Arquitectura de bases de datos

¿Qué es la arquitectura de datos?

Una arquitectura de datos describe cómo se gestionan los datos, desde la recopilación hasta la transformación, la distribución y el consumo, estableciendo el plan de cómo fluyen los datos a través de los sistemas de almacenamiento de datos. Es fundamental para las operaciones de procesamiento de datos y las aplicaciones de inteligencia artificial (IA).

El diseño de una arquitectura de datos suele basarse en los requisitos empresariales y las necesidades de datos, que son los que utilizan los arquitectos de datos y los ingenieros de datos para definir el modelo de datos y las estructuras de datos subyacentes que lo soportan. El diseño suele facilitar una estrategia o necesidad empresarial, como la elaboración de informes o una iniciativa de ciencia de datos.

¿Por qué es importante la arquitectura de datos?

A medida que las organizaciones escalan sus datos, la necesidad de una arquitectura bien estructurada y adaptable se ha vuelto primordial. Y, sin embargo, el 94 % de los líderes de datos mencionaron la ausencia de una arquitectura de datos definida entre sus principales desafíos1.

Una arquitectura de datos moderna puede ayudar a unificar y estandarizar los datos empresariales, lo que permite compartir datos sin problemas entre dominios empresariales. También proporciona una base escalable para casos de uso avanzados como el análisis de datos en tiempo real y la IA generativa, ayudando a los equipos a extraer valor de los datos de forma más rápida y fiable.

A medida que tecnologías como el Internet de las cosas (IoT) generan nuevas fuentes de datos, una arquitectura bien diseñada garantiza que los datos sigan siendo manejables, integrados y útiles durante todo su ciclo de vida. Puede reducir la redundancia, mejorar la calidad de los datos y ayudar a eliminar los silos conectando sistemas en toda la empresa.

Si se hace bien, la arquitectura de datos no es solo una estructura técnica: es una capacidad estratégica que convierte los datos sin procesar en un activo reutilizable.

Tipos de arquitectura de datos

La arquitectura de datos moderna tiende a seguir uno de los dos enfoques principales: centralizada o descentralizada. Estos modelos guían la forma en que se recopilan, almacenan y gobiernan los datos empresariales.

Las arquitecturas centralizadas llevan los datos a plataformas unificadas, como data lakes o almacenes de datos, gestionados bajo un único modelo de gobierno de datos. Esto ayuda a reducir la redundancia, mejorar la calidad de los datos y respaldar el modelado de datos estructurados utilizando lenguaje de consulta estructurado (SQL) y otras bases de datos relacionales.

Ventajas:

- Control y seguridad centralizados: Al tener todos los datos en un solo lugar, es más fácil aplicar políticas de seguridad, autenticación y encriptación.
- Facilidad de mantenimiento: Solo se necesita intervenir en un servidor para realizar actualizaciones o mantenimiento.
- Gestión eficiente de recursos: El procesamiento de datos se realiza en un único punto, lo que puede reducir costos operativos.
- Vistas unificadas de los datos: Se pueden crear modelos de datos integrados que facilitan el análisis global de la información.
- Menor complejidad técnica: No hay necesidad de sincronizar múltiples nodos o manejar replicación de datos.

Desventajas

- Dependencia crítica del servidor: Si el servidor central falla, todo el sistema se detiene.
- Escalabilidad limitada: A medida que crecen los datos o los usuarios, el rendimiento puede degradarse.
- Latencia para usuarios remotos: Los usuarios que están lejos del servidor pueden experimentar demoras en el acceso.
- Cuellos de botella: Todas las operaciones pasan por un único punto, lo que puede saturar el sistema en momentos de alta demanda.
- Costo de infraestructura: Requiere un servidor robusto y confiable, lo que puede implicar una inversión considerable.

Ejemplo: Sistema de gestión hospitalaria con Oracle Database

Contexto: Un hospital grande decide implementar un sistema centralizado para manejar toda su información clínica, administrativa y financiera.

Arquitectura

- **Servidor central**: Se instala Oracle Database en un servidor robusto ubicado en el centro de datos del hospital.
- **Clientes**: Las estaciones de trabajo en consultorios, recepción, farmacia y laboratorio acceden al sistema mediante una red local.
- **Procesamiento**: Toda la lógica de negocio (consultas, validaciones, procedimientos almacenados) se ejecuta en el servidor. Los clientes solo envían solicitudes y muestran resultados.

Este tipo de arquitectura sigue siendo común en instituciones donde la consistencia y el control centralizado son prioritarios.

Las arquitecturas descentralizadas distribuyen la propiedad de los datos entre los dominios empresariales. Los equipos gestionan los datos localmente, a menudo utilizando sistemas de bases de datos no relacionales (también

llamadas "bases de datos NoSQL") o pipelines basados en eventos con sus propios esquemas, metadatos y controles de acceso. Este enfoque admite casos de uso de integración y procesamiento de datos en tiempo real, transmisión de datos y machine learning (ML).

La mayoría de las organizaciones combinan ambos modelos para equilibrar la escalabilidad, la integración de datos y la agilidad. Este enfoque híbrido puede ayudar a admitir diferentes fuentes de datos, reducir los silos de datos y permitir operaciones nativas de la nube en plataformas como AWS o Microsoft Azure.

Independientemente del modelo arquitectónico que adopte una organización, el éxito depende de qué tan bien estén estructurados los datos subyacentes. Ahí es donde entra en juego el modelado de datos.

Ventajas

- Autonomía de dominios: Cada unidad (departamento, sede, servicio) gestiona sus propios datos, lo que permite mayor flexibilidad y rapidez en decisiones locales.
- Escalabilidad horizontal: Puedes añadir más nodos sin afectar el sistema global, ideal para grandes volúmenes de datos o usuarios.
- Reducción de cuellos de botella: Al distribuir la carga, se evita la saturación de un único servidor.
- Mayor resiliencia: Si un nodo falla, los demás pueden seguir operando, lo que mejora la disponibilidad.
- Adaptabilidad tecnológica: Cada nodo puede usar tecnologías distintas según sus necesidades (por ejemplo, bases relacionales en un nodo y NoSQL en otro).
- Velocidad de comercialización: Los equipos pueden desarrollar y desplegar soluciones más rápido sin depender de un centro único.

Desventajas

- Complejidad en la integración: Unir datos de múltiples fuentes requiere herramientas de federación o virtualización, lo que puede ralentizar el rendimiento.
- Gestión de consistencia: Mantener datos coherentes entre nodos es más difícil, especialmente si hay replicación o sincronización.
- Seguridad distribuida: Proteger múltiples puntos de acceso exige políticas más robustas y coordinadas.
- Costos operativos: Aunque escalable, requiere más infraestructura, monitoreo y personal técnico especializado.
- Desafíos en la gobernanza de datos: Definir quién es responsable de qué datos puede volverse confuso en entornos descentralizados.

Ejemplo: Tienda en línea con MySQL en múltiples sucursales

Contexto: Una cadena de tiendas tiene sucursales en distintas ciudades. Cada sucursal gestiona su propia base de datos local con MySQL, sin depender de un servidor central.

Cómo funciona

- Cada sucursal tiene su propia instalación de MySQL con los datos de sus productos, ventas y clientes.
- No hay un servidor central que controle todo: cada tienda opera de forma autónoma.
- Periódicamente, los datos se sincronizan entre sucursales mediante scripts o servicios de integración (por ejemplo, usando APIs REST o herramientas como SymmetricDS).

Pros

- Si una sucursal pierde conexión con otras, puede seguir operando sin problemas.
- Cada tienda puede adaptar su sistema a sus necesidades locales.
- Menor riesgo de caída total del sistema.

Este tipo de arquitectura es útil para negocios que necesitan independencia operativa pero aún desean compartir información de forma ocasional.

Las arquitecturas distribuidas son un grupo de datos que pertenecen a un sistema, pero a su vez está repartido entre ordenadores de una misma red, ya sea a nivel local o cada uno en una diferente localización geográfica, cada sitio en la red es autónomo en sus capacidades de procesamiento y es capaz de realizar operaciones locales y en cada uno de estos ordenadores debe estar ejecutándose una aplicación a nivel global que permita la consulta de todos los datos como si se tratase de uno solo

Ventajas

- Escalabilidad: Puedes añadir nuevos nodos fácilmente conforme crecen los datos o los usuarios2.
- Alta disponibilidad: Si un nodo falla, otros pueden seguir funcionando, lo que reduce el riesgo de interrupciones totales2.
- Acceso local más rápido: Los usuarios acceden a datos cercanos geográficamente, lo que mejora el rendimiento.
- Tolerancia a fallos: Los datos están replicados en varios lugares, lo que protege contra pérdidas por desastres.

- Flexibilidad geográfica: Ideal para empresas con oficinas en distintas ubicaciones, ya que cada sede puede operar con su propio nodo.
- Optimización de recursos: El procesamiento se distribuye, evitando cuellos de botella en un único servidor.

Desventajas

- Complejidad administrativa: Requiere conocimientos técnicos avanzados para configurar y mantener2.
- Seguridad más difícil de gestionar: Proteger múltiples nodos y canales de comunicación es más desafiante.
- Problemas de consistencia: Mantener los datos sincronizados entre nodos puede ser complicado.
- Redundancia de datos: Puede haber duplicación innecesaria si no se gestiona bien.
- Costos iniciales: Aunque puede ahorrar a largo plazo, la implementación inicial puede ser costosa.

Ejemplo: Red de bibliotecas con PostgreSQL distribuido

Contexto: Varias bibliotecas municipales comparten un sistema de gestión de libros, pero cada una tiene su propio servidor local con una base de datos PostgreSQL.

Cómo funciona

- Cada biblioteca almacena sus propios datos (libros disponibles, préstamos, usuarios) en su servidor local.
- Las bases de datos están conectadas por una red, y se sincronizan periódicamente para compartir información (por ejemplo, disponibilidad de libros entre sedes).
- Las consultas locales se hacen directamente en cada servidor, y las consultas globales se distribuyen entre nodos.

Pros

- Cada biblioteca puede seguir funcionando incluso si se cae la red.
- Las búsquedas locales son rápidas.
- Se puede escalar fácilmente añadiendo más bibliotecas al sistema.

Este tipo de arquitectura es ideal para organizaciones que necesitan autonomía local pero también quieren colaborar entre sedes.

RECOMENDACIÓN DE LA ARQUITECTURA MAS ADECUADA PARA UN SISTEMA DE GESTION ACADEMICA

Para un sistema de gestión académica, la arquitectura más adecuada depende del tamaño de la institución, sus necesidades de disponibilidad, escalabilidad y presupuesto. Pero si buscamos un equilibrio entre simplicidad, eficiencia y crecimiento futuro, aquí va una comparación clara y una recomendación:

La arquitectura que se recomendaría en este caso sería: Arquitectura distribuida

¿Por qué?

La arquitectura distribuida es:

- Escalable: Puedes empezar con una base central y luego distribuir por sedes o facultades.
- Alta disponibilidad: Si una sede pierde conexión, otras pueden seguir funcionando.
- Acceso rápido: Los estudiantes y docentes acceden a datos desde nodos cercanos.
- Modularidad: Puedes dividir el sistema por funciones (matrícula, notas, asistencia) y distribuirlas.

además, haciendo una comparación rápida

Arquitectura	Ideal para...	Limitaciones principales
Centralizada	Instituciones pequeñas o con una sola sede	Riesgo de caída total, difícil de escalar
Distribuida	Universidades con varias sedes o facultades	Requiere buena sincronización y diseño
Descentralizada	Redes de instituciones autónomas	Complejidad alta, difícil de integrar datos

Entonces, una universidad con tres campus puede usar PostgreSQL en cada sede, sincronizados mediante réplicas o APIs. Cada campus gestiona sus datos localmente, pero el sistema permite consultas globales (como ver el historial académico completo de un estudiante).