Building Medallion Architectures - Designing with Delta Lake and Spark

Resumen del capítulo 2

Piethein Strengholt

November 22, 2025

Capítulo 2. Sentando las Bases

Antes de construir una casa, se necesita una base sólida. La misma idea se aplica a la arquitectura Medallion. Este capítulo sirve como un puente preparatorio, introduciendo componentes y patrones clave que son recurrentes en las arquitecturas Medallion y sienta las bases para el Capítulo 3, donde se profundizará en la arquitectura Medallion y sus capas.

En este capítulo, se cubren varias áreas centrales:

- **Zonas de aterrizaje (landing zones) extra:** Áreas preliminares donde se ingieren los datos crudos antes de llegar a la arquitectura Medallion.
- **Datos crudos:** Los datos no procesados recolectados de diversas fuentes.
- **Procesamiento por lotes (batch):** Método de procesamiento de datos donde se recolectan, procesan y emiten en lotes a intervalos programados.
- **Procesamiento de datos en tiempo real:** Implica procesar los datos tan pronto como están disponibles.
- **Herramientas de ETL y orquestación:** Integrales para extraer, transformar, cargar y automatizar flujos de trabajo.

También se explorará la gestión de tablas Delta, que es crucial para mantener la integridad y eficiencia a través de las diferentes capas Medallion.

Precondiciones Fundamentales

Antes de diseñar cualquier arquitectura, se necesitan datos. El primer paso es siempre identificar los sistemas fuente objetivo y determinar la mejor manera de obtener los datos de ellos. En este contexto, surgen dos decisiones clave:

1. ¿Es necesario utilizar zonas de aterrizaje intermedias para la ingesta de datos?

2. ¿Qué método de ingesta se debe utilizar: por lotes o en tiempo real?

Estas decisiones influirán en las herramientas y técnicas que se elijan para la integración de datos, la orquestación y la gestión de tablas.

Zonas de Aterrizaje Adicionales

Una "zona de aterrizaje" (*landing zone*) se utiliza a menudo como un área preliminar donde se ingieren los datos crudos antes de que lleguen a la capa Bronce de la arquitectura Medallion. La necesidad de esta zona depende de las características de la fuente de datos.

Por ejemplo, al tratar con servicios externos o proveedores de SaaS, puede ser necesaria una zona de aterrizaje segura para almacenar inicialmente los datos. Del mismo modo, algunos equipos de aplicaciones pueden requerir sus propias zonas de aterrizaje dedicadas para gestionar su proceso de extracción.

Por otro lado, la ingesta directa en la capa Bronce es factible si el sistema fuente tiene medidas de seguridad robustas y técnicas de ingesta estables.

Conclusión sobre las Zonas de Aterrizaje Aunque la capa Bronce sirve principalmente como área de recolección de datos, es posible que se necesite una zona de aterrizaje preliminar adicional en entornos más grandes o complejos. La decisión de cómo etiquetar esta capa (por ejemplo, "capa de estaño", "pre-Bronce", "staging") y si se incluye dentro del diseño de la capa Bronce depende de la organización. Lo importante es ingerir y almacenar los datos de una manera que cumpla con los requisitos funcionales y no funcionales específicos de la organización.

Datos Crudos

Al hablar de recolección de datos para la capa Bronce, es necesario tener cuidado con términos como "crudo" o "tal cual" (*as-is*). En sistemas transaccionales complejos con miles de tablas, un ingeniero puede necesitar mediar durante la extracción.

Por ejemplo, un sistema bancario central como Temenos T24 almacena todos sus datos en formato XML en tablas con solo dos columnas. Extraer estos datos directamente resultaría en un conjunto de datos inmanejable. Se requiere una herramienta mediadora (como un extractor de datos) para transformar los datos en una estructura más manejable *antes* de que lleguen a la primera capa.

Conclusión sobre los Datos Crudos El preprocesamiento de datos complejos o la aplicación de mediación ocurre antes de que los datos lleguen a la primera capa. Esto no se considera una transformación dentro de la arquitectura Medallion. Por lo tanto, aunque los datos en la primera capa son típicamente "crudos", el preprocesamiento puede ser crucial para la manejabilidad en casos de sistemas complejos.

Procesamiento por Lotes (Batch)

El procesamiento por lotes es un método en el que los datos se recolectan durante un período y luego se procesan todos a la vez. Sigue siendo popular por varias razones: es rentable, permite procesar grandes conjuntos de datos en una sola operación y muchos sistemas existentes lo soportan ampliamente.

Al configurar la ingesta de datos por lotes en la capa Bronce, se deben tener en cuenta algunas consideraciones clave:

- **Mantener la integridad de los datos:** Utilizar métodos de validación como recuentos de filas, sumas de verificación (*checksums*) o totales hash.
- **Procesamiento a intervalos:** A diferencia del procesamiento continuo, el procesamiento por lotes ocurre a intervalos establecidos. Estos deben seleccionarse cuidadosamente.
- **Manejo de errores:** Es esencial establecer mecanismos robustos para el manejo de errores, como una "capa de error" u "orfanato de datos" para monitorear y gestionar problemas.
- **Selección de herramientas:** Elegir las herramientas adecuadas para la extracción e ingesta de datos es crucial.

Conclusión sobre el Procesamiento por Lotes Los desafíos de la recolección de datos no son nuevos. Muchas organizaciones reutilizan interfaces existentes para importar datos a sus *lakehouses*. Aunque el procesamiento por lotes puede parecer sencillo, a menudo sigue siendo una tarea compleja. Las arquitecturas de *lakehouse* modernas destacan por su versatilidad para gestionar eficientemente tanto la ingesta por lotes como en tiempo real.

Procesamiento de Datos en Tiempo Real

Las arquitecturas *lakehouse* modernas manejan eficientemente datos en *streaming* o en tiempo real. Al procesar los datos en el momento en que se generan, se habilitan *insights* inmediatos y decisiones oportunas, lo cual es crucial para aplicaciones como la detección de fraudes o interacciones personalizadas con el cliente.

Para configurar la ingesta de datos en tiempo real, generalmente se necesita agregar

componentes adicionales o modificar la configuración existente, ya que la mayoría de las aplicaciones no generan eventos automáticamente.

Advertencia En los movimientos de datos, es esencial diferenciar entre eventos que transportan estado (*state-carrying*) y notificaciones simples. Los primeros proporcionan una instantánea del estado de una aplicación en un momento específico, mientras que los segundos son alertas básicas que informan sobre ocurrencias y generalmente no incluyen información de estado detallada.

Spark Structured Streaming

El soporte robusto para **Structured Streaming** dentro del motor de Apache Spark mejora enormemente el procesamiento de datos en tiempo real. Permite el procesamiento continuo de flujos de datos de diversas fuentes (archivos de registro, dispositivos IoT, Kafka). En esta etapa, las transformaciones son críticas para convertir los datos crudos en un formato más utilizable (por ejemplo, aplanar JSON anidado, extraer campos de XML). Una vez transformados, los datos pueden ser escritos en diversos destinos para su análisis posterior.

Nota La ingesta en tiempo real con Spark requiere un clúster de cómputo "siempre encendido", lo que puede resultar en costos más altos. Si la latencia no es una preocupación, se puede considerar la activación de cargas de trabajo a intervalos (por ejemplo, cada cinco minutos) para reducir el costo.

Change Data Feed

Otro patrón en el procesamiento de datos en *streaming* es el **change data feed**. Esta característica permite capturar los cambios en las tablas Delta a medida que ocurren, permitiendo procesarlos en tiempo (casi) real. La combinación de *change data feed* y Spark Structured Streaming proporciona un mecanismo poderoso para el procesamiento de datos en tiempo real, acelerando y simplificando las operaciones de ETL/ELT al procesar solo los cambios.

Change Data Capture (CDC)

Change Data Capture (CDC) es una técnica utilizada para identificar y capturar los cambios realizados en los datos en tiempo real. Las herramientas de CDC monitorean activamente las modificaciones de la base de datos y las registran a medida que ocurren, permitiendo la replicación de datos a otros sistemas.

Consideraciones y Recursos de Aprendizaje

Spark Structured Streaming, *change data feed* y CDC son componentes integrales que permiten potentes *pipelines* de integración y análisis de datos en tiempo real. A continuación se muestra una comparación detallada de cómo cada componente contribuye a estos procesos.

Opción	Descripción
Spark Structured Streaming	Soporta operaciones complejas y se integra con diversas fuentes y sumideros; maneja el procesamiento continuo de datos.
Change data feed	Captura cambios en tablas Delta en tiempo real; permite el procesamiento eficiente de datos al transmitir solo los incrementos y la diferencia entre estados (también conocidos como deltas).
CDC	Rastrea y captura cambios en los registros de transacciones de la base de datos de manera efectiva, permitiendo la replicación e integración de datos en tiempo real.

Tabla 2-1: *Resumen de las opciones de procesamiento de datos en tiempo real*

Al diseñar la capa Bronce, la cuestión de si los datos replicados en tiempo real pertenecen a esta capa depende de las necesidades específicas. Si el objetivo principal es apilar extracciones completas de datos, los datos replicados podrían clasificarse mejor como parte de un área intermedia o de aterrizaje. Por el contrario, si el objetivo es mantener una capa Bronce consultable que refleje los datos de origen "tal cual", entonces los datos replicados en tiempo real pueden considerarse parte de la capa Bronce.

Conclusión sobre la Ingesta de Datos en Tiempo Real y Streaming La decisión de implementar la ingesta o replicación de datos en tiempo real, y de clasificar estos datos dentro de la capa Bronce o un área intermedia, está fundamentalmente impulsada por requisitos de uso y objetivos estratégicos distintos. Es importante entender que con la ingesta de datos en tiempo real, las actualizaciones a menudo ocurren en múltiples capas simultáneamente.

Herramientas de ETL y Orquestación

La elección de herramientas para extraer, transformar, cargar y orquestar es crucial, ya que puede influir significativamente en el diseño del *pipeline* de datos. Algunas herramientas populares disponibles son:

- **Apache Airflow:** Plataforma de código abierto para crear, programar y monitorear flujos de trabajo de forma programática.

- **Azure Data Factory:** Ampliamente utilizado en organizaciones que han adoptado el ecosistema de Microsoft.

- **Databricks Auto Loader:** Diseñado para el ecosistema de Databricks, destaca en el procesamiento incremental de nuevos archivos y el manejo de la evolución del esquema.

- **Databricks LakeFlow Connect:** Servicio específico de Databricks con conectores incorporados.

- **Databricks Workflows:** Servicio de orquestación gestionado como parte de Databricks.

- **Herramientas de terceros:** Fivetran, Qlik, StreamSets, Syncsort, Informatica y Stitch son opciones populares que ofrecen conectores extensos y capacidades de orquestación.

Gestión de Tablas Delta

La gestión de tablas es una preocupación en todas las capas, considerando su efecto en el rendimiento, la usabilidad y los costos. A continuación se presentan técnicas recomendadas para mejorar la gestión de los datos.

Z-Ordering

Es una técnica que mejora la eficiencia de la recuperación de datos al agrupar información relacionada dentro de los mismos archivos. Mejora la localidad, reduce las operaciones de E/S y es particularmente eficiente en escenarios que requieren filtrado y uniones (*joins*) eficientes. Se implementa usando la cláusula `ZORDER BY` en el comando `OPTIMIZE`.

V-Ordering

Es una optimización en tiempo de escritura para el formato de archivo Parquet, utilizada en Microsoft Fabric. Organiza lógicamente los datos basándose en el mismo algoritmo de almacenamiento que el motor VertiPaq de Power BI, lo que permite lecturas más rápidas. Puede mejorar los tiempos de lectura en un promedio del 10% (hasta un 50% en algunos escenarios), pero puede impactar los tiempos de escritura.

Particionamiento de Tablas

Es una estrategia efectiva para gestionar grandes conjuntos de datos. Implica dividir una tabla grande en segmentos más pequeños y manejables, lo que mejora significativamente el rendimiento de las consultas al reducir el volumen de datos que se necesita leer. En Delta Lake, se puede particionar una tabla por una columna específica (comúnmente una fecha) usando la cláusula `PARTITIONED BY`.

Liquid Clustering

Introducido a mediados de 2024, el *liquid clustering* es una nueva característica de Delta Lake diseñada para reemplazar el Z-ordering y el particionamiento tradicionales. Simplifica las decisiones de diseño de datos y aumenta significativamente el rendimiento de las consultas al optimizar automáticamente la disposición de los datos de forma dinámica según las claves de agrupamiento y los patrones de consulta en evolución.

Compactación y Escrituras Optimizadas

Delta Lake ofrece la operación `OPTIMIZE` para abordar el problema de los archivos pequeños, compactándolos en archivos más grandes y manejables para mejorar el rendimiento de las consultas. Además, se pueden activar las **escrituras optimizadas** (`delta.autoOptimize.optimizeWrite`) para que Delta Lake optimice automáticamente la disposición de los archivos de datos durante las escrituras.

DeltaLog

El registro de transacciones de Delta, o **DeltaLog**, es una característica clave de Delta Lake que registra cada cambio realizado en una tabla Delta. Mantiene un historial completo de la tabla como una serie de archivos JSON. Esto permite una gestión de datos robusta y la capacidad de auditar y revertir cambios. Para la arquitectura Medallion, el DeltaLog proporciona una medida de seguridad robusta, permitiendo revertir rápidamente a una versión anterior de una tabla usando el comando `RESTORE`. Sin embargo, su propósito principal es la recuperación de datos y la auditoría, no servir como una base de datos histórica convencional.

Conclusión sobre la Gestión de Tablas Para un rendimiento óptimo, alinee los métodos de ordenamiento y agrupamiento con las capas específicas de su arquitectura y sus patrones de acceso. Además, establecer umbrales de retención bien pensados para las tablas Delta es fundamental para equilibrar la frescura de los datos con los costos de almacenamiento.

Conclusión

Sentar las bases para la arquitectura Medallion es un proceso meticuloso que comienza con una comprensión integral de las precondiciones iniciales, como la identificación de los sistemas fuente y la determinación de los métodos más eficientes para la exportación e ingesta de datos. Además, la selección de herramientas de ETL y orquestación es un factor crítico que puede dictar el diseño de las capas iniciales e influir en la arquitectura general. Finalmente, la gestión de tablas y el enfoque del modelado de datos están estrechamente vinculados y son vitales para optimizar el rendimiento. Con estos aspectos fundamentales cubiertos, se ha preparado el escenario para construir una arquitectura Medallion.