Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Lecturas en Ciencias de la Computación
ISSN 1316-6239

Fundamentos de los Sistemas de Bases de Datos Distribuidas

Renny Hernández

ND 2013-01

Centro de Investigación en Sistemas de Información (CISI)
Caracas, febrero 2013

Fundamentos de los Sistemas de Bases de Datos Distribuidas

Renny A. Hernández

Junio 2012

1. Conceptos Fundamentales

La tecnología en las Bases de Datos Distribuidas surge al unir los enfoques propuestos en los sistemas de bases de datos y las redes de computadoras. En los sistemas de bases de datos existe un control y administración del procesamiento de datos centralizado, esto permite que cada aplicación que acceda a los datos no se vea afectada con cualquier cambio en la estructura física o lógica en la base de datos y viceversa. Por otro lado, las redes de computadoras promueven un modo de trabajo que dista de los enfoques de centralización propuestos en los SBD mencionados anteriomente. En este sentido, el objetivo más importante de los Sistemas de Bases de Datos es la integración de los datos operacionales de una organización y no la centralización de éstos. Aunque la presencia de una de estas cualidades no implique la otra, es posible obtener integración de los datos sin que exista un control centralizado, y esto es exáctamente lo que las Bases de Datos distribuidas plantean. A continuación, discutiremos brevemente el concepto general de los sistemas distribuidos para finalizar con algunos conceptos inherentes a los Sistemas de Bases de Datos Distribuidas.

1.1. Sistemas Distribuidos

Podemos definir un sistema distribuido, sistema de computación distribuída o sistemas de procesamiento distribuido como una composición de un número de elementos autónomos de procesamiento, no necesariamente homogeneos, interconectados mediante una red de computadoras que cooperan

entre sí para realizar cualquier tarea asignada. Se entiende por elemento de procesamiento a cualquier computador capaz de ejecutar un programa por sí mismo [OV11].

En los sistemas distribuidos se conocen distintos tipos de distribución: la distribución por procesamiento lógico, distribución por función, distribución por datos y distribución por control. En la definición de sistemas distribuídos presentada anterormente, se evidencia una distribución implícita del procesamiento lógico al asumir que distintos elementos de procesamiento son distribuidos. La distribución de función consiste en delegar funciones a los distintos elementos de hardware o software dentro del sistema. La distribución de acuerdo a los datos consiste en distribuir los datos utilizados por las aplicaciones por los distintos elementos de procesamiento. Por último, el procesamiento por control de las ejecución de las distintas tareas a realizar, en lugar de que éste sea realizado por un sólo computador.

Muchas de las tecnologías utilizadas actualmente están inherentemente distribuidas (aplicaciones web, Comercio electrónico sobre el Internet, aplicaciones multimedia con servicios news-on-demand, entre otras). El motivo principal para realizar cualquier procesamiento distribuido es el de lidiar con los problemas de gestión de datos a gran escala utilizando técnicas basadas en la regla *divide and conquer*. Si se desarrolla el soporte de software necesario para el procesamiento distribuido, es posible dividir un problema complejo en problemas más pequeños y asignarlos a diferentes grupos de programas que trabajan en distintas computadoras y producen un sistema que ejecutado en múltiples elementos de procesamiento podrá ejecutar mucho más eficientemente una tarea en particular. Los Sistemas de Bases de Datos Distribuidas se pueden ver dentro de este contexto y tratarse como herramientas capaces de realizar el procesamiento de datos distribuido de una manera rápida y eficiente.

2. Sistemas de Bases de Datos Distribuidas

Una base de datos distribuida es una colección de multiples bases de datos logicamente interrelacionadas sobre una red de computadoras. Un Sistema Manejador de Bases de Datos Distribuidas se define como el sistema de software que permite la gestión de una base de datos distribuida y hace transparente la distribución. Frecuentemente el término Sistema de Bases de Datos Distribuidas (SBDD) es utilizado para referirse a los Sistemas Mane-

jadores de Bases de Datos Distribuidas. Para ambos términos, los conceptos de interrelación lógica y de distribución sobre una red de computadoras juegan un papel importante que ayuda a definir los Sistemas de Bases de Datos Distribuidas y descartar algunas casos donde se aceptan algunas propuestas como tales.

Un SBDD es una colección de archivos almacenados individualmente en cada nodo de la red. Estos archivos, además de tener una relación lógica, deben contar con una estructura y proveer un acceso mediante una interfaz común. Existen en la actualidad muchos sistemas que poseen funcionalidades de los SMBD y trabajan sobre datos semi-estructurados, almacenados en archivos sobre el internet (e.g. Paginas web). Es importante distinguir entre un SBDD que cumple con las restricciones mencionadas y los sistemas distribuidos de manejo de datos que proveen acceso a datos de la misma tratando de emular los SMBD. También se asume que la distribución física de los datos no es el problema más importante en estos enfoques, tomando como una Base de Datos Distribuida a un conjunto de bases de datos relacionadas que residen en un computador. Sin embargo, la distribución de los datos en los distintos nodos trae como consecuencia problemas que no se encuentran al tener múltiples bases de datos en un sólo computador. Por ejemplo, los datos pueden estar duplicados en un ambiente distribuido, por lo que la BDD debe estar diseñada de manera que la base de datos entera o partes de ésta sean fragmentadas y distribuidas a través de una red de computadoras, trayendo como consecuencia que deba tomarse en cuenta cuál es la versión de los datos que debe accederse en cada lectura y cómo deben realizarse los cambios en los distintos nodos al momento de una modificación. Este inconveniente, entre otros como los relacionados a la fallas de la red de computadores y la sincronización de transacciones entre los distintos nodos, presentan una complejidad y retos que sólo se encuentran en los sistemas de bases de datos distribuidos.

El hecho de que exista una distribución física entre los nodos no implica necesariamente que éstos estén geográficamente distantes, es decir, que pueden estar en el mismo cuarto. La distribución implica símplemente que la comunicación entre éstos nodos deba hacerse por medio de una red de computadoras como único recurso compartido, en vez de realizarse a traves de sistemas de memoria o disco compartido (como los vistos en sistemas multiprocesadores). Esto sugiere que los sistemas de bases de datos multiprocesadores no sean considerados SBDDs. La diferencia entre estos reside en el modo de operación. El diseño de un sistema multiprocesador, consiste

comunmente de un conjunto de procesadores y elementos de memoria idénticos controlado por una o más copias del mismo sistema operativo, que lleva un control estricto de las tareas asignadas. Esto es totalmente distinto en los sistemas distribuidos, donde la heterogeneidad del sistema operativo así como de los componentes de hardware es bastante común. Los sistemas de bases de datos que corren sobre sistemas multiprocesadores son llamados bases de datos paralelas.

A pesar de la presencia de una red, en los sistemas de bases de datos distribuidas, las bases de datos no residen sólo en un nodo de la red. En este caso, los problemas de gestión de datos no son distintos a los vistos en los ambientes de bases de datos centralizados (las bases de datos cliente/servidor son un ejemplo de estos sistemas, donde existe un nodo que gestiona la base de datos de manera centralizada).

En resumen, la existencia de redes computadores o de una colección de archivos no es suficiente para formar un sistema de bases de datos distribuidas. Nuestros intereses se enfocan a un ambiente donde los datos están distribuidos en distintos nodos de una red.

2.1. Ventajas

La administración de bases de datos distribuidas ha sido propuesta por varias razones que van desde la centralización organizativa y el procesamiento económico y autónomo. A continuación, se describen algunas de las ventajas de estos sistemas.

Administración de datos distribuidos con distintos niveles de transparencia.

De manera ideal, un DBMS debe ser una distribución transparente en el sentido de abstraer al usuario de detalles de localización física de los archivos dentro del sistema. Por ejemplo, una taba puede estar fragmentada en conjunto de filas y replicada en distintos nodos ubicados en lugares geográficamente distintos. En este caso, son posibles los siguientes tipos de transparencias:

- **Transparencia de distribución o red:** hace referencia a la autonomía del usuario de los detalles operacionales de la red. Puede dividirse en transparencia de localización y de denominación. La transparencia de localización menciona el hecho de que un comando usado para llevar

a cabo una tarea es independiente de la ubicación de los datos y del sistema desde el que se ejecutó dicho comando. La transparencia de denominación implica que una vez especificado un nombre, puede accederse a los objetos nombrados sin ambigüedad y sin necesidad de ninguna especificación adicional.

- **Transparencia de replicación:** pueden almacenarse copias de los datos en distintos lugares para disponer de una mayor disponibilidad, rendimiento y fiabilidad. La transparencia de replicación permite al usuario abstraerse de las réplicas existentes.

- **Transparencia de Fragmentación:** existen dos tipos de fragmentación, la fragmentación horizontal que distribuye una relación en conjuntos de tuplas, mientras que la vertical lo hace en subrelaciones, donde cada relación está definida por un subconjunto de las columnas de la relación individual. Debido a esto, una consulta podría ser transformada en distintas subconsultas fragmentadas. La transparencia de fragmentación abstrae al usuario de la existencia de los fragmentos.

- **Transparencia de diseño y ejecución:** hacen referencia a la abstracción del usuario de saber cómo está diseñada la base de datos distribuida y dónde ejecuta una transacción.

Todas estas formas de transparencia proveen un acceso fácil y eficiente a los distintos usuarios (especialmente a los usuarios inexpertos). Sin embargo, cumplir con un completo nivel de transparencia representa un conflicto entre la facilidad de uso y la dificultad y costo de procesamiento inherente. Se discute que la transparencia total en un SMBDD hace la gestión de las bases de datos distribuidas pobres en los aspectos de manejabilidad, modularidad y desempeño en el envío de mensajes [Gray89]. Estas discusiones han llevado a la implementación de enfoques como los descritos en las arquitecturas cliente servidor y a la identificación de capas correspondiente a los distintos servicios de transparencia que pueden proveerse.

La primera capa provee acceso transparente a los recursos de datos y se conoce como capa de acceso. Las características de transparencia se construyen en un lenguaje de usuario, quien traduce los servicios requeridos en operaciones. En otras palabras, el compilador o intérprete toma la tarea sin que el implementador sea provisto por algún servicio de transparencia. La

segunda capa de transparencia se conoce como capa de nivel de sistema operativo, implementando la misma transparencia conocida en la definición de sistemas operativos, permitiendo acceder a recursos de hardware y software abstrayendo al usuario de la complejidad inherente a estas operaciones. Este nivel puede extenderse a los ambientes distribuidos, donde la gestión de la red puede ser llevada a cabo por el sistema operativo distribuido o el middleware, si el SMBDD está implementado sobre uno. Este enfoque conlleva problemas como la posiblidad de que el sistema distribuido no posea un nivel de transparencia frente al uso de la red razonable y la necesidad de las aplicaciones de conocer los elementos que son encapsulados para su uso en tareas de optimización y desempeño. Finalmente se tienen la capa del SMBD y la capa de lenguaje. En la capa de SMBD se proveen algunas primitivas del SMBD para realizar ciertas tareas, dejando al SMBD como responsable de la traducción desde el sistema operativo a interfaces de usuario de alto nivel. La capa de transparencia del lenguaje permite el acceso a datos mediante distintos lenguajes de alto nivel como lenguaje

Incremento de la fiabilidad y la disponibilidad.

Consideradas las más importantes de las ventajas de las bases de datos distribuidas. La fiabilidad está definida ampliamente como la probabilidad de que un sistema esté funcionando en un instante de tiempo cualquiera, mientras que la disponibilidad es la probabilidad de que el sistema esté continuamente disponible durante un intervalo de tiempo.Cuando un nodo del SMBD falla, sólo no estarán disponible los datos y el software almacenado en ese nodo, mientras que el resto del sistema continúa operativo. Se logra una mejora en estas dos características al mantener réplicas de datos en más de una ubicación. En un sistema centralizado, el fallo de la ubicación provoca la caída del sistema para todos. En una base de datos distribuida, parte de la información puede estar inaccesible, pero sí se podrá acceder a la parte de la base de datos almacenada en los demás nodos.

Mejoras en el rendimiento.

Un SMBDD fragmenta la base de datos manteniendo la información lo más cerca posible del punto donde es más necesaria. La localización de datos reduce el enfrentamiento en la asignación de CPU y los dispositivos de E/S.

Facilidad en la escalabilidad.

En un entorno distribuido, la escalabilidad en términos de manejo de una mayor cantidad de datos, el incremento de las bases de datos o la adición de más procesadores es mucho más sencilla.

Junto a estas ventajas, Date propone 12 reglas fundamentales [Dat01] que deben cumplir los SBDD, que exponen características de de transparencia, fiabilidad, disponbilidad y escalabilidad. Estas reglas, en su mayoría no son independientes entre sí y tampoco son igualmente importantes para el conjunto total de usuarios finales de un Sistema de Bases de Datos Distribuidas.

Ejercicio 1 — Consulte la bibliografía necesaria e investigue las 12 reglas fundamentales que deben cumplir los SBDD postuladas por Christopher Date.

2.2. Desventajas

Procesamiento de consultas. Se trata de diseñar algoritmos que analizan las consultas y las convierten en operacion de manipulación de datos. El problema es cómo decidir qué estrategia utilizar al ejecutar cualquier consulta sobre la red de la manera menos costosa y efectiva. Los factores a considerar son la distribución de los datos, los costos de comunicación y la falta de información suficiente que esté localmente disponible. El objetivo es optimizar el paralelismo implícito para mejorar el desempeño de la ejecución de la transacción, tomando en cuenta los factores mencionados. Este problema es de naturaleza NP-Completo y los enfoques utilizados normalmente utilizan técnicas heurísticas.

Administración del catálogo. El catálogo contiene información como la descripción y localización de los elementos de la base de datos. Los problemas relacionados a la gestión del catálogo se basan en cómo las bases de datos y las aplicaciones se ejecutan y cómo deben colocarse a través de los sitios. Un directorio puede ser:

1. Centralizado: el catálogo total es almacenado exactamente una vez en un sitio central.
2. Completamente replicado: el catálogo es almacenado por completo en cada uno de los sitios.

3. Dividido: cada sitio mantiene su propio catálogo de los objetos que están almacenados en ese sitio. El catálogo total es la unión de todos los catálogos locales disjuntos.
4. Centralizado y dividido: cada sitio mantiene su propio catálogo local, además, un único sitio central mantiene una copi unificada de todos esos catálogos locales.

Propagación de la actualización. Un problema básico que existe con la replicación de datos es que una actualización a cualquier objeto lógico dado debe ser propagada a todas las copias almacenadas a ese objeto. Una dificultad inmediata es que un sitio que mantiene una copia del objeto podría no estar disponible (debido a una falla del sitio o de la red) en el momento de la actualización. Un esquema común para atacar este problema (aunque no es el único posible) es el llamado esquema de copia primaria que funciona de la siguiente manera:

1. A una copia de cada objeto replicado se le designa como copia primaria. Todas las demás son copias secundarias.
2. Las copias primarias de diferentes objetos están en diferentes sitios (por lo tanto, éste es nuevamente un esquema distribuido)
3. Decimos que las operaciones de actualización quedaron lógicamente terminadas tan pronto como se actualizó la copia primaria. Entonces, el sitio que mantiene esa copia es responsable de la propagación de la actualización hacia las copias secundarias en algún tiempo subsecuente (Ese tiempo subsecuente, debe ser previo al `COMMIT`, si es que se van a conservar las propiedades ACID de la transacción)

Control de la concurrencia. En la mayoría de los sistemas distribuidos (al igual que en la mayoría de los sistemas centralizados) el control de concurrencia se basa en el bloqueo [Dat01]. Sin embargo, en un sistema distribuido, las solicitudes para probar, colocar y liberal bloqueos se convierten en mensajes (suponiendo que el objeto en consideración está en un sitio remoto) y los mensajes significan una sobrecarga. Por ejemplo, si una transacción T que necesita actualizar un objeto para el cual existen réplicas en n sitios remotos. Si cada sitio es responsable de los bloquesos por los objetos que están almacenados en ese

sitio (como sucedería si existe autonomía local), entonces una implementación directa requerirá al menos $5n$ mensajes (n solicitudes de bloqueo, n otorgamientos de bloqueo, n mensajes de actualización, n notificaciones, n solicitudes de desbloqueo). Por supuesto, podemos resolver esto facilmente si solapamos los mensajes (por ejemplo, es posible combiar los mensajes de solicitud de bloqueo y los de actualización, así como los mensajes de garantía de bloqueo y los de notificación), pero aún así tendríamos un tiempo total varias veces mayor en comparación con un sistema centralizado. El enfoque usual es el de adoptar la estrategia de copia primaria, utilizada al abordar el problema de propagación de actualización. Para un objeto dado A, el sitio que mantiene la copia primaria manejará todas las operaciones de bloqueo que involucren a ese A. Bajo esta estrategia, el conjunto de todas las copias de un objeto puede ser considerado como un sólo objeto para efectos de bloqueo, y la cantidad total de mensajes se reduce de $5n$ a $2n + 3$ (una solicitud de bloqueo, una garantía de bloqueo, n actualizaciones, n notificaciones y una solicitud de desbloqueo). Observemos que esta solución implica una pérdida valiosa de la autonomía; si una copia no está disponible, la transacción puede fallar aún cuando sea de sólo lectura. Por lo tanto, un efecto colateral del uso de esta técnica es la reducción del rendimiento y de la disponibilidad para las recuperaciones y actualizaciones. Otro problema con el bloqueo en un sistema distribuido es el que puede conducir a un abrazo mortal global. Un abrazo o bloqueo mortal global es aquél que involucra a dos o más sitios. Por ejemplo:

1. El agente de una transacción $T2$ en un sitio X está esperando al agente de la transacción T1 del sitio X para que libere el bloqueo.
2. El agente de una transacción $T1$ en el sitio X está esperando a que termine el agente de una transacción $T1$ en el sitio Y.
3. El agente de la transacción $T1$ en el sitio Y está esperando que el agente de la transacción $T2$ en el sitio Y libere un bloqueo.
4. El agente de la transacción $T2$ en el sitio Y está esperando que termine el agente de la trasnacción $T2$ en el sitio X. Aquí sucede el abrazo mortal.

La especificación de la arquitectura R* [WDH+82] cuenta con soluciones elegantes y distribuidas para detectar abrazos mortales globales.

En esta arquitectura también se implementan soluciones basadas en temporizadores para la prevención de estas situaciones.

Ejercicio 2 — Investigue sobre las características de los SMBDD SQL/Server, Oracle y SyBase, tomando en cuenta sus ventajas, desventajas, enfoque de distribución, y aspectos técnicos relevantes.

3. Arquitecturas de los SMBDD

La arquitectura de un sistema define su estructura, tomando en cuenta componentes identificados, la función de cada uno de éstos, su interrelación e interaccion entre sí. La especificación de la arquitectura de un sistema, requiere identificación de los modelos, interfaces y relaciones, tomando en cuenta los datos y el flujo de control a través de éste.

A continuación, se presentan tres modelos de arquitectura propuestos para los SBDD: Arquitectura Cliente/Servidor (*Client/Server*), *peer-to-peer* y sistemas multi bases de datos o sistemas de bases de datos múltiples. Para comenzar, presentamos el conocido modelo de arquitectura ANSI-SPARC y las posibles alternativas para su implementación en un SBDD. Ésto nos permitirá establecer relaciones entre éste y los distintos modelos de arquitectura.

3.1. Arquitectura ANSI-SPARC

En 1972, un grupo de estudios establecido por el Comité de Computadores y Procesamiento de Información (X3) del ANSI, patrocinado por el Comité de Panificación de Estándares y Requerimientos (SPARC) estudió la factibilidad de establecer estándares en el área de la gestión de bases de datos, así como determinar cuáles aspectos era conveniente establecer dichos estándares. Así surgió en 1977 el reporte final con un marco de trabajo propuesto que luego se conoció como la "Arquitectura ANSI/SPARC". Una versión simplificada de la arquitectura es ampliamente estudiada en el área y consiste de tres capas o niveles de datos: la capa externa o el nivel de vistas, que interactúa con el usuario, la capa interna relacionada con el nivel físico y la interacción con el computador y un nivel conceptual o empresarial.

En la arquitectura ANSI/SPARC (Figura 1), el nivel más bajo es la capa interna, que maneja la definición física y la organización de los datos: localización de los datos en los distintos dispositivos de almacenamiento y los mecanismos de acceso utilizados para acceder y manipular datos. Por otro

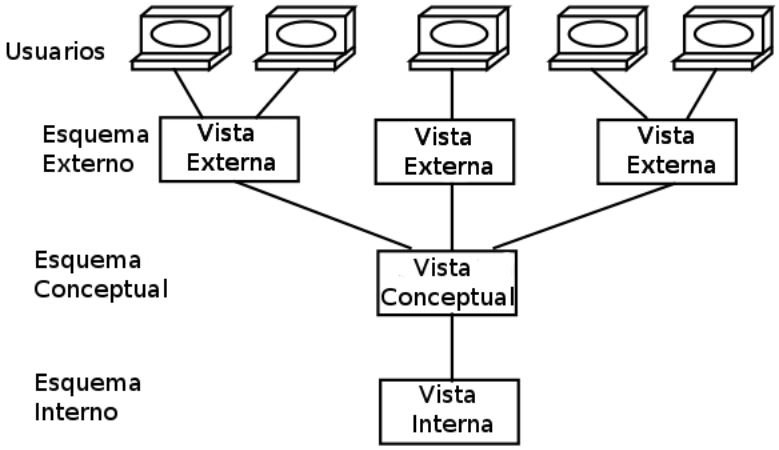

Figura 1: Arquitectura ANSI/SPARC

lado, el nivel externo se encarga de cómo son presentados los datos y cómo el usuario ve la base de datos; el uso de vistas individuales que representan la porción de la base de datos que pueda ser accedida por un usuario en partícular. Estas vistas pueden ser compartidas por distintos grupos de usuarios, conformando así un esquema externo. Entre estos dos niveles se encuentra el esquema conceptual, el cual es una abstracción de la definición de la base de datos, viene siendo la vista o el concepto que se tiene de la base de datos, el modelo de la base de datos enfocada a un contexto o empresa en particular. En teoría, el nivel conceptual permite suponer algunos aspectos de la representación y las relaciones entre los datos sin considerar los requerimientos de aplicaciones en particular o las restricciones del medio de almacenamiento físico. Sin embargo, estos aspectos son importantes en la práctica y deben ser considerados para evitar problemas de desempeño. La definición de uno de estos esquemas puede ser descrito a partir de un mapeo de la especificación del esquema anterior.

Este enfoque es importante, ya que nos permite evidenciar los conceptos básicos de la independencia de datos. La separación entre los esquemas externos y el nivel conceptual provee una independencia lógica de los datos, mientras que la separación entre el nivel conceptual y físico permiten mantener la independencia física de datos.

4. Modelos Arquitectónicos para los Sistemas de Bases de Datos Distribuidas

En la actualidad existen más de 15 modelos arquitecturales para las SBDD. A continuación, discutiremos los conceptos relacionados a los modelos Cliente/Servidor, *peer to peer* y Multi Bases de Datos. Estos modelos pueden ser clasificados según autonomía de los sistemas locales, su distribución y su heterogeneidad:

Autonomía. Se refiere a la distribución del control, más no de los datos. La autonomía indica el grado de independencia de los SMBD individuales. Los requerimientos de un sistema autónomo se definen a continuación:

1. Las operaciones locales del SMBD individual no son afectadas por su participación en el sistema distribuido
2. La manera en la que el SMBD procesa y optimiza consultas no debe ser afectada por la ejecución de las consultas globales que acceden a las múltiples bases de datos
3. La consistencia o la operacionalidad del sistema no debe comprometerse cuando algún SMBD se una o deje el sistema distribuido.

Asimismo, las dimensiones de la autonomía se definen tomando en cuenta los siguientes aspectos:

1. Autonomía de Diseño. Todos los SMBD que sirven de nodos son libres de usar los modelos de datos y técnicas de gestión de transacciones que consideren
2. Autonomía de Comunicación. Cada nodo es libre de tomar sus propias decisiones sobre qué información se provee a los demás nodos o el software que controle la ejecución global
3. Autonomía de Ejecución. Cada SMBD puede ejecutar las transacciones que reciba de modo que prefiera

A continuación, se describe una clasificación que cubre aspectos importantes discutidos anteriormente [OV11]:

1. Sistemas altamente Integrados. Una sóla imagen de la base de datos que puede estar almacenada en múltiples bases de datos es

disponible para cualquier usuario. En este enfoque, los gestores de datos son implementados de tal manera que uno de éllos sea el que tenga el control de procesamiento de las peticiones del usuario. Los gestores de datos no trabajan como típicos SMBD independientes aunque tengan la funcionalidad para que operen de esa manera.

2. Sistemas Semiautónomos. Los SMBD pueden operar independientemente, pero deciden participar en una "federación" de modo que puedan compartir sus datos locales. Cada uno de estos sistemas determina que parte de su propia base de datos será accesible a los demás SMBD. No son sistemas completamente autónomos, ya que necesitan modificaciones para permitir el intercambio de información entre sí.
3. Sistemas totalmente aislados. Cada sistema individual es autónomo y no conoce la existencia de otros SMBD ni como comunicarse con éstos. En estos sistemas no existe un control global por lo que se dificulta el procesamiento de transacciones y de peticiones de datos.

Distribución. Se refiere a la descentralización del control, la dimensión de distribución de esta clasificación se encarga de los datos. Consideramos una distribución física de los datos sobre múltiples sitios, y el usuario los ve como una sóla estructura lógica. Esto nos lleva a dos alternativas importantes: la distribución cliente/servidor y la distribución peer-to-peer (o distribución completa). Los sistemas cliente servidor se distinguen nodos que actúan como clientes y servidores con funcionalidades diferentes, mientras que en los sistemas peer-to-peer no existe una distinción de máquinas clientes y servidores, y cada máquina tiene la capacidad de comunicarse con otras y ejecutar consultas y transacciones.

Heterogeneidad. La heterogeneidad puede ocurrir de distintas formas en los sistemas distribuidos, desde clasificaciones de acuerdo al hardware y los diferentes protocolos de red hasta variaciones en los manejadores de datos. Además, la heterogeneidad cubre el uso de distintos modelos de datos, sin dejar a un lado los casos donde se tiene el mismo modelo de datos, pero existen diferencias en cuanto al lenguaje utilizado.

4.1. Arquitectura Cliente/Servidor

La idea principal de los sistemas Cliente/Servidor es la de diferenciar las funcionalidades quese proveen y dividirlas en dos clases: funciones de servidor y funciones de cliente. Esto provee una arquitectura a dos niveles, lo que facilita el manejo de complejidad en SMBD modernos, así como la complejidad en la distribución. Esta idea surge del enfoque utilizado por los sistemas compuestos por un servidor de aplicaciones conectado a un servidor de bases de datos, donde el servidor de aplicaciones realiza algunas funcionalidades y está conectado a un sistema de bases de datos o servidor de bases de datos mediante una red de computadores. La asignación de funcionalidades entre el cliente y el servidor difiere según las implementaciones. Por ejemplo, en sistemas relacionales, el servidor realiza la mayoría de las operaciones de gestión de datos mientras que en cada nodo cliente se tiene un cliente SMBD que es puede ser responsable de la gestión de datos localmente almacenada. Obviamente, se tienen sistemas operativos ejecutándose en ambos lados de la arquitectura y que se comunican por la red, comunmente a nivel de sentencias SQL y respuestas o relaciones resultado.

Existen distintos tipos de arquitectura modelo servidor. La implementación más sencilla es donde un servidor accesado por múltiples clientes conocido como múltiples clientes/un servidor. Este enfoque no dista de un sistema centralizado, pero presenta algunas diferencias de diseño en torno a ejecución de transacciones y manejo de cachés.

Una arquitectura más sofisticada es la de múltiples clientes/múltiples servidores, donde surgen dos alternativas para la gestión: Cualquier cliente tiene su propia conexión al servidor apropiado o cada cliente conoce cuál es su servidor local que se encarga de comunicarse con los demás servidores si fuera necesario. El primer enfoque sobrecarga el cliente de responsabilidades mientras el enfoque multiple cliente-multiple servidor concentra la gestión de funcionalidaes a los servidores, dando como resultado que la transparencia de acceso a los datos esté del lado de la interfaz de servidores y que exista el concepto de clientes ligeros.

Desde la perspectiva de lógica de datos, los sistemas cliente/servidor no son muy diferentes a los sistemas *peer-to-peer*; ambos dan una noción de una sóla base de datos lógica mientras que los datos físicos están distribuidos. Esto nos brinda un nivel de transparencia que en ambos casos es provisto por distintos paradigmas arquitecturales implementados.

Los sistemas cliente/servidor han permitido extender distribuciones más

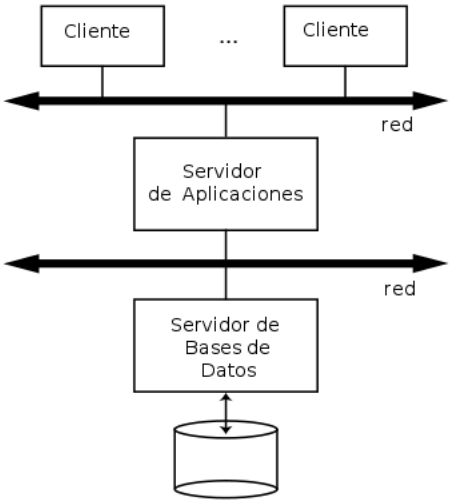

Figura 2: Modelo Multiples Cliente/Servidor

eficientes, como por ejemplo, realizar una distribución adicional en los diferentes tipos de servidores (Figura 2): los clientes ejecutan la interfaz de usuario, los servidores de aplicaciones ejecutan programas de aplicación y finalmente, los servidores de bases de datos realizan las funciones de gestión de la base de datos. Bajo este enfoque, se presenta una arquitectura a tres capas, donde los nodos se organizan como servidores especializados en vez de servir como máquinas de propósito general. Adicionalmente, se tiene el diseño bajo múltiples servidores de bases de datos y múltiples servidores de aplicaciones, donde cada servidor de aplicaciones está dedicado a unas cuantas aplicaciones, mientras que los servidores de bases de datos operan en distintas formas de modo de explotar técnicas para incrementar la fiabilidad y disponibilidad, así como mejorar el desempeño de la gestión de la bases de datos en general. Esta mejora en la gestión de la base es debido a la alta integración del sistema manejador de bases de datos con el sistema operativo. Finalmente, un servidor de bases de datos puede explotar arquitecturas de hardware de última generación como los multiprocesadores o clusters de servidores para mejorar el desempeño y la disponibilidad de datos.

Aunque estas ventajas son significativas, se ven afectadas por la sobrecarga de procesamiento implícito en las comunicaciones adicionales entre los servidores de aplicación y los servidores de datos. Este problema también se extiende a los sistemas cliente/servidor clásicos, pero en este caso en es-

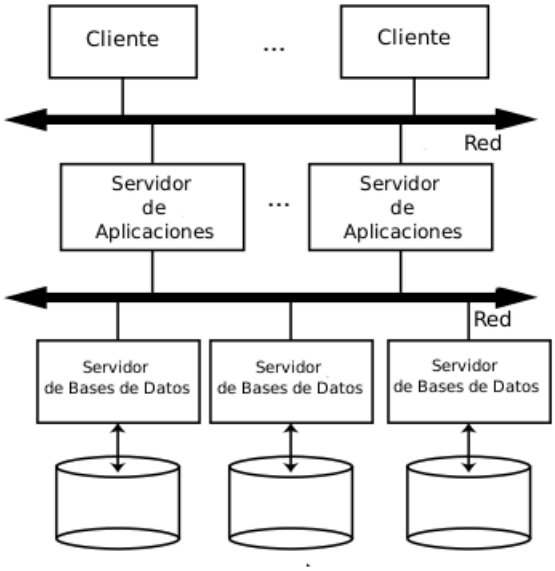

Figura 3: Modelo Multiples Clientes/Multiples Servidores

pecífico, existe una capa adicional de comunicación que no debe ignorarse. El costo relacionado a esta comunicación adicional sólo podría reducirse si la interfaz del servidor es de alto nivel y permite realizar consultas complejas aptas para un procesamiento intensivo de los datos.

El uso de servidores de aplicación (así como el enfoque clásico de la arquitectura cliente/servidor) puede extenderse agregando múltiples servidores de datos y múltiples servidores de aplicación (3). En este caso, cada servidor de aplicación se dedica a una o varias aplicaciones, mientras los servidores de bases de datos operan de distintas formas, tal como se discutió anteriormente.

4.2. Modelo *peer to peer*

En los sistemas *peer to peer*, a diferencia de los sistemas cliente/servidor, no existen diferencias entre las funcionalidades en cada uno de los nodos. Aunque este enfoque ha sido explotado por las aplicaciones de transferencia y compartición de archivos y ha surgido como una alternativa a la gestión de datos en los SMBD distribuidos de última generación, existen dos diferencias entre éstos y sus predecesores: la primera se refiere a la distribución masiva de los datos en los sistemas actuales, anteriormente se trabajaban con algunas

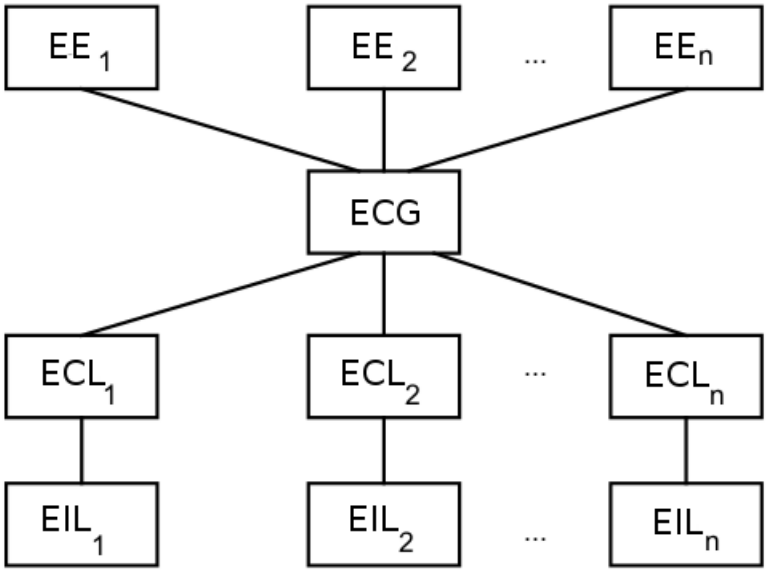

Figura 4: Arquitectura *peer to peer*

decenas de nodos y hoy en día se tienen arquitecturas con miles de sitios. La segunda diferencia es con respecto a la heterogeneidad de cada aspecto de los sitios y su autonomía. Ésto, junto a la distribución masiva, siempre ha sido un problema dentro de las bases de datos distribuidas.

La organización física de los datos en estos sistemas puede diferir de un nodo a otro (Figura 4). Existe una definición de esquema individual para cada nodo, el cual llamaremos esquema interno local (EIL). La vista empresarial de los datos es descrita en el esquema conceptual global (ECG), donde se describe la estructura lógica de los datos en todos los sitios.

Para manejar la fragmentación y la replicación de datos, debe describirse la organización lógica de los datos en cada uno de los sitios, por lo que se tiene una capa en la arquitectura llamada esquema conceptual local (ECL). Adicionalmente, tenemos que el esquema conceptual global es la unión de todos los esquemas conceptuales locales en el sistema. Para finalizar, las aplicaciones y accesos de los usuarios a la base de datos se soportan en los esquemas externos (EE), definidos sobre el esquema global. Es importante acotar que esta arquitectura brinda los niveles de transparencia y de independencia de datos descritos por la arquitectura ANSI/SPARC. La transparencia de red se mantiene mediante la definición de un esquema global, esto permite que las consultas de datos sean realizadas por usuarios independientemente de

la ubicación o los componentes locales que responderán a éstas. El SMBD distribuido traduce estas consultas hechas al esquema global a diferentes esquemas locales que serán ejecutados en los diferentes nodos.

4.3. Arquitectura Multi Bases de Datos

Los sistemas multi bases de datos representan un caso donde los SMBD individuales (distribuidos o no) son completamente autónomos y no hay concepto de cooperación, incluso pueden no conocer la existencia de otros. En nuestro caso, sólo nos interesan los sistemas Multi Bases de Datos Distribuidos o Sistemas de Integración de datos (evitamos este concepto debido a que engloba sistemas que no poseen bases de datos como fuente de datos).

Las diferencias entre los Sistemas Multi Bases de Datos Distribuidos (Multi-SMBD distribuidos) y los sistemas SMBD distribuidos es evidente en sus modelos arquitecturales. Esta reside en la definición del esquema conceptual global. En los SMBD distribuidos logicamente integrados, este esquema global define la vista global de la bases de datos entera, mientras que en el caso de los Multi-SMBD distribuidos, sólo representa una parta de las bases de datos locales que cada nodo SMBD desea compartir (esto se conoce como arquitectura de bases de datos federadas). Cada nodo debe decidir qué datos estarán disponibles, definiendo un esquema de exportación. Además, la definición de una bases de datos global es distinta entre un SMBD distribuido y un multi-SMBD distribuido). En los SMBD distribuidos, la base de datos global es la unión de las bases de datos locales, mientras que en los Sistemas Multi Bases de Datos Distribuidos, es sólo un subconjunto posible de la misma unión. En los Sistemas Multi Bases de Datos Distribuidos, el ECG llamado tambien esquema mediado, se define al integrar los esquemas externos de las bases de datos autónomas locales o posibles partes de sus esquemas conceptuales locales.

Además, los usuarios de un SMBD local definen sus propias vistas en la base de datos local y no necesitan cambiar sus aplicaciones si no desean que éstas accedan a datos desde otra base de datos. Esto obviamente genera un problema de autonomía.

El diseño del Esquema Conceptual Global en los sistemas Multi Bases de Datos se refiere a la integración de los esquemas conceptuales locales (ECL) o los esquemas externos locales (EEL) (Figura 5). Una diferencia entre el diseño en los sistemas multi bases de datos y en los SMBD distribuidos lógicamente integrados es que en los últimos el mapeo se realiza desde un esquema con-

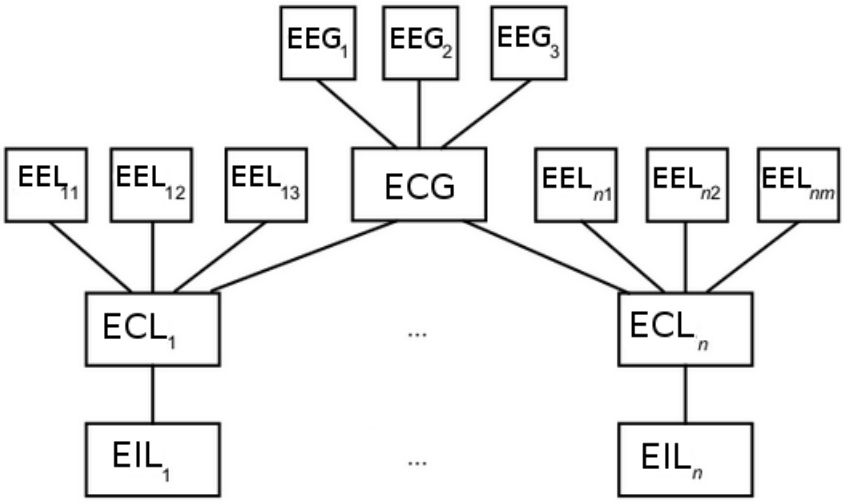

Figura 5: Esquemas lógicos de los sistemas multi-SMBD

ceptual local al esquema conceptual global. En los sistemas multibases de datos, el mapeo se realiza en dirección contraria.

En cuanto a la definición del modelo de datos y el acceso a los EEG y los ECG, no es necesario que estos sse definan usando el mismo modelo de datos y el lenguajes. Esto determina si el sistema es homogeneo o heterogeneo.

En los Sistemas Multi Bases de Datos heterogeneos, existen dos alternativas de implementación: unilingue y multilingue. La unilingue permite a los usuarios utilizar diferentes modelos de datos y lenguajes posibles al acceder a las bases de datos tanto local como global. Esto implica que un usuario de la bases de datos global es diferente a aquel que acceda solo a una base de datos local, utilizando distintos modelos de datos y diferentes lenguajes.

En una arquitectura multilingüe, la filosofía básica es la de permitir a cada usuario acceder a la base de datos global (datos desde otras bases de datos) utilizando un esquema externo definidio usando el lenguaje del usuario del SMBD local.

La arquitectura basada en componentes de un Sistema Multi Bases de Datos es distinta a un SMBDD. La principal diferencia es la existencia de manejadores que gestionan distintas bases de datos. El multi SMBD provee una capa que se ejecuta por encima de cada SMBD individual y provee funcionalidades al usuario para el acceso a varias bases de datos (Figura 6). En un MDBS distribuido, la capa de multi SMBD puede ejecutarse en un

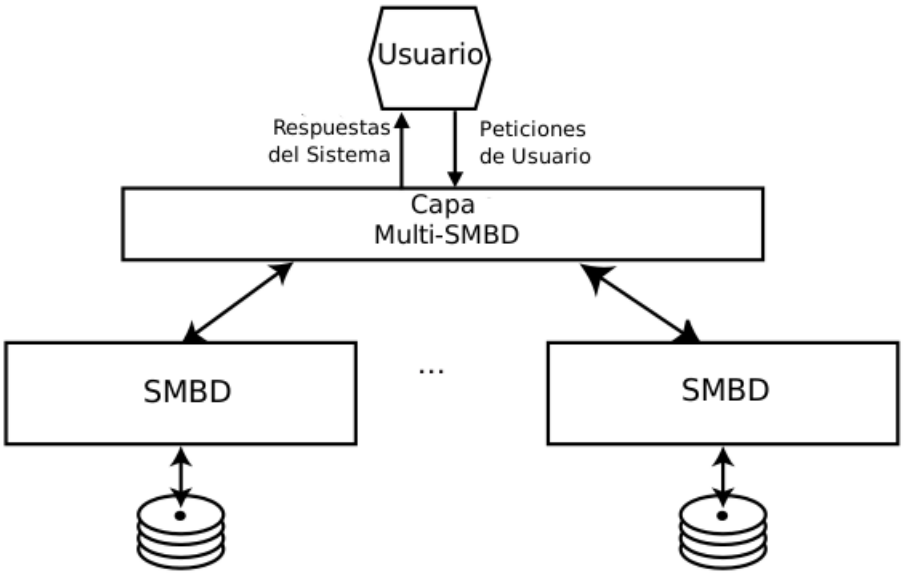

Figura 6: Componentes de un sistema multi SMBD

sólo nodo de manera centralizada o en distintos nodos para ofrecer estas funcionalidades. Finalmente, cada SMBD ignora esta capa viéndola como una aplicación que envía peticiones y recibe respuestas.

Una implementación popular de los Sistemas Multi Bases de Datos es el enfoque *mediator-wrapper*. Un mediador es el módulo de software que explota el conocimiento codificado acerca de ciertos conjuntos o subconjuntos de dato para crear información para aplicaciones de más alto nivel y cada mediador realiza una función en específico con interfaces claramente definidas. Para lidiar con posibles problemas de heterogeneidad, se implementan *wrappers*, cuya tarea es la de proveer un mapeo entre un SMBD que sirve como fuente de datos y los mediadores.

Referencias

[Dat01] Christopher Date. *Introduccion a Los Sistemas de Bases de Datos*. Pearson Educación, 2001.

[EN06] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems (5th Edition)*. Addison Wesley, March 2006.

[GMUW08] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems: The Complete Book*. Prentice Hall Press, Upper Saddle River, NJ, USA, 2 edition, 2008.

[OV11] Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems, 3rd edition*. Springer, 2011.

[WDH+82] R. Williams, Dean Daniels, Linda. Haas, George Lapis, Bruce. Lindsay, Pui Ng, Ron Obermarck, Patricia Selinger, Adrian Walker, Paul Wilms, and Robert Yost. R*: An overview of the architecture. In *JCDKB*, pages 1–27, 1982.