

CAPÍTULO 17.- BASES DE DATOS PARALELAS

17.1.- INTRODUCCIÓN.

El **paralelismo** se utiliza **para proporcionar**:

- **Aceleración** → las consultas se realizan más rápido.
- **Ampliabilidad** → las cargas de trabajo crecientes se tratan sin aumento en el tiempo de respuesta.

17.2.- PARALELISMO DE E/S.

Paralelismo de E/S → reducción tiempo recuperar relaciones del disco dividiendo las relaciones en varios discos (forma más frecuente → división horizontal).

División horizontal → las tuplas de las relaciones se dividen (desagrupan) entre muchos discos.

ESTRATEGIAS:

17.2.1.- TÉCNICAS DE DIVISIÓN. (3 estrategias básicas; n discos: D_0, D_1, \dots, D_{n-1})

1.- TURNO ROTATORIO

- La relación se explora **en cualquier orden** y la **i-ésima tupla** se envía al disco numerado $D_{i \bmod n}$.
- Asegura la distribución homogénea de las tuplas entre los discos (todos los discos aproximadamente el mismo número de tuplas).

2.- DIVISIÓN POR ASOCIACIÓN

- Uno o varios atributos del esquema de la relación dada se designan como **atributos de la división**.
- Se escoge una **función de asociación** cuyo rango sea $\{0, 1, \dots, n-1\}$.
- Cada tupla de la relación original se asocia en términos de los atributos de la división.
- Si la **función** de asociación **devuelve i**, la tupla se ubica en el disco D_i .

3.- DIVISIÓN EN RANGOS

- Distribuye **rangos contiguos** de **valores** de los **atributos a cada disco**.
- Se escoge **un atributo** de división **como vector de división**.
- El vector está formado por una **distribución de n valores del atributo**, tantos como discos.
- Si el atributo de división es menor que el primer valor del vector de división la tupla va primer disco, si es mayor o igual al primer valor y menor que el segundo va al 2º disco, y si es mayor o igual al último va en el último disco.
- Sea $[v_0, v_1, \dots, v_{n-2}]$ el vector de división, tal que si $i < j$, entonces $v_i < v_j$. Sea t una tupla tal que $t[A]=x$. La relación se divide como sigue:
 - Si $x < v_0 \rightarrow t$ se ubica en D_0 .
 - Si $x \geq v_{n-2} \rightarrow t$ se ubica en D_{n-1} .
 - Si $v_i \leq x < v_{i+1} \rightarrow t$ se ubica en D_{i+1} .

17.2.2.- COMPARACIÓN DE LAS TÉCNICAS DE DIVISIÓN.

CLASIFICACIÓN DEL ACCESO A LOS DATOS:

1. **Exploración de la relación** → lectura de una relación completa.
2. **Consultas concretas** → localizar tupla de manera asociativa (que tengan un valor concreto).
3. **Consultas de rangos** → localizar tuplas con valor de un atributo en un rango especificado.

EFICACIA TÉCNICAS PARA TIPOS ACCESO	Exploración de la relación	Consultas concretas	Consultas de rangos
Turno rotatorio	Se adapta perfectamente	Difíciles de procesar (búsqueda en cada disco)	Difíciles de procesar (búsqueda en cada disco)
División por asociación	Útil si f.asociación es una buena función aleatoria y los atributos de división forman una clave de la relación (nº tuplas/disco aprox. igual)	Perfecto si la consulta está basada en el atributo de división. No bueno si basada atributos que no sean de división.	No se adapta bien. (f.de asociación no conservan la proximidad por lo que hace falta explorar varios discos.)
División en rangos		Bien si la consulta está basada en el atributo de división.	Bien si la consulta está basada en el atributo de división.
	Inconveniente → Si hay muchas tuplas en el rango consultado, hay que recuperar muchas tuplas de pocos discos lo que origina un cuello de botella de E/S (ejemplo de sesgo de ejecución)		

El tipo de división también afecta a otras operaciones relacionales como las reuniones, por lo que habrá que elegir dependiendo de las operaciones a realizar.

En general se prefieren las divisiones por asociación y en rangos al turno rotatorio.

Número de discos en los que dividir una relación (en un sistema con muchos):

- Si tiene pocas tuplas y caben en un solo bloque de disco asignar la relación a un solo disco.
- Las relaciones grandes se dividen entre todos los discos disponibles.
- Si la relación consta de m bloques de disco y hay n discos, se ubicará en $\min(m,n)$ discos.

17.2.3.- TRATAMIENTO DEL SESGO.

El sesgo se clasifica por la manera de aparecer en :

- 1.- **Sesgo de los valores de los atributos** → algunos valores pueden aparecer en los atributos de división de muchas tuplas. Todas las tuplas con el mismo valor del atributo de división terminan en la misma partición, lo que da lugar al sesgo.
 - puede producir una división sesgada tanto en división en rangos como en división por asociación.
 - 2.- **Sesgo de la división** → cuando hay un desequilibrio en la carga de la división, aunque no haya sesgo en los atributos.
 - División en rangos → Si no se escoge adecuadamente el vector de división.
 - División por asociación → menos probable si se ha escogido una buena función de asociación.
- Incluso un sesgo pequeño puede dar lugar a una disminución significativa del rendimiento.
 - El sesgo se transforma en un problema creciente al aumentar el grado de paralelismo.

Si los atributos de división forman una clave, se puede construir un buen vector de división mediante su ordenación y al leerlas ordenadas cada $1/n$ tuplas se añade al vector el valor del atributo de la tupla siguiente. El inconveniente es la sobrecarga que supone la ordenación inicial.

17.3.- PARALELISMO ENTRE CONSULTAS.

- **Paralelismo entre consultas** → ejecución en paralelo de **diferentes consultas o transacciones**.
- **Aumenta** la **productividad** (NO el **tiempo de respuesta** de cada transacción).
- **Uso** → **ampliar** sistemas para permitir un **número mayor de transacciones** por segundo.
- **Forma** más **sencilla** de paralelismo, especialmente en **sistemas paralelos de memoria compartida**, (los SGBD diseñados para sistemas uniprosesor pueden funcionar sin apenas cambios ya que incluso los secuenciales admiten procesamiento concurrente).
- En arquitecturas de **disco compartido o sin compartimiento** es **más complicado**:
 - algunas tareas requieren intercambio de **mensajes**.
 - Se debe asegurar que 2 procesadores **no actualicen independientemente** los mismos datos.
 - **Problema de coherencia caché** → Que el procesador tenga la última versión de los datos en memoria intermedia.

Protocolo para garantizar la coherencia caché:

- 1.- Antes de cualquier acceso a una página, una transacción la bloquea en modo compartido o exclusivo según corresponda. Inmediatamente después lee la copia más reciente del disco.
- 2.- Antes de que una transacción libere un bloqueo exclusivo de la página, traslada la misma al disco compartido; luego libera el bloqueo.

Los **protocolos** de disco compartido pueden **extenderse** a las **arquitecturas sin compartimiento** haciendo que cuando un procesador quiera acceder a una página que de guarda en un disco que no es local para él, envíe una petición al procesador que tiene la página en su disco local. Las otras acciones son iguales que en los protocolos de disco compartido.

17.4.- PARALELISMO EN CONSULTAS.

Paralelismo en consultas → ejecución en paralelo de una consulta en varios procesadores y discos.

Puede conseguirse a través de 2 tipos de paralelismo:

- 1.- **Paralelismo EN operaciones** → hace paralela la ejecución de cada una de las operaciones (**actuar sobre cada tupla en paralelo** en operaciones que implican varias tuplas).
- 2.- **Paralelismo ENTRE operaciones** → ejecuta en paralelo las operaciones de las expresiones de las consultas (**evaluar en paralelo el árbol de operaciones** de la consulta).

Las 2 formas son complementarias y pueden utilizarse en una misma consulta.

17.5.- PARALELISMO EN OPERACIONES.

Dado que el n° de tuplas de una relación puede ser grande, el **grado de paralelismo** es **potencialmente enorme**.

Versiones paralelas de algunas **operaciones relacionales**:

17.5.1.- ORDENACIÓN EN PARALELO.

Si la **relación se ha dividido en rangos basándose en los atributos por los que se va a ordenar** → se puede ordenar por separado cada partición y concatenar los resultados para obtener la relación completa ordenada.

Si se ha dividido siguiendo algún otro método:

1. Ordenación por división en rangos.
2. Ordenación y reunión externas paralelas.

1.- ORDENACIÓN POR DIVISIÓN EN RANGOS.

No hace falta dividir en rangos la relación en los mismos procesadores o discos en los que se guarda la relación.

Pasos:

1. **Redistribuir** la relación utilizando una estrategia de división en rangos. Utilizar vector de división con carga equilibrada.
2. Cada procesador **ordena localmente** su partición de la relación (**Paralelismo de datos** → cada procesador la misma operación en un conjunto de datos diferente).
3. Operación final de **mezcla** (trivial).

2.- ORDENACIÓN Y REUNIÓN EXTERNAS PARALELAS.

- Se **divide** la relación entre los discos (no importa la manera).
- Cada procesador **ordena localmente** los datos que contiene su disco.
- Las partes ordenadas **se mezclan**. Se puede hacer **de la siguiente manera**:
 - Las particiones ordenadas se dividen en rangos (utilizando el mismo vector de división). Las tuplas se envían de acuerdo con el orden de ordenación al procesador que corresponda según indique el vector de división, por lo que a cada procesador le llegan las tuplas en corrientes ordenadas.
 - Cada procesador mezcla las corrientes que recibe.
 - Las partes ordenadas en cada procesador se concatenan.

17.5.2.- REUNIÓN PARALELA.

Se comparan pares de tuplas para ver si satisfacen la condición de reunión, si lo hacen, el par se añade al resultado.

1.- REUNIÓN POR DIVISIÓN.

- Para ciertos tipos de reuniones, como las **equirreuniones** y las **naturales**, **es posible dividir** las dos relaciones de entrada entre los procesadores **y procesar localmente** la reunión en cada procesador.
- Las relaciones **se pueden dividir por**:
 - División **en rangos** de los atributos de reunión.
 - División **por asociación** de los atributos de reunión.
 Utilizar la **misma función** de división **para las 2** relaciones, entiéndase mismo vector de división o misma función de asociación, según corresponda.
- Una vez divididas se puede utilizar **localmente cualquier técnica** de reunión.
- El **sesgo** representa un problema especial cuando se utiliza división por rangos, pues es difícil encontrar un mismo vector que no produzca sesgo en ninguna de las 2 relaciones. La división por asociación es probable que tenga menos sesgo, excepto cuando haya muchas tuplas con los mismos valores de los atributos de reunión.
- La división **no es aplicable a todos** los **tipos de reuniones** (ej. Cuando todas las tuplas de r se reúnen con alguna tupla de s – hay que poder comparar toda la relación r, con lo que no puede dividirse –).

2.- REUNIÓN CON FRAGMENTOS Y RÉPLICAS.

- Los fragmentos y réplicas **funcionan con cualquier condición** de **reunión**, dado que todas las tuplas de r pueden compararse con todas las tuplas de s. Por lo que puede utilizarse cuando no se puede emplear la división.
- Veamos primero un caso particular (fragmentos y réplicas asimétricos) y luego el general.

FRAGMENTOS Y RÉPLICAS ASIMÉTRICOS. (Caso particular $m=1$)

1. Se divide una de las relaciones r (con cualquier técnica de división. Si está guardado en particiones, no hace falta volver a dividirlo).
2. La otra relación s se replica en todos los procesadores.
3. Cada procesador P_i procesa localmente la reunión r_i con toda s , utilizando la técnica de reunión.

FRAGMENTOS Y RÉPLICAS. (Caso general)

- La relación r se divide en n particiones y s se divide en m particiones (con cualquier técnica).
- m y n pueden ser distintos.
- $m * n \leq n^{\circ}$ procesadores.
- El procesador P_{ij} procesa la reunión de r_i con s_j (con cualquier técnica de reunión).
- Suele tener mayor coste que la división excepto si una de las relaciones es pequeña (puede ser menos costoso replicarla que aplicar la división).

Explican como se paralelizan algunos algoritmos de operaciones del capítulo 12. No lo veo necesario para el examen.

3.-REUNIÓN POR ASOCIACIÓN DIVIDIDA EN PARALELO**4.-REUNIONES CON BUCLES ANIDADOS EN PARALELO****17.5.3.- OTRAS OPERACIONES RELACIONALES.****17.5.4.- COSTE DE LA EVALUACIÓN EN PARALELO DE LAS OPERACIONES.**

Tiempo empleado por una operación en paralelo:

$$T_{div} + T_{con} + \max(T_0, T_1, \dots, T_{n-1})$$

Donde:

$T_{div} \rightarrow$ tiempo necesario para dividir las relaciones.

$T_{con} \rightarrow$ tiempo empleado en construir los resultados.

$T_i \rightarrow$ tiempo utilizado por la operación en el procesador P_i .

- Suponiendo que las tuplas se distribuyen sin sesgo, el número de tuplas enviadas a cada procesador puede estimarse como $1/n$ del número total de tuplas.
- El rendimiento se ve muy afectado por cualquier sesgo en la distribución del trabajo entre los procesadores.
- Para evitar el sesgo \rightarrow Tabla de frecuencias, o histograma, de los valores de los atributos para cada atributo de cada relación.
- Se construyen las funciones de división con carga equilibrada utilizando histogramas.

17.6.- PARALELISMO ENTRE OPERACIONES

Hay 2 formas:

- El paralelismo de encauzamiento.
- El paralelismo independiente.

17.6.1.- PARALELISMO DE ENCAUZAMIENTO.

- **Encauzamiento** → las tuplas resultado de una operación, las consume una segunda operación, incluso antes de que la primera operación haya producido el conjunto completo de tuplas de su resultado.
- **Ventaja** → no es necesario escribir en disco los resultados intermedios.
- encauzamiento como fuente de paralelismo:
 - **Paralelismo encauzado** → Ejecutar simultáneamente las operaciones en procesadores diferentes, de modo que se consuman las tuplas en paralelo con su producción.
- Es útil con un número pequeño de procesadores pero **no se extiende bien**:
 1. Las cadenas de cauce no suelen lograr la longitud suficiente.
 2. No se pueden encauzar operaciones que no producen resultados hasta haber consumido toda la entrada.
 3. Aceleración marginal en los casos en los que el coste de un operador es mucho mayor que los de los demás.

17.6.2.- PARALELISMO INDEPENDIENTE.

Paralelismo independiente → se ejecutan en **paralelo** las **operaciones** en las expresiones de las consultas que son **independientes** entre sí.

No proporciona un **alto grado de paralelismo**.

17.6.3.- OPTIMIZACIÓN DE CONSULTAS.

- **Optimizador de consultas** → toma una consulta y encuentra el plan de ejecución más económico de entre los posibles que proporcionan la misma respuesta.
- Son más **complicados** que los optimizadores para evaluación secuencial:
Hay que tener en cuenta los costes de división, el sesgo y la contención de recursos.
- Se suelen utilizar **enfoques heurísticos** para reducir el nº de planes de ejecución que se consideran:
 1. Considerar únicamente los planes que **paralelizan todas las operaciones** de todos los procesadores y no utilizan encauzamiento (deben evitarse los cauces largos para no retener recursos).
 2. Escoger el plan **secuencial más eficiente** y luego paralelizar sus operaciones.
- Otra dimensión de la optimización es el diseño de la **organización del almacenamiento físico** para acelerar las consultas. Como la óptima es diferente para consultas diferentes, se escoge haciendo una estimación de la combinación de consultas esperada.

17.7.- DISEÑO DE SISTEMAS PARALELOS.

- SBD paralelos a gran escala → grandes volúmenes de datos y **consultas de ayuda a las decisiones**.
- Importante la carga de datos en paralelo desde fuentes externas.
- **Aspectos de disponibilidad**:
 1. Poder de **recuperación** frente al fallo de procesadores o discos. Replicación de los datos (los de 1 procesador entre varios para no cuello botella).
 2. **Reorganización** interactiva de los datos y los cambios de los esquemas. Seguimiento de procesadores con fallos y redistribución del trabajo.
- Con grandes volúmenes de datos **operaciones** sencillas pueden **tardar mucho** y es inaceptable que el sistema no este disponible mientras se llevan a cabo. Los sistemas paralelos, permiten que tales operaciones se lleven a cabo **interactivamente** (no bloquean toda la relación y en su lugar el proceso hace un seguimiento de las actualizaciones para realizar cambios en el resultado que está generando).