Selección de índices

Introducción

- La creación de índices implica analizar un trade-off:

1. La existencia de un índice sobre un atributo puede acelerar de manera importante la ejecución de consultas que especifican una condición sobre ese atributo y también joins que incluyen a ese atributo.
2. Cada índice construido sobre una relación hace las inserciones, borrados y actualizaciones sobre esa relación más lentas.

Un modelo simple de costo

- Para entender cómo seleccionar índices para una base de datos, primero tenemos que entender dónde se gasta el tiempo contestando una consulta.
- Simplificando, asumiremos que las tuplas están distribuidas en los distintos bloques físicos de un fichero que almacena cada relación.
- Normalmente un bloque físico tiene varios miles de bytes (8k-4k) y contiene muchas tuplas.

Un modelo simple de costo

- Como sabemos recuperar una tupla supone leer de disco el bloque físico completo, y una vez en memoria, se debe buscar dentro del bloque la tupla buscada.
- Puede ocurrir que la tupla buscada ya esté en memoria principal por la lectura del bloque físico que la contiene en una operación anterior, pero por simplicidad asumimos que esto nunca ocurre, y que cada bloque físico siempre se recupera de disco.

Algunos índices útiles

- Es muy común que el índice más útil sea sobre la clave primaria (CP) -y candidata-, de hecho los SGBD lo hacen automáticamente
- Hay 2 razones:
  - Son comunes las consultas que especifican un valor de la CP. Por lo tanto, un índice sobre la CP se usará frecuentemente.
  - Dado que hay como mucho una tupla con un valor de CP, el índice devuelve nada, o una posición de una tupla. Por lo tanto como mucho se tiene que recuperar un bloque físico (aunque puede que se tengan que recuperar otros bloques físicos en la búsqueda dentro del índice) y esto supone una importante ventaja sobre tener que buscar esa tupla secuencialmente sobre todo el fichero.

Ejemplo

- Usando las tablas EMP y DEPT, supongamos que queremos saber el nombre del departamento donde trabaja el empleado 7869.
- Una implementación ingenua debe recorrer la tabla de empleados para saber el número de departamento donde trabaja el empleado 7869, y luego recorrer la tabla DEPT para encontrar la tupla con ese número. Esto implica numerosas lecturas de bloques físicos.

Ejemplo

- Con un índice sobre EMPNO, recuperar el número de departamento donde trabaja el empleado 7869 supondría una lectura física.
- Y con un índice sobre DEPTNO, recuperar el nombre del departamento sería otra lectura física.
- Habría que añadir alguna lectura física por el uso de los índices.

Algunos índices útiles

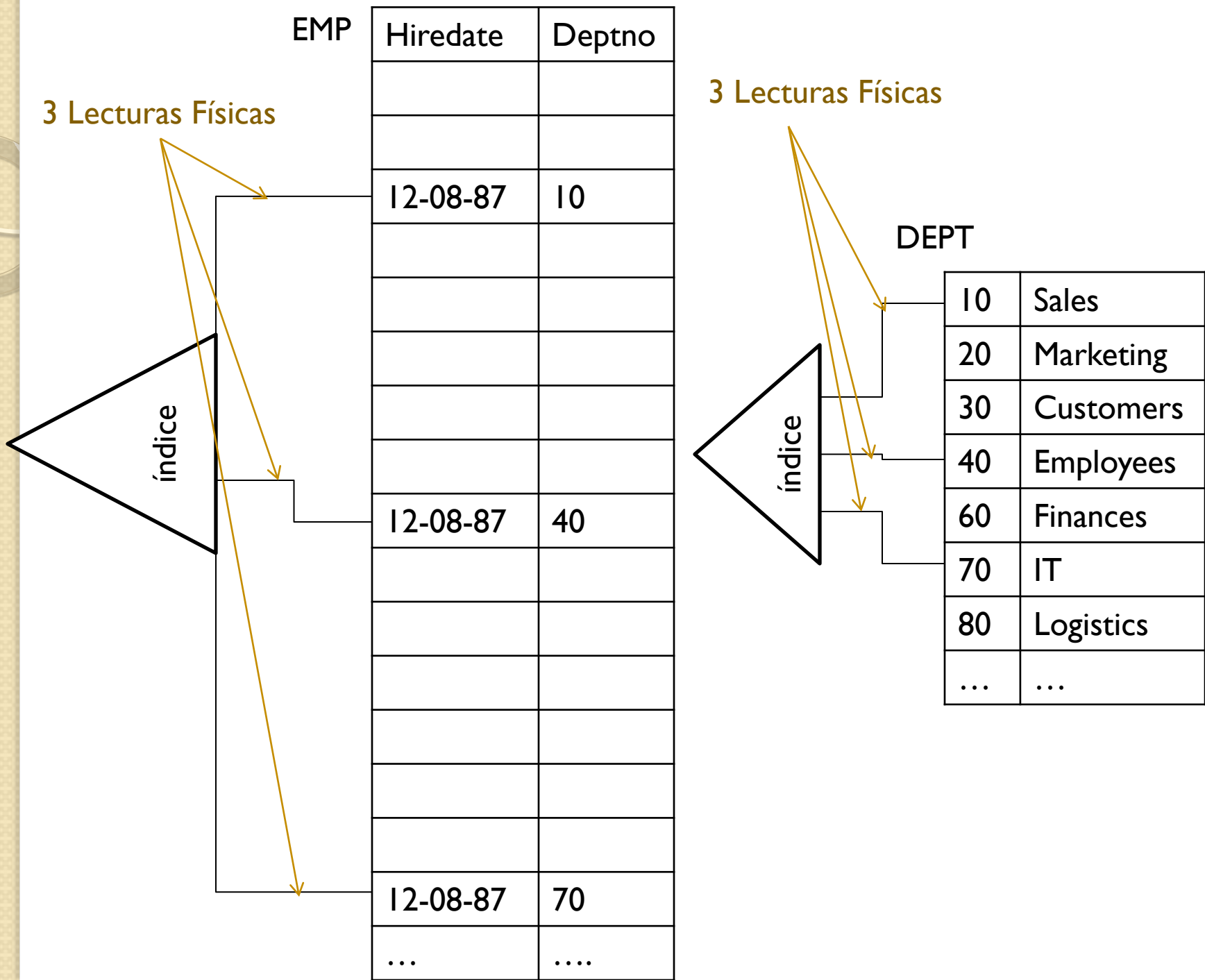
- Cuando el índice no indexa una clave, puede que mejore, o no mejore, el tiempo de recuperación.
- Hay dos situaciones donde un índice puede ser efectivo (incluso si no se creó sobre una clave):
  1. Si el atributo es casi una clave, esto es, si pocas tuplas tienen un valor dado. Aunque las tuplas que tienen un valor estén en bloques físicos distintos, tenemos que recuperar pocos bloques, y esto supone una mejora importante sobre la búsqueda secuencial sobre todo el fichero.
  2. Si las tuplas están agrupadas sobre ese atributo. Es decir tuplas con el mismo valor en un atributo se almacenan consecutivamente en los mismos bloques físicos. Aún si hay muchas tuplas, tenemos que recuperar pocos bloques físicos.

Ejemplo

- Como ejemplo del primer caso, supongamos que tenemos un índice sobre el campo HIREDATE y aunque puede haber empleados con la misma fecha de contratación, suponemos que tenemos muchos empleados, y sólo unos pocos tienen la misma fecha de contratación, sea cual fuere.

Ejemplo

- Por ejemplo si dada una fecha hay tres empleados (de muchos) que tienen la misma fecha de contratación, aunque las tuplas correspondientes a esos tres empleados estén en bloques distintos, se ahorra mucho con el uso del índice.
- Si quisiésemos saber el nombre del departamento para el cual trabajan esos tres empleados, usaríamos el índice sobre `DEPTNO` para encontrar las tres tuplas en `DEPT` con la información relevante, probablemente esto requiera traer tres bloques físicos a memoria, pero sería una importante mejora sobre traer toda la relación `DEPT` a memoria.


EMP
Hiredate
Deptno
3 Lecturas Físicas
índice
12-08-87
10
12-08-87
40
12-08-87
70
…
….
3 Lecturas Físicas
DEPT
índice
10
Sales
20
Marketing
30
Customers
40
Employees
60
Finances
70
IT
80
Logistics
…
…

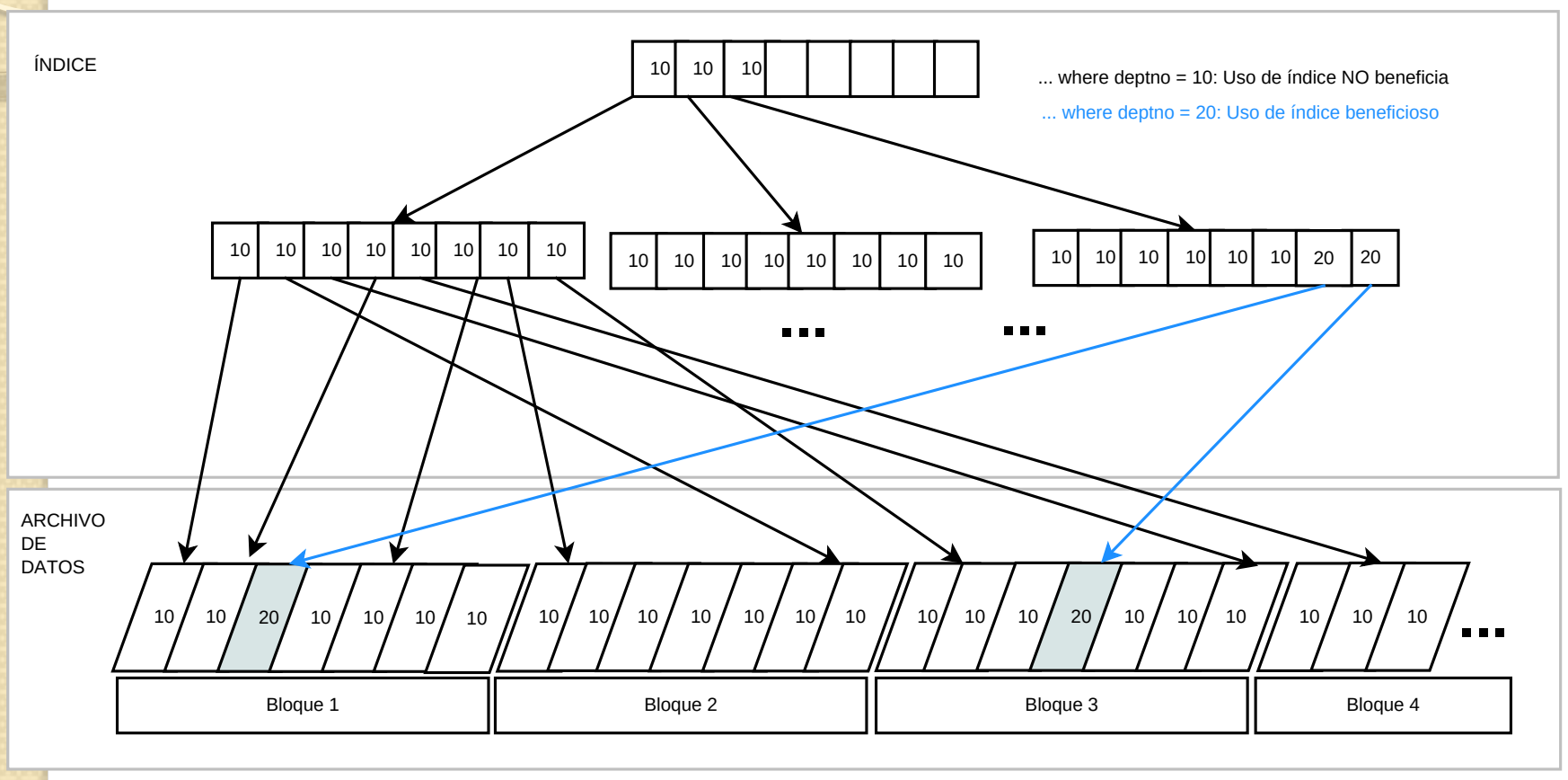
Ejemplo

- Supongamos que hacemos la siguiente consulta:

```
SELECT *
FROM emp
WHERE deptno=10
```

- Supongamos que las filas de EMP no están agrupadas por DEPTNO, digamos que están ordenadas por EMPNO.
- Esta consulta mejora poco con el índice.
- Si hay, digamos, 100 empleados por bloque físico, hay una buena posibilidad de que haya al menos un empleado del departamento 10 por cada bloque.
- Por lo tanto es necesario leer una gran parte de los bloques físicos que almacenan la tabla EMP, aunque usemos el índice, lo cual no va implicar nunca una mejora sustancial sobre la búsqueda secuencial. Es más, puede que la mejora sea nula o incluso se empeore.

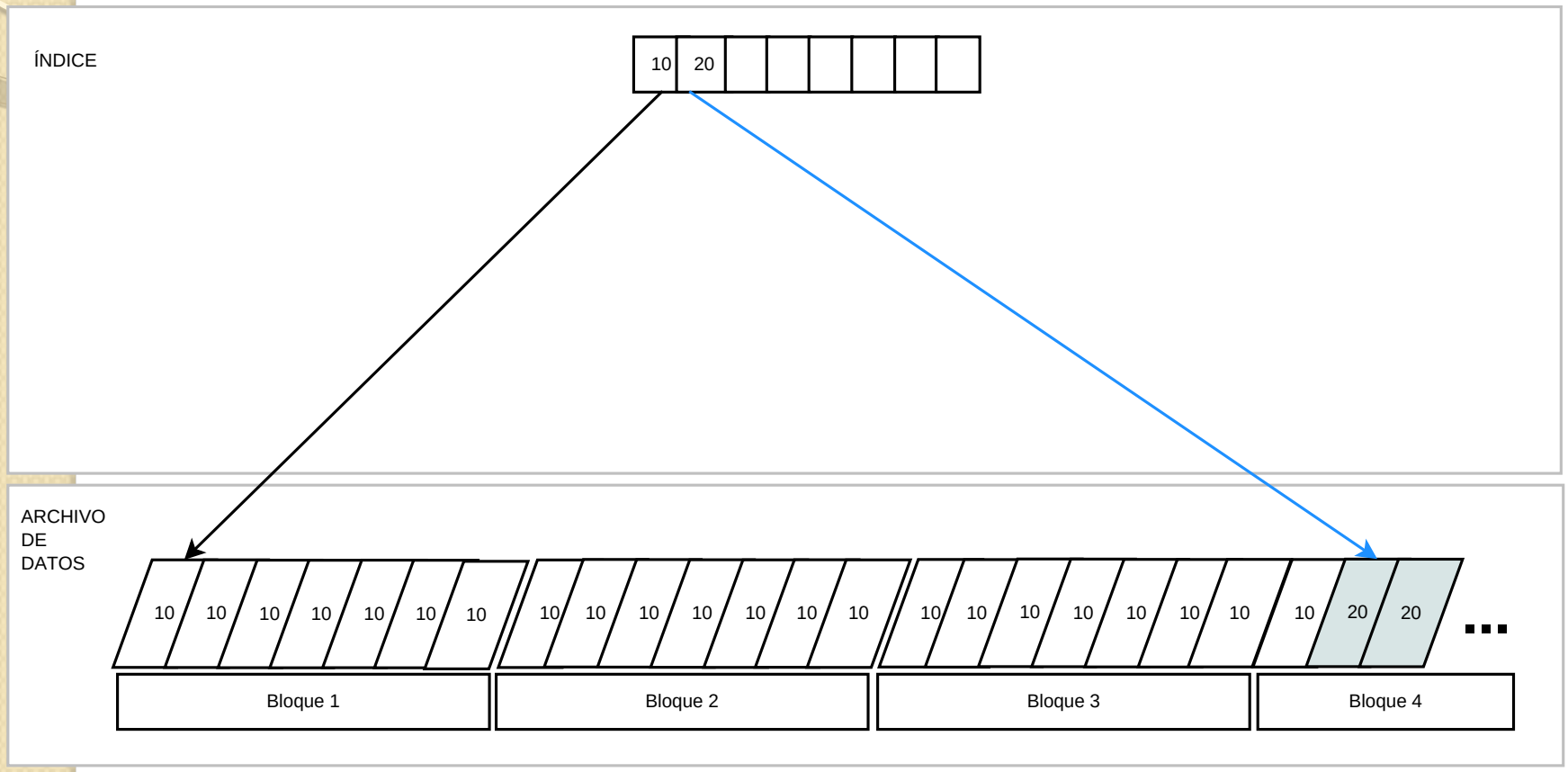
Ejemplo
Índice denso sin agrupamiento



Ejemplo

- Sin embargo supongamos que tenemos las tuplas de EMP agrupadas por departamento, es decir, tuplas que tienen el mismo valor de DEPTNO están almacenadas consecutivamente.
- Entonces podríamos usar un índice sobre DEPTNO para recuperar unos pocos bloques físicos que contienen a los empleados que tienen puesto de trabajo `10`.

Ejemplo
Índice disperso con agrupamiento



Cómo decidir qué índices crear

- Como sabemos, no podemos crear muchos índices, porque perjudicaríamos las operaciones de inserción, actualización y borrado.
- Sin embargo, seguramente valga la pena crear un índice sobre atributos que son usados frecuentemente.
- El principal coste de una consulta o modificación sobre un fichero (tabla) es el número de bloques físicos que tenemos que traernos a memoria.
- Los índices nos evitan tener que inspeccionar todos esos bloques, pero los índices también se tienen que almacenar (al menos en parte) en disco.
- Por lo tanto acceder y modificar los índices también tienen un coste en lecturas físicas.

Cómo decidir qué índices crear

- Recordar que una modificación requiere leer el bloque físico al buffer en memoria principal, y luego volver escribir ese bloque en disco, por lo tanto cuesta el doble que consultar (leer) un bloque.

Cómo decidir qué índices crear

- Para decidir qué índices usar necesitamos realizar suposiciones sobre qué consultas y actualizaciones van a ser comunes en la base de datos.
- A veces tenemos un historial de consultas pasadas, que nos pueden servir para predecir el futuro.
- Otras veces tenemos aplicaciones que podemos inspeccionar para ver las sentencias SQL que contienen.

Cómo decidir qué índices crear

Supongamos que en la tabla

`EMPPRO (EMPNO, PRONO, HOURS)`

realizamos las siguientes operaciones:

Q1: Buscamos los proyectos (número de proyecto) y número de horas dedicadas a dichos proyectos en los que trabaja un determinado empleado:

```
SELECT prono, hours
FROM emppro
WHERE empno=l;
```

Q2: Buscamos los empleados que trabajan en un determinado proyecto:

```
SELECT empno
FROM emppro
WHERE prono=p;
```

I: Insertamos una tupla en EMPPRO:

```
INSERT INTO emppro VALUES(e,p,h)
```

Cómo decidir qué índices crear

- Hagamos las siguientes suposiciones sobre los datos:

1. EMPPRO ocupa 10 bloques físicos, por lo tanto si tenemos que examinar toda la tabla el costo es 10.
2. Dado que las tuplas de un determinado empleado o un determinado proyecto están esparcidas sobre los 10 bloques físicos, incluso si tenemos un índice sobre EMPNO o PRONO, necesitaremos 3 accesos a disco para encontrar las 3 tuplas (un valor medio) correspondientes a un empleado o proyecto. Si no tenemos índice, necesitamos 10 accesos a disco.

Cómo decidir qué índices crear

4. La búsqueda en el índice de las tuplas con un valor dado del atributo indexado requiere un acceso a disco. Si un bloque físico (nodo) del índice se debe modificar, se requiere otro acceso a disco.
5. En el caso de una inserción, se requiere un acceso a disco para leer el bloque donde se debe insertar la tupla, y otro acceso para escribir en disco dicho bloque. Asumimos que incluso sin índice, podemos encontrar ese bloque sin necesidad de recorrer todo el fichero.

Cómo decidir qué índices crear

Operación	Sin índice	Índice EMPNO	Índice PRONO	Ambos índices
Q1	10	4	10	4
Q2	10	10	4	4
I	2	4	4	6
Media	**2+8p1+8p2**	**4+6p2**	**4+6p1**	**6-2p1-2p2**

- La inserción con índice requiere leer y escribir tanto la página del índice como la del fichero de datos, es decir, 4 accesos a disco.
- Si no hay índice, hay que recorrer todo el fichero.
- Si hay índice, nos cuesta un acceso a disco usar el índice, más 3 accesos recuperar tres tuplas.

Cómo decidir qué índices crear

Operación	Sin índice	Índice EMPNO	Índice PRONO	Ambos índices
Q1	10	4	10	4
Q2	10	10	4	4
I	2	4	4	6
Media	**2+8p1+8p2**	**4+6p2**	**4+6p1**	**6-2p1-2p2**

- En la fila *Media*, se asume que la frecuencia (en % sobre el total de operaciones de la BD) de la consulta Q1 es p1, la de la consulta Q2 es p2, y por tanto la frecuencia de la inserción I es 1-p1-p2.
- Dependiendo del valor de p1 y p2, una de las opciones es la mejor.

Cómo decidir qué índices crear

Operación	Sin índice	Índice EMPNO	Índice PRONO	Ambos índices
Q1	10	4	10	4
Q2	10	10	4	4
I	2	4	4	6
Media	**2+8p1+8p2**	**4+6p2**	**4+6p1**	**6-2p1-2p2**

- Por ejemplo con p1=p2=0.1, la expresión **2+8p1+8p2** es la más pequeña. Es decir, lo mejor es no crear índices.
- Si p1=p2=0.4, **6-2p1-2p2** es el mejor valor, es decir, crear índices en EMPNO y PRONO.
- Si tenemos p1=0.5 y p2=0.1, la mejor expresión es **4+6p2.**
- Con p1=0.1 y p2=0.5, la mejor es **4+6p1.**