UNIVERSIDADE FEDERAL DE ITAJUBÁ

Banco de Dados II

COM 231

Indexação

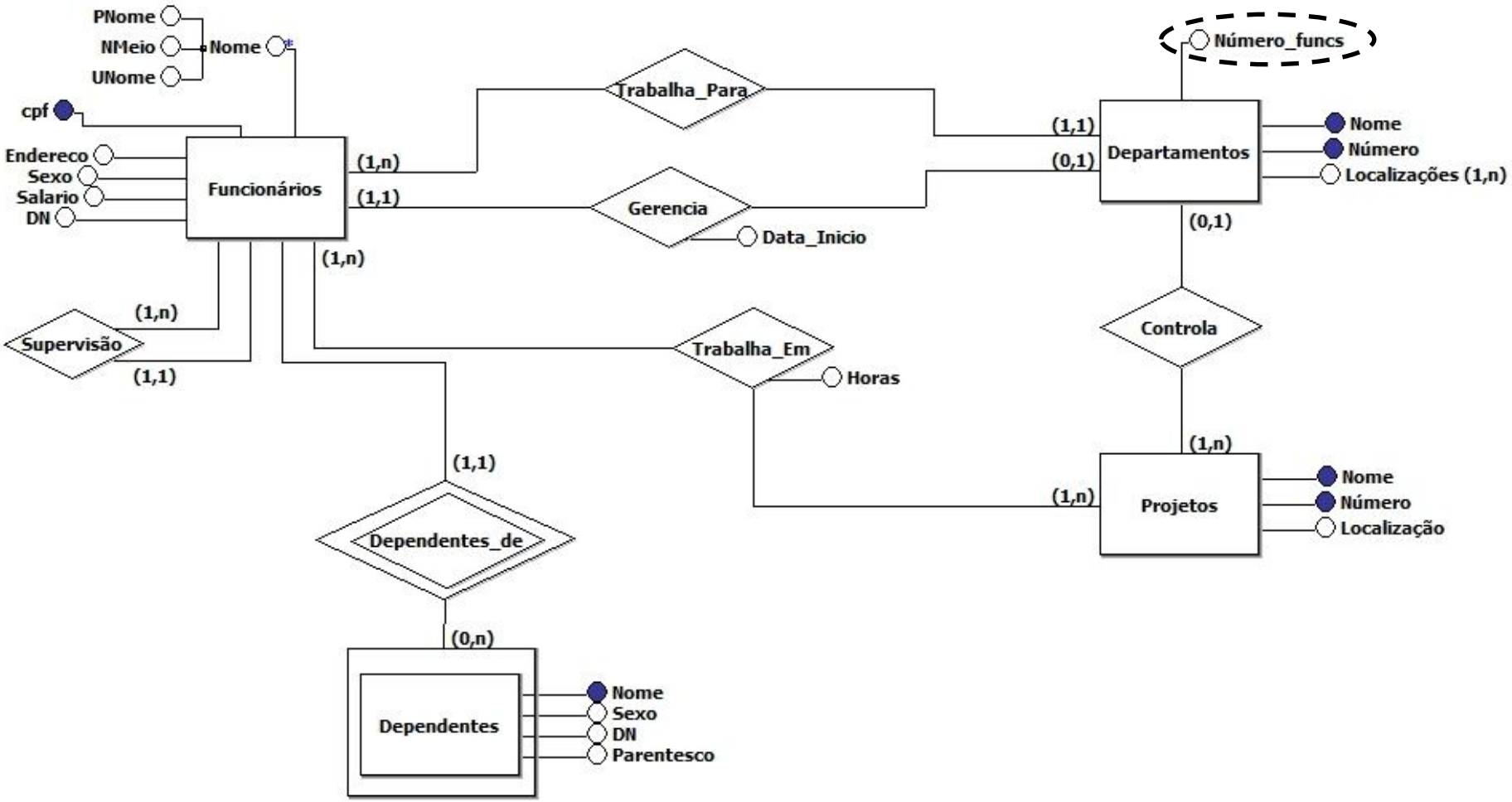
Vanessa Cristina Oliveira de Souza

Introdução

- No nível conceitual ou lógico, o banco de dados no modelo relacional, foi visto como uma coleção de tabelas.

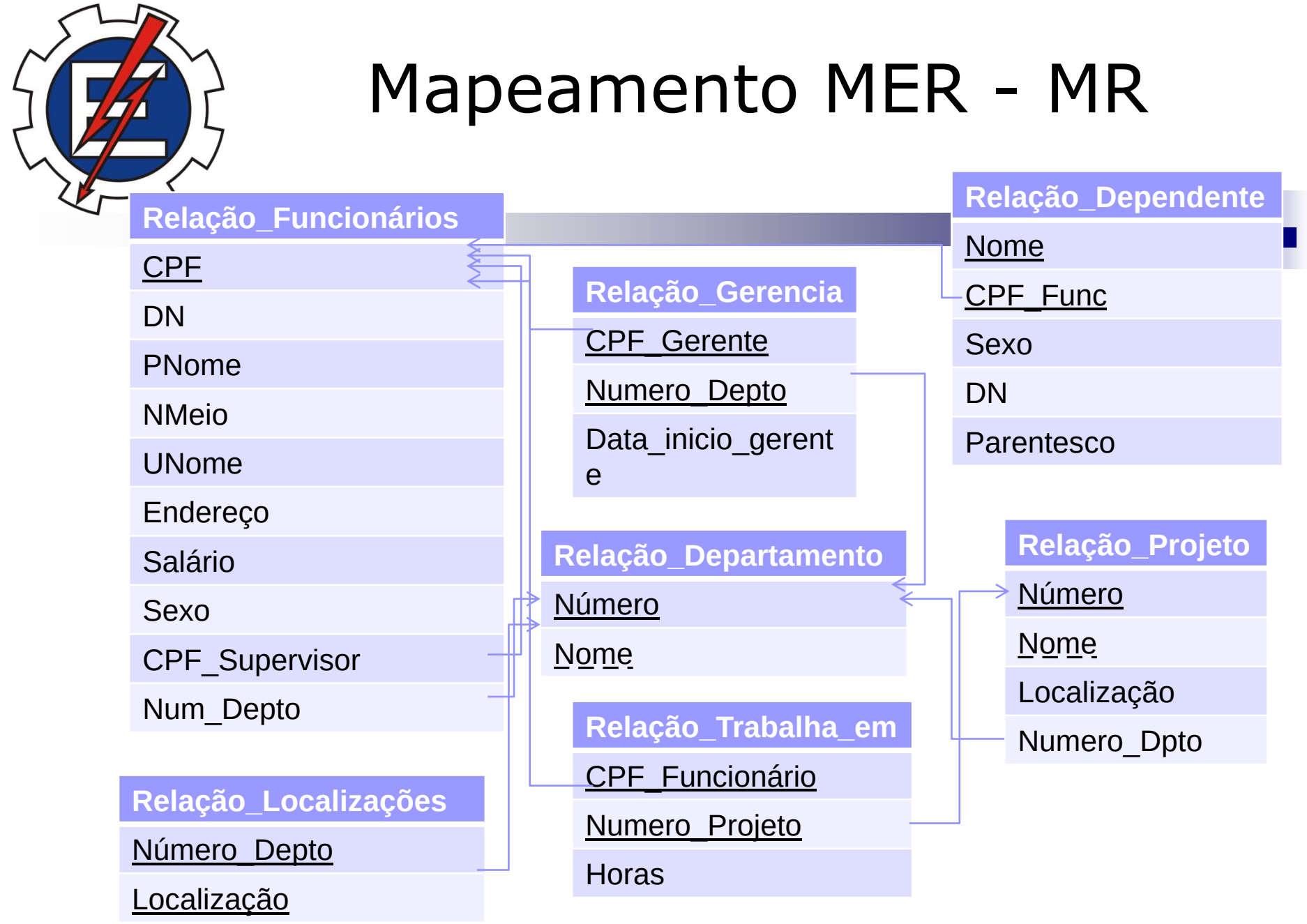
- O modelo lógico de um banco de dados é o nível correto para o usuário do banco de dados focalizar.

Mapeamento MER - MR



Fonte : Elsmari & Navathe – 6ª Edição

Mapeamento MER - MR



Mapeamento MER – MR
Outra 'nomenclatura'

Relacao_Departamentos(Número, Nome)

Relacao_Localizações (Numero_Depto, Localização)

Num_Depto REFERENCIA Relacao_Departamentos (Numero)

Relacao_Funcionarios(CPF, DN, Pnome, Nmeio, Unome, Endereço, Salário, Sexo, CPF_Supervisor, Num_Depto)

CPF_Supervisor REFERENCIA Relacao_Funcionarios (CPF)

Num_Depto REFERENCIA Relacao_Departamentos (Numero)

Relacao_Dependentes(CPF_Func, Nome, DN, Sexo, Parentesco)

CPF_Func REFERENCIA Relacao_Funcionarios (CPF)

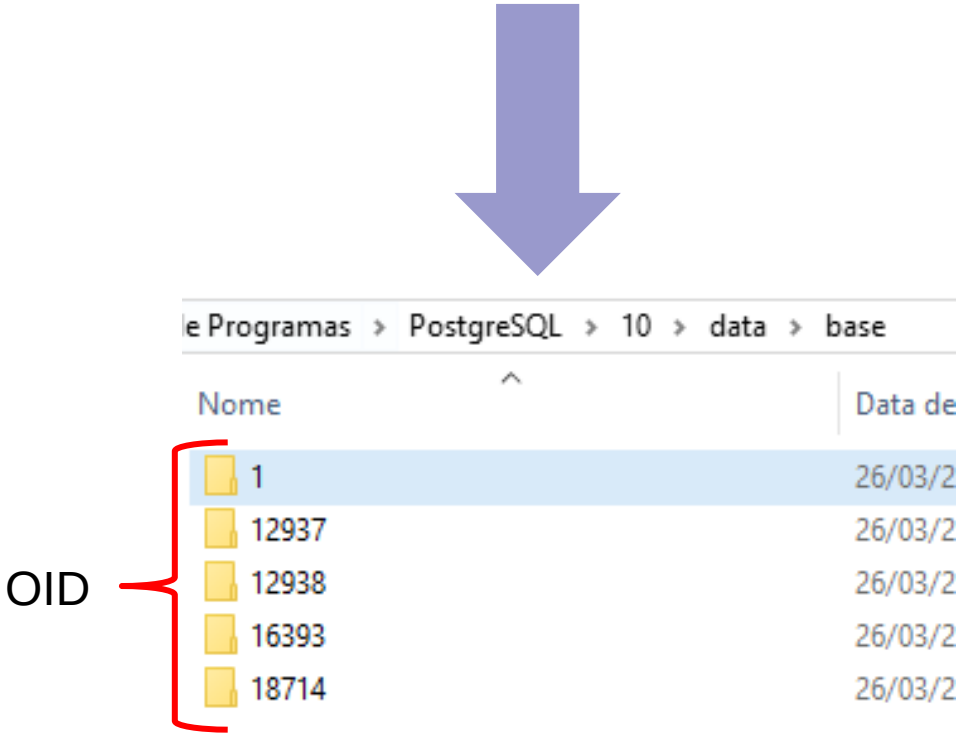
Relacao_Gerencia (CPF_Func, Numero_Depto, Data_inicio_gerente)

CPF_Func REFERENCIA Relacao_Funcionarios (CPF)

Num_Depto REFERENCIA Relacao_Departamentos (Numero)

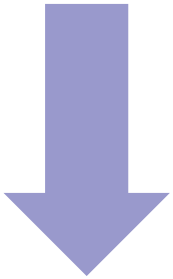
Modelo Físico

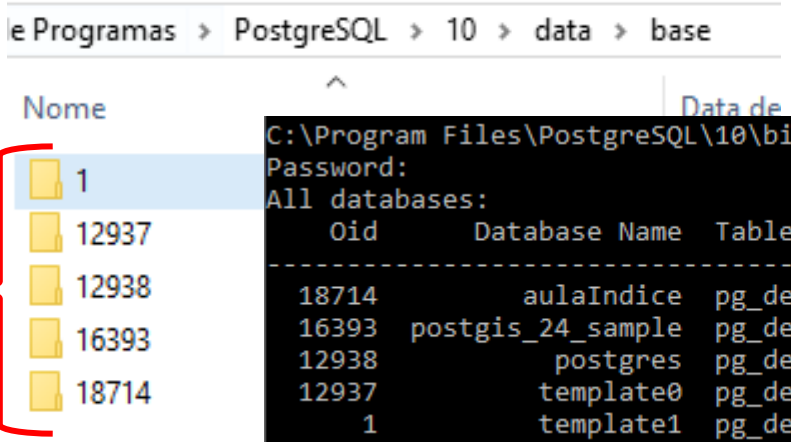
```
CREATE DATABASE aulaIndice;
```



Modelo Físico

```
CREATE DATABASE aulaIndice;
```



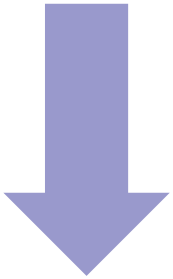


Modelo Físico

```
CREATE TABLE departamento(
    numeroDepto int,
    nomeDepto varchar(30) not null,
    primary key(numeroDepto)
);
```

Modelo Físico

```
CREATE TABLE departamento(
    numeroDepto int,
    nomeDepto varchar(30) not null,
    primary key(numeroDepto)
);
```



```
select pg_relation_filepath('departamento')
```

Output | Explain | Messages | Query History

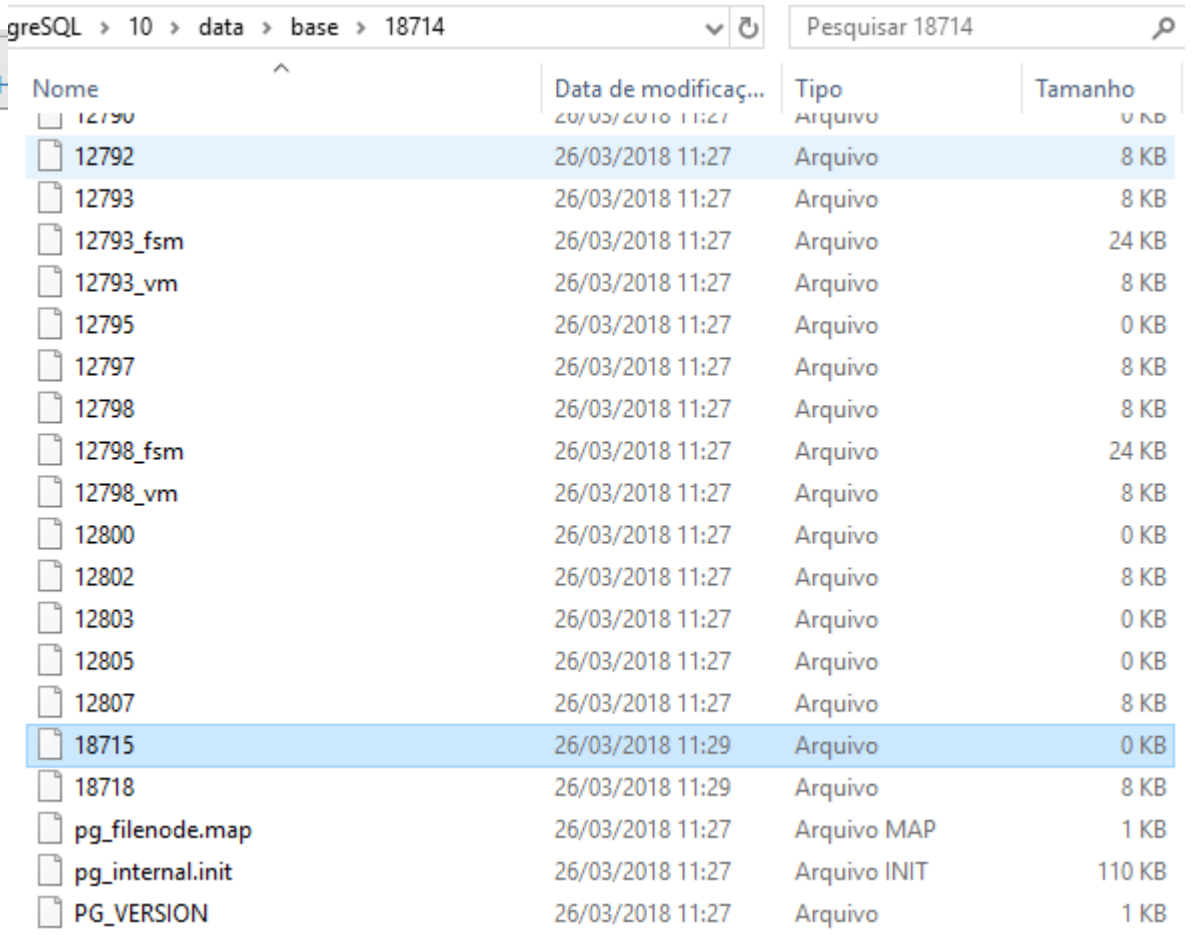
pg_relation_filepath text
base/18714/18715

Modelo Físico

```
select pg_relation_filepath('departamento')
```

Output | Explain | Messages | Query H

pg_relation_filepath text
base/18714/18715



Modelo Físico

```
INSERT INTO departamento VALUES(1, 'Recursos Humanos');
INSERT INTO departamento VALUES(2, 'Producao');
INSERT INTO departamento VALUES(3, 'Financeiro');
INSERT INTO departamento VALUES(4, 'Almoxarifado');
```

- Organização de registros em arquivos
  - Organização de arquivos em *heap*
  - Organização sequencial de arquivos
  - Organização de arquivos de *hashing*

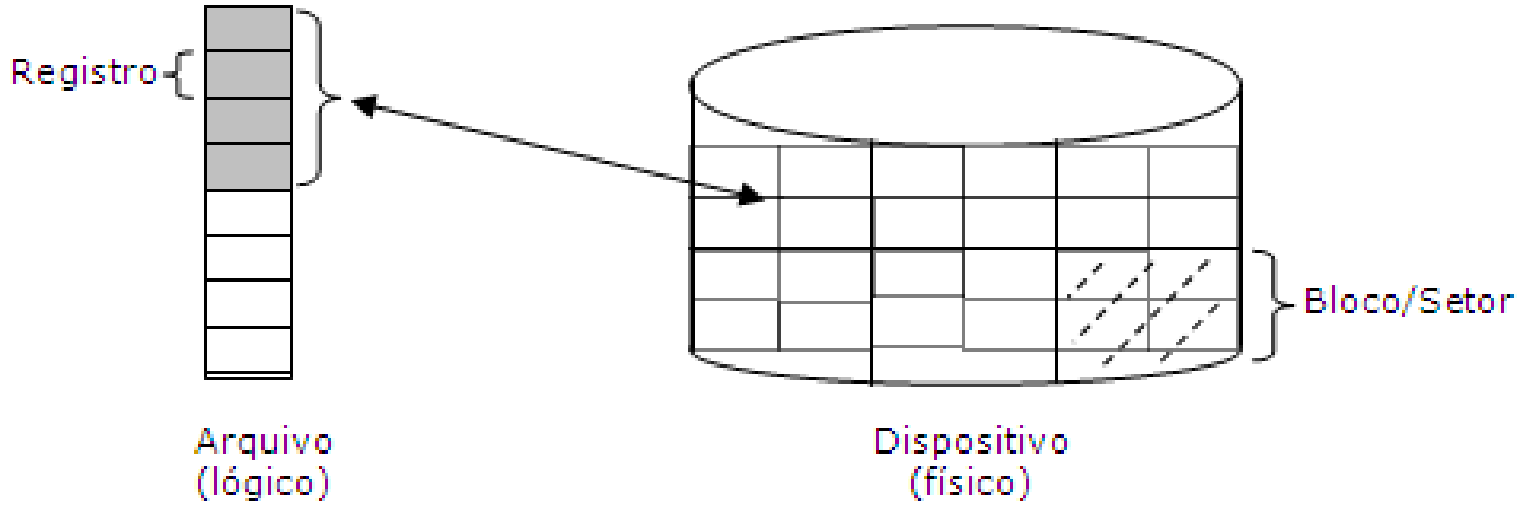
Registros Lógicos
x
Registros Físicos

- O registro **lógico** corresponde às instâncias (objetos) do mundo real, cujos atributos são representados pelos campos. Estes registros é que são manipulados pelos programas (sistemas) de aplicação.

- O registro **físico** corresponde à organização física onde as informações são armazenadas (no HD, setores).

Organização de Arquivos

- Organizar um arquivo significa relacionar (mapear) a estrutura lógica e a estrutura física de um arquivo.



Mapeamento entre estrutura física e lógica.

Organização de Arquivos

- A organização de arquivo trata do arranjo ou a forma de distribuição dos registros dentro do arquivo, objetivando **agilizar o processo de armazenamento e recuperação de dados**.

Modelo Físico

- Organização de arquivos em *heap*
  - Heap é sinônimo de aleatório, desordenado.
  - Um registro pode ser colocado em qualquer lugar no arquivo onde exista espaço para acomodá-lo
  - Não existe ordenação dos registros dentro do bloco
  - Normalmente existe um único arquivo para cada relação

Modelo Físico

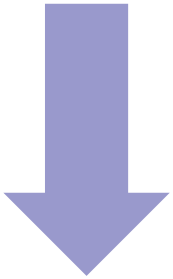
- Organização sequencial de arquivos
  - Os registros são armazenados em ordem sequencial, de acordo com o valor da “chave de busca” de cada registro
  - Em geral, a chave de busca é a chave primária da relação

Modelo Físico

- Organização de arquivos de Hashing
  - Uma função de hash é calculada sobre algum atributo de cada registro.
  - O resultado da função de hashing especifica em que bloco do arquivo o registro deve ser colocado.

Modelo Físico - PostgreSQL

```
INSERT INTO departamento VALUES(1, 'Recursos Humanos');
INSERT INTO departamento VALUES(2, 'Producao');
INSERT INTO departamento VALUES(3, 'Financeiro');
INSERT INTO departamento VALUES(4, 'Almoxarifado');
```



Data Output | Explain | Messages | Query History

	numeroDepto [PK] integer	nomeDepto character varying (30)
1	1	Recursos Humanos
2	2	Producao
3	3	Financeiro
4	4	Almoxarifado

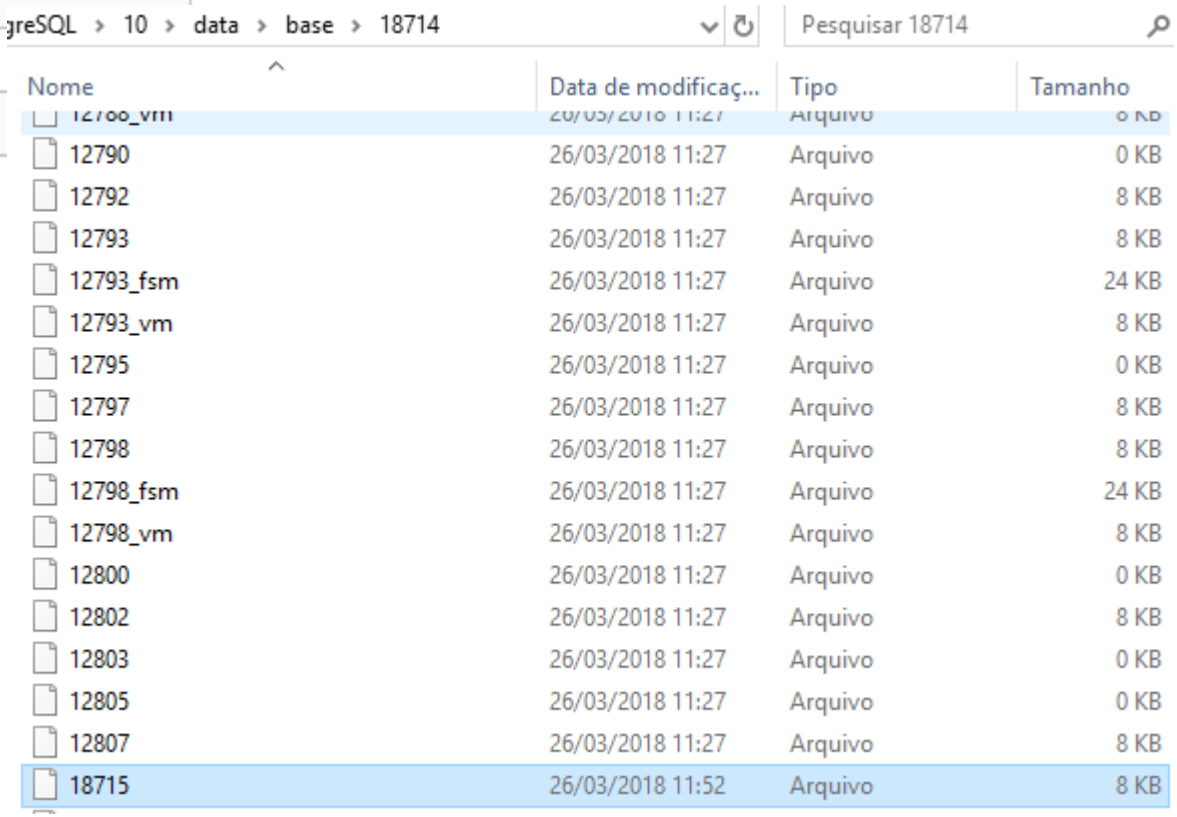
Modelo Físico – PostgreSQL

- Cada tabela é armazenada como um array de páginas (blocos) de tamanho fixo (normalmente 8kb).
- Na tabela, todas as páginas são logicamente equivalentes (mesma estrutura).
- A estrutura utilizada para armazenar uma tabela é um ***heap file***.

Modelo Físico – PostgreSQL

Data Output | Explain | Messages | Query History

	numeroDepto [PK] integer	nomeDepto character varying (30)
1	1	Recursos Humanos
2	2	Producao
3	3	Financeiro
4	4	Almoxarifado



Modelo Físico – PostgreSQL

aulaIndice on postgres@PostgreSQL 10

```
1  select ctid, * from departamento;
```

Data Output | Explain | Messages | Query History

	ctid tid	numerodepto integer	nomedepto character varying (30)
1	(0,1)	1	Recursos Humanos
2	(0,2)	2	Producao
3	(0,3)	3	Financeiro
4	(0,4)	4	Almoxarifado

Modelo Físico – PostgreSQL

```
DELETE FROM departamento;

INSERT INTO departamento VALUES(2, 'Producao');
INSERT INTO departamento VALUES(1, 'Recursos Humanos');
INSERT INTO departamento VALUES(4, 'Almoxarifado');
INSERT INTO departamento VALUES(3, 'Financeiro');
```

Modelo Físico – PostgreSQL

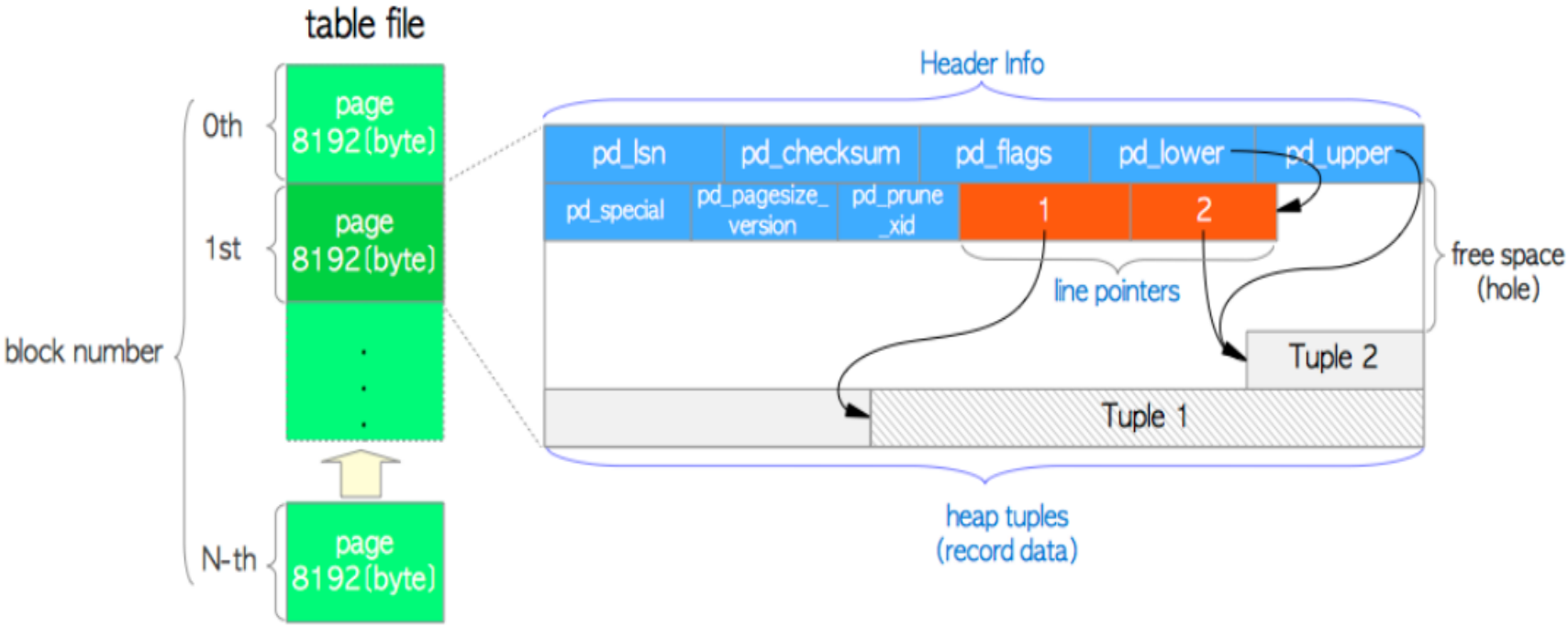
```
DELETE FROM departamento;

INSERT INTO departamento VALUES(2, 'Producao');
INSERT INTO departamento VALUES(1, 'Recursos Humanos');
INSERT INTO departamento VALUES(4, 'Almoxarifado');
INSERT INTO departamento VALUES(3, 'Financeiro');
```

```
8
9  SELECT ctid, * from departamento;
```

Data Output | Explain | Messages | Query History

	ctid tid	numerodepto integer	nomedepto character varying (30)
1	(0,5)	2	Producao
2	(0,6)	1	Recursos Humanos
3	(0,7)	4	Almoxarifado
4	(0,8)	3	Financeiro

Modelo Físico – PostgreSQL



O PostgreSQL usa tabelas Heap

Arquivos Sequenciais

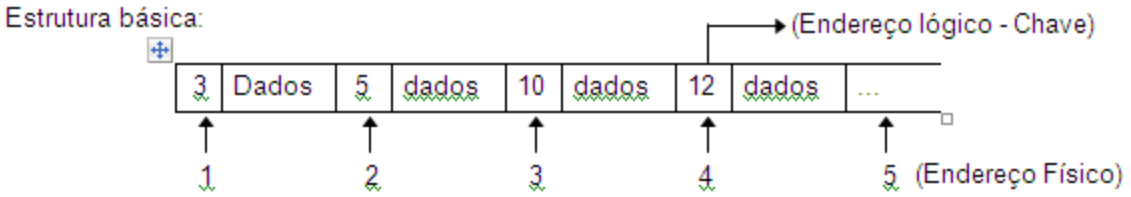
- Em um arquivo sequencial, os registros são dispostos **ordenadamente**, obedecendo à sequência determinada por uma chave primária, chamada de chave de ordenaçãc

Chave de pesquisa: *1030*

EMPREGADO

Matricula	Nome	Idade	Salário
3	Ademar	32	5000
5	Roberto	25	7500
10	Gerson	43	6000
12	Yeda	23	9000
30	Bernardo	21	4500
50	Ângela	29	5000

Chave de ordenação



Arquivos Sequenciais

- Para permitir uma recuperação rápida de registros na ordem da chave de busca, os registros são encadeados por ponteiros.

- O ponteiro em cada registro aponta para o próximo registro na ordem da chave de busca.

Arquivos Sequenciais

1	3	Ademar	32	5000
2	5	Roberto	25	7500
3	10	Gerson	43	6000
4	12	Yeda	23	9000
5	30	Bernardo	21	4500
6	50	Ângela	29	5000

Overflow

Arquivos Sequenciais

- Vantagem: Acesso sequencial de um conjunto de dados é o mais rápido.
- Útil quando é preciso processar quase todos os elementos.
- Desvantagens: Inclusão e Exclusão é extremamente custosa.

Arquivos Sequenciais
Exemplo de Inserção

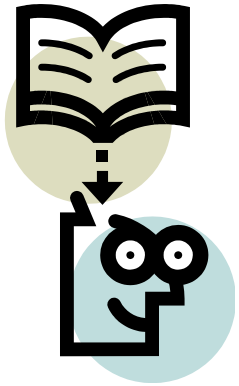
	3	Ademar	32	5000
	5	Roberto	25	7500
	10	Gerson	43	6000
	12	Yeda	23	9000
	30	Bernardo	21	4500
	50	Ângela	29	5000
	15	Vanessa	28	6000

Overflow

Inserir o registro de chave de ordenação 15.

INDEXAÇÃO

Indexação – Problema



Indexação – Problema

- Tempo para encontrar o livro aumenta significativamente com o número de livros que se tem na biblioteca em questão.

- Solução:
  - Um catálogo, organizado de alguma forma (por autor, por título, ...) que diga onde encontrar o livro na prateleira.

Índices

- Um índice permite localizar um registro sem ter que examinar mais que uma pequena fração dos registros possíveis;

- O(s) campo(s) cujos valores o índice se baseia formam a **chave de pesquisa**;

Índices

- Índices são, portanto, **estruturas de dados auxiliares** cujo único propósito é tornar mais **rápido** o acesso a registros baseados em certos campos, chamados campos de indexação.

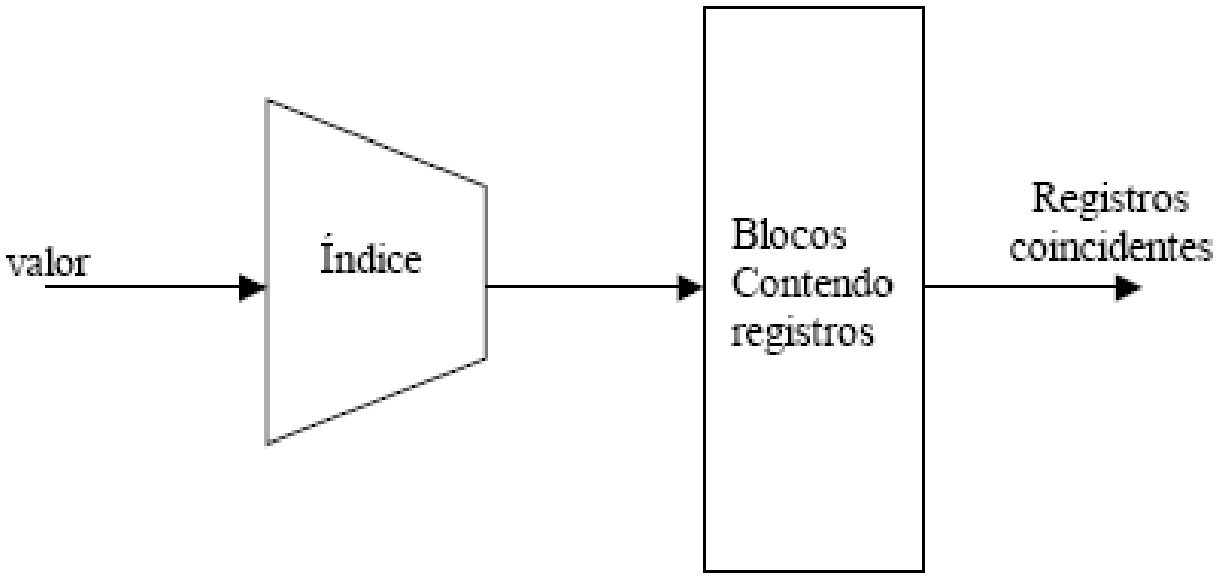
Índices

- Cada índice é formado pelo valor de um atributo chave do registro e pela sua localização física no interior do arquivo.

- A estrutura com a tabela de índices é também armazenada e mantida em disco.

Índices

- Cada estrutura de índice está associada a uma chave de busca particular.



Tipos básicos de Índices

- Ordenados
  - Baseiam-se na ordenação dos valores.
- Hash
  - Baseiam-se na distribuição uniforme dos valores determinados por uma função (função de hash).

Classificação de Índices Ordenados

- Considerando a quantidade de entradas
  - Denso
  - Esparso
- Considerando a organização do arquivo
  - Primário
  - Clustering
  - Secundário
- Considerando os níveis de indirecionamento
  - Mononível
  - Multinível

UNIVERSIDADE FEDERAL DE ITAJUBÁ

ÍNDICES ORDENADOS

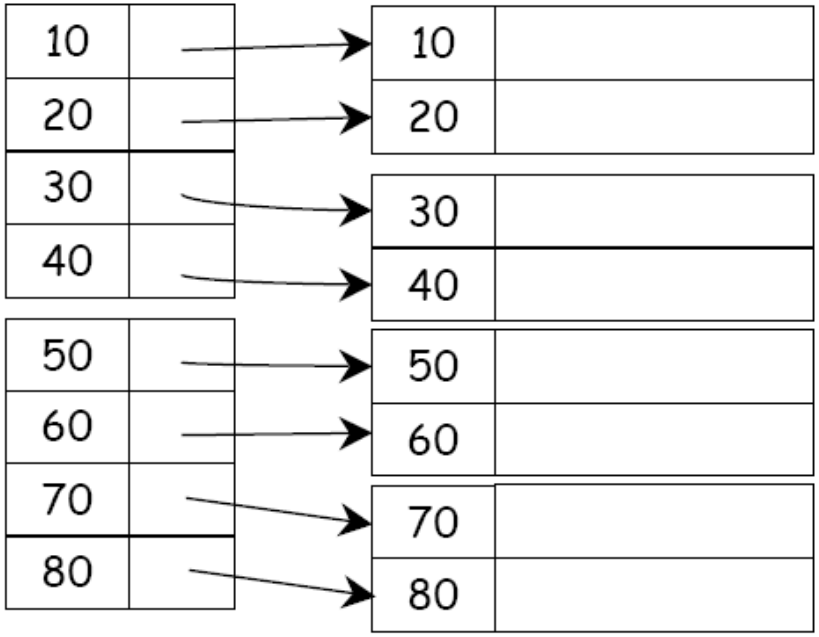
DENSO X ESPARSO

Índices sobre arquivos sequenciais

- Os ponteiros em um arquivo de índices podem se referir a **cada registro** do arquivo ou a um **bloco de disco**, originando dois tipos de índices:
  - Índice denso
  - Índice esparso

Índice Denso

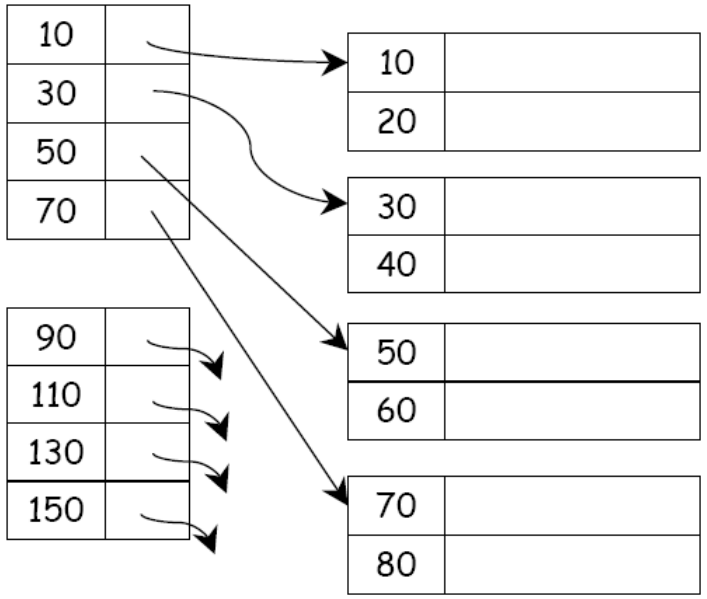
- Há uma entrada no índice para cada valor de chave que ocorre em um registro de dados.



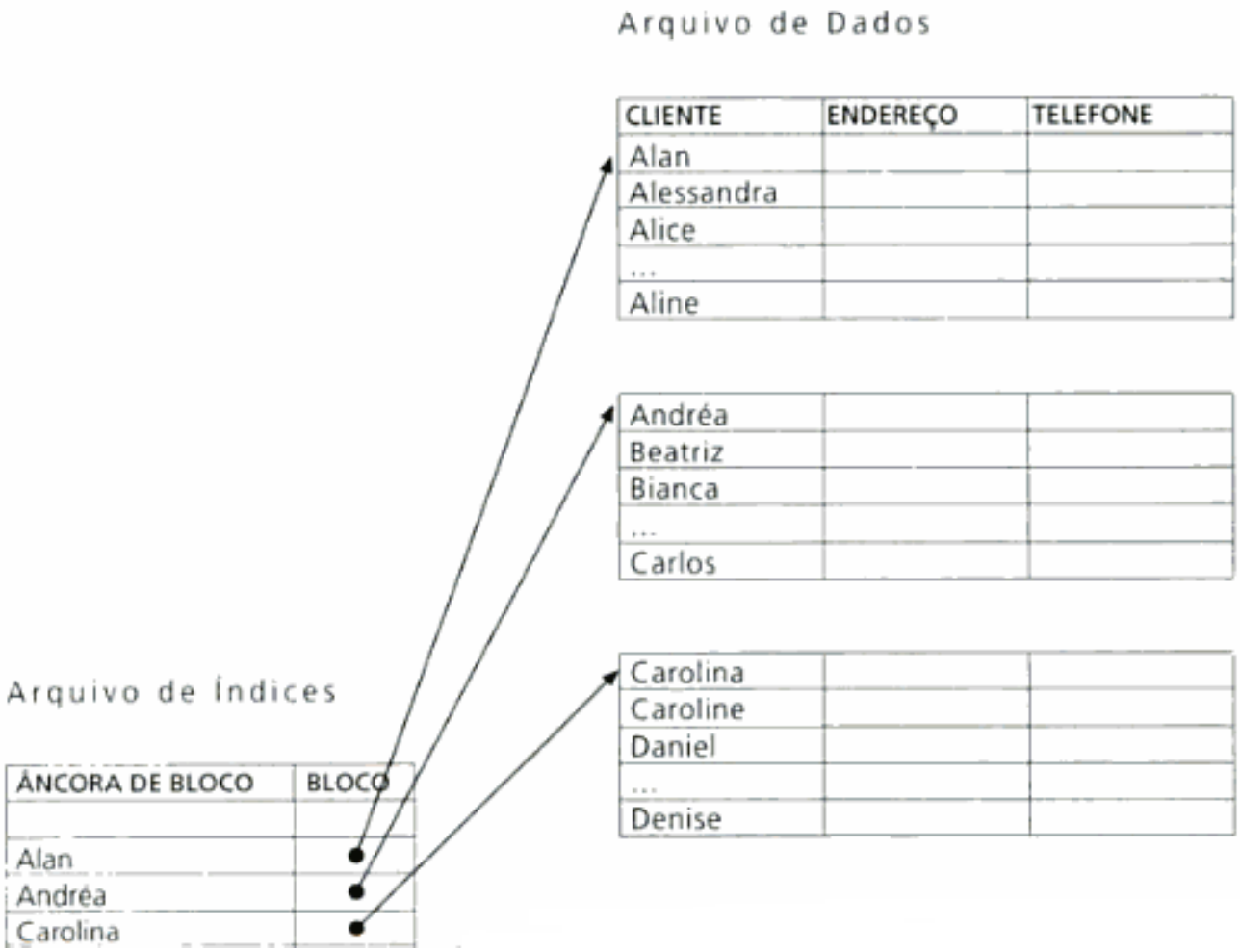
Esse registro de índice contém o valor da chave de busca e um ponteiro para o registro.

Índice Esparso

- Há um entrada no índice apenas para alguns valores de chave (um para cada bloco).
- Para localizar um registro com chave K, procura-se a entrada E do índice com o maior valor de chave menor ou igual a K e pesquisa-se o arquivo a partir do registro apontado por E.



Índice Esparso



Índice Esparso
x
Índice Denso

- Um índice esparso utiliza menos espaço para armazenamento ao custo de um tempo um pouco maior para encontrar um registro dada sua chave.

- No índice esparso, inserções e remoções são menos custosas quando comparadas ao índice denso.

UNIVERSIDADE FEDERAL DE ITAJUBÁ

ÍNDICES ORDENADOS

PRIMÁRIO X CLUSTERING X SECUNDÁRIO

Índice Primário

- No arquivo sequencial, um campo chave é usado para ordenar fisicamente os registro de arquivos em disco.

- Se a chave de ordenação for um campo exclusivo (sem duplicatas), então o índice é chamado de **Índice Primário**.

Índice Primário

- A chave de busca de um índice primário é **usualmente** a chave primária.

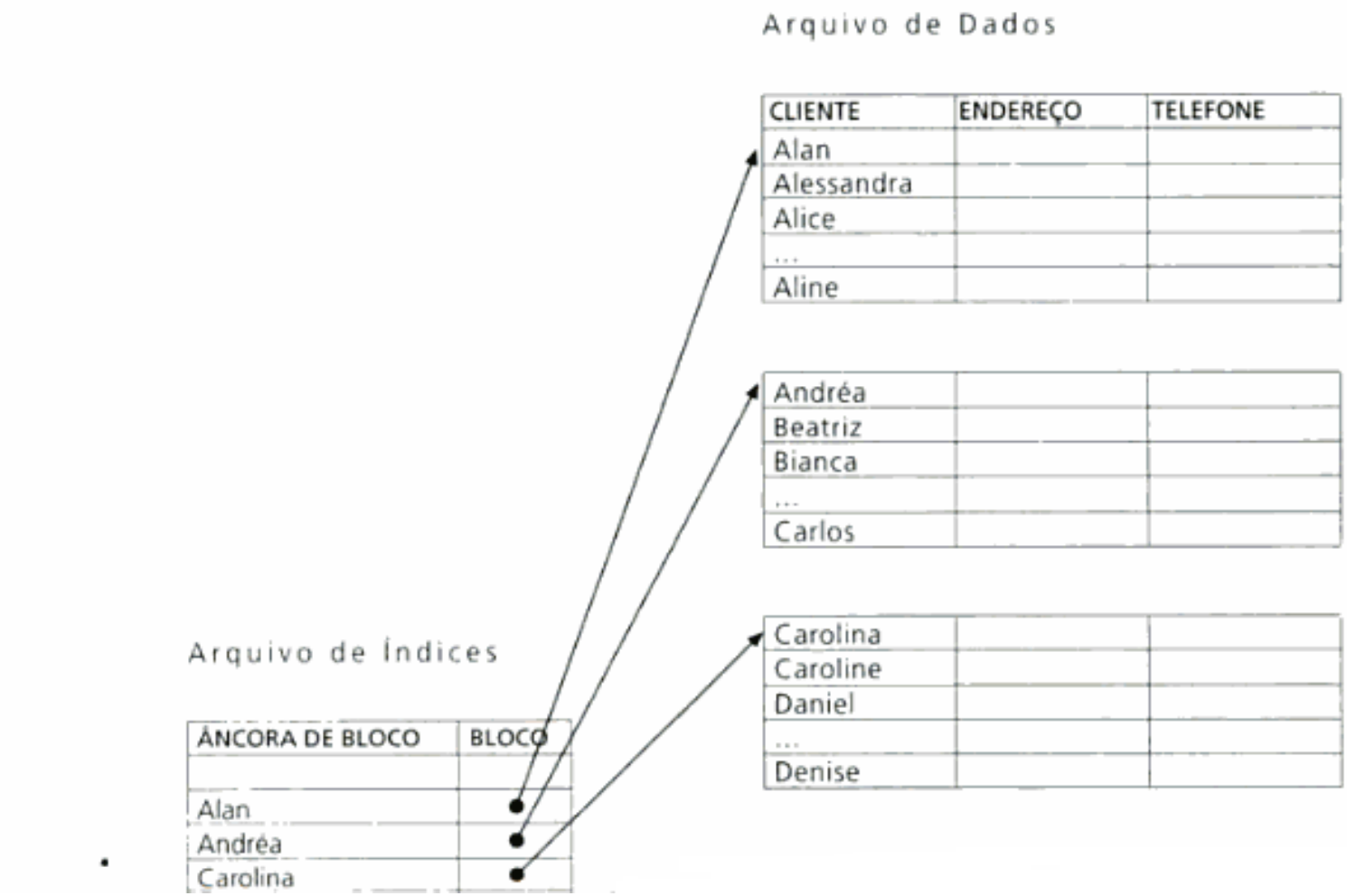
Índice Primário

- Vantagens
  - Economia de acesso aos blocos
  - Busca binária
- Desvantagem
  - Inclusão e remoção de registros

Índices primários em geral são esparsos

(o número de entradas é
igual ao número de blocos do arquivo de dados)

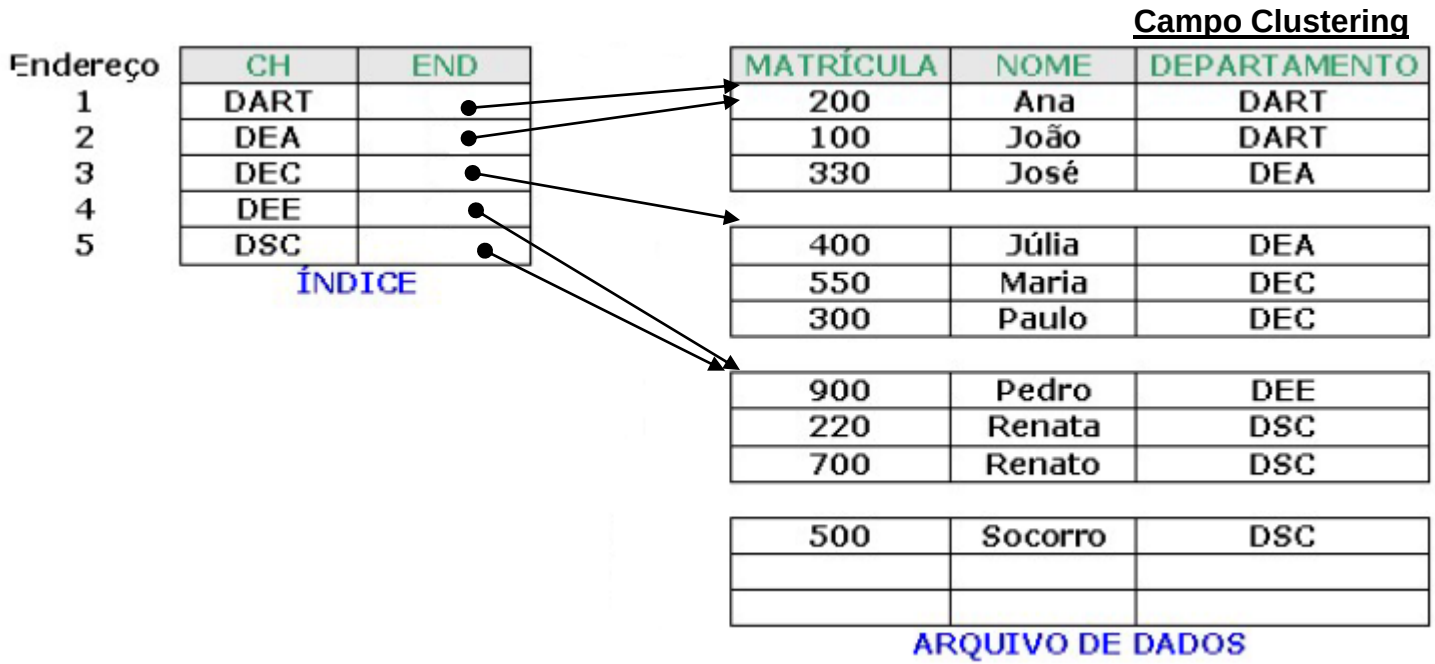
Índice Primário Esparso



Índice Clustering

- Se a chave de ordenação for um campo não chave (com duplicatas), então o índice é chamado de **Índice Clustering**.
- Existe uma entrada no índice clustering para cada valor distinto do campo clustering.
- O índice contém o valor e um ponteiro para o primeiro bloco no arquivo de dados que possua um registro com aquele valor para seu campo clustering.

Índice de Clustering



No **Índice de Cluster** os registros de um arquivo são fisicamente ordenados por um campo de ordenação que permite duplicatas.

Índices de clustering são esparsos

(o número de entradas é igual ao número de valores distintos do atributo de clustering)

Índice de Clustering

- Vantagens
  - Economia de acesso aos blocos
- Desvantagem
  - Inclusão e remoção de registros

Índices sobre arquivos sequenciais

- Um arquivo pode possuir, no máximo, um índice primário ou um índice de clustering, **porém não ambos**.

- Porque um arquivo pode possuir, no máximo, um campo de ordenação física.

Índice Secundário

- Frequentemente desejamos ter vários índices em uma relação, a fim de facilitar consultas variadas.
- Um terceiro tipo de índice, chamado de índice secundário, pode ser especificado sobre qualquer campo não-ordenado de um arquivo.
- Um arquivo pode possuir diversos índices secundários, além de seu método de acesso principal.

Índice Secundário

- Um índice secundário é um índice cuja chave não é aquela da ordem sequencial do arquivo.

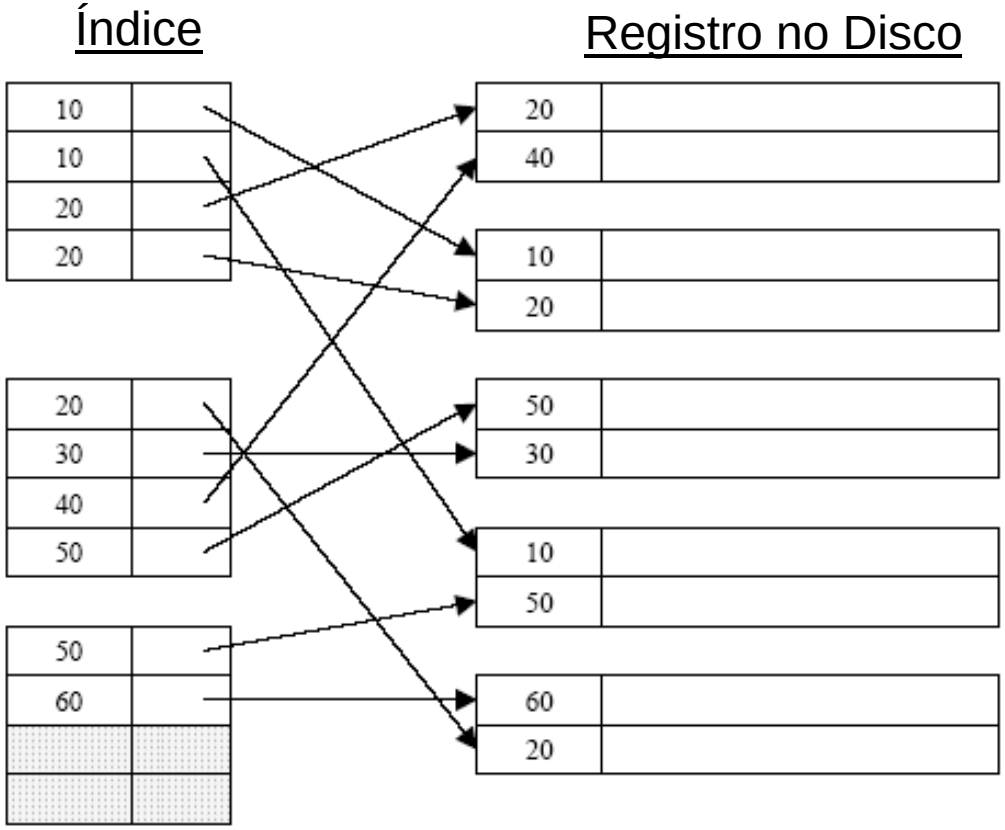
- Portanto, a diferença entre índice secundário e índice primário está no fato de que o índice secundário não determina a colocação de registros no arquivo de dados.

Índices secundários são sempre densos!!!

Projeto de índices secundários

- Um índice secundário é um índice denso, normalmente com duplicatas.
- O índice consiste em pares chave-ponteiro.
- Os pares no arquivo de índices são classificados pelo valor da chave.

Índices Secundários - Exemplo



- Os dados não estão classificados pela chave de pesquisa.
- As chaves no arquivo de índices estão classificadas.

Modelo Físico – PostgreSQL

```
DELETE FROM departamento;

INSERT INTO departamento VALUES(2, 'Producao');
INSERT INTO departamento VALUES(1, 'Recursos Humanos');
INSERT INTO departamento VALUES(4, 'Almoxarifado');
INSERT INTO departamento VALUES(3, 'Financeiro');
```

```
8
9  SELECT ctid, * from departamento;
```

Data Output | Explain | Messages | Query History

	ctid tid	numerodepto integer	nomedepto character varying (30)
1	(0,5)	2	Producao
2	(0,6)	1	Recursos Humanos
3	(0,7)	4	Almoxarifado
4	(0,8)	3	Financeiro

Modelo Físico – PostgreSQL

```
CREATE TABLE cep.log_logradouro
(
    log_nu_sequencial integer NOT NULL DEFAULT 0,
    ufe_sg character varying(2) COLLATE pg_catalog."default" NOT NULL,
    loc_nu_sequencial integer NOT NULL DEFAULT 0,
    log_no character varying(70) COLLATE pg_catalog."default" NOT NULL,
    log_nome character varying(125) COLLATE pg_catalog."default" NOT NULL,
    bai_nu_sequencial_ini integer NOT NULL DEFAULT 0,
    bai_nu_sequencial_fim integer DEFAULT 0,
    cep character varying(16) COLLATE pg_catalog."default" NOT NULL,
    log_complemento character varying(100) COLLATE pg_catalog."default",
    log_tipo_logradouro character varying(72) COLLATE pg_catalog."default",
    log_status_tipo_log character varying(1) COLLATE pg_catalog."default" NOT NULL,
    log_no_sem_acento character varying(70) COLLATE pg_catalog."default" NOT NULL,
    ind_uop character varying(1) COLLATE pg_catalog."default",
    ind_gru character varying(1) COLLATE pg_catalog."default",
    temp character(8) COLLATE pg_catalog."default",
    CONSTRAINT log_logradouro_pkey PRIMARY KEY (log_nu_sequencial)
)
```

```
SELECT * FROM pg_indexes WHERE tablename = 'log_logradouro';
```

schemaname name	tablename name	indexname name
cep	log_logradouro	log_logradouro_pkey

indexdef text
CREATE UNIQUE INDEX log_logradouro_pkey ON cep.log_logradouro USING btree (log_nu_sequencial)

Modelo Físico – PostgreSQL

- Se a tabela *heap* não ordena os dados fisicamente pela chave primária, como se dá o índice de chave primária?
  - Arquivos Heap são listas de registros **não ordenados** de tamanho variado.
  - Uma vez que não há ordenação, para localizar uma informação em um Heap é necessário realizar sempre um "*table scan*".
  - Heap é uma tabela **sem índice clusterizado** (esparso)
  - O índice é do tipo UNIQUE
    - Todo índice UNIQUE é secundário

Modelo Físico – MySQL INNODB

```
mysql> show indexes from departamento \G;
*************************** 1. row ***************
        Table: departamento
   Non_unique: 0
     Key_name: PRIMARY
 Seq_in_index: 1
  Column_name: numeroDepto
    Collation: A
  Cardinality: 0
     Sub_part: NULL
       Packed: NULL
         Null:
   Index_type: BTREE
      Comment:
Index_comment:
1 row in set (0.00 sec)
```

Modelo Físico – MySQL INNODB

```
INSERT INTO departamento VALUES(2, 'Producao');
INSERT INTO departamento VALUES(1, 'Recursos Humanos');
INSERT INTO departamento VALUES(4, 'Almoxarifado');
INSERT INTO departamento VALUES(3, 'Financeiro');
```

```
mysql> Select * from departamento;
+-------------+------------------+
| numeroDepto | nomeDepto        |
+-------------+------------------+
|           1 | Recursos Humanos |
|           2 | Producao         |
|           3 | Financeiro       |
|           4 | Almoxarifado     |
+-------------+------------------+
4 rows in set (0.00 sec)
```

Exercícios

- Considere uma relação com 1.000.000 registros (ou tuplas) que cabem dez a dez em um bloco de 4.096 bytes.

  a) Qual é o espaço necessário para os dados?

  b) Se o campo da chave tem 32 bytes e um ponteiro 8 bytes.
     a) Quantos pares chave-ponteiro cabem em um bloco?
     b) Qual o espaço ocupado por um índice denso para esta relação?
     c) E se um índice esparso for utilizado, qual o espaço gasto pelo índice?

Resumo Importante!

- Utilizando arquivos sequenciais:
  - Existe índice primário (esparso) para a chave primária da tabela
  - Existe índice secundário (denso) para os demais campos
- Utilizando arquivos heaps:
  - Existe apenas índice secundário, mesmo para a chave primária
- Utilizando arquivos hash:
  - Existe apenas índice secundário, mesmo para a chave primária
- Em todos os casos, o comando SQL de CREATE ÍNDEX só cria índice SECUNDÁRIO.