Tema 2: Ingestas

1. Ingestas

Procesamiento por Lotes (Batch Processing)

Consiste en **recopilar y procesar grandes conjuntos de datos en intervalos específicos**. Los datos se acumulan durante un periodo y se procesan juntos como un "*lote*".

Se utiliza para generar informes diarios, semanales o mensuales, procesamiento de transacciones bancarias al final del día, cálculos de nómina, ejecución de modelos de aprendizaje automático para análisis descriptivos y análisis de tendencias de ventas

Características:

- **Los datos deben ser contextualizados en un marco temporal** para ser tratados de forma conjunta.
- **Pueden ser almacenados temporalmente** en varias ocasiones, los datos durante el procesamiento.
- **Los procesos de tratamiento suelen estar planificados** o pueden desencadenarse por algún evento.
- Las operaciones y análisis deben adaptarse a las dependencias de los datos de negocio que la ventana de ejecución de los procesos o viceversa.

Ventajas:

- **Eficiencia:** en el procesamiento de grandes volúmenes de datos es muy eficiente, ya que los datos se procesan en grupos.
- **Costo-efectividad:** más económico que el procesamiento en tiempo real, no requiere ninguna infraestructura en tiempo real.
- **Simplicidad:** simples en el diseño, implantación y mantenimiento comparado con los de tiempo real.

Desventajas:

- **Latencia alta:** No apto para necesidades inmediatas.
- **Falta de inmediatez:** Los resultados no están disponibles de inmediato. No apto para hacerlo en tiempo real.
- **Menos flexible:** los cambios en el proceso de procesamiento pueden requerir reprocesar todo el lote de datos.

Procesamiento en Tiempo Real

El Big Data, puede estar en lo que se denomina Streaming, **entre el "Soft" y el "Near" Real Time** (Véase tabla). El "Soft" es **rápido, pero puede tolerar algún retraso**.

Se utiliza para el monitoreo de sistemas, detección de fraude, aplicaciones de streaming, juegos en línea y sistemas de control industrial

Características:

- Es necesario **gestionar eventos de forma individual, tan pronto como se generan**.
- Los datos asociados a los eventos **son gestionados en memoria**, con una persistencia mínima o nula.
- El procesamiento de eventos **se realiza de forma continua e ininterrumpida**, con la mínima demora.
- La captura, transformación y análisis del dato conforman un flujo continuo desde los orígenes de los eventos hasta las aplicaciones consumidoras finales.
- Un tratamiento intensivo del dato impactará en el tiempo de entrega.

Ventajas:

- **Inmediatez:** los resultados están disponibles de inmediato, permite una toma de decisiones más rápida.
- **Mejor experiencia del usuario:** proporciona una experiencia más interactiva y atractiva para los usuarios.

- **Mayor flexibilidad:** permite adaptar el procesamiento de datos a las condiciones cambiantes.

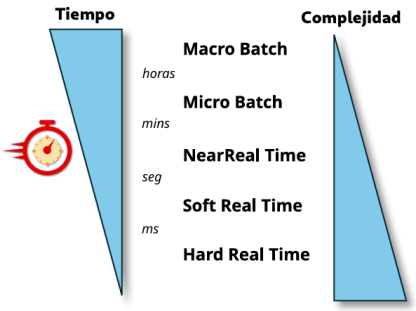
Desventajas:

- **Complejidad:** estos sistemas son más complejos de diseñar e implementar. Entre las más famosas están la arquitectura Lambda y Kappa.
- **Costo:** Mayor consumo de recursos y necesidades de mantenimiento. Requiere una infraestructura de procesamiento continuo.
- **Escalabilidad:** difícil de escalar dichos sistemas para manejar grandes volúmenes de datos

Tipo	Tiempo	Críticidad	Ejemplo
Hard Real-Time	Inmediato, estrictamente dentro del límite.	Fallo crítico si no se cumple.	**Control de vuelo, sistemas médicos.**
Soft Real-Time	Rápido, pero puede tolerar algunos retrasos.	No catastrófico, afecta el rendimiento.	**Streaming de video, videojuegos online.**
Near Real-Time	Segundos o minutos de retraso.	No crítico, mantiene actualización rápida.	**Monitoreo de redes, paneles de control.**

Si la forma de trabajar de la organización es mediante “microbatchs”, o sea, procesos que se lanzan cada 15 minutos. Una infraestructura de Real Time resultaría muy caro y complejo.

Latencia vs Complejidad



Patrones de captura o de dónde se captura la información

Petición-respuesta

Este es el patrón más usado. Se realiza la **conexión a una base de datos y se obtienen los resultados en formato JSON o en XML**. La comunicación es **síncrona**, un cliente **envía una solicitud y espera una respuesta** del servidor. Sencillo, pero nada práctico para grandes cantidades de datos.

Ventajas

- Simplicidad en la implementación y comprensión
- Facilidad para manejar errores y excepciones

Desventajas

- No escalable para grandes volúmenes de datos
- Puede generar latencia y cuellos de botella en sistemas con alta concurrencia

One way

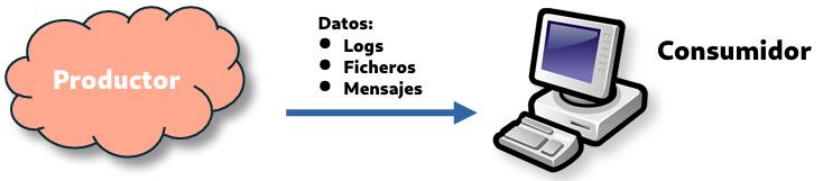
Comunicación **asíncrona** donde un cliente envía un mensaje **sin esperar una respuesta**.

Ventajas

- Mayor rendimiento y escalabilidad
- Desacoplamiento entre productor y consumidor

Desventajas

- Dificultades para garantizar la entrega y el procesamiento
- Menor control sobre el flujo de datos y manejo de errores



Publicador-Suscriptor

Los productores **publican mensajes en canales o temas**, y los consumidores **se suscriben** para recibirlos. Basado en sistemas de mensajería como **Apache Kafka**. Soporta procesamiento **tanto batch como en tiempo real.**

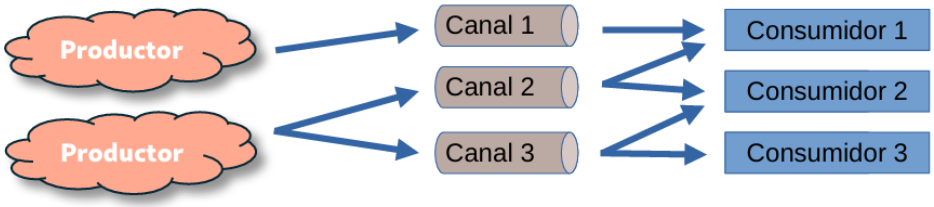
Las colas de mensajes son **Real Time**. Cada generador está generando **datos** (eventos).

Ventajas

- Alta escalabilidad y tolerancia a fallos
- Desacoplamiento temporal entre productores y consumidores

Desventajas

- Complejidad en la configuración y mantenimiento
- Requiere gestionar la consistencia y el orden de los mensajes



Steam processing

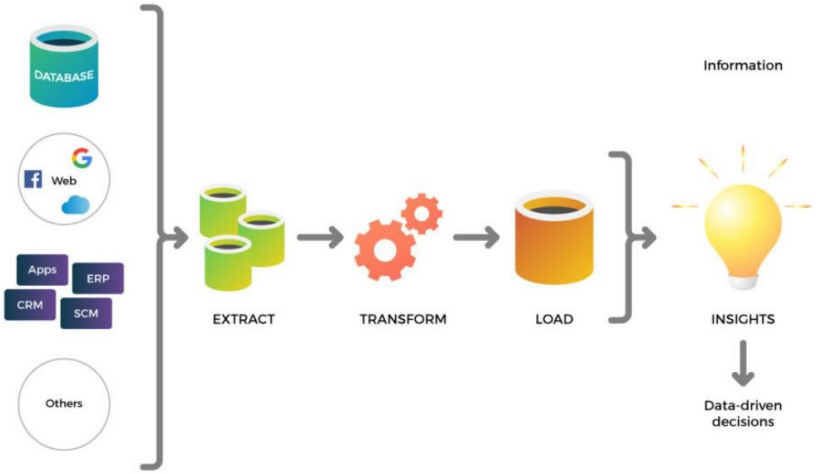
Ha salido hace poco el **procesamiento continuo de datos** con Apache Flink como herramienta destacada (Real Time).

Este **recibe un conjunto de datos y los procesa en Tiempo Real.**

2. Procesamiento Lotes

Extracción, Transformación y Carga

El **ETL** es un proceso fundamental en la gestión integral de datos, que se utiliza para **recopilar, procesar y cargar datos desde diferentes fuentes a un destino final**, como un almacén de datos o un **data lake**.



Extracción

Las estrategias de extracción se refieren a los **métodos utilizados para recopilar datos de diferentes fuentes durante la etapa inicial del ETL**.
El objetivo principal es **obtener los datos necesarios para su posterior análisis y procesamiento**.

Estrategias de extracción principales:

- **Extracciones Totales:** consiste en **extraer todos los datos de la fuente en cada ejecución del proceso ETL**. Es un **enfoque simple**, pero puede ser **ineficiente si el volumen de datos es grande**, ya que implica procesar todos los datos, incluso aquellos que no han cambiado desde la última extracción.
- **Extracciones Diferenciales:** solo extraen los datos que han sido **modificados o añadidos desde la última extracción**. Esta estrategia es **más eficiente**, ya que **solo se procesan los datos nuevos o actualizados**, lo que **reduce el tiempo de procesamiento y el consumo de recursos**.

Elección de la estrategia de extracción adecuada

- **Frecuencia de actualización:** Si los datos se actualizan con frecuencia, las extracciones diferenciales son más adecuadas, ya que solo procesan los cambios.
- **Complejidad del sistema:** Si el sistema es complejo, las extracciones diferenciales pueden ser más difíciles de implementar.
- **Volumen de datos:** Si el volumen de datos es pequeño, las extracciones totales pueden ser una opción viable. Si el volumen de datos es grande, las extracciones diferenciales son más eficientes.

Una estrategia de extracción inadecuada puede resultar en:

- **Ineficiencia en el procesamiento:** Si se extraen más datos de los necesarios, se desperdician recursos y tiempo.
- **Inconsistencias en los datos:** Si la estrategia no captura todos los cambios, los datos pueden ser inconsistentes.
- **Mayor complejidad en la implementación:** Algunas estrategias de extracción diferencial pueden ser más complejas de implementar que las extracciones totales.

Inventariado de datos

El inventario de datos consiste en **identificar y documentar todas las fuentes de datos que se utilizarán en el proceso ETL**. Esto incluye bases de datos, archivos planos, APIs, páginas web, etc.

El objetivo es tener un **control exhaustivo de las fuentes de datos y poder detectar posibles problemas**.

Beneficios:

- **Mejor comprensión de los datos:** proporciona una visión completa de los datos, incluyendo su origen, formato, calidad y contenido.
- **Detección temprana de problemas:** como valores faltantes, duplicados o inconsistencias, antes de que afecten las etapas posteriores del proceso ETL.

Perfilado de datos

El perfilado de datos (data profiling) es un proceso que se utiliza para **analizar y auditar la estructura y la calidad de los datos**.

Permite **obtener estadísticas, identificar valores anómalos, inconsistencias en la codificación, o posibles relaciones entre valores de diferentes orígenes**.

Esta información es fundamental para poder **realizar acciones correctoras en la etapa de transformación**.

Beneficios:

- **Mejor comprensión de los datos:** proporciona una visión completa de los datos, incluyendo su origen, formato, calidad y contenido.
- **Toma de decisiones informadas:** permite tomar decisiones más informadas sobre cómo limpiar, transformar y cargar los datos.
- **Mejora de la calidad de los datos:** ayuda a identificar y corregir problemas de calidad, lo que mejora la calidad general de los datos.
- **Reducción de costes:** La detección temprana de problemas en los datos puede ayudar a reducir costes asociados a la limpieza y corrección de datos en etapas posteriores.

Proceso de perfilado de datos:

- **Análisis de la estructura de la tabla:** Determinar el número de columnas, sus nombres y tipos de datos.
- **Análisis del contenido de las columnas:** Obtener estadísticas como la media, la mediana, el valor mínimo y máximo, la frecuencia de valores nulos, etc.
- **Identificación de valores atípicos:** Detectar valores que se desvían significativamente de la norma.
- **Análisis de la consistencia de los datos:** Verificar si los datos cumplen con las reglas de negocio y las restricciones definidas.

Pasos:

- **Acceso a la fuente de datos:** ya sea una tabla de origen, una API o un archivo.
- **Inventario de los orígenes de datos:** para tener un control de las fuentes y detectar posibles problemas.
- **Filtrado de datos:** para seleccionar solo los datos necesarios.
- **Agregación de datos:** para combinar datos de diferentes fuentes.
- **División de datos:** para separar los datos en diferentes conjuntos.

Aspectos para seguir **antes de implementar la etapa de extracción**:

- **Variedad de los Orígenes de Datos:** Considerar que los datos pueden originarse de diversas fuentes y en una variedad de formatos. Requiere tecnologías y técnicas de extracción específicas para cada caso.

  Ejemplos:

  - **Bases de datos relacionales**: Son el tipo de base de datos más común y almacenan los datos en tablas con filas y columnas.
  - **Archivos planos**: Son archivos de texto sin formato que almacenan los datos en un formato delimitado, como comas o tabulaciones.
  - **APIs** (Interfaces de Programación de Aplicaciones): Permiten acceder a datos de aplicaciones de terceros, como redes sociales, servicios web, etc.

- **No Interferir con los Sistemas Operacionales:** La extracción de datos **no debe afectar negativamente el funcionamiento de los sistemas de origen**. Si se extraen datos de un sistema de producción, es importante asegurarse de que la extracción no afecte el rendimiento del sistema.
- **Frecuencia de las Extracciones:** dependerá de las necesidades de la aplicación. Si se necesita información actualizada diariamente, las extracciones serán diarias. Si se necesita mensualmente, las extracciones serán mensuales.

- **Recuperación de los Datos Extraídos:** Existe la posibilidad de que alguna etapa posterior del proceso ETL falle. Es fundamental garantizar la posibilidad de recuperar los datos extraídos en caso de que esto ocurra. Se logra almacenando los datos extraídos en un almacenamiento intermedio.
- **Creación de Metadatos:** Los metadatos son **datos que describen otros datos**, como el origen, la fecha de extracción, el formato, etc. La creación de metadatos es **fundamental para el seguimiento del origen de los datos y para garantizar su calidad**.

Transformación

Este proceso busca **preparar los datos para su uso efectivo en análisis y toma de decisiones**. Hace que los datos sean **confiables, consistentes y listos para su análisis**, permitiendo obtener información valiosa para la toma de decisiones

Esta acción sería mínima o inexistente en una **data lake**, es consistente debido a que se guardan los datos en crudo (antes de ser procesados, sin ordenar).

En un **data warehouse** estaría poblada de actividades, hay que hacerlo de forma conformada y estandarizada (datos estructurados y procesados para su análisis).

En un **data lake**, la transformación **empieza una vez los datos están en el repositorio**. El objetivo **es acondicionarlos para su consumo**, medie o no un punto de almacenaje.

Etapas:

- **Definición:** La transformación implica **convertir los datos extraídos de diversas fuentes a un formato compatible** con el modelo de datos destino.
- **Objetivo:** Asegurar que los datos sean **consistentes, precisos y estén estructurados** de manera que se **ajusten a las necesidades del sistema de destino**, como un **data lake** o **data warehouse**.

Tareas principales:

- **Limpieza:** Eliminar datos duplicados, gestionar valores erróneos u omitidos, y estandarizar valores para garantizar la calidad de los datos.
- **Normalización:** Ajustar los datos a un formato estándar para facilitar su análisis y comparación.
- **Combinación y Agregación:** Unir datos de diferentes fuentes y agregarlos para obtener información resumida.
- **Filtrado:** Seleccionar solo los datos relevantes para el análisis, descartando los que no son necesarios.
- **Cálculo y Derivación:** Realizar cálculos y derivar nuevas variables a partir de los datos existentes.
- **Conformación:** Adaptar los datos a la estructura del modelo de datos de destino.
- **Formateado:** Dar a los datos un formato final adecuado para su consumo.

Otras características:

- **Complejidad:** Varía como el **tamaño de los datos, la calidad inicial de los datos y las necesidades del sistema de destino**. Los **data lakes** suelen requerir una transformación mínima inicial. Los **data warehouses** necesitan una transformación más completa.
- **Ubicación:** La transformación se realiza **después de la extracción de los datos y antes de la carga** en el sistema de destino.
- **Implementación:** La transformación se hacer utilizando diversas tecnologías, desde Python hasta Spark.

Carga

Es la etapa final del proceso ETL. Una vez que los datos han sido extraídos de sus fuentes originales y transformados a un formato adecuado, la fase de carga **se encarga de insertarlos en el sistema de destino**. Este destino **puede ser un simple archivo, una base de datos transaccional, un data warehouse o un data lake**.

Tipos de carga:

- **Carga completa (full load):** Se purgan **todos los datos existentes en el destino y se carga el conjunto completo de datos**. Este método es sencillo, pero poco eficiente si solo se han modificado algunos datos.
- **Carga incremental:** Solo se cargan los **datos nuevos o modificados desde la última carga**. Esto mejora la eficiencia y reduce el tiempo de carga.
    - **Programación de la carga:** depende de los requisitos del proyecto. Puede ser por hora, diaria, semanal, mensual o anual.
    - **Impacto en el rendimiento:** Una carga bien diseñada minimiza el impacto en el rendimiento del sistema de destino, especialmente en el caso de **data warehouses**, donde el volumen de datos suele ser muy grande.
    - **CDC (Change Data Capture):** Esta técnica se utiliza para **identificar los cambios realizados en los datos y solo cargar las modificaciones**, optimiza aún más la carga incremental.

Consideraciones Adicionales:

- La fase de carga debe tener en cuenta la conectividad del sistema de destino. Si la conectividad es limitada, puede ser necesario implementar mecanismos para asegurar la integridad de los datos durante la carga.

- El diseño del data warehouse influye en la complejidad de la carga. En data warehouses con estructuras dimensionales, la carga puede requerir actualizaciones en varias tablas (tabla de hechos y tablas de dimensiones).
- La elección entre carga completa e incremental se basa en la frecuencia de las actualizaciones, el volumen de datos y el impacto en el rendimiento del sistema de destino.
- El diseño e implementación de la fase de carga deben considerar las necesidades del proyecto, el tipo de destino y el rendimiento general del sistema.
- La gestión de los metadatos. Toda área intermedia debería contar con un repositorio de metadatos dirigido a los ingenieros de datos, con el fin de soportar tareas de monitorización, análisis comparativo y auditoría.

Data Replication: Mediante un modelo de publicación-subscripción, permite tener en sincronía estructuras de datos que residen en repositorios diferentes y heterogéneos, de manera que se mantengan consistentes en tiempo real. Minimizando el impacto en los sistemas de origen.

Aspectos importantes de la fase de carga:

- El **lugar** donde se cargan los datos depende de las necesidades del proyecto pudiendo ser:
    - **Archivo:** sencillo de almacenar datos, pero con limitaciones en términos de análisis.
    - **Base de datos transaccional:** ideal para datos que necesitan ser actualizados con frecuencia, como los datos de transacciones comerciales.
    - **Data warehouse:** diseñado para análisis y almacenamiento de grandes volúmenes de datos históricos.
    - **Data lake:** un repositorio centralizado que almacena datos en su formato raw, permitiendo flexibilidad en su posterior uso.