

Prueba técnica para ingeniero en datos, banco base – ejercicio 2

Este escenario es complejo por que las fuentes f1, f2 y f3 son heterogéneas y corren 24/7. primero, reunimos datos de clientes y transacciones desde f1 (crm propietario), f2 (sql server) y f3 (postgres) identificando sólo las columnas críticas (id cliente, datos demográficos, contacto, transacciones y producto). minimizamos el impacto sobre sistemas origen con cdc, réplicas read-only o extracciones batch en horas de bajo trafico. así obtenemos un conjunto de datos coherente, incremental y listo para procesar.

Luego, transformamos en etapas: raw para mantener datos sin alterar, staging para normalizar tipos y nombres, y una capa final de integración donde se consolidan dimensiones (cliente, producto) y hechos (transacciones) en un warehouse para consultas sql eficientes. para análisis más complejos (clustering, grafos), almacenamos datos en un data lake (para flexibilidad de formatos) y en una base de grafos, respectivamente. herramientas como spark, dbt y airflow nos permiten transformar, orquestar y versionar el pipeline, mientras que un catálogo de datos y mdm aseguran la gobernanza.

Finalmente, se refuerza la seguridad con cifrado, roles, vpc privada, control de accesos y auditoría continua para evitar fugas o intrusiones. cada etapa del pipeline se documenta con un data catalog y tracking de linaje, garantizando transparencia y manteniendo la coherencia del sistema. en suma, la arquitectura integra fuentes heterogéneas, permite consultas operativas, análisis avanzado, seguridad y gobernanza de punta a punta.

A. De cada fuente de datos se tienen identificados que campos requiere el área operativa. ¿Para cumplir con los dos objetivos que subconjunto de cada fuente de datos extraerías?

- Extraería de f1 los datos de cliente (id cliente, info demográfica, contacto), de f2 y f3 las columnas transaccionales relevantes (id producto, id cliente, monto, fecha, canal, etc.). no me llevaría campos irrelevantes ni logs internos, solo lo estrictamente necesario para análisis operativo y data science.

B. ¿Qué posibles retos implica la extracción de cada una de las fuentes de datos por separado y qué herramientas utilizas ?

- Para extraer f1 (crm propietario) usaria el api propietaria del propio crm, o sdk del vendor. Para f2 y f3 (sql server y postgresql) es más sencillo: usaría una herramienta cdc (ej. debezium) o queries incrementales sobre las tablas.

C. ¿Qué posibles retos implica la independencia en el modelo de datos de las tres fuentes y cómo los resolverías?

- Existen retos en la independencia de los modelos de datos y su solución: cada fuente fue diseñada independiente, así que las llaves, nombres de campos, e incluso definiciones semánticas difieren, el reto es alinear estas entidades.

Solución:

- un diccionario global: defino una taxonomía estandarizada (ej. campo "id_cliente" en todas las fuentes, "fecha_transacción" unificado)
- un proceso de staging: normalizar, mapear los ids locales de producto a una tabla global de dimensión productos.
- reconciliar duplicados inconsistencias en clientes o productos.
- si tengo dimensionadores globales (ej. tabla dimensión clientes y dimensión productos en el dwh), luego todas las transacciones se mapean a esas dimensiones estandarizadas. así normalizo y unifico el modelo.

D. ¿Aparte de un proceso batch en la hora de menor uso, cómo podrías mitigar el impacto de tu pipeline sobre las fuentes originales?

- extraer desde réplicas read-only: configuro réplicas de las bdds transaccionales para no afectar las consultas de producción.
- usar cdc/streams en vez de batch queries full: cdc registra cambios incrementales y minimiza la carga.

E. ¿Cuáles etapas considerarías en tu proceso de transformación de datos y qué uso les darías?

1. raw: dump crudo tal cual llegan
2. staging: limpieza mínima, tipificación, normalización básica
3. curated: joins, conformación de dimensiones, métricas calculadas, usando las llaves comunes, creando tablas dimensión (dim_cliente, dim_producto) y tablas fact (fact_transacciones). esta es la capa donde creo el modelo star-schema o similar.
4. analítica: sobre el dwh y sobre el data lake, creo vistas materializadas, agregados, métricas derivadas para análisis operativo y también exporto features para el ds team (ej. últimas compras por cliente, frecuencia, segmentación).

F. ¿Qué herramientas utilizas para las etapas de transformación?

- spark (batch o streaming) para grandes volúmenes de datos y transformaciones complejas, por que es escalable y soporta múltiples fuentes.
- dbt para transformar datos en sql dentro del dwh, en especial para testing y versionado.
- python scripts para validaciones, cálculos puntuales, lógica custom.

G. ¿Qué storage usarías para cada propósito y por qué ?

- para consultas sql operativas: un data warehouse cloud (snowflake, bigquery, redshift) ya que ofrecen performance, escalabilidad, y soporte nativo para sql analítico. además soportan seguridad y gobernanza integrada, lo que facilita acceso a usuarios no técnicos.
- para data science: un data lake (s3, gcs, o adls) así guardo datos en formatos abiertos (parquet, delta), a menor costo, flexible para análisis exploratorio
- el porqué: dwh = excelente para sql, data lake = flexibilidad a bajo costo para ds, graph db = optimizada para grafos. así cada propósito en su mejor entorno.

H. Recuerda que al menos a diario tendrás que llevar data nueva a tu etapa de transformación final, ¿Como orquestarias tu pipeline y con qué herramienta?

I. Proporciona un diagrama de tu propuesta de arquitectura.



II. Seguridad (manteniendo tu rol de ingeniero de datos).

A. ¿Cómo mantendrías la seguridad de tu flujo de datos end-to-end? Es decir disminuir riesgos de posibles fugas o intrusiones no deseadas al entorno de ejecución que estás construyendo.

- cifrado en tránsito (tls) entre todas las conexiones, así evito sniffing.
- cifrado en reposo usando kms en s3, at-rest encryption en dwh.
- control de acceso a través de roles, ldap, o azure ad, asignando permisos mínimos necesarios.
- auditoría continua: registro de accesos, queries y cambios en metadatos
- segmentación de redes para aislar entornos de producción.

III. Gobernanza de datos

A. ¿Cómo llevarías control de la metadata y sus cambios al igual que los procesos de tu pipeline y cómo almacenarías estos datos?

- data catalog para metadatos, ej. aws glue data catalog, collibra o apache atlas, para mantener glosarios, linaje de datos, definiciones de campos. bc así el equipo sabe qué significa cada campo y su origen.
- versionado de transformaciones con git + dbt. la metadata de pipelines (ej. steps, dependencias, tablas generadas) se documenta.
- almacenar metadatos en un repositorio central (puede ser un dwh de metadatos), así trackeo la evolución del esquema en el tiempo.
- tener lineage tracking (openlineage o amundsen) para ver de dónde viene cada dato y qué se alimenta de él, útil para troubleshooting y cumplimiento normativo.