DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo práctico Nro. 2

1/12/2010 Bases de Datos

Integrante	LU	Correo electrónico
Facundo Carrillo	693/07	facu.zeta@gmail.com
Rodrigo Castaño	602/07	castano.rodrigo@gmail.com
Dardo Marasca	227/07	dmarasca@yahoo.com.ar
Federico Pousa	221/07	fedepousa@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Heurísticas

En esta sección presentaremos las diferentes heurísticas que se aplicaron para el presente trabajo práctico.

Para implementar las heurísticas se tomó como punto de partida el hecho de que el input del programa sea un arbol canónico. Además, se destaca que para la implementación de cada heurística se asume que el arbol ya sufrió los cambios de las heurísticas que la preceden.

1.1. Cascada de Selecciones

El input de esta heurística es directamente el arbol canónico por lo que, al momento de ejecutarse la misma, se puede asumir que el arbol solo tendra un nodo de selección, donde es probable que la seleccion tenga condiciones multiples. La idea de esta heurística es entonces poder partir este único nodo de selección con varias condiciones en muchos nodos de selección con una sola condicion en cada uno de los nodos. Esto es importante dado que luego, al estar cada selección por separado, las mismas se pueden manipular mejor para optimizar el arbol, ya sea bajando las selecciones lo más cerca de las hojas posible, o bien juntandolas con los productos cartesianos para generar natural joins.

La implementación de esta heurística es bastante straightforward, simplemente se levanta la lista de condiciones del único nodo de selección y luego se va creando un nodo de selección por cada condicion de esta lista. Por último, se reemplaza el nodo original del arbol por esta nueva cascada de selecciones.

1.2. Intercambiar hojas

El input de esta heurística es un arbol optimizado, donde la única optimización realizada hasta el momento, fue la separación de las condiciones de selección en varios nodos diferentes. Asumir este paso es importante ya que si las condiciones se encontrasen todas juntas lo más probable es que esa selección no coincida con las condiciones necesarias para realizar un natural joins, dado que es probable que la selección original tenga condiciones sobre varias tablas y no solo sobre igualdades. La idea de esta heurística es entonces poder ver si se pueden cambiar las hojas entre sí para que luego, en la aplicación de las heurísticas siguientes, se puedan cambiar condiciones de seleccion y productos cartesianos por natural joins lo cual representa, en la mayoría de los casos, una gran mejora para el costo de procesamiento de la consulta.

Para implementar esta heurística se listaron todas las condiciones de junta para ver que tablas se encontraban afectadas por las mismas. Una vez que se encuentran dos tablas afectadas por una condicion de junta, se pegan juntas en el arbol con las primeras dos hojas desde más a la izquierda y luego se sigue viendo si existen otras condiciones de junta para esas mismas relaciones. Luego, se sigue procesando las demás tablas para ver si existen condiciones de junta para relacionarlas con las tablas que ya fueron utilizadas.

Para la implementación de esta heurística se utilizaron varios métodos auxiliares para listar las tablas, listar todas las condiciones, saber cuales de estas son condiciones de junta, entre otras funcionalidades requeridas. Todos estos se métodos se encuentran implementados en la clase TreeHelper.java

1.3. Bajar selecciones con condiciones de junta

El input de esta heurística es un arbol optimizado, donde el único nodo de selección del arbol canónico ya fue partido en varias selecciones diferentes y además las hojas ya fueron intercambiadas en base a las condiciones de junta presentes. Luego, lo que se quiere lograr en este paso es poder identificar dichas condiciones de junta para así bajarlas hasta el producto

cartesiano que le corresponde para que luego, en la próxima heurística, dicha condición y dicho producto se puedan juntar y formar un join.

Para implementar esta heurística se listaron todas las condiciones de junta presentes en el arbol, utilizando el método auxiliar creado para la heurística anterior. Una vez obtenidas dichas condiciones, se itera sobre el arbol hasta encontrar productos, a medida que se van encontrando los productos se va chequeando si los hijos de ese producto son los correspondientes a las condiciones de junta listadas anteriormente, de ser así se posiciona la condición de junta pertinente como nodo padre del producto que se estaba analizando, de lo contrario se sigue con la iteración.

1.4. Reemplazar productos cartesianos

El input de esta heurística es el arbol optimizado dado como output de la heurística anterior por lo que se puede asumir que todas las condiciones de junta necesarias para cambiar los productos cartesianos por joins se encuentran encima de los productos correspondientes. Dicho esto, lo único que se tiene que hacer en la heurística es cambiar estas condiciones de junta y los productos cartesianos por nodos nuevos correspondientes a los joins.

La implementación de la heurística es bastante directa dada la fuerte asunción sobre el input, lo que hace la misma es iterar sobre el arbol buscando los productos, una vez que encuentra un producto se fija si los nodos superiores al mismo corresponden a condiciones de junta de los hijos del producto que se esta analizando. En caso afirmativo se pueden reemplazar por un join.

1.5. Bajar Selecciones

El input de esta heurística es el arbol que se pudieron bajar las selecciones que se juntan con los productos para cambiarlos por juntas. Es por esto, que para esta heurística se asume que todas las selecciones que existen todavía en el arbol, se presentan como una secuencia de selecciones simples luego de la única proyeccion y antes del primer producto o junta.

La implementación de esta heurística se basa en recortar todas las selecciones anteriormente mencionadas y se van bajando recursivamente hacia las diferentes ramas del arbol, chequeando previamente que selecciones se deben bajar por cuales ramas. Cuando se llega al primer producto o junta, se procesa la rama derecha que, dado que se tratan de arboles sesgados a izquierda, se sabe que es una relación. Por lo cual, lo único que se tiene que hacer es ver cuales de las selecciones que se habían recolectado al bajar por arbol corresponden a dicha tabla. Luego, se sigue recursivamente con el algoritmo por la rama izquierda.

1.6. Bajar Proyecciones

Para esta última heúristica se tomaron las siguientes decisiones.

- Crear proyecciones nuevas despues de las selecciones: Se crearon nuevos nodos de proyección arriba de los nodos de selección para que solamente perduren los campos necesarios en los nodos más altos del arbol. Se podría haber puesto nuevas proyecciones en cada uno de los nodos del arbol de entrada siempre que se pudiese achicar algo, pero se tomó esta decisión dado que de otra manera el arbol podría quedar muy sobrecargado, quedando un arbol de 2n nodos, con n proyecciones. Esta desición es tan solo eso, sería bueno ver como una extensión a este trabajo si la desición tomada es la mejor.

- Crear proyecciones antes de las relaciones, proyectando solamente los campos que son utilizados en algún nodo del arbol. Es importante destacar que no se pueden desprender cuales son los campos de una relación por lo que siempre se proyectan todos los campos usados en el arbol para esa relación. La solución a esto sería trabajar con el catálogo de

la base, pero en este caso, solo agrega complejidad sin añadir ninguna mejora a la idea del tp.

Para la implementación de esta heurística lo que se hizó fue implementar una función que se aplica recursivamente desde la raíz del arbol hasta las relaciones. La función identifica que tipo de nodo se esta trabajando y actua en consecuencia, si se trata de una selección o una relación se crea un nodo de proyección intermedio con los campos pertinentes para esa rama del arbol.

2. Casos de Test