Building Medallion Architectures - Designing with Delta Lake and Spark

Resumen del capítulo 1

Piethein Strengholt

November 23, 2025

Capítulo 1. La Evolución de la Arquitectura de Datos

Crear una arquitectura de datos robusta es uno de los aspectos más desafiantes de la gestión de datos. A pesar de las diferencias entre organizaciones, toda arquitectura de datos comparte componentes fundamentales. El autor utiliza la metáfora de un diseño de arquitectura de tres capas:

1. **Proveedores de datos:** Las diversas fuentes de las que se extraen los datos.

2. **Plataforma de distribución:** El conjunto de herramientas y tecnologías para integrar y procesar los datos.

3. **Consumidores de datos:** Quienes utilizan los datos a través de servicios de BI, machine learning, etc.

Además, una capa superior de **metadatos y gobernanza** es crucial para gestionar toda la arquitectura.

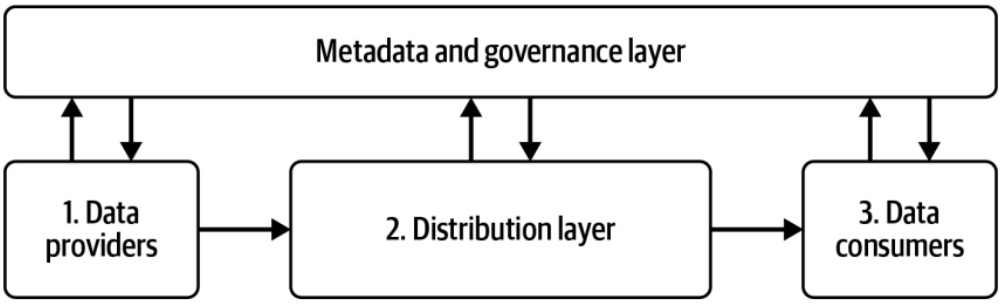

Figura 1-1: El diseño de la arquitectura de tres capas

A continuación, un breve resumen de cada capa:

- **La primera capa (Proveedores):** Representa las diversas fuentes de datos, caracterizadas por una mezcla de tipos de datos, formatos y ubicaciones.

- **La segunda capa (Distribución):** Es la plataforma de distribución, compleja debido a la enorme cantidad de herramientas y tecnologías disponibles para la integración.

- **La tercera capa (Consumidores):** Comprende los consumidores de datos que utilizan servicios de datos para obtener predicciones, automatización e insights en tiempo real.

La evolución de la capa de distribución ha pasado de los sistemas de *data warehouse* propietarios tradicionales a arquitecturas más adaptables y distribuidas, conocidas como el **modern data stack**. El desafío de este enfoque es que requiere la integración de muchos servicios y herramientas independientes, lo que representa una barrera de entrada significativa.

Los proveedores de tecnología han simplificado esto creando plataformas de software completas, especialmente con Apache Spark y formatos de tabla de código abierto como Delta Lake. Las organizaciones que utilizan Spark y Delta Lake encuentran ventajosa la **Arquitectura Medallion** porque aprovecha las fortalezas de este marco para la gestión de datos de extremo a extremo.

¿Qué es una Arquitectura Medallion?

Una Arquitectura Medallion es un patrón de diseño de datos para organizar lógicamente los datos dentro de un *lakehouse*. Utiliza tres capas (Bronce, Plata y Oro) con el objetivo de mejorar incremental y progresivamente la estructura y calidad de los datos a medida que fluyen a través de ellas.

- **Capa Bronce:** Almacena los datos crudos de diversas fuentes en su estructura nativa, sirviendo como un registro histórico.

- **Capa Plata:** Refina y estandariza los datos crudos a través de chequeos de calidad, estandarización y deduplicación. Actúa como una etapa de transición para datos procesados y granulares.

- **Capa Oro:** Optimiza los datos refinados para obtener insights de negocio específicos. Agrega, resume y enriquece los datos para informes y análisis de alto nivel, enfatizando el rendimiento y la escalabilidad.

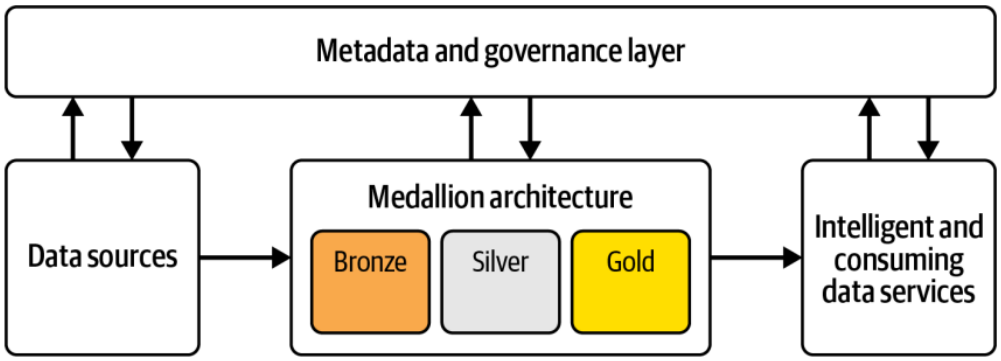

Figura 1-2: Una arquitectura Medallion, que organiza los datos en tres capas, mejorando la estructura y calidad de los datos a medida que progresan a través de las capas

Aunque las etiquetas son intuitivas, muchas empresas tienen dificultades para implementar eficazmente este modelo en capas. Para diseñar una Arquitectura Medallion, es crucial comprender primero la evolución de las arquitecturas de datos, desde los *data warehouses* tradicionales hasta los *data lakes* y, finalmente, el *lakehouse*.

Una Breve Historia de la Arquitectura de Data Warehouse

En la década de 1990, el *data warehousing* surgió para crear una "versión consistente de la verdad" para la toma de decisiones empresariales. El proceso implicaba recolectar datos de varias fuentes, transformarlos y cargarlos en un repositorio central.

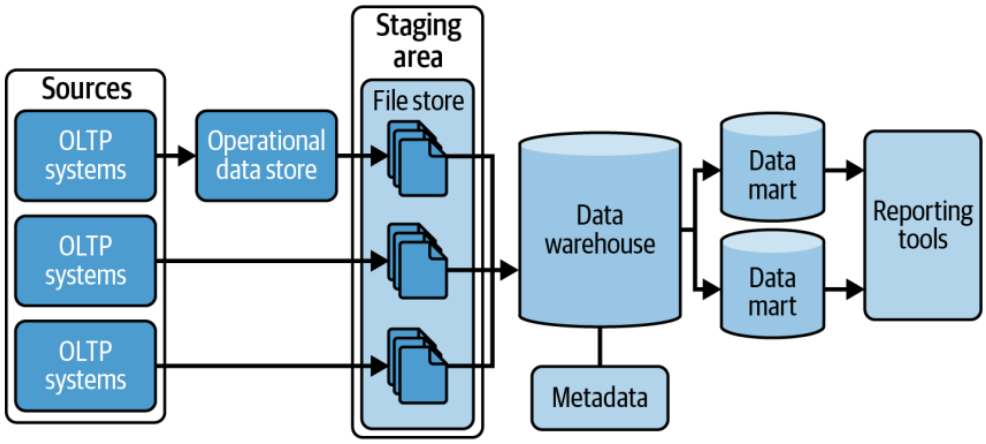

Figura 1-3: Arquitectura típica de un data warehouse

Sistemas OLTP

La mayoría de los sistemas fuente eran para propósitos transaccionales u operacionales, conocidos como sistemas de procesamiento de transacciones en línea (OLTP). Estos sistemas tienen cargas de trabajo predecibles y consultas de bajo volumen (leer, actualizar, eliminar un registro). Para optimizar estas consultas, los sistemas OLTP suelen estar **normalizados**.

Normalización versus Desnormalización de Bases de Datos

- **Normalización:** Proceso que reestructura los datos para reducir la redundancia y mejorar la integridad (la tercera forma normal, 3NF, es la más común).

- **Desnormalización:** Proceso inverso que introduce redundancia intencionadamente para mejorar el rendimiento de las consultas, común en el *data warehousing*.

Los sistemas OLTP, diseñados con modelos normalizados y priorizando las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad), presentan varias implicaciones:

1. No están diseñados para ofrecer una vista analítica consolidada. Extraer datos para consultas complejas ejerce una gran presión sobre ellos.

2. Los requisitos de alto rendimiento los hacen costosos, por lo que a menudo solo almacenan los datos más recientes.

3. Fueron diseñados de forma aislada, lo que dificulta la creación de una vista unificada

sin un esfuerzo de integración significativo.

Data Warehouses

El *data warehouse* sirve como un centro central para el procesamiento analítico en línea (OLAP). A diferencia de los sistemas OLTP, los requisitos de integridad pueden ser menos estrictos y están optimizados para lecturas intensivas. Los datos suelen estar desnormalizados y duplicados para facilitar diferentes patrones de lectura.

El Área de Staging Para cargar datos en el *data warehouse*, primero se extraen a una ubicación intermedia llamada **área de staging**. Esta área, que puede ser una base de datos relacional o un almacén de archivos, se utiliza para la manipulación posterior y a menudo retiene copias históricas para reprocesamiento o auditoría. El proceso completo se conoce como **extract, transform, and load (ETL)**.

Metodología Inmon Propuesta a principios de los 90, la metodología Inmon es un enfoque de arriba hacia abajo (*top-down*). Primero se construye un *data warehouse* empresarial (EDW) centralizado y altamente estructurado, con un modelo de datos normalizado (3NF). A partir de este EDW, se crean **data marts** (subconjuntos de datos para casos de uso específicos) con modelos desnormalizados (esquemas de estrella) para un mejor rendimiento de las consultas.

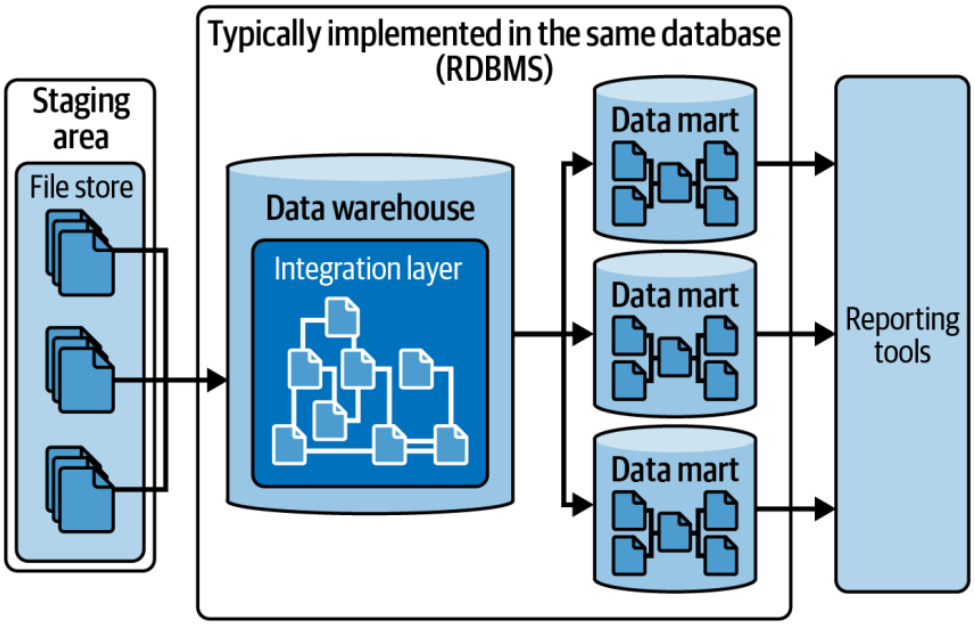

Figura 1-4: El enfoque Inmon; un diseño de arriba hacia abajo donde se construye primero un data warehouse centralizado y luego se crean data marts a partir de este almacén central

Este enfoque de dos pasos fue criticado por requerir más tiempo y recursos, y por la duplicación de datos.

Metodología Kimball Introducida en 1996, la metodología Kimball es un enfoque de abajo hacia arriba (*bottom-up*). Se centra en construir primero **data marts dimensionales** para responder a las necesidades del negocio, utilizando un esquema de estrella. El *data warehouse* se ve como una colección de estas tablas dimensionales.

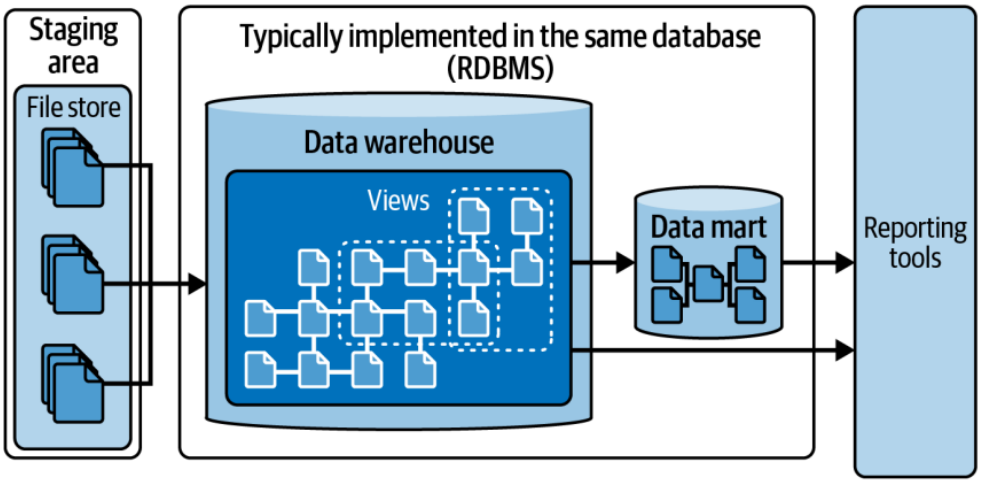

Figura 1-5: La metodología Kimball; un enfoque de abajo hacia arriba para construir el data warehouse

Confusión sobre las Funciones de las Capas del Data Warehouse Típicamente, un *data warehouse* tiene tres capas funcionales:

1. **Capa de Staging/Ingesta:** Donde se almacenan los datos crudos.

2. **Capa de Integración/Transformación:** Donde los datos se limpian, enriquecen y unifican.

3. **Capa de Presentación:** Donde se seleccionan y remodelan los datos para casos de uso específicos.

El número de capas puede variar según las necesidades. Para el modelado dimensional, Kimball introdujo el concepto de **dimensiones de cambio lento (SCDs)** para historizar los datos.

- **SCD1 (sobrescribir):** Actualiza el registro existente. No se guarda historial.

- **SCD2 (añadir nueva fila):** Crea un nuevo registro para cada cambio, preservando el historial completo.

- **SCD3 (añadir nuevo atributo):** Añade un nuevo atributo para rastrear un cambio limitado (normalmente solo el estado anterior).

Conclusiones Clave de los Data Warehouses Tradicionales

1. **El concepto de capas** es una estrategia eficaz para separar responsabilidades.

2. **El modelado de datos** es crucial para la flexibilidad, el rendimiento y como interfaz para el negocio.

3. **La integración estrecha entre software y hardware** (escalado vertical) era la norma, lo que los hacía costosos y con limitaciones de escalabilidad.

Aunque ofrecen datos estandarizados de alta calidad, los *data warehouses* tradicionales luchan con el volumen de datos, la flexibilidad para cargas de trabajo no estructuradas y los costos, lo que llevó a la exploración de nuevas arquitecturas.

Una Breve Historia de los Data Lakes

Los *data lakes* surgieron a mediados de la década de 2000 como una solución a las deficiencias de los *data warehouses*. Introdujeron una nueva arquitectura distribuida que podía gestionar enormes cantidades de datos en diversos estados (estructurados, semiestructurados y no estructurados) utilizando software de código abierto sobre hardware básico.

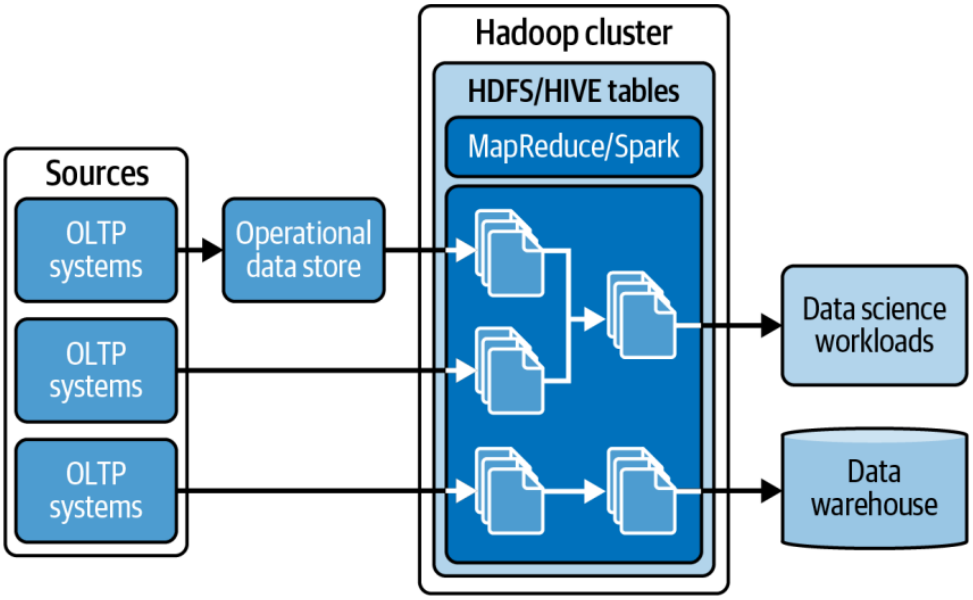

Figura 1-6: Arquitectura típica de un data lake con copias de datos sin procesar

La primera generación de *data lakes* utilizó principalmente **Hadoop**, con sus componentes principales:

Hadoop's Distributed File System (HDFS)

Un sistema de archivos distribuido conocido por su tolerancia a fallos y su capacidad para escalar horizontalmente (añadiendo más máquinas), separando el cómputo del almacenamiento. Divide los datos en grandes bloques inmutables (solo se puede añadir, no actualizar) que se replican a través del clúster.

MapReduce

Un modelo de programación para procesar datos en paralelo a través de un clúster. Utiliza tres fases: **map**, **shuffle** y **reduce**. Su rendimiento puede verse afectado si los datos no se distribuyen uniformemente.

Apache Hive

Desarrollado por Facebook, proporciona una capa SQL sobre Hadoop, permitiendo consultas tipo SQL (HiveQL) que se traducen en trabajos de MapReduce. Introduce el concepto de **schema-on-read** (el esquema se aplica al leer los datos, no al escribirlos), a diferencia del **schema-on-write** de las bases de datos tradicionales.

- **Tablas externas e internas:** Las externas enlazan a datos fuera de Hive (ej.

CSV), mientras que las internas son gestionadas por Hive, a menudo en formatos columnares como ORC y Parquet para un mejor rendimiento analítico.

- **Hive Metastore:** Un repositorio central que almacena metadatos sobre las tablas, crucial en las arquitecturas Medallion actuales.

Los desafíos iniciales de esta arquitectura incluían el manejo ineficiente de archivos pequeños, la falta de transacciones ACID y la lentitud de MapReduce.

Proyecto Spark

Iniciado en 2009 en UC Berkeley para acelerar los trabajos de Hadoop, **Apache Spark** es un marco de computación en memoria. Evolucionó para incluir **Spark SQL**, que mantuvo la compatibilidad con el Hive Metastore pero ofreció mejoras de velocidad significativas. Sin embargo, Spark requiere un tiempo de arranque para cargar los datos en memoria.

Avanzando con los Data Lakes

Los *data lakes* son robustos para almacenar datos crudos en formatos abiertos e interoperables. Sin embargo, transformar esos datos en valor de negocio es complejo, y luchan con la latencia, el rendimiento de las consultas y la falta de soporte transaccional. Esto llevó a la industria a buscar una solución que combinara lo mejor de los *data lakes* y los *data warehouses*.

Una Breve Historia de la Arquitectura Lakehouse

La arquitectura *lakehouse* surge para combinar la escalabilidad y flexibilidad del *data lake* con la fiabilidad y el rendimiento del *data warehouse*.

Fundadores de Spark

Los creadores de Spark fundaron **Databricks** en 2013, adoptando una estrategia de distribución solo en la nube. Esta decisión fue acertada, ya que la industria se alejó de las implementaciones de Hadoop locales. Hoy, Hadoop sigue vivo en la nube, pero con cambios significativos: HDFS ha sido reemplazado por almacenamiento de objetos en la nube, y Spark puede operar de forma independiente y elástica.

Emergencia de Formatos de Tabla Abiertos

Para abordar la necesidad de garantías transaccionales e integridad de datos en los *data lakes*, surgieron varios proyectos de formatos de tabla de código abierto:

- **Apache Hudi (2017):** Iniciado por Uber, se centró en *upserts*, eliminaciones y procesamiento incremental eficientes.

- **Apache Iceberg (2018):** Lanzado por Netflix, abordó problemas de rendimiento y complejidad en sistemas de análisis a gran escala.

- **Delta Lake (2019):** Lanzado por Databricks, trajo transacciones ACID, manejo de metadatos escalable y unificación de streaming y batch. Utiliza un **registro de transacciones (DeltaLog)** para rastrear cada cambio como *commits* atómicos, lo que permite la función de **"viaje en el tiempo" (*time travel*)** para ver estados anteriores de una tabla.

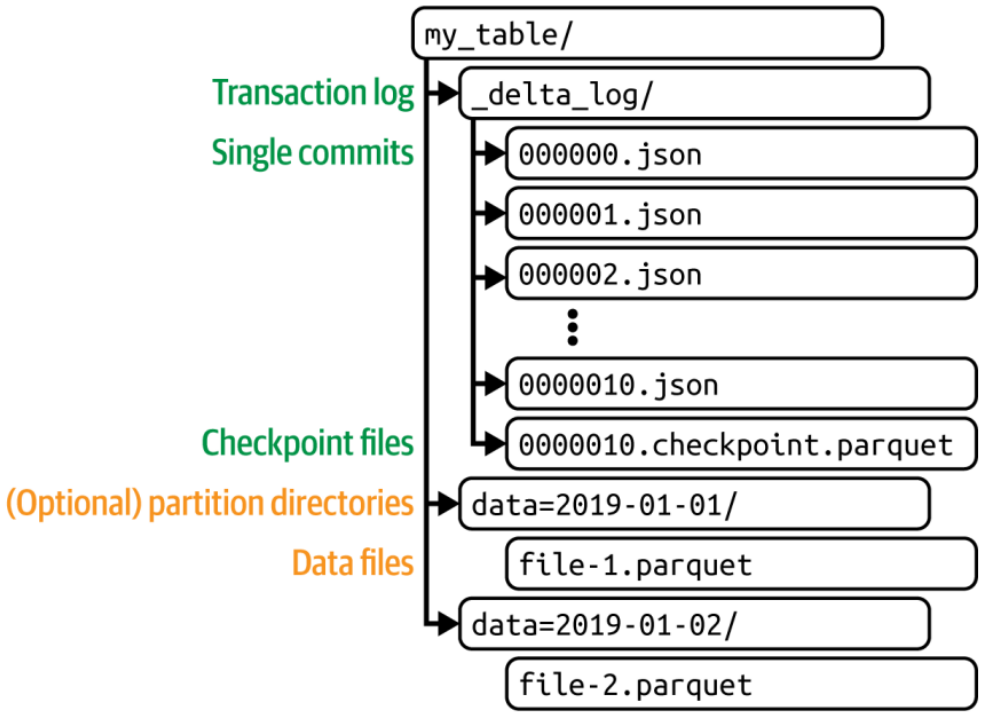

Figura 1-7: Ejemplo de cómo Delta Lake estructura sus datos y su registro de transacciones

El Auge de las Arquitecturas Lakehouse

El concepto de *lakehouse* combina los beneficios de los *data lakes* y los *data warehouses* en una plataforma unificada, típicamente construida sobre Apache Spark (para el cómputo) y Delta Lake (para la capa de almacenamiento).

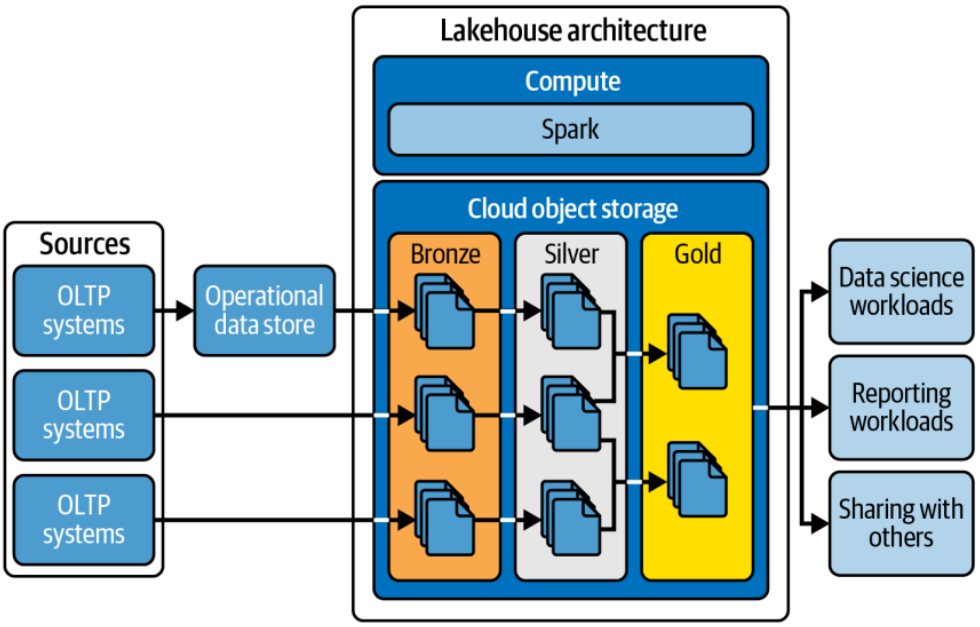

Figura 1-8: Arquitectura típica de un lakehouse, con una representación de las capas Bronce, Plata y Oro incluidas

Esta arquitectura se distingue por soportar almacenamiento de bajo costo en la nube mientras proporciona transacciones ACID y un rendimiento significativamente mejorado. A partir de 2025, los principales actores en este espacio incluyen:

- **Databricks:** Pionero del término *lakehouse*, integra Delta Lake y Apache Spark.
- **Azure HDInsight, Azure Synapse Analytics, Microsoft Fabric:** Servicios de Microsoft que combinan big data y *data warehousing*.
- **Cloudera, Dremio, Starburst:** Otras plataformas robustas que soportan arquitecturas *lakehouse*.
- Otros gigantes como **AWS, Google Cloud Platform y Snowflake** también han comenzado a incorporar el concepto.

Databricks y Microsoft han respaldado la Arquitectura Medallion como una mejor práctica para organizar los datos en capas dentro de arquitecturas basadas en Spark y Delta Lake.

La Arquitectura Medallion y sus Desafíos Prácticos

La Arquitectura Medallion es un patrón de diseño lógico para organizar datos en un *lakehouse*. Sus etiquetas (Bronce, Plata, Oro) son intuitivas, pero a menudo falta una guía práctica clara, lo que genera confusión sobre los roles específicos de cada capa. Existe una brecha significativa entre la guía teórica y la ejecución real.

Conclusión

La evolución de las arquitecturas de datos (de *data warehouse* a *lakehouse*) ha sido impulsada por la necesidad de manejar la escala, diversidad y complejidad de los datos modernos. Este cambio ha trasladado la gestión de la infraestructura local a servicios distribuidos y gestionados en la nube, donde la configuración, el diseño en capas y el modelado de datos son cruciales.

La Arquitectura Medallion reconoce estos desafíos, pero su aplicación práctica expone una brecha entre los modelos teóricos y las implementaciones del mundo real, subrayando la necesidad continua de un modelado de datos preciso y estrategias de gobernanza adaptadas a cada organización.