Bases de datos distribuidas

ALFONSO FERNANDEZ Y ADRIÁN ÁLVAREZ

Contenido

1. Introducción 2
2. Transparencia 2
3. Arquitecturas 3
3.1 Arquitecturas comunes 3
Cliente-Servidor 3
P2P 4
4. Estrategias de diseño 5
4.1 Fragmentación 6
4.2 Asignación 6
4.3 Replicación 7
5. Etapas del procesamiento 7
6. Funciones del administrador 7

1. Introducción

Las BDD son el resultado de combinar las bases de datos con las redes de computadores. Esto implica que podemos conectar distintos ordenadores independientes y hacer que trabajen juntos.

Esto nos da muchas posibilidades de diseño (Que veremos más adelante) para adaptarla a nuestras necesidades:

- Podemos desarrollar la BD a trozos, añadiendo, modificando o eliminando nuevos nodos según necesitemos.
- Si la información esta replicada estaremos preparados en caso de accidente.
- Cada nodo podrá contener solo la información que regularmente, proporcionándole un acceso veloz a la vez que es accesible por otros nodos.

Pero también nos añade problemas:

- Necesitamos un software potente y fiable en todas las maquinas. Además del coste del personal necesario y mantenimiento.
- Es más difícil mantener la integridad de la información cuando existe redundancia.
- Evidentemente es más compleja de diseñar, y el uso de telecomunicaciones nos crea una gran dependencia en el funcionamiento.

Ventajas	Desventajas
Modular	**Coste**
Fiable	**Integridad**
Velocidad	**Complejidad**

2. Transparencia

La transparencia se refiere a la capacidad que tiene la base de datos para ocultar los detalles de implementación al usuario. Normalmente estos detalles no importan al cliente y, cuando son percibidos, normalmente es porque causan alguna molestia.

Podemos distinguir 3 tipos elementales de transparencia:

- **Transparencia de distribución:** Como veremos mas adelante, hay muchas formas de distribuir una BD. Ubicación física de los nodos, ubicación, fragmentación y replicado de la información… Todo esto debe ser desconocido e imperceptible para el usuario.
- **Transparencia de transacción:** Para ocultar el reparto y fragmentación de la información, las transacciones deberán mantener en todo momento la integridad de la BD en todos los nodos involucrados en ellas.

- **Transparencia de gestor (Solo aplica cuando existe heterogeneidad):** El usuario no debe percibir las inconveniencias de que algún nodo use otro sistema gestor de bases de datos.
- **Transparencia de rendimiento:** El rendimiento de una BDD debe ser los más próximo posible al de una BD centralizada local.

3. Arquitecturas

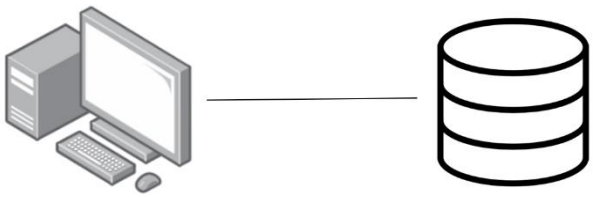
Para determinar la arquitectura debemos tener en cuenta 3 parámetros:

- **Distribución:** Como vamos a repartir los datos entre los nodos.
- **Autonomía:** Como de independientemente trabajaran los nodos.
- **Heterogeneidad:** Como de similar es el software (SGBD) de los nodos.

Oracle está diseñado para funcionar bien en una base de datos homogénea (Todos los nodos usan el mismo SGBD), e incluso admite que los nodos usen versiones distintas (Siempre que todos los nodos sean conscientes de estas diferencias).
Oracle también incluye herramientas específicas (Heterogeneous Services) que permiten la integración de otros SGBD.

3.1 Arquitecturas comunes

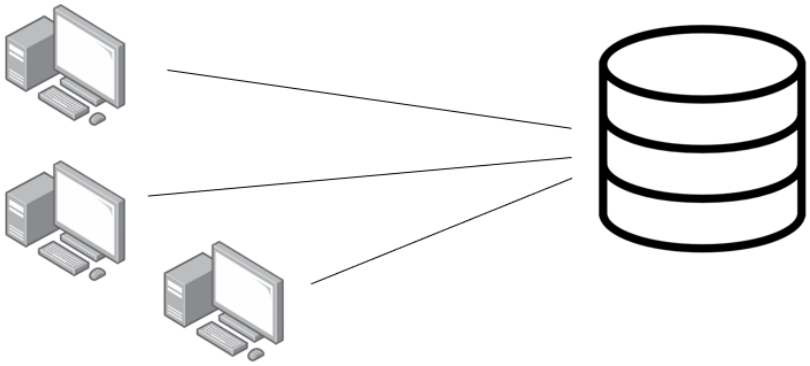
Cliente-Servidor



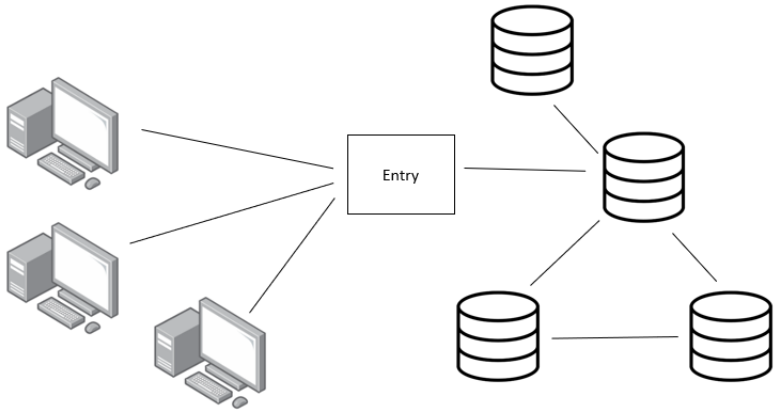
La arquitectura Cliente-Servidor consiste en dividir la funcionalidad.

El cliente es el encargado de la interfaz de usuario y de llevar un cierto control sobre la consistencia de los datos introducidos, mientras que el servidor es el encargado del manejo de los datos, consultas, transacciones...

Este esquema tiene algunas variaciones en función del numero de nodos que forman la base de datos. El mas simple sería un único sistema para un numero n de clientes, lo que llamamos base de datos remota.

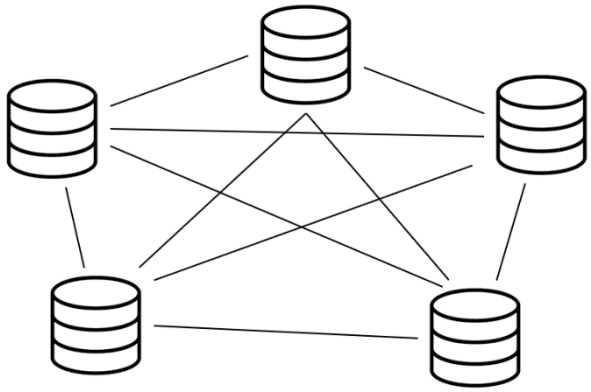


En su defecto podemos tener una base de datos formada por varios nodos a los que acceden varios clientes a la vez a través de la red.



P2P

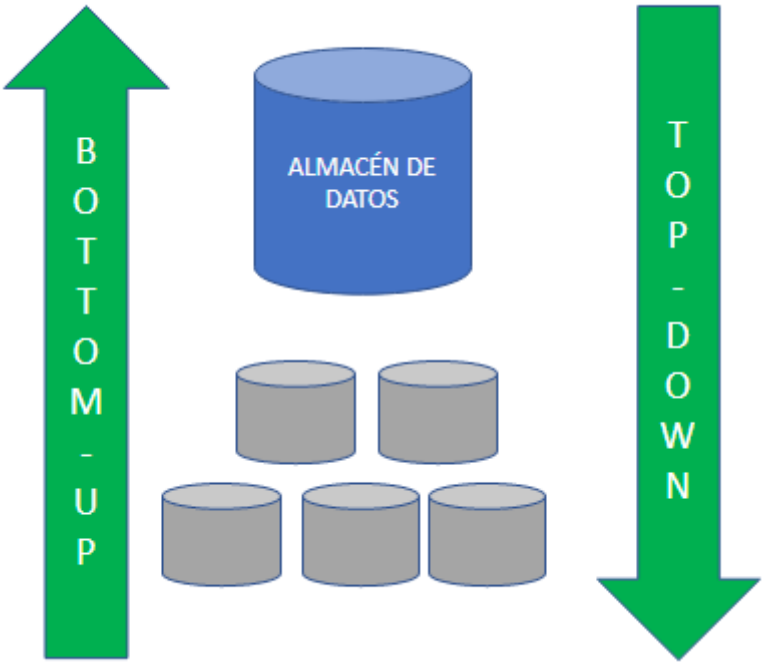
El Peer-to-Peer consiste en que todos los nodos están interconectados, actúan a la vez como cliente y como servidor para los demás y son los encargados de administrar sus propios datos y transacciones con otros nodos.



4. Estrategias de diseño

Para el diseño de las Base de Datos Distribuidas BDD, primero hay que definir como empiezan esa creación, en que modelo vamos a enfocarnos:

- **TOP-DOWN:** Este modelo está inspirado para aplicaciones nuevas y sistemas homogéneos principalmente, ya que hay que empezar por el diseño primero de todo. Se comienza diseñando el esquema global, luego se concibe la fragmentación de la BD y la localización de los fragmentos en los sitios. Para completar este diseño, se acaba ejecutando el diseño físico de los datos.
- **BOTTOM-UP:** Modelo en el que contamos con BD existentes donde queremos integrar varias de ellas para crear un esquema global. Cuando ya tenemos una idea definida, tenemos que traducir cada esquema local, y finalmente se hace la integración del esquema local en un esquema global.



Una de las etapas más importantes del diseño, donde diferenciamos una base de datos centralizada a una distribuida, es como la distribuimos, nos enfocamos en el Diseño de la Distribución, que consta de 2 actividades:

Fragmentación: Decir “COMO” dividimos la BD y en “QUÉ” partes.

Asignación: Decir “DONDE” ubicamos cada parte, así como si tenemos replicación de datos.

Antes de cada explicación hay que tener en cuenta varios aspectos a considerar en el diseño de una BDD:

- Fragmentación: Definimos este apartado como el acto en la que una relación puede ser dividida en un número de sub-relaciones.
- Asignación: cada fragmento debe ser almacenado en un sitio en base a una distribución óptima.
- Replicación: El SGDBB puede mantener una copia de un fragmento en diferentes ubicaciones.

4.1 Fragmentación

Hablamos de fragmentación a la técnica de Optimizar las tablas distribuidas entre los distintos servidores de la base de datos.

- Fragmentación Horizontal: En ella particionamos/dividimos la relación en subconjuntos de tuplas, con datos comunes. Para acceder a ellos solo necesitamos la operación SELECT sobre la relación global. Cada fragmento se sitúa en un nodo. Lo bueno de este tipo de fragmentaciones, es para consultas donde devuelven subconjuntos de filas, es decir, que se quiere mucha información, como nombre/email/dirección etc.
- Fragmentación Vertical: A diferencia de la fragmentación horizontal, que divide la tabla en función de las filas, Estos subconjuntos, para que tengan concordancia, están unidos por un valor común, como un id. A diferencia de la fragmentación horizontal, esta es más utilizada sobre todo para cuando quieres pocos valores: ID y nombre por ej.
- Fragmentación Híbrida: es la mezcla de las dos anteriores, aprovechando lo mejor de cada una en este sentido, la fragmentación es como la partición, que divide las tablas grandes en otras más pequeñas.

Ventajas	Desventajas
Procesamiento Concurrente	**Coste**
Paralelización de Consultas	**Complejidad semántica**

4.2 Asignación

Proceso en el cual decidimos donde se almacenarán los fragmentos que se generan tras la fragmentación y si se harán réplicas de los mismos.

Asumamos que hay un conjunto de fragmentos F = {F1, F2, …, Fn} y una red que consiste de los sitios S = {S1, S2, …, Sm} en los cuales un conjunto C = {q1, c2, …, cq} de consultas se van a ejecutar.

El problema de asignación consiste en determinar la distribución "óptima" de F en S.

La optimalidad puede ser definida de acuerdo a dos medidas:

- **Costo mínimo**. Consiste, entre los valores del costo de comunicación de datos, del costo de almacenamiento, y del costo procesamiento (lecturas y actualizaciones a cada

fragmento), encontrar un esquema de asignación o ubicación que minimiza la función de costo combinada.

- **Rendimiento.** La estrategia de asignación se diseña para mantener una métrica de rendimiento. Las dos métricas más utilizadas son el tiempo de respuesta y el "throughput" (número de trabajos procesados por unidad de tiempo).

4.3 Replicación

Hablamos de replicación como el poder de guardar fragmentos en diferentes sitios. El problema de ello es en qué sitios hacemos la copia y los problemas que conlleva actualizarlos.

Hacer replicas tiene sentido por:

- Confiabilidad: Mayor seguridad ante perdida de datos
- Disponibilidad: Mayor tolerancia a fallos ante caídas de los computadores
- Aumento del paralelismo: Mayor eficiencia en las consultas de lectura al posibilitarse su descomposición en subconsultas y la paralelización de éstas.

Según el grado de replicación, distinguimos entre :

- BD fragmentada: Fragmentos disjuntos, cada uno en un nodo (no hay replicas)
- BD totalmente replicada: Se encuentra una copia de toda la BD en cada nodo
- BD parcialmente replicada: Mezcla las anteriores. Algunos fragmentos están replicados.

5. Etapas del procesamiento

1. DESCOMPOSICIÓN DE CONSULTAS GLOBALES

Proceso donde lo que queremos es simplificar las consultas por eliminación, hasta alcanzar el rendimiento esperado.

2. LOCALIZACIÓN DE LOS DATOS

Se basa en encontrar los datos, traducirlos los datos en la consulta que queremos, y mejorarla hasta mostrarla.

3. OPTIMIZACIÓN DE CONSULTAS GLOBALES

Al momento de escribir las consultas, se reordenan buscando la mayor eficiencia para aprovechar los recursos de computación de los que disponemos.

4. OPTIMIZACIÓN DE CONSULTAS LOCALES

Aquí el tiempo de respuesta es más dinámica, como se emplea una búsqueda total depende de los algoritmos utilizados.

6. Funciones del administrador

- Creación y eliminación de las bases de datos.
- Configuración de las opciones de seguridad y auditoría.
- Integración de los esquemas para proporcionar una visión global.
- Realización de procedimientos de backup y recuperación adecuados.
- Determinar la fragmentación de las relaciones, y la replicación de datos.

- Aplicación de restricciones de integridad y seguridad, y de diversas políticas.
- Definición de estándares.

Luego tenemos unas funciones un poco más exclusivas como:

- Gestión de nombres globales
- Creación y gestión de links entre BD (permite el acceso a objetos de una BD remota).
- Creación y gestión de link compartidos entre BD (permite limitar el número de conexiones entre el servidor local y el servidor remoto).
- Cambiar el dominio en una BD.
- Desarrollo de funciones y procedimientos PL/SQL con soporte para BDD