Aulas práticas - Ficha 7

Bases de Dados (CC2005)

Objectivos: Exercícios sobre organização física de dados.

Sumário de noções básicas

Assumiremos nesta aula armazenamento de registos em ficheiro tal que o tamanho de cada registo é **fixo** com tamanho R e com **tamanho igual ou inferior** a B a dimensão de um bloco de disco em bytes.

Notação	Significado
B	tamanho de um bloco em disco em bytes
N	nº de registos em um ficheiro
R	tamanho de um registo em bytes
$bfr = \lfloor B / R \rfloor$	"blocking factor": nº de registos por bloco
$n = \lceil N / bfr \rceil$	nº de blocos em disco ocupados pelo ficheiro

Em todos os exercícios assuma $B = 4096$ bytes (tamanho de um bloco em disco).

Tabela de valores logarítmicos

$x \in]2^{k-1}, 2^k]$	$k = \lceil \log_2(x) \rceil$
$]1, 2]$	1
$]2, 4]$	2
$]4, 8]$	3
$]8, 16]$	4
$]16, 32]$	5
$]32, 64]$	6
$]64, 128]$	7
$]128, 256]$	8
$]256, 512]$	9
$]512, 1024]$	10
$]1024, 2048]$	11
$]2048, 4096]$	12
$]4096, 8192]$	13
$]8192, 16384]$	14

1. Ficheiros de tabelas

Considere o armazenamento de duas tabelas T_1 e T_2 com a seguinte caracterização:

$$T_1 : \quad N = 10,000 \quad R = 32$$

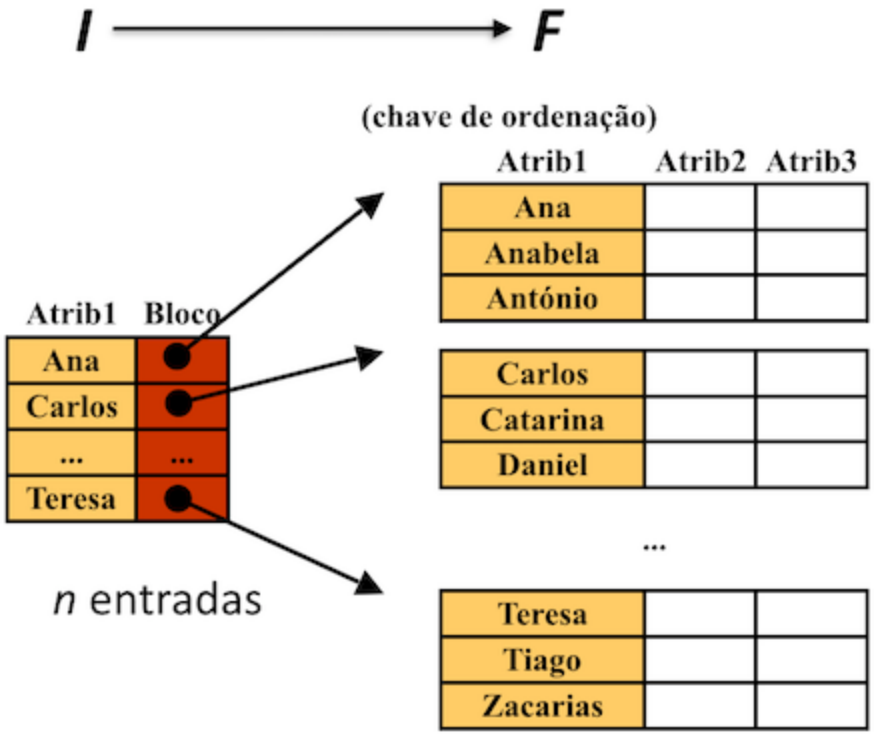
$$T_2 : \quad N = 100,000 \quad R = 130$$

Para cada um dos casos:

1. Calcule o "blocking factor" bfr, nº de bytes não usados por bloco $B - bfr \times R$, e nº de blocos n necessários a guardar os dados da tabela em disco.

2. Suponha que os ficheiros de T_1 e T_2 estão ordenados pelos valores dos atributos que constituem a chave primária de cada tabela. Quantos acessos a blocos de disco no máximo precisamos de aceder para localizar um registo a partir de um valor da chave primária nos dois casos? E no caso de a pesquisa ser baseada no valor de um outro atributo?

2. Índices primários



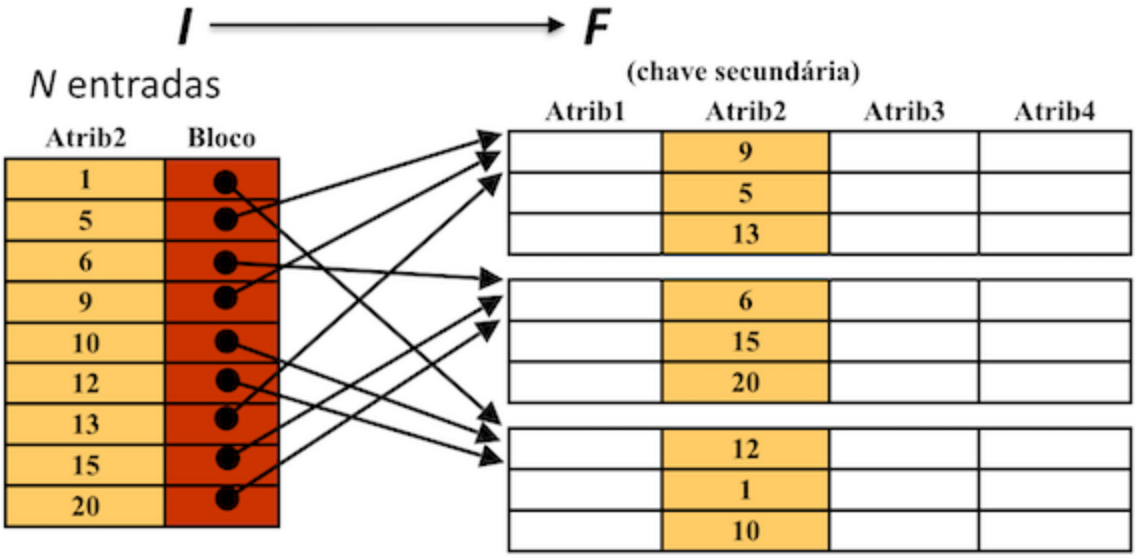
Considere a construção de índices primários para tabelas cujos ficheiros são ordenados pelos atributos de chave primária nos seguintes dois casos:

- tabela T_1 ocupando $n = 100$ blocos em disco e com tamanho de 4 bytes para um valor da chave primária;
- tabela T_2 ocupando $n = 10000$ blocos em disco e com tamanho de 12 bytes para um valor da chave primária.

Assumindo também que referências para blocos em disco têm um tamanho de 8 bytes, calcule em cada caso:

- quantos blocos são necessários para armazenar os índices;
- quantos acessos a blocos de disco são no máximo necessários para localizar registos da tabela usando o índice;
- quantos acessos em comparação seriam no máximo necessários sem o índice.

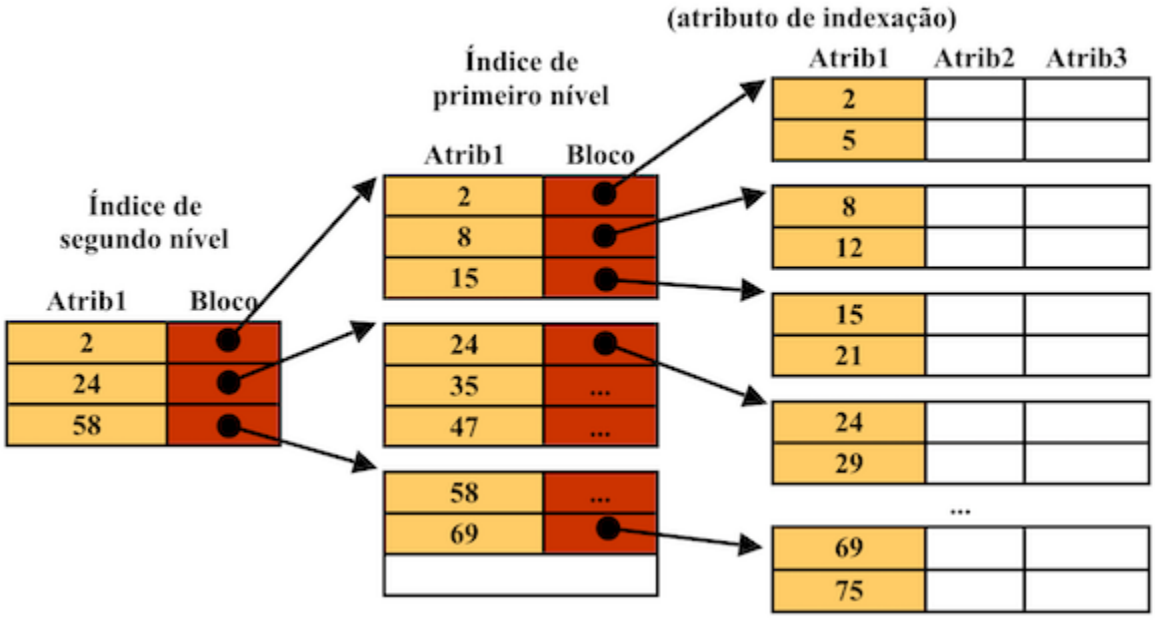
3. Índices sobre chaves secundárias



Considere uma tabela contendo $N = 25,000$ registos armazenados em $n = 400$ blocos de disco.

Repita a análise do exercício anterior, mas para um índice sobre uma chave secundária com tamanho de 8 bytes e considerando de novo que referências para blocos em disco ocupam 8 bytes.

4. Índices multi-nível



Considere uma tabela T ocupando $n = 256,000$ blocos em disco e com tamanho de 8 bytes para um valor da chave primária.

Assuma que apontadores para blocos em disco tomam 8 bytes e considere um índice primário de 1º nível, e ainda um de 2º nível sobre este. Quantos blocos em disco seriam adicionalmente necessários em cada nível e qual seria a melhoria de desempenho no acesso a disco usando o índice multi-nível

resultante?

5. Uso de índices em SQL (BD de recintos)

Use o ficheiro Recintos.db de recintos culturais e o programa Python benchmark.py na resolução das questões a seguir.

5.1

Considere a consulta

```
SELECT * FROM concelhos WHERE cod=9999;
```

Esta consulta não devolve qualquer resultado, pois não existe concelho com o código 9999. O interesse aqui será perceber de que forma será executada. Para tal use o comando `EXPLAIN QUERY PLAN` para perceber o plano de execução na consola sqlite ou no SQLiteStudio

```
EXPLAIN QUERY PLAN SELECT * FROM concelhos WHERE cod=9999;
```

Deverá observar que o plano de consulta refere o uso de um índice para a execução. Este foi criado automaticamente pelo SQLite para a chave primária `cod` da tabela.

De seguida execute o programa `benchmark.py` para medir o desempenho da execução consulta, por exemplo da seguinte forma:

```
$ python3 benchmark.py 100000 "SELECT * FROM concelhos WHERE cod=9999"
100000 queries in 1.25 seconds :: 80028 queries/second
```

5.2

Repita o processo da alínea anterior agora para a consulta

```
EXPLAIN QUERY PLAN SELECT * FROM concelhos WHERE designacao='XYZ';
```

Observe que o plano de execução (via `EXPLAIN QUERY PLAN`) inclui agora um "scan" da tabela em vez do uso de índice. Observe também que o desempenho (via `benchmark.py`) deverá ser inferior ao da alínea anterior.

5.3

Usando `CREATE INDEX`, crie um índice para o campo `designacao` na tabela `concelhos`.

Repita depois o processo da alínea anterior e observe as diferenças no plano de execução e no seu desempenho.