capitulo 23

Conceptos de bases de datos distribuidas

En este capítulo, prestamos atención a los bases de datos (DDB), gestión de bases de datos distribuidas sistemas (DDBMS), y cómo se utiliza la arquitectura cliente-servidor como plataforma para desarrollo de aplicaciones de bases de datos. Las bases de datos distribuidas aportan las ventajas de computación distribuida al dominio de la base de datos. Un **sistema informático distribuido** Consiste en varios sitios de procesamiento o nodos que están interconectados por un red informática y que cooperan en la realización de determinadas tareas asignadas. Como general objetivo, los sistemas informáticos distribuidos dividen un gran problema inmanejable en piezas más pequeñas y resolverlo de manera eficiente y coordinada. Por lo tanto, más computación El poder de ing se aprovecha para resolver una tarea compleja, y el procesamiento autónomo Los nodos se pueden administrar de forma independiente mientras cooperan para proporcionar los funcionalidades para solucionar el problema. La tecnología DDB resultó de la fusión de dos tecnologías: tecnología de bases de datos y tecnología de sistemas distribuidos.

En la década de 1980 se desarrollaron varios sistemas prototipo de bases de datos distribuidas y 1990 para abordar los problemas de distribución de datos, replicación de datos, consulta distribuida y procesamiento de transacciones, gestión de metadatos de bases de datos distribuidas y otros temas. Más recientemente, han surgido muchas tecnologías nuevas que combinan la distribución tecnologías uted y de bases de datos. Estas tecnologías y sistemas se están desarrollando operado para tratar con el almacenamiento, análisis y extracción de grandes cantidades de datos que se están produciendo y recopilando, y se les conoce generalmente como **big data tecnologías** . Los orígenes de las tecnologías de big data provienen de sistemas distribuidos y sistemas de bases de datos, así como algoritmos de minería de datos y aprendizaje automático que puede procesar estas grandes cantidades de datos para extraer el conocimiento necesario.

En este capítulo, discutimos los conceptos que son fundamentales para la distribución de datos y la gestión de datos distribuidos. Luego, en los dos capítulos siguientes, damos una descripción general de algunas de las nuevas tecnologías que han surgido para gestionar y procesar

Big Data. El capítulo 24 analiza la nueva clase de sistemas de bases de datos conocida como NOSQL

841

Página 2

842 Capítulo 23 Conceptos de bases de datos distribuidas

sistemas, que se centran en proporcionar soluciones distribuidas para gestionar grandes cantidades de datos que se necesitan en aplicaciones como las redes sociales, la atención médica y la seguridad, para nombrar unos pocos. El capítulo 25 presenta los conceptos y sistemas que se utilizan para cese y análisis de big data, como map-reduce y otros procesos distribuidos tecnologías de ing. También discutimos los conceptos de computación en la nube en el Capítulo 25.

La Sección 23.1 presenta la administración de bases de datos distribuidas y conceptos relacionados. Problemas de diseño de bases de datos distribuidas, que implican fragmentación y fragmentación de datos y distribuirlo en varios sitios, así como la replicación de datos, se analizan en Sección 23.2. La sección 23.3 ofrece una descripción general del control de concurrencia y la recuperación en bases de datos distribuidas. Las secciones 23.4 y 23.5 introducen el pro- técnicas de procesamiento de consultas cesante y distribuido, respectivamente. Secciones 23.6 y 23.7 introducir diferentes tipos de sistemas de bases de datos distribuidas y su arquitectura turas, incluidos los sistemas federados y de bases de datos múltiples. Los problemas de heterogeneidad También se destacan la autonomía y las necesidades de autonomía en los sistemas de bases de datos federados. La sección 23.8 analiza los esquemas de administración de catálogos en bases de datos distribuidas. Segundo- La sección 23.9 resume el capítulo.

Para una breve introducción al tema de bases de datos distribuidas, las Secciones 23.1 a 23.5 puede cubrirse y las otras secciones pueden omitirse.

23.1 Conceptos de bases de datos distribuidas

Podemos definir una **base de datos distribuida (DDB)** como una colección de múltiples Bases de datos interrelacionadas distribuidas a través de una red informática, y una **distribución sistema de gestión de base de datos (DDBMS)** como un sistema de software que gestiona una base de datos tributada al tiempo que hace que la distribución sea transparente para el usuario.

23.1.1 Qué constituye una DDB

Para que una base de datos se llame distribuida, las siguientes condiciones mínimas deben estar satisfecho:

- **Conexión de nodos de bases de datos a través de una red informática.** Hay múltiples múltiples computadoras, llamadas **sitios** o **nodos** . Estos sitios deben estar conectados por un **red** subyacente para transmitir datos y comandos entre sitios.
- **Interrelación lógica de las bases de datos conectadas.** Es fundamental que el la información en los distintos nodos de la base de datos esté relacionada lógicamente.
- **Posible ausencia de homogeneidad entre los nodos conectados.** No es necesario ruego que todos los nodos sean idénticos en términos de datos, hardware y software.

Todos los sitios pueden estar ubicados en proximidad física, por ejemplo, dentro del mismo edificio o grupo de edificios adyacentes, y conectados a través de una **red de área local** , o pueden ser distribuidos geográficamente en grandes distancias y conectados a través de un **largo** o **ancho red de área** . Las redes de área local suelen utilizar concentradores o cables inalámbricos, mientras que Las redes de larga distancia utilizan líneas telefónicas, cables, infraestructura de comunicación inalámbrica. tures, o satélites. Es común tener una combinación de varios tipos de redes.

Página 3

Las redes pueden tener diferentes **topologías** que definen la comunicación directa caminos entre sitios. El tipo y la topología de la red utilizada pueden tener un significado No puede afectar el rendimiento y, por lo tanto, las estrategias de consulta distribuida. procesamiento y diseño de bases de datos distribuidas. Para problemas arquitectónicos de alto nivel, sin embargo, no importa qué tipo de red se utilice; lo que importa es que cada sitio pueda comunicarse, directa o indirectamente, con cualquier otro sitio. Para el resto de este capítulo, suponemos que existe algún tipo de red entre nodos, independientemente de cualquier topología en particular. No abordaremos ninguna red temas específicos, aunque es importante entender que para una operación eficiente de un sistema de base de datos distribuida (DDBS), diseño de red y problemas de rendimiento son fundamentales y forman parte integral de la solución global. Los detalles de la comprensión La red mentirosa es invisible para el usuario final.

23.1.2 Transparencia

El concepto de transparencia amplía la idea general de ocultar la implementación detalles de los usuarios finales. Un sistema altamente transparente ofrece mucha flexibilidad al usuario final / desarrollador de aplicaciones, ya que requiere poca o ninguna conciencia de los detalles de su parte. En el caso de una base de datos centralizada tradicional, la transparencia Ency simplemente se refiere a la independencia lógica y física de los datos para la aplicación. desarrolladores. Sin embargo, en un escenario de DDB, los datos y el software se distribuyen sobre varios nodos conectados por una red informática, por lo que los tipos adicionales de se introducen transparencias.

Considere la base de datos de la empresa en la Figura 5.5 que hemos estado discutiendo a través de: sacar el libro. Las tablas EMPLOYEE , PROJECT y WORKS_ON pueden estar fragmentadas horizontalmente (es decir, en conjuntos de filas, como veremos en la Sección 23.2) y almacenados con posible replicación, como se muestra en la Figura 23.1. Los siguientes tipos de transparencias encies son posibles:

- **Transparencia de la organización de datos (también conocida como distribución o red transparencia).** Esto se refiere a la libertad para el usuario de la operación detalles de la red y la ubicación de los datos en el sistema distribuido tem. Puede dividirse en transparencia de ubicación y transparencia de nomenclatura. **La transparencia de ubicación se** refiere al hecho de que el comando utilizado para realizar una tarea es independiente de la ubicación de los datos y la ubicación del nodo donde se emitió el comando. **La transparencia de los nombres** implica que una vez

nombre está asociado con un objeto, se puede acceder a los objetos nombrados una de manera ambigua sin especificación adicional sobre dónde se encuentran los datos.

- **Transparencia de replicación.** Como mostramos en la Figura 23.1, copias del mismo Los objetos de datos se pueden almacenar en varios sitios para una mejor disponibilidad, rendimiento mance y fiabilidad. La transparencia de la replicación hace que el usuario no se dé cuenta la existencia de estas copias.

- **Transparencia de fragmentación.** Son posibles dos tipos de fragmentación. **La fragmentación horizontal** distribuye una relación (tabla) en subrelaciones que son subconjuntos de las tuplas (filas) en la relación original; esto también se conoce

Página 4

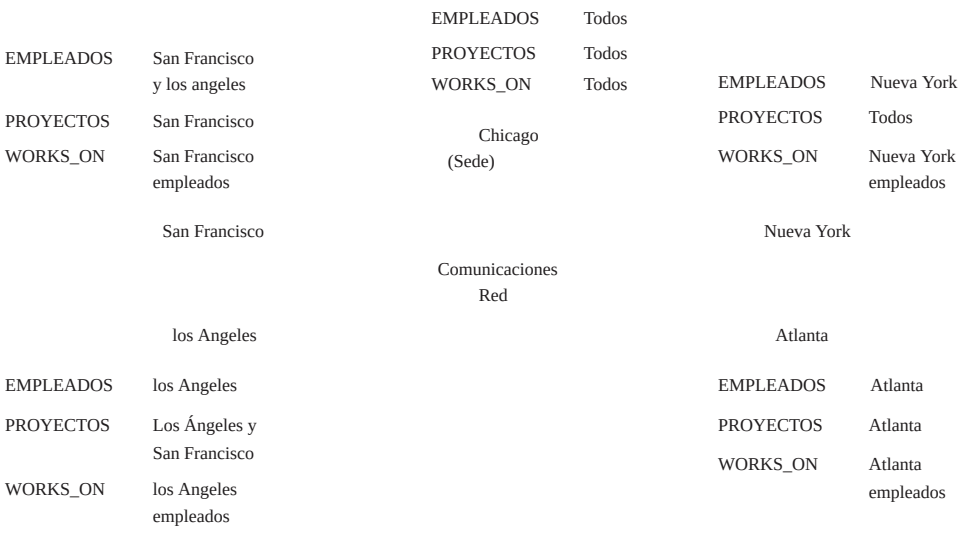

Figura 23.1
Distribución y replicación de datos entre bases de datos distribuidas.

como **fragmentación** en los nuevos sistemas de big data y cloud computing. **Vertical La fragmentación** distribuye una relación en subrelaciones donde cada subrelación está definido por un subconjunto de las columnas de la relación original. Fragmentación la transparencia hace que el usuario desconozca la existencia de fragmentos.

- Otras transparencias incluyen **transparencia** y **ejecución del diseño transparencia,** que se refieren, respectivamente, a la libertad de saber cómo La base de datos distribuida está diseñada y donde se ejecuta una transacción.

23.1.3 Disponibilidad y confiabilidad

La confiabilidad y la disponibilidad son dos de las ventajas potenciales más comunes citadas para bases de datos distribuidas. **La confiabilidad** se define ampliamente como la probabilidad de que un El sistema está funcionando (no inactivo) en un momento determinado, mientras que la **disponibilidad** es la probabilidad de que el sistema esté continuamente disponible durante un intervalo de tiempo. Podemos relacionar directamente la confiabilidad y disponibilidad de la base de datos con las fallas, errores y fallas asociadas con él. Una falla puede describirse como una desviación de la comportamiento de lo que se especifica con el fin de garantizar la ejecución correcta de la operación ciones. **Los errores** constituyen ese subconjunto de estados del sistema que causa la falla. **La culpa** es la causa de un error.

Para construir un sistema que sea confiable, podemos adoptar varios enfoques. Una com El enfoque mon enfatiza la tolerancia a fallas; reconoce que se producirán fallas, y diseña mecanismos que pueden detectar y eliminar fallas antes de que puedan resultar en un

Página 5

fallo de sistema. Otro enfoque más estricto intenta asegurar que el El sistema no contiene fallas. Esto se hace a través de un exhaustivo programa de diseño. proceso seguido de un extenso control de calidad y pruebas. Un DDBMS confiable tolera fallas de los componentes subyacentes, y procesa las solicitudes de los usuarios siempre que los datos no se viola la consistencia básica. Un administrador de recuperación de DDBMS tiene que lidiar con fallas uras que surgen de transacciones, hardware y redes de comunicación. Hardware Las fallas pueden ser aquellas que resultan en la pérdida del contenido de la memoria principal o la pérdida de contenido de almacenamiento secundario. Las fallas de red ocurren debido a errores asociados con mensajes y fallas de línea. Los errores de mensaje pueden incluir su pérdida, corrupción o llegada fuera de servicio al destino.

Las definiciones anteriores se utilizan en los sistemas informáticos en general, donde existe un distinción técnica entre fiabilidad y disponibilidad. En la mayoría de las discusiones relacionadas a DDB, el término **disponibilidad** se utiliza generalmente como un término general para cubrir tanto conceptos.

23.1.4 Escalabilidad y tolerancia de partición

La escalabilidad determina hasta qué punto el sistema puede expandir su capacidad mientras seguir funcionando sin interrupción. Hay dos tipos de escalabilidad:

1. **Escalabilidad horizontal: se** refiere a expandir el número de nodos en el Sistema distribuido. A medida que se agregan nodos al sistema, debería ser posible para distribuir algunos de los datos y cargas de procesamiento de los nodos existentes a los nuevos nodos.
2. **Escalabilidad vertical: se** refiere a expandir la capacidad del individuo. nodos en el sistema, como expandir la capacidad de almacenamiento o el proceso ing poder de un nodo.

A medida que el sistema expande su número de nodos, es posible que la red, que conecta los nodos, puede tener fallas que hacen que los nodos se dividan en grupos de nodos. Los nodos dentro de cada partición todavía están conectados por una subred funciona, pero se pierde la comunicación entre las particiones. El concepto de **partición La tolerancia** establece que el sistema debe tener la capacidad de seguir funcionando. mientras la red está particionada.

23.1.5 Autonomía

La autonomía determina la medida en que los nodos o bases de datos individuales en un DDB puede funcionar de forma independiente. Es deseable un alto grado de autonomía para mayor flexibilidad y mantenimiento personalizado de un nodo individual. Autonomía se puede aplicar al diseño, la comunicación y la ejecución. **La autonomía de diseño se** refiere a la independencia del uso del modelo de datos y las técnicas de gestión de transacciones entre nodos. **La autonomía de la comunicación** determina el grado en que cada El nodo puede decidir compartir información con otros nodos. **Autonomía de ejecución** se refiere a la independencia de los usuarios para actuar como les plazca.

Página 6

23.1.6 Ventajas de las bases de datos distribuidas

Algunas ventajas importantes de DDB se enumeran a continuación.

1. **Mejora de la facilidad y flexibilidad del desarrollo de aplicaciones** . Desarrollando y mantener aplicaciones en sitios distribuidos geográficamente de una La organización se ve facilitada debido a la transparencia de la distribución de datos y controlar.
2. **Mayor disponibilidad.** Esto se logra mediante el aislamiento de fallas en su sitio de origen sin afectar a los otros nodos de la base de datos conectados al red. Cuando los datos y el software DDBMS se distribuyen en muchos sitios, un sitio puede fallar mientras otros sitios continúan funcionando. Solo los datos y no se puede acceder al software que existe en el sitio fallido. Más lejos La mejora se logra replicando con criterio los datos y el software en más de un sitio. En un sistema centralizado, la falla en un solo sitio hace todo el sistema no está disponible para todos los usuarios. En una base de datos distribuida, algunos Es posible que no se pueda acceder a los datos, pero los usuarios aún pueden acceder a otros partes de la base de datos. Si los datos del sitio fallido se han replicado en otro sitio antes de la falla, el usuario no se verá afectado en absoluto. los La capacidad del sistema para sobrevivir a la partición de la red también contribuye a alta disponibilidad.
3. **Rendimiento mejorado.** Un DBMS distribuido fragmenta la base de datos manteniendo los datos más cerca de donde más se necesitan. **Localización de datos** reduce la contienda por los servicios de CPU y E / S y simultáneamente reduce las demoras de acceso involucradas en redes de área extensa. Cuando un gran

La base de datos se distribuye en varios sitios, existen bases de datos más pequeñas en cada sitio. Como resultado, las consultas y transacciones locales que acceden a datos en un un solo sitio tiene un mejor rendimiento debido a la menor cantidad de datos locales bases. Además, cada sitio tiene un número menor de transacciones ejecutadas recortando que si todas las transacciones se envían a una única centralizada base de datos. Además, el paralelismo entre consultas e intraconsultas se puede logrado mediante la ejecución de múltiples consultas en diferentes sitios, o rompiendo convertir una consulta en varias subconsultas que se ejecutan en paralelo. Esta contribuye a mejorar el rendimiento.

4. **Expansión más sencilla gracias a la escalabilidad** . En un entorno distribuido, expansión del sistema en términos de agregar más datos, aumentar el tamaño de la base de datos o agregar más nodos es mucho más fácil que en centralizado (no distribuido) sistemas.

Las transparencias que discutimos en la Sección 23.1.2 conducen a un compromiso entre facilidad de uso y los gastos generales de transparencia. Transparencia total proporciona al usuario global una vista de todo el DDBS como si fuera una única central sistema. La transparencia se brinda como complemento a la **autonomía** , lo que otorga al los usuarios tienen un control más estricto sobre las bases de datos locales. Se pueden implementar características de transparencia mentado como parte del idioma del usuario, que puede traducir los servicios requeridos en las operaciones apropiadas.

Página 7

23.2 Fragmentación de datos, replicación, y técnicas de asignación para distribuidos Diseño de base de datos

En esta sección, discutimos las técnicas que se utilizan para dividir la base de datos en unidades lógicas, llamadas **fragmentos** , que pueden asignarse para su almacenamiento en los distintos nodos. También discutimos el uso de **la replicación** de **datos** , que permite que ciertos datos sean almacenados en más de un sitio para aumentar la disponibilidad y confiabilidad; y el proceso de **asignar** fragmentos, o réplicas de fragmentos, para su almacenamiento en los distintos nodos. Estas técnicas se utilizan durante el proceso de **diseño de bases** de **datos distribuidas** . los La información relativa a la fragmentación, asignación y replicación de datos se almacena en un **directorio global al** que acceden las aplicaciones DDBS según sea necesario.

23.2.1 Fragmentación y fragmentación de datos

En una DDB, se deben tomar decisiones con respecto a qué sitio se debe usar para almacenar qué partes de la base de datos. Por ahora, asumiremos que no hay replicación; es decir, cada relación, o parte de una relación, se almacena en un solo sitio. Dis- maldice la replicación y sus efectos más adelante en esta sección. También usamos la terminología de bases de datos relacionales, pero conceptos similares se aplican a otros modelos de datos. Asumimos

que estamos comenzando con un esquema de base de datos relacional y debemos decidir cómo distribuir las relaciones en los distintos sitios. Para ilustrar nuestra discusión, usamos el esquema de la base de datos relacional que se muestra en la Figura 5.5.

Antes de decidir cómo distribuir los datos, debemos determinar las unidades lógicas de la base de datos que se distribuirá. Las unidades lógicas más simples son las relaciones sí mismos; es decir, cada relación completa debe almacenarse en un sitio en particular. En nuestro examen Por ejemplo, debemos decidir un sitio para almacenar cada una de las relaciones EMPLEADO , DEPARTAMENTO , PROYECTO , TRABAJA_ON y DEPENDIENTE en la Figura 5.5. En muchos casos, sin embargo, un La relación se puede dividir en unidades lógicas más pequeñas para su distribución. Por ejemplo, Considere la base de datos de la empresa que se muestra en la Figura 5.6, y suponga que hay tres sitios de computadoras, uno para cada departamento de la empresa. [1]

Es posible que deseemos almacenar la información de la base de datos relacionada con cada departamento en el sitio de computadora para ese departamento. Una técnica llamada fragmentación horizontal o La fragmentación se puede utilizar para dividir cada relación por departamento.

Fragmentación horizontal (fragmentación). Un **fragmento horizontal** o **fragmento** de un relación es un subconjunto de las tuplas en esa relación. Las tuplas que pertenecen a la horizontal El fragmento se puede especificar mediante una condición en uno o más atributos de la relación, o por algún otro mecanismo. A menudo, solo un atributo está involucrado en la condición. Por ejemplo, podemos definir tres fragmentos horizontales en la relación EMPLEADO en Figura 5.6 con las siguientes condiciones: (Dno = 5), (Dno = 4) y (Dno = 1) —cada una

[1] Por supuesto, en una situación real, habrá muchas más tuplas en la relación que las que se muestran en Figura 5.6.

Página 8

El fragmento contiene las tuplas EMPLEADO que trabajan para un departamento en particular. Sim- Análogamente, podemos definir tres fragmentos horizontales para la relación PROYECTO , con el condiciones (Dnum = 5), (Dnum = 4) y (Dnum = 1) - cada fragmento contiene el Tuplas de PROYECTO controladas por un departamento en particular. **Fragmentación horizontal** divide una relación horizontalmente agrupando filas para crear subconjuntos de tuplas, donde cada subconjunto tiene un cierto significado lógico. Estos fragmentos se pueden asignar a diferentes sitios (nodos) en el sistema distribuido. **Fragmentación horizontal derivada** aplica la partición de una relación primaria (DEPARTAMENTO en nuestro ejemplo) a otras relaciones secundarias (EMPLEADO y PROYECTO en nuestro ejemplo), que son relacionado con el primario a través de una clave externa. Por tanto, los datos relacionados entre el primario y el las relaciones secundarias se fragmentan de la misma manera.

Fragmentación vertical. Es posible que cada sitio no necesite todos los atributos de una relación, lo que indicaría la necesidad de un tipo diferente de fragmentación. **Vertical la fragmentación** divide una relación "verticalmente" por columnas. Un **fragmento vertical** de un la relación conserva sólo ciertos atributos de la relación. Por ejemplo, es posible que queramos fragmentar la relación EMPLEADO en dos fragmentos verticales. El primer fragmento incluye información personal (nombre , fecha de nacimiento , dirección y sexo) y la segunda

incluye información relacionada con el trabajo: Ssn , Salario , Super_ssn y Dno . Esta vertical fragmentación no es del todo adecuada, porque si los dos fragmentos se almacenan por separado raramente, no podemos volver a juntar las tuplas de empleados originales ya que no hay atributo común entre los dos fragmentos. Es necesario incluir el primario clave o algún atributo clave único en cada fragmento vertical para que la relación completa se puede reconstruir a partir de los fragmentos. Por lo tanto, debemos agregar el atributo Ssn a el fragmento de información personal.

Observe que cada fragmento horizontal de una relación R se puede especificar en la relación álgebra nacional por una σ (R) (seleccionar) operación. Un conjunto de fragmentos horizontales cuyo las condiciones C 1 , C 2 ,..., C n incluyen todas las tuplas en R, es decir, cada tupla en R satisfies (C 1 O C 2 O ... O C n): se denomina **fragmentación horizontal completa** de R. En muchos casos, una fragmentación horizontal completa también es **inconexa** ; es decir, sin tupla en R satisface (C i Y C j) para cualquier i ≠ j. Nuestros dos ejemplos anteriores de fragmentación horizontal mentalidad para el EMPLEADO y el PROYECTO las relaciones fueron completas y articulación. Para reconstruir la relación R a partir de una completa fragmentación horizontal, es necesario aplicar la operación UNION a los fragmentos.

Un fragmento vertical en una relación R se puede especificar mediante un π (R) operación en el álgebra relacional. Un conjunto de fragmentos verticales cuya proyección enumera L 1 , L 2 ,..., L n incluir todos los atributos en R pero compartir solo el atributo de clave principal de R se llama una **completa fragmentación vertical** de R. En este caso, las listas de proyección satisfacen el siguientes dos condiciones:

- L 1 ∪ L 2 ∪... ∪ L n = ATTRS (R)
- L i ∩ L j = PK (R) para cualquier i ≠ j, donde ATTRS (R) es el conjunto de atributos de R y PK (R) es la clave principal de R

Para reconstruir la relación R a partir de una completa fragmentación vertical, aplicamos la operación UNIÓN EXTERIOR a los fragmentos verticales (asumiendo que no hay horizontal

Página 9

se utiliza fragmentación). Observe que también podríamos aplicar una operación FULL OUTER JOIN y obtener el mismo resultado para una completa fragmentación vertical, incluso cuando también puede haberse aplicado cierta fragmentación horizontal. Los dos fragmentos verticales de la relación EMPLEADO con listas de proyección L 1 = { Ssn , Name , Bdate , Address , Sex } y L 2 = { Ssn , Salary , Super_ssn , Dno } constituyen una vertical completa fragmentación del EMPLEADO .

Dos fragmentos horizontales que no son completos ni disjuntos son los definidos en la relación EMPLEADO en la Figura 5.5 por las condiciones (Salario > 50000) y (Dno = 4); pueden no incluir todas las tuplas EMPLEADOS y pueden incluir tuplas comunes. Dos fragmentos verticales que no están completos son los definidos por las listas de atributos L 1 = { Nombre , Dirección } y L 2 = { Ssn , Nombre , Salario }; estas listas violan ambas condiciones de una completa fragmentación vertical.

Fragmentación mixta (híbrida). Podemos entremezclar los dos tipos de fragmentaciones
ción, produciendo una **fragmentación mixta** . Por ejemplo, podemos combinar el horizonte
fragmentaciones verticales y verticales de la relación EMPLEADO dada anteriormente en un
fragmentación que incluye seis fragmentos. En este caso, la relación original puede ser
reconstruido aplicando operaciones UNION y OUTER UNION (o OUTER JOIN)
en el orden apropiado. En general, un **fragmento** de una relación R se puede especificar mediante un
Combinación de operaciones SELECT-PROJECT $\pi_L(\sigma_C(R))$. Si C = VERDADERO (es decir, todos
tuplas) y L ≠ ATTRS (R), obtenemos un fragmento vertical, y si C ≠ TRUE y
L = ATTRS (R), obtenemos un fragmento horizontal. Finalmente, si C ≠ TRUE y L ≠ ATTRS (R),
obtenemos un fragmento mixto. Observe que una relación en sí misma puede considerarse un fragmento
con C = VERDADERO y L = ATTRS (R). En la siguiente discusión, el término fragmento es
se utiliza para referirse a una relación oa cualquiera de los tipos de fragmentos anteriores.

Un **esquema** de **fragmentación** de una base de datos es una definición de un conjunto de fragmentos que incluye
todos los atributos y tuplas de la base de datos y satisface la condición de que todos los datos
la base se puede reconstruir a partir de los fragmentos aplicando alguna secuencia de EXTERIOR
Operaciones UNION (o OUTER JOIN) y UNION . A veces también es útil, aunque
no es necesario: hacer que todos los fragmentos sean disjuntos excepto por la repetición de
mary keys entre fragmentos verticales (o mixtos). En el último caso, toda la replicación
y la distribución de fragmentos se especifica claramente en una etapa posterior, por separado
de la fragmentación.

Un **esquema de asignación** describe la asignación de fragmentos a nodos (sitios) del
DDBS; por lo tanto, es un mapeo que especifica para cada fragmento el sitio (s) en el que
está almacenado. Si un fragmento se almacena en más de un sitio, se dice que está **replicado** . Nosotros
discutir la replicación y asignación de datos a continuación.

23.2.2 Replicación y asignación de datos

La replicación es útil para mejorar la disponibilidad de datos. El caso más extremo es
replicación de toda la base de datos en cada sitio del sistema distribuido, creando así
ing una **base de datos distribuida completamente replicada** . Esto puede mejorar el comentario de disponibilidad.
hábilmente porque el sistema puede continuar funcionando mientras al menos un sitio esté activo. Eso

Página 10

también mejora el rendimiento de la recuperación (rendimiento de lectura) para consultas globales
porque los resultados de tales consultas se pueden obtener localmente desde cualquier sitio; por lo tanto,
una consulta de recuperación se puede procesar en el sitio local donde se envía, si ese sitio
incluye un módulo de servidor. La desventaja de la replicación completa es que puede ralentizar
las operaciones de actualización (rendimiento de escritura) drásticamente, ya que una única lógica
La actualización debe realizarse en cada copia de la base de datos para mantener la coherencia de las copias.
tienda. Esto es especialmente cierto si existen muchas copias de la base de datos. Replicación completa
hace que las técnicas de recuperación y control de concurrencia sean más caras de lo que
sería si no hubiera replicación, como veremos en la Sección 23.3.

El otro extremo de la replicación completa implica **no** tener **replicación** , es decir,

cada fragmento se almacena en exactamente un sitio. En este caso, todos los fragmentos deben articulación, excepto por la repetición de claves primarias entre fragmentos verticales (o mixtos) mentos. Esto también se denomina **asignación no redundante** .

Entre estos dos extremos, tenemos un amplio espectro de **replicación parcial** de los datos, es decir, algunos fragmentos de la base de datos se pueden replicar mientras que otros Los ers no pueden. El número de copias de cada fragmento puede variar desde una hasta el número total de sitios en el sistema distribuido. Un caso especial de réplica parcial cación se está produciendo con gran frecuencia en aplicaciones donde los trabajadores móviles, como ventas fuerzas, planificadores financieros y ajustadores de reclamaciones, llevan datos parcialmente replicados bases con ellos en portátiles y PDA y sincronizarlos periódicamente con la base de datos del servidor. Una descripción de la replicación de fragmentos a veces es llamado **esquema de replicación** .

Cada fragmento, o cada copia de un fragmento, debe asignarse a un sitio en particular en el sistema distribuido. Este proceso se denomina **distribución de datos** (o **asignación de datos**). La elección de los sitios y el grado de replicación dependen del rendimiento y objetivos de disponibilidad del sistema y sobre los tipos y frecuencias de las transacciones enviado en cada sitio. Por ejemplo, si se requiere alta disponibilidad, las transacciones pueden enviarse en cualquier sitio, y la mayoría de las transacciones son solo de recuperación, una copia La base de datos es una buena opción. Sin embargo, si determinadas transacciones que acceden a partes de la base de datos se envían principalmente a un sitio en particular, el conjunto correspondiente de fragmentos se pueden asignar en ese sitio solamente. Datos a los que se accede en varios sitios puede replicarse en esos sitios. Si se realizan muchas actualizaciones, puede resultar útil limitar la replicación. Encontrar una solución óptima o incluso buena para los datos distribuidos La asignación es un problema de optimización complejo.

23.2.3 Ejemplo de fragmentación, asignación y replicación

Consideremos ahora un ejemplo de fragmentación y distribución de los datos de la empresa: base en las Figuras 5.5 y 5.6. Suponga que la empresa tiene tres sitios de computadoras: uno para cada departamento actual. Los sitios 2 y 3 son para los departamentos 5 y 4, respectivamente. En cada uno de estos sitios, esperamos un acceso frecuente al EMPLEADO y Información del PROYECTO para los empleados que laboran en ese departamento y el proyectos controlados por ese departamento. Además, suponemos que estos sitios principalmente acceda a los atributos Nombre , Ssn , Salario y Super_ssn de EMPLOYEE . Se utiliza el sitio 1

Página 11

por la sede de la empresa y accede a toda la información de los empleados y del proyecto generalmente, además de realizar un seguimiento de la información de DEPENDIENTES para fines de seguros.

De acuerdo con estos requisitos, toda la base de datos de la Figura 5.6 se puede almacenar en sitio 1. Para determinar los fragmentos que se replicarán en los sitios 2 y 3, primero podemos fragmentar horizontalmente DEPARTAMENTO por su clave Dnumber . Luego aplicamos derivados fragmentación de las relaciones EMPLOYEE , PROJECT y DEPT_LOCATIONS basadas

en sus claves externas para el número de departamento, llamado Dno , Dnum y Dnumber , respectivamente, en la Figura 5.5. Podemos fragmentar verticalmente el EMPLEADO resultante fragmentos para incluir solo los atributos { Nombre , Ssn , Salario , Super_ssn , Dno }. La figura 23.2 muestra los fragmentos mixtos EMPD_5 y EMPD_4 , que incluyen el

Tuplas EMPLEADOS que satisfacen las condiciones Dno = 5 y Dno = 4, respectivamente. los Los fragmentos horizontales de PROJECT , DEPARTMENT y DEPT_LOCATIONS son igualmente fragmentado por número de departamento. Todos estos fragmentos, almacenados en sitios 2 y 3: se replican porque también se almacenan en la sede, sitio 1.

Ahora debemos fragmentar la relación WORKS_ON y decidir qué fragmentos de

WORKS_ON para almacenar en los sitios 2 y 3. Nos enfrentamos al problema de que no atributo de WORKS_ON indica directamente el departamento al que cada tupla pertenece. De hecho, cada tupla en WORKS_ON relaciona a un empleado e con un proyecto P. Nosotros podría fragmentar WORKS_ON en función del departamento D en el que e trabaja o con base en el departamento D ′ que controla P. La fragmentación se vuelve fácil si tenemos un restricción que indica que D = D ′ para todas las tuplas WORKS_ON , es decir, si los empleados pueden trabajar solo en proyectos controlados por el departamento para el que trabajan. Sin embargo, hay no existe tal restricción en nuestra base de datos en la Figura 5.6. Por ejemplo, WORKS_ON tupla <333445555, 10, 10.0> relaciona un empleado que trabaja para el departamento 5 con un proyecto controlado por el departamento 4. En este caso, podríamos fragmentar WORKS_ON basado en el departamento en el que trabaja el empleado (que se expresa por el condición C) y luego fragmentar aún más en función del departamento que controla el proyectos en los que el empleado está trabajando, como se muestra en la Figura 23.3.

En la figura 23.3, la unión de los fragmentos G_1 , G_2 y G_3 da todas las tuplas WORKS_ON para los empleados que trabajan para el departamento 5. Asimismo, la unión de los fragmentos G_4 , G_5 , y G_6 proporciona todas las tuplas WORKS_ON para los empleados que trabajan para el departamento 4. En por otro lado, la unión de los fragmentos G_1 , G_4 y G_7 da todas las tuplas WORKS_ON para proyectos controlados por el departamento 5. La condición de cada uno de los fragmentos G_1 a G_9 se muestra en la Figura 23.3. Las relaciones que representan relaciones M: N, como WORKS_ON , suelen tener varias posibles fragmentaciones lógicas. En nuestra distri- En la figura 23.2, elegimos incluir todos los fragmentos que se pueden unir a un

Tupla EMPLEADO o una tupla PROYECTO en los sitios 2 y 3. Por lo tanto, colocamos la unión de fragmentos G_1 , G_2 , G_3 , G_4 y G_7 en el sitio 2 y la unión de los fragmentos G_4 , G_5 , G_6 , G_2 , y G_8 en el sitio 3. Observe que los fragmentos G_2 y G_4 se replican en ambos sitios. Este alo- La estrategia catiónica permite la unión entre los fragmentos de EMPLEADO o PROYECTO local en el sitio 2 o el sitio 3 y el fragmento WORKS_ON local se realizará de forma completamente local. Esto demuestra claramente cuán complejo es el problema de la fragmentación de la base de datos y La asignación es para grandes bases de datos. La bibliografía seleccionada al final de este capítulo analiza algunos de los trabajos realizados en esta área.

Pagina 12

(un) EMPD_5

Fname	Minit	Lname	Ssn	Salario	Super_ssn	Dno

Figura 23.2
Asignación de fragmentos a

Juan	segundo	Herrero	123456789	30000	333445555	5
Franklin	T	Wong	333445555	40000	888665555	5
Ramesh	K	Narayan	666884444	38000	333445555	5
Joyce	UN	Inglés	453453453	25000	333445555	5

sitios. (a) Fragmentos de relación en el sitio 2 correspondiente a departamento 5. (b) Relación fragmentos en el sitio 3 correspondiente a departamento 4.

DEP_5

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Investigación	5	333445555	1988-05-22

DEP_5_LOCS

Dnumber	Ubicación
5	Bellaire
5	La tierra del azúcar
5	Houston

WORKS_ON_5

Essn	Pno	Horas
123456789	1	32,5
123456789	2	7.5
666884444	3	40,0
453453453	1	20,0
453453453	2	20,0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0

PROJS_5

Pname	Pnumber	Ubicación	Dnum
Producto X	1	Bellaire	5
Producto Y	2	La tierra del azúcar	5
Producto Z	3	Houston	5

Datos en el sitio 2

(segundo)

EMPD_4

Fname	Minit	Lname	Ssn	Salario	Super_ssn	Dno
Alicia	J	Zelaya	999887777	25000	987654321	4
Jennifer	S	Wallace	987654321	43000	888665555	4
Ahmad	V	Jabbar	987987987	25000	987654321	4

DEP_4

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Administración	4	987654321	1995-01-01

DEP_4_LOCS

Dnumber	Ubicación
4	Stafford

WORKS_ON_4

Essn	Pno	Horas
333445555	10	10.0
999887777	30	30,0
999887777	10	10.0
987987987	10	35,0
987987987	30	5,0
987654321	30	20,0
987654321	20	15.0

PROJS_4

Pname	Pnumber	Ubicación	Dnum
Informatización	10	Stafford	4
Nuevos_beneficios	30	Stafford	4

Datos en el sitio 3

Página 13

Figura 23.3
Fragmentos completos y disjuntos de la relación WORKS_ON. (a) Fragmentos de WORKS_ON para empleados trabajando en el departamento 5 (C = [Essn en (SELECT Ssn FROM EMPLOYEE WHERE Dno = 5)]). (b) Fragmentos de WORKS_ON para los empleados que trabajan en el departamento 4 (C = [Essn in (SELECT Ssn FROM EMPLOYEE WHERE Dno = 4)]). (c) Fragmentos de WORKS_ON para empleados que trabajan en el departamento 1 (C = [Essn in (SELECT Ssn DEL EMPLEADO DONDE Dno = 1)]).

(un) Empleados del Departamento 5

G1

Essn	Horas pno	
123456789	1	32,5
123456789	2	7.5
666884444	3	40,0
453453453	1	20,0
453453453	2	20,0
333445555	2	10.0
333445555	3	10.0

1C = C y (Pno en (SELECT Pnumber DEL PROYECTO DONDE Dnum = 5))

G2

Essn	Horas pno	
333445555	10	10.0

C2 = C y (Pno en (SELECT Pnumber DEL PROYECTO DONDE Dnum = 4))

G3

Essn	Horas pno	
333445555	20	10.0

C3 = C y (Pno en (SELECT Pnumber DEL PROYECTO DONDE Dnum = 1))

(segundo) Empleados del Departamento 4

G4

Essn	Pno	Horas

C4 = C y (Pno en (SELECT Pnumber DEL PROYECTO DONDE Dnum = 5))

G5

Essn	Horas pno	
999887777	30	30,0
999887777	10	10.0
987987987	10	35,0
987987987	30	5,0
987654321	30	20,0

C5 = C y (Pno en (SELECT Pnumber DEL PROYECTO DONDE Dnum = 4))

G6

Essn	Pno	Horas
987654321	20	15.0

C6 = C y (Pno en (SELECT Pnumber DEL PROYECTO DONDE Dnum = 1))

(C) Empleados del Departamento 1

G7

Essn	Horas pno

C7 = C y (Pno en (SELECT Pnumber DEL PROYECTO DONDE Dnum = 5))

G8

Essn	Horas pno

C8 = C y (Pno en (SELECT Pnumber DEL PROYECTO DONDE Dnum = 4))

G9

Essn	Pno	Horas
888665555	20	Nulo

C9 = C y (Pno en (SELECT Pnumber DEL PROYECTO DONDE Dnum = 1))

23.3 Descripción general del control de concurrencia y recuperación en bases de datos distribuidas

Para fines de control de concurrencia y recuperación, surgen numerosos problemas en una entorno DBMS tributado que no se encuentran en un entorno DBMS centralizado ambiente. Estos incluyen los siguientes:

- **Manejo de múltiples copias de los elementos de datos.** El control de concurrencia El método es responsable de mantener la coherencia entre estas copias. los método de recuperación es responsable de hacer una copia consistente con otras copias Es si el sitio en el que se almacena la copia falla y se recupera más tarde.
- **Fracaso de sitios individuales.** El DDBMS debe seguir funcionando con su ejecutar sitios, si es posible, cuando uno o más sitios individuales fallan. Cuando un sitio se recupera, su base de datos local debe estar actualizada con el resto de sitios antes de que vuelva a unirse al sistema.
- **Fallo de enlaces de comunicación.** El sistema debe poder hacer frente a falla de uno o más de los enlaces de comunicación que conectan los sitios. Un El caso extremo de este problema es que puede producirse una **partición de red** . Esta divide los sitios en dos o más particiones, donde los sitios dentro de cada La partición solo puede comunicarse entre sí y no con sitios en otros particiones.
- **Confirmación distribuida.** Pueden surgir problemas al realizar una transacción que accede a bases de datos almacenadas en varios sitios si algunos sitios fallan durante el proceso de compromiso. El **protocolo de compromiso de dos fases** (consulte la Sección 21.6) es a menudo se utiliza para hacer frente a este problema.
- **Bloqueo distribuido.** El interbloqueo puede ocurrir entre varios sitios, por lo que Las técnicas para hacer frente a los interbloqueos deben ampliarse para tomar este en cuenta.

Las técnicas de recuperación y control de simultaneidad distribuidas deben abordar estos y otros problemas. En las siguientes subsecciones, revisamos algunas de las niques que se han sugerido para tratar la recuperación y el control de concurrencia en DDBMS.

23.3.1 Basado en control de simultaneidad distribuido en una copia distinguida de un elemento de datos

Para lidiar con elementos de datos replicados en una base de datos distribuida, Se han propuesto métodos de control de concurrencia que amplían el control de concurrencia técnicas que se utilizan en bases de datos centralizadas. Discutimos estas técnicas en el contexto de la ampliación del cierre centralizado. Se aplican extensiones similares a otros técnicas de control de divisas. La idea es designar una copia particular de cada dato artículo como **copia distinguida** . Los bloqueos para este elemento de datos están asociados con el copia distinguida, y todas las solicitudes de bloqueo y desbloqueo se envían al sitio que contiene esa copia.

Varios métodos diferentes se basan en esta idea, pero difieren en su método de elección de las copias distinguidas. En la **técnica del sitio primario** , todos copias distinguidas se guardan en el mismo sitio. Una modificación de este enfoque es el sitio principal con un **sitio de respaldo** . Otro enfoque es la **copia principal** método, donde las copias distinguidas de los diversos elementos de datos se pueden almacenar en diferentes sitios. Un sitio que incluye una copia distinguida de un elemento de datos básicamente actúa como el **sitio coordinador** para el control de concurrencia en ese elemento. Discutimos estos técnicas a continuación.

Técnica del sitio principal. En este método, se designa un único **sitio primario** para ser el **sitio** del **coordinador** para todos los elementos de la base de datos. Por lo tanto, todas las cerraduras se mantienen en ese sitio y todas las solicitudes de bloqueo o desbloqueo se envían allí. Este método es, por tanto, una extensión sión del enfoque de bloqueo centralizado. Por ejemplo, si todas las transacciones siguen el Protocolo de bloqueo de dos fases, la serialización está garantizada. La ventaja de esto enfoque es que es una simple extensión del enfoque centralizado y por lo tanto no es demasiado complejo. Sin embargo, tiene ciertas desventajas inherentes. Uno es que todo Las solicitudes de bloqueo se envían a un único sitio, posiblemente sobrecargando ese sitio y provocando un cuello de botella del sistema. Una segunda desventaja es que la falla del parámetro del sitio primario liza el sistema, ya que toda la información de bloqueo se guarda en ese sitio. Esto puede limitar confiabilidad y disponibilidad del sistema.

Aunque se accede a todas las cerraduras en el sitio principal, se puede acceder a los elementos en sí en cualquier sitio en el que residan. Por ejemplo, una vez que una transacción obtiene un Read_lock en un elemento de datos del sitio principal, puede acceder a cualquier copia de ese elemento de datos. Sin embargo, una vez que una transacción obtiene un Write_lock y actualiza un elemento de datos, el DDBMS responde Es posible actualizar todas las copias del elemento de datos antes de liberar el bloqueo.

Sitio principal con sitio de respaldo. Este enfoque aborda la segunda desventaja tage del método del sitio principal mediante la designación de un segundo sitio como un **sitio de respaldo** . Todos La información de bloqueo se mantiene tanto en el sitio principal como en el de respaldo. En caso de falla del sitio primario, el sitio de respaldo se convierte en el sitio primario, y un nuevo Se elige el sitio de respaldo. Esto simplifica el proceso de recuperación de la falla del sitio principal, ya que el sitio de respaldo se hace cargo y el procesamiento puede reanudarse después de un nuevo Se elige el sitio de respaldo y la información del estado del bloqueo se copia en ese sitio. Se ralentiza el proceso de adquisición de bloqueos, sin embargo, porque todas las solicitudes de bloqueo y Los bloqueos deben registrarse tanto en el sitio principal como en el de respaldo antes de La respuesta se envía a la transacción solicitante. El problema de la primaria y los sitios de respaldo se sobrecargan con solicitudes y ralentizan el sistema permanece inalterado.

Técnica de copia primaria. Este método intenta distribuir la carga de la cerradura coordinación entre varios sitios al tener las copias distinguidas de diferentes elementos de datos almacenados en diferentes sitios. La falla de un sitio afecta cualquier transacción que están accediendo a bloqueos en elementos cuyas copias principales residen en ese sitio, pero otros las transacciones no se ven afectadas. Este método también puede usar sitios de respaldo para mejorar la confianza capacidad y disponibilidad.

Elección de un nuevo sitio para el coordinador en caso de falla. Siempre que una coordina-
Si el sitio falla en cualquiera de las técnicas anteriores, los sitios que aún se están ejecutando deben
elija un nuevo coordinador. En el caso del enfoque de sitio primario sin respaldo
sitio, todas las transacciones en ejecución deben abortarse y reiniciarse en una tediosa recuperación
proceso. Parte del proceso de recuperación implica elegir un nuevo sitio primario y crear
realizar un proceso de administrador de bloqueo y un registro de toda la información de bloqueo en ese sitio. por
métodos que utilizan sitios de respaldo, el procesamiento de transacciones se suspende mientras
El sitio de respaldo se designa como el nuevo sitio principal y se elige un nuevo sitio de respaldo.
y se envían copias de toda la información de bloqueo del nuevo sitio principal.

Si un sitio de respaldo X está a punto de convertirse en el nuevo sitio principal, X puede elegir el nuevo
sitio de respaldo de entre los sitios en ejecución del sistema. Sin embargo, si no hay un sitio de respaldo
existía, o si tanto el sitio principal como el de respaldo están inactivos, un proceso llamado
La elección se puede utilizar para elegir el nuevo sitio del coordinador. En este proceso, cualquier sitio Y
que intenta comunicarse con el sitio del coordinador repetidamente y no lo hace
puede asumir que el coordinador está caído y puede iniciar el proceso de elección enviando
enviando un mensaje a todos los sitios en ejecución proponiendo que Y se convierta en el nuevo coordinador. Como
tan pronto como Y reciba una mayoría de votos a favor, Y puede declarar que es la nueva coordina-
colina. El algoritmo electoral en sí es complejo, pero esta es la idea principal detrás del
método de elección. El algoritmo también resuelve cualquier intento de dos o más sitios de
convertirse en coordinador al mismo tiempo. Las referencias en la bibliografía seleccionada
al final de este capítulo, analice el proceso en detalle.

23.3.2 Control de concurrencia distribuido basado en votación

Los métodos de control de concurrencia para elementos replicados discutidos anteriormente utilizan
idea de una copia distinguida que mantiene las cerraduras de ese artículo. En la **votacion**
método , no hay copia distinguida; más bien, se envía una solicitud de bloqueo a todos los sitios que
incluye una copia del elemento de datos. Cada copia mantiene su propio bloqueo y puede otorgar o
denegar la solicitud. Si una transacción que solicita un bloqueo recibe ese bloqueo mediante un
mayoría de las copias, mantiene el candado e informa a todas las copias que ha sido
concedido el candado. Si una transacción no recibe la mayoría de votos otorgándole una
bloquear dentro de un cierto período de tiempo de espera, cancela su solicitud e informa a todos los sitios de
la cancelación.

El método de votación se considera un método de control de concurrencia verdaderamente distribuido,
ya que la responsabilidad de una decisión recae en todos los sitios involucrados. Simulación
Los estudios han demostrado que la votación tiene un mayor tráfico de mensajes entre los sitios que el
métodos de copia distinguidos. Si el algoritmo tiene en cuenta la posible falla del sitio
Durante el proceso de votación, se vuelve extremadamente complejo.

23.3.3 Recuperación distribuida

El proceso de recuperación en bases de datos distribuidas es bastante complicado. Damos solo un muy
breve idea de algunos de los problemas aquí. En algunos casos es difícil incluso determinar
si un sitio está caído sin intercambiar numerosos mensajes con otros sitios. por

Página 17

Por ejemplo, suponga que el sitio X envía un mensaje al sitio Y y espera una respuesta de Y pero no lo recibe. Hay varias explicaciones posibles:

- El mensaje no se entregó a Y debido a un error de comunicación.
- El sitio Y está inactivo y no pudo responder.
- El sitio Y se está ejecutando y envió una respuesta, pero la respuesta no se entregó.

Sin información adicional o el envío de mensajes adicionales, es difícil para determinar qué sucedió realmente.

Otro problema con la recuperación distribuida es el compromiso distribuido. Cuando una transacción está actualizando datos en varios sitios, no puede confirmar hasta que esté seguro de que el efecto de la transacción en cada sitio no se puede perder. Esto significa que cada sitio debe primero haber registrado los efectos locales de las transacciones de forma permanente en el registro del sitio local en disco. El protocolo de compromiso de dos fases se utiliza a menudo para garantizar la corrección de confirmación distribuida (consulte la Sección 21.6).

23.4 Descripción general de la gestión de transacciones en bases de datos distribuidas

Los módulos de software de gestión de transacciones globales y locales, junto con control de concurrencia y administrador de recuperación de un DDBMS, garantizan colectivamente la Propiedades ACID de las transacciones (ver Capítulo 20).

Se introduce un componente adicional llamado **administrador de transacciones global** para apoyando transacciones distribuidas. El sitio donde se originó la transacción puede asumir temporalmente el rol de gerente de transacciones globales y coordinar el Corrección de las operaciones de la base de datos con los administradores de transacciones en varios sitios. Los administradores de transacciones exportan su funcionalidad como una interfaz a la aplicación programas. Las operaciones exportadas por esta interfaz son similares a las cubiertas en Sección 20.2.1, a saber, BEGIN_TRANSACTION , READ o WRITE , END_TRANSACTION , COMMIT_TRANSACTION y ROLLBACK (o ABORT). El gerente almacena libros mantener información relacionada con cada transacción, como un identificador único, nating sitio, nombre, etc. Para operaciones READ , devuelve una copia local si es válida y disponible. Para las operaciones de ESCRITURA , asegura que las actualizaciones sean visibles en todos los sitios que contiene copias (réplicas) del elemento de datos. Para operaciones ABORT , el gerente asegura que ningún efecto de la transacción se refleje en ningún sitio de la distribución base de datos. Para las operaciones COMMIT , asegura que los efectos de una escritura sean persistentes registrado en todas las bases de datos que contienen copias del elemento de datos. Terminación atómica (COMMIT / ABORT) de las transacciones distribuidas se implementa comúnmente utilizando el protocolo de compromiso de dos fases (consulte la Sección 22.6).

El administrador de transacciones pasa al módulo controlador de concurrencia la base de datos operaciones e información asociada. El controlador es responsable de la adquisición y liberación de bloqueos asociados. Si la transacción requiere acceso a un recurso, se bloquea hasta que se adquiere el bloqueo. Una vez que se adquiere la cerradura, el operador se envía al procesador en tiempo de ejecución, que maneja la ejecución real de la

Página 18

operación de la base de datos. Una vez finalizada la operación, se liberan los bloqueos y
El administrador de transacciones se actualiza con el resultado de la operación.

23.4.1 Protocolo de compromiso de dos fases

En la Sección 22.6, describimos el protocolo de compromiso de dos fases (**2PC**), que requiere un **gerente de recuperación global** , o **coordinador** , para mantener la información necesaria para recuperación, además de los gerentes de recuperación locales y la información que mantener (registro, tablas). El protocolo de compromiso de dos fases tiene ciertos inconvenientes que condujo al desarrollo del protocolo de compromiso de tres fases, que discutiremos a continuación.

23.4.2 Protocolo de compromiso trifásico

El mayor inconveniente de 2PC es que es un protocolo de bloqueo. Fallo de la coordinación nator bloquea todos los sitios participantes, lo que hace que esperen hasta que el coordinador se recupera. Esto puede provocar una degradación del rendimiento, especialmente si los participantes manteniendo bloqueos a los recursos compartidos. También pueden ocurrir otros tipos de problemas que hacer que el resultado de la transacción no sea determinista.

Estos problemas se resuelven mediante el protocolo de confirmación de tres fases (3PC), que es esencial divide inicialmente la segunda fase de confirmación en dos subfases llamadas **preparación para la confirmación** y **comprometerse** . La fase de preparación para el compromiso se utiliza para comunicar el resultado de la fase de votación a todos los participantes. Si todos los participantes votan que sí, entonces el coordinador les indica que pasen al estado de preparación para la confirmación. La subfase de confirmación es idéntica a su contraparte de dos fases. Ahora bien, si el coordinador se bloquea durante este subfase, otro participante puede ver la transacción hasta su finalización. Puede simplemente pregúntele a un participante fallado si recibió un mensaje de preparación para confirmar. Si se no lo hizo, entonces asume con seguridad abortar. Por tanto, el estado del protocolo puede recuperarse ered independientemente de qué participante se estrella. Además, al limitar el tiempo requerido para que una transacción se comprometa o cancele hasta un período de tiempo máximo, el protocolo garantiza que una transacción que intenta comprometerse a través de 3PC libera bloqueos en el tiempo de espera.

La idea principal es limitar el tiempo de espera de los participantes que se han preparado para mit y están esperando una confirmación global o un aborto del coordinador. Cuando un participante recibe un mensaje de compromiso previo, sabe que el resto de los participantes han votado para comprometerse. Si no se ha recibido un mensaje de compromiso previo, el par- El participante abortará y liberará todos los bloqueos.

23.4.3 Soporte del sistema operativo para la gestión de transacciones

Los siguientes son los principales beneficios de la transacción compatible con el sistema operativo (SO) administración:

- Normalmente, los DBMS utilizan sus propios semáforos [2] para garantizar la exclusión mutua sive acceso a recursos compartidos. Dado que estos semáforos se implementan en

[2] Los semáforos son estructuras de datos que se utilizan para el acceso sincronizado y exclusivo a recursos compartidos para Previniendo las condiciones de carrera en un sistema informático paralelo.

Página 19

espacio de usuario al nivel del software de aplicación DBMS, el sistema operativo no tiene conocimiento sobre ellos. Por tanto, si el sistema operativo desactiva un proceso DBMS, Al realizar un bloqueo, se bloquean otros procesos DBMS que desean este recurso bloqueado. Tal situación puede causar una seria degradación del rendimiento. Nivel de SO el conocimiento de los semáforos puede ayudar a eliminar tales situaciones.

- Se puede aprovechar el soporte de hardware especializado para el bloqueo para reducir la asociación. Costos calculados. Esto puede ser de gran importancia, ya que el bloqueo es uno de los operaciones comunes de DBMS.
- Proporcionar un conjunto de operaciones de soporte de transacciones comunes a través del kernel permite a los desarrolladores de aplicaciones concentrarse en agregar nuevas funciones a sus productos ucts en lugar de volver a implementar la funcionalidad común para cada aplicación cación. Por ejemplo, si diferentes DDBMS van a coexistir en la misma máquina y eligieron el protocolo de compromiso de dos fases, entonces es más beneficioso tener este protocolo implementado como parte del kernel para que el DDBMS los desarrolladores pueden concentrarse más en agregar nuevas funciones a sus productos.

23.5 Procesamiento y optimización de consultas en Bases de datos distribuidas

Ahora damos una descripción general de cómo un DDBMS procesa y optimiza una consulta. primero discutimos los pasos involucrados en el procesamiento de consultas y luego detallamos la comunicación Costos de comunicación de procesar una consulta distribuida. Luego discutimos una operación especial, llamado semiunión, que se utiliza para optimizar algunos tipos de consultas en un DDBMS. UN La discusión detallada sobre los algoritmos de optimización está más allá del alcance de este texto. Intentamos ilustrar los principios de optimización utilizando ejemplos adecuados. [3]

23.5.1 Procesamiento de consultas distribuidas

Una consulta de base de datos distribuida se procesa en las siguientes etapas:

1. **Asignación de consultas.** La consulta de entrada sobre datos distribuidos se especifica formalmente utilizando un lenguaje de consulta. Luego se traduce a una consulta algebraica en global relaciones. Esta traducción se realiza refiriéndose al concepto global esquema y no tiene en cuenta la distribución real y la réplica ción de datos. Por tanto, esta traducción es en gran parte idéntica a la realizada en un DBMS centralizado. Primero se normaliza, se analiza en busca de errores semánticos, simplificado y finalmente reestructurado en una consulta algebraica.
2. **Localización.** En una base de datos distribuida, la fragmentación da como resultado relaciones almacenados en sitios separados, con algunos fragmentos posiblemente siendo replicados cated. Esta etapa mapea la consulta distribuida en el esquema global para separar Evaluar consultas sobre fragmentos individuales mediante la distribución y replicación de datos. información.

[3] Para una discusión detallada de los algoritmos de optimización, ver Ozsu y Valduriez (1999).

Página 20

3. **Optimización de consultas globales.** La optimización consiste en seleccionar una estrategia de una lista de candidatos más cercana a la óptima. Una lista de consultas de candidatos se puede obtener permutando el orden de las operaciones dentro de un fragmento consulta generada por la etapa anterior. El tiempo es la unidad preferida para medir costo de aumento. El costo total es una combinación ponderada de costos como CPU costo, costos de E / S y costos de comunicación. Dado que los DDB están conectados por un red, a menudo los costes de comunicación a través de la red son los más importantes significativo. Esto es especialmente cierto cuando los sitios están conectados a través de una amplia red de área (WAN).
4. **Optimización de consultas locales.** Esta etapa es común a todos los sitios de la DDB. Las técnicas son similares a las que se utilizan en los sistemas centralizados.

Las primeras tres etapas discutidas anteriormente se realizan en un sitio de control central, mientras que la última etapa se realiza localmente.

23.5.2 Costos de transferencia de datos del procesamiento de consultas distribuidas

Discutimos los problemas relacionados con el procesamiento y la optimización de una consulta en una central DBMS en el Capítulo 19. En un sistema distribuido, varios factores adicionales complicar el procesamiento de consultas. El primero es el costo de transferir datos a través de la red. trabajo. Estos datos incluyen archivos intermedios que se transfieren a otros sitios para procesamiento posterior, así como los archivos de resultados finales que pueden tener que ser transferidos a el sitio donde se necesita el resultado de la consulta. Aunque estos costos pueden no ser muy altos si los sitios están conectados a través de una red de área local de alto rendimiento, se convierten significativo en otros tipos de redes. Por lo tanto, el algoritmo de optimización de consultas DDBMS Los ritmos consideran el objetivo de reducir la cantidad de transferencia de datos como una optimización criterio para elegir una estrategia de ejecución de consultas distribuidas.

Ilustramos esto con dos consultas de muestra simples. Suponga que el EMPLEADO y

Las relaciones de DEPARTAMENTO en la Figura 3.5 se distribuyen en dos sitios como se muestra en la Figura. ure 23,4. Asumiremos en este ejemplo que ninguna relación está fragmentada. Acuerdo- Según la Figura 23.4, el tamaño de la relación EMPLEADO es 100 * 10,000 = 10^6 bytes, y el tamaño de la relación DEPARTAMENTO es 35 * 100 = 3500 bytes. Considere la consulta Q : Para cada empleado, recupere el nombre del empleado y el nombre del departamento para que trabaja el empleado. Esto se puede establecer de la siguiente manera en el álgebra relacional:

$$P: \pi_{Fname,\ Lname,\ Dname} (EMPLOYEE_{Dno\ =\ Dnumber}\ DEPARTMENT)$$

El resultado de esta consulta incluirá 10,000 registros, asumiendo que cada empleado está relacionado con un departamento. Suponga que cada registro en el resultado de la consulta tiene 40 bytes largo. La consulta se envía a un sitio 3 distinto, que se denomina **sitio de resultados.**

porque allí se necesita el resultado de la consulta. Ni el EMPLEADO ni el
Las relaciones de DEPARTAMENTO residen en el sitio 3. Hay tres estrategias simples para ejecutar ing esta consulta distribuida:

1. Transferir las relaciones EMPLEADO y DEPARTAMENTO al resultado sitio y realice la combinación en el sitio 3. En este caso, un total de 1,000,000 + 3,500 = Se deben transferir 1,003,500 bytes.

Página 21

Sitio 1:

EMPLEADO

Fname	Minit	Lname	Ssn	Bdate	Habla a	Sexo	Salario	Super_ssn	Dno

10,000 registros
cada registro tiene 100 bytes de longitud
El campo SSN tiene una longitud de 9 bytes El campo Fname tiene 15 bytes de longitud
El campo Dno tiene 4 bytes de longitud El campo Lname tiene 15 bytes de longitud

Sitio 2:

DEPARTAMENTO

Dname	Dnumber	Mgr_ssn	Mgr_start_date

100 registros
cada registro tiene 35 bytes de longitud
El campo Dnumber tiene 4 bytes de longitud El campo Dname tiene 10 bytes de longitud
El campo Mgr_ssn tiene 9 bytes de longitud

Figura 23.4
Ejemplo para ilustrar volumen de datos transferido.

2. Transfiera la relación EMPLEADO al sitio 2, ejecute la combinación en el sitio 2 y envíe el resultado al sitio 3. El tamaño del resultado de la consulta es 40 * 10,000 = 400,000 bytes, por lo que se deben transferir 400.000 + 1.000.000 = 1.400.000 bytes.
3. Transfiera la relación DEPARTAMENTO al sitio 1, ejecute la combinación en el sitio 1 y enviar el resultado al sitio 3. En este caso, 400.000 + 3500 = 403.500 bytes deben ser transferido.

Si minimizar la cantidad de transferencia de datos es nuestro criterio de optimización, deberíamos elija la estrategia 3. Ahora considere otra consulta Q ′: Para cada departamento, recupere el nombre del departamento y el nombre del director del departamento. Esto se puede afirmar como sigue en el álgebra relacional:

$$Q' : \pi_{Fname,\ Lname,\ Dname} (DEPARTAMENTO\ _{Mgr\_ssn\ =\ Ssn}\ EMPLOYEE)$$

Nuevamente, suponga que la consulta se envía en el sitio 3. Las mismas tres estrategias para la ejecución de la consulta Q se aplica a Q ′, excepto que el resultado de Q ′ incluye solo 100 registros, asumiendo que cada departamento tiene un gerente:

1. Transferir las relaciones EMPLEADO y DEPARTAMENTO al resultado sitio y realice la combinación en el sitio 3. En este caso, un total de 1,000,000 + 3,500 =

Se deben transferir 1,003,500 bytes.

2. Transfiera la relación EMPLEADO al sitio 2, ejecute la combinación en el sitio 2 y envíe el resultado al sitio 3. El tamaño del resultado de la consulta es 40 * 100 = 4.000 bytes, por lo que Se deben transferir 4,000 + 1,000,000 = 1,004,000 bytes.

3. Transfiera la relación DEPARTAMENTO al sitio 1, ejecute la combinación en el sitio 1 y enviar el resultado al sitio 3. En este caso, 4,000 + 3,500 = 7,500 bytes deben ser transferido.

Nuevamente, elegiríamos la estrategia 3, esta vez por un margen abrumador sobre estrategias 1 y 2. Las tres estrategias anteriores son las más obvias para el

Página 22

caso en el que el sitio de resultados (sitio 3) es diferente de todos los sitios que contienen archivos involucrados en la consulta (sitios 1 y 2). Sin embargo, suponga que el sitio de resultados es el sitio 2; entonces tenemos dos estrategias simples:

1. Transfiera la relación EMPLEADO al sitio 2, ejecute la consulta y presente el resultado para el usuario en el sitio 2. Aquí, la misma cantidad de bytes (1,000,000) debe transferirse tanto para Q como para Q ′.

2. Transfiera la relación DEPARTAMENTO al sitio 1, ejecute la consulta en el sitio 1 y enviar el resultado al sitio 2. En este caso 400,000 + 3,500 = 403,500 bytes deben transferirse para Q y 4000 + 3500 = 7500 bytes para Q ′.

Una estrategia más compleja, que a veces funciona mejor que estas simples estrategias gies, utiliza una operación llamada **semiunión** . Introducimos esta operación y discutimos ejecución distribuida utilizando semiuniones a continuación.

23.5.3 Procesamiento de consultas distribuidas mediante Semijoin

La idea detrás del procesamiento de consultas distribuidas utilizando la operación semiunión es reducir el número de tuplas en una relación antes de transferirla a otro sitio. Intuitivamente, la idea es enviar la columna de unión de una relación R al sitio donde la otra relación S está ubicada; esta columna se une luego con S. A continuación, los atributos de unión, junto con los atributos requeridos en el resultado, se proyectan y enviado de vuelta al sitio original y unido con R. Por lo tanto, solo la unión columna de R se transfiere en una dirección, y un subconjunto de S sin extrane- Nuestras tuplas o atributos se transfieren en la otra dirección. Si solo una pequeña fracción de las tuplas en S participan en la combinación, esta puede ser una solución eficiente para mizing transferencia de datos.

Para ilustrar esto, considere la siguiente estrategia para ejecutar Q o Q ′:

1. Proyecte los atributos de unión de DEPARTMENT en el sitio 2 y transfiéralos al sitio 1. Para Q , transferimos F = $\pi_{Dnumber}$ (DEPARTAMENTO), cuyo tamaño es 4 * 100 = 400 bytes, mientras que para Q ′, transferimos F ′ = π_{Mgr_ssn} (DEPARTAMENTO), cuyo tamaño es

9 * 100 = 900 bytes.

2. **Una** el archivo transferido con la relación EMPLEADO en el sitio 1 y transfiera los atributos requeridos del archivo resultante al sitio 2. Para Q , transferimos $R = \pi_{Dno, Fname, Lname} (F_{Dnumber = Dno} EMPLOYEE)$, cuyo tamaño es 34 * 10,000 = 340.000 bytes, mientras que para Q ′, transferimos $R' = \pi_{Mgr\_ssn, Fname, Lname} (F'_{Mgr\_ssn = Ssn} EMPLOYEE)$, cuyo tamaño es 39 * 100 = 3.900 bytes.

3. Ejecute la consulta uniendo el archivo transferido R o R ′ con DEPARTMENT , y presentar el resultado al usuario en el sitio 2.

Usando esta estrategia, transferimos 340,400 bytes para Q y 4,800 bytes para Q ′. Nosotros limitó los atributos de EMPLEADO y las tuplas transmitidas al sitio 2 en el paso 2 solo a aquellos que en realidad se unirá con una tupla DEPARTMENT en el paso 3. Para la consulta Q , este resultó incluir todas las tuplas de EMPLEADOS , por lo que se logró poca mejora. Sin embargo, para Q ′ solo se necesitaron 100 de las 10,000 tuplas de EMPLEADOS .

Página 23

La operación de semiunión fue ideada para formalizar esta estrategia. Una **operación semiunida** $R_{A = B} S$, donde A y B son atributos compatibles con el dominio de R y S, respectivamente, produce el mismo resultado que la expresión de álgebra relacional $\pi R (R_{A = B} S)$. En un disentorno tributario donde R y S residen en diferentes sitios, la semiunión es típicamente implementado transfiriendo primero $F = \pi_B (S)$ al sitio donde reside R y luego uniring F con R, lo que conduce a la estrategia discutida aquí.

Observe que la operación de semiunión no es conmutativa; es decir,

$$RS \neq SR$$

23.5.4 Descomposición de consultas y actualizaciones

En un DDBMS sin transparencia de distribución, el usuario formula una consulta directamente en términos de fragmentos específicos. Por ejemplo, considere otra consulta P : recuperar el nombres y horas por semana para cada empleado que trabaja en algún proyecto controlado por departamento 5, que se especifica en la base de datos distribuida donde las relaciones en los sitios 2 y 3 se muestran en la Figura 23.2, y los del sitio 1 se muestran en la Figura ure 5.6, como en nuestro ejemplo anterior. Un usuario que envía una consulta de este tipo debe especificar si hace referencia a las relaciones PROJS_5 y WORKS_ON_5 en el sitio 2 (Fig. ure 23.2) o las relaciones PROJECT y WORKS_ON en el sitio 1 (Figura 5.6). El usuario también debe mantener la coherencia de los elementos de datos replicados al actualizar un DDBMS sin transparencia de replicación.

Por otro lado, un DDBMS que admita distribución completa, fragmentación y La transparencia de replicación permite al usuario especificar una consulta o una solicitud de actualización en el esquema de la Figura 5.5 como si el DBMS estuviera centralizado. Para actualizaciones, el DDBMS es responsable de mantener la coherencia entre los elementos replicados mediante el uso de uno de los algoritmos de control de concurrencia distribuidos discutidos en

Sección 23.3. Para las consultas, un módulo de **descomposición de consultas** debe dividirse o **descomponer** una consulta en **subconsultas** que se pueden ejecutar en los sitios individuales. Además, una estrategia para combinar los resultados de las subconsultas para formar el se debe generar el resultado de la consulta. Siempre que el DDBMS determine que un artículo referenciado en la consulta es replicado, debe elegir o **materializar** un particular réplica durante la ejecución de la consulta.

Para determinar qué réplicas incluyen los elementos de datos a los que se hace referencia en una consulta, DDBMS se refiere a la información de fragmentación, replicación y distribución almacenados en el catálogo DDBMS. Para la fragmentación vertical, la lista de atributos para cada fragmento se guarda en el catálogo. Para la fragmentación horizontal, una condición, a veces llamado **guardia** , se guarda para cada fragmento. Esta es básicamente una selección condición que especifica qué tuplas existen en el fragmento; se llama guardia porque solo las tuplas que satisfacen esta condición pueden almacenarse en el fragmento. Para fragmentos mixtos, tanto la lista de atributos como la condición de protección se mantienen en el catálogo.

En nuestro ejemplo anterior, las condiciones de protección para los fragmentos en el sitio 1 (Figura 5.6) son VERDADERO (todas las tuplas) y las listas de atributos son * (todos los atributos). Por los fragmentos

Página 24

(a) EMPD5
lista de atributos: Fname, Minit, Lname, Ssn, Salario, Super_ssn, Dno
condición de guardia: Dno = 5
DEP5
lista de atributos: * (todos los atributos Dname, Dnumber, Mgr_ssn, Mgr_start_date)
condición de guardia: Dnumber = 5
DEP5_LOCS
lista de atributos: * (todos los atributos Dnumber, Location)
condición de guardia: Dnumber = 5
PROJS5
lista de atributos: * (todos los atributos Pname, Pnumber, Plocation, Dnum)
condición de guardia: Dnum = 5
WORKS_ON5
lista de atributos: * (todos los atributos Essn, Pno, Horas)
condición de protección: Essn IN (π_{Ssn} (EMPD5)) O Pno IN ($\pi_{Pnumber}$ (PROJS5))

(b) EMPD4
lista de atributos: Fname, Minit, Lname, Ssn, Salario, Super_ssn, Dno
condición de guardia: Dno = 4
DEP4
lista de atributos: * (todos los atributos Dname, Dnumber, Mgr_ssn, Mgr_start_date)
condición de guardia: Dnumber = 4
DEP4_LOCS
lista de atributos: * (todos los atributos Dnumber, Location)

condición de guardia: Dnumber = 4

PROJS4

lista de atributos: * (todos los atributos Pname, Pnumber, Plocation, Dnum)

condición de guardia: Dnum = 4

WORKS_ON4

lista de atributos: * (todos los atributos Essn, Pno, Horas)

condición de guardia: Essn IN (π_{Ssn} (EMPD4))

O Pno IN ($\pi_{Pnumber}$ (PROJS4))

Figura 23.5
Condiciones de guardia y listas de atributos para fragmentos.
(a) Fragmentos del sitio 2.
(b) Fragmentos del sitio 3.

como se muestra en la Figura 23.2, tenemos las condiciones de protección y las listas de atributos que se muestran en Figura 23.5. Cuando el DDBMS descompone una solicitud de actualización, puede determinar qué fragmentos deben actualizarse examinando sus condiciones de protección. Para examen- Por ejemplo, una solicitud de usuario para insertar una nueva tupla EMPLEADO <'Alex', 'B', 'Coleman', '345671239', '22 -APR-64 ',' 3306 Arenisca, Houston, TX ', M, 33000,' 987654321 ', 4> sería descompuesto por el DDBMS en dos solicitudes de inserción: las primeras inserciones la tupla anterior en el fragmento EMPLOYEE en el sitio 1, y la segunda inserta el tupla proyectada <'Alex', 'B', 'Coleman', '345671239', 33000, '987654321', 4> en el EMPD4 fragmento en el sitio 3.

Para la descomposición de consultas, el DDBMS puede determinar qué fragmentos pueden contener las tuplas requeridas comparando la condición de consulta con las condiciones de protección. por

Página 25

Por ejemplo, considere la consulta P : recuperar los nombres y las horas por semana para cada empleado que trabaja en algún proyecto controlado por el departamento 5. Esto se puede especificar fied en SQL en el esquema en la Figura 5.5 como sigue:

```
P: SELECCIONE Fname, Lname, Horas
     DE EMPLOYEE, PROJECT, WORKS_ON
     DONDE Dnum = 5 Y Pnumber = Pno Y Essn = Ssn;
```

Suponga que la consulta se envía en el sitio 2, que es donde estará el resultado de la consulta. necesario. El DDBMS puede determinar a partir de la condición de protección en PROJS5 y WORKS_ON5 que todas las tuplas satisfacen las condiciones (Dnum = 5 AND Pnumber = Pno) residir en el sitio 2. Por lo tanto, puede descomponer la consulta en la siguiente alge- subconsultas de sujetador:

$T_1 \leftarrow \pi_{Essn}$ (PROJS5 $_{Pnumber = Pno}$ WORKS_ON5)

$T_2 \leftarrow \pi_{Essn, Fname, Lname}$ (T_1 $_{Essn = Ssn}$ EMPLOYEE)

RESULTADO $\leftarrow \pi_{Fname, Lname, Horas}$ (T_2 * WORKS_ON5)

Esta descomposición se puede utilizar para ejecutar la consulta mediante una estrategia de semiunión. El DDBMS sabe por las condiciones de guardia que PROJS5 contiene exactamente esos tuplas satisfactorias (Dnum = 5) y que WORKS_ON5 contiene todas las tuplas que se van a unir con PROJS5 ; por lo tanto, la subconsulta T_1 se puede ejecutar en el sitio 2, y la columna proyectada

Essn se puede enviar al sitio 1. La subconsulta T_2 se puede ejecutar en el sitio 1 y el resultado se puede enviar de vuelta al sitio 2, donde el resultado final de la consulta se calcula y se muestra a el usuario. Una estrategia alternativa sería enviar la propia consulta Q al sitio 1, que incluye todas las tuplas de la base de datos, dónde se ejecutaría localmente y desde qué el resultado se enviaría de vuelta al sitio 2. El optimizador de consultas calcularía los costos de ambas estrategias y elegiría la que tenga el costo estimado más bajo.

23.6 Tipos de sistemas de bases de datos distribuidos

El término sistema de gestión de bases de datos distribuidas puede describir varios sistemas que se diferencian entre sí en muchos aspectos. Lo principal que todos estos sistemas tienen en común el hecho de que los datos y el software se distribuyen en varios sitios conectado por alguna forma de red de comunicación. En esta sección, discutimos un número de tipos de DDBMS y los criterios y factores que hacen que algunos de estos sistemas diferentes.

El primer factor que consideramos es el **grado de homogeneidad** del software DDBMS. Si todos los servidores (o DBMS locales individuales) utilizan software idéntico y todos los usuarios (clientes) utilizan software idéntico, el DDBMS se denomina **homogéneo** ; de lo contrario, se llama **hetero-generoso** . Otro factor relacionado con el grado de homogeneidad es el **grado de autonomía** . Si no existe ninguna disposición para que el sitio local funcione como un DBMS independiente, entonces el sistema **no** tiene **autonomía local** . Por otro lado, si el acceso directo por locales Las transacciones a un servidor están permitidas, el sistema tiene cierto grado de autonomía local.

La figura 23.6 muestra la clasificación de alternativas de DDBMS a lo largo de ejes ortogonales de distribución, autonomía y heterogeneidad. Para una base de datos centralizada, hay

Página 26

866 Capítulo 23 Conceptos de bases de datos distribuidas

Distribución

segundo

Autonomía

UN

C re

Leyenda:

A: base de datos centralizada tradicional sistemas

B: sistemas de bases de datos distribuidos puros

Figura 23.6
Clasificación de distribuido bases de datos.

completa autonomía, pero una total falta de distribución y heterogeneidad (punto A en la figura). Vemos que el grado de autonomía local proporciona más terreno para clasificación en sistemas federados y de bases de datos múltiples. En un extremo del espectro de autonomía, tenemos un DDBMS que parece un DBMS centralizado para el usuario, con autonomía cero (punto B). Existe un solo esquema conceptual, y todos El acceso al sistema se obtiene a través de un sitio que forma parte del DDBMS, que significa que no existe autonomía local. A lo largo del eje de la autonomía encontramos dos tipos de DDBMS llamados sistema de base de datos federado (punto C) y base de datos múltiple sistema (punto D). En tales sistemas, cada servidor es independiente y autónomo. DBMS centralizado que tiene sus propios usuarios locales, transacciones locales y DBA, y por tanto, tiene un grado muy alto de autonomía local. El término **base de datos federada El sistema (FDBS)** se utiliza cuando hay alguna vista o esquema global de la federación. de bases de datos que comparten las aplicaciones (punto C). Por otro lado, un **El sistema de bases de datos múltiples** tiene total autonomía local en el sentido de que no tiene un esquema, pero construye interactivamente uno según lo necesite la aplicación (punto D). Ambos sistemas son híbridos entre sistemas distribuidos y centralizados, y el La distinción que hicimos entre ellos no se sigue estrictamente. Nos referiremos a ellos como FDBS en un sentido genérico. El punto D en el diagrama también puede representar un sistema con completa autonomía local y total heterogeneidad: esta podría ser una base de datos de igual a igual sistema. En un FDBS heterogéneo, un servidor puede ser un DBMS relacional, otro un DBMS de red (como IDMS de Computer Associates o HP'S IMAGE / 3000), y

Página 27

un tercero, un DBMS de objeto (como ObjectStore de Object Design) o jerárquico DBMS (como el IMS de IBM); en tal caso, es necesario tener un sistema canónico idioma e incluir traductores de idiomas para traducir subconsultas del idioma canónico al idioma de cada servidor.

A continuación, discutimos brevemente los problemas que afectan el diseño de los FDBS.

23.6.1 Problemas de los sistemas de gestión de bases de datos federadas

El tipo de heterogeneidad presente en las FDBS puede surgir de varias fuentes. Nosotros discutir estas fuentes primero y luego señalar cómo los diferentes tipos de autonomías contribuir a una heterogeneidad semántica que debe resolverse de forma heterogénea FDBS.

- **Diferencias en los modelos de datos.** Las bases de datos en una organización provienen de una variedad ety de modelos de datos, incluidos los llamados modelos heredados (jerárquicos y red), el modelo de datos relacionales, el modelo de datos de objetos e incluso archivos. Las capacidades de modelado de los modelos varían. Por tanto, para tratar con ellos formalmente a través de un solo esquema global o procesarlos en un solo idioma es desafiante. Incluso si dos bases de datos son ambas del entorno RDBMS, la misma información puede representarse como un nombre de atributo, como una relación nombre, o como valor en diferentes bases de datos. Esto requiere una consulta inteligente Mecanismo de procesamiento que puede relacionar información en base a metadatos.

- **Diferencias en las limitaciones.** Facilidades de restricción para la especificación e implementación mentación varían de un sistema a otro. Hay características comparables que debe conciliarse en la construcción de un esquema global. Por ejemplo, el Las relaciones de los modelos ER se representan como integridad referencial, tensiones en el modelo relacional. Puede que sea necesario utilizar activadores para implementar ciertas restricciones en el modelo relacional. El esquema global también debe tratar con posibles conflictos entre las limitaciones.

- **Diferencias en los lenguajes de consulta.** Incluso con el mismo modelo de datos, el lenguaje Los calibres y sus versiones varían. Por ejemplo, SQL tiene varias versiones como SQL-89, SQL-92, SQL-99 y SQL: 2008, y cada sistema tiene su propio conjunto de tipos de datos, operadores de comparación, funciones de manipulación de cadenas, etc.

Heterogeneidad semántica. La heterogeneidad semántica ocurre cuando hay diferencias diferencias en el significado, interpretación y uso previsto de los mismos datos o de datos relacionados. La heterogeneidad semántica entre los sistemas de bases de datos de componentes (DBS) crea la mayor obstáculo en el diseño de esquemas globales de bases de datos heterogéneas. El **diseño La autonomía** de los componentes DBS se refiere a su libertad de elegir lo siguiente parámetros de diseño; los parámetros de diseño, a su vez, afectan la eventual complejidad de el FDBS:

- **El universo de discurso del que se extraen los datos.** Por ejemplo, para dos cuentas de clientes, las bases de datos de la federación pueden ser de los Estados Unidos Estados y Japón y tienen conjuntos de atributos completamente diferentes sobre el cliente cuentas requeridas por las prácticas contables. Fluctuaciones del tipo de cambio

Página 28

también presentaría un problema. Por lo tanto, las relaciones en estas dos bases de datos que tienen nombres idénticos, CLIENTE o CUENTA, pueden tener algunos y alguna información completamente distinta.

- **Representación y denominación.** La representación y denominación de elementos de datos Mentos y la estructura del modelo de datos pueden ser preespecificados para cada local base de datos.

- **La comprensión, el significado y la interpretación subjetiva de los datos.** Esta es un contribuyente principal a la heterogeneidad semántica.

- **Restricciones de transacciones y políticas.** Estos tratan con los protocolos de serialización,
- **Derivación de resúmenes.** Agregación, resumen y otros datos funciones de procesamiento y operaciones admitidas por el sistema.

Los problemas anteriores relacionados con la heterogeneidad semántica están siendo enfrentados por todos los principales Organizaciones multinacionales y gubernamentales en todas las áreas de aplicación. En el de hoy entorno comercial, la mayoría de las empresas están recurriendo a FDBS heterogéneos, haber invertido mucho en el desarrollo de sistemas de bases de datos individuales utilizando diversos modelos de datos en diferentes plataformas durante los últimos 20 a 30 años. Empresas están utilizando varias formas de software, normalmente llamado **middleware** ; o Web-paquetes basados en **servidores de aplicaciones** (por ejemplo, WebLogic o WebSphere); e incluso sistemas genéricos, llamados sistemas de **planificación de recursos empresariales (ERP)** (para ejemplo, SAP, JD Edwards ERP), para gestionar el transporte de consultas y acciones de la aplicación global a bases de datos individuales (con posibles procesamiento para reglas comerciales) y los datos de los servidores de bases de datos heterogéneos a la aplicación global. Una discusión detallada de estos tipos de sistemas de software es fuera del alcance de este texto.

Así como proporcionar la máxima transparencia es el objetivo de cualquier base de datos distribuida arquitectura, las bases de datos de componentes locales se esfuerzan por preservar la autonomía. **La autonomía** de **comunicación** de un componente DBS se refiere a su capacidad para decidir si comunicarse con otro componente DBS. **Autonomía de ejecución** se refiere a la capacidad de un DBS de componente para ejecutar operaciones locales sin inter-diferencia de operaciones externas por parte de otros DBS componentes y su capacidad para decidir el orden en el que se ejecutarán. La **autonomía** de **asociación** de un componente DBS implica que tiene la capacidad de decidir si desea compartir sus funcionalidad (operaciones que soporta) y recursos (datos que gestiona) con otros DBS de componentes. El principal desafío de diseñar FDBS es permitir que los componentes Los DBS interoperan al mismo tiempo que les proporcionan los tipos de autonomías anteriores.

23.7 Arquitecturas de bases de datos distribuidas

En esta sección, primero señalamos brevemente la distinción entre paralelo y distribución arquitecturas de base de datos uted. Aunque ambos prevalecen en la industria actual, existen diversas manifestaciones de las arquitecturas distribuidas que evolucionan continuamente entre las grandes empresas. La arquitectura paralela es más común en

Página 29

computación formal, donde hay una necesidad de arquitecturas multiprocesador para hacer frente con el volumen de datos que se procesan y almacenan en transacciones aplicaciones. Luego presentamos una arquitectura genérica de una base de datos distribuida. Esto es seguido por discusiones sobre la arquitectura de cliente / servidor de tres niveles y sistemas de bases de datos federadas.

23.7.1 Arquitecturas paralelas versus distribuidas

Hay dos tipos principales de arquitecturas de sistemas multiprocesador que monplace:

- **Arquitectura de memoria compartida (estrechamente acoplada).** Comparten varios procesadores almacenamiento secundario (disco) y también comparten la memoria primaria.
- **Arquitectura de disco compartido (débilmente acoplado).** Múltiples procesadores comparten sec- almacenamiento secundario (disco), pero cada uno tiene su propia memoria primaria.

Estas arquitecturas permiten a los procesadores comunicarse sin la sobrecarga de intercambiar mensajes a través de una red. [4] Sistemas de gestión de bases de datos desarrollados el uso de los tipos de arquitecturas anteriores se denomina **gestión de bases de datos paralelas sistemas en** lugar de DDBMS, ya que utilizan tecnología de procesador paralelo. Otro tipo de arquitectura de multiprocesador se llama **arquitectura de nada compartido** . En esta arquitectura, cada procesador tiene su propio (disco) primario y secundario memoria, no existe una memoria común, y los procesadores se comunican a través de un alto Red de interconexión de velocidad (bus o conmutador). Aunque la nada compartida La arquitectura se asemeja a un entorno informático de base de datos distribuida, Existen diferencias en el modo de funcionamiento. En sistemas multiprocesador de nada compartido, hay simetría y homogeneidad de nodos; esto no es cierto para el distribuido entorno de base de datos, donde la heterogeneidad del hardware y el sistema operativo en cada nodo es muy común. La arquitectura de nada compartido también se considera una entorno para bases de datos paralelas. La figura 23.7 (a) ilustra una base de datos paralela (nada compartido), mientras que la Figura 23.7 (b) ilustra una base de datos centralizada con acceso tributado y la Figura 23.7 (c) muestra una base de datos distribuida pura. Nosotros no amplíe aquí las arquitecturas paralelas y los problemas relacionados con la gestión de datos.

23.7.2 Arquitectura general de bases de datos distribuidas puras

En esta sección, discutimos los modelos arquitectónicos lógicos y de componentes de un DDB. En la Figura 23.8, que describe la arquitectura de esquema genérico de un DDB, el La empresa se presenta con una vista coherente y unificada que muestra la estructura lógica. de datos subyacentes en todos los nodos. Esta visión está representada por la concepción global esquema tual (GCS), que proporciona transparencia de red (consulte la Sección 23.1.2). A acomodar la heterogeneidad potencial en el DDB, cada nodo se muestra como teniendo su propio esquema interno local (LIS) basado en detalles de la organización física en ese

[4] Si se comparten las memorias primarias y secundarias, la arquitectura también se conoce como shared-every- arquitectura de cosas.

Página 30

(un) Sistema informático 1 Sistema informático 2

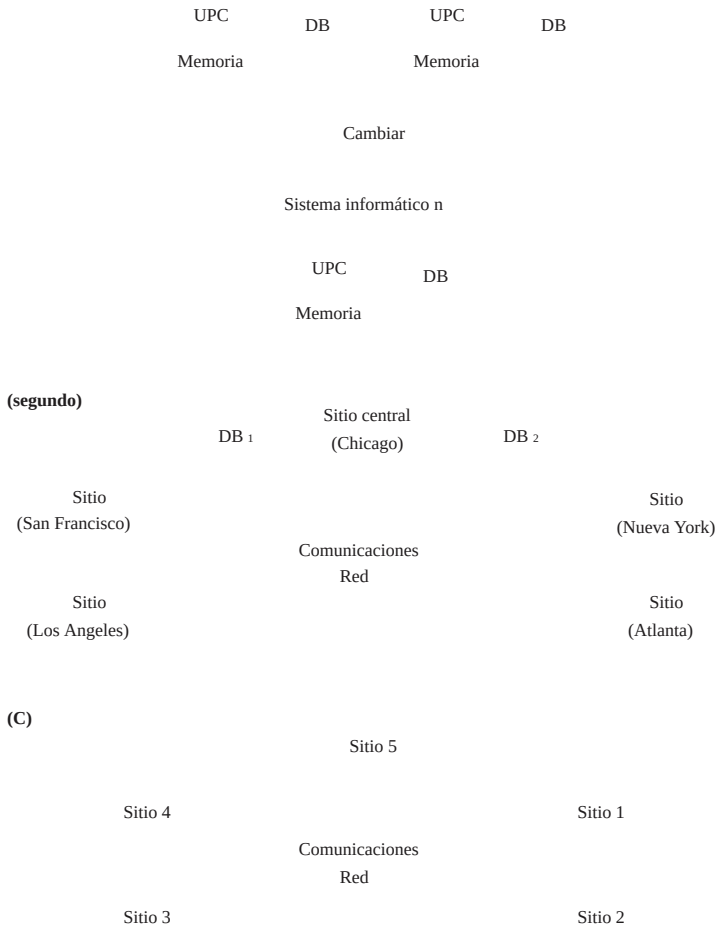

Figura 23.7
Algunas arquitecturas de sistemas de bases de datos diferentes. (a) Arquitectura de nada compartido. (b) Una arquitectura en red con un base de datos centralizada en uno de los sitios. (c) Una arquitectura de base de datos verdaderamente distribuida.

sitio en particular. La organización lógica de los datos en cada sitio la especifica el local esquema conceptual (LCS). GCS, LCS y sus asignaciones subyacentes proporcionan la transparencia de fragmentación y replicación discutida en la Sección 23.1.2. Higo- ure 23.8 muestra la arquitectura de componentes de una DDB. Es una extensión de su centro contraparte tralizada (Figura 2.3) en el Capítulo 2. En aras de la simplicidad, los

Página 31

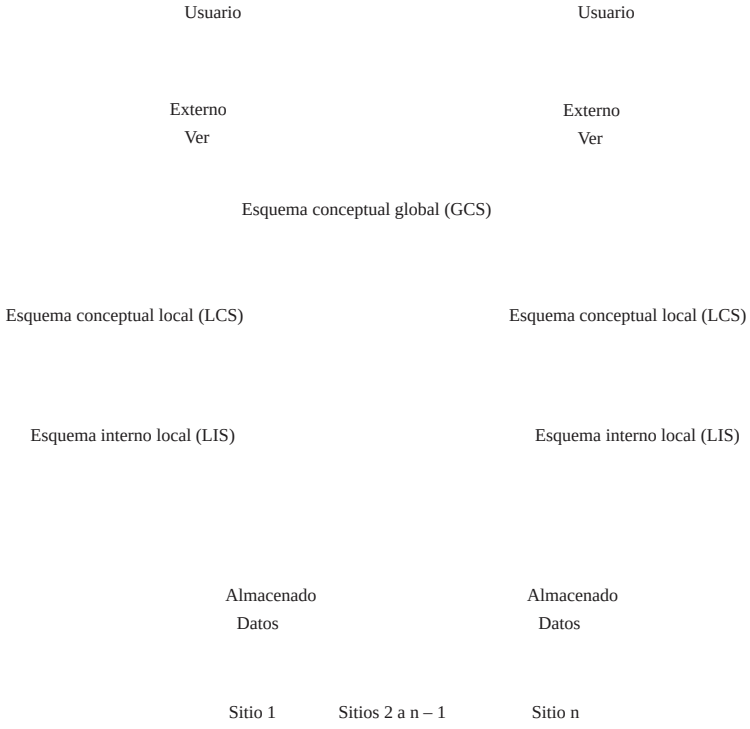

Figura 23.8
Arquitectura de esquema de bases de datos distribuidas.

los elementos no se muestran aquí. El compilador de consultas globales hace referencia al esquema conceptual del catálogo del sistema global para verificar e imponer definido limitaciones. El optimizador de consultas global hace referencia a conceptos tanto globales como locales. esquemas y genera consultas locales optimizadas a partir de consultas globales. Evalúa todo Estrategias candidatas que utilizan una función de costo que estima el costo en función de la respuesta. tiempo (CPU, E / S y latencias de red) y tamaños estimados de resultados intermedios. Este último es particularmente importante en consultas que involucran combinaciones. Habiendo calculado el costo para cada candidato, el optimizador selecciona al candidato con el costo mínimo para su ejecución. Cada DBMS local tendría su optimizador de consultas local, transacción administrador y motores de ejecución, así como el catálogo del sistema local, que alberga el esquemas locales. El gerente de transacciones globales es responsable de coordinar ejecución en varios sitios junto con el administrador de transacciones local en esos sitios.

23.7.3 Arquitectura de esquema de base de datos federada

Arquitectura de esquema de cinco niveles típica para admitir aplicaciones globales en el FDBS El entorno se muestra en la Figura 23.9. En esta arquitectura, el **esquema local** es el

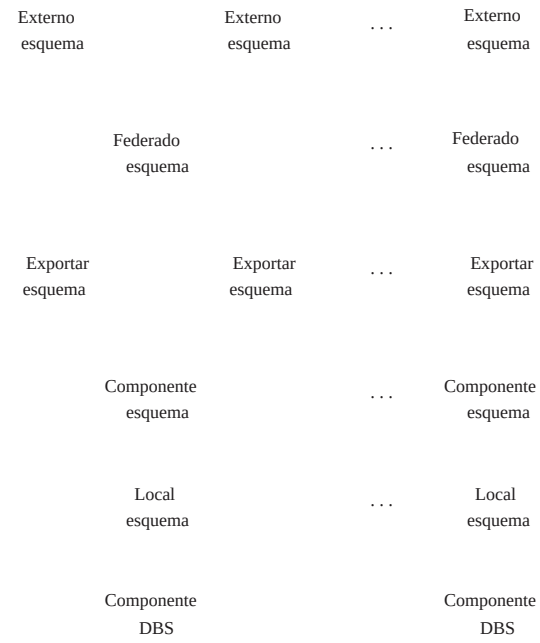

Figura 23.9
La arquitectura de esquema de cinco niveles en un sistema de base de datos federado (FDBS).

Fuente: Adaptado de Sheth y Larson, “Sistemas de bases de datos federadas para gestionar distribuido, Heterogéneo y autónomo Bases de datos ". Encuestas de computación ACM (Vol. 22: No. 3, septiembre de 1990).

esquema conceptual (definición de base de datos completa) de una base de datos de componentes, y el **El esquema de componentes** se deriva traduciendo el esquema local en datos canónicos. modelo o modelo de datos comunes (CDM) para el FDBS. Traducción de esquema del esquema local al esquema de componente se acompaña de la generación de asignaciones a transformar comandos en un esquema de componente en comandos en el correspondiente ing esquema local. El **esquema de exportación** representa el subconjunto de un esquema de componente. que está disponible para FDBS. El **esquema federado** es el esquema o vista global, que es el resultado de integrar todos los esquemas de exportación compartibles. El **externo Los esquemas** definen el esquema para un grupo de usuarios o una aplicación, como en los tres niveles. arquitectura de esquema.

Todos los problemas relacionados con el procesamiento de consultas, el procesamiento de transacciones y el directorio. y la gestión y recuperación de metadatos se aplican a los FDBS con consideraciones adicionales aciones. No está dentro de nuestro alcance discutirlos en detalle aquí.

23.7.4 Descripción general de la arquitectura cliente / servidor de tres niveles

Como señalamos en la introducción del capítulo, los DDBMS a gran escala no han sido desarrollado para soportar todos los tipos de funcionalidades que hemos comentado hasta ahora. En cambio, se están desarrollando aplicaciones de bases de datos distribuidas en el contexto de la arquitecturas cliente / servidor. Introdujimos la arquitectura cliente / servidor de dos niveles en

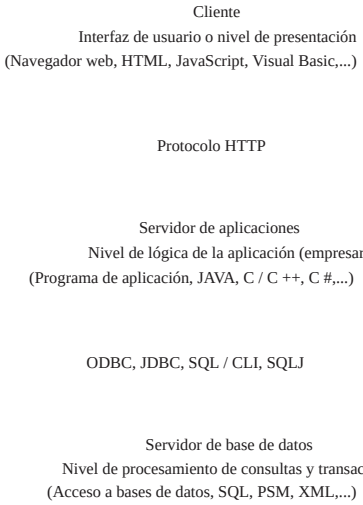

Figura 23.10
El cliente / servidor de tres niveles arquitectura.

Sección 2.5. Ahora es más común utilizar una arquitectura de tres niveles en lugar de una Arquitectura de dos niveles, particularmente en aplicaciones web. Esta arquitectura es ilustrazada en la Figura 23.10.

En la arquitectura cliente / servidor de tres niveles, existen las siguientes tres capas:

1. **Capa de presentación (cliente).** Esto proporciona la interfaz de usuario e interactúa con el usuario. Los programas en esta capa presentan interfaces web o formularios para el cliente para interactuar con la aplicación. Los navegadores web suelen utilizado, y los idiomas y especificaciones utilizados incluyen HTML, XHTML, CSS, Flash, MathML, Gráficos vectoriales escalables (SVG), Java, JavaScript, Adobe Flex y otros. Esta capa maneja la entrada, salida y navegación del usuario. aceptando comandos de usuario y mostrando la información necesaria, generalmente en forma de páginas web estáticas o dinámicas. Estos últimos están empleados cuando la interacción implica el acceso a la base de datos. Cuando una interfaz web utilizado, esta capa normalmente se comunica con la capa de aplicación a través de la Protocolo HTTP.
2. **Capa de aplicación (lógica empresarial).** Esta capa programa la aplicación lógica. Por ejemplo, las consultas se pueden formular basándose en la entrada del usuario desde el cliente, o los resultados de la consulta se pueden formatear y enviar al cliente para su presentación ción. La funcionalidad adicional de la aplicación se puede manejar en esta capa, como como controles de seguridad, verificación de identidad y otras funciones. La aplicación La capa puede interactuar con una o más bases de datos o fuentes de datos según sea necesario por conectarse a la base de datos mediante ODBC, JDBC, SQL / CLI u otra base de datos técnicas de acceso.

3. **Servidor de base de datos.** Esta capa maneja las solicitudes de consulta y actualización desde el capa de aplicación, procesa las solicitudes y envía los resultados. Generalmente SQL se utiliza para acceder a la base de datos si es relacional u objeto-relacional, y se almacena también pueden invocarse procedimientos de base de datos. Los resultados de las consultas (y las consultas) pueden ser formateados en XML (consulte el Capítulo 13) cuando se transmite entre servidor de aplicaciones y servidor de base de datos.

Exactamente cómo dividir la funcionalidad DBMS entre el cliente, el servidor de aplicaciones, y el servidor de la base de datos puede variar. El enfoque común es incluir el funcional- idad de un DBMS centralizado a nivel de servidor de base de datos. Varios DBMS relacionales Los productos han adoptado este enfoque, en el que se proporciona un **servidor SQL** . La aplicación El servidor de cation debe formular las consultas SQL apropiadas y conectarse al servidor de base de datos cuando sea necesario. El cliente proporciona el procesamiento de la interfaz de usuario. interacciones. Dado que SQL es un estándar relacional, varios servidores SQL, posiblemente Vido por diferentes proveedores, puede aceptar comandos SQL a través de estándares como ODBC, JDBC y SQL / CLI (consulte el Capítulo 10).

En esta arquitectura, el servidor de aplicaciones también puede hacer referencia a un diccionario de datos. que incluye información sobre la distribución de datos entre los distintos SQL servidores, así como módulos para descomponer una consulta global en varios consultas locales que se pueden ejecutar en los distintos sitios. Interacción entre un El servidor de aplicaciones y el servidor de base de datos pueden proceder de la siguiente manera durante el proceso. cesación de una consulta SQL:

1. El servidor de aplicaciones formula una consulta de usuario basada en la entrada del cliente. ent y lo descompone en una serie de consultas de sitio independientes. Cada La consulta del sitio se envía al sitio del servidor de base de datos correspondiente.
2. Cada servidor de base de datos procesa la consulta local y envía los resultados al sitio del servidor de aplicaciones. XML se promociona cada vez más como el estándar para intercambio de datos (vea el Capítulo 13), por lo que el servidor de la base de datos puede formatear la consulta resultado en XML antes de enviarlo al servidor de aplicaciones.
3. El servidor de aplicaciones combina los resultados de las subconsultas para producir el resultado de la consulta requerida originalmente, la formatea en HTML o en algún otro formulario aceptado por el cliente y lo envía al sitio del cliente para su visualización.

El servidor de aplicaciones es responsable de generar un plan de ejecución distribuido para una consulta o transacción multisitio y para supervisar la ejecución distribuida mediante envío enviar comandos a los servidores. Estos comandos incluyen consultas y transacciones locales a ejecutar, así como comandos para transmitir datos a otros clientes o servidores. Otra función controlada por el servidor de aplicaciones (o coordinador) es la de garantizar la coherencia de las copias replicadas de un elemento de datos mediante el empleo distribuido (o global) técnicas de control de concurrencia. El servidor de aplicaciones también debe garantizar la atomicidad de las transacciones globales realizando una recuperación global cuando los sitios fallan.

Si el DDBMS tiene la capacidad de ocultar los detalles de la distribución de datos del servidor de aplicaciones, habilita al servidor de aplicaciones para ejecutar consultas globales y transacciones como si la base de datos estuviera centralizada, sin tener que especificar

Página 35

los sitios en los que residen los datos a los que se hace referencia en la consulta o transacción. Esta La propiedad se llama **transparencia de distribución** . Algunos DDBMS no proporcionan distribución transparencia global, requiriendo que las aplicaciones conozcan los detalles de distribución de datos.

23.8 Gestión de catálogos distribuidos

La gestión eficiente del catálogo en bases de datos distribuidas es fundamental para garantizar rendimiento histórico relacionado con la autonomía del sitio, la gestión de vistas y la distribución de datos ción y replicación. Los catálogos son bases de datos que contienen metadatos sobre el sistema de base de datos distribuida.

Tres esquemas de gestión populares para catálogos distribuidos son cata- registros, catálogos completamente replicados y catálogos particionados. La elección del esquema depende de la base de datos en sí, así como de los patrones de acceso de las aplicaciones a los datos subyacentes.

Catálogos centralizados. En este esquema, el catálogo completo se almacena en un solo sitio. Debido a su naturaleza central, es fácil de implementar. Por otro lado, el ventajas de confiabilidad, disponibilidad, autonomía y distribución del procesamiento la carga se ve afectada negativamente. Para operaciones de lectura de sitios no centrales, el Los datos solicitados del catálogo se bloquean en el sitio central y luego se envían al sitio solicitante. Al finalizar la operación de lectura, se recibe un acuse de recibo. enviado al sitio central, que a su vez desbloquea estos datos. Todas las operaciones de actualización debe procesarse a través del sitio central. Esto puede convertirse rápidamente en un mance cuello de botella para aplicaciones de escritura intensiva.

Catálogos completamente replicados. En este esquema, copias idénticas del completo catálogo están presentes en cada sitio. Este esquema facilita lecturas más rápidas al permitir para que sean respondidas localmente. Sin embargo, todas las actualizaciones deben transmitirse a todos sitios. Las actualizaciones se tratan como transacciones y un compromiso centralizado de dos fases se emplea un esquema para garantizar la coherencia del catálogo. Al igual que con el centralizado esquema, las aplicaciones de escritura intensiva pueden causar un aumento del tráfico de red debido a la transmisión asociada con las escrituras.

Catálogos parcialmente replicados. Los esquemas centralizados y completamente replicados restringir la autonomía del sitio, ya que deben garantizar una visión global coherente de la catalogar. Bajo el esquema parcialmente replicado, cada sitio mantiene completo catalogar información sobre datos almacenados localmente en ese sitio. Cada sitio también está permitido ted para almacenar en caché las entradas recuperadas de sitios remotos. Sin embargo, no existen garantías tees que estas copias en caché serán las más recientes y actualizadas. El sistema rastrea las entradas del catálogo para los sitios donde se creó el objeto y para los sitios que contener copias de este objeto. Cualquier cambio en las copias se propaga de inmediato. inmediatamente al lugar original (nacimiento). Recuperar copias actualizadas para reemplazar datos obsoletos puede retrasarse hasta que se produzca un acceso a estos datos. En general, fragmentos de relaciones Las conexiones entre sitios deben ser de acceso exclusivo. Además, para garantizar la distribución de datos transparencia, los usuarios deben poder crear sinónimos para objetos y utilice estos sinónimos para referencias posteriores.

Página 36

23.9 Resumen

En este capítulo, proporcionamos una introducción a las bases de datos distribuidas. Esto es muy tema amplio, y discutimos sólo algunas de las técnicas básicas que se utilizan con la distribución bases de datos uted. Primero, en la Sección 23.1 discutimos las razones de la distribución y Conceptos de DDB en la Sección 23.1.1. Luego definimos el concepto de distribución trans- parency y los conceptos relacionados de fragmentación, transparencia y replicación transparencia en la Sección 23.1.2. Discutimos los conceptos de disponibilidad distribuida y confiabilidad en la Sección 23.1.3, y brindó una descripción general de escalabilidad y partición cuestiones de tolerancia en la Sección 23.1.4. Discutimos la autonomía de los nodos en un distribuido sistema en la Sección 23.1.5 y las ventajas potenciales de las bases de datos distribuidas sobre sistema centralizado en la Sección 23.1.6.

En la Sección 23.2, discutimos los problemas de diseño relacionados con la fragmentación de datos, replicación y distribución. Distinguimos entre fragmentaciones horizontales ción (fragmentación) y fragmentación vertical de las relaciones en la Sección 23.2.1. Nosotros luego discutido en la Sección 23.2.2 el uso de la replicación de datos para mejorar el sistema fiabilidad y disponibilidad. En la Sección 23.3, discutimos brevemente la concurrencia técnicas de recuperación y control de frecuencia utilizadas en DDBMS, y luego revisadas algunos de los problemas adicionales que deben tratarse en un entorno distribuido entornos que no aparecen en un entorno centralizado. Luego en la Sección 23.4 discutimos la gestión de transacciones, incluidos diferentes protocolos de compromiso (Compromiso de 2 fases, compromiso de 3 fases) y soporte del sistema operativo para transacciones gestión de la información.

Luego ilustramos algunas de las técnicas utilizadas en el procesamiento de consultas distribuidas en Sección 23.5, y discutió el costo de la comunicación entre sitios, que consideró un factor importante en la optimización de consultas distribuidas. Comparamos las diferencias ent técnicas para ejecutar uniones, y luego presentamos la técnica de semiunión para unir relaciones que residen en diferentes sitios en la Sección 23.5.3.

A continuación, en la Sección 23.6, categorizamos los DDBMS utilizando criterios como el grado de homogeneidad de los módulos de software y el grado de autonomía local. En la Sección 23.7 distinguimos entre arquitectura de sistema paralelo y distribuida tures y luego introdujo la arquitectura genérica de bases de datos distribuidas de tanto un componente como una perspectiva arquitectónica esquemática. En la sección 23.7.3 discutimos con cierto detalle los problemas de la gestión de bases de datos federadas, y centrado en las necesidades de sustentar diversos tipos de autonomías y afrontar heterogeneidad semántica. También revisamos los conceptos de arquitectura cliente / servidor. y los relacionó con bases de datos distribuidas en la Sección 23.7.4. Revisamos el catálogo gestión en bases de datos distribuidas y resumió sus ventajas relativas y desventajas en la Sección 23.8.

Los capítulos 24 y 25 describirán los avances recientes en bases de datos distribuidas y Computación tributaria relacionada con big data. El capítulo 24 describe el llamado NOSQL sistemas, que son sistemas de bases de datos distribuidas altamente escalables que manejan grandes volúmenes de datos. El capítulo 25 analiza la computación en la nube y la computación distribuida tecnologías necesarias para procesar macrodatos.

Página 37

Preguntas de revisión

23.1. ¿Cuáles son las principales razones y las posibles ventajas de los bases de datos?

23.2. ¿Qué funciones adicionales tiene un DDBMS sobre un DBMS centralizado?

23.3. Analice qué se entiende por los siguientes términos: grado de homogeneidad de un DDBMS, grado de autonomía local de un DDBMS, DBMS federado, distribución transparencia, transparencia de fragmentación, transparencia de replicación, sistema de bases de datos múltiples.

23.4. Analice la arquitectura de un DDBMS. En el contexto de una centralizada DBMS, explique brevemente los nuevos componentes introducidos por la distribución de datos.

23,5. ¿Cuáles son los principales módulos de software de un DDBMS? Discuta los principales funciones de cada uno de estos módulos en el contexto del cliente / servidor arquitectura.

23,6. Compare las arquitecturas cliente / servidor de dos y tres niveles.

23,7. ¿Qué es un fragmento de una relación? ¿Cuáles son los principales tipos de fragmentos? ¿Por qué la fragmentación es un concepto útil en el diseño de bases de datos distribuidas?

23,8. ¿Por qué es útil la replicación de datos en DDBMS? Qué unidades típicas de datos son replicado?

23,9. ¿Qué se entiende por asignación de datos en el diseño de bases de datos distribuidas? ¿Qué tipo- las unidades cal de datos se distribuyen entre sitios?

23.10. ¿Cómo se especifica una partición horizontal de una relación? ¿Cómo puede una relación volver a armar a partir de una división horizontal completa?

23.11. ¿Cómo se especifica una partición vertical de una relación? ¿Cómo puede ser una relación volver a armar a partir de una división vertical completa?

23.12. Analice el problema de los nombres en las bases de datos distribuidas.

23.13. ¿Cuáles son las diferentes etapas del procesamiento de una consulta en un DDBMS?

23.14. Analice las diferentes técnicas para ejecutar una combinación de dos archivos ubicados en diferentes sitios. ¿Qué factores principales afectan el costo de la transferencia de datos?

23.15. Analice el método de semiunión para ejecutar una combinación de dos archivos ubicados en diferentes sitios. ¿En qué condiciones es eficaz una estrategia de equijoin?

23.16. Analice los factores que afectan la descomposición de consultas. ¿Cómo están las condiciones de guardia listas de atributos y atributos de fragmentos utilizados durante la descomposición de consultas ¿proceso?

23.17. ¿En qué se diferencia la descomposición de una solicitud de actualización de la descomposición
sition de una consulta? ¿Cómo son las condiciones de protección y las listas de atributos de fragmentos?
utilizado durante la descomposición de una solicitud de actualización?

Página 38

23.18. Enumere el soporte ofrecido por los sistemas operativos a un DDBMS y también las ventajas
efectos de estos apoyos.

23.19. Analice los factores que no aparecen en los sistemas centralizados pero que afectan
control de concurrencia y recuperación en sistemas distribuidos.

23.20. Analizar el protocolo de compromiso de dos fases utilizado para la gestión de transacciones.
en un DDBMS. Enumere sus limitaciones y explique cómo se superan utilizando
el protocolo de compromiso de tres fases.

23.21. Compare el método del sitio principal con el método de copia principal para
control de concurrencia tributado. ¿Cómo afecta el uso de sitios de respaldo a cada uno?

23.22. ¿Cuándo se utilizan las votaciones y las elecciones en las bases de datos distribuidas?

23.23. Analice la gestión de catálogos en bases de datos distribuidas.

23.24. ¿Cuáles son los principales desafíos que enfrenta un DDBMS tradicional en el contexto de
aplicaciones de Internet de hoy? ¿Cómo intenta la computación en la nube abordar
¿ellos?

23.25. Discuta brevemente el soporte ofrecido por Oracle para homogéneos, heterogéneos
arquitecturas de bases de datos distribuidas basadas en cliente / servidor.

23.26. Analice brevemente los directorios en línea, su gestión y su función en la distribución.
bases de datos tributadas.

Ejercicios

23.27. Considere la distribución de datos de la base de datos de la COMPAÑÍA, donde la
Los fragmentos en los sitios 2 y 3 son como se muestra en la Figura 23.3 y los fragmentos en el sitio 1
son como se muestra en la Figura 3.6. Para cada una de las siguientes consultas, muestre al menos
dos estrategias de descomposición y ejecución de la consulta. ¿En qué condiciones
¿Funcionaría bien cada una de sus estrategias?

a. Para cada empleado del departamento 5, recupere el nombre del empleado y el
nombres de los dependientes del empleado.

segundo. Escriba los nombres de todos los empleados que trabajan en el departamento 5 pero que
trabajar en algún proyecto no controlado por el departamento 5.

23.28. Considere las siguientes relaciones:

LIBROS (N.º de libro, Autor_primario, Tema, Stock_total, $ precio)
LIBRERÍA (número de tienda, ciudad, estado, código postal, Inventory_value)
STOCK (número de tienda, número de libro, cantidad)

Total_stock es el número total de libros en stock, y Inventory_value es el valor total del inventario de la tienda en dólares.

a. Dé un ejemplo de dos predicados simples que serían significativos para la relación BOOKSTORE para la división horizontal.

Página 39

segundo. ¿Cómo se definiría una partición horizontal derivada de STOCK basándose en la partición de BOOKSTORE ?

C. Muestre predicados mediante los cuales los LIBROS pueden dividirse horizontalmente por tema.

re. Muestre cómo el STOCK puede dividirse aún más de las particiones en (b) agregando los predicados en (c).

23.29. Considere una base de datos distribuida para una cadena de librerías llamada National Books con tres sitios llamados EAST, MIDDLE y WEST . Los esquemas de relación son dado en el ejercicio 23.28. Considere que los LIBROS están fragmentados por $ precio asciende a:

B 1 : BOOK1: $ precio hasta $ 20
B 2 : BOOK2: $ precio de $ 20.01 a $ 50
B 3 : BOOK3: $ precio de $ 50.01 a $ 100
B 4 : BOOK4: $ precio $ 100.01 y superior

De manera similar, BOOK_STORES se dividen por códigos postales en:

S 1 : ESTE: Zip hasta 35000
S 2 : MEDIO: Cremallera 35001 a 70000
S 3 : OESTE: Zip 70001 a 99999

Suponga que STOCK es un fragmento derivado basado únicamente en BOOKSTORE .

a. Considere la consulta:

```
SELECCIONAR Libro #, Total_stock
DESDE       Libros
DONDE $ precio> 15 Y $ precio <55;
```

Suponga que los fragmentos de BOOKSTORE no están replicados y asignados basado en la región. Suponga además que los LIBROS se asignan como:

ESTE: B 1 , B 4
MEDIO: B 1 , B 2
OESTE: B 1 , B 2 , B 3 , B 4

Suponiendo que la consulta se envió en EAST , ¿qué subconsultas remotas hace?

¿generar? (Escriba en SQL).
segundo. Si el precio del Libro # = 1234 se actualiza de $ 45 a $ 55 en el sitio MIDDLE , ¿Qué actualizaciones genera eso? Escriba en inglés y luego en SQL.

C. Proporcione una consulta de muestra emitida en WEST que generará una subconsulta para MEDIO .

re. Escriba una consulta que implique selección y proyección sobre la relación anterior. y muestran dos posibles árboles de consulta que denotan diferentes formas de ejecución.

23,70. Considere que se le ha pedido que proponga una arquitectura de base de datos en un gran organización (General Motors, por ejemplo) para consolidar todos los datos

Página 40

incluidas bases de datos heredadas (de modelos jerárquicos y de red; sin es necesario un conocimiento específico de estos modelos), así como bases de datos relacionales, que están distribuidos geográficamente para que las aplicaciones globales puedan ser compatibles portado. Suponga que la alternativa 1 es mantener todas las bases de datos como están, mientras que La alternativa 2 es convertirlos primero en relacionales y luego apoyar la aplicación cationes sobre una base de datos integrada distribuida.

a. Dibuje dos diagramas esquemáticos para las alternativas anteriores que muestren vínculos entre esquemas apropiados. Para la alternativa 1, elija el enfoque de proporcionar esquemas de exportación para cada base de datos y construcción ing esquemas unificados para cada aplicación.

segundo. Enumere los pasos que tendría que seguir en cada alternativa desde la situación actual hasta que las aplicaciones globales sean viables.

C. Compare estas alternativas de las cuestiones de:
yo. consideraciones de tiempo de diseño
ii. consideraciones de tiempo de ejecución

Bibliografía seleccionada

Los libros de texto de Ceri y Pelagatti (1984a) y Ozsu y Valduriez (1999) son dedicado a bases de datos distribuidas. Peterson y Davie (2008), Tannenbaum (2003), y Stallings (2007) cubren comunicaciones de datos y redes informáticas. Contendiente (2008) analiza las redes e internets. Ozsu y col. (1994) tiene una colección de artículos sobre la gestión de objetos distribuidos.

La mayor parte de la investigación sobre diseño de bases de datos distribuidas, procesamiento de consultas y optimización la nacionalización ocurrió en las décadas de 1980 y 1990; revisamos rápidamente las referencias importantes aquí. El diseño de bases de datos distribuidas se ha abordado en términos de horizontal y fragmentación, asignación y replicación vertical. Ceri y col. (1982) definió el concepto de fragmentos horizontales de minitérmino. Ceri y col. (1983) desarrolló un número entero modelo de optimización basado en programación para la fragmentación horizontal y la asignación

ción. Navathe y col. (1984) desarrollaron algoritmos para la fragmentación vertical basados en la afinidad de atributos y mostró una variedad de contextos para la asignación de fragmentos verticales ción. Wilson y Navathe (1986) presentan un modelo analítico para la asignación óptima de fragmentos. Elmasri y col. (1987) discuten la fragmentación del modelo ECR; Karla- palem y col. (1996) discuten cuestiones para el diseño distribuido de bases de datos de objetos. Navathe et al. (1996) discuten la fragmentación mixta combinando horizontales y verticales fragmentación; Karlapalem y col. (1996) presentan un modelo de rediseño de sistemas distribuidos bases de datos.

El procesamiento, la optimización y la descomposición de consultas distribuidas se tratan en Hevner y Yao (1979), Kerschberg et al. (1982), Apers et al. (1983), Ceri y Pela- gatti (1984) y Bodorick et al. (1992). Bernstein y Goodman (1981) discuten el teoría detrás del procesamiento de semiunión. Wong (1983) analiza el uso de relaciones en relación a la fragmentación. Se discuten los esquemas de recuperación y control de concurrencia en Bernstein y Goodman (1981a). Kumar y Hsu (1998) compilan algunos artículos

Página 41

relacionados con la recuperación en bases de datos distribuidas. Las elecciones en sistemas distribuidos son discutido en García-Molina (1982). Lamport (1978) discute problemas con creación de marcas de tiempo únicas en un sistema distribuido. Rahimi y Haug (2007) discuten un forma más flexible de construir consultas de metadatos críticos para bases de datos P2P. Ouzzani y Bouguettaya (2004) describen problemas fundamentales en el pro- pasando a través de fuentes de datos basadas en la Web.

Una técnica de control de concurrencia para datos replicados que se basa en la votación se presentado por Thomas (1979). Gifford (1979) propone el uso del voto ponderado, y Paris (1986) describe un método llamado votación con testigos. Jajodia y Mutchler (1990) discuten el voto dinámico. Una técnica llamada copia disponible es propuesta por Bernstein y Goodman (1984), y se presenta uno que utiliza la idea de grupo en ElAbbadi y Toueg (1988). Otro trabajo que analiza los datos replicados incluye Gladney (1989), Agrawal y ElAbbadi (1990), ElAbbadi y Toueg (1989), Kumar y Segev (1993), Mukkamala (1989) y Wolfson y Milo (1991). Bassiouni (1988) analiza los protocolos optimistas para el control de concurrencia DDB. García-Molina (1983) y Kumar y Stonebraker (1987) discuten técnicas que utilizan el método semanal tics de las transacciones. Técnicas de control de simultaneidad distribuidas basadas en bloqueo Menasce et al. (1980) y Minoura y Wiederhold (1982). Obermark (1982) presenta algoritmos para distribuciones detección de interbloqueo. En un trabajo más reciente, Vadivelu et al. (2008) proponen utilizar mecanismo de respaldo y seguridad multinivel para desarrollar algoritmos para mejorar concurrencia. Madria y col. (2007) proponen un mecanismo basado en una multiversión esquema de bloqueo de dos fases y marca de tiempo para abordar problemas de concurrencia específicos a los sistemas de bases de datos móviles. Boukerche y Tuck (2001) proponen una técnica que permite que las transacciones estén fuera de orden hasta cierto punto. Intentan aliviar el cargar en el desarrollador de aplicaciones explotando el entorno de red y produciendo un horario equivalente a un horario serial ordenado temporalmente. Han y col.

(2004) proponen un modelo de red de Petri extendido serializable y libre de interbloqueo para Web-
Basadas en bases de datos distribuidas en tiempo real.

Kohler (1981) ofrece un estudio de las técnicas de recuperación en sistemas distribuidos.
Reed (1983) analiza las acciones atómicas sobre datos distribuidos. Bhargava (1987) presenta
una compilación editada de varios enfoques y técnicas para la concurrencia y
confiabilidad en sistemas distribuidos.

Los sistemas de bases de datos federadas se definieron por primera vez en McLeod y Heimbigner (1985).
Elmasri presenta técnicas para la integración de esquemas en bases de datos federadas
et al. (1986), Batini et al. (1987), Hayne y Ram (1990) y Motro (1987).
Elmagarmid y Helal (1988) y Gamal-Eldin et al. (1988) discuten la actualización
problema en DDBS heterogéneos. Los problemas de bases de datos distribuidas heterogéneas son
discutido en Hsiao y Kamel (1989). Sheth y Larson (1990) presentan un resumen
estudio dinámico de la gestión de bases de datos federadas.

Desde finales de la década de 1980, los sistemas de bases de datos múltiples y la interoperabilidad se han convertido en
temas importantes. Técnicas para tratar las incompatibilidades semánticas entre
Se examinan múltiples bases de datos en DeMichiel (1989), Siegel y Madnick (1991),
Krishnamurthy y col. (1991) y Wang y Madnick (1989). Castano y col. (1998)

Página 42

presentar un excelente repaso de técnicas para el análisis de esquemas. Pitoura y col.
(1995) analizan la orientación a objetos en sistemas de bases de datos múltiples. Xiao y col. (2003) pro
plantear un modelo basado en XML para un modelo de datos común para sistemas de bases de datos múltiples
y presentar un nuevo enfoque para el mapeo de esquemas basado en este modelo. Lakshmanan
et al. (2001) proponen extender SQL para la interoperabilidad y describen la arquitectura
tura y algoritmos para lograr lo mismo.

El procesamiento de transacciones en bases de datos múltiples se analiza en Mehrotra et al. (1992),
Georgakopoulos y col. (1991), Elmagarmid et al. (1990) y Brietbart et al.
(1990), entre otros. Elmagarmid (1992) analiza el procesamiento de transacciones para
aplicaciones avanzadas, incluidas las aplicaciones de ingeniería que se tratan en
Heiler y col. (1992).

Los sistemas de flujo de trabajo, que se están volviendo populares para administrar información en
organizaciones complejas, utilizan transacciones multinivel y anidadas junto con
bases de datos distribuidas. Weikum (1991) analiza la gestión de transacciones multinivel
ment. Alonso y col. (1997) discuten las limitaciones de los sistemas de flujo de trabajo actuales. Lopes et
Alabama. (2009) proponen que los usuarios definan y ejecuten sus propios flujos de trabajo utilizando un cliente
navegador web lateral. Intentan aprovechar las tendencias de la Web 2.0 para simplificar la
trabajar para la gestión del flujo de trabajo. Jung y Yeom (2008) explotan el flujo de trabajo de datos para
desarrollar un sistema de gestión de transacciones mejorado que proporcione
acceso transparente a los almacenamientos heterogéneos que constituyen el HVEM
Cuadrícula de datos. Deelman y Chervanak (2008) enumeran los desafíos en la ciencia intensiva en datos
Flujos de trabajo interesantes. Específicamente, miran a la gestión automatizada de datos,
técnicas de mapeo cient y problemas de retroalimentación de los usuarios en mapeo de flujo de trabajo Ellos

también abogan por la reutilización de datos como un medio eficiente para administrar datos y presentar el desafío lenges en el mismo.

Se han implementado varios DBMS distribuidos experimentales. Estas incluyen INGRES distribuidos por Epstein et al. (1978), DDTS de Devor y Weeldreyer (1980), SDD-1 de Rothnie et al. (1980), System R * de Lindsay et al. (1984), SIRIUS-DELTA de Ferrier y Stangret (1982) y MULTIBASE de Smith et al. (1981). El sistema OMNIBASE de Rusinkiewicz et al. (1988) y la Federación Base de información desarrollada utilizando el modelo de datos Candide por Navathe et al. (1994) son ejemplos de DDBMS federados. Pitoura y col. (1995) presentan un comparativo estudio de los prototipos del sistema de bases de datos federadas. La mayoría de los DBMS comerciales Los dors tienen productos que utilizan el enfoque cliente / servidor y ofrecen versiones distribuidas. de sus sistemas. Algunos problemas del sistema relacionados con las arquitecturas DBMS cliente / servidor se discuten en Carey et al. (1991), DeWitt et al. (1990) y Wang y Rowe (1991). Khoshafian y col. (1992) discuten cuestiones de diseño para DBMS relacionales en el entorno cliente / servidor. Los problemas de administración de cliente / servidor se discuten en muchos libros, como Zantinge y Adriaans (1996). Di Stefano (2005) analiza la distribución de datos problemas de distribución específicos de la computación en red. Una parte importante de esta discusión también puede aplicar a la computación en la nube.