BASE DE DATOS DISTRIBUIDAS

Ari Merma Carlos,
Gonzales Flores Alejandra,
Manrique Salas,Brenda ,
Postigo Toledo Rosmery,
Tapia Cossio Anthony

"23 de Noviembre de 2010"

Resumen

Las bases de datos distribuidas se desarrollaron para que la información sea compartida por distintos usuarios y los datos sean utilizados desde distintos sitios, siendo la característica más importante la transparencia de los procesos. Este sistema involucra diferentes objetivos los cuales generan diversos problemas los que no permiten un sistema de bases de datos distribuida ideal. Además se trata el concepto de middleware para la comunicación entre los tipos de bases de datos.

Palabras Claves

Sistemas Distribuidos, Bases de Datos Distribuidas, DBMS.

Abstract

The distributed databases are developed to ensure that information is shared by several users and the data are used from different places, being transparency of processes the most important characteristic. Although this system involves different objectives that drive the use of such database, weâll see that sometimes they generate different problems from getting an ideal system for distributed databases. In addition we talk about the concept of middleware for communication between the types of databases.

Key Words

Distributed Systems, Distributed Database, DBMS.

1 Introducción

Desde los años 70 se almacenaba la información de manera centralizada, pero con el paso del tiempo las necesidades aumentaron y esto produjo ciertos inconvenientes que no era posible solucionarlos o volverlos eficientes de la forma centralizada. Estos problemas impulsaron la creación de almacenamiento distribuido, los cuales hoy en día proveen características indispensables en el manejo de información; es decir, la combinación de las redes de comunicación y las bases de datos.

Según [1] Una BDD es en realidad un tipo de base de datos virtual cuyas partes componentes están almacenadas en varias de datos reales que se encuentran en varias sitios distintos, en otras palabras es la unión lógica de esas bases de datos reales.

2 Definición

Según [3] "El soporte completo para las bases de datos distribuidas implica que una sola aplicación debe ser capaz de operar de manera transparente sobre los datos que están dispersos en una variedad de bases de datos diferentes, administradas por una variedad de distintos DBMSs, ejecutadas en diversas máquinas diferentes, conectadas a una variedad de redes de comunicación distintas; donde el término de **manera transparente** significa que la aplicación opera desde un punto de vista lógico como si todos los datos fueran manejados por un solo DBMS y ejecutados en una sola máquina."

Se considera una base de datos distribuida a aquellas que cumplen lo siguiente:

- Los datos deben estar físicamente en más de un sitio.
- Los sitios deben estar interconectadas.
- Los datos han de estar lógicamente integrados.
- En una única operación se puede acceder (recuperar o actualizar) datos que se encuentran en más de un sitio.
- Todas las acciones que necesiten realizarse sobre más de un sitio serán transparentes al usuario.

Cada sitio local contiene:

- Sus propias bases de datos reales.
- Sus propios usuarios locales.
- Su propio DBMS local.
- Su propia administración de datos locales.

Un BDD es bastante conveniente pero a la vez más complicado, es por ello que diversos fabricantes y desarrolladores dedican gran esfuerzo en la implementación de este servicio cada vez mejor.

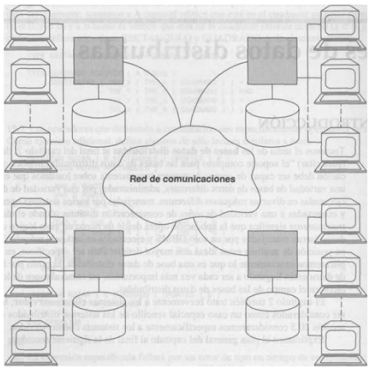

Figure 1: Ejemplo BDD típica

Es importante distinguir entre sistemas de bases de datos distribuidas **verdaderos y generalizados**, de los sistemas que simplemente proporcionan algún tipo de acceso a datos remotos (que son en realidad sistemas cliente-servidor).

En un **sistema de acceso a datos remotos**, el usuario puede operar simultáneamente sobre datos que están en un sitio remoto o incluso en varios sitios remotos de los cuales se está definitivamente consciente, de alguna manera, de que los datos son remotos; por el contrario, en un **verdadero sistema de base de datos distribuida** es totalmente transparente al usuario.

Existen dos tipos de transacciones. Tenemos las **transacciones locales**, que se dan cuando se accede a un único sitio remoto, y las **transacciones globales** las que recopilan datos de diferentes sitios.

Las bases de Datos Distribuidas utilizan la técnica de **réplica de datos**, que está basada en la copia de datos de disco a disco, donde existe un gestor de replicas que las administra para un mejor rendimiento, disponibilidad y tolerancia a fallos, las que trabajan de manera transparente y coherente. Esto involucra la **propagación de la actualización** que no es nada más que actualizar todas las copias de un objeto replicado.

También se da la **Fragmentación de Datos** que divide los datos en varios sitios de los cuales son obtenidos mediante transacciones globales. Existen básicamente dos tipos de fragmentación, **horizontal y vertical**, que corresponden a las operaciones relaciónales de **restricción y proyección**, respectivamente.

Se distinguen dos tipos de sistemas, un **sistema homogéneo** donde las DBMS son las mismas y resulta más flexible la administración; además de un **sistema heterogéneo** donde se trabaja con diferentes DBMS (p.e MySQL, PostgreSQL, etc.) lo que significa relacionar todas estas para su distribución.

3 Ventajas[2]

- **Flexibilidad y fiabilidad**, acceso desde distintos lugares y por distintas personas a la vez sin inconvenientes.
- Mejora del **rendimiento**, BD más pequeñas, operaciones de menor volumen.
- **Compartimiento de datos**. Los usuarios de un sitio son capaces de acceder a los datos de uno o más sitios.
- **Autonomía**. Cada sitio tiene cierto grado de control sobre sus datos.
- **Disponibilidad**. Si en un sistema distribuido falla un sitio, los sitios restantes pueden seguir funcionando. Si se duplican los datos en varios sitios, la transacción que necesite un determinado dato puede encontrarlo en cualquiera de los diferentes sitios.

4 Desventajas[2]

- **Coste de desarrollo del software**. La complejidad añadida que es necesaria para mantener la coordinación entre nodos hace que el desarrollo de software sea más costoso.
- **Mayor probabilidad de errores**. Como los nodos que constituyen el sistema funcionan en paralelo, es más difícil asegurar el funcionamiento correcto de los algoritmos, así como de los procedimientos de recuperación de fallos del sistema.
- **Mayor sobrecarga de procesamiento**. El intercambio de mensajes y ejecución de algoritmos para el mantenimiento de la coordinación entre sitios supone una sobrecarga que no se da en los sistemas centralizados.

5 Un principio fundamental

El principio fundamental de la base de datos distribuida segun [3]
"Ante el usuario, un sistema distribuido debe lucir exactamente igual que un sistema que no es distribuido"
Un sistema distribuido debe ser capaz de comportarse exactamente como si el sistema no fuera distribuido. Todos los problemas de los sistemas distribuidos son, o deberían ser, problemas internos o en el nivel de implementación, y no externos o en el nivel del usuario.
Este principio nos conduce a doce reglas complementarias

5.1 Autonoma local

Los sitios en un sistema distribuido deben ser autónomos.
Idealmente la autonomía local significa que las operaciones de un sitio X no debe depender de algún otro sitio Y. Los datos locales son poseídos y

administrados localmente. Por lo tanto, la seguridad, integridad y representación de almacenamiento de los datos locales permanecen bajo el control y jurisdicción del sitio local.

Sin embargo este concepto no es totalmente alcanzable, existen varias situaciones en las que un sitio X dado debe transferir un determinado grado de control a algún otro sitio Y. Por lo tanto, el objetivo de autonomía queda establecido con mayor precisión como: los sitios deben ser autónomos en el mayor grado posible

Aquí resumimos algunas situaciones:

- Normalmente, no es posible acceder directamente a fragmentos individuales de una VR fragmentada, ni siquiera desde el sitio en el que están almacenados.
- Normalmente, no es posible acceder directamente a copias individuales de una VR replicada (o fragmentada), ni siquiera desde el sitio en el que están almacenadas.
- Sea P la copia primaria de alguna VR R replicada (o fragmentada), y sea que P está almacenada en el sitio X. Entonces, todo sitio que acceda a R es dependiente del sitio X, aunque otra copia de R está de hecho almacenada en el sitio en cuestión.
- No es posible acceder a una VR que participa en una restricción de integridad de varios sitios, para efectos de actualización, dentro del contexto local del sitio en el que está almacenada, sino sólo dentro del contexto de la base de datos distribuida en el que está definida la restricción.
- Un sitio que está actuando como participante en un proceso de confirmación de dos fases, debe obrar de acuerdo a las decisiones (es decir confirmar o deshacer) del sitio coordinador correspondiente.

5.2 No dependencia de un sitio central

La autonomía local implica que todos los sitios deben ser tratados como iguales. Por lo tanto, no debe haber particularmente ninguna dependencia de un sitio "maestro" central para algún servicio central.
La dependencia de un sitio central sería indeseable por las siguientes dos razones

- Primero, el sitio central puede ser un cuello de botella.
- Segundo y más importante, el sistema sería vulnerable; es decir, si el sitio central falla, también fallará todo el sistema

5.3 Operación continúa

En general, una ventaja de los sistemas distribuidos es que deben proporcionar mayor confiabilidad y mayor disponibilidad.

– La confiabilidad (es decir, la probabilidad de que el sistema está listo y funcionando en cualquier momento dado) Los sistemas distribuidos pueden continuar operando (en un nivel reducido) cuando hay alguna falla en algún componente independiente
– La disponibilidad (es decir, la probabilidad de que el sistema está listo y funcionando continuamente a lo largo de un período especificado) también mejora, en parte por la misma razón y en parte debido a la posibilidad de replicación de datos

5.4 Independencia de ubicación

Los usuarios no tienen que saber dónde están almacenados físicamente los datos, sino que deben ser capaces de comportarse al menos desde un punto de vista lógico como si todos los datos estuvieran almacenados en su propio sitio local.
La independencia de ubicación es necesaria debido a que simplifica los programas de usuario y las actividades terminales

5.5 Independencia de fragmentación

Un sistema soporta la fragmentación de datos cuando una VR dada puede ser dividida en partes o fragmentos, para efectos de almacenamiento físico. La fragmentación es necesaria por razones de rendimiento: los datos pueden estar almacenados en la ubicación donde son usados más frecuentemente para que la mayoría de las operaciones sean locales y se reduzca el tráfico en la red.

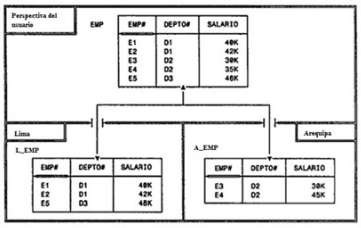

Figure 2: Fragmentación horizontal 1

Por ejemplo, considere una VR de empleados EMP con los valores de ejemplo que muestra la parte superior de la figura 2 En un sistema que soporta la fragmentación, podríamos definir dos fragmentos de la siguiente forma:
FRAGMENT EMP AS
L_ EMP AT SITE 'Lima' WHERE DEPTO = 'D1' OR DEPTO = 'D3';
A_ EMP AT SITE 'Arequipa' WHERE DEPTO = 'D2';

La reconstrucción de la VR original a partir de los fragmentos es lograda por medio de las operaciones de junta y de unión adecuadas (junta para los fragmentos verticales y unión para los horizontales).

La facilidad de fragmentación y la facilidad de reconstrucción son dos de las razones principales por las que los sistemas distribuidos son relacionales; el modelo relacional proporciona exactamente las operaciones necesarias para estas tareas
La independencia de fragmentación implica que a los usuarios se les presentará una vista de los datos en la cual los fragmentos estarán recombinados lógicamente por medio de juntas y de uniones adecuadas. Es responsabilidad del optimizador del sistema determinar cuáles fragmentos necesitan ser accedidos físicamente para satisfacer cualquier solicitud de usuario dada. Por ejemplo, dada la fragmentación que muestra la figura si el usuario hace la solicitud
EMP WHERE SALARIO > 40K AND DEPTO = 'D1'
El optimizador sabrá, por medio de las definiciones de fragmentos (guardadas en el catálogo, por supuesto), que el resultado completo puede ser obtenido desde el sitio de Lima; no hay necesidad de acceder al sitio de Arequipa.

La VR EMP, tal como es percibida por el usuario, puede ser considerada (aproximadamente) como una vista de los fragmentos subyacentes L_ EMP y A_ EMP:
VAR EMP VIEW
L_ EMP UNION A_ EMP;

El optimizador transforma entonces la solicitud original del usuario en lo siguiente:
(L_ EMP UNION A_ EMP) WHERE SALARIO > 40K AND DEPTO 'D1?

Esta expresión puede ser transformada todavía más en:
(L_ EMP WHERE SALARIO > 40K AND DEPTO = 'D1')
UNION
(A _ EMP WHERE SALARIO > 40K AND DEPT0 = 'D1')

A partir de la definición del fragmento A_ EMP en el catálogo, el optimizador sabe que el segundo de estos dos operandos de UNION da como resultado una relación vacía

La expresión general puede entonces ser simplificada a sólo
L_ EMP WHERE SALARIO > 40K AND DEPTO = 'D1'
Ahora el optimizador sabe que necesita acceder solamente al sitio de Lima

5.6 Independencia de replicación

Un sistema soporta la replicación de datos cuando una VR almacenada puede ser representada por muchas copias distintas, o réplicas, guardadas en muchos sitios distintos.
Las réplicas son necesarias por dos razones (como mínimo).

- Rendimiento (las aplicaciones pueden operar sobre copias locales en lugar de tener que comunicarse con sitios remotos).

– Fiabilidad(al haber múltiples copias de los datos disponibles en el sistema, se dispone de un mecanismo excelente de recuperación cuando existan fallos en sitios).

```
REPLICATE N_EMP AS
LN_EMP AT SITE
'Arequipa';

REPLICATE L_EMP AS
NL_EMP AT SITE 'Lima';
```

Figure 3: Ejemplo

– Desventaja: al actualizar un objeto replicado es necesario actualizar todas las copias de ese objeto (propagación de la actualización).
– La replicación debe ser "transparente ante el usuario".
– Un sistema que soporta la replicación de datos también debe soportar la independencia de replicación es necesaria debido a que simplifica los programas de usuario y las actividades terminales; en particular, permite que las réplicas sean creadas y destruidas en cualquier momento en respuesta a los distintos requerimientos, sin invalidar ninguno de esos programas de usuario o actividades.
– La independencia de replicación implica que es responsabilidad del optimizador del sistema determinar cuáles réplicas necesitan ser accedidas físicamente para satisfacer cualquier solicitud de usuario dada.

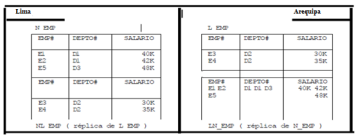

Figura 20.3 Un ejemplo de replicación.

Figure 4: Ejemplo

5.7 Procesamiento de consultas distribuidas

La optimización es importante en un sistema distribuido.
El punto básico es que en una consulta que involucra a varios sitios, habrá muchas formas posibles de mover los datos en el sistema para satisfacer la solicitud, y es crucialmente importante que se encuentre una estrategia eficiente.

5.8 Administración de transacciones distribuidas

Hay dos aspectos principales en la administración de transacciones: el control de la recuperación y el control de la concurrencia.

Una transacción es manejada por varios agentes, donde la ejecución de código puede ser ejecutada en muchos sitios y un agente es el proceso realizado en nombre de una transacción en un sitio dado.
El sistema necesita saber cuándo dos agentes son parte de la misma transacción (por ejemplo, para no caer en bloqueo).
Recuperación: para asegurarse de que una transacción dada sea atómica en el ambiente distribuido, el sistema debe por lo tanto asegurarse de que el conjunto de agentes para esa transacción sea confirmado o deshecho al unísono.
Control de concurrencia: el control de concurrencia está basado generalmente en el bloqueo.

5.9 Independencia de hardware

Soporte para un gran número de máquinas diferentes. Poder integrar todos los datos de todos estos sistemas y presentar al usuario una "imagen del sistema único".

5.10 Independencia de sistema operativo

Es necesario tener la posibilidad de ejecutar el mismo DBMS en diferentes plataformas de sistema operativo.

5.11 Independencia de red

Si el sistema va a tener la posibilidad de soportar muchos sitios distintos es obviamente necesario tener la posibilidad de soportar también una variedad de redes de comunicación distintas.

5.12 Independencia de DBMS

Lo que se necesita es que todos los ejemplares de DBMS en sitios diferentes soporten la misma interfaz.

- Aunque no tienen que ser necesariamente copias del mismo software DBMS.
- En otras palabras, sería posible que el sistema distribuido fuera heterogéneo, al menos en cierto grado.
- Sería muy bueno si diferentes DBMS pudieran participar de alguna forma en un sistema distribuido.

6 Problemas

6.1 Procesamiento de consultas

6.2 Administración de catálogo

6.3 Propagación de actualización

El problema en la replicación de datos es que una actualización a cualquier objeto debe ser propagada a todas las copias almacenadas de ese objeto.

Una dificultad es que si un sitio que mantiene una copia del objeto no está disponible en el momento de la actualización, debido a una falla del sitio o de la red, esta transacción fallaría. Esto debilita a una de las ventajas que mencionamos anteriormente.

Una solución a este problema es el esquema de **copia primaria**, que funciona de la siguiente forma:

- A una copia de cada objeto replicado se le designa como copia primaria. Todas las demás son copias secundarias.
- Las copias primarias de diferentes objetos están en diferentes sitios.
- Cuando se actualiza la copia primaria, el sitio que mantiene esa copia es responsable de la propagación de la actualización hacia las copias secundarias.

6.4 Control de recuperación

El control de la recuperación en sistemas distribuidos está basado típicamente en el protocolo de **confirmación de dos fases**. La confirmación de dos fases nos permite, no solo reconstruir lo que sucedió justo antes del fallo, sino que protege a los nodos que no han fallado para que no lean los datos que representan un estado inconsistente[4].

La confirmación de dos fases se da cuando una transacción interacciona con varios gestores autónomos con el propósito de asegurar que todos los gestionadores de recursos relativos a esta transacción la aceptaran toda o bien la rechazaran, garantizando que la transacción sea todo o nada[4].

6.5 Control de la concurrencia

El control de concurrencia requiere mucho estudio, pero a pesar de esto no hay ningún algoritmo aceptado para solucionar el problema[**?**]. Esto se debe a varios factores de los cuales se consideran a los siguientes tres los más determinantes:

1. La data puede estar duplicada. por tanto, el manejador es responsable de localizar y actualizar la data duplicada.

2. Si un sitio falla o la comunicación con un sitio falla mientras se realiza una actualización, el manejador debe asegurarse de que los efectos se reflejen una vez el sitio se recupere del fallo.
3. La sincronización de transacciones en sitios múltiples es difícil ya que estos no pueden obtener información inmediata de las acciones realizadas en otros sitios concurrentemente.

El ejemplo más común de un bloqueo mutuo es cuando un recurso A está siendo utilizado por una transacción A que a su vez solicita un recurso B que está siendo utilizado por una transacción B que solicita el recurso A. Entre los ejemplos específicos para las bases de datos distribuidas podemos destacar:

- Actualización perdida: cuando dos transacciones concurrentes borran sus actualizaciones.
- La extracción inconsistente: acceder a información modificada parcialmente por una transacción.

Para el control de bloqueos mutuos no se ha desarrollado ninguna solución viable y la forma más simple y que la mayoría de productos utilizan es la implementación de un tiempo máximo de espera en las peticiones de bloqueos.

7 Independencia de DBMS

Regresando uno de los doce objetivos para los sistemas de bases de datos distribuidos en general. La suposición de homogeneidad estricta es discutible; todo lo que en realidad necesitamos es que los DBMSs que están en sitios diferentes soporten la misma interfaz. Por ejemplo, MySQL y Oracle soportan el estándar oficial de SQL debería ser posible hacer que se comportaran en sociedad dentro de un sistema distribuido heterogéneo

Pasarela

Para que 2 o más DBMS de comuniquen se debe proporcionar un programa especial, llamado pasarela, su función es hacer que los DBMSs se parezcan". Como lo indica la Figura 1.

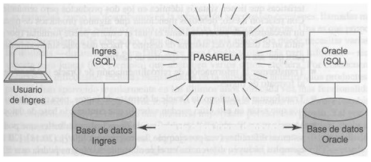

Figure 5: Una pasarela hipotética proporcionada por Ingres para Oracle

Implementar los protocolos para intercambiar información entre diferentes

DBMSs implica transformar el formato de mensaje en el cual son enviadas las instrucciones fuente.

En otras palabras, la **pasarela** debe ser capaz de ejecutar cualquier instrucción SQL no planeada en un tipo de Base de Datos.

Hacer transformaciones entre los tipos de datos de cada DBMS incluyen subproblemas ya que tienen que ver con situaciones tales como las diferencias en el procesador, las diferencias en el código de caracteres, las diferencias en el formato de punto flotante, etcétera.

Transformar la información de retroalimentación de Oracle (códigos de retorno, etcétera) al formato de Ingres.

Transformar el catálogo de Oracle al formato de Ingres, para que el sitio Ingres y los usuarios que están en este sitio puedan saber lo que contiene la base de datos Oracle.

Servir como participante en (la variante de Ingres para) el protocolo de confirmación de dos fases (suponiendo que se permita que las transacciones de Ingres realicen actualizaciones en la base de datos Oracle). La posibilidad de que la pasarela realice esta función, dependerá de las propiedades proporcionadas por el administrador de transacciones en el sitio.

Middleware para acceso a datos

Las pasarelas, sufren varias limitaciones. Una es que proporcionan poca independencia de ubicación. Otra es que la misma aplicación puede necesitar varias pasarelas distintas, p.e. una para PostgreSQL, otra para Oracle y otra más para MySQL.

Por consecuencia, los productos del tipo pasarela han aparecido regularmente en los últimos años con cada vez más funcionalidad sofisticada. De hecho, todo el negocio de lo que ha venido siendo llamado middleware (también conocido como mediadores) es ahora una industria importante.

Los middleware se adaptan a las diferentes plataformas para su correcto funcionamiento y para que todo proceso se realice de manera transparente de tal manera que no importe que sistema operativo se utilice, la comunicación se realiza con total flexibilidad.

8 Conclusiones

- En este documento se ha intentado dado una visión general de los problemas y características de una base de datos distribuida. Se ha tocado el tema de las técnicas de fragmentación horizontal y vertical, las técnicas son sencillas y son de fácil el entendimiento.
- Además, están apareciendo métodos de fragmentación mixta como el que se ha comentado. Si bien, estos métodos son enfoques formales

más que prácticos, aun siendo desarrollados.

- Los sistemas gestión de base de datos distribuidos heterogéneas son una realidad que aún están limitados por la experiencia actual. También necesitan seguir afrontando distintos entornos y variaciones durante un tiempo
- Pese a la aparición de los métodos de bases de datos distribuidas hace ya años, parece que el salto de lo centralizado a lo distribuido a escala comercial está por venir. Todavía no se ha extendido suficientemente el esquema distribuido, pero se espera que próximamente se produzca el avance definitivo.
- Considerando los dos componentes básicos de los sistemas de bases de datos distribuidos (la propia base de datos y la red de ordenadores) y además, la situación actual de la informática. Si las bases de datos es una de las ramas más antiguas e importantes de la informática, muchas empresas compran ordenadores para dedicarlos exclusivamente a la gestión de sus datos, como parece ser que se ha asumido por parte de todo tipo de empresarios los beneficios que acarrea la conexión de los ordenadores, la instalación de una red, se puede concluir diciendo que el terreno ya está abonado para su extensión comercial.
- Sólo hace falta que determinadas multinacionales decidan apostar más fuerte por este enfoque a través de sus famosos sistemas gestores de bases de datos y que se produzca la consolidación de la resolución de los problemas que el enfoque distribuido acarrea.

References

[1] *Web Autores*:Antolino Hernandez Anastacio *Título*:Base de datos Distribuidas,*Pagina Web*: http://antares.itmorelia.edu.mx/∽antolino/bdd-master-Precent_BDD-3.ppt

[2] *Libro Autores*:Carlos A. Olarte *Título*:Base de datos distribuido

[3] *Libro Autor*:C.J Date *Título*:Introducción a los sistemas de bases de datos, Séptima Edición, 2001.

[4] *Libro Autores*:Oscar Pastor Lopez, Pedro Besa Pons *Título*:Gestion de Bases de datos, Edición Universidad Politécnica de Valencia. Pag 213

[5] *Acta Autores*:Oscar Pastor Lopez, Pedro Besa Pons *Título*:Actas de las VIII Jornadas de Concurrencia , Cuenca, Junio 2000, Pag 1906