Bases de Datos Distribuidas

M.C.C. Eliezer Alcázar Silva

¿Qué son las Bases de datos distribuidas?

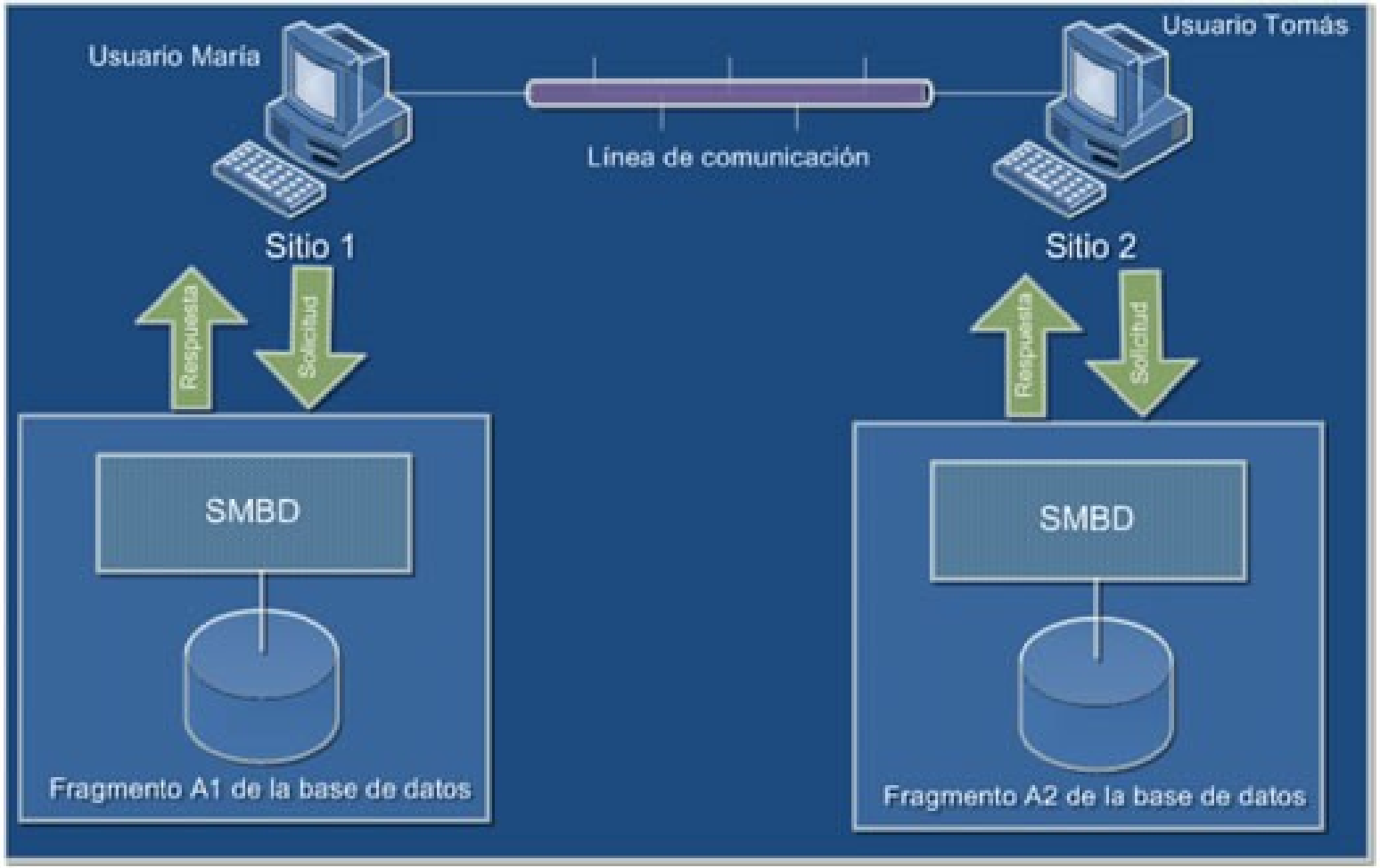
Una **base de datos distribuida**, o por sus siglas en inglés **DDB** (Distributed Database), la podemos entender como una base de datos tradicional dividida en diferentes partes físicamente dispersas y que se acceden de forma lógica, tal como se accede a una base de datos centralizada por medio de un Sistema de Administración de Bases de Datos.

¿Qué son las Bases de datos distribuidas?

Un **Sistema de Administración de Bases de Datos Distribuidas**, o por si siglas en inglés **DDBMS** (Distributed Database Managment System), rige el almacenamiento y procesamiento de datos lógicamente relacionados a través de sistemas de computadoras interconectadas, en las cuales tanto las funciones de datos como de procesamiento se distribuyen entre varios sitios (Rob, Peter 2004)

¿Qué son las Bases de datos distribuidas?

Un Sistema de Administración de Base de Datos Distribuida debe realizar todas las funciones de un sistema de administración de base de datos centralizado y manejar todas las funciones necesarias impuestas por la distribución de los datos y procesamiento; además, debe realizar estas funciones adicionales *transparentemente* para el usuario.

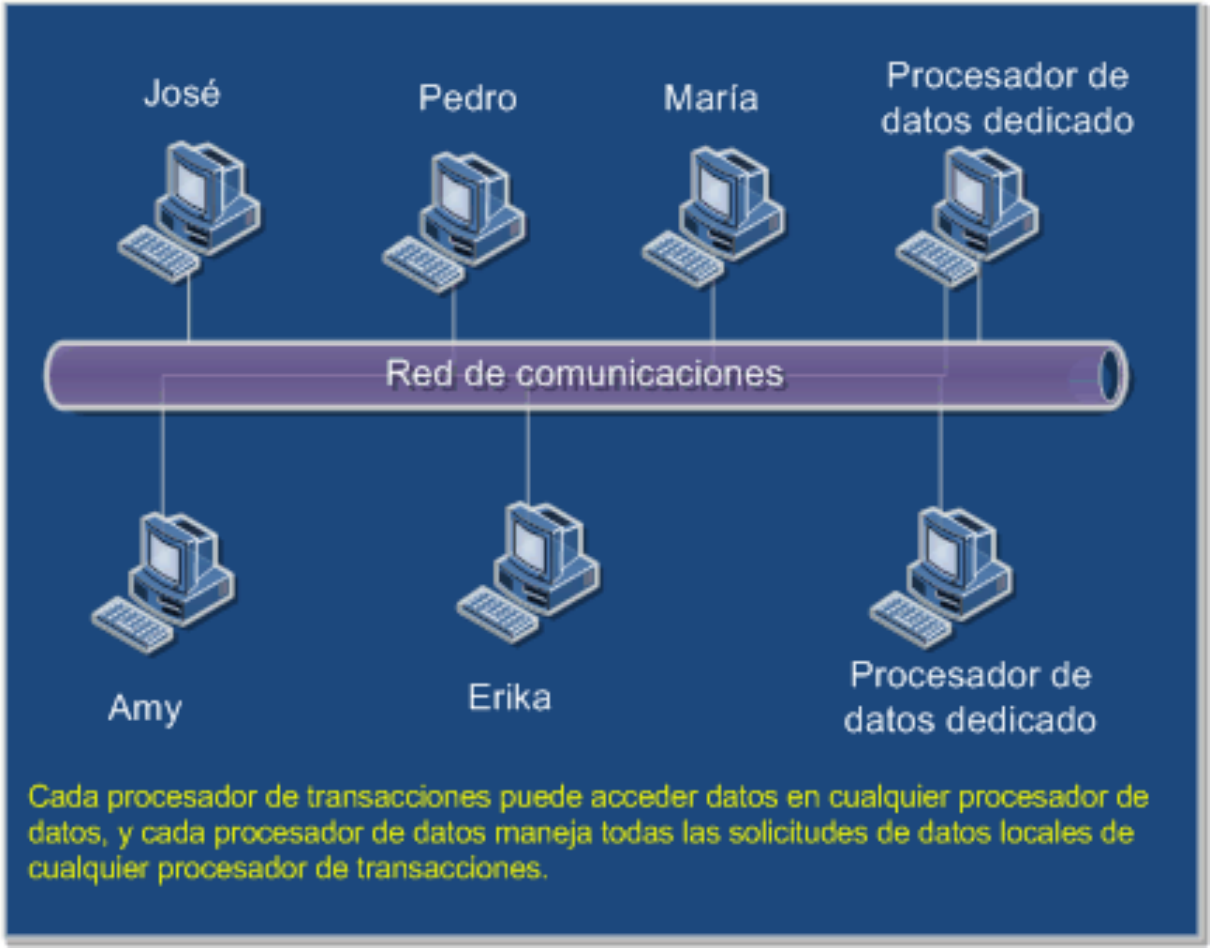

Usuario María
Usuario Tomás
Línea de comunicación
Sitio 1
Sitio 2
Respuesta
Solicitud
Respuesta
Solicitud
SMBD
SMBD
Fragmento A1 de la base de datos
Fragmento A2 de la base de datos

Componentes de una Bases de datos distribuidas

De acuerdo con Peter Rob (2004), el Sistema de Administración de Bases de Datos Distribuidas debe incluir (por lo menos) los siguientes componentes:

- ***Estaciones de trabajo** (sitios y nodos) que formen el sistema de red. El Sistema de Bases de Datos Distribuidas debe ser independiente del hardware del sistema de computadoras.*
- ***Componentes de software y hardware** residentes en cada estación de trabajo. Los componentes de red permiten que todos los sitios interactúen e intercambien datos.*

- ***Medios de comunicación*** *que transporten los datos de una estación de trabajo a otra. El Sistema de Administración de bases de Datos Distribuidas debe ser independiente de los medios de comunicación; es decir, debe ser capaz de soportar varios tipos de medios de comunicación.*
- ***El Procesador de Transacciones*** *(**TP**, por sus siglas en inglés), el cual es el componente de software encontrado en cada computadora que solicita datos. El PT recibe y procesa las solicitudes de datos de la aplicación (remotas y locales); el TP también es conocido como **Procesador de Aplicaciones** (**AP**, por sus siglas en inglés) o como **Administrador de Transacciones** (**TM**, por sus siglas en inglés).*
- ***El Procesador de Datos*** *(**DP**, por sus siglas en inglés), el cual es el componente de software residente en cada computadora que guarda y recupera datos localizados en el sitio. El DP también se conoce como **administrador de datos** (**DM**); incluso, un procesador de datos puede ser un DBM centralizado.*


José
Pedro
María
Procesador de datos dedicado
Red de comunicaciones
Amy
Erika
Procesador de datos dedicado
Cada procesador de transacciones puede acceder datos en cualquier procesador de datos, y cada procesador de datos maneja todas las solicitudes de datos locales de cualquier procesador de transacciones.

DDBMS

De acuerdo con Peter Rob (Rob, 2004), un DDBMS rige el almacenamiento y procesamiento de datos lógicamente relacionados a través de sistemas de computadoras interconectadas, en las cuales tanto las funciones de datos como de procesamiento se distribuyen entre varios sitios. De esta manera, un DDBMS debe contar por lo menos con las siguientes características o funciones para ser considerado como distribuido:

- ***Interface de aplicación*** para interactuar con el usuario final o con programas de aplicación y con otros sistemas de administración de base de datos (DBMS) dentro de la base de datos distribuida.
- ***Validación*** para analizar las solicitudes de datos.
- ***Transformación*** para determinar qué componentes de solicitud de datos se distribuyen y cuáles son locales.
- ***Optimización de consultas*** para encontrar la mejor estrategia de acceso. (¿Cuáles fragmentos deben ser accesados por la consulta y cómo, si las hay, se deben sincronizar las actualizaciones de los datos?)
- ***Mapeo*** para determinar la ubicación de los datos de fragmentos locales y remotos.
- ***Interface de E/S*** para leer o escribir datos de o en medios de almacenamiento locales y permanentes.

- ***Formateo*** para presentar los datos para su representación al usuario final o un programa de aplicación.
- ***Seguridad*** para proporcionar privacidad tanto en bases de datos locales como en remotas.
- ***Respaldo y recuperación*** para garantizar la disponibilidad y recuperabilidad de la base de datos en caso de una falla.
- ***Administración de base de datos*** para el administrador de la base de datos.
- ***Control de concurrencia*** para manejar el acceso simultáneo a los datos y para garantizar su consistencia a través de los fragmentos en el DDBMS.
- ***Manejo de transacciones*** para garantizar que los datos pasen de un estado consistente a otro. Esta actividad incluye la sincronización de transacciones locales y remotas, lo mismo que transacciones a través de segmentos múltiples distribuidos.

- Un **DDBMS** debe realizar todas las funciones de un sistema de administración de base de datos centralizado, y además manejar todas las funciones necesarias impuestas por la distribución de los datos y procesamiento, además de realizar estas funciones adicionales *transparentemente* para el usuario.

Evolución en la gestión de datos

Décadas de 1950 y 1960: Almacenamiento manual y secuencial

- Uso de archivos, tarjetas perforadas y cintas magnéticas para guardar datos en medios físicos.
- Procesamiento por lotes: operaciones manuales y acceso secuencial, limitando la velocidad y flexibilidad.

Evolución en la gestión de datos

Década de 1970: Primeros SGBD y modelos primitivos

- Surgimiento de los Sistemas de Gestión de Bases de Datos (SGBD).
- Modelos jerárquicos y de red: permitían estructuras complejas, pero con acceso rígido y relaciones predefinidas.

Evolución en la gestión de datos

Década de 1980: La revolución relacional

- Bases de datos relacionales (RDBMS), introducidas por Edgar F. Codd, usando tablas estructuradas (filas y columnas).
- Aparición de SQL, que estandarizó consultas y manipulación de datos, aumentando la flexibilidad.

Décadas de 1990 y 2000: Inteligencia de negocios y el auge de la web

- Bases de datos orientadas a objetos para datos complejos como imágenes y videos.
- Almacenes de datos (Data Warehouses) para centralizar información de múltiples fuentes y facilitar análisis.
- Bases de datos para la web, impulsadas por la expansión de Internet y el acceso vía navegadores.

Década de 2010: El Big Data y la nube

- Explosión del Big Data debido a redes sociales, IoT y grandes volúmenes de datos no estructurados.
- Sistemas distribuidos como Hadoop y Spark para procesamiento escalable.
- Bases de datos NoSQL para gestionar datos diversos y veloces.
- Computación en la nube (AWS, Azure) ofreciendo soluciones escalables de almacenamiento y procesamiento.

Década de 2020: IA, análisis en tiempo real y gobernanza

- Integración de IA y Machine Learning para análisis predictivo y decisiones automáticas.
- Análisis en tiempo real para respuestas inmediatas a cambios del mercado.
- Arquitectura de malla de datos (Data Mesh): enfoque descentralizado en gestión de datos.
- Énfasis en gobernanza, calidad, seguridad y cumplimiento normativo en ecosistemas de datos complejos.

- Los sistemas de base de datos actuales se clasifican con base en cómo la distribución de los procesos y datos son soportados; por ejemplo, un **Sistema de Administración de Bases de Datos** (**DBMS**) puede guardar datos en un solo sitio (**DB centralizada**) o en múltiples sitios (**DB distribuida**) y puede soportar procesamiento de datos en un solo sitio o en varios.

	Datos en un solo sitio	Datos en sitios múltiples
Proceso en un solo sitio	Un solo DBMS anfitrión	No aplicable (requiere procesos múltiples)
Proceso en múltiples sitios	Servidor de archivos Varios DBMS de LAN	DDBMS Cliente/Servidor totalmente distribuido

Los sistemas de administración de base de datos distribuida (DDBMS) ofrecen varias ventajas en relación con los sistemas tradicionales

- ***Los datos se localizan cerca del sitio de "mayor demanda"***. Los datos en un sistema de base de datos distribuida se dispersan de acuerdo con los requerimientos del negocio.
- ***Acceso más rápido a los datos***. Los usuarios a menudo trabajan sólo con un subconjunto de los datos de la compañía. Si tales subconjuntos de datos se guardan y acceden localmente, el sistema de base de datos permitirá un acceso a los datos más rápido que con datos centralizados remotamente localizados.
- ***Procesamiento más rápido de los datos***. Un sistema de base de datos distribuida hace posible procesar datos en varios sitios, con lo que se reparte la carga de trabajo del sistema.
- ***Facilitación del crecimiento***. Pueden agregarse sitios nuevos a la red sin afectar las operaciones de otros sitios. Tal flexibilidad permite que la compañía se expanda con relativa facilidad y rapidez.

- ***Comunicaciones mejoradas***. Como los sitios locales son más pequeños y están más cerca de los clientes, promueven una mejor comunicación entre departamentos y entre los clientes y el personal de la compañia. La mejor y más rápida comunicación, con frecuencia ayuda a mejorar los sistemas de información; por ejemplo, una operación de cuentas por pagar local utiliza los datos del Departamento de Ventas directamente, sin tener que depender de los reportes tardados de la oficina central.
- ***Costos de operación reducidos***. Es mucho más barato agregar estaciones a una red que actualizar un sistema *mainframe*. El costo de las líneas de comunicación de datos dedicadas y el software de *mainframe* se reducen proporcionalmente. El trabajo de desarrollo se realiza con más rapidez y menor costo con computadoras personales baratas que con *mainframes*. Las *mainframes* no están condenadas a desaparecer de la escena de la base de datos, su poder no puede ser desconectado; no obstante, la existencia de redes de computadoras personales hace posible distribuir las cargas de trabajo con más prudencia y reservar las costosas *mainframes* para tareas de procesamiento más especializadas.

- ***Interface de usuario fácil de usar***. Las computadoras personales y las estaciones de trabajo en general vienen equipadas con una interface de usuario gráfica (GUI, por sus siglas en inglés), la cual simplifica el uso y el entrenamiento de los usuarios finales.
- ***Menos peligro de falla en un solo punto***. En un sistema centralizado, la falla de una *mainframe* acaba con todas las operaciones del sistema; por el contrario, un sistema distribuido es capaz de desplazar las operaciones cuando falla una de las computadoras. La carga de trabajo del sistema es absorbida por otras estaciones de trabajo porque una de las características del sistema distribuido es que los datos existen en múltiples sitios.
- ***Independencia del procesador***. El usuario final es capaz de acceder cualquier copia disponible de los datos y la solicitud de un usuario es procesada por cualquier procesador disponible en la ubicación de los datos. En otras palabras, las solicitudes no dependen de un procesador específico, cualquier procesador disponible puede manejar dicha solicitud.

Toda base de datos distribuida debe estar diseñada con base en los 12 objetivos planteados por C. J. Date (Date, 2001) en 1987.

Los objetivos, reglas o mandamientos de Date describen una base de datos totalmente distribuida y, aunque ningún sistema de administración de base de datos distribuida (DDBMS) actual se adapta a todos ellos, las reglas sí constituyen un útil objetivo de base de datos distribuida. Los 12 objetivos son (Date, 2001):

1. ***Independencia del sitio local***. Cada sitio local puede actuar como un sistema de administración de base de datos (DBMS) independiente, autónomo y centralizado. Cada sitio es responsable de la seguridad, el control de concurrencia, el respaldo y la recuperación.
2. ***Independencia del sitio central***. Ningún sitio en la red depende de un sitio central o de cualquier otro sitio. Todos los sitios tienen las mismas capacidades.
3. ***Independencia de fallas***. El sistema no se ve afectado por fallas de nodos. El sistema continúa operando, incluso en el caso de una falla de nodo o de una expansión de la red.
4. ***Transparencia de ubicación***. El usuario ve sólo una base de datos lógica. La fragmentación de los datos es transparente para el usuario. El usuario no necesita saber la ubicación de los datos para recuperarlos.

5.- Transparencia de fragmentación. El usuario ve sólo una base de datos lógica. La fragmentación de los datos es transparente para el usuario. El usuario no necesita conocer el nombre de los fragmentos de la base de datos para recuperarlos.

6.- Transparencia de replicación. El usuario ve sólo una base de datos lógica. El DDBMS selecciona de una manera transparente el fragmento que va a acceder; además, maneja todos los fragmentos transparentemente ante el usuario.

7.- Procesamiento de consulta distribuida. Una consulta distribuida puede ser ejecutada en varios sitios diferentes de procesamiento de datos. La optimación de las consultas es realizada transparentemente por el usuario.

8.- Procesamiento de transacciones distribuidas. Una transacción puede actualizar datos en varios sitios diferentes. La transacción es transparente ejecutada en varios sitios diferentes de procesamiento de datos.

9.- Independencia del hardware. El sistema debe funcionar en cualquier plataforma de hardware.

10.- Independencia del sistema operativo. El sistema debe funcionar con cualquier plataforma de software de sistema operativo.

11.- Independencia de la red. El sistema debe funcionar con cualquier plataforma de red.

12.- Independencia de la base de datos. El sistema debe soportar cualquier producto de base de datos provisto por cualquier proveedor.

¿Qué son los Modelos de Arquitectura en DDBMS?

Ejemplos de sistemas de tipo BASE incluyen:

Los modelos arquitectónicos para DDBMS (o múltiples DBMS) se clasifican en tres dimensiones principales:

1. **Autonomía**
2. **Distribución**
3. **Heterogeneidad**

Estas dimensiones determinan cómo se organizan los datos, el control y los componentes del sistema en los diferentes sitios.

Distribución

Indica la **distribución física** de los datos entre los distintos sitios.

Define cómo se almacenan y acceden los fragmentos o réplicas de datos.

Afecta el rendimiento, la escalabilidad y la disponibilidad de la información.

Autonomía

Señala el **grado de independencia** que posee cada sistema o sitio dentro del DDBMS.

Refleja cómo se distribuye el control del sistema de bases de datos.

Cuanta más autonomía tiene un sitio, menor coordinación central se requiere.

Heterogeneidad

Se refiere a las **diferencias entre los modelos de datos, software de DBMS y componentes de hardware.**

Puede implicar distintos lenguajes de consulta, estructuras de datos u operaciones del sistema.

El sistema debe gestionar la **integración y traducción** entre los diferentes componentes.

Autonomía

1. Según el grado de autonomía

Este criterio se refiere a **cuánta independencia tienen las bases de datos locales** dentro del sistema distribuido.

🔹 a) Sistema Centralizado (sin autonomía)

- Existe **una sola base de datos** y un solo sitio de control.
- Los usuarios acceden remotamente, pero **toda la gestión se hace desde un único nodo**.
- **Ejemplo:** Un servidor principal con múltiples clientes conectados.

🔹 b) Sistema Distribuido Tightly Coupled (autonomía parcial)

- Los nodos **trabajan cooperativamente bajo un mismo control de transacciones y catálogo global**.
- Hay una **coordinación fuerte** entre los nodos, pero **las bases locales conservan algo de control** sobre sus datos.
- **Ejemplo:** Un sistema de sucursales bancarias donde todas usan la misma política de acceso y transacciones coordinadas.

🔹 c) Sistema Federado (autonomía alta)

- Cada base de datos **mantiene su propio esquema, control y administrador**, pero colabora mediante una **capa de federación o mediador**.
- Se permite la **integración lógica de datos heterogéneos** sin perder la independencia local.
- **Ejemplo:** Varias universidades que comparten un sistema para consultar información académica sin modificar sus bases internas.

Heterogeneidad

2. Según la heterogeneidad

Este criterio evalúa **las diferencias en hardware, software o modelos de datos** entre los nodos del sistema.

🔹 a) Homogéneas

- Todos los nodos usan el **mismo SGBD (Sistema Gestor de Base de Datos)** y el **mismo modelo de datos**.
- Simplifica la coordinación y las consultas distribuidas.
- **Ejemplo:** Varias instancias de PostgreSQL o CockroachDB distribuidas en red local.

🔹 b) Heterogéneas

- Los nodos pueden usar **diferentes SGBD** (por ejemplo, Oracle, MySQL, MongoDB) o **modelos distintos** (relacional, objeto, documento).
- Requiere **traductores o mediadores** para comunicar los sistemas.
- **Ejemplo:** Un sistema que integra información de ventas en MySQL y logística en MongoDB.

Distribución

3. Según el grado de distribución

Este criterio mide **cómo se distribuyen físicamente los datos y el procesamiento** entre los nodos.

🔹 a) No distribuidas

- Todos los datos residen en un solo sitio.
- No existe fragmentación ni replicación.
- **Ejemplo:** Base de datos tradicional instalada en un único servidor.

🔹 b) Parcialmente distribuidas

- Algunas partes de la base de datos se **replican o fragmentan** entre nodos, pero la coordinación se maneja centralmente.
- **Ejemplo:** Un sistema con base de datos maestra y varias réplicas de solo lectura.

🔹 c) Totalmente distribuidas

- **Todos los nodos son autónomos y contienen fragmentos de datos** (horizontal o verticalmente distribuidos).
- No hay un nodo central; la **coordinación es colaborativa**.
- **Ejemplo:** Sistemas como **CockroachDB** o **Google Spanner**, donde todos los nodos actúan como iguales.

Autonomía	Heterogeneidad	Distribución	Modelo	Características
0	0	0	Centralizado clásico	Todos los datos y control en un único nodo; homogéneo, sin distribución
1	0	0	Autonomía local sin distribución	Nodos con control local limitado, pero datos permanecen centralizados; homogéneo
2	0	0	Autonomía total sin distribución	Nodos completamente autónomos, datos centralizados; homogéneo
0	0	1	Distribución parcial sin autonomía	Datos parcialmente replicados o fragmentados; control centralizado; homogéneo
1	0	1	Distribución parcial con autonomía parcial	Nodos con algo de control, datos parcialmente distribuidos; homogéneo
2	0	1	Distribución parcial con autonomía total	Nodos autónomos, datos parcialmente distribuidos; homogéneo
0	0	2	Distribución total sin autonomía	Control central, datos totalmente fragmentados y replicados; homogéneo

1	0	2	Distribución total con autonomía parcial	Nodos con autonomía parcial, datos totalmente distribuidos; homogéneo
2	0	2	Distribución total con autonomía total	Nodos autónomos, datos fragmentados y replicados; homogéneo
0	1	0	Sistema centralizado heterogéneo	Datos centralizados pero con distintos SGBD o modelos de datos; sin autonomía
1	1	0	Autonomía parcial heterogénea	Nodos parcialmente autónomos, datos centralizados, heterogéneo
2	1	0	Autonomía total heterogénea	Nodos completamente autónomos, datos centralizados, heterogéneo
0	1	1	Distribución parcial heterogénea	Datos parcialmente distribuidos, control centralizado, heterogéneo
1	1	1	Distribución parcial heterogénea con autonomía parcial	Nodos parcialmente autónomos, datos parcialmente distribuidos, heterogéneo
2	1	1	Distribución parcial heterogénea con autonomía total	Nodos autónomos, datos parcialmente distribuidos, heterogéneo

0	1	2	Distribución total heterogénea	Control central, datos totalmente distribuidos, heterogéneo
1	1	2	Distribución total heterogénea con autonomía parcial	Nodos parcialmente autónomos, datos totalmente distribuidos, heterogéneo
2	1	2	Distribución total heterogénea con autonomía total	Nodos totalmente autónomos, datos totalmente distribuidos, heterogéneo

Modelos Arquitectónicos Comunes

Arquitectura Cliente-Servidor para DDBMS

Arquitectura Peer-to-Peer (P2P) para DDBMS

Arquitectura Multi-DBMS

Arquitectura Cliente-Servidor

- Es una arquitectura de **dos niveles** que divide las funciones entre **servidores** y **clientes.**
- **Funciones del servidor:** gestión de datos, procesamiento de consultas, optimización y administración de transacciones.
- **Funciones del cliente:** interfaz de usuario, verificación de consistencia y gestión parcial de transacciones.

Subtipos:

- Un Servidor – Múltiples Clientes
- Múltiples Servidores – Múltiples Clientes

Arquitectura Peer-to-Peer (P2P) para DDBMS

- Cada nodo o “peer” actúa **tanto como cliente como servidor**, compartiendo y coordinando servicios de bases de datos.
- Promueve la cooperación y el uso compartido de recursos entre los nodos.
- Se organiza en **cuatro niveles de esquemas:**
    a. Esquema Conceptual Global
    b. Esquema Conceptual Local
    c. Esquema Interno Local
    d. Esquema Externo

Arquitectura Multi-DBMS

Es un **sistema integrado** formado por dos o más bases de datos autónomas.

Proporciona una vista unificada sin perder la independencia de cada base local.

Se define a través de **seis niveles de esquemas:**

1. Nivel de Vista Multibase de Datos
2. Nivel Conceptual Multibase de Datos
3. Nivel Interno Multibase de Datos
4. Nivel de Vista de Base de Datos Local
5. Nivel Conceptual de Base de Datos Local
6. Nivel Interno de Base de Datos Local

Tipos de Bases de datos distribuidas

Una base de datos distribuida puede tomar muchas formas diferentes. La mejor opción para una organización dependerá de su caso de uso principal. Las formas que puede adoptar una base de datos distribuida incluyen las siguientes:

Homogénea. Se refiere a una base de datos distribuida en la que todos los nodos de la red utilizan la misma base de datos y sistema de gestión de bases de datos (DBMS), y comparten la misma estructura de datos.

Heterogénea. Aquí, los nodos en la red podrían usar diferentes sistemas de gestión de bases de datos y tener distintas estructuras de datos, lo que requiere una aplicación para traducir e integrar los datos de varios sistemas. Un sistema de base de datos heterogéneo puede resultar de agregar nuevas funcionalidades a una base de datos heredada o de integrar rápidamente diferentes sistemas, como cuando una organización adquiere a otra.

Tipos de Bases de datos distribuidas

Base de datos federada. Una base de datos federada permite que múltiples bases de datos trabajen juntas como si fueran una sola. Cada base de datos mantiene su autonomía, pero puede compartir datos y colaborar con todas las bases en el sistema federado.

Base de datos particionada. En una base de datos particionada, los datos se dividen en partes más pequeñas y manejables llamadas particiones, y cada partición se almacena en un nodo diferente de la red.

Tipos de Bases de datos distribuidas

Híbrida. Una base de datos distribuida híbrida combina elementos de bases de datos federadas y particionadas. Puede incluir bases de datos autónomas que cooperan y datos particionados que están distribuidos en diferentes nodos.

Replicada. En una base de datos replicada, las copias de los datos se almacenan en múltiples nodos para permitir alta disponibilidad y tolerancia a fallos. Para mantener la coherencia, los cambios en los datos se propagan de manera inmediata a las réplicas de respaldo.

Aplicaciones de las BD Distribuidas

Las organizaciones optan por ejecutar bases de datos distribuidas por muchas razones. El beneficio principal es el balanceo de carga, que permite que el sistema divida automáticamente la demanda elevada, ya sea de transacciones o análisis, a través de múltiples instancias para evitar que un solo servidor se convierta en un cuello de botella.

Aplicaciones de las BD Distribuidas

Estas son otras formas en las que estos sistemas pueden ayudar a abordar los muchos desafíos y requisitos que enfrentan las organizaciones hoy en día.

- **Distribución global de datos.** Para aplicaciones con alcance global, una base de datos distribuida es esencial. En primer lugar, permite que el sistema almacene los datos más cerca de los usuarios, reduciendo la latencia tanto en lecturas como en escrituras. Esto mejora la experiencia del usuario, especialmente para aplicaciones diseñadas para ofrecer rendimiento en tiempo real. La resiliencia también es una consideración importante. Si un centro de datos o servidor en una región sufre una interrupción, los usuarios en otras regiones aún pueden acceder a los datos y a la aplicación desde nodos operativos locales.
- **Escalabilidad eficiente.** Las bases de datos distribuidas permiten una escalabilidad horizontal rápida, ya que los administradores pueden añadir más nodos al sistema en respuesta a un aumento en el volumen de datos, solicitudes más complejas y más usuarios concurrentes. Además, esta escalabilidad puede configurarse para que ocurra de manera autónoma.

Aplicaciones de las BD Distribuidas

- **Localidad de datos y cumplimiento.** Para las organizaciones con estrictas reglas de gobernanza de datos o requisitos regionales de residencia de datos, las bases de datos distribuidas permiten a los administradores almacenar los datos en la ubicación geográfica donde se generan, posibilitando que estas empresas cumplan con las normativas locales.
- **Alta disponibilidad y tolerancia a fallos.** Una base de datos distribuida es fundamental para cargas de trabajo críticas, ya que al replicar los datos en múltiples nodos, estos permanecen disponibles incluso si fallan ciertos nodos. La mayoría de las arquitecturas de recuperación ante desastres se basan en un modelo de base de datos distribuida: tras un evento catastrófico, los datos pueden recuperarse rápidamente desde un nodo o una región remota.
- **Rendimiento.** Con sus datos y cargas de trabajo distribuidos en múltiples servidores, un sistema de bases de datos puede mejorar el rendimiento reduciendo la carga en cualquier nodo individual. Esto puede conducir a respuestas más rápidas en las consultas y a un mejor rendimiento general del sistema.

Ejemplos de Bases de datos distribuidas

El modelo de base de datos distribuida sobresale por ofrecer mayor capacidad de procesamiento a través de la escalabilidad horizontal, la localidad de datos y la tolerancia a fallos. Sin embargo, la base de datos específica que una organización elige, generalmente, dependerá de las necesidades y limitaciones de las aplicaciones de la empresa. Algunos casos de uso requieren la consistencia absoluta de una base de datos relacional compatible con ACID, mientras que otros son más adecuados para una base de datos tipo BASE que ofrece consistencia eventual. Veamos algunos ejemplos de cada una.

¿Qué es una base de datos ACID?

- **ACID** significa:
  - **Atomicidad**: Las operaciones se completan por completo o no se hacen.
  - **Consistencia**: La base de datos permanece en un estado válido antes y después de la transacción.
  - **Aislamiento**: Las transacciones se ejecutan de manera independiente.
  - **Durabilidad**: Los cambios realizados en la base de datos son permanentes, incluso en caso de fallos.
- **Ideal para**:
  - Aplicaciones que requieren una alta precisión y consistencia, como sistemas bancarios, comercio electrónico, sistemas de inventarios.

¿Qué es una base de datos BASE?

- **BASE** significa:
  - **Basically Available**: La base de datos está disponible la mayoría del tiempo.
  - **Soft state**: El estado puede cambiar con el tiempo, incluso sin nuevas entradas.
  - **Eventual consistency**: La consistencia se alcanza en el tiempo, no inmediatamente.

- **Ideal para**:
  - Aplicaciones distribuidas y escalables que toleran cierta inconsistencia temporal, como redes sociales, sistemas de análisis, big data.

Como elegir

¿Por qué elegir ACID?

- Necesidad de **transacciones confiables y precisas**.
- Datos críticos donde los errores son inaceptables.
- Requisitos de **consistencia fuerte** y reglas estrictas.
- Ejemplos: sistemas bancarios, comercio electrónico, registros médicos.

¿Por qué elegir BASE?

- Sistemas que necesitan **gran escalabilidad y disponibilidad**.
- Toleran cierta **inconsistencia temporal**.
- Ideal para **big data**, redes sociales y aplicaciones distribuidas.
- Ejemplos: plataformas en la nube, sistemas de análisis en tiempo real.

Diferencias clave entre ACID y BASE

Característica	ACID	BASE
Consistencia	Garantizada en cada transacción	Eventual, puede variar temporalmente
Escalabilidad	Vertical (aumenta la potencia del servidor)	Horizontal (añadir más nodos)
Rendimiento	Menor, por la garantía de coherencia	Mayor, por la flexibilidad y escalabilidad
Uso típico	Sistemas críticos, empresariales	Sistemas distribuidos, big data

Ejemplos de Bases de datos distribuidas

Ejemplos de sistemas que cumplen con las propiedades ACID incluyen:

Oracle Database: Oracle Database facilita bases de datos multimodelo distribuidas a escala global y lineal, que soportan datos estructurados y no estructurados. A través de acuerdos multicloud exclusivos con Microsoft Azure, AWS y Google Cloud, el servicio de base de datos distribuida de Oracle opera en los centros de datos de estos hyperscalers para unificar la gestión y las operaciones, ayudando a eliminar la latencia de red y el costo de mover datos entre nubes.

Google Cloud Spanner: Cloud Spanner es un servicio de base de datos distribuida globalmente y escalable horizontalmente que funciona en Google Cloud. El servicio proporciona distribución global, consistencia fuerte y particionado automático.

CockroachDB: Esta base de datos SQL distribuida presenta transacciones ACID, arquitectura distribuida y despliegues en múltiples regiones.

YugabyteDB: Una base de datos SQL distribuida de código abierto, YugabyteDB combina las características de las bases de datos relacionales con la escalabilidad y resiliencia de las bases NoSQL.

Ejemplos de Bases de datos distribuidas

Ejemplos de sistemas de tipo BASE incluyen:

- **Oracle Database:** La base de datos multimodal de Oracle soporta el desarrollo de aplicaciones sin esquema convencional mediante el modelo de datos JSON. Esto permite un enfoque de desarrollo híbrido, combinando las características del desarrollo de aplicaciones con bases de datos NoSQL orientadas a documentos y las funciones de una base de datos relacional de clase empresarial.
- **Apache Cassandra:** Cassandra es una base de datos NoSQL distribuida diseñada para manejar grandes volúmenes de datos en servidores económicos. Ofrece alta disponibilidad a un costo mínimo.
- **Couchbase:** Una base de datos NoSQL distribuida, Couchbase presenta almacenamiento orientado a documentos, caché en memoria y particionado automático, además de replicación entre múltiples centros de datos.
- **MongoDB:** MongoDB es una base de datos NoSQL muy popular que utiliza un modelo de datos orientado a documentos. Está diseñada para alto rendimiento y fácil escalabilidad, con esquemas flexibles, particionado automático y conjuntos de réplicas para alta disponibilidad.
- **Redis:** Este almacén de datos en memoria, NoSQL, puede usarse como base de datos, caché y/o intermediario de mensajes. Soporta una amplia variedad de estructuras de datos, incluyendo cadenas, hashes, listas y conjuntos. En su versión empresarial, Redis intenta ofrecer sistemas que cumplen tanto con BASE como con ACID.