



Control 2

Pregunta 1

1.-

```
SELECT * FROM Cervezas WHERE id = 20
```

$$I/O = 1$$

```
SELECT * FROM Cervezas WHERE id <= 3
```

$$I/O = 4$$

```
SELECT * FROM Cervezas WHERE precio > 2000 AND precio < 3000
```

$$I/O = (h - 1) + \frac{B}{250 \cdot 0.6} + B$$

B = Numero de tuplas afectadas.

```
SELECT * FROM Cervezas WHERE marca = 'Tamango Brebajes'
```

$$I/O = 2$$

```
SELECT * FROM Cervezas WHERE estilo = 'Hazy IPA'
```

$$I/O = \frac{10\,000}{80} = 125$$

2.-

```
SELECT * FROM Cervezas WHERE precio < 3000 AND marca = 'Tamango Brebajes'
```

Primero accedo mediante B+Tree al indice con aquel valor y para ese grupo aislado de tuplas reviso la condición verificando que se cumpla la marca. Terminando con un:

$$I/O = ((h - 1) + \frac{B}{250 \cdot 0.6} + B) + 2$$

Que es la suma de ambos costos individuales.

Pregunta 2

```
#Pre-procesamiento
```

```
Hacer un sort para cada relacion con respecto a b, ya que es el atributo comun.  
Ademas de esto, eliminamos las tuplas duplicadas
```

```
#Ejecucion  
#Mark: Condicion de join  
R.open()  
S.open()  
  
r := R.next()  
s := S.next()  
  
if !mark:  
    while (r < s):  
        r := R.next()  
    while (r > s):  
        s := S.next()  
    mark = s  
if (r == s)  
    result = <r, s>  
    s := S.next()  
    return result  
else:  
    resetear s a mark  
    r := R.next()  
    mark = NULL  
  
R.close()  
S.close()
```

Extraido de https://www.youtube.com/watch?v=jiWCPJtDE2c&t=73s&ab_channel=CS186Berkeley.

Pregunta 3

1.-

Primero se comienza recorriendo las tuplas de R y por cada una se recorren todas las tuplas de S. Se compara la igualdad del valor de los atributos $R.a$ y $S.a$ para cada combinación, si son iguales se agrega la tupla a la unión.

Costo Total: $\text{Costo}(R) + \text{Costo}(S) * \text{Tuplas}(R)$

2.-

Yo agregaría una indexación de tipo hash clustered en el atributo a en la relación S, ya que es un atributo que comparten ambas relaciones para optimizar el join. De esta forma se recorre la relación no indexada y se busca el join apropiado en la tabla hash generada utilizando el atributo en la función de hash para encontrar el índice complementario.

Costo Total: $\text{Costo}(R) * 2$

Ya que se recorren todas las tuplas de R y por cada una accedemos al bucket correcto sumando el costo de recuperar la información de dicha página.

3.-

Aprovechando de mejor manera el buffer, como lo hace el Block Nested Loop Join, podemos disminuir el costo de la unión sin necesitar crear nuevos índices. Lo hace manteniendo toda o la mayoría de la relación en la memoria (más tuplas), así disminuyendo las veces que se necesita acceder al disco.

Costo Total: $\text{Costo}(R) + \text{Costo}(S) * \text{Páginas}(R) / |\text{Buffer}|$

Donde $|X| = \text{Capacidad de } X$. En la fórmula disminuye por la división del buffer que será siempre mayor o igual a 1, y la utilización de páginas en vez de tuplas ya que sera siempre menor o igual al número de tuplas.