UNIDAD III – DISEÑO DE BDD

3.1 El problema de diseño

El problema de diseño de bases de datos distribuidos se refiere, en general, al tomar decisiones **acerca de la ubicación de datos y programas** a través de los diferentes sitios de una red de computadoras. Este problema debería estar relacionado al diseño de la misma red de computadoras, sin embargo, únicamente el diseño de la base de datos será tomado en cuenta. La decisión de donde colocar a las aplicaciones tiene que ver tanto con el software del SMBDD como con las aplicaciones que se van a ejecutar sobre la base de datos.

El diseño de las bases de datos centralizadas contempla los dos puntos siguientes:

1. Diseño del "esquema conceptual" el cual describe la base de datos integrada (esto es, todos los datos que son utilizados por las aplicaciones que tienen acceso a las bases de datos).
2. Diseño "físico de la base de datos", esto es, mapear el esquema conceptual a las áreas de almacenamiento y determinar los métodos de acceso a las bases de datos.

En el caso de las bases de datos distribuidas se tienen que considerar los dos siguientes:

3. Diseño de la fragmentación, este se determina por la forma en que las relaciones globales se subdividen en fragmentos horizontales, verticales o mixtos.
4. Diseño de la asignación de los fragmentos, esto se determina en la forma en que los fragmentos se mapean a las imágenes físicas, en esta forma, también se determina la solicitud de fragmentos.

Objetivos del Diseño de la Distribución de los Datos.

En el diseño de la distribución de los datos, se deben de tomar en cuenta los siguientes objetivos:

• **Procesamiento local.** La distribución de los datos, para maximizar el procesamiento local corresponde al principio simple de colocar los datos tan cerca como sea posible de las aplicaciones que los utilizan. Se puede realizar el diseño de la distribución de los datos para maximizar el procesamiento local agregando el número de referencias locales y remotas que le corresponden a cada fragmentación candidata y la localización del fragmento, que de esta forma se seleccione la mejor solución de ellas.

• **Distribución de la carga de trabajo.** La distribución de la carga de trabajo sobre los sitios, es una característica importante de los sistemas de cómputo distribuidos. Esta distribución de la carga se realiza para tomar ventaja de las diferentes características (potenciales) o utilizaciones de las computadoras de cada sitio, y maximizar el grado de ejecución de paralelismo de las aplicaciones.

• **Costo de almacenamiento y disponibilidad.** La distribución de la base de datos refleja el costo y disponibilidad del almacenamiento en diferentes sitios. Para esto, es posible tener sitios especializados en la red para el almacenamiento de datos. Sin embargo el costo de almacenamiento de datos no es tan relevante si éste se compara con el del CPU, I/O y costos de transmisión de las aplicaciones.

Enfoques al problema de diseño de bases de datos distribuidas

Existen dos estrategias generales para abordar el problema de diseño de bases de datos distribuidas:

1. El enfoque de arriba hacia abajo (top-down). Este enfoque es más apropiado para aplicaciones nuevas y para sistemas homogéneos. Consiste en partir desde

el análisis de requerimientos para definir el diseño conceptual y las vistas de usuario. A partir de ellas se define un esquema conceptual global y los esquemas externos necesarios. Se prosigue con el diseño de la fragmentación de la base de datos, y de aquí se continúa con la localización de los fragmentos en los sitios, creando las imágenes físicas. Esta aproximación se completa ejecutando, en cada sitio, "el diseño físico" de los datos, que se localizan en éste.

2. El diseño de abajo hacia arriba (bottom-up). Se utiliza particularmente a partir de bases de datos existentes, generando con esto bases de datos distribuidas. En forma resumida, el diseño bottom-up de una base de datos distribuida requiere de la selección de un modelo de bases de datos común para describir el esquema global de la base de datos. Después se hace la traducción de cada esquema local en el modelo de datos común y finalmente se hace la integración del esquema local en un esquema global común.

El diseño de una base de datos distribuida, cualquiera sea el enfoque que se siga, debe responder satisfactoriamente las siguientes preguntas:

- ¿Por qué hacer una fragmentación de datos?
- ¿Cómo realizar la fragmentación?
- ¿Qué tanto se debe fragmentar?
- ¿Cómo probar la validez de una fragmentación?
- ¿Cómo realizar la asignación de fragmentos?
- ¿Cómo considerar los requerimientos de la información?

Figura 1. El problema de fragmentación de relaciones.

El problema de fragmentación

El problema de fragmentación se refiere al rompimiento de la información para distribuir cada parte a los diferentes sitios de la red, como se observa en la Figura 1. Inmediatamente aparece la siguiente pregunta: ¿cuál es la unidad razonable de distribución?. Se puede considerar que una relación completa es lo adecuado ya que las vistas de usuario son subconjuntos de las relaciones. Sin embargo, el uso completo de relaciones no favorece las cuestiones de eficiencia sobre todo aquellas relacionadas con el procesamiento de consultas.

La otra posibilidad es usar fragmentos de relaciones (sub-relaciones) lo cual favorece la ejecución concurrente de varias transacciones que accesan porciones diferentes de una relación. Sin embargo, el uso de sub-relaciones también presenta inconvenientes. Por ejemplo, las vistas de usuario que no se pueden definir sobre un solo fragmento necesitarán un procesamiento adicional a fin de localizar todos los fragmentos de una vista. Aunado a esto, el control semántico de datos es mucho más complejo ya que, por ejemplo, el manejo de llaves únicas requiere considerar todos los fragmentos en los que se distribuyen todos los registros de la relación. En resumen, el objetivo de la fragmentación es encontrar un nivel de particionamiento adecuado en el rango que va desde tuplas o atributos hasta relaciones completas (ver Figura 2).

Alternativas sobre replicación para la asignación de fragmentos

La replicación de información es de utilidad para obtener un mejor rendimiento y para ofrecer un mayor grado de confiabilidad (tolerancia a fallas). La replicación se complica cuando es necesario hacer actualizaciones a las copias múltiples de un dato. Por tanto, respecto a la replicación, en la asignación de fragmentos se tienen tres estrategias:

1. No soportar replicación. Cada fragmento reside en un solo sitio.
2. Soportar replicación completa. Cada fragmento en cada uno de los sitios.
3. Soportar replicación parcial.

La relación J se puede fragmentar horizontalmente produciendo los siguientes fragmentos

J1: proyectos con presupuesto menor que $200,000

JNO	JNOMBRE	PRESUPUESTO	LUGAR
J1	Instrumentación	150000	Monterrey
J2	Desarrollo de bases de datos	135000	México

J2: proyectos con presupuesto mayor que o igual a $200,0(

JNO	JNOMBRE	PRESUPUESTO	LUGAR
J3	CAD/CAM	250000	Puebla
J4	Mantenimiento	310000	México
J5	CAD/CAM	500000	Guadalajara

La relación J se puede fragmentar verticalmente produciendo los siguientes fragmentos

J1: información acerca de presupuestos de proyectos

JNO	PRESUPUESTO
J1	150000
J2	135000
J3	250000
J4	310000
J5	500000

J2: información acerca de los nombres y ubicaciones de proyectos

JNO	JNOMBRE	LUGAR
J1	Instrumentación	Monterrey
J2	Desarrollo de bases de datos	México
J3	CAD/CAM	Puebla
J4	Mantenimiento	México
J5	CAD/CAM	Guadalajara

Figura 2 Ejemplo de fragmentación horizontal y vertical

Cada fragmento en algunos de los sitios. Como regla general se debe considerar que la replicación de fragmentos es de utilidad cuando el número de consultas de solo lectura es (mucho) mayor que el número de consultas para actualizaciones. En la Tabla 1 se comparan la complejidad de implementar o tomar ventaja de las diferentes alternativas de replicación, respecto de los diferentes aspectos importantes en bases de datos distribuidas.

	Replicación Completa	Replicación Parcial	Particionamiento
Procesamiento de Consultas	Fácil	Moderado	Moderado
Manejo de Directorios	Fácil o no existente	Moderado	Moderado
Control de Concurrencia	Moderado	Difícil	Fácil

Confiabilidad	Muy alto	Alto	Bajo
Realidad	Aplicación posible	Realista	Aplicación posible

Tabla 1. Comparación de las estrategias de replicación de fragmentos.

3.2. Conceptos de diseño de base de datos distribuidos.

Asignación. Lugar dónde se sitúan los fragmentos y réplicas, La elección del lugar y el grado de replicación depende de los objetivos de rendimiento y disponibilidad. También del tipo de transacciones y su frecuencia.

Atomicidad: Una transacción es una unidad atómica de procesamiento, esta se realiza o no se realiza.

Back-End: Se refiere a la vista disponible para el usuario administrador de un sistema. En este se manejan configuraciones, administración y edición de información de carácter importante para el correcto funcionamiento del sistema.

BDD. Son varias BD interrelacionadas lógicamente y situadas en diferentes nodos de una red de ordenadores.

Bloqueo. Es una acción que debe ser realizada pero que está esperando a un evento. Para manejar los bloqueos hay distintos acercamientos: prevención, detección, y recuperación.

COMMIT. Señala el término exitoso de la transacción, le dice al manejador de transacciones que se ha finalizado con éxito una unidad lógica de trabajo, que la base de datos esta de nuevo en un estado consistente, y que se pueden hacer permanentes todas las modificaciones efectuadas por esa unidad de trabajo.
Directorio global. Contiene la información de fragmentación. Lo utiliza el SBDD

Distribución: Los componentes del sistema están localizados en la misma computadora o no.

Fragmentación. Decidir dónde situar las partes de la BDD – Se puede plantear top-down o bottom-up.

Fragmento. Una porción de una tabla; una tabla se divide y almacena como un número de fragmentos. Un fragmento es a veces llamado “partición”; sin embargo, “fragmento” es la denominación preferida. Las tablas fragmentadas se usan para facilitar balanceo de carga entre máquinas y nodos.

Fron-End: Se refiere a la vista que está disponible para el usuario final de dicho sistema.

Heterogeneidad: Un sistema es heterogéneo cuando existen en él componentes que se ejecutan en diversos sistemas operativos, de diferentes fuentes, etc.

Partición. Este modelo consiste en que solo hay una copia de cada elemento, pero la información está distribuida a través de los nodos. En cada nodo se aloja uno o más fragmentos disjuntos de la base de datos. Como los fragmentos no se replican esto disminuye el costo de almacenamiento, pero también sacrifica la disponibilidad y fiabilidad de los datos

Replicación. El esquema de BDD de replicación consiste en que cada nodo debe tener su copia completa de la base de datos. Es fácil ver que este esquema tiene un alto costo en el almacenamiento de la información. Debido a que la actualización de los datos debe ser realizada en todas las copias, también tiene un alto costo de escritura, pero todo esto vale la pena si tenemos un sistema en el que se va a escribir pocas veces y leer muchas, y dónde la disponibilidad y fiabilidad de los datos sea de máxima importancia.

ROLLBACK. Señala el término no exitoso de la transacción, le dice al manejador de transacciones que algo salió mal, que la base de datos podría estar en un estado inconsistente y que todas las modificaciones efectuadas hasta el momento por la unidad lógica de trabajo deben retroceder o anularse.

SGBD. Es el que gestiona BD distribuidas de forma transparente para el usuario (éste ve las BD como si fueran una sola BD centralizada).

Transacción. Es una secuencia de una o más operaciones agrupadas como una unidad. El inicio y el final de la transacción definen los puntos de consistencia de la base de datos. Si una acción de la transacción no se puede ejecutar, entonces ninguna acción dentro de la secuencia que conforma la transacción tendrá efecto.

3.3. Fragmentación

Existen tres tipos de fragmentación la horizontal, la vertical y la mixta.

Fragmentación Horizontal

Una tabla T se divide en subconjuntos, T1, T2, ...Tn. Los fragmentos se definen a través de una operación de selección y su reconstrucción se realizará con una operación de unión de los fragmentos componentes.

Cada fragmento se sitúa en un nodo.
Pueden existir fragmentos no disjuntos: combinación de fragmentación y replicación.

Ejemplo:

Tabla inicial de alumnos

DNI	Escuela	Nombre	Nota ingreso	Beca
87633483	EUI	Concha Queta	5.6	No
99855743	EUI	Josechu Letón	7.2	Si
33887293	EUIT	Oscar Romato	6.1	Si
05399075	EUI	Bill Gates	5.0	No
44343234	EUIT	Pepe Pótamo	8.0	No
44543324	EUI	Maite Clado	7.5	Si
66553234	EUIT	Ernesto Mate	6.6	No

Tabla de alumnos fragmentada

Fragmento de la EUI: $\sigma_{Escuela="EUI"}(T)$

DNI	Escuela	Nombre	Nota ingreso	Beca
87633483	EUI	Concha Queta	5.6	No
99855743	EUI	Josechu Letón	7.2	Si
05399075	EUI	Bill Gates	5.0	No
44543324	EUI	Maite Clado	7.5	Si

Fragmento de la EUIT: $\sigma_{Escuela="EUIT"}(T)$

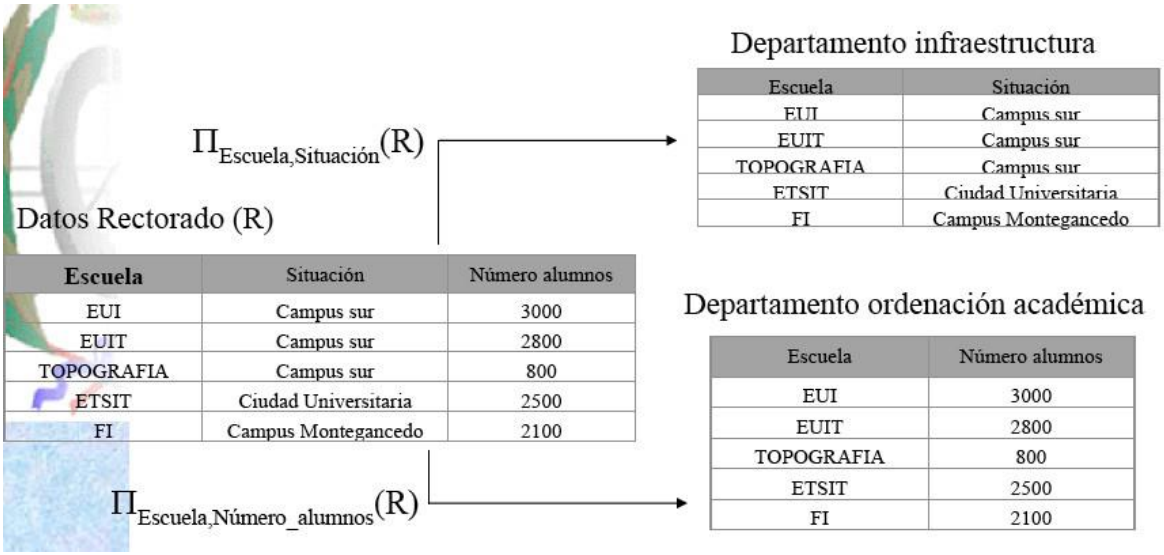
DNI	Escuela	Nombre	Nota ingreso	Beca
33887293	EUIT	Oscar Romato	6.1	Si
44343234	EUIT	Pepe Pótamo	8.0	No
66553234	EUIT	Ernesto Mate	6.6	No

Fragmentación Vertical

Una tabla T se divide en subconjuntos, T1, T2, ...Tn. Los fragmentos se definen a través de una operación de proyección.

Cada fragmento debe incluir la clave primaria de la tabla. Su reconstrucción se realizará con una operación de join de los fragmentos componentes, pueden existir fragmentos no disjuntos: combinación de fragmentación y replicación.

Ejemplo:



Fragmentación Mixta

Como el mismo nombre indica es una combinación de las dos anteriores vistas he aquí un ejemplo a partir de una tabla fragmentada horizontalmente.

DNI	Escuela	Nombre	Beca
87633483	EUI	Concha Queta	No
99855743	EUI	Josechu Letón	Si
05399075	EUI	Bill Gates	No
44543324	EUI	Maite Clado	Si

$\Pi_{DNI,Escuela,Escuela,Nombre,Beca}(E)$

Fragmento de la EUI: $\sigma_{Escuela="EUI"}(T)$

DNI	Escuela	Nombre	Nota ingreso	Beca
87633483	EUI	Concha Queta	5.6	No
99855743	EUI	Josechu Letón	7.2	Si
05399075	EUI	Bill Gates	5.0	No
44543324	EUI	Maite Clado	7.5	Si

$\Pi_{DNI,Escuela,Nombre,Nota\ ingreso}(E)$

DNI	Escuela	Nombre	Nota ingreso
87633483	EUI	Concha Queta	5.6
99855743	EUI	Josechu Letón	7.2
05399075	EUI	Bill Gates	5.0
44543324	EUI	Maite Clado	7.5

3.4. Integridad

El término **integridad de datos** se refiere la correctitud y completitud de la información en una base de datos. Cuando los contenidos se modifican con sentencias *INSERT*, *DELETE* o *UPDATE*, la integridad de los datos almacenados puede perderse de muchas maneras diferentes. Pueden añadirse datos no válidos a la base de datos, tales como un pedido que especifica un producto no existente. Pueden modificarse datos existentes tomando un valor incorrecto, como por ejemplo si se reasigna un vendedor a una oficina no existente. Los cambios en la base de datos pueden perderse debido a un error del sistema o a un fallo en el suministro de energía. Los cambios pueden ser aplicados parcialmente, como por ejemplo si se añade un pedido de un producto sin ajustar la cantidad disponible para vender.

Una de las funciones importantes de un DBMS relacional es preservar la integridad de sus datos almacenados en la mayor medida posible.

- **Datos Requeridos**: establece que una columna tenga un valor no NULL. Se define efectuando la declaración de una columna es NOT NULL cuando la tabla que contiene las columnas se crea por primera vez, como parte de la sentencia CREATE TABLE.
- **Chequeo de Validez**: cuando se crea una tabla cada columna tiene un tipo de datos y el DBMS asegura que solamente los datos del tipo especificado sean ingresados en la tabla.
- **Integridad de entidad**: establece que la clave primaria de una tabla debe tener un valor único para cada fila de la tabla; si no, la base de datos perderá su integridad. Se especifica en la sentencia CREATE TABLE. El DBMS comprueba automáticamente la unicidad del valor de la clave primaria con cada sentencia INSERT Y UPDATE. Un intento de insertar o actualizar una fila con un valor de la clave primaria ya existente fallará.

- **Integridad referencial**: asegura la integridad entre las llaves foráneas y primarias (relaciones padre/hijo). Existen cuatro actualizaciones de la base de datos que pueden corromper la integridad referencial:
  - La inserción de una fila hijo se produce cuando no coincide la llave foránea con la llave primaria del padre.
  - La actualización en la llave foránea de la fila hijo, donde se produce una actualización en la clave ajena de la fila hijo con una sentencia UPDATE y la misma no coincide con ninguna llave primaria.
  - La supresión de una fila padre, con la que, si una fila padre -que tiene uno o más hijos- se suprime, las filas hijos quedarán huérfanas.
  - La actualización de la clave primaria de una fila padre, donde si en una fila padre, que tiene uno o más hijos se actualiza su llave primaria, las filas hijos quedarán huérfanas.

Fuentes:

A. Jaime, Tema 9. Bases de Datos Distribuidas (BDD), http://www.unirioja.es/cu/arjaime/Temas/09.Distribuidas.pdf

Toledo Vicente, Miralles Isarael, Bases de datos distribuidas http://docplayer.es/1922298-Vicente-toledo-israel-miralles-base-de-datos-distribuidas.html

Actividades de la Unidad

I. Por equipo, contesta las siguientes preguntas y envíalas al correo telleslui@hotmail.com

1. ¿Qué tareas que se deben considerar en el diseño de las BDD?
2. Explica los objetivos que se deben cumplir en el diseño de las BDD
3. Ejemplifica las dos estrategias generales para abordar el problema de diseño de bases de datos distribuidas:
4. En que consiste el problema de la fragmentación
5. ¿Qué tipo de fragmentación emplearías en el modelo de la Red de bibliotecas
6. ¿Qué es la integridad de base de daot

II. Realiza un crucigrama con los conceptos de BDD, busca una herramienta que permita su elaboración, se recomienda eclipse crossword