TÉCNICO LISBOA

Bases de Dados

T21 - Índices Parte II

Prof. Daniel Faria
Prof. Flávio Martins

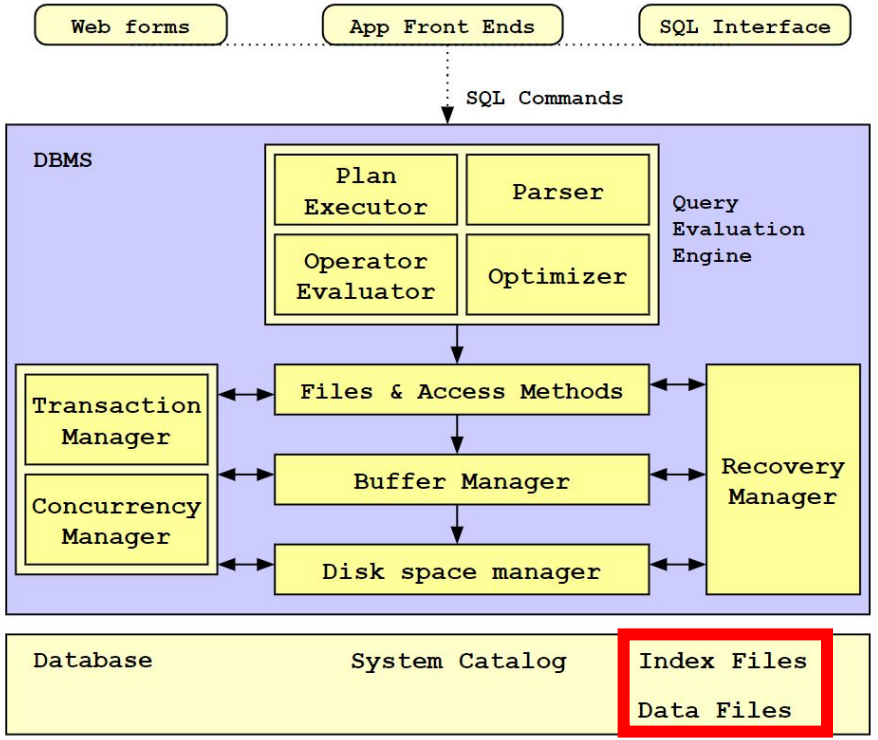
Sumário

- Índices: Desenho Físico
- Seletividade de Interrogações
- Índices e Chaves
- Interrogações e Índices
- Interrogações Apenas com Índices

TÉCNICO LISBOA

Índices: Desenho Físico

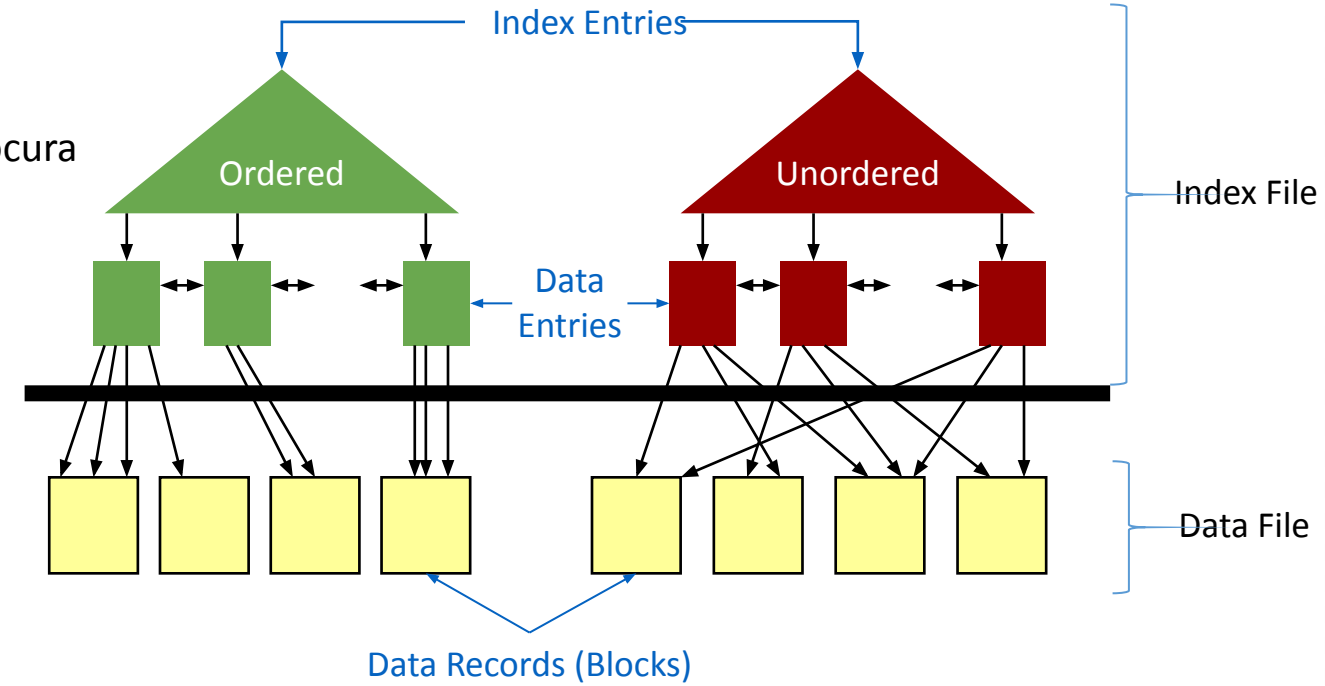
Componentes de um SGBD



Índices

Index Entries:

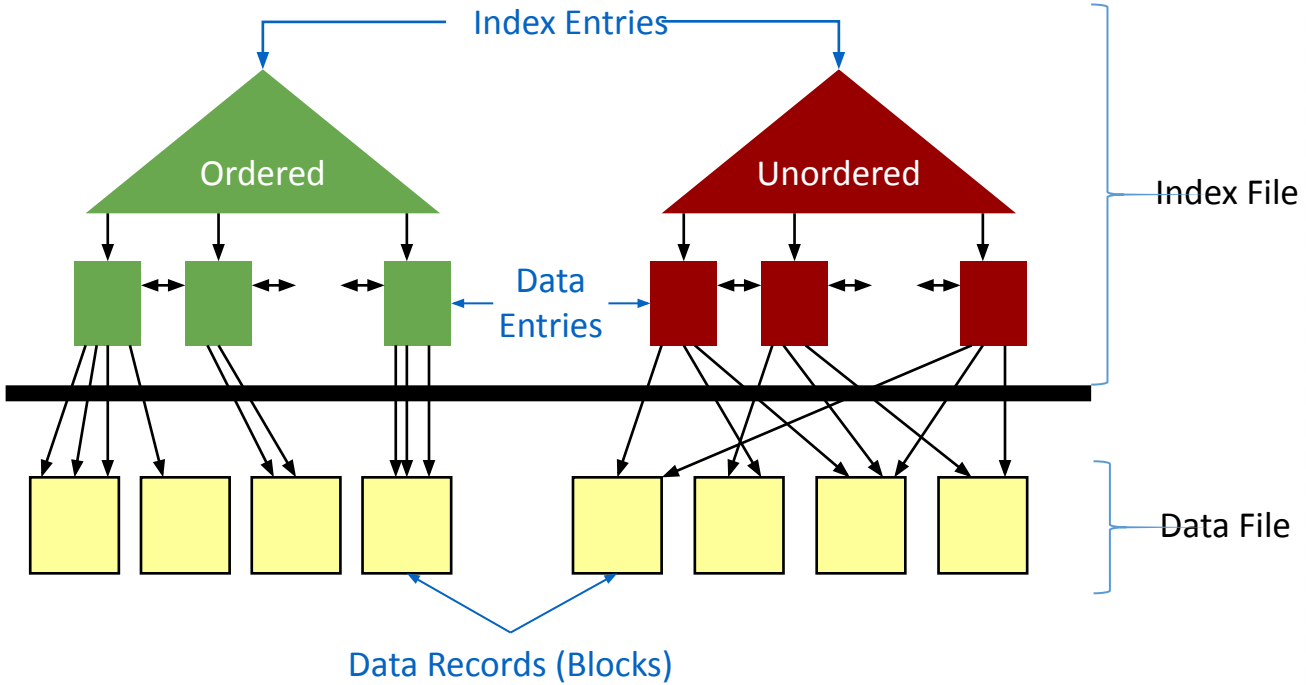
- Definem algoritmo de procura da entrada com a chave
  - Sequencial
  - Hash Table
  - B-tree
  - BitMap



Índices

Data Entries:

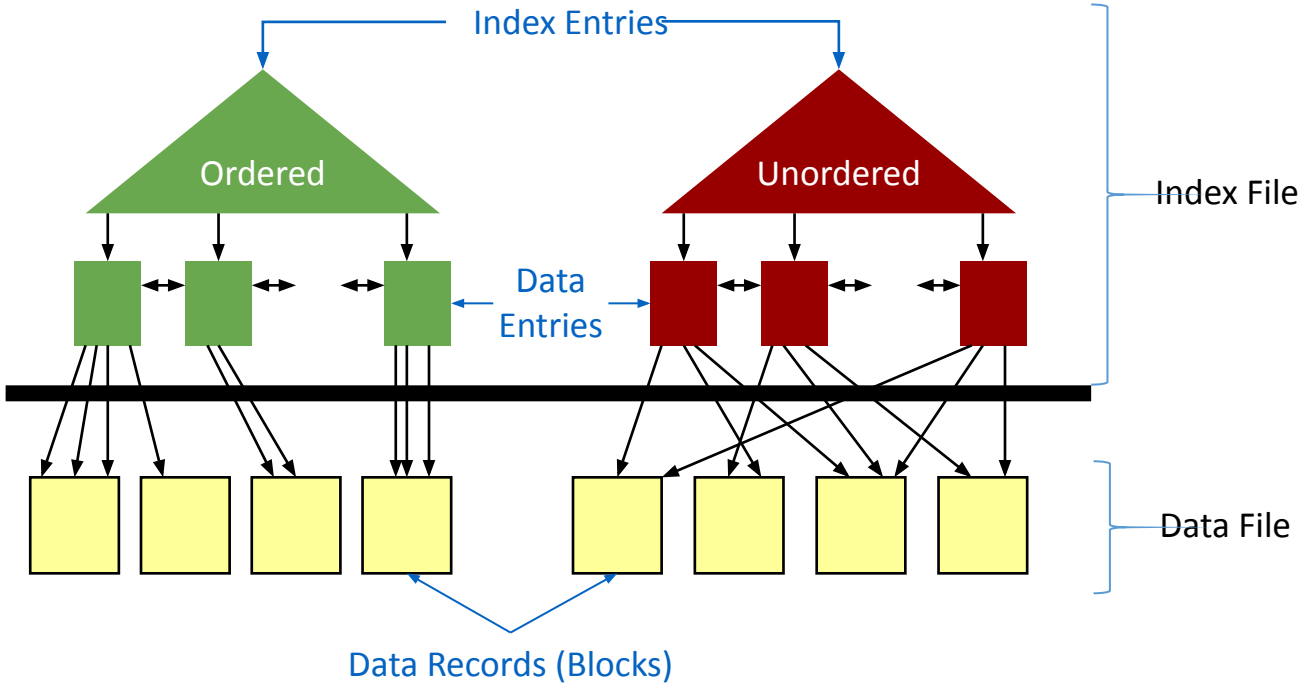
- Contêm o endereço dos blocos de dados no disco
  - O data record com chave *k*, *r*(*k*)
  - <*k*, block(*r*(*k*))>
  - <*k*, {block(*r*(*k*))}>



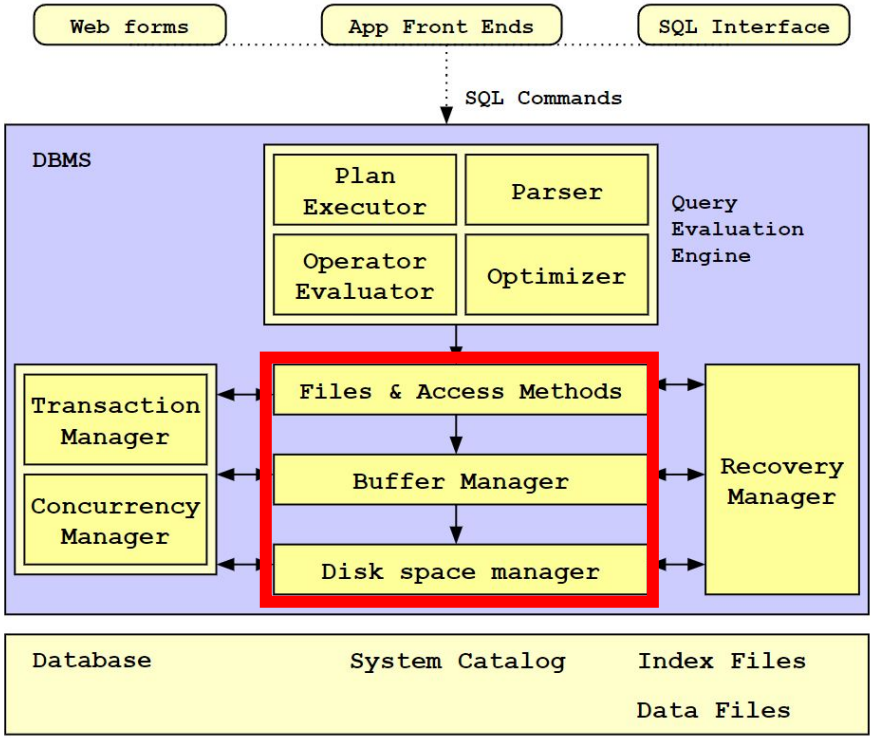
Índices

Data Records:

- Contêm os dados propriamente ditos
- Podem estar ou não ordenados fisicamente pela chave *k*
  - Se estiverem ordenados diz-se que a tabela está *clustered* por *k*



Componentes de um SGBD



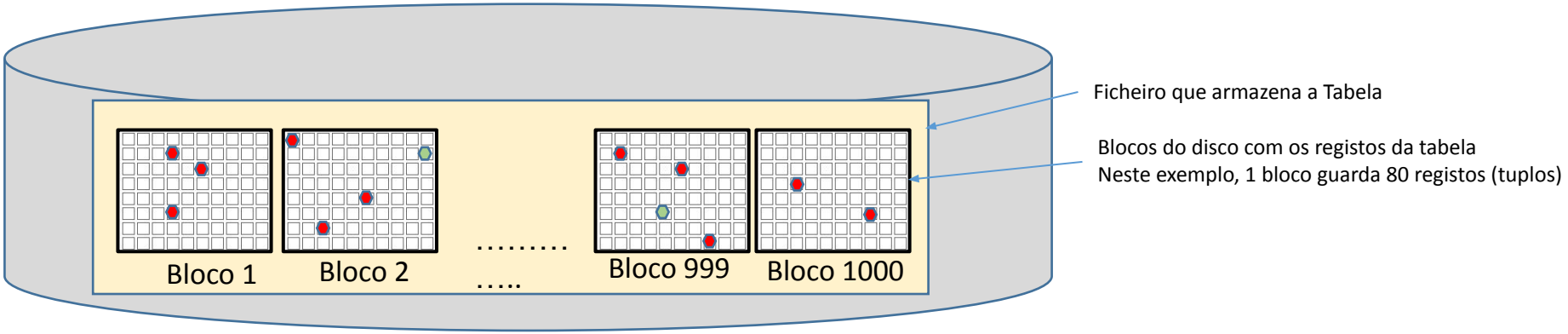
TÉCNICO LISBOA

Seletividade de Interrogações

Seletividade de uma Query

- Indica, para uma tabela, a % dos registos que serão incluídos na resposta à query
- Quanto mais elevada for a seletividade menor é a % de registos (e menos operações de transferência de dados do sistema de armazenamento e processamento)
    - Uma baixa seletividade reduz a utilidade dos índices
- Valor depende do número médio de registos que cabem num bloco de disco e do nº de blocos a transferir

Seletividade, Blocos e Registos



- Quantos blocos teriam que ser lidos do disco para memória se a query devolver os registos vermelhos? E se for os verdes?
- O principal fator da otimização que os índices trazem é a redução do nº de blocos que têm que ser lidos do disco para a memória.
- Ou seja, a probabilidade de um bloco conter algum registo a incluir na resposta é o principal fator de decisão na criação de um índice.

Seletividade de uma Query

Exemplo:

- Considere-se que:
  - Os blocos do disco são de 4KB
  - Cada registo tem em ~100 bytes (cada bloco armazena ~40 registos)
  - A tabela tem 1 milhão de registos contidos em 25,000 blocos
  - Registos não estão ordenados pelo critério de selecção
- Numa query de seletividade de 5%, quantos blocos esperamos ler do disco ?
  - Se as respostas estiverem distribuídas uniformemente, a probabilidade de um bloco não ter respostas é de $0{,}95^{40} \approx 13\%$
  - Teremos de ler 87% dos blocos, ou seja 21,750 blocos

Seletividade de uma Query

Exemplo:

- Considere-se que:
  - Os blocos do disco são de 4KB
  - Cada registo tem em ~100 bytes (cada bloco armazena ~40 registos)
  - A tabela tem 1 milhão de registos contidos em 25,000 blocos
  - Registos não estão ordenados pelo critério de selecção
- Numa query de seletividade de 1%, quantos blocos esperamos ler do disco?
  - Se as respostas estiverem distribuídas uniformemente, a probabilidade de um bloco não ter respostas é de $0{,}99^{40} \approx 67\%$
  - Teremos de ler 33% dos blocos, ou seja 16,750 blocos

TÉCNICO LISBOA

Índices e Chaves

Índices e Chaves

- Declarações de PRIMARY KEY e UNIQUE criam implicitamente índices
  - Em ambos os casos são índices UNIQUE (e portanto do tipo B-tree)
  - No caso da PRIMARY KEY há ainda uma restrição NOT NULL (que nada tem a ver com o índice)
    - Criar um índice UNIQUE para “NULLS NOT DISTINCT” continua a permitir um único NULL
- Declarações de FOREIGN KEY em PostgreSQL requerem que haja um índice sobre a(s) coluna(s) referenciadas
  - Não é criado implicitamente um índice sobre a(s) coluna(s) referenciadoras, mas é prática recomendada criar um explicitamente

Análise de Execução de Queries

Explain (versão abreviada)

```
EXPLAIN [ANALYSE] statement
```

- Ferramenta indispensável para suportar decisões de criação de índices
  - Apresenta o plano de execução da *statement* (query)
    - Com a opção ANALYSE (ou ANALYZE) apresenta também o tempo estimado de execução
  - Deve ser usado em transação (com rollback) sempre que se quer analisar operações de escrita sem alterar a base de dados

Índices e Chaves Estrangeiras

Exemplo: Criemos duas tabelas com grande volume de dados e chave estrangeira

```
CREATE TABLE a(
    a_id    int PRIMARY KEY);

INSERT INTO a SELECT x FROM GENERATE_SERIES(1, 5000000) AS x;

CREATE TABLE b(
    b_id    int,
    a_id    int REFERENCES a(a_id) ON UPDATE CASCADE ON DELETE CASCADE);

INSERT INTO b SELECT a_id,a_id FROM a;
```

Índices e Chaves Estrangeiras

```
EXPLAIN ANALYSE DELETE FROM a WHERE a_id = 10;
QUERY PLAN
------------------------------------------------------------------------------
 Delete on a  (cost=0.43..8.45 rows=1 width=6)
    (actual time=0.263..0.263 rows=0 loops=1)
   ->  Index Scan using a_pkey on a  (cost=0.43..8.45 rows=1 width=6)
    (actual time=0.245..0.246 rows=1 loops=1)
         Index Cond: (a_id = 10)
 Planning time: 5.350 ms
 Trigger for constraint b_a_id_fkey: time=301.526 calls=1
 Execution time: 301.811 ms
(6 rows)
```

Chegar ao registo a_id = 10 foi através do índice da PK. Mas para apagar o registo, é preciso validar que não existe nenhum registo na tabela B que o referencie, o que implica um scan da tabela B.

Índices e Chaves Estrangeiras

```
CREATE INDEX idx_b ON b (a_id);
EXPLAIN ANALYSE DELETE FROM a WHERE a_id = 10;
QUERY PLAN
------------------------------------------------------------------------
 Delete on a  (cost=0.43..8.45 rows=1 width=6)
    (actual time=0.037..0.037 rows=0 loops=1)
   ->  Index Scan using a_pkey on a  (cost=0.43..8.45 rows=1 width=6)
    (actual time=0.037..0.037 rows=0 loops=1)
         Index Cond: (a_id = 11)
 Planning time: 0.062 ms
 Execution time: 0.054 ms
(5 rows)
```

Criando um índice para a chave estrangeira de B para A, reduzimos o tempo por um factor de 1000

TÉCNICO LISBOA

Interrogações e Índices

Interrogações sem Condições

Exemplo: Consideremos a tabela *Employee* com PK *eID*

```
SELECT * FROM Employee;
```

- Esta interrogação não envolve condições de filtragem ou agrupamento, portanto não beneficia da criação de um novo índice

```
SELECT name, age FROM Employee1
UNION ALL
SELECT name, age FROM Employee2;
```

- Esta interrogação também não envolve condições de filtragem ou agrupamento (devido ao ALL), portanto não beneficia da criação de um novo índice

Interrogações e Índices

```
SELECT * FROM Employee JOIN Skill USING (eID)
    WHERE skill_name = 'SQL';
```

- Como *eID* é PK em ambas as tabelas, não é necessário nenhum índice adicional para o join; mas o filtro por *skill_name* beneficiaria de um índice.

```
SELECT name, age FROM Employee1
UNION
SELECT name, age FROM Employee2;
```

- Neste caso os duplicados são removidos, o que envolve sorting, pelo que beneficiaria de um índice em (*name, age*) em ambas as tabelas.

Interrogações e Índices

```
SELECT e.name, d.mgr FROM Employee e JOIN Department d USING (dno)
  WHERE e.salary BETWEEN 10000 AND 20000 AND e.hobby='Stamps';
```

- Assumindo que *dno* é chave primária em *Department*, não é necessário mais índices para o join
- Que mais índices podemos criar em Employee para otimizar a interrogação?
  - B-tree em *salary*
  - Hash em *hobby*
    - Apenas um deles é necessário, e determinar qual é melhor requer determinar a seletividade das condições

Interrogações e Índices

```
SELECT name, age FROM Employee WHERE name LIKE '%Greg%';
```

- Como % pode ser qualquer caracter, um eventual índice em *name* não seria usado nesta interrogação

```
SELECT name, age FROM Employee WHERE name LIKE 'Greg%';
```

- Aqui um índice em *name* seria utilizado, pois % só aparece no fim da string (sabemos qual o prefixo)

Interrogações e Índices

```
SELECT * FROM Employee WHERE b = 5 AND c > 10 AND d = 15 AND e <= 20;
```

- Há igualdade nos campos *b* e *d*, pelo que devem ser os primeiros de um eventual índice
- Há a operação de range sobre *c* e *e*, mas apenas devemos colocar um dos campos no filtro
- Portanto temos as seguintes hipóteses de índices (todos B-tree) a escolher em função da seletividade de cada coluna:

```
CREATE INDEX idx ON tblEmployee(b,d,c);
CREATE INDEX idx ON tblEmployee(b,d,e);
CREATE INDEX idx ON tblEmployee(d,b,c);
CREATE INDEX idx ON tblEmployee(d,b,e);
```

Interrogações e Índices

```
SELECT * FROM Employee WHERE skill_name = 'SQL' AND level = 10 AND salary = 50000;
```

- Aqui, temos igualdade em 3 variáveis. Devemos criar o índice por ordem de seletividade de cada condição. Por exemplo, podemos contar os valores distintos de cada variável.

```
SELECT COUNT(DISTINCT skill_name) AS skills, COUNT(DISTINCT level) AS levels,
COUNT(DISTINCT salary) AS salaries FROM Employee;
 skills | levels   | salaries
--------+----------+----------
  5000  | 8000000 |  3000
```

- Indica-nos que *level* deve ser o primeiro atributo do índice

TÉCNICO LISBOA

Interrogações e Índices

```
SELECT * FROM Employee WHERE skill_name = 'SQL' AND level = 10 AND salary = 50000;
```

- Sendo level o primeiro atributo do filtro, temos de testar a seletividade dos outros dois (para cada valor do primeiro)

```
SELECT COUNT(DISTINCT skill_name) AS skills, COUNT(DISTINCT salary) AS salaries
FROM Employee WHERE level = 10;
 skills | salaries
--------+-----------
     60 |     2000
```

- Indica-nos que *salary* deve ser o segundo atributo do índice

```
CREATE INDEX emp_x ON employee (level,salary,skill);
```

TÉCNICO LISBOA

Interrogações e Índices

```
SELECT * FROM Employee WHERE b = 5 AND e < 10 AND f = 15 ORDER BY d,c;
```

- O índice deverá começar (à partida) por (b, f) ou (f, b)
  - O atributo *e* pode ser inserido como último elemento do índice (B-tree)
- Para melhorar o desempenho de "ORDER BY", devemos inserir (d, c) no índice, mas não pode ser (c, d)
- Pelo que temos 2 alternativas:

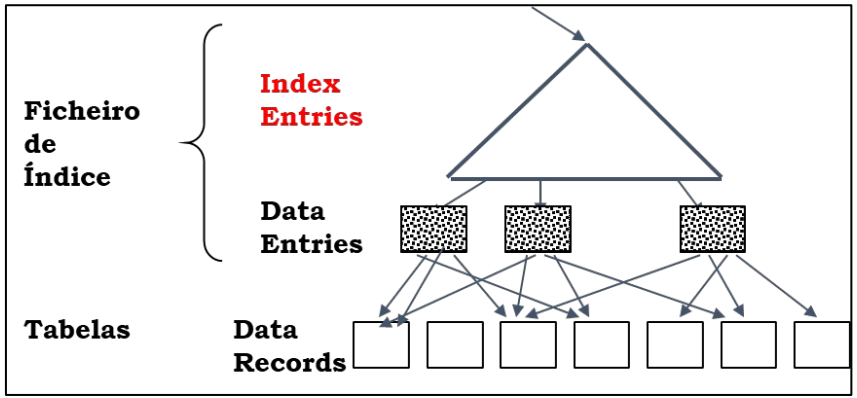
```
CREATE INDEX emp_y ON Employee(b,f,e)
--ou
CREATE INDEX emp_y ON Employee(b,f,d,c)
```

TÉCNICO LISBOA

Interrogações Apenas com Índices

Interrogações Apenas com Índices

- Há muitas consultas que podem ser respondidas apenas com informação do próprio índice, i.e., não requerem consultar a tabela
  - A opção INCLUDE no CREATE INDEX do PostgreSQL permite alargar este leque de consultas, incluindo colunas adicionais no índice que não são usadas como chaves
    - Trade-off: espaço de armazenamento vs. tempo de consulta



Interrogações Apenas com Índices

Exemplos:

E(dno)
D(mgr)

```
SELECT mgr FROM Department JOIN
Employee ON dno=mgr;
```

E(dno,eid)
D(mgr,eid)

```
SELECT mgr, eid FROM Department
JOIN Employee ON dno=mgr;
```

E(dno)

```
SELECT dno, COUNT(*)
FROM Employee GROUP BY dno;
```

E(age,sal)
ou
E(sal,age)

```
SELECT AVG(salary) FROM
Employee WHERE age=25 AND
salary BETWEEN 3000 AND 5000;
```

E(dno,sal)

```
SELECT dno, MIN(salary) FROM
Employee GROUP BY dno;
```

TÉCNICO LISBOA

TÉCNICO LISBOA