IIC2440 - Procesamiento de Datos Masivos

Actividad 01 - Índices

Problema 1. Consideremos la siguiente tabla que guarda información de contacto de personas.

```
Personas(
  id INT,
  nombre VARCHAR(200),
  email VARCHAR(200),
  teléfono VARCHAR(200),
  edad INT
)
```

La tabla tiene un millón de tuplas pero actualmente no está indexada, por lo que las tuplas no están ordenadas en el disco duro. Además, consideremos que el atributo *id* es candidato a llave (es decir, son valores únicos) y los valores van del 1 al 1000000. Consideremos además que cada página puede contener T tuplas.

Queremos que analices el costo I/O para un conjunto de consultas en cada uno de los siguientes casos:

- Analizar la tabla sin ningún índice.
- Usar un *B+Tree clustered* sobre el atributo `id`. El árbol es de altura h_c y supongamos que cada hoja con datos está llena (es decir, cada hoja tiene T tuplas).
- Usar un *B+Tree unclustered* sobre el atributo *id*. El árbol es de altura h_u y cada página con datos contiene P punteros a las páginas con los datos.
- Usar un *Hash Index clustered* que es dinámico y evita la creación de *overflow pages*.
- Usar un *Hash Index unclustered* que es dinámico y evita la creación de *overflow pages*.

Las consultas son:

1. Encontrar todas las tuplas de `Personas`.
2. Encontrar todas las tuplas de `Personas` donde `id = 50`.
3. Encontrar todas las tuplas de `Personas` donde `id <51`.
4. Encontrar todas las tuplas de `Personas` donde `id >100 AND id <= 200`.

Solución. Más que entender el número exacto nos interesa la lógica detrás.

- Para las tablas sin indexar siempre tendremos que recorrer todas las páginas de la tabla en el disco duro.
- Para el B+Tree *clustered* debemos leer tantas páginas como altura tiene el árbol. Ahí encontraremos un puntero a la página del disco con datos. Si tenemos una consulta de rango, podemos recorrer los datos de forma secuencial, porque el orden de los datos es el mismo que el del atributo indexado.
- Para el B+Tree *unclustered* tenemos la misma idea, pero para las consultas de rango debemos leer siempre el puntero hacia la tupla, porque el orden de los datos no necesariamente es el mismo que el del atributo indexado. Por esto mismo, se suma al costo el número de resultados, porque haremos un llamado adicional por cada resultado. **Nota**: restamos 1 al descenso por la altura porque al dividir los punteros en P estamos considerando de nuevo la última hoja, pero agregar este costo no es realmente relevante.

- Al recorrer las hojas e ir a buscar las tuplas en el caso *unclustered*, no estamos asumiendo ningún algoritmo más elaborado para traer los datos (por ejemplo, cachear las páginas de alguna forma).
- El costo para el *Hash Index* es claro en consultas de igualdad, pero para consultas de rango estamos asumiendo una llamada por posible valor del rango. Si bien podemos encontrar un algoritmo mejor, asumimos el uso de este que es el más básico. Recordemos que en el caso *unclustered* asumimos que en los buckets tenemos punteros en vez de tuplas.

Los costos son los siguientes:

Query	**Sin índice**	**B+Tree Cl.**	**B+Tree Uncl.**	**Hash Cl.**	**Hash Uncl.**
`SELECT *`	$\frac{10^6}{T}$	$h_c + \frac{10^6}{T}$	$h_u + \frac{10^6}{P} + 10^6$	10^6	$2 \cdot 10^6$
`id` < 51	$\frac{10^6}{T}$	$h_c + 1$	$h_u + 1$	1	2
`id` < 51	$\frac{10^6}{T}$	$h_c + \frac{50}{T}$	$(h_u + \frac{50}{P} - 1) + 50$	50	$50 + 50$
$100 <$ `id` $<= 200$	$\frac{10^6}{T}$	$h_c + \frac{100}{T}$	$(h_u + \frac{100}{P} - 1) + 100$	100	$100 + 100$

Problema 2. Consideremos la misma tabla del punto anterior. Supongamos que queremos hacer consultas de igualdad y de rango para los atributos `id` y `edad`. Recordando que el atributo `id` es candidato a llave, responda las siguientes preguntas.

1. ¿Cómo esperas que sea la indexación de la tabla?
2. Considerando la indexación propuesta en el punto anterior, calcula el costo I/O de la consulta `SELECT * FROM Personas WHERE edad>=25`. Asume que las personas con edad igual o mayor a 25 es el 70 % de los registros de la tabla. Asuma los valoresde altura, T y P del punto anterior.

Solución. La respuesta esperada es la siguiente.

- La indexación sobre el campo `id` debería ser con un B+Tree *clustered*. La indexación sobre el campo `edad` debería ser con un B+Tree *unclustered*.
- El costo debería ser descender por el arbol, recorrer las hojas con punteros, sumado con un I/O por cada una de las respuestas

Problema 3. Ahora considera que sumamos esta tabla al esquema.

```
Mascotas(
  mid INT PRIMARY KEY,
  nombre VARCHAR(200),
  pid INT,
  FOREIGN KEY (pid) REFERENCES Personas(id)
)
```

Asume que las consultas de interés son:

1. Obtener los datos de una mascota en particular por su `id`.
2. Dada una persona, entrega todos los atributos de la persona junto al de sus mascotas.
3. Dada una mascota, entrega sus atributos junto a los de su persona correspondiente.

¿Cómo esperas que sea la indexación de la tabla `Mascotas`? Además, explica el plan de consulta en cada caso.
Solución. La llave primaria debería estar indexada con un B+Tree *clustered* y el atributo `pid` con un B+Tree *unclustered*. Respecto a los planes de consultas, proponemos lo siguiente (pueden haber más opciones).

1. Usar el índice primario de la tabla `Mascotas`.
2. Usar el índice primario de la tabla `Personas` para obtener sus atributos. Usar el índice sobre `pid` en la tabla `Mascotas` para obtener las mascotas correspondientes.
3. Usar el índice primario de la tabla `Mascotas` para obtener sus atributos. Luego, tomar el `pid` e ir a buscar la persona correspondiente con el índice primario de la tabla `Personas`.