

Administración de Bases de Datos

PRÁCTICA 2 – OPTIMIZACIÓN DE CONSULTAS



4 de mayo de 2025

Prof. David Criado Ramón

Autor:

Ana Aragón Jerónimo

UNIVERSIDAD DE GRANADA

E.T.S. de Ingenierías Informática y de Telecomunicación

Índice

Introducción.....	2
Tablas a usar.....	2
Planteamiento de la consulta.....	2
Árbol de expresión algebraica.....	3
Representación en memoria de una consulta.....	4
Optimización de una consulta.....	6


Introducción

La optimización de consultas en bases de datos (BD) es un aspecto crucial en el rendimiento de los sistemas de gestión de bases de datos (SGBD) , ya que una consulta bien optimizada puede reducir significativamente el tiempo de respuesta y el uso de recursos.

Esta segunda práctica tiene como objetivo profundizar en la representación y optimización de consultas a través del uso de estructuras de datos, particularmente árboles de expresión algebraica, para ilustrar cómo se pueden aplicar técnicas de optimización que mejoren el rendimiento de las consultas, concretamente aplicando heurísticas de optimización.

Tablas a usar

A continuación, se mostrarán las tablas que se van a utilizar para realizar la consulta posteriormente.

```
C: > Users > anara > Documents >  tablas-USS-Callister.sql
1  /*Tablas del USS-Callister*/
2  CREATE TABLE Tripulantes (
3      id_tripulante NUMBER PRIMARY KEY,
4      nombre VARCHAR2(100),
5      estado VARCHAR2(20) /*Vivo, desaparecido, fallecido*/
6  );
7
8  CREATE TABLE Recompensas (
9      id_recompensa NUMBER PRIMARY KEY,
10     id_tripulante NUMBER,
11     titulo VARCHAR2(50),
12     valor_creditos NUMBER,
13     FOREIGN KEY (id_tripulante) REFERENCES Tripulantes(id_tripulante)
14 );
```

La primera tabla es la tabla de los **Tripulantes** de la nave, cuya llave primaria (PK) es el **id_tripulante** (es decir, el identificador para cada tripulante).

La segunda es la tabla de las **Recompensas**, que tiene por llave primaria (PK) el **id_recompensa** (el identificador de la recompensa dada).

Además, el **id_tripulante** de la tabla **Recompensas** (Recompensas.id_tripulante) es una llave externa a **Tripulantes**.

Planteamiento de la consulta

Se va a realizar la consulta siguiente: la obtención de los nombres de los tripulantes en estado “Activo”, el título de las recompensas obtenidas por estos y su valor, siempre y cuando superen los 1000 créditos.

-Álgebra relacional

Π nombre, titulo, valor_creditos (σ estado = 'Activo' (σ valor_creditos > 1000 (Tripulantes \bowtie Recompensas)))

Se usa la operación de proyección (Π) para seleccionar las columnas nombre, titulo y valor_creditos de las tablas, mientras que la operación de selección (σ) se ha empleado para filtrar por aquellos tripulantes que estén activos (estado = 'Activo') y que además tengan más de 1000 créditos (valor_creditos > 1000).

Esto quiere decir que en esta última operación se han incluido 2 condiciones con el operador σ , para filtrar dados 2 criterios.

Además, se combinan las 2 tablas a usar para esta consulta (**Tripulantes** y **Recompensas**) con la operación de Join (\bowtie), usando para ello la columna id_tripulante, común en ambas tablas.

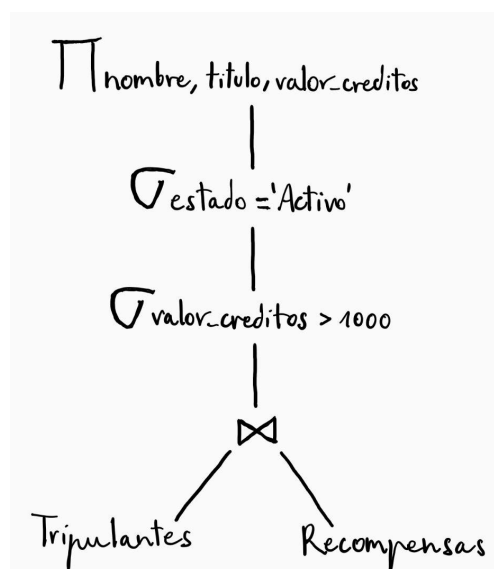
-SQL

Esta consulta en SQL sería de la siguiente forma:

```
17  /*Consulta*/
18  SELECT T.nombre, R.titulo, R.valor_creditos
19  FROM Tripulantes T
20  JOIN Recompensas R ON T.id_tripulante = R.id_tripulante
21  WHERE T.estado = 'Activo' AND R.valor_creditos > 1000;
```

El **SELECT** sería la operación de proyección (Π) que selecciona las columnas mencionadas anteriormente, el **JOIN** (\bowtie) une ambas tablas usando la columna **id_tripulante** que está presente en las 2 tablas, y **WHERE** filtra con las condiciones dadas, unidas con **AND**, siendo estas las 2 operaciones consecutivas de selección (σ).

Árbol de expresión algebraica



Representación en memoria de una consulta

-Plan lógico

A la hora de diseñar el tipo de dato abstracto necesario para el plan lógico, se ha optado por representar este plan mediante un árbol binario.

La elección de este tipo de estructura es debido a que se está trabajando sobre un conjunto de datos pequeño, al tratarse de 2 tablas, y las operaciones de la consulta siguen un orden jerárquico, trabajando sobre uno o dos conjuntos de datos. Esto concuerda con la estructura de un árbol binario, debido a que es un tipo de árbol en el que cada nodo puede llegar a tener como máximo 2 nodos hijos (hasta llegar a los nodos hoja): uno a la izquierda y otro a la derecha.

Algunas ventajas del uso de un árbol binario como estructura son las siguientes:

- ❖ Simplicidad: esta consulta es una con estructura clara y sencilla de seguir, al trabajar únicamente sobre 2 tablas, por lo que se ajusta de manera fácil a la estructura del árbol binario. Además, esta estructura se suele usar para las consultas menos complejas como esta.
- ❖ Fácil optimización: se pueden realizar manipulaciones para mejorar la consulta de manera sencilla, moviendo por ejemplo alguna de las operaciones, como puede ser la selección (anticipación de selección).
- ❖ Claridad: con esta estructura se logra visualizar a simple vista que cada nodo (operación) depende directamente de un nodo o dos anteriores, cumpliendo con la condición de ser un árbol binario.

La estructura a seguir del árbol binario, como dice el enunciado del árbol de expresión, es de almacenamiento de las operaciones en los nodos intermedios, mientras que se almacenan las tablas en los nodos hoja. Por lo que cada nodo representa una operación (proyección, selección, reunión natural) o una tabla.

Para la implementación se haría uso de una clase en la que se almacenaría el valor del nodo (para averiguar si se trata de un nodo intermedio (operación) o de un nodo hoja (tabla)), los valores de los nodos hijos (uno o dos según el nivel del árbol) y un puntero a la estructura de las tablas para poder manejarlas.

Cabe mencionar que para almacenar los valores de los nodos se haría uso de constantes string, cada una representando una operación dada.

-Plan físico

En este caso, se ha escogido una lista enlazada, almacenándose en un nodo cada operación en la representación secuencial, manteniéndose el orden de ejecución al recorrer la lista.

Esta lista enlazada permite añadir operaciones fácilmente, así como calcular los costes de ejecución de forma directa.

Varias de las ventajas del uso de una lista enlazada son las enumeradas a continuación:

- ❖ Sencillez y eficiencia: la lista permite almacenar operaciones de manera secuencial, facilitándose la gestión de las operaciones. Además, es ideal para representar una secuencia de operaciones en el plan físico al mantener el orden de ejecución de las operaciones de forma simple.
- ❖ Optimización de costes: cada nodo de la lista enlazada puede almacenar el costo estimado de la operación que representa. Recorriendo posteriormente esta lista, se puede calcular el coste total de la consulta de manera directa y eficiente.
- ❖ Flexibilidad: se pueden agregar, modificar o eliminar operaciones de la consulta si fuera necesario, sin tener que reorganizar toda la estructura, como podría ocurrir en otro tipo de estructuras (como por ejemplo, en un grafo).

Para representar esta estructura de datos se puede hacer con las siguientes clases:

-Una primera clase para las operaciones, almacenando el tipo de operación (proyección, selección y la reunión natural), el coste estimado de cada operación (para su posterior cálculo total) y las referencias a las operaciones en el caso de la reunión natural (los nodos hoja, a la izquierda y derecha).

- Una segunda clase para cada nodo de operación, que contiene una instancia de la clase anterior y con un puntero al siguiente nodo en la lista, ordenándose las operaciones en una secuencia ordenada.

-Y una última clase con la lista de todas las operaciones, que se encarga de gestionar la lista enlazada, calculando el costo total de la consulta y manteniendo un puntero al primer nodo de la lista para enlazarla.

Esta estructura facilita tanto la representación de la consulta como el cálculo de los costos de ejecución posteriores de forma directa, sencilla y clara.

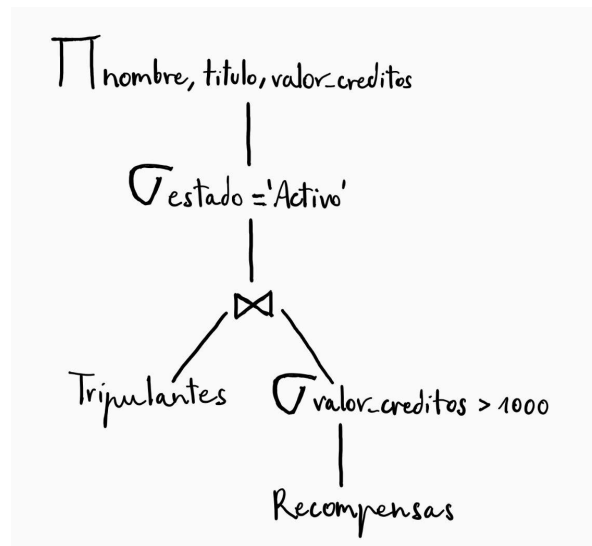
Optimización de la consulta

-Primera heurística: anticipación de la selección

En primer lugar se va a recurrir a una anticipación de una de las selecciones, concretamente la que está justo por encima de la operación de reunión natural ($\sigma_{\text{valor_creditos} > 1000}$), colocando la selección encima de la tabla de Recompensas y por debajo de la reunión natural.

Con este cambio local, se reduce el tamaño de la tabla Recompensas antes de la reunión natural, dado que aquellas recompensas con 1000 o menos créditos ya no se tienen en cuenta a la hora de hacer el join con la tabla de Tripulantes.

Esto reduce el número de combinaciones posibles en el join (reunión natural), que además es una de las operaciones más costosas en bases de datos. Al haber menos combinaciones, se hace un menor uso de la memoria, ahorrando tiempo y procesamiento.



-Segunda heurística: conmutatividad de la reunión natural

Para optimizar esta consulta con otro método que no sea una anticipación de selección o de proyección, se ha recurrido a la conmutatividad de la reunión natural. Para ello, lo que se hace es mover las tablas de orden a la hora de hacer la reunión natural. Así se intercambia el orden de las tablas (**Recompensas** ⋈ **Tripulantes**) cuando se hace el join.

Al realizar este intercambio de tablas, puede mejorar el rendimiento de la consulta (siempre y cuando una de las tablas sea significativamente más pequeña o tenga mejores condiciones de acceso). En esta consulta se ha supuesto que la tabla Recompensas es relativamente más pequeña que la tabla Tripulantes para que sea posible una mejora considerable.

Cuando se usa un join, internamente se tienen que combinar filas de ambas tablas que cumplan una condición. Existen varios algoritmos para hacer esto, y su eficiencia depende del tamaño de las tablas. Cambiando el orden poniendo la tabla más pequeña (Recompensas), ayuda a que el optimizador elija un plan de ejecución más eficiente, reduzca el número de operaciones intermedias y use menos recursos, mejorando el tiempo total de respuesta de la consulta.

