



DATASPHERE

WORKSHOP

DATA FOUNDATIONS

INGENIERÍA DE DATOS Y SNOWFLAKE EN ACCIÓN

MÓDULO 2: PROCESOS DE CARGA



Nelson Zepeda

nelson.zepeda@datasphere.tech

Mayo 2025

WWW.DATASPHERE.TECH

Acerca de Datasphere

Datasphere es un proveedor pionero en soluciones avanzadas de análisis y IA, diseñadas para transformar organizaciones en empresas impulsadas por datos.

Fundada en **2019**, está posicionada para aprovechar la creciente demanda de soluciones basadas en datos, utilizando herramientas analíticas modernas y enfoques centrados en el cliente, ayudamos a transformar modelos de negocio y lograr objetivos estratégicos.

Datasphere ha desarrollado con éxito proyectos en El Salvador, Honduras, Bolivia, Tel Aviv, EE. UU. y Sudáfrica.



+60 Proyectos Exitosos
+6 Mercados

Nuestro Stack



www.datasphere.tech

<https://www.linkedin.com/company/datasphere-consulting/>

Agenda

- **SCD**
 - **Late Arriving Dimensions**
 - **ETL / ELT**
 - **DBT**
-

Dimensiones Lentamente Cambiantes

Las dimensiones lentamente cambiantes (**SCD**) son aquellas tablas de dimensiones donde los atributos pueden cambiar con el tiempo.

- El cliente Juan Pérez cambia de dirección, o
- El producto A cambia su categoría.

En sistemas operacionales (OLTP), ese cambio simplemente sobrescribe el dato viejo. Pero en un Data Warehouse, queremos preservar el historial para poder responder preguntas como:

“¿Cuál era la dirección de Juan cuando hizo esa compra en 2021?”

SK	ID	Name	Range	Start Date	End Date	Active
1	SA-23	Samsung Galaxy S7 Edge	High	28/05/2017	31/12/9999	Yes
2	IP-08	Iphone 8	High	01/07/2017	30/06/2018	Yes
3	XI-M5	Xiaomi MI 5	Medium	15/10/2017	14/05/2018	Yes

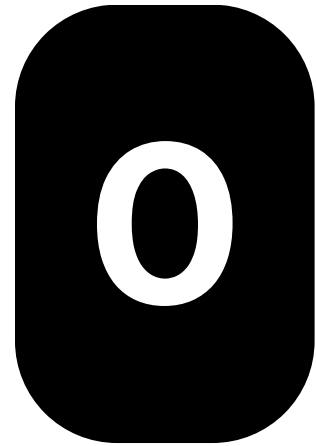


SK	ID	Name	Range	Start Date	End Date	Active
1	SA-23	Samsung Galaxy S7 Edge	High	28/05/2017	15/06/2018	No
2	IP-08	Iphone 8	High	01/07/2017	30/06/2018	Yes
3	XI-M5	Xiaomi MI 5	Medium	15/10/2017	14/05/2018	Yes
4	SA-23	Samsung Galaxy S7 Edge	Medium	16/06/2018	31/12/9999	Yes



SK	ID	Name	Range	Start Date	End Date	Active
1	SA-23	Samsung Galaxy S7 Edge	High	28/05/2017	15/06/2018	No
2	IP-08	Iphone 8	High	01/07/2017	30/06/2018	Yes
3	XI-M5	Xiaomi MI 5	Medium	15/10/2017	14/05/2018	Yes
4	SA-23	Samsung Galaxy S7 Edge	Medium	16/06/2018	31/04/2019	No
5	SA-23	Samsung Galaxy S7 Edge	Low	01/05/2019	31/12/9999	Yes

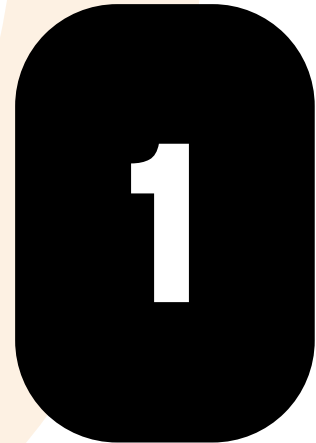
Tipos de SCD



SCD Tipo 0 – Retención estática (no se permiten cambios)

- Los valores no cambian nunca, ni se actualizan ni se versionan.
- Útil para atributos que deben permanecer inmutables una vez definidos (por ejemplo: fecha de nacimiento, país de origen).
- Ideal para datos que son parte de la identidad histórica.

Ejemplo: la nacionalidad del cliente cuando se registró, aunque luego se naturalice.



SCD Tipo 1 – Sobrescritura directa

- El valor anterior se reemplaza completamente por el nuevo.
- No se guarda historial.
- Se usa cuando el valor no afecta análisis históricos, o cuando el cambio es una corrección de error.

Ejemplo: corregir un error tipográfico en el nombre del cliente.

Tipos de SCD

2

SCD Tipo 2 – Versionado completo con historial

- Se crea una nueva fila para cada cambio, y se conserva la fila anterior.
- Permite versionar los datos y mantener un historial completo.
- Usa campos como: fecha_inicio, fecha_fin, activo, o version.

Ejemplo: conocer la dirección que tenía el cliente cuando hizo una compra en el pasado.

3

SCD Tipo 3 – Estado anterior y actual (cambio limitado)

- Se conserva el valor actual y el anterior en la misma fila.
- No permite más de un histórico; útil cuando solo interesa el último cambio.
- Se agregan columnas como estado_anterior, categoria_previa, etc.

Ejemplo: comparar el valor anterior del salario sin guardar cada versión.

Tipos de SCD

4

SCD Tipo 4 – Historial externo (tabla de historial separada)

- La dimensión principal tiene solo el valor actual.
- El historial completo se guarda en otra tabla externa.
- Más simple para modelos donde la dimensión actual es muy consultada, y el historial es ocasional.

Ejemplo: tabla dim_empleado para reportes actuales, y tabla historial_empleado para trazabilidad.

5

SCD Tipo 5 – Combinación de Tipo 1 y 4

- La tabla actual (dimensión) se actualiza como Tipo 1 (sobrescribe).
- Pero hay una tabla de historial separada, como en Tipo 4.
- Permite acceder tanto al estado actual optimizado como al historial detallado.

Ejemplo: consulta rápida con valores actuales y auditoría completa si se necesita.

Tipos de SCD

6

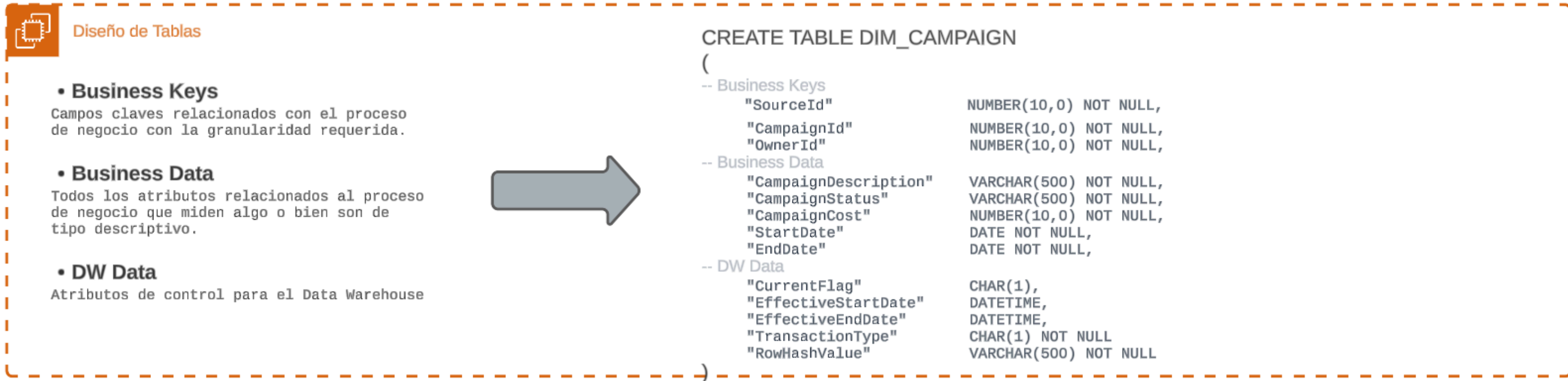
SCD Tipo 6 – Combinación de Tipo 1, 2 y 3 (híbrida)

- Es el más complejo: combina:
 - Versión histórica (Tipo 2),
 - Sobrescritura parcial (Tipo 1),
 - Columna de cambio previo (Tipo 3).
- Se usa cuando necesitas responder distintos tipos de preguntas históricas y actuales sin perder rendimiento.

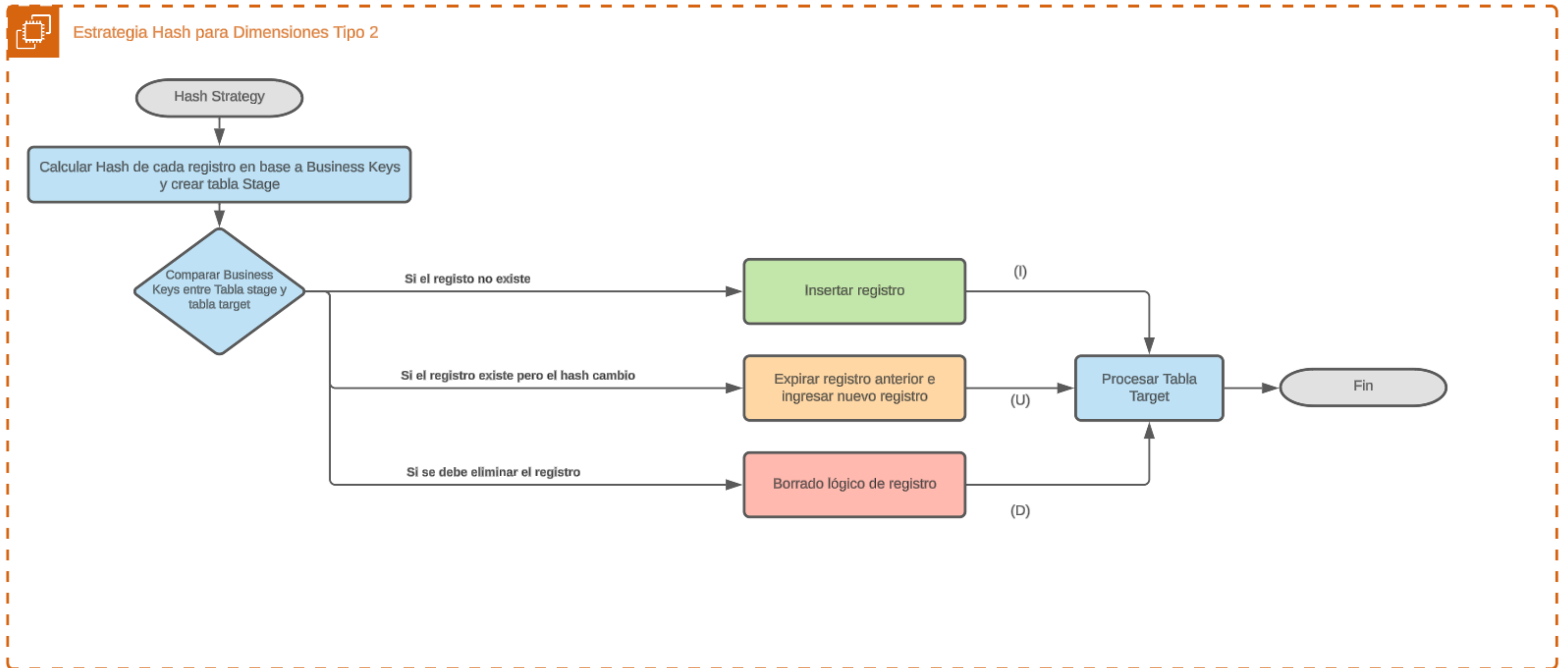
Ejemplo: conocer el estado actual, el anterior y cuándo ocurrió el cambio.

- **SCD Tipo 0 – Retención estática:** No se permite ningún cambio en el valor.
- **SCD Tipo 1 – Sobrescritura directa:** El valor se actualiza sin guardar historial.
- **SCD Tipo 2 – Versionado con historial:** Se crea una nueva fila por cada cambio.
- **SCD Tipo 3 – Valor anterior limitado:** Se mantiene el valor actual y uno anterior en la misma fila.
- **SCD Tipo 4 – Historial en tabla separada:** La dimensión guarda solo el valor actual.
- **SCD Tipo 6 – Combinación híbrida (Tipo 1 + 2 + 3):** Mantiene múltiples versiones, guarda el valor anterior y actualiza también el valor presente.

Tipos de SCD



SCD 2



Late Arriving Dimensions

Late Arriving Dimensions

Las Late Arriving Dimensions o dimensiones que llegan tarde representan un problema común en soluciones de Data Warehouse. Ocurren cuando los registros de dimensión (como clientes, productos o proveedores) no están disponibles en el momento en que se cargan los datos de hechos (por ejemplo, ventas, pagos o visitas).

Esto provoca que los hechos lleguen antes que las dimensiones correspondientes, generando inconsistencias en las relaciones y afectando directamente la integridad y calidad del modelo de datos.

- Las dimensiones y los hechos provienen de sistemas diferentes, con ventanas de procesamiento distintas.
- Existen dependencias técnicas entre la llegada de datos (por ejemplo, procesos ETL separados).
- Se utilizan fuentes externas o procesos manuales que demoran la generación de ciertos atributos de dimensión.
- El modelo de datos no contempla una estrategia para manejar la asincronía natural de los eventos frente a sus contextos de negocio.

¿Por qué ocurre este problema?

Sales Order Fact Table

Fact SK	Order Date SK	Customer SK	Store SK	Order ID	Order Amount	Date
1	20171210	97	351	319	96	09th Dec

Fact SK	Order Date SK	Customer SK	Store SK	Order ID	Order Amount	Date
1	20171210	97	351	319	96	10th Dec
2	20171211	0	274	213	75	10th Dec

Fact SK	Order Date SK	Customer SK	Store SK	Order ID	Order Amount	Date
1	20171211	97	351	319	96	11th Dec
2	20171211	65	274	213	75	11th Dec

¿Cómo se resuelve?

1. Uso de claves temporales (surrogate keys genéricas)

Cuando una dimensión aún no está disponible al cargar un hecho, se puede asignar una clave genérica como -1, representando una entidad “desconocida” o “pendiente”.

- En la dimensión se crea un registro especial con esa clave (id_cliente = -1, nombre = 'Cliente Desconocido').
- El hecho se carga sin errores de integridad referencial.
- Una vez que la dimensión real llega, se puede actualizar el hecho.

2. Uso de fechas efectivas (Effective Start Date / Effective End Date)

En lugar de confiar únicamente en la fecha de carga, se utilizan fechas efectivas para registrar cuándo los datos de dimensión deberían haber sido válidos.

- La dimensión se modela con rangos de validez.
- El motor de análisis (o proceso ETL) selecciona la versión correcta de la dimensión en función de la fecha del hecho.

Ejemplo:

Un producto se vendió el 2 de febrero, pero su registro llegó el 10 de febrero.

Se inserta en la dimensión con Effective Start Date = 02/02 para que los hechos se puedan unir correctamente.

¿Cómo se resuelve?

3. Implementación de Slowly Changing Dimensions (SCD), especialmente Tipo 2

Al aplicar SCD Tipo 2:

- Cada versión de la dimensión se almacena como un nuevo registro.
- El hecho se enlaza con la versión de la dimensión vigente en la fecha del evento.

Aplicación al caso:

Una dimensión llega tarde, pero al insertarla con la fecha efectiva correcta, se puede reconstruir la relación sin perder la historia ni modificar los hechos originales.

4. Reprocesamiento de hechos

Cuando una dimensión llega tarde, se pueden marcar los hechos pendientes de actualización, y ejecutar un proceso que los actualice con la clave correcta una vez disponible.

Etapas típicas del proceso:

1. Cargar el hecho con clave genérica.
2. Al detectar la llegada de la dimensión, identificar los hechos relacionados.
3. Ejecutar un UPDATE para asignar la clave surrogate real.

¿Cómo se resuelve?

5. Diseño de un proceso ETL con detección de dimensiones faltantes

El proceso de carga puede incluir una etapa intermedia que detecte hechos con referencias a dimensiones aún no disponibles.

- Estas filas se pueden almacenar en una tabla de staging o en una cola de pendientes.
- Una vez que llega la dimensión, se activa el flujo de carga de los hechos retenidos.

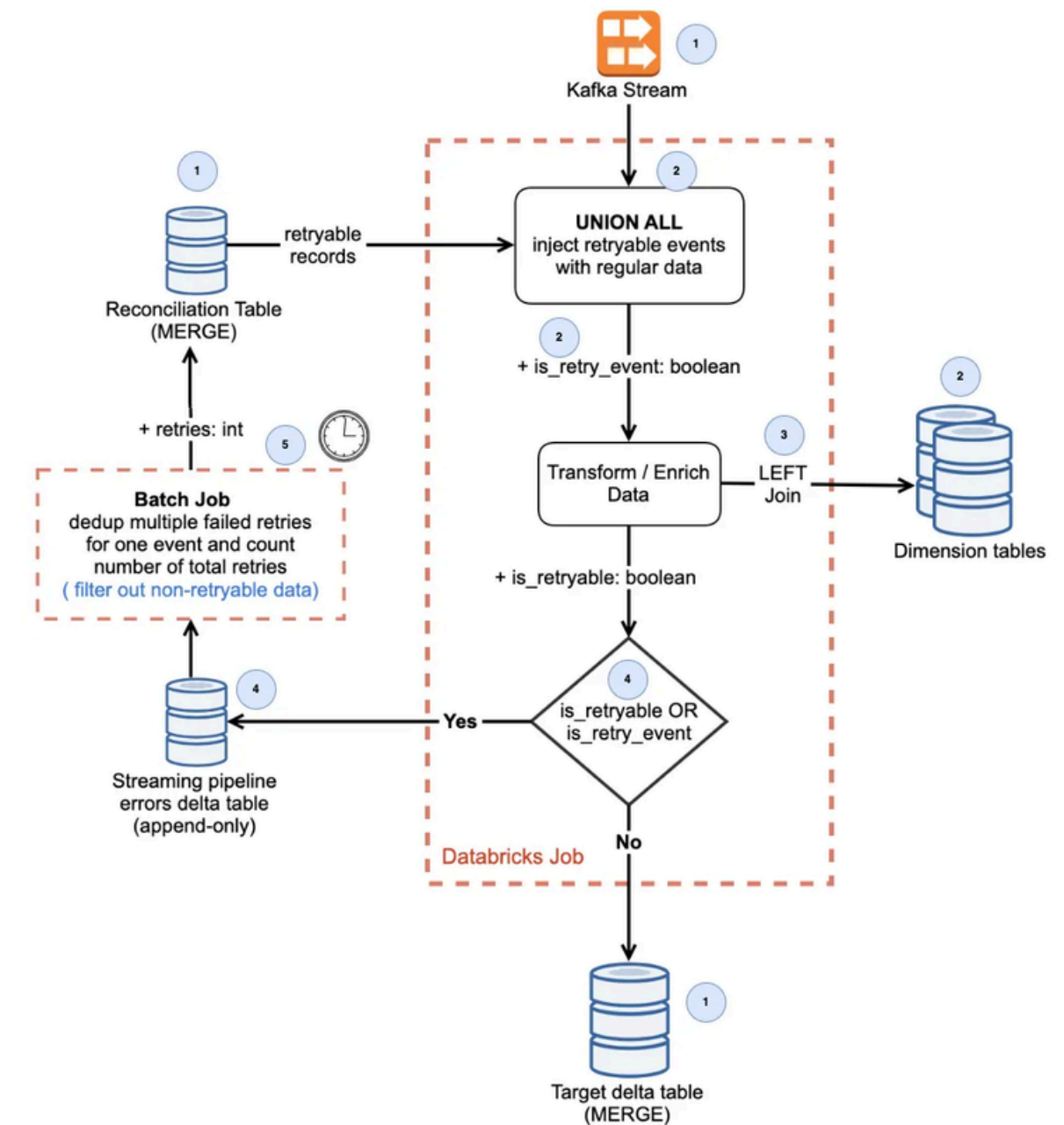
La elección de la estrategia depende de:

El volumen de datos y la frecuencia de llegada de hechos y dimensiones.

Las necesidades analíticas del negocio (¿importa la historia? ¿se permiten sobrescrituras?).

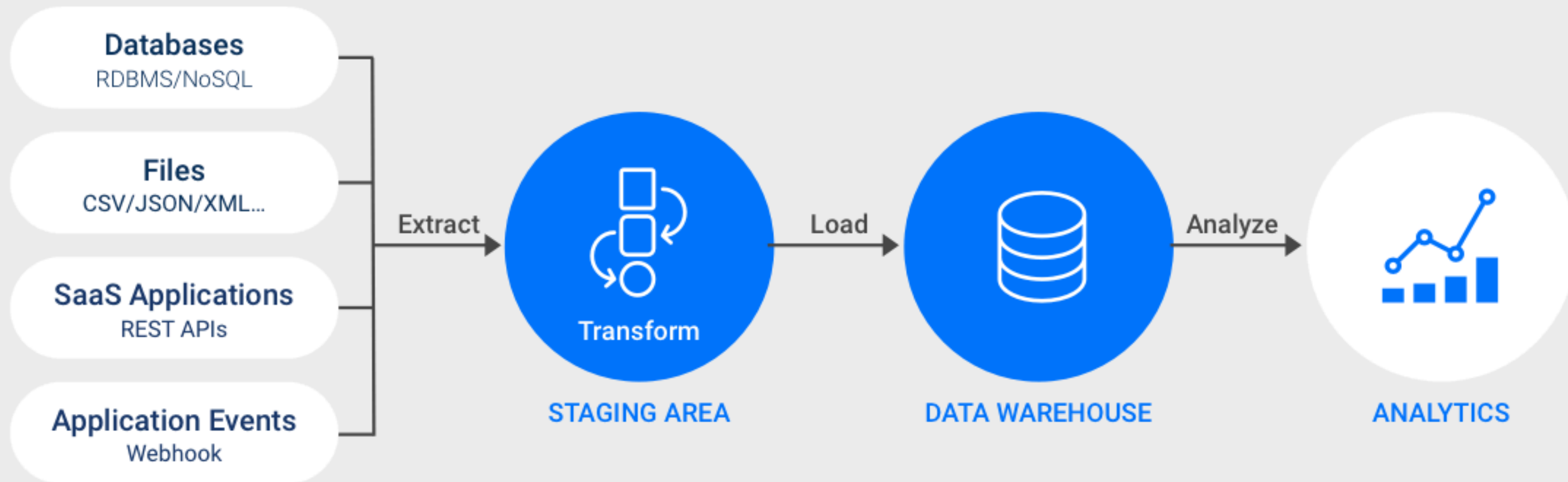
El nivel de automatización y control del proceso ETL.

Una solución robusta suele combinar fechas efectivas, claves genéricas, y procesos de reprocesamiento.

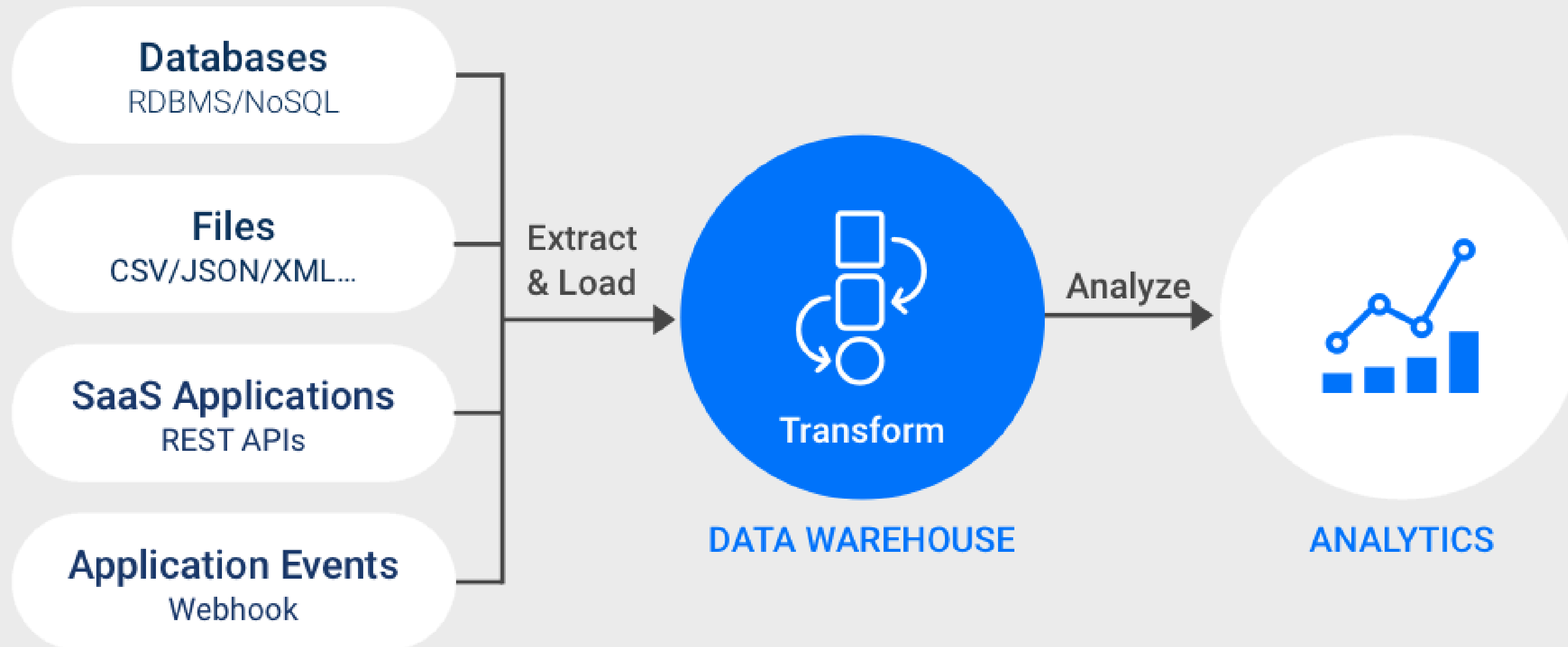


ETL / ELT

ETL PROCESS



ELT PROCESS



Un proceso ELT consiste en extraer los datos desde las fuentes operacionales, cargarlos directamente y sin transformar al entorno de almacenamiento analítico

ELT

Mover los datos sin transformarlos de inmediato —como ocurre en el enfoque ELT— no es una omisión, sino una estrategia deliberada que responde a nuevas capacidades tecnológicas y necesidades analíticas modernas.

1. Desacoplar la ingesta de la transformación

Separar la carga de los datos de su transformación permite:

- Asegurar primero la disponibilidad del dato, incluso si no está claro aún cómo se va a transformar.
- Evitar bloqueos o cuellos de botella en los procesos ETL tradicionales, donde si una transformación falla, el dato nunca llega al DWH.

3. Aprovechar la potencia del motor de BD

Con tecnologías modernas como Snowflake, BigQuery, Redshift o Databricks, el motor de cómputo:

- Es altamente paralelo y escalable.
- Está optimizado para procesar grandes volúmenes.
- Permite ejecutar transformaciones complejas directamente con SQL, Spark o dbt.

2. Preservar datos en su forma original (raw layer)

Cargar primero los datos crudos permite:

- Auditoría y trazabilidad completa.
- Volver a aplicar reglas de negocio si cambian en el futuro.
- Comparar lo que llegó versus lo que se transformó (útil en contextos regulatorios o financieros).

5. Velocidad y confiabilidad en la entrega de datos

- Es más rápido cargar datos "tal cual" que transformarlos al vuelo.
- Las tareas de ingesta pueden correr de forma continua y simple.
- Se asegura que los datos están disponibles en cuestión de minutos, y se transforman bajo demanda.

Herramientas

Herramienta	ETL	ELT	Orquestación
Apache Airflow	✓ (via scripts)	✓ (si se conecta a dbt, SQL, etc.)	✓
dbt (Data Build Tool)	X	✓	X
Fivetran	X	✓	X
Airbyte	X	✓	X
Talend	✓	X (parcial)	X
Pentaho (Kettle)	✓	X	X

Herramienta	ETL	ELT	Orquestación
Informatica PowerCenter	✓	X	X
Microsoft SSIS	✓	X	X
Matillion	✓	✓	X
Hevo Data	X	✓	X
Dataform	X	✓	X
Dagster	✓	✓	✓



Herramientas

Herramienta	ETL	ELT	Orquestación
Prefect	✓	✓	✓
Luigi	✓	X	✓
Apache NiFi	✓	X	X
Databricks Workflows	✓	✓	✓



DBT



No extrae ni carga datos.

Su enfoque es **ELT**: transforma datos dentro del Data Warehouse (Snowflake, BigQuery, Redshift, Databricks, etc.).

dbt es una herramienta de transformación de datos que permite a los equipos de analítica escribir consultas SQL transformacionales como código, con control de versiones, modularidad, pruebas y documentación integrados.

¿Por qué dbt es tan popular?

1. SQL como lenguaje principal

- No requiere aprender Python ni herramientas ETL complejas.
- Cualquier analista o ingeniero con conocimiento en SQL puede usarlo.

2. Transformaciones declarativas

- Los modelos (tablas, vistas) se definen en SQL como pasos consecutivos.
- dbt se encarga de construir la dependencia entre ellos.

3. Versionado y control como en desarrollo de software

- Se gestiona con git, permite revisar cambios, ramas, pull requests.
- El pipeline de datos es auditable y colaborativo.

4. Documentación automática y linaje

- dbt genera documentación navegable con descripciones y dependencias entre modelos.
- Ofrece un diagrama de linaje que muestra cómo se conecta todo el flujo de datos.

Bibliografía

Zagni, Roberto

- Data Engineering with dbt: A practical guide to building a cloud-based, pragmatic, and dependable data platform with SQL.
- Independently published, 2022.

Reis, Joe & Housley, Matt

- Fundamentals of Data Engineering: Plan and Build Robust Data Systems.
- O'Reilly Media, 2022. ISBN: 9781098108304.

Kimball, Ralph & Caserta, Joe

- The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data.
- Wiley, 2004. ISBN: 9780764567575.

Coyne, Sean

- Ace the Data Engineering Interview: Questions and Answers for Python, SQL, Data Modeling and More.
- Independently published, 2022.

Preguntas

¿Qué ventajas ofrece un Data Warehouse moderno para ejecutar procesos ELT?

¿Qué es dbt y por qué se ha vuelto una herramienta clave en ingeniería de datos?

¿Cómo se puede manejar una dimensión tardía con SCD Tipo 2?

¿Cuál es la diferencia entre el enfoque de Inmon y el de Kimball?

¿Qué ventajas tiene el SCD Tipo 2 frente al Tipo 1?

¿Cómo garantizarías integridad referencial en un pipeline real-time donde las dimensiones pueden llegar horas después del hecho y no se puede usar reprocesamiento completo?

En un entorno regulado (ej. banca), ¿por qué el modelo de Inmon podría ser preferido sobre Kimball, y cómo manejarías el rendimiento en consultas?

¿Por qué Databricks no es considerado una MPP tradicional, y qué implicaciones tiene eso para particionamiento y paralelismo horizontal?

Compara la ejecución distribuida de Spark en Databricks con un motor MPP como Redshift

Preguntas

¿Qué implicaciones tiene mantener dos lógicas de procesamiento (batch y streaming) sobre la mantenibilidad y gobernanza de datos?

¿En qué casos concretos dejarías de usar Lambda Architecture y migrarías a un enfoque alternativo como Kappa o Medallion?

¿Qué estrategia aplicarías para escalar un pipeline que hoy procesa 10M de filas y debe crecer a 1B?

¿Qué ventajas ofrece una arquitectura MPP frente a SMP?

Define Schema Evolution en un Data Lake

Diferencias entre Data Mesh y Data Lake

Que es Apache Iceberg



DATASPHERE

GRACIAS!



Nelson Zepeda

nelson.zepeda@datasphere.tech

Mayo 2025

WWW.DATASPHERE.TECH