OPTIMABDD: HERRAMIENTA CASE PARA EL DISEÑO DE BASES DE DATOS DISTRIBUIDAS[1]

Francisco Javier Moreno*
John Freddy Duitama Muñoz**
Fernando Osorio Giraldo***
Juan Bernardo Quintero****

RESUMEN

Se describen las facilidades que ofrece OPTIMABDD, una herramienta CASE construida para diseñar Bases de Datos Distribuidas. El CASE utiliza de una manera complementaria dos metas en la distribución de los datos: Ubicar cada elemento de información en el sitio donde más se use y evaluar el costo de las decisiones que se tomen, para medir, por ejemplo, en que medida se distribuye la carga de trabajo o los tiempos de procesamiento esperados para cada aplicación.

Palabras claves: Diseño de Bases De Datos, Herramientas CASE, Bases de datos Distribuidas.

1. INTRODUCCIÓN

Una Base de Datos Distribuida está conformada por un conjunto de localidades interconectadas por una red de computador; en cada sitio hay un Sistema de Gestión de Bases de Datos que funciona autónomamente pero que requiere interactuar con los demás , pues los datos que solicitan las aplicaciones posiblemente se encuentran ubicados en diferentes nodos.

La tarea del diseñador de la Base de Datos consiste en definir cómo distribuir la información entre todos los sitios al trazar como objetivo entre otros el lograr máxima localidad,o sea que los datos se encuentren en el sitio donde más se necesiten, o

* **Ingeniero de Sistemas, Universidad de Antioquia.**
** **Profesor Facultad de Ingeniería, Universidad de Antioquia.**
*** **Ingeniero de Sistemas, Universidad de Antioquia.**
**** **Ingeniero de Sistemas, Universidad de Antioquia.**

1 Proyecto de Grado desarrollado en el Departamento de Ingeniería de Sistemas por los estudiantes *Francisco Javier Moreno Arboleda, Fernando Osorio Giraldo, Juan Bernardo Quintero* y dirigido por el Profesor *John Freddy Duitama M.*

el aprovechar la distribución para paralelizar ciertas operaciones o el distribuir la carga de trabajo.

El procedimiento a seguir le exige resolver dos problemas fundamentales: cómo fragmentar las relaciones que conforman la Base de Datos y donde ubicar cada uno de estos fragmentos, de tal manera que se asignen según el objetivo de optimización definido. En algunas ocasiones será necesario tener réplicas de éstos en varias localidades.

Tomar las anteriores decisiones no es una tarea fácil y exige acudir a heurísticas que sirvan de punto de apoyo al diseñador. A continuación describimos cómo OPTIMABDD interactúa con el analista , el tipo de resultados que entrega y las ayudas que otorga, pues fue construida como la fusión de dos metodologías complementarias que resuelven tal tipo de interrogantes.

La importancia de la herramienta radica en que permite al diseñador concentrarse en aspectos fundamentales como la definición de los fragmentos y la obtención de heurísticas realistas, mientras ella se ocupa de una serie de cálculos muy útiles pero engorrosos de realizar. Además facilita simular diferentes escenarios para determinar posibles variaciones en los resultados.

2. METODOLOGÍA DATAID-D.

Desarrollada por Ceri, Pernici y Wiederhold [CERI-87], DATAID-D le ayuda al diseñador a distribuir los datos de tal manera que se minimicen los accesos remotos, o sea buscando la máxima localidad de los datos, objetivo especialmente útil en redes de baja velocidad o de mucho tráfico.

Es importante mencionar que la distribución de fragmentos puede ser:

No redundante: Cada fragmento está mapeado exactamente en uno y sólo un sitio.
Redundante: Cada fragmento es mapeado en uno o más sitios.

El replicar es de mucho beneficio desde el punto de vista del respaldo porque es poco probable la pérdida de varias copias de la misma información, además de que las aplicaciones pueden accesar copias alternativas cuando hay fallas en la que generalmente usan. Pero por otro lado implica incrementar los costos de almacenamiento y de mantener actualizadas todas las copias del mismo fragmento.

El analista debe entonces establecer un balance para decidir cuando realizar réplicas. La metodología trata de ayudarle en este aspecto al evaluar la diferencia entre el número de accesos de lectura locales ganados con cada réplica que se agregue, contra el número de accesos por actualizaciones requeridas para mantener las copias consistentes.

La siguiente información debe ser entrada al sistema:

- Cómo se ha particionado cada tabla (Fragmentos).

  Los fragmentos consisten de subconjuntos de tuplas (registros) disjuntos de una tabla, en donde cada fragmento tiene asociado un predicado que indica la propiedad que distingue a sus tuplas de las de otro fragmento. Este tipo de fragmentación es denominado horizontal, y con él trabaja DATAID-D. También existe la fragmentación vertical, es decir, particionar una tabla de acuerdo con subconjuntos de sus atributos(columnas), pero no está soportada por la metodología.

- Tabla de Acceso Lógico.
  Es una matriz cuyas celdas contienen el tipo de operación que realiza cada aplicación sobre cada tabla (Lectura, Escritura, Acceso por clave primaria, etc.).

- Tabla de Frecuencias.
  Cada celda representa la frecuencia de ejecución para un período de tiempo dado de la aplicación i desde el sitio k. Se lee fik.

- Tabla de Polarización.
  Donde cada celda representa la probabilidad de uso del fragmento j por la aplicación i desde el sitio k. Se lee pikj.

Con la anterior información suministrada, se proceden a realizar una serie de cálculos para determinar la distribución de los fragmentos:

La herramienta calcula entonces:

$$N_{jk} = \sum_{i:\ i\ usa\ k} f_{ik} * p_{ikj}$$

Número de veces que se usa el fragmento j desde el sitio k para todas las aplicaciones i que se corren en tal localidad.

Por lo tanto, el fragmento j es almacenado en el sitio k, tal que:

$$N_{jk} = Max\ \forall_k N_{jk}$$

Finalmente, para la posible presencia de réplicas se aplica la siguiente formula[2]:

$$d_{jk} = \sum_{i:i\ lee\ en\ j} f_{ik} * p_{ikj} - \sum_{i:escribe\ en\ j} \sum_{h \neq k} f_{ih} * p_{ihj}$$

Diferencia entre lecturas locales ganadas por la presencia de una réplica del fragmento j en el sitio k, contra actualizaciónes requeridas desde otros nodos diferentes a k, para mantener consistente esta nueva copia.

3. METODOLOGÍA OZSU-VALDURRIEZ: [ÖZSU-91]

Al partir de una distribución de fragmentos, que puede ser la suministrada por DATAID-D, esta metodología pretende darle al analista un costo aproximado del desempeño del sistema que está siendo distribuido. Para ello se requiere entrar una información adicional al CASE mientras que otra es calculada automáticamente:

3.1. Datos de entrada :

- Sel i (Fj) = Número de tuplas que necesitan ser accesadas en el fragmento j cuando se procesa la aplicación i.

- Size (Fj) = cardinalidad (Fj) * longitud (Fj)
  Donde longitud es el tamaño en bytes de una tupla típica del fragmento j y la cardinalidad es el número de tuplas promedio que se espera que tenga dicho fragmento.

- USCk = Costo en unidades de tiempo para almacenar un bloque en un sitio k.

- LPCk = Costo en unidades de tiempo para procesar un bloque en un sitio k.

- Informacion de la red.

  Gk1,k2 = Costo en unidades de tiempo de comunicación por "frame" (trama) entre dos sitios del sistema.

  Fsizek1,k2 = Tamaño en bytes de un frame entre dos sitios. Se asume como constante en Optimabdd.

- Es necesario especificar el tamaño en bytes de un bloque físico en las localidades: blk_fisico. (igualmente constante)

3.2. El sistema calcula automáticamente:

- Las matrices[3]:

  RRij = Número de bloques leídos por la aplicación i sobre el fragmento j.

---

2 Para justificar réplica, la diferencia debe ser altamente positiva

3 La herramienta calcula estos valores de acuerdo con fórmulas planteadas por [ULLM-88], según el tipo de almacenamiento físico que posea cada fragmento (B+, hashing, heap, etc.) y apoyándose en la tabla de accesos lógicos suministrada por DATAID-D.

URij = Número de bloques actualizados por la aplicación i en el fragmento j.

- Las matrices booleanas :

$$u_{ij} = \begin{cases} 1 & \text{Si } i \text{ actualiza a } j \\ 0 & \text{De lo contrario} \end{cases}$$

$$r_{ij} = \begin{cases} 1 & \text{Si } i \text{ lee en } j \\ 0 & \text{De lo contrario} \end{cases}$$

Información que se deriva respectivamente de URij y RRij.

- La distribución de fragmentos se encuentra en una matriz que denominaremos X, suministrada igualmente por DATAID-D[4]:

$$X_{jk} = \begin{cases} 1 & \text{Si el fragmento } j \text{ está almacenado o hay una réplica en el sitio } k. \\ 0 & \text{De lo contrario} \end{cases}$$

Con la información suministrada se procede a realizar una serie de cálculos para hallar el desempeño total del sistema.

3.3 Cálculo de costos :

El costo total del sistema de acuerdo con la decisión de almacenamiento de los fragmentos, está dado por:

$$TOC = \sum_{\forall q_i} QPC_i + \sum_{\forall s_k} \sum_{\forall f_j} STC_{jk}$$

qi = aplicación i.
Sk = Sitio k.
Fj = Fragmento j.

Donde QPCi = Costo de procesar la aplicación i en cada sitio que la use y STCjk = Costo de almacenar fj en sk.

Esta fórmula se desagrega de la siguiente manera:

3.3.1. Costo de almacenar Fj en el sitio k.

$$STC_{jk} = USC_k * \left\lceil \frac{Size(F_j) * X_{jk}}{blk\_fisico} \right\rceil$$

Costo de almacenar F_j en el sitio k.

3.3.2. QPCi = PCi + TCi donde:

- PCi = Costo de procesamiento de la aplicación i.
- TCi = Costo de transmisión de la aplicación i.

3.3.2.1. A su vez

PCi = AC_i + IE_i + CC_i .

- AC_i: Costo de acceso a la información.
- IE_i: Costo de control de integridad de la base de datos.
- CC_i: Costo de control de concurrencia.

Se asumen IEi y CCi como constantes. Luego:

$$AC_i = \sum_{\forall s_k} \sum_{\forall f_j} (U_{ij} * UR_{ij} + R_{ij} * RR_{ij}) * X_{jk} * LPC_k$$

- U_{ij}: qi escribe en Fj
- R_{ij}: qi lee en Fj
- X_{jk}: Fragmento j en k.

3.3.2.2. TCi = TCUi + TCRi.

- TCUi: Costo de transmisión por lecturas.
- TCRi: Costo de transmisión por actualización de réplicas.

4 Pero que puede ser modificada por el analista.

Cada término se obtiene de la siguiente manera:

$$TCU_i = \sum_{\forall s_k \in S} \sum_{\forall F_j \in F} U_{ij} * X_{jk} * g_{o(i),k} + \sum_{\forall s_k \in S} \sum_{\forall F_j \in F} U_{ij} * X_{jk} * g_{k,o(i)}$$

Primer término: Costo de enviar todos los mensajes de actualización desde el sitio origen o(i) de la aplicación qi a todas las réplicas del fragmento que deben actualizarse.

Segundo término: confirmación de la operación.

$$MTCRi = \sum_{\forall F_j \in F} \min_{S_k \in S} (R_{ij} * X_{jk} * g_{o(i),k} + R_{ij} * X_{jk} * \frac{sel(F_j) * long(F_j)}{F_{size}} * g_{k,o(i)})$$

Primer término: Costo de transmitir la solicitud de información a los sitios que tienen copia de los fragmentos que quieren ser utilizados.

Segundo término: Costo de transmitir el resultado de la consulta desde el sitio de ubicación del fragmento hasta el sitio que hace la solicitud.

Con TCRi se escoge para consulta el sitio que provoque un mínimo costo entre todas las réplicas.

En resumen el término QPCi incluye entre otros costos : Los de procesar en cada nodo las aplicaciones que corren allí; los de transmitir desde el sitio k, donde se hallan las tuplas que requiere la consulta llevada a cabo por la aplicación i que corre en el sitio h y los costos de actualizar todas las réplicas donde se encuentren ubicadas cuando una aplicación i modifica datos.

En este punto el sistema arroja varios informes : espacio ocupado en disco para cada localidad, tiempo de procesamiento de cada aplicación en cada localidad, costo total del sistema. El usuario entonces es quien procede a evaluar los costos arrojados y a determinar si satisfacen sus requerimientos.

La herramienta permite al analista jugar con la distribución de fragmentos, de tal manera que se pueden conformar varias alternativas de distribución y hallar así un nuevo costo. Por supuesto que si él decide cambiar la distribución arrojada por DATAID-D, la nueva distribución no será óptima desde el punto de vista de minimización de accesos remotos, pero quizás por algún otro criterio sea óptima, por ejemplo de almacenamiento; adicionalmemte se puede prescindir de o agregar nuevas réplicas y observar cómo se afecta el desempeño.

4. CONCLUSIONES.

Optimabdd ayuda al diseñador de BDD en dos frentes:

- Obtener una distribución óptima de los datos.
- Informar el costo de dicha distribución.

Hay ya propuestas algunas mejoras para el sistema, lo que ha dado origen a nuevos proyectos:

- Se requiere un cálculo más preciso para algunos términos de costos que por ahora se asumieron constantes. Es el caso de los costos de comunicación que exigen ser calculados al tener en cuenta protocolos de comunicaciones, topología de red, etc.
- Se requiere adicionar la posibilidad de fragmentar verticalmente las relaciones.
- Un tercer trabajo consiste en crear un modelo de optimización que permita al analista encontrar de acuerdo con otros criterios de optimalidad, distribuciones alternativas de los fragmentos en el sistema.

REFERENCIAS BIBLIOGRÁFICAS

[CERI-84] STEFANO Ceri y GIUSEPPE Pelagatti. Distributed Database: Principles and System. Nueva York, N.Y.: Mc Graw-Hill(1984).

[CERI-87] S. CERI, B. Percini and G. Wiederhold, "Distributed Database Design Methodologies", Proceedings of the IEEE, Vol 75, Nro. 5, pp. 533-546, Mayo 1987.

[MORE-97] MORENO Francisco J., Osorio G. Fernando, Quintero Juan B. y Duitama M. Freddy. Optimabdd: Herramienta CASE para bases de datos distribuidas. Proyecto de grado Facultad de Ingeniería. U. de Antioquia. 1997.

[ÖZSU-91] M.Tamer Özsu and Patrick Valduriez. Principles of Distributed Database Systems. Prentice Hall. 1991.

[ULLM-88] JEFFREY D. Ullman. Principles of Database and Knowledge-Base System. Volúmenes I y II. Computer Science Press. 1988.