1. CONCEPTOS BÁSICOS

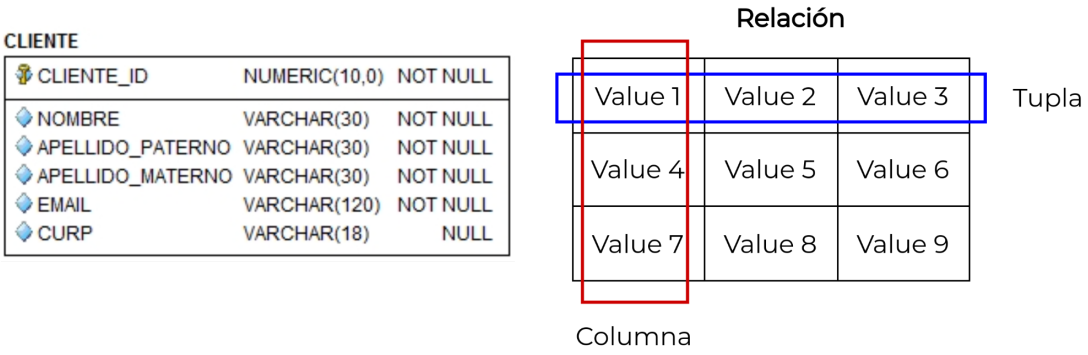
Índice

1.1. SQL, NoSQL, NewSQL

Durante los últimos años el dominio de los RDBMS como solución principal para implementar la capa de persistencia de las aplicaciones ha cambiado considerablemente.

El auge de diversas tecnologías, movimientos, y nuevas oportunidades de negocio como son: Internet de las cosas, Cloud computing, Big Data, han generado volúmenes impresionantes de datos organizados en 3 grupos principales:

- ***Datos estructurados:*** Representados por una estructura o esquema bien definido (estricta y fija) llamado *modelo de datos*. Ejemplos: tablas, hojas de excel.



- ***Datos semi-estructurados***: También son representados por una estructura, pero no tan estricta y uniforme como en el caso de los datos estructurados. Ejemplos: JSON, XML. Ambos pueden o no estar asociados a una estructura. Por ejemplo, pueden existir 2 documentos JSON que contengan datos de alumnos con diferentes atributos.

```
[{"id":10234,
  "nombre": "Gerardo",
  "apPat":"Juarez",
  "email":"ja@mail.com",
  "asignaturas": [1213,1873]
},
{"id":123493,
  "nombre": "Fabián",
  "apPat":"Luna",
  "apMat":"Perez",
  "cursos": ["algebra","calculo"]
}]
```

- ***Datos no estructurados***: No cuentan con una organización o formato predefinido, no cuentan con un modelo de datos. Ejemplos: contenido multimedia (audio, video, imágenes). También pudiera ser un documento de texto plano que no cuenta con alguna organización, solo texto.

Lo anterior genera una serie de requerimientos que deberán ser implementados en diferentes soluciones:

- Escalabilidad horizontal en lugar de escalabilidad vertical.
- Alta disponibilidad
- Replicación y partición
- Flexibilidad de modelos
- Confiabilidad, etc.

Para implementar estos requerimientos han surgido 3 grandes grupos de Bases de datos:

- ***Bases de datos SQL*** conocidas también como RDBMS.
- ***NoSQL***: Conjunto de conceptos que permite el procesamiento rápido y eficiente de grandes conjuntos de datos con un enfoque constante en desempeño, confiabilidad y agilidad.
- ***NewSQL***: Versión o evolución de las bases de datos relacionales que incorporan (mezclan) algunas funcionalidades de los sistemas NoSQL, por ejemplo, escalabilidad, disponibilidad sin dejar de lado las funcionalidades de los RDBMS, por ejemplo, las transacciones y el uso de las propiedades ACID.

Las 3 categorías anteriores pueden implementar los conceptos de distribución de datos, cada una con sus respectivos pros y contras.

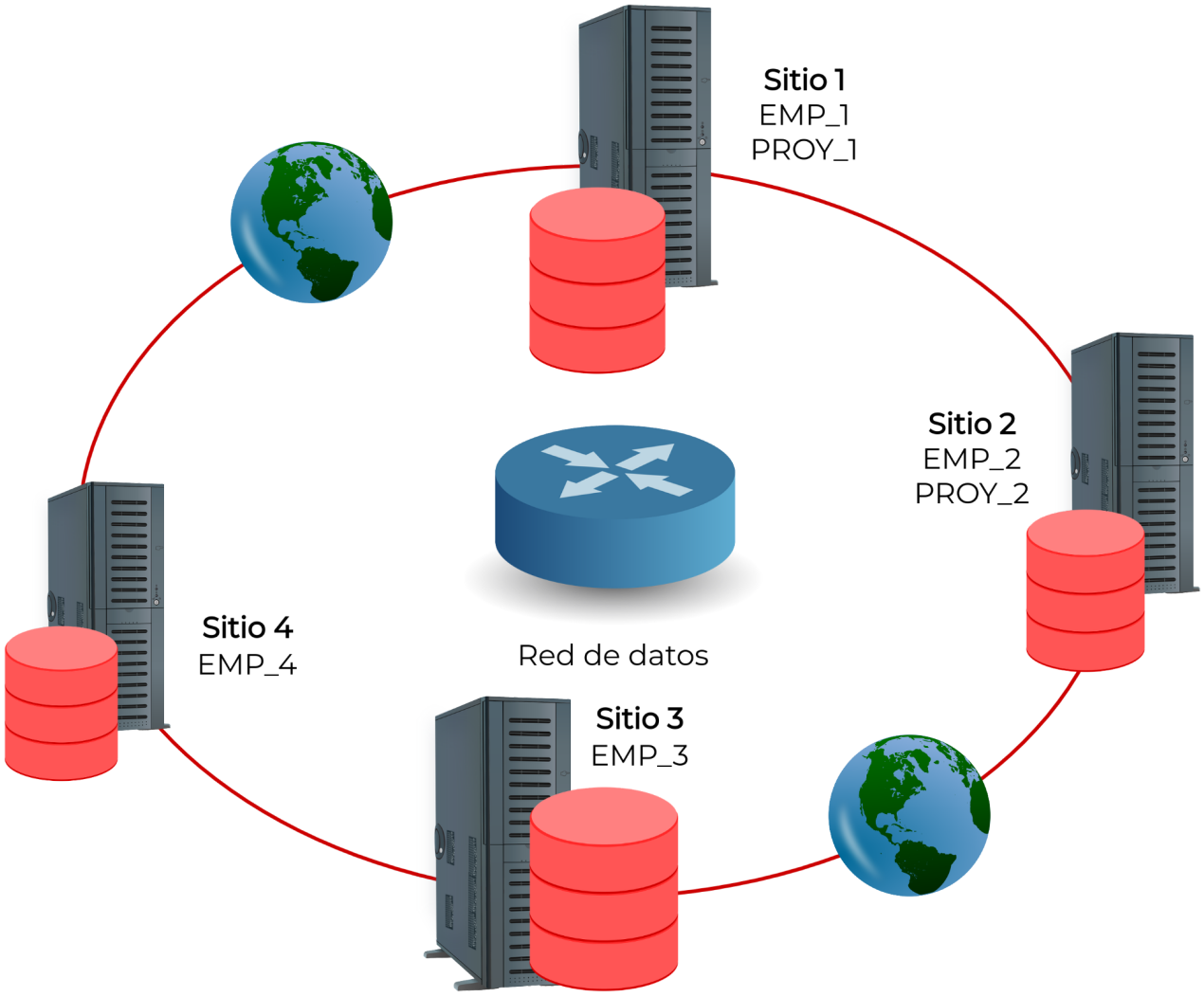
El curso se enfoca principalmente a los sistemas SQL. Las otras 2 categorías se revisarán a nivel general más adelante. Por ahora, el término Base de Datos distribuida hará referencia a sistemas SQL. Las principales técnicas para distribuir una BD relacional son:

- Particionamiento
- Sharding (implementado por el RDBMS)
- Fragmentación (Implementada por la aplicación)

Tarea

Investigar 2 sistemas de bases de datos tipo ***NewSQL***. Indicar principales características y usos prácticos.

1.2. Bases de datos distribuida

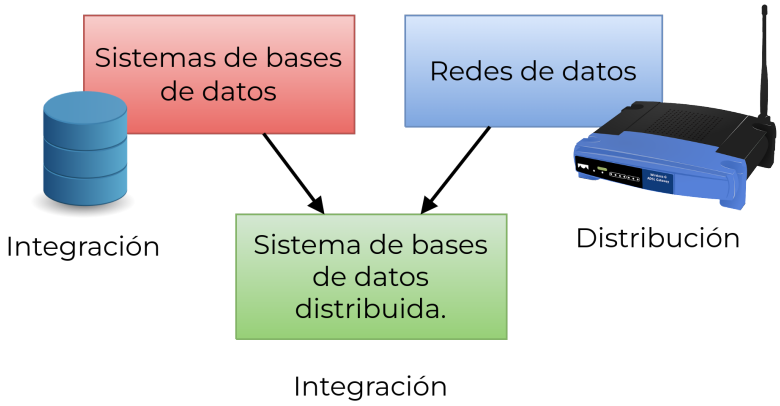


- Colección de múltiples bases de datos lógicamente inter-relacionadas a través de una red de datos.
- El único recurso compartido es una red de datos.
- Lo anterior no implica que la BD tenga que estar distribuida a grandes distancias.
- La base de datos se encuentra distribuida en ubicaciones diferentes llamados sitios.
- Cada una de las máquinas que forman parte de la red tiene autonomía local.
- La base de datos está dividida en un conjunto de ***fragmentos***.
- En un sitio pueden existir 1 o más fragmentos.
- Un fragmento puede ser replicado en uno o más ***sitios***.
- Se realizan continuas transacciones de información entre nodos.

1.2.1. Sistema Administrador de Bases de Datos Distribuidas (DDBMS)

Software encargado de administrar una BDD. Proporciona un mecanismo de acceso en el que la distribución de los datos se vuelve transparente para el usuario final.

Un RDBMS implementa todos los conceptos, algoritmos y protocolos que se requiere un DDBMS. No se requiere adquirir software adicional o instalar algún componente adicional.



En la figura anterior se puede observar las leyendas ***integración*** y ***distribución***.

- Uno de los objetivos de una BD es la integración de los datos
- Por otro lado, uno de los objetivos de una red es la “distribución” .
- Ambos conceptos parecieran ser contrarios, pero al unirlos, se obtiene una BDD que permite la integración de la BD y permite su distribución.

1.2.2. Beneficios de las Bases de Datos Distribuidas

- Administración transparente de datos distribuidos y replicados. El usuario no se percata de la naturaleza distribuida de los datos. Existen diversos tipos de transparencia.
- Acceso seguro a datos a través de transacciones distribuidas.
- Mejora en desempeño comparado con los accesos remotos realizados por una BD centralizada.
    - Los datos deben localizarse lo más cerca posible con respecto a las ubicaciones donde más se emplearán. Se reducen accesos remotos a grandes distancias y se sustituyen por accesos locales o cercanos.
    - La distribución del contenido de una tabla en fragmentos reduce la cantidad de datos que se procesa en cada nodo. Cada sitio maneja solo una porción de la BD, permite reducir el uso de recursos como memoria, procesador y almacenamiento.
    - Permite paralelismo
        - Inter paralelismo: Capacidad de ejecutar N sentencias al mismo tiempo.
        - Intra paralelismo: Una sentencia es dividida en N sub sentencias y cada una de ellas es ejecutada en un sitio distinto accediendo a partes diferentes de la BDD.
- Facilidad de expansión.
    - Disminución de costos al adoptar crecimiento horizontal vs Vertical

    - Disminuye limitantes de crecimiento por cuestión de espacios físicos
- Confiabilidad y alta disponibilidad.
    - Resuelve problemas de confiabilidad eliminando la dependencia de un sitio central (Síndrome del único punto de falla).
    - La replicación permite que la BD esté disponible a pesar de existir fallas en uno o más nodos.

1.2.3. Retos a resolver al distribuir una base de datos

- El diseño de una BDD es más complicado que el centralizado.
    - ¿Cómo fragmentar?
    - Particionamiento vs replicación
- Procesamiento distribuido de consultas es diferente que el centralizado
    - Obtención de planes de ejecución distribuidos y optimizados.
- Administración del directorio de administración distribuido (Diccionario de datos distribuido: DDD).
- Control distribuido de concurrencia.
    - Sincronizar los accesos concurrentes en todos los sitios para garantizar integridad.
    - Si existe replicación, garantizar la integridad de todas las copias.
- Administración distribuida de Deadlocks.
- Confiabilidad.
    - Implementación de algoritmos para garantizar que la BDD siga operando a pesar de fallas en nodos.
    - Implementación de transacciones distribuidas.
    - Implementación de las propiedades ACID para BDD.
- Replicación
- Seguridad
- Falta de estándares.
- Falta de experiencia.

1.3. Transparencia en BDD

- Representa una de las características más importantes de una BDD.
- Permite ocultar por completo la naturaleza y complejidad de distribución de tal forma que el usuario no se percata de ello.
- Existen diversos tipos de transparencia.

1.3.1. Transparencia de distribución

La transparencia de distribución se implementa tomando como elemento principal el llamado Esquema global o Diccionario de Datos Distribuido (DDD), llamado también Catálogo de Datos Distribuido (DDC). El esquema global contiene:

- La definición y descripción de la BD completa tal cual como la visualiza un usuario final, similar al modelo relacional de una BD centralizada.
- La definición de cada uno de los esquemas locales que se encuentran en cada sitio.

Este tipo de transparencia requiere ser implementado por la aplicación. Existen 3 niveles de transparencia de distribución:

- Transparencia de fragmentación
- Transparencia de localización
- Transparencia de mapeo local.

El nivel indica la información que tendría que proporcionar el usuario final para obtener un dato:

Nivel de transparencia de distribución	¿Requiere conocer el nombre del fragmento?	¿Requiere conocer la ubicación del fragmento?
Transparencia de fragmentación	NO	NO
Transparencia de localización	SI	NO
Transparencia de mapeo local	SI	SI

Ejercicio en clase 1

Realizar las actividades del siguiente ejercicio. La respuesta no se incluye en estas notas, se sugiere agregar la respuesta en sus apuntes.

Suponer que una empresa global decide distribuir los datos de sus clientes y sus adeudos. Se requiere almacenar: nombre, apellidos, adeudo total y clave del país en el que labora. Se consideran las siguientes reglas de fragmentación:

- Registros de MX en el sitio S1 con hostname `bdmx.com`
- Registros de JAP en el sitio S2 con hostname `bdjap.com`

A. Generar un modelo relacional que represente el esquema global de esta BD.
B. Generar un modelo relacional que represente el esquema de fragmentación
C. Generar 3 consultas que muestre todos los datos de los empleados empleando cada uno de los niveles de transparencia de distribución.

1.3.2. Transparencia de transacciones

- Garantiza la integridad y el estado consistente de los datos en una BDD particularmente cuando se ejecutan diversas sentencias DML en varios sitios.
- Garantizar lo anterior requiere que el DDBMS implemente diversos protocolos y algoritmos diferentes a los empleados en las BD centralizadas, por ejemplo, protocolo ***two phase commit***.
- Este tipo de transparencia es implementada por el DDBMS.

Ejercicio en clase 2

Realizar las actividades del siguiente ejercicio. La respuesta no se incluye en estas notas, se sugiere agregar la respuesta en sus apuntes.

Generar el código necesario que permita garantizar transparencia de transacciones para la siguiente operación. Considere el mismo esquema de fragmentación del ejercicio en clase anterior.

- Decrementar 10% el adeudo de los clientes en donde dicho adeudo sea más de $1000.
- Considerar que solo se cuenta con transparencia de localización

El código anterior debe cumplir con los siguientes requerimientos:

- Si ocurre un error en alguno de los sitios, el DDBMS garantiza que se hará rollback en todos los sitios donde se hicieron cambios.
- De no ocurrir errores, el DDBMS debe garantizar que se hace `commit` en cada sitio.

Si la transacción se programa de forma correcta, este tipo de transparencia existe y lo implementa el DDBMS.

1.3.3. Transparencia de fallas

- Asegura que la BDD seguirá operando de forma correcta en el caso de presentarse fallas en alguno de los nodos.
- Las peticiones serán redirigidas a otros nodos para ser procesadas.
- Implementada por el DDBMS

1.3.4. Transparencia de desempeño

- Permite que el desempeño sea por lo menos similar al que ofrece una BD centralizada.
- Permite la selección de la mejor estrategia que minimice los costos de ejecución de una consulta distribuida.
- El DDBMS implementa diversas estrategias para optimizar la ejecución de consultas distribuidas.
    - Es muy importante contar con un buen diseño y escritura de sentencias SQL de la forma más óptima posible.

1.3.5. Transparencia de heterogeneidad

- BDD heterogénea: Aquella formada por diferentes manejadores de bases de datos. P.e. Oracle, DB2. SQL server.
- Permite la integración de diferentes manejadores en una misma BDD. Inclusive BDD con otros tipos de modelos de datos. Jerárquico, etc.
- Implementada por el DDBMS.

1.3.6. Transparencia de replicación

- La replicación se emplea para cubrir requerimientos como desempeño, confiabilidad y disponibilidad.
- En este tipo de transparencia el usuario no requiere saber si existe una copia en la máquina local, y de la lógica que se tendría que ejecutar en caso de que no existiera.
- Implementada por el DDBMS.

1.4. Variables empleadas en arquitecturas de Bases de Datos Distribuidas

Existen 3 variables que pueden influir en el diseño de un DDBMS.

1.4.1. Autonomía

Se refiere a la distribución del control (no de los datos) e indica en grado con el cual un DDBMS puede operar de manera independiente con respecto a otros sitios: libertad para ejecutar transacciones, libertad para decidir qué información compartir, etc.

1.4.2. Distribución

Se refiere a la distribución física de los datos sobre diferentes sitios:

- Sin distribución
- Distribución cliente/servidor:
- Distribución punto a punto: (llamada también distribución completa):
    - No existe distinción entre la máquina cliente y servidor
    - Cada máquina tiene funcionalidad completa de un DBMS.

1.4.3. Heterogeneidad

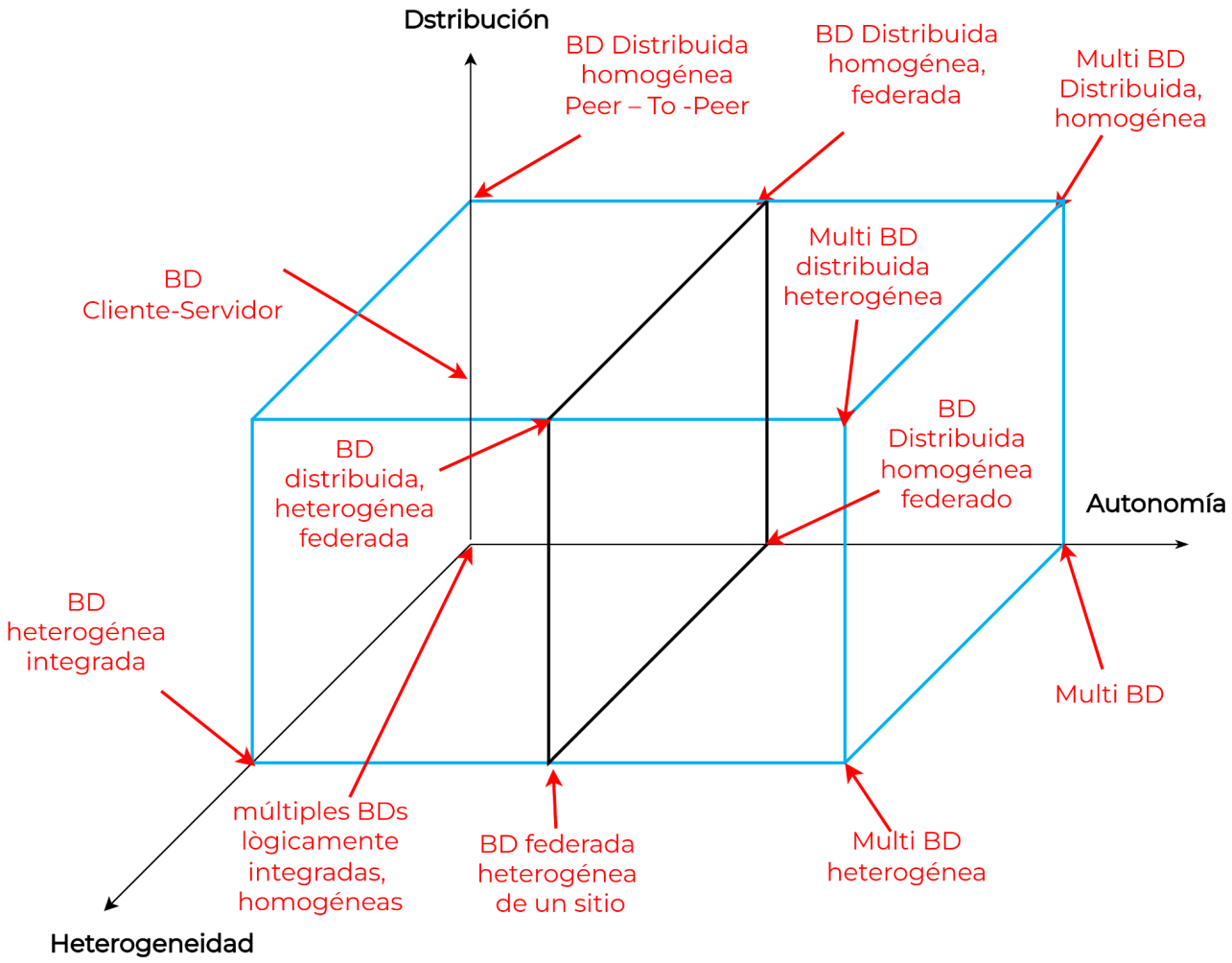
Se refiere a la diversidad de componentes que pueden emplearse en cada sitio:

- Hardware
- Sistema operativo
- Protocolos de comunicación
- Componentes de bases de datos.

Un DDBMS es homogéneo cuando se emplea un mismo DBMS en todos los sitios y heterogéneo cuando se emplean diversos DBMS, e incluso diferentes modelos de datos, no solo el relacional.

La combinación de estas 3 dimensiones permite identificar varias arquitecturas de bases de datos.

- Sistemas cliente servidor.
- Sistemas Multibases de datos.
- Sistemas federados
- Sistemas Punto a punto (Peer to Peer).



Tarea Investigar:

- Multibase de datos
- Sistemas de BD federados
- Sistemas de BD Cliente- Servidor