What the actual fuck is a Lakehouse?

... and Why?

Juamini 3 Pro Preview

November 23, 2025

Esta es la síntesis definitiva para tu examen. Vamos a recorrer la evolución lógica de la arquitectura de datos para entender el "por qué" de cada una y cómo el Lakehouse cierra el círculo.

Aquí tienes la explicación comparativa y la solución final.

---

1. El Pasado: Data Warehouse (El Almacén Ordenado)

La era de la certeza y los reportes financieros.

- **Qué es:** Es un repositorio centralizado diseñado para **analítica estructurada**.
- **Filosofía:** *Schema-on-Write* (Esquema al escribir). Antes de guardar un dato, debes definir la tabla, las columnas y limpiar el dato. Si el dato no encaja, rebota.
- **Tecnología:** Bases de datos relacionales y columnares (Snowflake, Redshift, Teradata).
- **Lo Bueno:**
  - **Alto Rendimiento:** Consultas SQL rapidísimas para BI (Business Intelligence).
  - **Confiabilidad:** Soporta transacciones **ACID**. Los datos siempre son consistentes.
  - **Calidad:** Los datos están limpios y ordenados ("Single Source of Truth").

- **El Problema:**
  - **Costo:** El almacenamiento es caro.
  - **Rigidez:** No soporta datos no estructurados (imágenes, logs, texto libre).
  - **Lentitud de Cambio:** Cambiar una tabla requiere mucho trabajo de ingeniería.

2. La Reacción: Data Lake (El Repositorio Flexible)

La era del Big Data y la Ciencia de Datos.

- **Qué es:** Un repositorio centralizado para guardar **todos** los datos (estructurados y no estructurados) a bajo costo.
- **Filosofía:** *Schema-on-Read* (Esquema al leer). Guardas el archivo tal cual llega (Raw). Te preocupas por la estructura recién cuando lo vas a usar.
- **Tecnología:** Almacenamiento de Objetos barato (AWS S3, Azure Blob) + Motores de procesamiento (Hadoop, Spark).
- **Lo Bueno:**
  - **Barato:** Puedes guardar petabytes por centavos.
  - **Flexible:** Acepta cualquier formato (video, JSON, CSV).
  - **Ideal para ML:** Los científicos de datos aman tener la data cruda para explorar.
- **El Problema:**
  - **Data Swamp (Pantano):** Sin gobernanza, se llena de basura que nadie sabe qué es.
  - **Sin ACID:** Si una escritura falla a la mitad, el archivo queda corrupto. No es confiable para datos críticos.

- **Lento para BI:** Hacer un dashboard sobre millones de archivos CSV es lentísimo.

---

3. El Conflicto: La Arquitectura de Dos Niveles

Como ninguno de los dos hacía todo bien, las empresas usaban **ambos**.

1. Guardaban todo en el **Data Lake**.

2. Copiaban y limpiaban una parte pequeña hacia el **Data Warehouse** para los reportes.

Consecuencia: Duplicidad de datos, costos dobles y pipelines ETL frágiles que se rompen constantemente.

---

4. La Solución: Data Lakehouse (La Convergencia)

La arquitectura moderna unificada.

- **Qué es:** Una arquitectura que combina la **flexibilidad, costo y escala** del Data Lake con la **gestión, transacciones y rendimiento** del Data Warehouse.

¿CÓMO lo soluciona? (La Clave Técnica) El Lakehouse no es magia. Funciona agregando una **Capa de Metadatos Inteligente** (como **Delta Lake**, Iceberg o Hudi) sobre el almacenamiento barato del Data Lake.

1. **Almacenamiento Físico:** Sigue usando el almacenamiento barato de objetos (S3/ADLS) como un Data Lake.

2. **Capa de Metadatos (Delta Log):** Agrega un registro de transacciones sobre los archivos. Esto permite saber exactamente qué archivos pertenecen a qué versión de la tabla.

¿POR QUÉ es la solución definitiva?

Problema Anterior	Solución del Lakehouse
Falta de Confianza en el Lake	Implementa **ACID**. Ahora el Data Lake es transaccional. Si falla una carga, no se rompe nada.
Datos Sucios	Implementa **Schema Enforcement**. Aunque son archivos, el sistema impide que guardes datos con el tipo incorrecto (como un Warehouse).
Lentitud en Consultas	Usa motores optimizados (como Spark/Photon) que leen los metadatos para saltarse archivos irrelevantes (*Data Skipping*), logrando velocidad de Warehouse.
Duplicidad (Dos Niveles)	**Elimina el ETL de copia.** Puedes hacer BI y Machine Learning sobre los **mismos datos** físicos.
Gobernanza	Permite funciones como **Time Travel** (ver el dato como estaba ayer) y auditoría completa sobre archivos abiertos.

Resumen Final para tu Examen

El **Lakehouse** soluciona la dicotomía histórica al permitirte:

1. Guardar datos a **bajo costo** (como un Lake).
2. Gestionarlos con **calidad y transacciones** (como un Warehouse).
3. Evitar moverlos de un lado a otro, permitiendo aplicar la **Arquitectura Medallion** (Bronze/Silver/Gold) dentro de una misma plataforma unificada.