UNIVERSIDAD PERUANA LOS ANDES FACULTAD DE INGENIERÍA

"ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN"

Base de Datos II

ARQUITECTURA CENTRALIZADA

Alumno: SORIANO TIMOTEO Joel Kevin

Catedrático: RAUL FERNANDEZ Bejarano

Ciclo: V

Huancayo - Perú - 2025

Arquitectura de Bases de Datos

¿Qué es la arquitectura de datos?

La arquitectura de datos define cómo se **gestionan los datos** desde su origen (recolección) hasta su uso final (consumo). Incluye procesos de **almacenamiento, transformación y distribución**, estableciendo un plan para el flujo de datos en los sistemas.

Es la base de operaciones de procesamiento de datos y de aplicaciones de **analítica e inteligencia artificial (IA)**.
El diseño depende de los **requerimientos del negocio** y de las necesidades de datos, que los arquitectos e ingenieros utilizan para definir modelos y estructuras de datos.

Una arquitectura bien diseñada **convierte los datos en un activo estratégico**, útil para reportes, análisis o proyectos de ciencia de datos.

¿Por qué es importante?

En organizaciones modernas, donde el volumen de datos crece sin parar, contar con una arquitectura clara es esencial.
Un 94% de líderes de datos reconocen que **no tener una arquitectura definida es uno de sus principales problemas**.

Beneficios de una arquitectura de datos moderna:

- **Unificación** de información entre áreas.
- **Estándares comunes** para compartir datos.
- **Escalabilidad** para soportar casos de uso avanzados como análisis en tiempo real o IA generativa.
- **Calidad y confiabilidad** en la información.
- Reducción de **duplicidad** y eliminación de silos.

En resumen: no es solo algo técnico, es una **ventaja estratégica**.

Tipos de arquitectura de datos

Existen tres enfoques principales:

1. Arquitectura centralizada

Consiste en almacenar los datos en una plataforma unificada (data warehouse, data lake, Oracle, etc.), bajo un único modelo de gobierno.

Ventajas:

- Control central de seguridad.
- Fácil mantenimiento.
- Vistas unificadas.
- Menor complejidad técnica.
- Procesamiento eficiente en un único punto.

Desventajas:

- Dependencia de un solo servidor (si falla, se cae todo).
- Escalabilidad limitada.
- Latencia para usuarios lejanos.
- Cuellos de botella en picos de uso.
- Requiere infraestructura robusta.

Ejemplo:
Sistema hospitalario con Oracle Database: un servidor central almacena información clínica, administrativa y financiera, y todas las áreas acceden desde clientes conectados en red.

2. Arquitectura descentralizada

Los datos se distribuyen entre diferentes dominios (por ejemplo, sucursales o departamentos), y cada uno gestiona sus propios datos de forma local. Suelen usarse bases **NoSQL** o pipelines de eventos.

Ventajas:

- Autonomía local.

- Escalabilidad horizontal.
- Menos cuellos de botella.
- Resiliencia: si un nodo falla, otros siguen operando.
- Flexibilidad tecnológica.

Desventajas:

- Integración de datos compleja.
- Mayor dificultad para mantener consistencia.
- Seguridad más dispersa.
- Costos de infraestructura y gestión.

Ejemplo:
Cadena de tiendas con MySQL local en cada sucursal. Cada sede opera independiente, pero periódicamente sincroniza información con las demás.

3. Arquitectura distribuida

Los datos se reparten en varios nodos de una red (pueden estar en distintas ubicaciones geográficas). Cada nodo es autónomo, pero forma parte de un sistema global.

Ventajas:

- Escalabilidad.
- Alta disponibilidad.
- Acceso local más rápido.
- Tolerancia a fallos.
- Flexibilidad geográfica.

Desventajas:

- Mayor complejidad técnica.
- Seguridad difícil de unificar.
- Problemas de consistencia.
- Redundancia de datos si no se gestiona bien.
- Costos iniciales altos.

Ejemplo:
Red de bibliotecas con PostgreSQL distribuido. Cada sede maneja su propio catálogo y se sincroniza con las demás para consultas globales.

Recomendación para un Sistema de Gestión Académica

Tras comparar los modelos, la arquitectura más adecuada es la **arquitectura distribuida**.

¿Por qué?

- Es **escalable**, ideal para crecer con nuevos campus o facultades.
- Permite **alta disponibilidad**, asegurando continuidad incluso si un nodo falla.
- Proporciona **acceso local más rápido** a estudiantes y docentes.
- Posibilita dividir funciones (notas, matrículas, asistencia) en módulos distribuidos.

Ejemplo de aplicación en una universidad:
Cada campus tiene su propio servidor PostgreSQL, pero sincronizados entre sí mediante réplicas o APIs.
De este modo, un alumno puede consultar su historial completo, aunque haya cursado en distintos campus.