Bases de Datos

Segunda Parte - Trabajo Práctico

Fecha de Entrega: 12/11/2010
Fecha de Recuperatorio: 26/11/2010

Enunciado

En la segunda parte del Trabajo Práctico, el objetivo será desarrollar una herramienta vinculada a la optimización de consultas en una base de datos.

SQL es un lenguaje común que conocen el usuario y la base de datos, sin embargo, a la hora de resolver una consulta, la base de datos utiliza otro tipo de representación.

Las consultas suelen ser convertidas en árboles, sobre los cuales se realizan modificaciones con el fin de lograr una ejecución eficiente. La idea es intentar aplicar heurísticas y también utilizar inteligentemente los recursos disponibles para conseguir mejorar los tiempos de las consultas.

En este trabajo nos concentraremos en la primera etapa de la optimización, donde se aplican heurísticas que consisten en realizar modificaciones algebraicas estándar con el objetivo de mejorar los costos de la consulta.

1) **Implementar heurísticas**

En este punto deberán implementar una serie de heurísticas partiendo desde un árbol canónico. Las heurísticas a implementar tienen dependencias entre sí, es decir, una heurística puede asumir que ya se aplicaron ciertas operaciones en el árbol.

a. **Cascada de selecciones**: descomponer las selecciones conjuntivas en una secuencia de selecciones simples formadas por cada uno de los términos de la selección original.
b. **Intercambiar hojas**: cambiar de orden las hojas del árbol para que luego puedan evitarse los productos cartesianos.
c. **Bajar selecciones con condición de junta**: bajar aquellas selecciones con condición de junta hasta los productos cartesianos para poder permitir luego reemplazarlos por un join o natural join.
d. **Reemplazar productos cartesianos**: cambiar las selecciones con condición de junta más producto cartesiano por natural join o join (siempre que sea posible, utilizar natural join).
e. **Bajar selecciones:** llevar las selecciones lo más cercano posible a las hojas del árbol.
f. **Bajar proyecciones**: descomponer las listas de atributos de las proyecciones y llevarlas lo más cercano posible a las hojas del árbol, creando nuevas proyecciones cuando sea posible de manera de no propagar hacia niveles superiores atributos innecesarios.

Cada una de estas heurísticas hereda de la clase *Heuristic* y debe implementar el método *internalApplyHeuristic*. Las dependencias entre heurísticas ya están reflejadas en el código.

Puede que exista cierta funcionalidad común entre las heurísticas, por lo cual sugerimos evitar la duplicación de código e incluir operaciones comunes sobre el árbol en la clase *TreeHelper*.

2) **Presentar casos de test para cada heurística**

Por cada una de las anteriores heurísticas deberán presentar casos de test que verifiquen su implementación.

Es fundamental que los casos de prueba abarquen diferentes escenarios (no hace falta que consideren consultas de más de 4 relaciones) y que estén debidamente documentados.

Para facilitar esta etapa, les entregamos un visor de árboles que les será de ayuda para comparar el árbol original con el modificado.

Consideraciones

Para la resolución deberán tener en cuenta las siguientes consideraciones:

- Para la corrección no sólo se evaluará que lo implementado funcione correctamente sino también la calidad del código generado. Con calidad nos referimos a usar comentarios, nombre declarativo para las variables o métodos, uso de métodos auxiliares, evitar duplicar código, etc.

- La interfaz gráfica cuenta con la funcionalidad básica y es bienvenida cualquier modificación que tienda a mejorarla.

- La creación de los árboles actualmente se realiza "a mano" en el código. Si les resulta más práctico, pueden implementar un parser que tome como entrada una consulta SQL y construya directamente el árbol. Si desean hacer esto y necesitan ayuda, no duden en consultar.

Entrega

La entrega deberá contar con el código que implementa lo pedido, un breve informe con detalles de implementación, decisiones tomadas y todo lo que crean conveniente y, finalmente, casos de test *propios* (incluye la entrada y los resultados esperados) para la demo que se realizará en los laboratorios el día de la entrega. Los casos de prueba también tienen que estar documentados en el informe.

Bibliografía

- "Database Management Systems", Raghu Ramakrishnan – Johannes Gehrke, 2° Edición