Bases de Datos Distribuídas

20/Octubre/2017

DDB - Definiciones

Base de Datos Distribuída (DDB)

Colección de múltiples BD que están lógicamente relacionadas y se encuentran distribuídas en una red de computadoras

Sistema de Gestión de DDB (DDBMS)

Software que permite gestionar la DDB y que hace transparente la distribución al usuario

Condiciones que debe satisfacer una DDB

- Interconexión entre BDs a través de una red de computadoras
- Relación lógica entre las BDs interconectadas
- Posible heterogeneidad entre los nodos

DDB - Queries - Ejemplo 1

Site 1:

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

10,000 records
each record is 100 bytes long
Ssn field is 9 bytes long — Fname field is 15 bytes long
Dno field is 4 bytes long — Lname field is 15 bytes long

Site 2:

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

100 records
each record is 35 bytes long
Dnumber field is 4 bytes long — Dname field is 10 bytes long
Mgr_ssn field is 9 bytes long

Figure 23.4
Example to illustrate volume of data transferred.

Figura de Elmasri/Navathe-Fundamentals of DB Systems, 7th Ed., Pearson, 2015

DDB - Queries - Ejemplo 1

- ¿Cuánto espacio ocupa cada relación?
  EMPLOYEE: 10.000 registros * 100 bytes/registro = 10^6 bytes
  DEPARTMENT: 100 registros * 35 bytes/registro = 3.500 bytes
- Traducir la siguiente consulta Q al álgebra relacional: *"Para cada empleado, obtener su nombre y el nombre de departamento donde trabaja"*
  $Q : \pi_{Fname, Lname, Dname}(EMPLOYEE \bowtie_{Dno=Dnumber} DEPARTMENT)$
- ¿Cuántos registros retorna la consulta Q? (asumiendo que todo empleado se relaciona con un departamento)
  10.000 registros
- ¿Cuánto espacio ocupa cada registro de Q?
  15 bytes (Fname) + 15 bytes (Lname) + 10 bytes (Dname) = 40 bytes

DDB - Queries - Ejemplo 1 (Cont.)

- Si Q se ejecuta en Sitio 3 (EMPLOYEE se encuentra en Sitio 1 y DEPARTMENT en Sitio 2), mencionar 3 posibles estrategias para ejecutar Q y calcular costo (cantidad de bytes a transferir en cada caso)
  1. Transferir (EMPLOYEE y DEPARTMENT) → Sitio 3
     Ejecutar JOIN en Sitio 3
     Costo: 1.000.000 + 3.500 = **1.003.500 bytes deberán ser transferidos**
  2. Transferir EMPLOYEE → Sitio 2
     Ejecutar JOIN en Sitio 2
     Transferir resultado a Sitio 3
     Costo: 1.000.000 (Sitio 1 → Sitio 2) + 400.000 (Site 2 → Site 3)
     = **1.400.000 bytes deberán ser transferidos**
     (Tamaño de Site 2 → Site 3 = 40 bytes/registro * 10.000 registros = 400.000 bytes)
  3. Transferir DEPARTMENT → Sitio 1
     Ejecutar JOIN en Sitio 1
     Transferir resultado a Sitio 3
     Costo: 3.500 (Sitio 2 → Sitio 1) + 400.000 bytes (Site 1 → Site 3)
     = **403.500 bytes deberán ser transferidos**
     (Tamaño de Site 1 → Site 3 = 40 bytes/registro * 10.000 registros = 400.000 bytes)

DDB - Interconexión

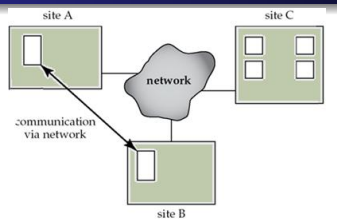


Figura de Elmasri/Navathe-Fundamentals of DB Systems, 7th Ed., Pearson, 2015

- **Tipo.** LAN, WAN, etc.
- **Topología.** "Caminos" de comunicación entre sitios (directa, indirecta, etc.)
- **Tipo** y **Topología** impactan en
  - Estrategia de Queries distribuídos
  - Diseño de DDB
- **Red.** Detalles subyacentes de red invisibles para usuario final
- **Conexo.** Asumimos que existe interconexión entre todos los nodos (sin importar topología subyacente). Sin embargo es importante tener en cuenta este tipo de información al momento de implementar un sistema de DDB

DDB - Transparencia

Idea

Ocultar detalles de implementación a los usuarios finales

- **BD Centralizadas.** Independencia entre datos físicos y lógicos
- **BD Distribuídas.** Datos y Software están distribuídos, entonces se agregan nuevos tipos de transparencia
  1. Organización de los datos
  2. Fragmentación
  3. Replicación
  4. Otras

DDB - Transparencia (Cont.)

1. **Organización de Datos**
   - También conocida como de distribución o de red
   - Libera al usuario de conocer detalles operativos de la red y ubicación de los datos
   - Dividir en ...
     - **Transparencia de Ubicación.** El comando utilizado para ejecutar una tarea guarda independencia con respecto a la ubicación de los datos y dónde se lanzó
     - **Transparencia de Nombres.** Una vez asociado el nombre con un objeto, puede ser accedido de manera unívoca sin datos adicionales, como ser la ubicación

DDB - Transparencia (Cont.)

2. **Fragmentación**
   - Libera al usuario de conocer detalles sobre la fragmentación de los datos
   - Dos tipos ...
     - **Horizontal (o sharding)**: Distribuye la relación (tabla) en subrelaciones que son subconjuntos de las tuplas (filas) de la relación original
     - **Vertical**: Distribuye la relación en subrelaciones donde cada subrelación es definida por un subconjunto de columnas de la relación original
   - En ambos casos el usuario desconoce la existencia de fragmentos
3. **Replicación**
   - Objetos pueden ser almacenados en varios sitios simultáneamente para mejorar la disponibilidad, performance y confiabilidad. El usuario desconoce la existencia de las copias
4. **Otras Transparencias**
   - **Diseño** y **Ejecución**, libera al usuario de conocer cómo está diseñada la DDB y dónde es ejecutada una transacción

DDB - Disponibilidad y Confiabilidad

- **Disponibilidad.** Probabilidad de que un sistema se encuentre continuamente disponible en un intervalo de tiempo
- **Confiabilidad.** Probabilidad de que un sistema se encuentre operando en un momento determinado del tiempo
- Fallos y errores asociados a la Confiabilidad y Disponibilidad ...
  - **Failure** (Fallo). Desviación del comportamiento esperado del sistema
  - **Errores**. Subconjunto de estados del sistema que causan failures
  - **Fault** (Fallo). Causa de un error
- Para construir un sistema confiable ...
  - **Stresses fault tolerance.** Reconocer que los fallos van a ocurrir y diseñar mecanismos que puedan detectar y remover fallos previo a que ocurran
  - **Alternativa.** Asegurar que el sistema final no contenga faults. Se hace siguiendo procesos de desarrollo que incluyen control de calidad y testing
- Un DDBMS debe tolerar fallos de componentes subyacentes, sin alterar el procesamiento de pedidos de usuarios ni la consistencia de de la DB
- **Recovery Manager** tiene que tratar con los fallos de las transacciones, del hardware y de las comunicaciones en las redes
- **"Disponibilidad".** En DDB, se suele usar indistintamente para **Confiabilidad** y **Disponibilidad**

DDB - Escalabilidad y Tolerancia a Partición

- **Escalabilidad.** Determina la medida en que el sistema puede expandir su capacidad mientras continúa operando sin interrupción
- Clasificación
  - **Horizontal.** Expandir la cantidad de nodos del sistema distribuído
  - **Vertical.** Expandir la capacidad de un nodo individual del sistema
- **Tolerancia a Partición.** El sistema debería tener la capacidad de continuar operando a pesar de que la red sea particionada

DDB - Autonomía

- **Autonomía.** Determina en qué medida nodos individuales (o BDs) de una DDB pueden operar independientemente
- En principio es deseable un alto grado de autonomía
- Tipos de autonomía
  - **Diseño.** Independencia del uso del modelo de datos / técnicas de gestión de transacciones entre los nodos
  - **Comunicación.** Determina en qué medida los nodos pueden decidir compartir o no información con otros nodos
  - **Ejecución.** Independencia de que los usuarios puedan realizar "acciones a su gusto"

DDB - Beneficios

- **Mejora de manera sencilla y flexible el desarrollo de aplicaciones.** Desarrollar y mantener aplicaciones en sitios geográficamente distribuídos se ve facilitado debido a la transparencia en el control y la distribución de los datos
- **Aumento de la disponibilidad.** Lo consigue aislando los fallos a su sitio de origen, sin afectar a las BDs de otros nodos
- **Mejora en la performance.** Un DDBMS fragmenta la BD manteniendo los datos cerca de donde más se necesita
- **Facilita la expansión vía escalabilidad**: En esquemas distribuídos, expandir el sistema en términos de agregar más datos, incrementar el tamaño del dataset, o agregar más nodos puede llegar a ser más sencillo que en un esquema centralizado

Transparencia

- **Transparencia Total** provee al usuario de una visión del DDBS como si fuera un sistema centralizado.
- Existe un compromiso entre facilidad en el uso y el sobrecosto de proveer transparencia

DDB - Introducción

- **Fragmentos.** Partes que resultan de dividir la BD en unidades lógicas
- **Asignación de fragmentos.** Permite almacenarlos en distintos nodos
- **Replicación de los datos.** Permite a ciertos datos ser almacenados en más de un sitio, para incrementar la disponibilidad y confiabilidad
- **Diseño.** Estas técnicas son tenidas en cuenta durante el proceso de diseño de la DDB
- **Catálogo Global.** La información acerca de la fragmentación, asignación de fragmentos y replicación de los datos es almacenada en un catálogo global

DDB - Fragmentación Horizontal (o sharding)

Distribuye la relación (tabla) en ... subrelaciones que son subconjuntos de las tuplas (filas) de la relación original

Formalización

- En álgebra relacional: $\sigma_{C_i}(R)$ (selección)
- Si las condiciones $C_1, C_2, ..., C_n$ incluyen a todas las tuplas de R (todas las tuplas satisfacen C_1 OR C_2 OR ... OR C_n), se denomina *fragmentación horizontal completa* de R
- En varios casos, este tipo de fragmentación también es disjunta. Es decir, ninguna tupla en R satisface (C_i AND C_j) para $i \neq j$
- Para reconstruir la relación de una fragmentación horizontal completa, se debe aplicar la operación UNION a todos los fragmentos

DDB - Fragmentación Vertical

Distribuye la relación (tabla) en ... subrelaciones donde cada una de ellas es definida por un subconjunto de columnas de la relación original

Formalización

- En álgebra relacional: $\pi_{L_i}(R)$ (proyección)
- Si las listas de proyecciones $L_1, L_2, ..., L_n$ incluyen todos los atributos de R ($ATTRS(R)$), compartiendo sólo los atributos correspondientes a la clave primaria ($PK(R)$), se denomina *fragmentación vertical completa* de R
- En este caso, la lista de proyecciones satisface a la vez:
  - $L_1 \cup L_2 \cup ... \cup L_n = ATTRS(R)$
  - $L_i \cap L_j = PK(R)$ para $i \neq j$
- Para reconstruir la relación de una fragmentación vertical completa, se debe aplicar la operación OUTER UNION a todos los fragmentos (asumiendo que no hay fragmentación horizontal)
- Observar que también es posible aplicar FULL OUTER JOIN, incluso cuando hay fragmentación horizontal

DDB - Fragmentación Mixta (o Híbrida)

Combina la fragmentación horizontal y vertical

Formalización

- En álgebra relacional: $\pi_L(\sigma_C(R))$ (proyección de una selección).
  - Si $C = TRUE$ (todas las tuplas son seleccionadas) y $L \neq ATTRS(R)$, entonces se trata de una fragmentación vertical
  - Si $C \neq TRUE$ y $L = ATTRS(R)$, entonces se trata de una fragmentación horizontal
  - Si $C \neq TRUE$ y $L \neq ATTRS(R)$, entonces se trata de una fragmentación mixta
  - Si $C = TRUE$ y $L = ATTRS(R)$, entonces no hay fragmentación
- Para reconstruir la relación de una fragmentación mixta, se deben aplicar las operaciones UNION y OUTER UNION (ó OUTER JOIN) en el orden apropiado

DDB - Fragmentación - Esquemas

- Un **Esquema de Fragmentación** de una BD es la definición de un conjunto de fragmentos que:
  - Incluye a TODOS los atributos y tuplas de la BD
  - La totalidad de la BD puede ser reconstruída a partir de los fragmentos aplicando una secuencia de operaciones (OUTER UNION o OUTER JOIN + UNION)
- A veces es útil (pero no necesario) que los fragmentos sean disjuntos (excepto por la PK)
- Un **Esquema de Asignación** describe la ubicación de los fragmentos en los nodos (sites) de la DDB
- Si el fragmento es almacenado en en más de un lugar, se dice que se encuentra **replicado**

DDB - Replicación Total

- **Objetivo.** Tener una mayor disponibilidad de los datos
- **Caso extremo.** Replicación TOTAL de la DDB
- **Ventajas.**
  - Sistema continúa operando mientra haya al menos un sitio disponible
  - Mejora la performance de las lecturas (pueden realizarse de manera local, mejora query, etc.)
- **Desventajas.**
  - Baja performance de las escrituras (tienen que realizarse en TODAS las copias)
  - Las técnicas de Control de Concurrencia y Recovery son más costosas

Pregunta

¿Qué sucede en el otro extremo, es decir, sin replicación?

DDB - Replicación Parcial

- Escenario entre ambos extremos
- Algunos fragmentos son replicados, mientras que otros no
- $1 \leq \#\text{copias} \leq |\text{sitios}|$
- Un **Esquema de Replicación** es una descripción de la replicación de los esquemas de la DDB
- **Distribución de Datos.** Cada fragmento (o copia del fragmento) debe ser asignado a un sitio particular. Este proceso de denomina distribución de datos
- La elección de los sitios y el grado de replicación depende de:
  - Las metas de disponibilidad del sistema
  - Las metas de performance del sistema
  - Tipo de transacciones realizadas en cada sitio

DDB - Control de Concurrencia (CC) y Recovery

Problemas adicionales en ambientes distribuídos

- Tratar con **múltiples copias** de un data item
  - Método de **CC** responsable de mantener **consistencia** entre las copias
  - Método de **Recovery** responsable de realizar copia consistente con otras copias si **falla el sitio**, y luego de restablecerlo
- Ante **fallas**, el DDBMS debería continuar operando.
  - **Sitio.** Cuando el sitio se recupera, su DB local debe estar actualizada con respecto al resto de los sitios, antes de unirse nuevamente
  - **Enlaces de comunicación.** Caso extremo: Partición de la red
- **Commit Distribuído.** Sitios pueden fallar durante el proceso de commit. Protocolo especial para tratar con este problema
- **Deadlock Distribuído.** Pueden ocurrir en varios sitios, por lo tanto debe tenerse en cuenta

DDB - CC

- **Consistencia.** Responsable de mantener consistencia entre las copias
- **Propuesta.** Extender técnicas de BD centralizadas
- **Locking**. Designar una copia de cada data item como copia distinguida. Pedidos de lock y unlock enviados al sitio de la copia distinguida
- **Variantes.**
  - Sitio Primario
  - Sitio Primario con sitio de backup
  - Copia Primaria

DDB - CC - Sitio Primario

- **Coordinador.** Todas las copias distinguidas se mantienen en un único sitio
- **Funcionamiento**
  - Si T obtiene *read_lock(X)* de sitio primario, puede acceder a cualquier copia de X
  - Si T obtiene *write_lock(X)* de sitio primario y actualiza X, el DDBMS es responsable de actualizar todas las copias de X antes de realizar *unlock(X)*
- **2PC.** Si todas las T siguen protocolo 2PC, la seriabilidad está garantizada
- **Ventaja.** Simple extensión del enfoque centralizado
- **Desventajas.**
  1. Todos los locks/unlock son enviados a un único sitio. Cuello de Botella
  2. Una falla en el Sitio Primario paraliza el sistema

DDB - CC - Sitio Primario + Backup

- **Backup.** Intenta solucionar Desventaja Nro. 2: Si falla Sitio Primario paraliza el sistema
- **Funcionamiento**
  - Información sobre locks se mantiene en ambos sitios (Primario y Backup)
  - En caso de falla de Sitio Primario, Sitio Backup toma el control
- **Ventaja** Simplifica proceso de Recovery ante falla en Sitio Primario
- **Desventajas**
  - Disminuye performance en procesos de lock (copia a Sitio Backup)
  - Persiste el problema de Cuello de Botella

DDB - CC - Copia Primaria

- **Copia Primaria.** Intenta solucionar Desventaja Nro. 1: Cuello de Botella
- **Funcionamiento.** Admite copias distinguidas en diferentes sitios
- **Ventaja Adicional.** Falla de un sitio sólo afecta a T con locks en data items cuya copia primaria se encuentran en dicho sitio. Las demás transacciones no se ven afectadas
- **Mejora.** Pueden agregarse sitios de Backup para mejorar la Confiabilidad y Disponibilidad

DDB - CC - Elección Nuevo Coordinador

- **Sitio Primario sin Backup.** Todas las T en ejecución deben ser abortadas y reiniciadas
- **Métodos con Backup.** Las T son suspendidas mientras el Sitio Backup es designado como nuevo Sitio Primario
- Si fallan en simultáneo Sitio Primario y Backup: Proceso de Elección
- Algoritmo de elección es complejo
- **Funcionamiento**
  - Si sitio Y intenta reiteradamente comunicarse con coordinador y no lo logra, asume que coordinador falló y puede iniciar proceso de elección
    - Envía a todos los nodos activos que Y va a ser el nuevo coordinador
    - Si Y recibe mayoría de votos afirmativos, Y puede se declara nuevo coordinador
- Algoritmo también puede resolver intentos de dos o más sitios intentando simultáneamente ser coordinadores

DDB - CC - Votación

- **Votación.** Idea alternativa a Copia Distinguida
- **Funcionamiento**
  - Se hace pedido de *lock(X)* a todos los sitios que contienen copia de X
  - Cada copia de X es reponsable de su propio lock y puede aceptar o denegar locks
  - Si T pide un *lock(X)*
    - Si recibe OK de mayoría de las copias, toma el lock y avisa a todas las copias que lo tomó
    - Si NO Recibe el ok de la mayoría de las copias dentro de un cierto tiempo, cancela su pedido e informa a todos los sitios de la cancelación
- **Ventaja.** Es considerado un método de CC realmente distribuído (decisión reside en los nodos)
- **Desventajas.**
  - Mayor tráfico de mensajes
  - Tener en cuenta la caída de sitios durante votación, torna al algoritmo muy complejo

Recovery - Problemas

1. **Problema.** Fallos en sitios y comunicaciones
   - En ciertos casos es muy difícil determinar cuándo un sitio se encuentra caído, sin intercambiar gran catidad de mensajes con otros sitios
   - **Ejemplo.** Sitio X envía un mensaje a sitio Y del cual espera una respuesta, pero ésta no es recibida.
   - Posibles causas.
     - Mensaje no llegó a Y debido a un corte en la comunicación
     - Sitio Y está caído y no responde
     - Sitio Y se encuentra funcionando y envía respuesta, pero no llega
   - En todos estos casos es complicado que X conozca la causa
2. **Problema.** Commit Distribuído
   - Si T actualiza información en varios sitios, no puede realizar el commit hasta asegurarse que los efectos de T no se van a perder en NINGÚN sitio (grabación de log)
   - Es frecuente usar 2PC para asegurar correctitud de Commit Distribuído

Recovery - 2PC

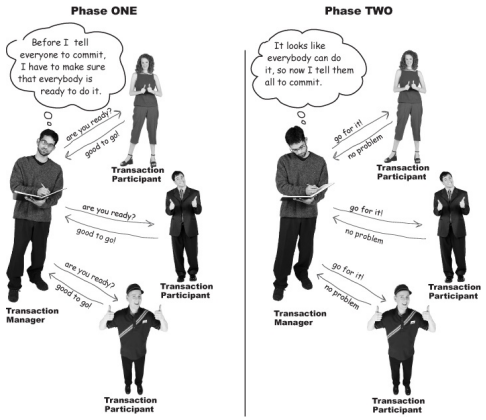


Figura de https://www.safaribooksonline.com/library/view/head-first-ejb/0596005717/ch09s02.html

DDB - Propiedades deseables en las transacciones

Propiedades ACID

- **Atomicity (Atomicidad):** Transacción como unidad atómica. Las operaciones de una transacción se ejecutan en su totalidad o no se ejecuta ninguna.
- **Consistency preservation (Preservación de Consistencia):** Si la transacción se ejecuta completamente sin interferencia de otra transacción, entonces mueve a la BD de un estado consistente a otro también consistente.
- **Isolation (Aislamiento):** La ejecución de una transacción no debe interferir con la de otra transacción que se ejecute de manera concurrente. Una transacción debe aparentar ser ejecutada como si lo hiciera de forma aislada a las otras. Incluso si varias de ellas son ejecutadas a la vez.
- **Durability (Durabilidad):** Los cambios aplicados a la BD por una transacción commiteada debe persistir en la BD. Estos cambios no se pueden perder a causa de ningún fallo.

DDB - Transacciones

- **Módulos.** Responsables de garantizar propiedades ACID
  - Gestor Global de Transacciones
  - Gestor Local de Transacciones
  - Gestor de CC
  - Gestor de Recovery
- **Gestión.**
  - Sitio origen de T, puede asumir rol de Gestor Global de Transacciones, coordinando ejecución en los múltiples sitios
  - Para cada T el gestor almacena: id único, sitio origen, etc.
- **Interfaces.**
  - BEGIN_TRANSACTION / END_TRANSACTION
  - READ(X). Retorna una copia de X, si es válida y está disponible
  - WRITE(X). Asegura que las actualizaciones sean visibles a través de todos los sitios que tienen copia de X
  - ABORT (o ROLLBACK). Asegura que los efectos de T no tengan efecto en ninguno de los sitios
  - COMMIT. Asegura que los efectos de las escrituras sean persistidas en todas las BDs que contienen copia de los data items afectados

DDB - Transacciones (Cont.)

- **Funcionamiento.**
  - Ante una operación de BD, el Gestor de Transacciones le pasa la operación al Módulo Gestor de CC
  - Módulo Gestor de CC es responsable de adquisión y liberación de locks
  - Si T requiere acceso a recurso lockeado, T es bloqueada hasta que es posible brindarle lock
  - Una vez obtenido el lock, la operación es enviada al Runtime Processor, para que ejecute la operación
  - Al completar la operación, se libera el lock y el Gestor de Transacciones es actualizado con el resultado de la operación
- **2PC.** COMMIT/ABORT suele implementarse utilizando 2PC

DDB - Transacciones - 2PC

- **Requerimiento.** 2PC requiere
  - Gestor de Recovery Global para mantener información de recovery
  - Gestor de Recovery Locales y la información que mantienen (logs, etc.)
- **Problema.** Protocolo de bloqueo. Si falla el coordinador una vez pedido el lock, bloquea a todos los sitios participantes, causando que tengan que esperar hasta que el coordinador se recobre

DDB - Transacciones - 3PC

- **Solución.** 3PC.
- **Funcionamiento.** Divide segunda fase de commit en dos subfases
  1. **prepare-to-commit**
     - Comunica resultado de la fase de votación a todos los participantes
     - Si todos los participantes votan afirmativamente, coordinador pide que pasen a estado prepare-to-commit
  2. **commit**
     - Idéntico a su contraparte de 2PC
     - Si el coordinador cae durante en esta subfase, otro participante puede observar esta situación
     - Consultar al sitio caído si recibió el mensaje **prepare-to-commit**
     - Si no lo recibió, puede asumir abortar. Esto permite al protocolo recuperarse ante caídas de participantes
     - Limitando tiempo requerido por una T para realizar un commit o abort, 3PC asegura liberar locks en un cierto tiempo

DDB - Queries - Pasos para el procesamiento

Pasos para el procesamiento de Queries distribuídos ...

1. **Mapeo.** Sea una consulta SQL de la DDB. Se mapea a una consulta algebraica sobre las relaciones globales, referenciando al esquema conceptual global (no se toma en cuenta la distribución y replicación de datos). Así, el mapeo es idéntico al de BDs centralizadas: normalización, análisis de errores semánticos, simplificación y restructuración en consulta algebraica
2. **Localización.** Se mapea el Query del paso anterior a múltiples queries sobre fragmentos individuales (se utiliza información de distribución y replicación de datos)
3. **Optimización de Query Global.** Selección de una estrategia de una lista de candidatos cercanos al óptimo. Tiempo es la unidad más utilizada para medir costo. Costo total es una ponderación de costos de CPU, I/O y comunicación. Dada naturaleza distribuída de las DDB, costo de comunicación suele ser muy significativo
4. **Optimización de Query Local.** Se realiza en todos los sitios de la DDB. Técnicas similares a las de BDs centralizadas

DDB - Queries - Costos

- **Problema.** Alto costo en transferencia de datos sobre la red (archivos intermedios y resultado final)
- **Solución.** Algoritmos de Optimización de Queries deben considerar reducir la "cantidad de datos a transferir"

DDB - Queries - Ejemplo 1

Site 1:

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

10,000 records
each record is 100 bytes long
Ssn field is 9 bytes long
Fname field is 15 bytes long
Dno field is 4 bytes long
Lname field is 15 bytes long

Site 2:

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

100 records
each record is 35 bytes long
Dnumber field is 4 bytes long
Dname field is 10 bytes long
Mgr_ssn field is 9 bytes long

Figure 23.4
Example to illustrate volume of data transferred.

Figura de Elmasri/Navathe-Fundamentals of DB Systems, 7th Ed., Pearson, 2015

- **Asumimos.** Ninguna relación fragmentada

DDB - Queries - Ejemplo 1

- ¿Cuánto espacio ocupa cada relación?
  EMPLOYEE: 10.000 registros * 100 bytes/registro = 10^6 bytes
  DEPARTMENT: 100 registros * 35 bytes/registro = 3.500 bytes
- Traducir la siguiente consulta Q al álgebra relacional: *"Para cada empleado, obtener su nombre y el nombre de departamento donde trabaja"*
  $Q : \pi_{Fname,Lname,Dname}(EMPLOYEE \bowtie_{Dno=Dnumber} DEPARTMENT)$
- ¿Cuántos registros retorna la consulta Q? (asumiendo que todo empleado se relaciona con un departamento)
  10.000 registros
- ¿Cuánto espacio ocupa cada registro de Q?
  15 bytes (Fname) + 15 bytes (Lname) + 10 bytes (Dname) = 40 bytes

DDB - Queries - Ejemplo 1 (Cont.)

- Si Q se ejecuta en Sitio 3 (EMPLOYEE se encuentra en Sitio 1 y DEPARTMENT en Sitio 2), mencionar 3 posibles estrategias para ejecutar Q y calcular costo (cantidad de bytes a transferir en cada caso)
  1. Transferir (EMPLOYEE y DEPARTMENT) → Sitio 3
     Ejecutar JOIN en Sitio 3
     Costo: 1.000.000 + 3.500 = **1.003.500 bytes deberán ser transferidos**
  2. Transferir EMPLOYEE → Sitio 2
     Ejecutar JOIN en Sitio 2
     Transferir resultado a Sitio 3
     Costo: 1.000.000 (Sitio 1 → Sitio 2) + 400.000 (Site 2 → Site 3)
     = **1.400.000 bytes deberán ser transferidos**
     (Tamaño de Site 2 → Site 3 = 40 bytes/registro * 10.000 registros = 400.000 bytes)
  3. Transferir DEPARTMENT → Sitio 1
     Ejecutar JOIN en Sitio 1
     Transferir resultado a Sitio 3
     Costo: 3.500 (Sitio 2 → Sitio 1) + 400.000 bytes (Site 1 → Site 3)
     = **403.500 bytes deberán ser transferidos**
     (Tamaño de Site 1 → Site 3 = 40 bytes/registro * 10.000 registros = 400.000 bytes)

DDB - Queries - Ejemplo 2

- Traducir la siguiente consulta Q' al álgebra relacional: *"Para cada departamento, obtener su nombre y el del encargado"*
  $Q' : \pi_{Fname,Lname,Dname}(DEPARTMENT \bowtie_{Mgr\_ssn=Ssn} EMPLOYEE)$
- Si Q' se ejecuta en Sitio 3 (EMPLOYEE se encuentra en Sitio 1 y DEPARTMENT en Sitio 2), mencionar 3 posibles estrategias para ejecutar Q' y calcular costo (cantidad de bytes a transferir en cada caso)
  1. Transferir (EMPLOYEE y DEPARTMENT) → Sitio 3
     Ejecutar JOIN en Sitio 3
     Costo: 1.000.000 + 3.500 = **1.003.500 bytes deberán ser transferidos**
  2. Transferir EMPLOYEE → Sitio 2
     Ejecutar JOIN en Sitio 2
     Transferir resultado a Sitio 3
     Costo: 1.000.000 (Sitio 1 → Sitio 2) + 4.000 bytes (Site 2 → Site 3)
     = **1.004.000 bytes deberán ser transferidos**
     (Tamaño de Site 2 → Site 3 = 40 bytes/registro * 100 registros = 4.000 bytes)

DDB - Queries - Ejemplo 2 (Cont.)

3. Transferir DEPARTMENT → Sitio 1
   Ejecutar JOIN en Sitio 1
   Transferir resultado a Sitio 3
   Costo: 3.500 (Sitio 2 → Sitio 1) + 4.000 bytes (Site 2 → Site 3)
   = **7.500 bytes deberán ser transferidos**
   (Tamaño de Site 2 → Site 3 = 40 bytes/registro * 100 registros = 4.000 bytes)

DDB - Queries - Ejemplo 3

- Si Q/Q', en vez de ejecutarse en Sitio 3, se ejecutan en Sitio 2 (como siempre, EMPLOYEE se encuentra en Sitio 1 y DEPARTMENT en Sitio 2), mencionar 2 posibles estrategias para ejecutar Q/Q' y calcular costos (cantidad de bytes a transferir en cada caso)
  1. Transferir EMPLOYEE → Sitio 2
     Ejecutar JOIN en Sitio 2
     Costo de Q/Q': **1.000.000 bytes deberán ser transferidos**
  2. Transferir DEPARTMENT → Sitio 1
     Ejecutar JOIN en Sitio 1
     Transferir resultado a Sitio 2

     Costo de Q: 3.500 (Sitio 2 → Sitio 1) + 400.000 bytes (Site 1 → Site 2)
     = **403.500 bytes deberán ser transferidos**
     (Tamaño de Site 1 → Site 2 = 40 bytes/registro * 10.000 registros = 400.000 bytes)

     Costo de de Q': 3.500 (Sitio 2 → Sitio 1) + 4.000 bytes (Site 2 → Site 3)
     = **7.500 bytes deberán ser transferidos**
     (Tamaño de Site 1 → Site 2 = 40 bytes/registro * 10.000 registros = 400.000 bytes)

DDB - Queries - Semijoin

- **Idea.** Reducir cantidad de tuplas de una relación **antes** de transferir a otro sitio
- **Implementación.**
  a) Enviar sólo columna de JOIN
  b) Realizar JOIN
  c) Del resultado retornar columna de join + atributos necesarios en Consulta
  d) Completar la Consulta
- **Ventaja.** Si solo una pequeña fracción de la relación participa en el JOIN, minimiza la transferencia de datos

DDB - Queries - Ejemplo 3 - Q - con SEMIJOIN

- Si Q se ejecuta en Sitio 2 (EMPLOYEE se encuentra en Sitio 1 y DEPARTMENT en Sitio 2)

  $Q : \pi_{Fname,Lname,Dname}(EMPLOYEE \bowtie_{Dno=Dnumber} DEPARTMENT)$

  1. $F = \pi_{Dnumber}(DEPARTMENT)$ en Sitio 2
     Transferir F → Sitio 1
     Costo: 4 bytes/registro * 100 registros
     = **400 bytes deberán ser transferidos**
  2. $R = \pi_{Dno,Fname,Lname}(F \ltimes_{Dnumber=Dno} EMPLOYEE)$
     Transferir R → Sitio 2
     Costo: 34 bytes/registro * 10.000 registros
     = **340.000 bytes deberán ser transferidos**
  3. Ejecutar JOIN de R con DEPARTMENT en Sitio 2 y presentar resultado
  4. Costo total: **340.400 bytes deberán ser transferidos**

DDB - Queries - Ejemplo 3 - Q' - con SEMIJOIN

- Si Q' se ejecuta en Sitio 2 (EMPLOYEE se encuentra en Sitio 1 y DEPARTMENT en Sitio 2)

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

1. $F' = \pi_{Mgr\_ssn}(DEPARTMENT)$ en Sitio 2
   Transferir F' → Sitio 1
   Costo: 9 bytes/registro * 100 registros
   = **900 bytes deberán ser transferidos**
2. $R' = \pi_{Mgr\_ssn,Fname,Lname}(F' \ltimes_{Mgr\_ssn=Ssn} EMPLOYEE)$
   Transferir R' → Sitio 2
   Costo: 39 bytes/registro * 100 registros
   = **3.900 bytes deberán ser transferidos**
3. Ejecutar JOIN de R' con DEPARTMENT en Sitio 2 y presentar resultado
4. Costo total: **4.800 bytes deberán ser transferidos**

DDB - Tipos de DDB

- **DDB Difieren.** Término "DDB" se utiliza para distintos sistemas que en realidad pueden diferir mucho entre sí
- **Definición Común.** Datos y Software distribuídos en múltiples lugares, pero conectados a través de una red de comunicaciones
- **Factores que distinguen a estos sistemas entre sí.**
  1. **Grado de Homogeneidad.** Si todos los servers y usuarios utilizan el mismo software (Homogéneo vs. Heterogéneo)
  2. **Grado de Autonomía Local.** Si no se le permite al sitio local funcionar como un DBMS standalone, entonces el sistema no posee autonomía local

DDB - Tipos de DDB

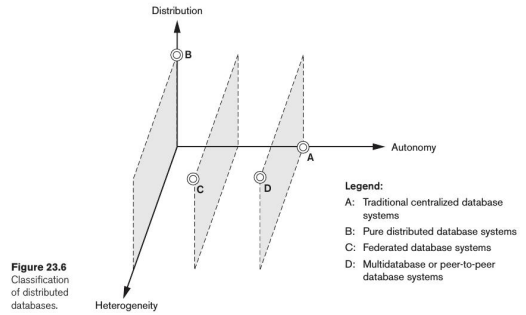


Figure 23.6 Classification of distributed databases.

Figura de Elmasri/Navathe-Fundamentals of DB Systems, 7th Ed., Pearson, 2015

- **FDBS y p2pDBS.** Cada Server es un DBMS independiente y autónomo que posee sus usuarios locales propios, transacciones locales, un DBA y por lo tanto, un alto grado de autonomía
- **Heterogeneidad.** Es necesario un lenguaje canónico y un traductor que traduzca desde los subqueries del lenguaje canónico al lenguaje de cada uno de los servers

DDB - Tipos de DDB

- **p2pBS.** No hay esquema global, pero se construye uno a medida que lo necesita alguna aplicación
- **FDBS.** Existe alguna vista o esquema global de la federación de bases de datos que es compartida por aplicaciones

DDB - FDBS - Heterogeneidad

- **Origen de la Heterogeneidad.**
  - **Diferencias en Modelo de Datos.** DBs pueden provenir de distintos modelos de datos (modelo relacional, objetos o incluso archivos). Incluso si comparten el modelo relacional. Ej. En 2 DBs la misma información puede ser representada por un atributo o una relación. Esto requiere de un mecanismo inteligente de procesamiento de queries que pueda relacionar la información basado en metadata.
  - **Diferencias en Restricciones.** Varían de sistema en sistema. Ej. En ER, algunas relaciones imponen integridad referencial. Se pueden utilizar triggers para imponer ciertas restricciones en el modelo relacional. El esquema global debe lidiar también con conflictos entre restricciones.
  - **Diferencias en Lenguajes de Query.** Incluso dentro del mismo modelo de datos, los lenguajes de query poseen varias versiones. Ej. SQL-89, SQL-92, SQL-99, SQL-2008.

DDB - FDBS - Heterogeneidad (Cont.)

- **Heterogeneidad Semántica.** Ocurre cuando existen diferencias en el significado, interpretación y uso de los datos. La distintas BDs tienen **autonomía de diseño**, que les permite seguir sus propios parámetros de diseño.
- **Solución actual.** Utilización de software variado. Responsables de administrar los queries y transacciones desde la aplicación global hacia las DBs individuales (adicionalmente pudiendo incorpor reglas del negocio) y viceversa.
  - **Middleware.**
  - **Application Servers.** Paquetes de base web. Ej. WebLogic o WebSphere
  - **Enterprise Resource Planning(ERP).** Sistemas genéricos. Ej. SAP, J.D.Edwards ERP

DDB - FDBS - Autonomía

- **Otros tipos de autonomía.**
  - **Comunicación.** Capacidad de decidir cuándo comunicarse con otra DBs
  - **Ejecución.** Capacidad de un componente DB para ejecutar operaciones locales sin interferir en las operaciones externas realizadas por otros componentes DB, y también capacidad para decidir el orden en que se ejecutan.
  - **Asociación.** Capacidad de decidir si compartir y cuánto compartir su funcionalidad (operaciones que soporta) y los recursos (datos que gestiona) con otros componentes DB.

Desafío

Diseñar FDBSs es permitir que los componentes DBs interoperen, y que mantengan la autonomía anterior a la federación.

DDB - Arquitecturas Paralelas vs Distribuídas

- **Prevalencia.** Ambas prevalecen en la industria
- **Paralela.** Común en High-preformance Computing
- **Distribuída.** Evoluciona en grandes Empresas
- **Arquitecturas de sistemas multiprocesador.**
  - **Memoria compartida.** Múltiples procesadores comparten memoria primaria y almacenamiento secundario (disco)
  - **Disco compartido.** Múltiples procesadores comparten almacenamiento secundario (disco), pero cada uno tiene su propia memoria primaria
- **Red.** Este tipo de arquitectura permite a los procesadores comunicarse evitando intercambiar mensaje a través de la red
- **Parallel Database Management Systems (PDBMS).** DBMS desarrolladas usando **Arquitecturas de sistemas multiprocesador**.

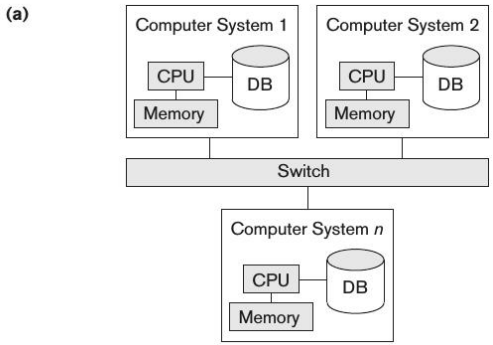
DDB - Arquitecturas Paralelas vs Distribuídas (Cont.)

- **Arquitectura multiprocesador Nada-Compartido (Shared-Nothing).** Cada procesador posee su propia memoria y su propio disco. Los procesadores se comunican a través de redes de alta velocidad.

Parallel Database Management Systems

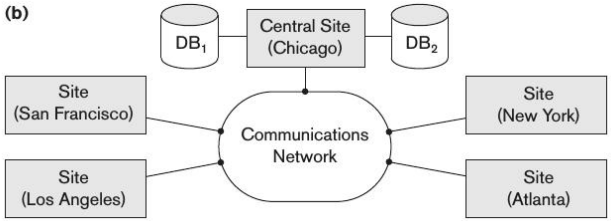
A pesar de que shared-nothing se asemeja a una arquitectura distribuída, la mayor diferencia reside en su modo de operación. En shared-nothing existe una simetría y homogeneidad entre los nodos. Esto no es así en un ambiente de BDs distribuídas.

DDB - Arquitectura Paralela



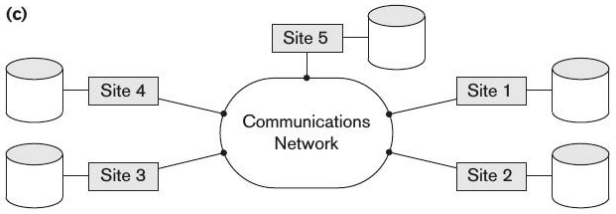
Paralell Database (shared-nothing)
Figura de Elmasri/Navathe-Fundamentals of DB Systems, 7th Ed., Pearson, 2015

DDB - Arquitectura Centralizada / Acceso Distribuído



Base de Datos Centralizada con Acceso Distribuído
Figura de Elmasri/Navathe-Fundamentals of DB Systems, 7th Ed., Pearson, 2015

DDB - Arquitectura DDB



Base de Datos Distribuída Pura
Figura de Elmasri/Navathe-Fundamentals of DB Systems, 7th Ed., Pearson, 2015

DDB - Arquitectura DDB (Cont.)

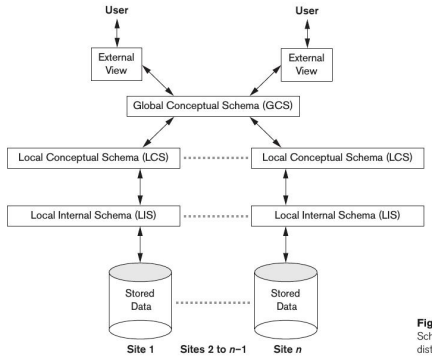


Figure 23.8
Schema architecture of distributed databases.

Esquema Arquitectura Base de Datos Distribuída Pura
Figura de Elmasri/Navathe-Fundamentals of DB Systems, 7th Ed., Pearson, 2015

Se presenta una vista coherente y unificada que muestra la estructura lógica de los datos subyacentes en todos los nodos.

DDB - Arquitectura DDB (Cont.)

- **Esquema Conceptual Global (GCS).** Representa la vista. Provee la transparencia sobre la red
- **Esquema Interno Local (LIS).** Detalles de la organización física de cada nodo de la DDB. Acomoda la potencial heterogeneidad
- **Esquema Conceptual Local (LCS).** Se ocupa de la organización lógica de los datos en cada nodo
- **Transparencia de Fragmentación y Replicación.** Provista por GCS, LCS y los mapeos subyacentes
- **Queries.** El compilador de queries globales hace referencia al GCS desde el catálogo del sistema global, para verificar e imponer las restricciones definidas. El optimizador de consultas globales hace referencia tanto al GCS como al LCS generando consultas locales optimizadas a partir de consultas globales. El costo se puede basar en tiempo de respuesta (CPU, I/O, latencia de la red, etc.) y el tamaño de resultados intermedios
- **Transacciones.** Cada DBMS local puede tener su propio optimizador de queries local, su administrador de transacciones y sus motores de ejecución, así también como su catálogo de sistema local (que mantiene los esquemas locales). El administrador global de transacciones es el responsable de coordinar su ejecución en los múltiples sitios, en conjunto con el administrador local de transacciones en dichos sitios

DDB - Arquitectura FDBS

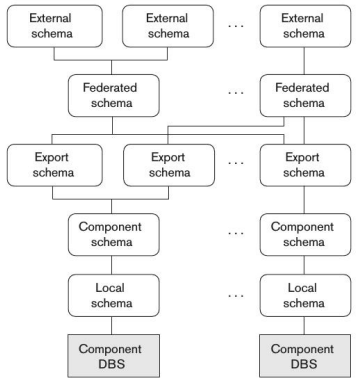


Figure 23.9
The five-level schema architecture in a federated database system (FDBS).

Source: Adapted from Sheth and Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases." *ACM Computing Surveys* (Vol. 22: No. 3, September 1990).

Esquema de Arquitectura de un Sistema de Base de Datos Federada
Figura de Elmasri/Navathe-Fundamentals of DB Systems, 7th Ed., Pearson, 2015

DDB - Arquitectura FDBS (Cont.)

- **Esquema Local (Local Schema).** Es el esquema conceptual de un componente de base de datos (Component DBS)
- **Esquema del Componente (Component Schema).** Se obtiene al traducir el Esquema Local a un modelo de datos canónico o modelo de datos común (CDM) para FDBS
- **Esquema de Exportación (Export Schema).** Representa el subconjunto del Esquema de Componentes que está disponible para el FDBS
- **Esquema Federado (Federated Schema).** Esquema o vista global, resultado de integrar todos los esquemas de exportación compartidos
- **Esquema Externo.** Define el esquema para un grupo de usuarios o una aplicación

DDB - Arquitectura 3-Tier

- DDBMS a gran escala no soportan todas la funcionalidades que hemos visto
- Aplicaciones de DDBS se desarrollan en contexto de arquitecturas cliente/servidor
- Actualmente se usa la arquitectura Three-tier (en particular en aplicaciones web)

DDB - Arquitectura 3-Tier (Cont.)

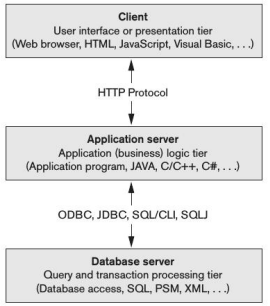


Figure 23.10
The three-tier client/server architecture.

Arquitectura Three-Tier Cliente/Servidor
Figura de Elmasri/Navathe-Fundamentals of DB Systems, 7th Ed., Pearson, 2015

- **Presentation layer (cliente).** Interfaz de usuario e interactúa con el usuario.
- **Application layer (lógica del negocio).** Lógica de la aplicación
- **Database server.** Maneja los pedidos de queries y actualizaciones de la Application layer, los procesa y le retorna los resultados. El resultado puede ser devuelto en formato XML

DDB - Arquitectura 3-Tier (Cont.)

- No existe única estrategia para dividir funcionalidades DBMS
- Ejemplo de procesamiento de Query
  1. Application server genera un query a partir de una entrada del client layer. Lo descompone en múltiples queries para cada sitio. Cada uno de ellos es enviado a la Database server del sitio correspondiente
  2. Cada Database Server procesa su query de manera local y retorna la respuesta al Application server
  3. Application server combina los resultados de los subqueries y produce el resultado, lo formatea y lo envía al sitio del cliente para ser mostrado
- **Application Server. Responsable de:**
  - Generar un plan de ejecución distribuído para los queries y transacciones multi-sitio y de supervisar dicha ejecución enviando comandos a los servers
  - Asegurar consistencia de las réplicas, aplicando técnicas de control de concurrencia distribuída
  - Asegurar atomicidad de las transacciones globales, ejecutando un recovery global cuando un sitio determinado falla

DDB - Arquitectura 3-Tier (Cont.)

- **Transparencia de Distribución.** DDBMS tiene la capacidad de ocultar los detalles de la distribución de los datos desde el Application Server. Permite al Application Server ejecutar consultas y transacciones globales como si la BDs fuera centralizada, sin tener que especificar en qué sitios residen los datos de los queries o transacciones
- Algunos DDBMS no proveen **Transparencia de Distribución**

DDB - Catálogo

- Catálogo es una BDs en sí misma
- Contiene metadata acerca del DDBMS
- La administración del catálogo de DDBs debe asegurar
  - Autonomía del sitio
  - Vistas
  - Distribución y replicación de los datos
- Esquemas de Admistración de Catálogos de DDBMS
  - Centralizado
  - Totalmente replicado
  - Parcialmente replicado

DDB - Catálogo Centralizado

- **Centralizado.** El esquema completo es almacenado en un único sitio.
  - **Ventajas.**
    - Simple de implementar
  - **Desventajas.**
    - Confiabilidad
    - Disponibilidad
    - Autonomía
    - Distribución de la carga de procesamiento
    - Locks pueden generar un cuello de botella en aplicaciones escritura-intensivas

DDB - Catálogo Totalmente Replicado

- **Totalmente replicado.** En cada sitio se almacena una copia del catálogo completo
  - **Ventajas.**
    - Lecturas pueden responderse localmente
  - **Desventajas.**
    - Actualizaciones en el catálogo deben ser transmitidas a todos los sitios
    - Esquema centralizado de 2-Phase Commit para mantener consistencia del catálogo
    - Aplicaciones escritura-intensivas (locks) pueden causar un incremento en el tráfico de la red

DDB - Catálogo Parcialmente Replicado

- **Parcialmente replicado.**
  - Cada sitio mantiene la información completa del catátologo de los datos que están almacenados de manera local en ese sitio
  - Cada sitio, permite también, almacenar en cache entradas obtenidas de otros sitios
  - Entradas en cache no garantiza tener información actualizada
  - El sistema lleva registro de las entradas del catálogo para los sitios donde se creó el objeto y para los sitios que contienen copias de este objeto
  - Cualquier cambio en las copias se propaga inmediatamente al sitio original (de creación)

DDB - Bibliografía

- Capítulo 23 Elmasri/Navathe - **Fundamentals of Database Systems, 7th Edition** Pearson, 2015.