Engineer

ETL

🔄 mover datos entre sistemas y transformarlos por el camino

Qué es

ETL, abreviatura de extract, transform, load, consiste en copiar datos de los sistemas que los generan, darles la forma y el significado que espera otro sistema y escribirlos allí, normalmente en un almacén de datos. Una variante cercana, ELT, conserva los tres pasos, pero primero carga los datos en bruto y los transforma dentro del destino; muchos equipos escriben ETL/ELT para abarcar ambos.

Cada letra corresponde a una tarea distinta. Extraer consiste en leer de una fuente: una tabla entera, un archivo exportado, una API o solo las filas que cambiaron desde la última ejecución, identificadas mediante una marca de tiempo `updated_at` o mediante captura de datos modificados, CDC, que lee el registro de cada cambio de la propia base de datos. Transformar es limpiar y conciliar los datos: convertir texto en fechas y números, unificar unidades y zonas horarias, eliminar duplicados, consultar códigos en tablas de referencia y adaptar el resultado al modelo del destino, como tablas de hechos y dimensiones. El paso de carga escribe el resultado, mediante una inserción masiva en una tabla vacía o una combinación que actualiza las filas existentes y añade las nuevas.

El orden de los dos últimos pasos determina dónde se hace el trabajo. El ETL clásico transforma en un servidor aparte antes de que nada llegue al destino, algo razonable cuando el almacenamiento y el cómputo del almacén eran escasos y caros. ELT coloca los datos en bruto en tablas de preparación dentro del destino y ejecuta allí la transformación en SQL. La copia en bruto sigue disponible, así que, cuando cambia una regla, se puede repetir la transformación sobre el historial sin volver a extraerlo. El coste es que todo lo enviado por la fuente, incluidas columnas que nadie pidió, vive ahora en el almacén y debe someterse allí al gobierno de datos.

PostgreSQL cubre cada paso con herramientas propias. `COPY` mueve filas entre una tabla y un archivo en formato de texto, CSV o binario y es la vía más rápida de entrada. Leer un archivo en el servidor requiere privilegios de superusuario o el rol `pg_read_server_files`; la orden `\copy` de psql lee, en cambio, el archivo en el cliente. `INSERT ... ON CONFLICT` y `MERGE` integran filas nuevas con las existentes. La decodificación lógica convierte el registro de escritura anticipada en un flujo de cambios de filas para CDC, y un procedimiento PL/pgSQL puede mantener la transformación junto a los datos que modifica.

Utilice el concepto en una decisión

Aplique «¿Qué es ETL?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Las cifras llegan donde hacen falta según un calendario, sin que nadie tenga que copiar una hoja de cálculo al terminar el mes.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Un plan de infraestructura para 14 departamentos — 14 departamentos de investigación con un único plan de infraestructura

Un plan de infraestructura para 14 departamentos Póngase en contacto →

Por qué importa

La mayoría de las preguntas que merece la pena hacer a una empresa requieren datos de más de un sistema. Los pedidos están en una base, los pagos en una exportación del proveedor y los registros de clientes en un CRM. Hasta que algo los reúne en un lugar con un mismo conjunto de reglas, cada respuesta se prepara a mano, y el método cambia con la persona que lo hace. Eso es ETL, y un almacén solo es tan correcto como las cargas que lo llenan.

El paso de transformación también es donde una empresa deja por escrito qué significan sus datos. Qué filas cuentan como una venta, a qué día pertenece un pago de medianoche, qué registro prevalece cuando coinciden dos clientes: si se decide una vez en la carga, la respuesta se aplica a todos los informes posteriores. Si se deja a cada informe, las respuestas divergen y dos paneles muestran dos totales.

La forma de extraer determina cómo crece el trabajo de carga. Una recarga completa lo copia todo cada vez. Es sencilla y se corrige sola, porque el error de ayer se sobrescribe esta noche, pero su coste crece con el tamaño total de los datos. Una carga incremental solo mueve lo que cambió, así que su coste crece con el ritmo de cambio. Tiene que recordar dónde acabó la última ejecución,

detectar las filas eliminadas en origen y aceptar las que llegan tarde con marcas de tiempo antiguas. La mayoría de los sistemas empieza con recargas completas y cambia tabla por tabla cuando la noche deja de ser suficientemente larga.

El coste continuo es que las cargas son código que se ejecuta sin supervisión, normalmente de noche, contra fuentes que cambian sin avisar. Necesitan registros que indiquen qué se ejecutó y cuántas filas se movieron, alertas cuando se interrumpe una ejecución y, por encima de todo, una propiedad: repetir una ejecución con la misma entrada debería dejar el destino exactamente como una sola ejecución. Una carga así se puede reintentar tras cualquier fallo sin que nadie tenga que comprobar nada antes.

Cómo se aplica en la práctica

Unos flujos fragmentados retrasaban informes críticos para el negocio en varios entornos de desarrollo y producción de PostgreSQL, y se escribieron 600 flujos ETL/ELT en PL/pgSQL para sustituirlos: ETL a escala de cientos de cargas rutinarias. Las restricciones de clave primaria y foránea protegían la integridad referencial durante las transformaciones, los índices únicos y compuestos apoyaban las combinaciones y el tratamiento de excepciones con control transaccional revertía una ejecución si algo salía mal. Las cargas incrementales usaban CDC y deltas basados en marcas de tiempo; los registros de ejecución y la auditoría permitían el seguimiento y la depuración. El trabajo entregado comprende lógica de transformación, restricciones, carga incremental y registros de ejecución; el registro de ese trabajo no establece ninguna comparación de rendimiento ni cálculo de tasa de errores.

Una migración es ETL que se ejecuta una vez. Trasladar una aplicación cartográfica de Microsoft SQL Server a PostgreSQL con PostGIS supuso llevar más de 50 millones de registros espaciales por flujos ETL, cuyas transformaciones convertían cada coordenada al sistema de referencia EPSG:4326. Una vez cargados los registros, con índices GiST y SP-GiST y una configuración ajustada en el destino, las consultas espaciales que antes tardaban una media de 2,5 segundos respondían en menos de 250 milisegundos, diez veces más rápido, y la carga de CPU del backend bajó un 65 %.

A través de una frontera de red, seis flujos ETL/ELT muestran en acción cada letra del nombre. La extracción consistía en descargas programadas de archivos comprimidos desde un servidor accesible solo por una VPN IPsec, con scripts de Bash y Python que clasificaban los archivos Parquet, CSV y .bak de su interior por

tipo y disposición de columnas; las copias .bak se restauraban en una instancia local de MSSQL Server. Los datos se cargaban en esquemas de preparación de MSSQL y PostgreSQL con bcp, psql o SSIS, según el formato, se limpiaban, normalizaban y enriquecían con procedimientos PL/pgSQL y T-SQL, y después se comprimían con gzip y se llevaban por un bucket de Google Cloud Storage hasta Google BigQuery. Lo que a mano había tardado al menos tres horas pasó a ejecutarse de extremo a extremo en menos de 20 minutos, y la sincronización pasó de semanal a diaria.

Dónde surgen los problemas

Una ejecución reintentada carga dos veces las mismas filas. Un trabajo inserta con un `INSERT` simple, se detiene a mitad por un error de red y el planificador lo ejecuta otra vez desde el principio. Las filas que entraron la primera vez están ahora duplicadas y todos los totales basados en ellas son demasiado altos. Dé al destino una clave que identifique cada fila en términos del negocio e integre el lote con `INSERT ... ON CONFLICT DO UPDATE`, que PostgreSQL documenta como una inserción o actualización atómica incluso con mucha concurrencia, o con `MERGE`. Hay una salvedad: una sola sentencia `ON CONFLICT DO UPDATE` se niega a cambiar la misma fila dos veces y genera una violación de cardinalidad, por lo que un lote con claves repetidas debe reducirse primero a una fila por clave.

```
INSERT INTO meter_reading (meter_id, read_at, kwh)
SELECT DISTINCT ON (meter_id, read_at) meter_id, read_at, kwh
FROM staging_meter_reading
ORDER BY meter_id, read_at, received_at DESC
ON CONFLICT (meter_id, read_at)
DO UPDATE SET kwh = EXCLUDED.kwh;
```

Una carga masiva se detiene cerca del final y deja el disco más lleno que antes. `COPY FROM` escribe las filas en la tabla a medida que avanza. Cuando se detiene ante una fila incorrecta, las ya escritas quedan en estado eliminado, invisibles para las consultas pero aún en disco hasta que `VACUUM` recupera el espacio; un fallo cerca del final de un archivo grande puede desperdiciar mucho espacio. PostgreSQL 17 añadió `ON_ERROR ignore`, que omite filas de entrada de texto o CSV cuyos valores no se pueden convertir al tipo de su columna, y PostgreSQL 18 añadió `REJECT_LIMIT` para limitar cuántas puede omitir. Solo cubre la conversión de tipos: una fila que incumple una restricción sigue deteniendo la orden. Cargar primero en una tabla de preparación, comprobar allí los datos y trasladar las filas correctas a la tabla real en una transacción mantiene un archivo incorrecto lejos de los datos que lee la gente.

La captura de datos modificados llena el disco. La decodificación lógica lee los cambios mediante una ranura de replicación, y la ranura conserva toda la parte del registro de escritura anticipada que aún necesita su consumidor, esté conectado o no. Cuando el lector se detiene, el registro crece hasta llenar el disco y, en casos extremos, el servidor se apaga para evitar el desbordamiento cíclico de los identificadores de transacción. La documentación de PostgreSQL lo dice con claridad: una ranura que ya no se necesita debería eliminarse. Tras una caída, una ranura también puede enviar por segunda vez cambios recientes, de modo que el consumidor debe tolerar duplicados: de nuevo, la misma propiedad para repetir una ejecución.

El primer informe de la mañana va lento tras una carga grande. El planificador elige cómo ejecutar una consulta a partir de estadísticas de cada tabla, y una carga masiva cambia los datos sobre los que se basan. Autovacuum puede actualizar esas estadísticas por sí mismo, pero no siempre antes del primer informe. Ejecutar `ANALYZE` como último paso de la carga lo resuelve. Para cargas muy grandes en una tabla existente, la guía de PostgreSQL para llenar una base añade que quitar índices y claves foráneas y recrearlos después puede compensar, porque construirlos en conjunto es más rápido que actualizarlos fila por fila. A cambio, desaparece la comprobación de errores mientras no están, así que conviene pensárselo dos veces antes de quitar un índice único.

Lecturas adicionales

- COPY — La orden de carga masiva, sus formatos y opciones, y lo que deja una carga fallida.
- Llenar una base de datos — Las recomendaciones de PostgreSQL para cargas grandes, desde una sola transacción y `COPY` hasta reconstruir índices y ejecutar `ANALYZE` después.
- MERGE — Insertar, actualizar y eliminar en una sentencia, según cómo coincida cada fila de origen con el destino.
- INSERT — La cláusula `ON CONFLICT`, su inserción o actualización atómica y la regla que impide tocar dos veces la misma fila.
- Conceptos de decodificación lógica — Cómo se leen cambios del registro de escritura anticipada y por qué hay que quitar una ranura de replicación sin uso.



Glosario

Elija según sus restricciones reales

Traiga un conjunto de datos representativo, las preguntas que debe responder y el cuello de botella actual. Podemos acordar cómo medir la calidad, la capacidad de procesamiento y la recuperación.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es ETL?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/etl/