IIC 2413 – Bases de Datos

Guía I2

Pregunta 1: Procedimientos Almacenados

Suponga que tiene una base de datos con las siguientes tablas:

- `A(id int, Anombre varchar(20))`
- `B(id int, Bnombre varchar(20))`
- `C(id int, Cnombre varchar(20))`

Sin embargo, sobre la base de datos solamente se pueden hacer las consultas `SELECT * FROM A`, `SELECT * FROM B` y `SELECT * FROM C`.

- Cree un procedimiento almacenado llamado `intersección_conjunto()`. Este procedimiento calcula la intersección entre `A` y `B`.
- Cree un procedimiento almacenado llamado `join()`. Este procedimiento calcula el join entre `A`, `B`, `C` cuando `A.id = B.id` y cuando `B.id = C.id`.

Pregunta 2: Recursión

Considere la siguiente tabla de la base de datos de una página de documentos en línea:

- `Proviene(Documento1, Editor, Documento2)`.

En ella se indica que el Documento 1 pasó a ser el Documento 2 tras la edición de un Editor determinado (piense en artículos de Wikipedia que son editados por alguien para pasar de una versión a otra). Un ejemplo de la instancia es:

Documento1	Editor	Documento2
Artículo de Chile V1	Usuario 1	Artículo de Chile V2
Artículo de Chile V2	Usuario 1	Artículo de Chile V3
Artículo de Chile V3	Usuario 1	Artículo de Chile V4
Artículo de Chile V4	Usuario 2	Artículo de Chile V5
Artículo de Chile V5	Usuario 1	Artículo de Chile V6
Artículo de Argentina V1	Usuario 3	Artículo de Argentina V2
Artículo de Argentina V2	Usuario 3	Artículo de Argentina V3

Cuadro 1: Instancia de Proviene

Así, en el ejemplo anterior, el Documento "Artículo de Chile V1" pasó a ser "Artículo de Chile V2" tras la edición del editor "Usuario 1". Lo mismo para las demás tuplas.

Entregue una consulta que retorne todos los pares de documentos tal que para llegar del primero al segundo, solamente paso por ediciones consecutivas hechas por el mismo autor. Para la instancia anterior, el output de la consulta sería:

Pregunta 3: SQL + Programación

En clases vimos el operador `Full Outer Join`. Este operador hace el join de dos tablas A y B. Si alguna tupla de A no hace match con ninguna tupla de B o una tupla de B no hace match con ninguna de A, de todas formas se incluyen en el output, llenando con nulos cuando sea necesario.

Considere dos tablas:

- `A(id int, Anombre varchar(20))`

Documento1	Usuario1	Documento2
Artículo de Chile V1	Usuario 1	Artículo de Chile V2
Artículo de Chile V2	Usuario 1	Artículo de Chile V3
Artículo de Chile V3	Usuario 1	Artículo de Chile V4
Artículo de Chile V4	Usuario 2	Artículo de Chile V5
Artículo de Chile V5	Usuario 1	Artículo de Chile V6
Artículo de Argentina V1	Usuario 3	Artículo de Argentina V2
Artículo de Argentina V2	Usuario 3	Artículo de Argentina V3
Artículo de Chile V1	Usuario 1	Artículo de Chile V3
Artículo de Chile V2	Usuario 1	Artículo de Chile V4
Artículo de Argentina V1	Usuario 3	Artículo de Argentina V3
Artículo de Chile V1	Usuario 1	Artículo de Chile V4

Cuadro 2: Ejemplo Output

- `B(id int, Bnombre varchar(20))`

En el siguiente ejemplo se muestra el resultado al hacer el Full Outer Join para una instancia del esquema:

A	id	$Anombre$
	1	A1
	2	A2
	3	A3

B	id	$Anombre$
	1	B1
	2	B2
	2	B2.2
	4	B4

$A \rtimes\ltimes B$	$A.id$	$A.Anombre$	$B.id$	$B.Bnombre$
	1	A1	1	B1
	2	A2	2	B2
	2	A2	2	B2.2
	3	A3	`null`	`null`
	`null`	`null`	4	B4

Sobre la base de datos solamente se pueden hacer las consultas `SELECT * FROM A`, `SELECT * FROM B`. Se pide que haga un programa en Java o Python que imprima en consola el resultado del Full Outer Join entre las dos tablas.

Pregunta 4: Índices y algoritmos internos

a) Considere una relación de 3.000.000 páginas.

- Indique el número de fases y de I/O del External Merge Sort no optimizado.
- Indique el número de fases y de I/O del External Merge Sort optimizado que imprime directamente el resultado final, sin materializar el resultado final. Suponga buffer óptimo.
- ¿Qué significa que el buffer sea óptimo? ¿Por qué un buffer en el que caben 3.000.000 de páginas no es necesario? Demuestre su respuesta.

b) Sea la relación $R(a, b, c, d)$ cuyo tamaño es de 2 millón de tuplas, en que cada página contiene en promedio P tuplas. Las tuplas de R están ordenados de manera aleatoria (cuando no hay índice). El atributo a es además un candidato a llave primaria, cuyos valores van del 0 al 3.999.999 (distribuídos uniformemente). Para cada una de las consultas a continuación, diga el número de I/O que se harán en cada uno de los siguientes casos:

- Analizar R sin ningún índice.
- Usar un *B+Tree Unclustered* sobre el atributo a. El árbol es de altura h. La cantidad de punteros por pagina es de M. El índice de ocupación de las hojas es del 90 %.

Las consultas son:

1. Encontrar todas las tuplas de R.

2. Encontrar todas las tuplas de R tal que $a < 100$.
3. Encontrar todas las tuplas de R tal que $a = 100303$.
4. Encontrar todas las tuplas de R tal que $a > 50$ y $a \leq 150$.

Comente sus resultados e indique que índice prefiere en los distintos casos.

c) En clases se vio que el costo en I/O del Block Nested Loop Join es de $Pags(R) + (Pags(R)/Buffer) \cdot Pags(S)$. Existen otros algoritmos de join que pueden hacer usos de índices sobre las bases de datos. Entregue un algoritmo (explicado con palabras) que compute el join $R \bowtie S$ entre `R(a, b)` y `S(a, c)` en el que ambas llaves primarias están indexadas por un B+Tree clustered. Entregue el costo estimado en I/O (debería ser mejor que el costo del Block Nested Loop Join).
Hint: El B+Tree mantiene la relación ordenada. Debe hacer uso de esto.

d) Considere las relaciones `R(a int)` y `S(a int)`, en donde las relaciones están indexadas por un B+Tree clustered. Tienes acceso a Python y `psycopg2`, pero sólo puedes hacer consultas a la base de datos del tipo:

- `CREATE TABLE`
- `SELECT *`
- `INSERT INTO`

Entregue un algoritmo en Python que haga uso la librería `psycopg2` que calcule la consulta `SELECT * FROM R EXCEPT SELECT * FROM S`, pero que lea cada tupla una única vez. Asuma que los valores en las relaciones son entregados según el orden en que están almacenadas por los B+Tree. Además asuma que los valores no caben en memoria.