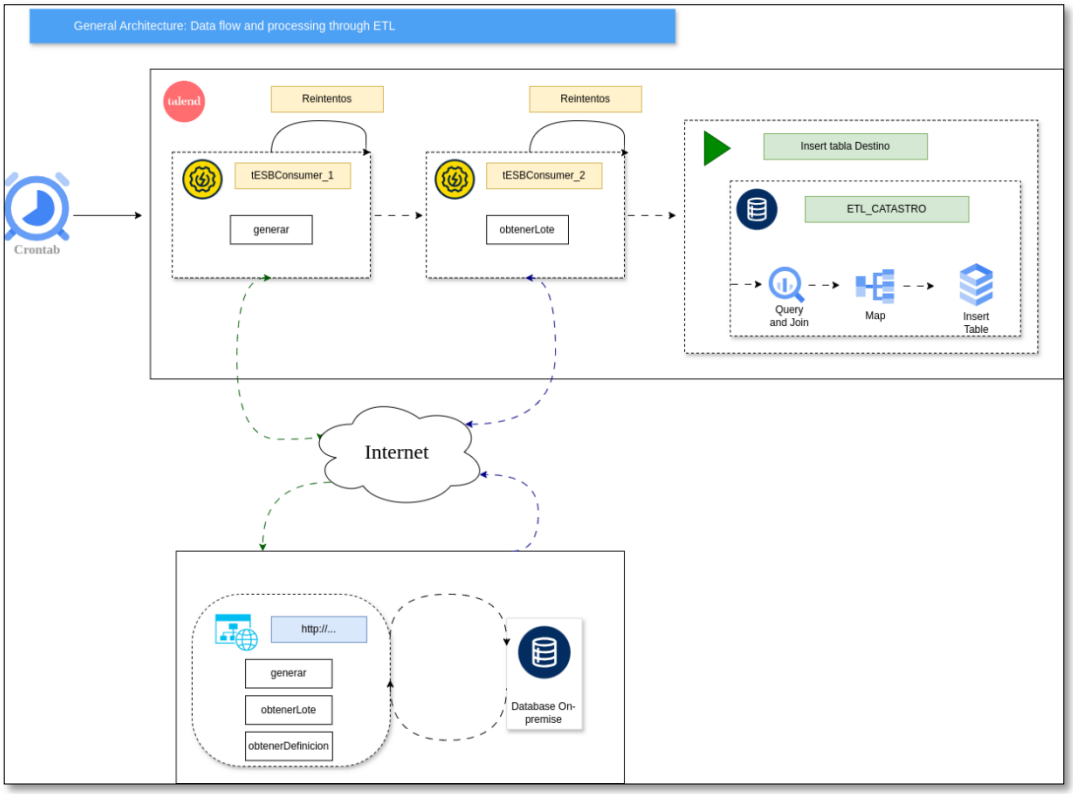


Figura 12. Arquitectura General de ETL

12. CONCLUSIONES

En base a los análisis realizados en el presente estudio y a la experiencia práctica, resulta evidente que la consolidación y crecimiento de soluciones de información geoespacial dependen, en gran medida, del nivel de interoperabilidad que se logre con la diversidad de sistemas y tecnologías que dan soporte a los procesos e información de los distintos organismos públicos y privados.

Sobre la base de los diferentes formatos portables de información geográfica, es posible imaginar escenarios de interoperabilidad con sistemas heterogéneos de una amplísima gama de tecnologías, con diferente soporte de datos, desde bases de datos relacionales, bases de datos NoSQL, archivos, APIs, servicios, etc., dónde la interoperabilidad sea una realidad implementada de manera confiable y eficiente.

El paradigma de integración ETL cuenta con el respaldo de muchos años de madurez y evolución de las herramientas que lo soportan. Si bien los últimos años han perdido alguna relevancia frente a las integraciones orientadas a servicios o

microservicios, consideramos que aún sigue siendo una herramienta, valiosa, simple y económica para determinados ámbitos de interoperabilidad, dentro de los cuales encontramos la información geoespacial. En este estudio hemos intentado demostrarlo presentando ejemplos sobre escenarios hipotéticos simples, hacia modelos más complejos operativos y que extienden el horizonte de las soluciones IDE.

13. AGRADECIMIENTOS

Especial agradecimiento a Aldo Algorry, José Jachuf, Nicolas Oller integrantes del equipo de sistemas de IDECOR. Marcelo Pais, Denis Medel, Marco Muriel, Matías Scantamburlo, Federico Funes, Alexander Liamine y a todo el equipo de Holon Software | AMDG S.A.S.

14. REFERENCIAS

Documentación Apache Airflow (2024). https://airflow.apache.org/docs/

Repositorio Talend Spatial (2024). https://talend-spatial.github.io/

Sitio Web Airbyte (2024). https://airbyte.com/

Especificación de GeoJSON - RFC 7946 (2024). https://geojson.org/

Documentación Postgis (2024). https://postgis.net/documentation/

Sitio Web Geopandas (2024). https://geopandas.org/en/stable/

15. ANEXOS

A continuación, se adjuntan imágenes del desarrollo de algunos casos de uso en su respectiva aplicación.

15.1. Invocación a servicio web y almacenamiento en PostGIS (Airflow)

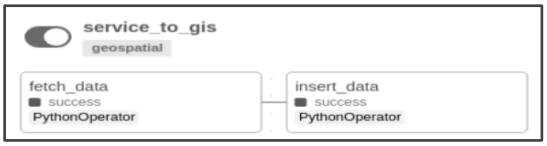


Figura 13. Anexo 15.1 diagrama

```
default_args = {
    'owner': 'IDECOR',
    'retries': 0,
    'retry_delay': timedelta(minutes=1),
}

def fetch_data():
    url = 'http://10.10.4.14:12001/localidades/'
    response = requests.get(url)
    response_json = response.json()
    print(response_json)
    return response_json

def insert_data_to_postgis(ti):
    resultado = ti.xcom_pull(task_ids='fetch_data')
    conn = psycopg2.connect(CONEXION)
    cursor = conn.cursor()
    for city in resultado:
        nombre = city['nombre']
        geometria = json.dumps(city['geometria'])
        cursor.execute(f"INSERT INTO localidad (nombre, geometria)
        VALUES (%s, ST_GeomFromGeoJSON(%s));", (nombre, geometria))
    conn.commit()
    cursor.close()
    conn.close()

with DAG(
    dag_id='service_to_gis',
    default_args=default_args,
    start_date=datetime(2024,3,27),
    schedule_interval='@daily',
    tags=['geospatial'],
    catchup=False
) as dag:

    fetch_data_task = PythonOperator(
        task_id='fetch_data',
        python_callable=fetch_data
    )
    insert_data_task = PythonOperator(
        task_id='insert_data',
        python_callable=insert_data_to_postgis
    )

    fetch_data_task >> insert_data_task
```

Figura 14. Anexo 15.1 código

15.2. Conversión de archivos “.shp” a “.geojson” (Airflow)

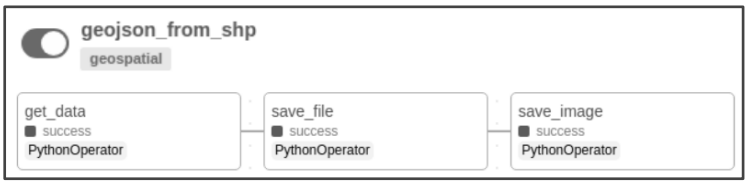


Figura 15. Anexo 15.2 diagrama

```python
def shp_file_to_json():
    ruta_archivo = airflow_home + '/exec/' + nombre_archivo + ".shp"
    gdf = gpd.read_file(ruta_archivo)
    return gdf.to_json()

def save_file_geojson(ti):
    geojson_file = airflow_home + "/exec/" + nombre_archivo + '.geojson'
    geojson_data = ti.xcom_pull(task_ids='get_data')
    with open(geojson_file, 'w') as f:
        f.write(geojson_data)

def save_image_map_geojson():
    image_file = airflow_home + "/exec/" + nombre_archivo + '.png'
    geojson_data = airflow_home + "/exec/" + nombre_archivo + '.geojson'
    gdf = gpd.read_file(geojson_data)
    fig, ax = plt.subplots(figsize=(10, 10))
    gdf.plot(ax=ax, color='red', linewidth=0)
    plt.savefig(image_file, bbox_inches='tight', pad_inches=0)

with DAG(
    'geojson_from_shp', default_args=default_args, tags=['geospatial'],
    schedule_interval='@daily', start_date = days_ago(1),
    params={
        "nombre_archivo": ""
    }
) as dag:

    get_data_from_file_task = PythonOperator(
        task_id="get_data",
        python_callable=shp_file_to_json
    )
    save_file_geojson_task = PythonOperator(
        task_id="save_file",
        python_callable=save_file_geojson
    )
    save_image_map_geojson_task = PythonOperator(
        task_id="save_image",
        python_callable=save_image_map_geojson
    )

    get_data_from_file_task >> save_file_geojson_task >> save_image_map_geojson_task
```

Figura 16. Anexo 15.2 código

15.3. Migración de registros masiva (Talend)

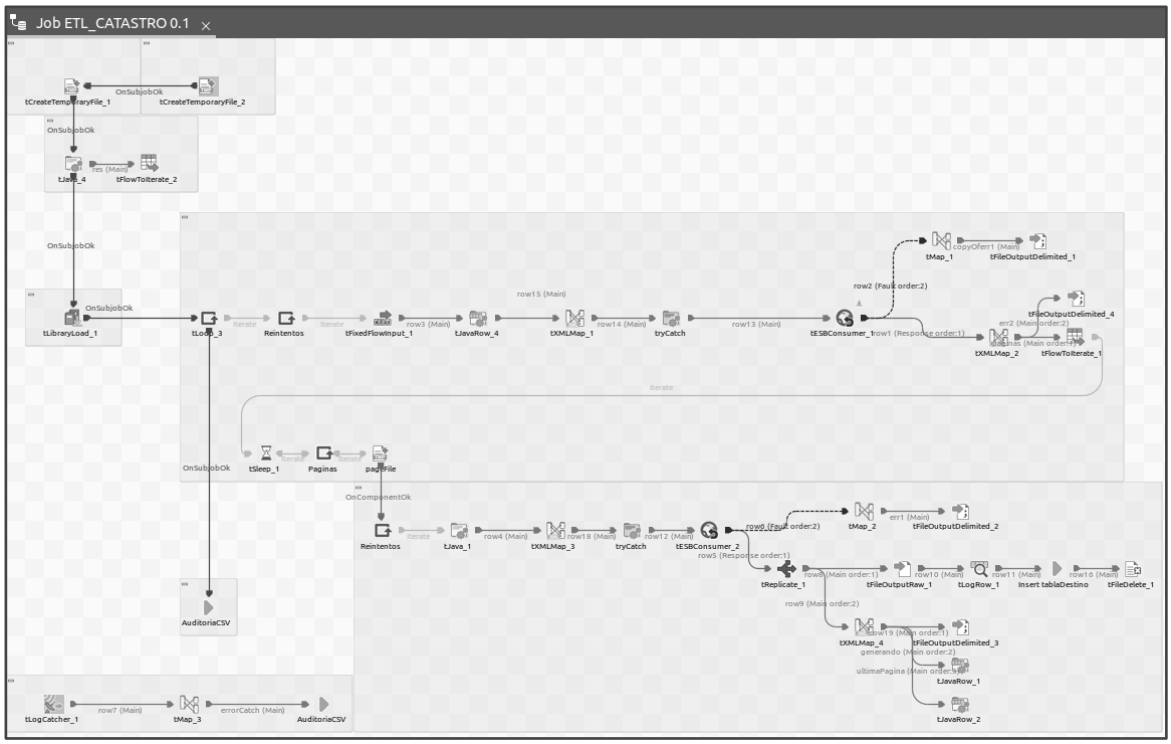


Figura 17. Anexo 15.3 job principal

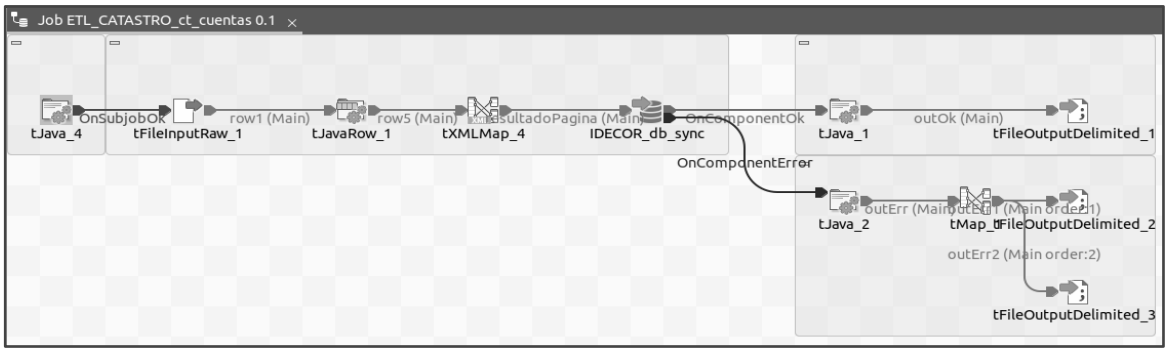
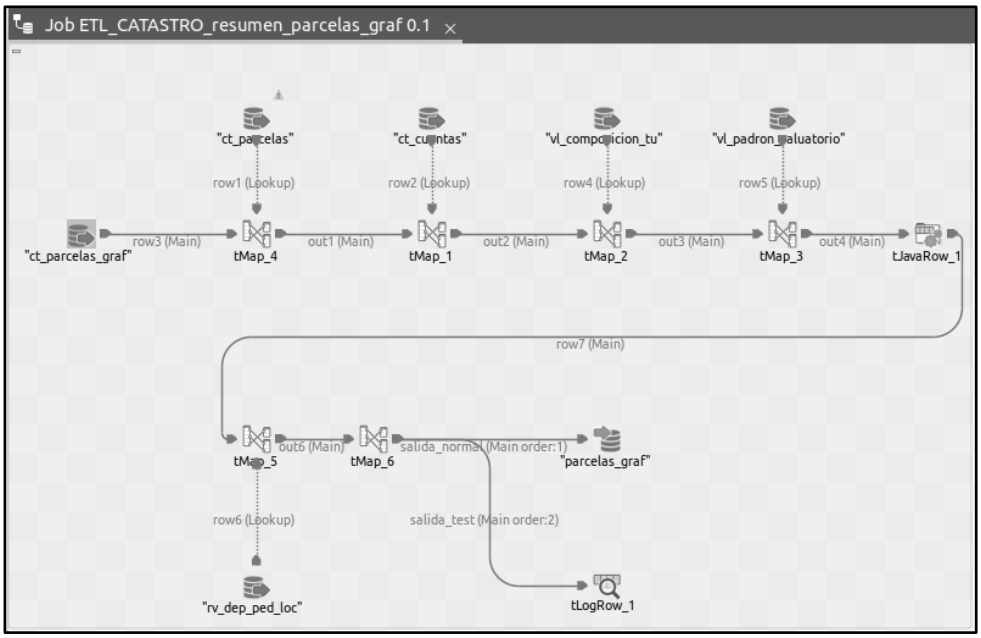


Figura 18. Anexo 15.3 job específico

15.4. Generación de nuevas tablas y vistas (Talend)



18. Anexo 15.4 job resumen