Arquitetura de SGBD Relacionais
— Organização da Memória em Disco para SGBD Relacionais —

Caetano Traina Jr.

Grupo de Bases de Dados e Imagens
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - São Carlos
caetano@icmc.usp.br

5 de maio de 2010
São Carlos, SP - Brasil

Outline

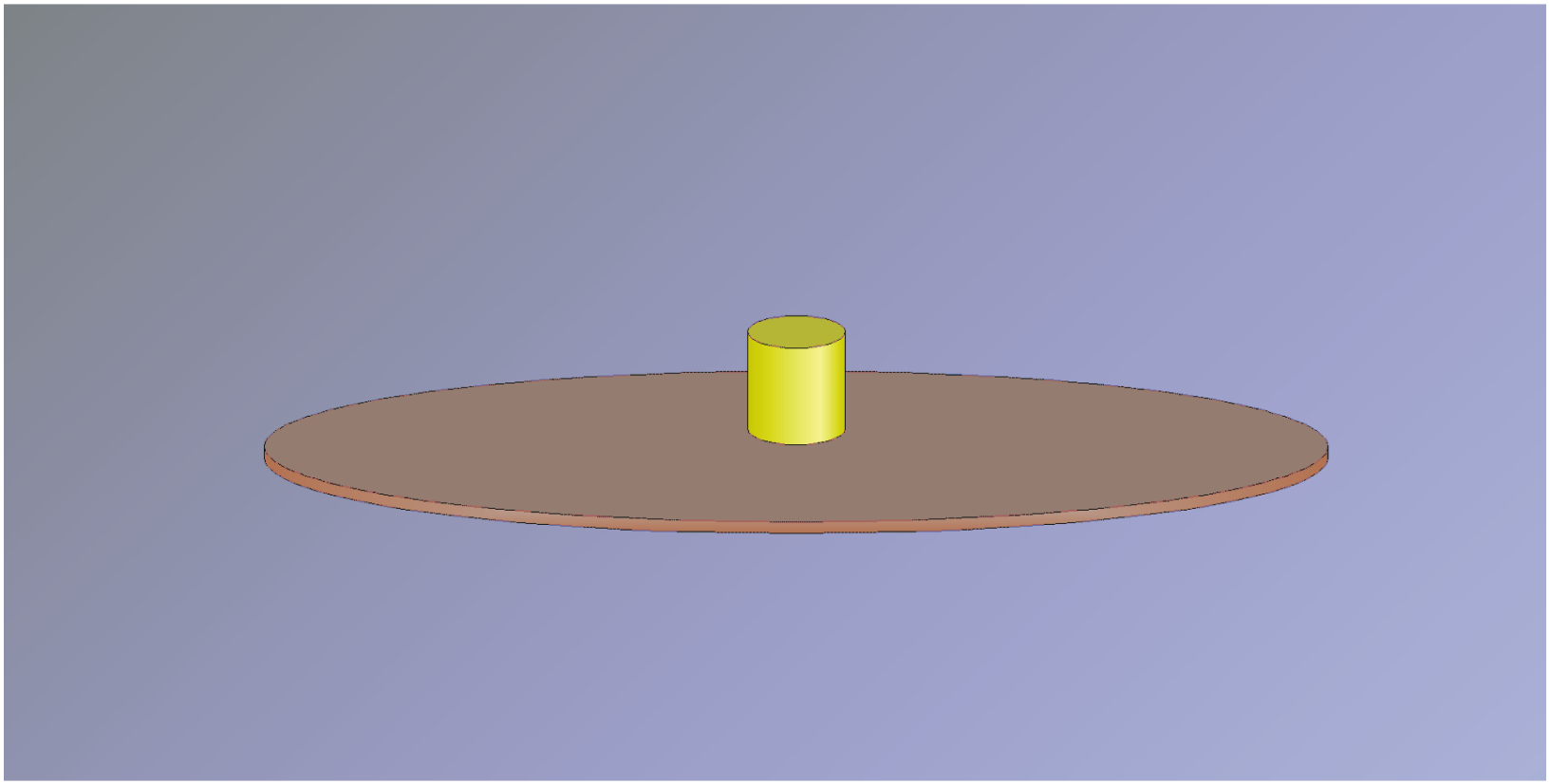
Estrutura Física dos discos

- Dois pontos fundamentais de um SGBD Relacional estão intrinsecamente associados às propriedades do meio físico onde os dados são armazenados:
  - Confiabilidade;
  - rapidez de acesso.
- Nesta apresentação, serão estudadas as características físicas do meio físico de armazenagem dos dados que possam afetar o desempenho das consultas.

e o meio físico mais usado hoje são os DISCOS!

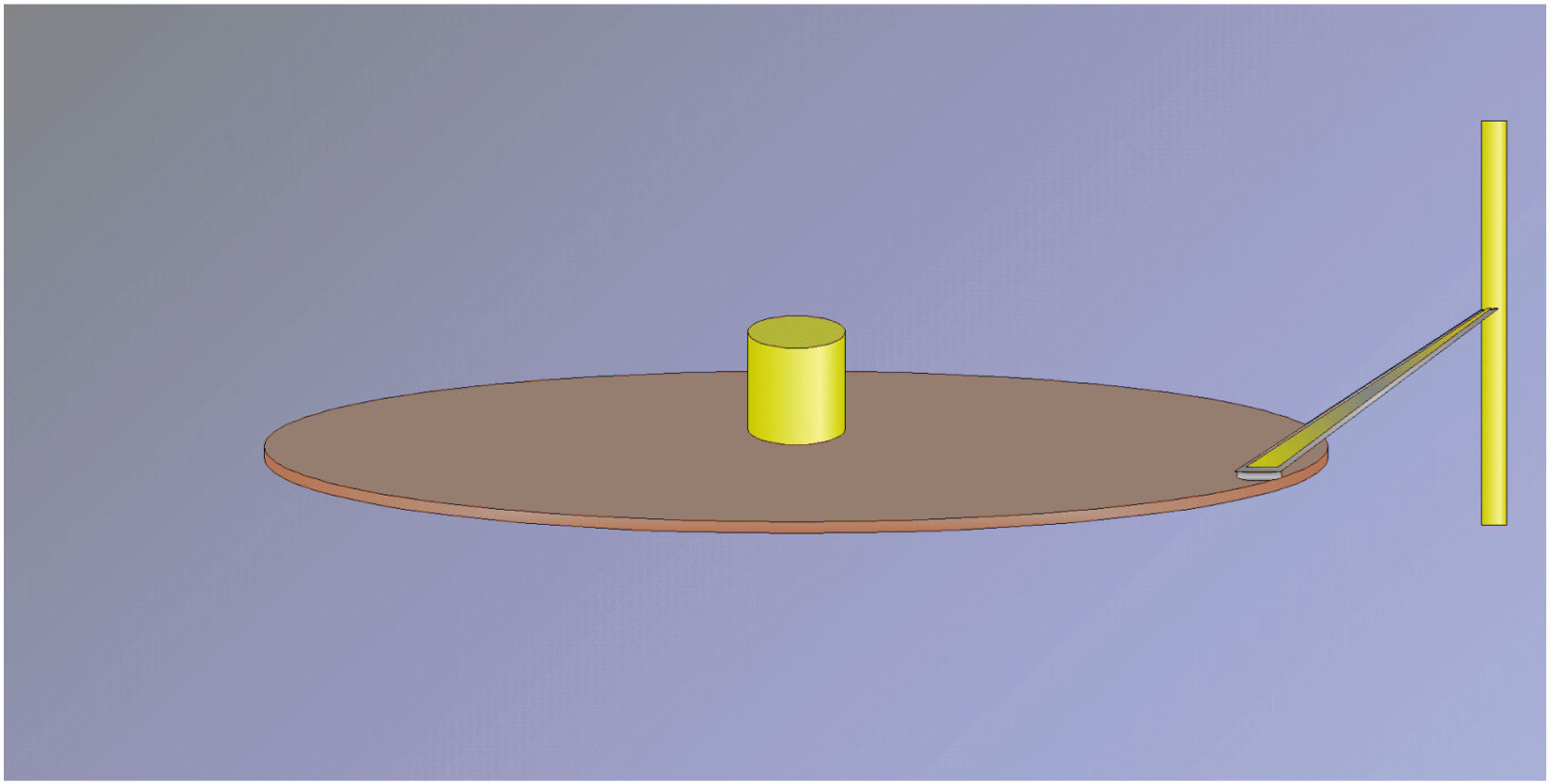
Estrutura dos discos

Que propriedades dos discos são importantes para agilizar o acesso?



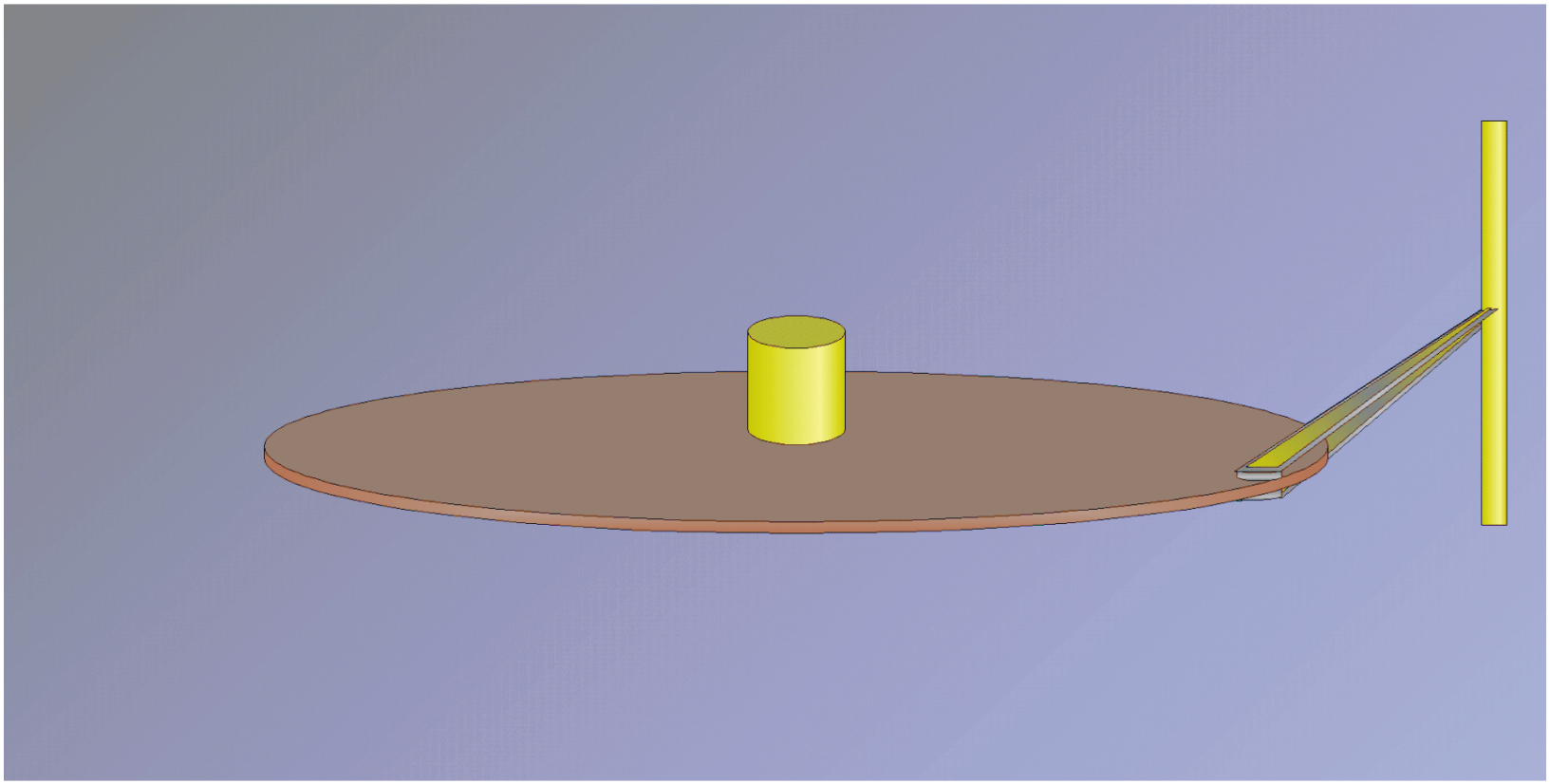
Estrutura dos discos

O superfície do disco é um meio magnético lido por uma cabeça,



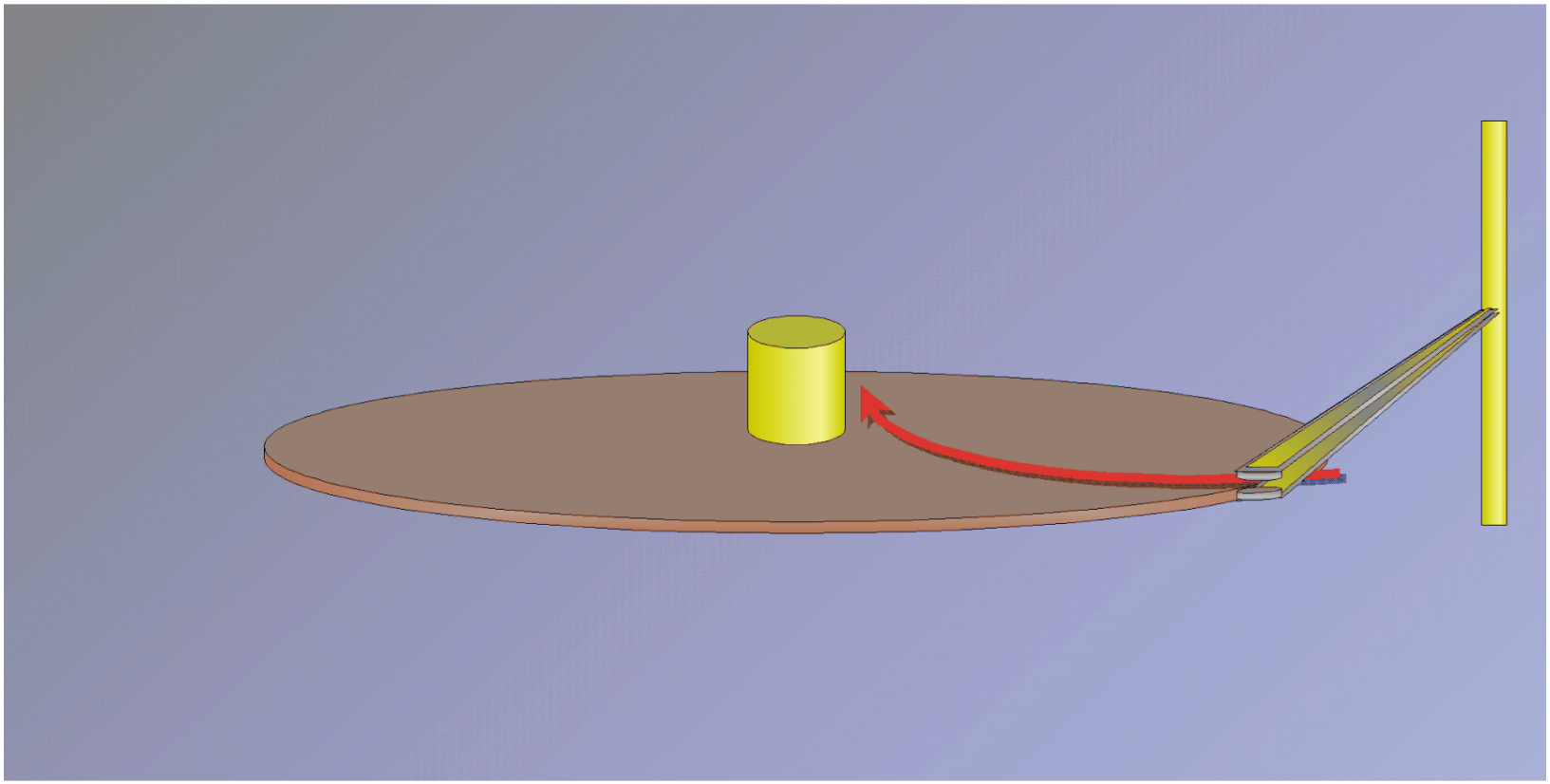
Estrutura dos discos

hoje em dia, cada superfície tem só uma cabeça,



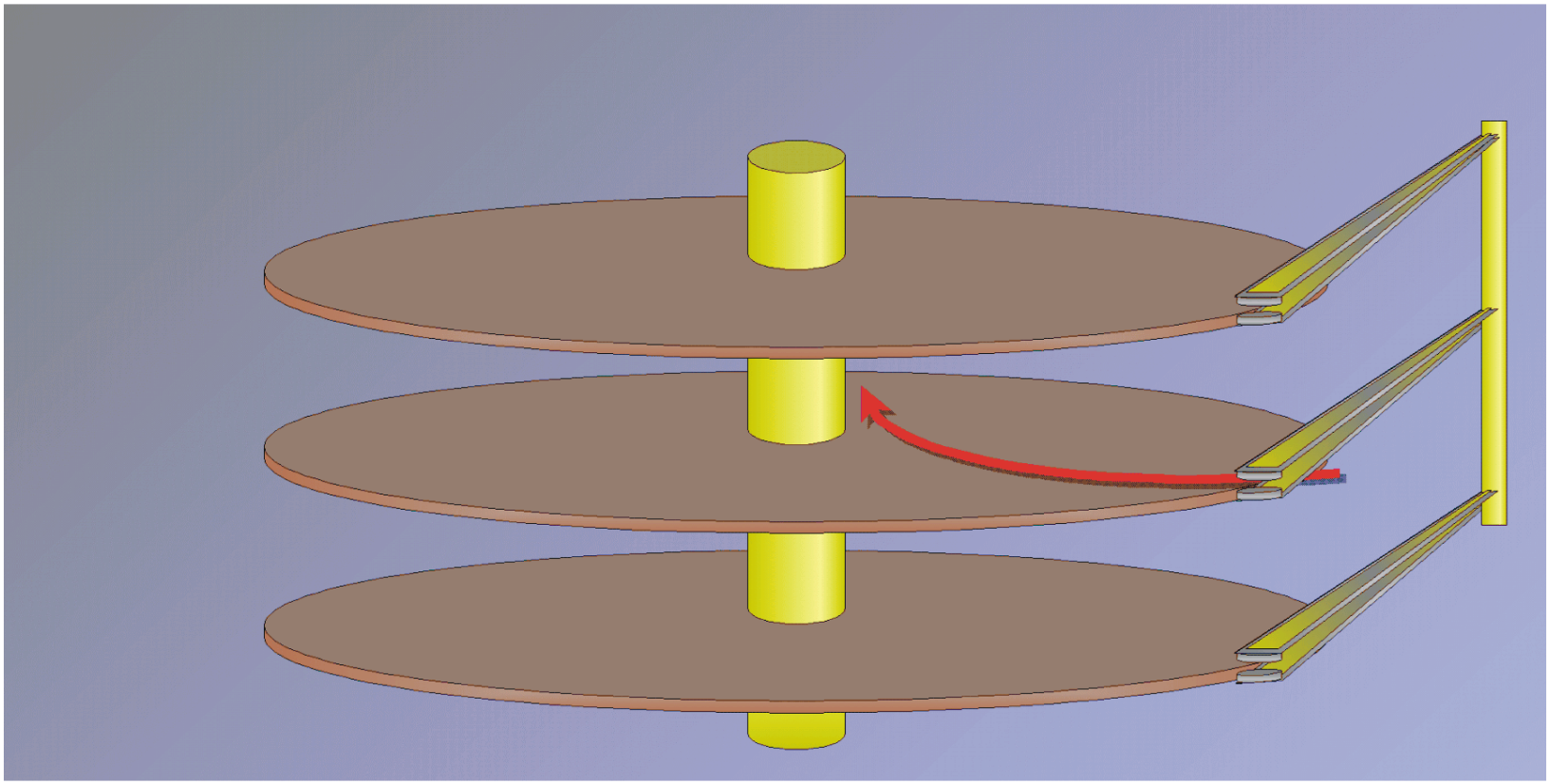
Estrutura dos discos

e todas as cabeças pivotam juntas no mesmo eixo.



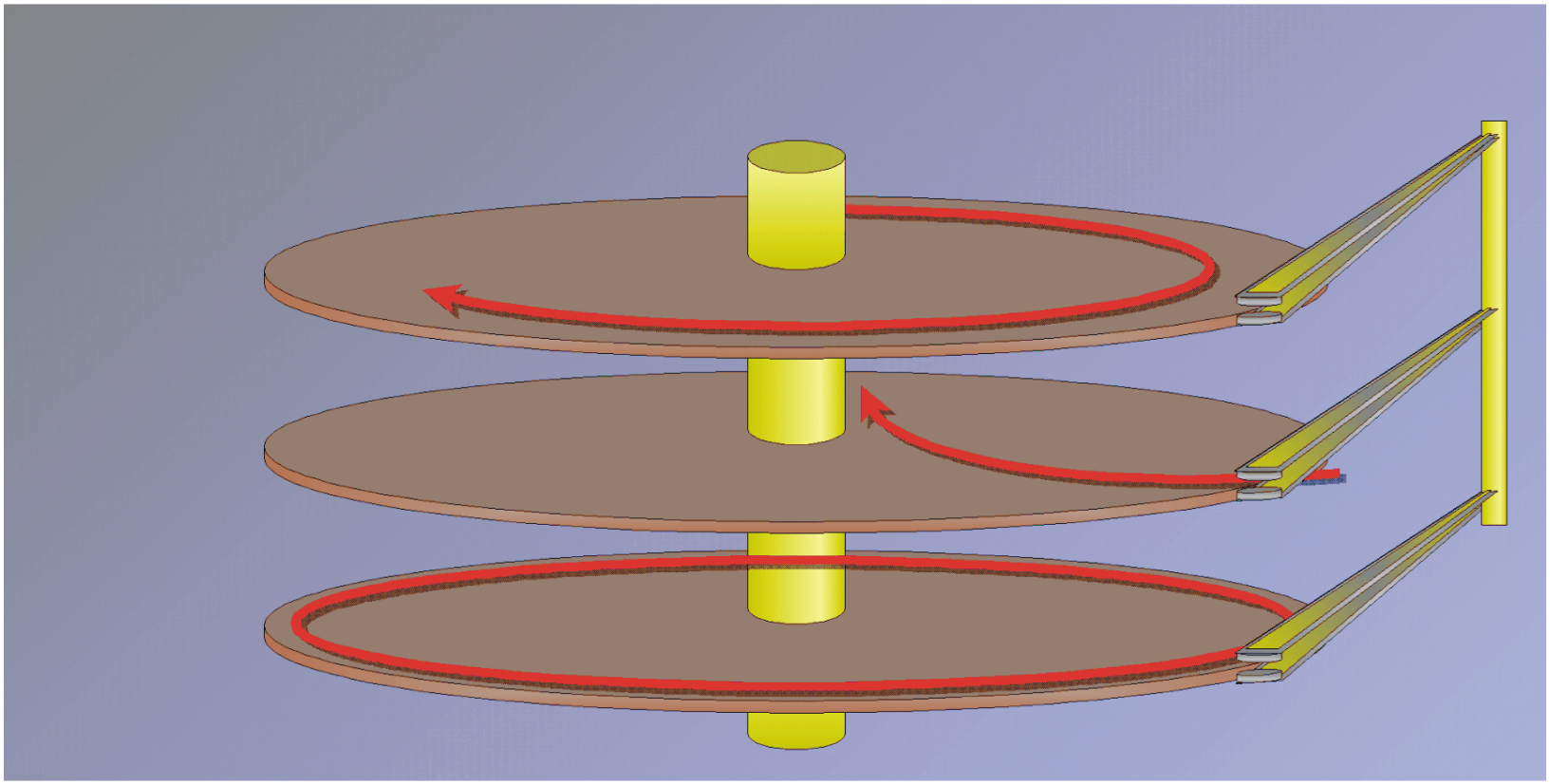
Estrutura dos discos

Uma unidade de disco pode ser composta por diversos “discos”,



Estrutura dos discos

todos os discos e todas as cabeças se movimentam juntos.



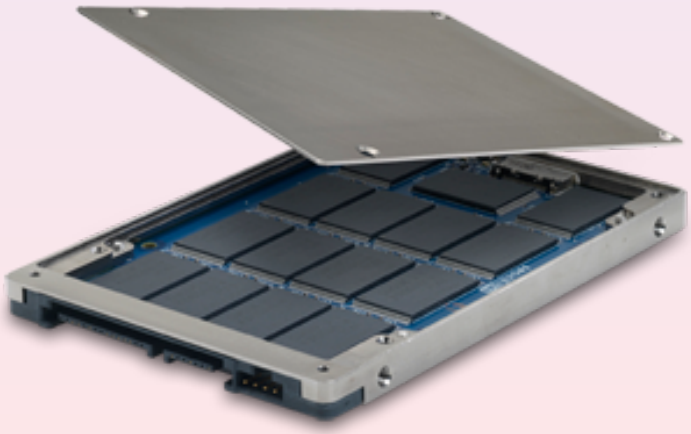
Estrutura dos discos

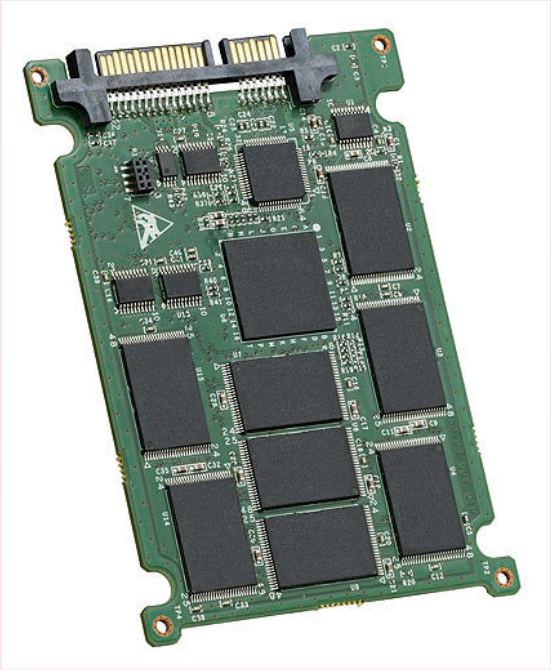
(Seagate)

(Seagate)

Estrutura dos discos

Atualmente, está se iniciando o uso da tecnologia de "Discos de Estado Sólido" (Solid Dtate Drive – SSD), que usa memória *flash* não volátil (a mesma tecnologia utilizada em *pen-drives*) para substituir os discos magnéticos. No entanto, essa tecnologia ainda está em início de adoção, e para a armazenagem de grandes volumes com alta performace e confiabilidade, a tecnologia de ponta ainda é a de discos magnéticos.





Estrutura dos discos

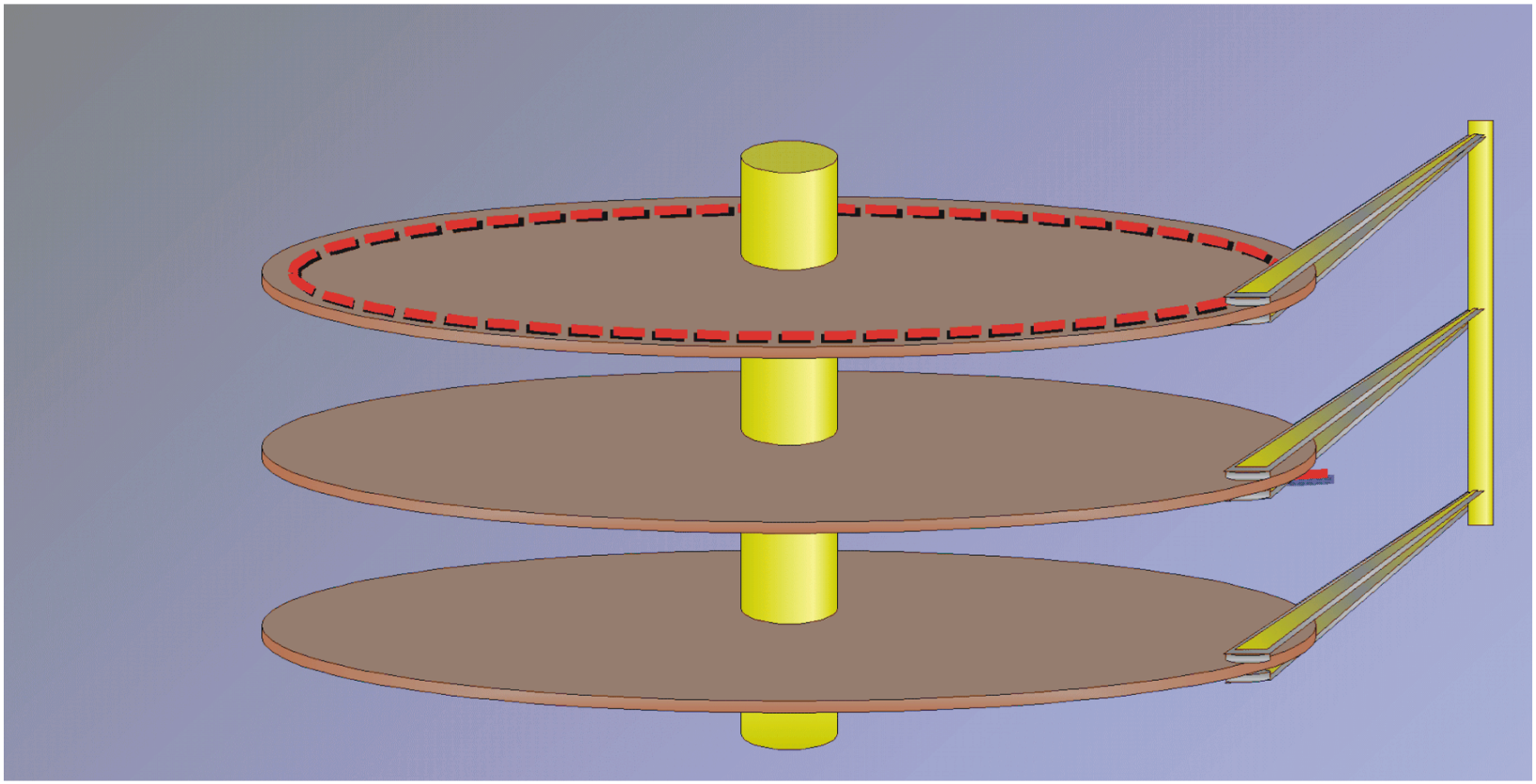
Como existe movimento, os tempos envolvidos nos movimentos afetam o tempo de execução das consultas.
São parâmetros importantes para isso:

- A velocidade angular dos discos (RPM)
- O tempo médio para posicionar as cabeças (mS)
- A taxa de transferência (MBytes/s)

Além disso, é importante conhecer a estrutura de organização dos dados no disco.

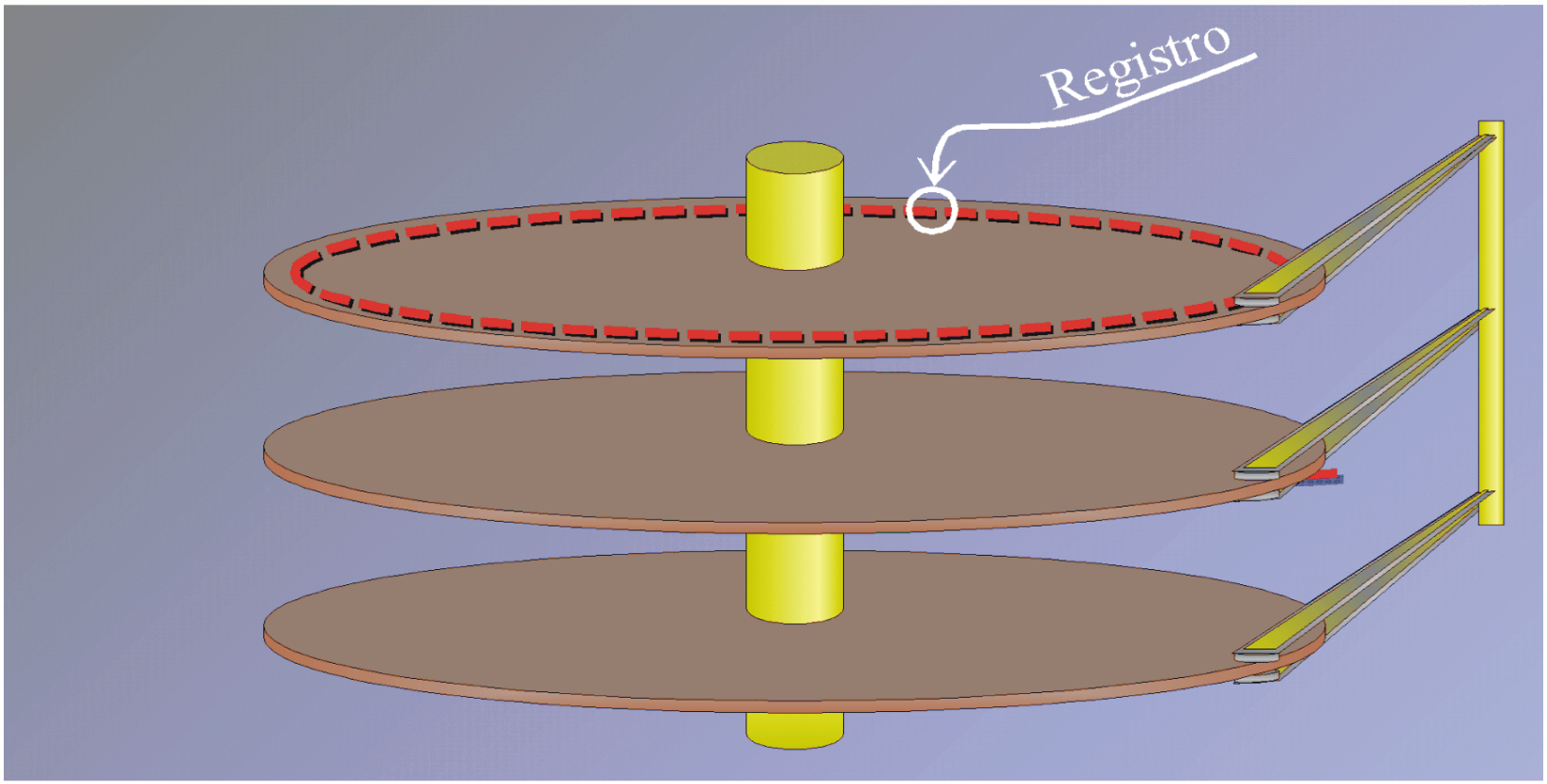
Estrutura dos discos

Cada cabeça acessa os dados em circunferências concêntricas, e todas se movimentam juntas para mudar a respectiva circunferência.



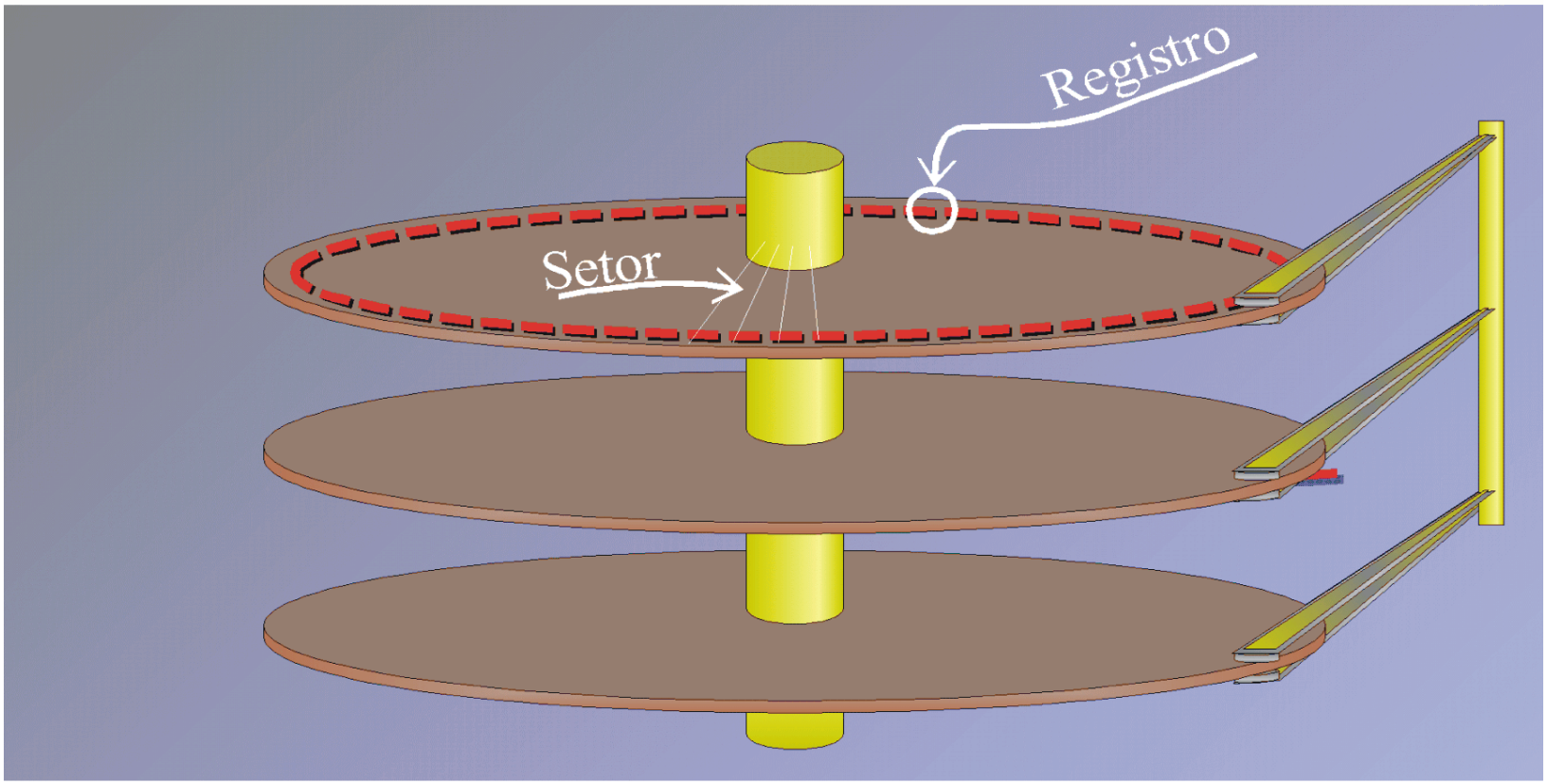
Estrutura dos discos

Os dados são separados em blocos de tamanho fixo, chamados registros. Atualmente, todos os padrões de disco usam registros de 512 Bytes.



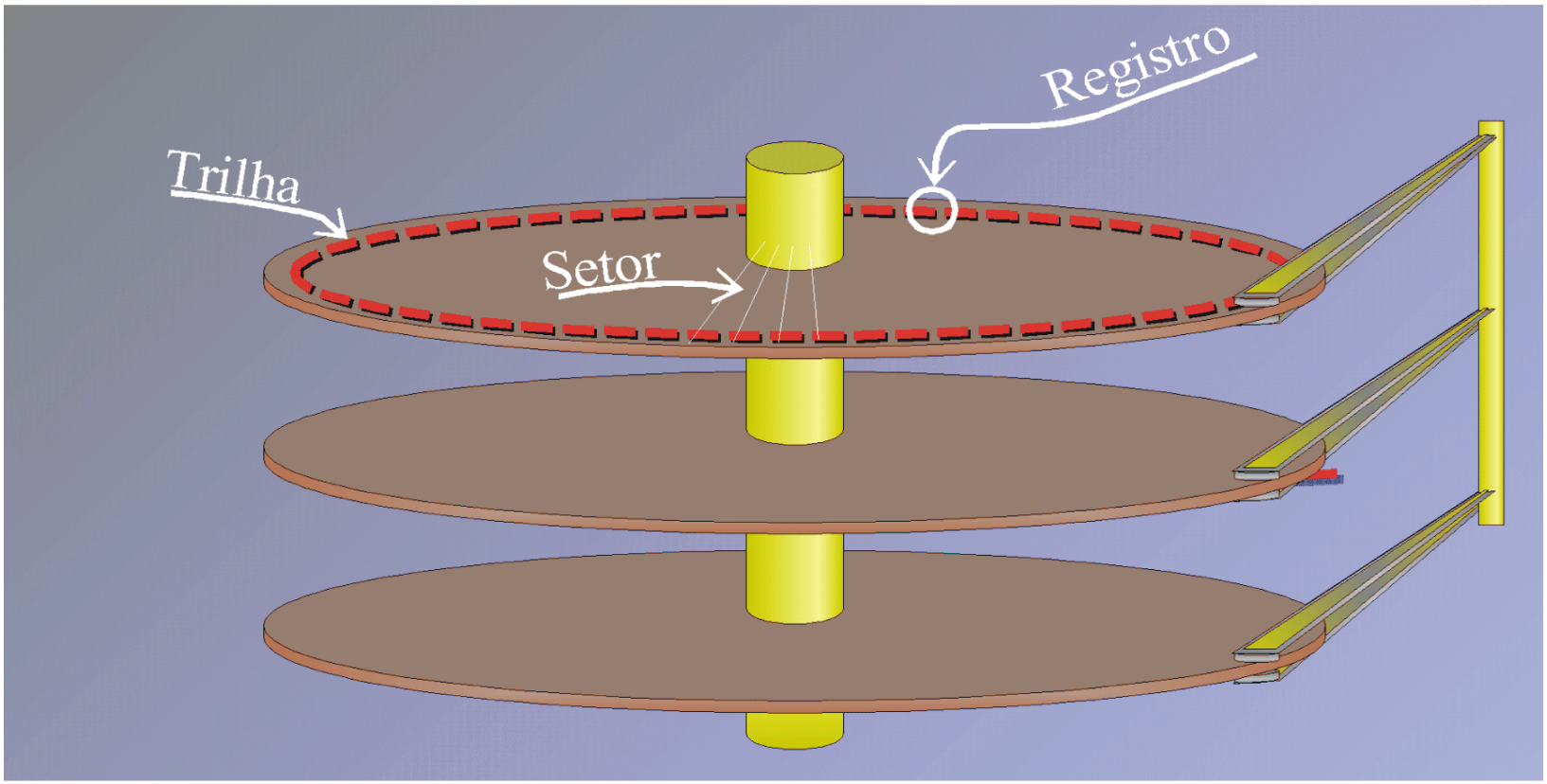
Estrutura dos discos

A coleção de todos os registros numa mesma abertura angular é chamada Setor. Cada modelo de disco ten um dado número de setores.



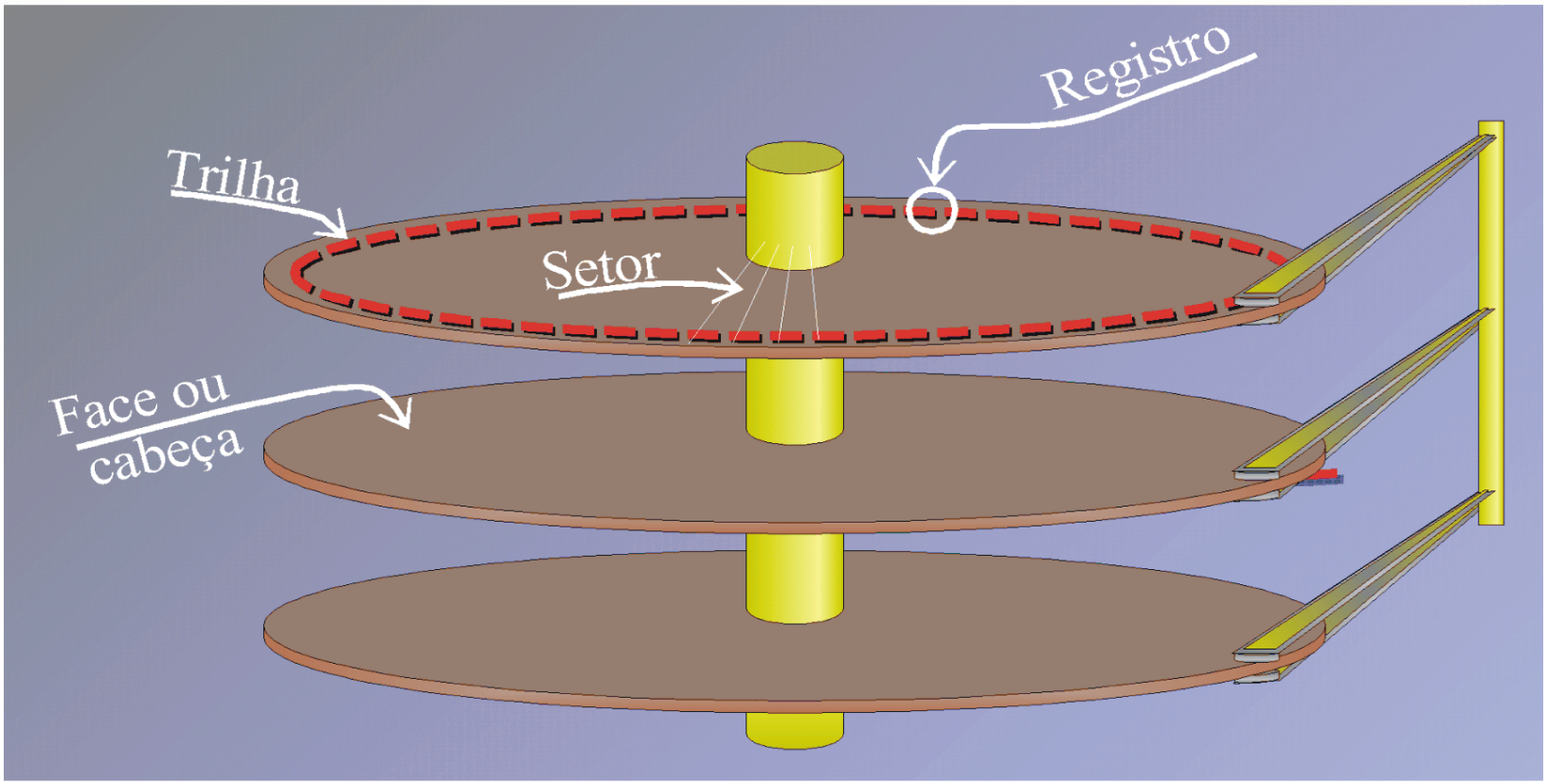
Estrutura dos discos

Cada cabeça acessa os dados em circunferências concêntricas, cada circunferência em uma superfície é uma trilha,



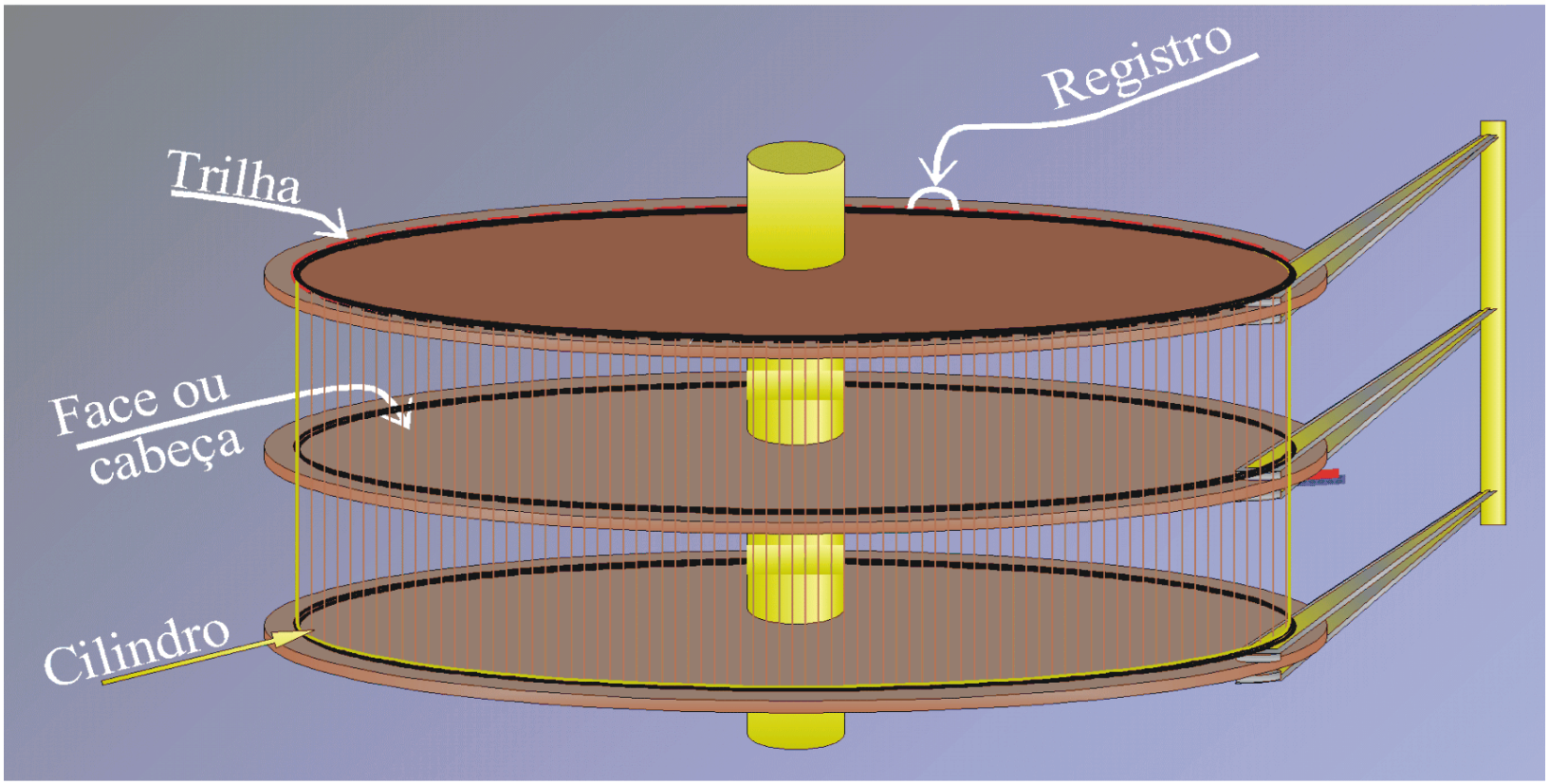
Estrutura dos discos

Cada cabeça corresponde a uma face. Praticamente todas as unidades de disco usam as duas faces de todos os discos.



Estrutura dos discos

O conjunto de todas as trilhas disponíveis para acesso sem precisar mover as cabeças é chamado cilindro.



Acesso aos dados em disco

- O endereço de um byte em um disco é dado por:
  - **Cilindro** ☞ precisa mover as cabeças até aquele cilindro;
  - **Setor** ☞ precisa esperar o disco girar até a cabeça estar posicionada no inicio do setor;
  - **Trilha** ☞ precisa ligar a cabeça;
  - **Offset** ☞ posição do byte dentro do registro.
- A unidade mínima de dados que pode ser lida/escrita numa operação do disco é um registro (Hoje em dia é sempre 512 bytes — IDE, SCSI, SAS, etc.)
- Assim, sempre que se quiser escrever um byte no disco, deve-se
  - ler o registro correspondente em um *buffer* na memória;
  - mudar o byte no *buffer*;
  - escrever o *buffer* no registro correspondente.

Acesso aos dados em disco

- Portanto, uma operação no disco gasta em média os seguintes tempos:
  - Posicionar as cabeças: gasta o tempo médio para posicionar as cabeças;
  - esperar o tempo médio para o disco girar meia volta;
  - ligar a cabeça é instantâneo;
  - Esperar a transferência do registro;
  - Acessar os dados no *buffer* (tempo de instruções de máquina – μs)

Acesso aos dados em disco

- Vamos considerar um disco típico de alta performance:

`Seagate Savvio 10K.4 SAS`

- Interface: SAS 6-Gb/s
- Cache: 16MB
- Capacity: 600GB
- Areal density (avg): 252Gb/in2
- Guaranteed Sectors: 1.172.123.569
- Spindle Speed: 10.000 RPM
- Sustained data transfer rate: 141Mb/s
- Average latency: 3,0ms
- Random read seek time: 3,89ms
- Random write seek time: 4,54ms
- I/O data transfer rate: 600MB/s
- MTBF: 2.000.000 hours

Acesso aos dados em disco

- Vamos considerar um disco típico para *desktop*:

`Seagate Barracuda ES.2 SATA 1TB`

- Interface: SATA 3Gb/s
- Cache: 16MB
- Capacity: 1TB
- Areal density (avg): -
- Guaranteed Sectors: 1.953.525.168
- Spindle Speed: 7.200 RPM
- Sustained data transfer rate: -
- Average latency: 4,16ms
- Random read seek time: 8,5ms
- Random write seek time: 9,5ms
- I/O data transfer rate: 300MB/s
- MTBF: 1.200.000 hours

Acesso aos dados em disco

- Vamos considerar um disco típico:
  - A velocidade angular dos discos (RPM): 10.000 RPM
  - O tempo médio para posicionar as cabeças (seek time): 6 ms
  - A taxa de transferência: 100 MBytes/s
- Tempo para acessar um bloco típico de 8 registros (4096 KB):
  - Posicionar as cabeças: 6 ms
  - Latência rotacional (10.000 RPM)=tempo de 1/2 rotação: 3 ms
  - Tempo de transferência (4.000 KBytes, 100 MBytes/s): 0,05ms

  ☞ Portanto $6 + 3 + 0,05 \approx 9,05$ms.
- Tempo para acessar o próximo registro na mesma operação:
  - Tempo de transferência (4.000 KBytes, 100 MBytes/s): 0,05 ms

  ☞ Portanto $\approx 0,05$ms (se estiver no mesmo cilindro).
- Tempo para acessar dois registros na mesma operação: $\approx 9,10$ms.

Acesso aos dados em disco

- Tempo para acessar o próximo bloco de 8 registros (4096 KB) numa operação separada:
  - As cabeças já estão na trilha correta, portanto posicionar as cabeças: 0 ms
  - Latência rotacional (10.000 RPM)=tempo de 1/2 rotação: 3 ms
  - Tempo de transferência (4.000 KBytes, 100 MBytes/s): 0,05 ms
- ☞ Portanto $\approx 3,05$ms.
- Tempo para acessar dois registros em duas operações consecutivas: $\approx 12,1$ms.
- Tempo para acessar 20 KBytes em 5 operações de 4 Kbytes: $\approx 9,05 + 4 * (3,05) = 21,25$ms
- Tempo para acessar 20 KBytes em uma operação: $\approx 9,05 + 4 * (0,05) = 9,25$ms

Acesso aos dados em disco

Portanto, é importante acessar o máximo possível de dados em uma única operação.

Problemas:

- Necessidade de memória para armazenar todos os dados;
- Os dados tem que estar em registros contínuos no disco.

Solução:

Gerenciar os dados em uma estrutura que permita agrupar blocos de registros contínuos, segundo as necessidades.

Estrutura de uma base de dados em disco

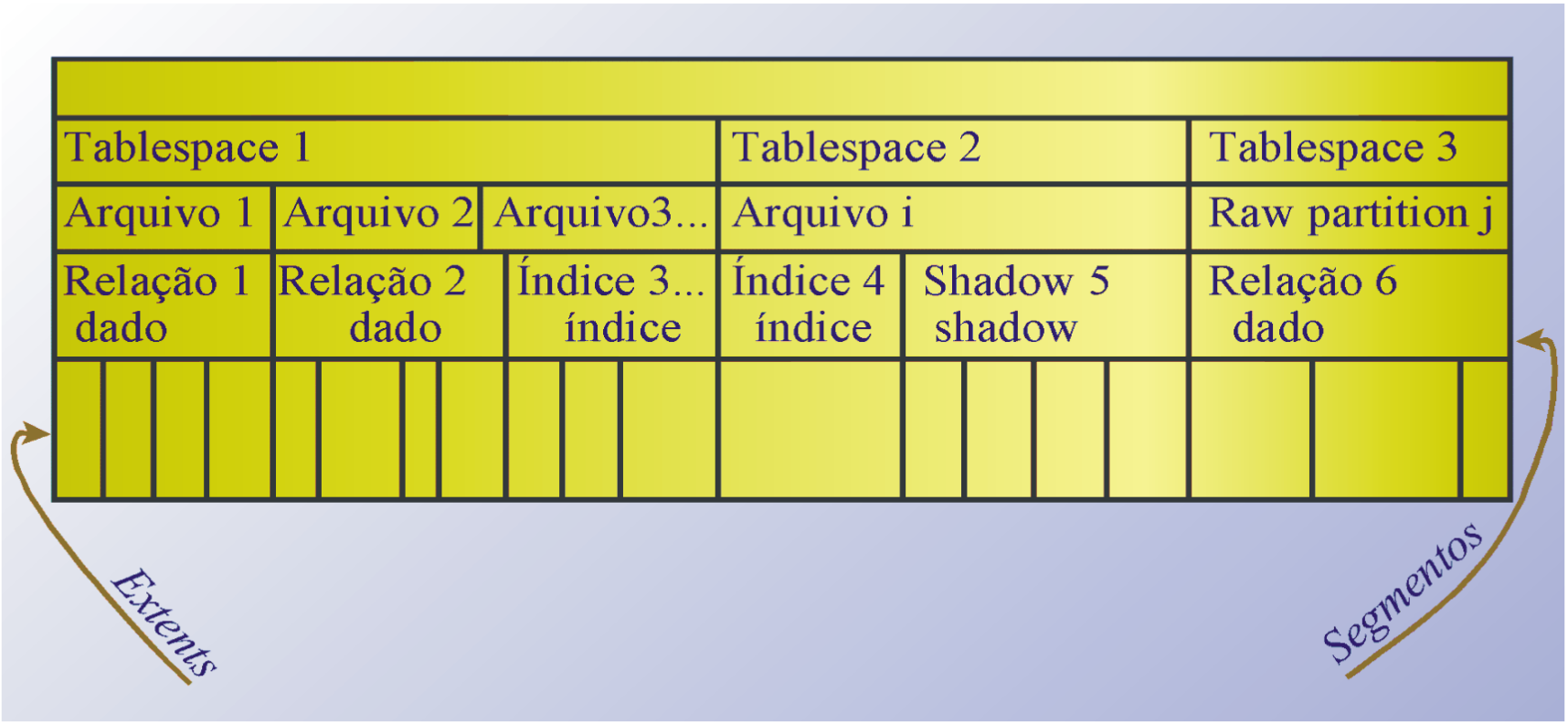
- Cada relação é guardada como uma seqüência de tuplas, de tamanho fixo ou aproximadamente fixo (média).
- Pode haver espaços em branco (não usados).
- Agrupam-se as tuplas em blocos de tamanho fixo, chamados páginas,
  - de tamanho em potência de dois de registros em disco.
- Uma página é sempre lida inteira.
- Agrupam-se as páginas em blocos contínuos, chamados *extents*,
- Acessar todas as páginas de um *extent* numa só operação é mais eficiente.

Estrutura de uma base de dados em disco

- Existem basicamente
  - *extent* de dados,
  - *extent* de índices,
  - *extent* de sombra, e
  - *extent* temporários (ou de *overflow*),
- cada tipo com propriedades diferentes (tamanho, forma de agrupamento e acesso, etc.)
- Extents que se destinam ao mesmo objeto gerenciado são agrupadas em um mesmo segmento.
  - Cada tabela é armazenada em um segmento de dados;
  - Uma tabela pode ter tantos segmentos de índices quanto necessário;
  - Páginas sombra (*shadow*) são mantidas por transação;
  - Páginas temporárias duram no máximo até o final do *thread* corrente.

Estrutura de uma base de dados em disco

A estrutura típica de uma base de dados em disco é a seguinte (Oracle):



Estrutura de uma base de dados em disco

- **Tablespace**
  - Pode ser composto por um ou mais arquivos do sistema operacional,
  - ou por *raw disk* partitions.
  - Potencialmente, pode envolver diversos discos.
- Cada "objeto" da base é alocado em um **segmento**
  - Dados de uma relação,
  - um índice,
  - sombra de uma relação por transação, etc.
- Cada **segmento** é composto por
  - um $\Rightarrow$ *extent* inicial,
  - Tantos $\Rightarrow$ *extents* de extensão quanto necessário.

Estrutura de uma base de dados em disco

- O tamanho de um *extent* é múltiplo do tamanho da página,
- Um *extent* é gravado num cilindro em setores contínuos, de maneira que sua leitura é feita numa única operação.
- Páginas de dados podem ser alocadas em *extents* de qualquer tamanho (dentro dos limites do *hardware* e das partições),
- Páginas índice e *shadow* têm tamanhos pequenos (mínimos ?)

Estrutura de uma base de dados em disco

- *Extents* são interessantes para permitir leitura contínua de blocos de dados grandes, mas a unidade básica de operações de leitura/escrita em disco é a página.
- A estrutura de *buffer pool* em memória é pré-formatada para gerenciar *buffers* onde todos têm o tamanho de uma página em disco.
- Portanto, embora possam ser lidas/escritas muitas páginas em uma mesma operação, cada página tem que ter sua estrutura íntegra, pois em situações extremas de pouca memória disponível, as páginas podem ser lidas uma a uma.

Componentes da Estrutura Física da Base de Dados em Disco

- Recordando: Componentes da estrutura física de uma base de dados em disco:

Registro Físico $\xrightarrow{\boxed{\text{Potência de 2}}}$ Página $\xrightarrow{\boxed{\text{Inteiro}}}$ Extent $\xrightarrow{\boxed{\text{Um objeto}}}$ Segmento $\xrightarrow{\boxed{\text{Partição}}}$ *Tablespace* $\longrightarrow$ Base de Dados

Especificação da Estrutura Física de uma base de dados

- A estrutura física de uma base de dados é especificada pela definição:
  - dos *tablespaces*,
  - das relações,
  - e dos índices,
- nos respectivos comandos `CREATE`.
- *Tablespaces* podem ser declarados explicitamente usando o comando `CREATE TABLESPACE`, cuja sintaxe varia muito entre os produtos. Exemplos de sintaxe em Postgres e Oracle são:

Comando CREATE TABLESPACE

Postgres

CREATE TABLESPACE – Postgres

```
CREATE TABLESPACE tblspc-name [LOCATION file-name]
       <outras...>
```

Comando CREATE TABLESPACE

Oracle

CREATE TABLESPACE – Oracle

```
CREATE TABLESPACE tblspc-name
      [DATAFILE file-name [SIZE ext-size] [REUSE]]
          AUTOEXTEND OFF |
          AUTOEXTEND ON [NEXT add-size [MAXSIZE {UNLIMITED |
                                                max-size}],
      [DEFAULT STORAGE ([INITIAL init-size] [NEXT next-size]
           [MINEXTENTS n] MAXEXTENTS {n | UNLIMITED}]
           [PCTINCREASE n])
       [MINIMUM EXTENT min-size]
       <outras...>
```

onde size é <integer>[K, M, G, T, P, E]

Comando `CREATE TABLESPACE`

A cláusula `LOCATION` em Postgres ou `DATAFILE` em Oracle servem para indicar onde os dados serão armazenados.

- Os dados são armazenados:
  - Em um arquivo do sistema operacional;
  - ou em um disco (ou partição), chamados *raw disk partition*.
- Postgres somente usa arquivos do sistema operacional.
- *Raw partitions* são a opção preferida quando se deseja confiabilidade e eficiência.
- Arquivos do Sistema são adequados para as máquinas de desenvolvimento, já que não é necessário criar uma partição para a base.

Comando CREATE TABLESPACE

Oracle

```
[DATAFILE file-name [SIZE ext-size] [REUSE]]
    AUTOEXTEND OFF |
    AUTOEXTEND ON [NEXT add-size [MAXSIZE {UNLIMITED |
                                           max-size}],
```

Os parâmetros da cláusula DATAFILE se referem ao *tablespace* como um todo.

- Quando é especificado um arquivo do sistema, ele pode existir ou não.
- Se não existir, ele é criado com SIZE ext-size.
- Se existir e for usado REUSE, o arquivo terá o tamanho existente.
- Quando o *extent* for completamente usado e precisar crescer, se for permitido AUTOEXTEND ON, o aumento é igual ao valor inicial de SIZE ext-size, a menos que seja indicado o valor de aumento NEXT add-size.

Comando CREATE TABLESPACE

Oracle

```
[DATAFILE file-name [SIZE ext-size] [REUSE]]
    AUTOEXTEND OFF |
    AUTOEXTEND ON [NEXT add-size [MAXSIZE {UNLIMITED |
                                           max-size}],
```

Os parâmetros da cláusula DATAFILE se referem ao *tablespace* como um todo.

- Se o *extent* puder crescer, pode ser indicado o tamanho máximo que ele pode atingir MAXSIZE max-size},
- ou pode ser indicado que não ha limite MAXSIZE UNLIMITED.

Comando CREATE TABLESPACE

Oracle

```
[DEFAULT STORAGE ([INITIAL init-size] [NEXT next-size]
     [MINEXTENTS n1] MAXEXTENTS {n2 | UNLIMITED}]
     [PCTINCREASE n3])
        [MINIMUM EXTENT min-size]
```

- Os parâmetros da cláusula `STORAGE` se referem aos *extents* dos objetos armazenados nesse *tablespace*.
- Um *extent* pode ter tamanho entre 2KB e 4095MBytes,
- mas note que não adianta ser maior do que um cilindro no disco.
- Os tamanhos indicados em `INITIAL init-size` e `NEXT next-size` são sempre um múltiplo de tamanhos de páginas, em Kilobytes.
- Se outros tamanhos forem indicados, eles são arredondados para cima.
- A cláusula `STORAGE` é opcional, e pode ser usada para indicar:

Comando CREATE TABLESPACE

Oracle

```
[DEFAULT STORAGE ([INITIAL init-size] [NEXT next-size]
     [MINEXTENTS n1] MAXEXTENTS {n2 | UNLIMITED}]
     [PCTINCREASE n3])
        [MINIMUM EXTENT min-size]
```

- `INITIAL init-size` – o tamanho em bytes do *extent* inicial de cada objeto (4K ☞ 4096 bytes). O default é 5 blocos.
- `NEXT next-size` – o tamanho em bytes dos *extent* de extensão de cada objeto. O default é 5 blocos.
- `MINEXTENTS n1` – o número mínimo de *extents* com que cada segmento é criado. O default é 50.
- `MAXEXTENTS {n2 | UNLIMITED}` o número máximo de *extents* que cada segmento pode ter. O default é `UNLIMITED`.
- `PCTINCREASE n3` é possível que os *extents* de extensão sejam cada vez maiores, pelo fator `n3`. O default é 50.

Comando CREATE TABLESPACE

- Note-se que o tamanho de cada bloco é um valor fixo para uma base de dados.
- Em Postgres, ele é definido na instalação do servidor. O Default é 8KB.
- Em Oracle, ele é definido pela instância do sistema (pelo parâmentro DB_BLOCK_SIZE), O Default é 4KB.
- O tamanho do bloco de dados de qualquer SGBD deve sempre ser uma potência de 2 em Kilobytes.

Comando `CREATE TABLESPACE`

Oracle

`[MINIMUM EXTENT min-size]`

- O valor `min-size` permite garantir um tamanho mínimo para os *extents* deste *tablespace*, sobrepondo-se ao valor eventualmente indicado nos `CREATE TABLE` e `CREATE INDEX`.

Comando `CREATE TABLE`

- O comando de declaração de tabelas tem construções especiais para indicar as *tablespaces* que devem usar e como devem ser usados os *extents* para elas.
- O comando `CREATE TABLE` tem uma definição geral bem mais padronizada do que o comando `CREATE TABLESPACE`.
- No entanto, a parte do comando que indica a estrutura física das tabelas tem uma sintaxe que também varia muito entre os produtos.

Comando CREATE TABLE

Postgres

CREATE TABLE – Postgres

```
CREATE TABLE tbl-name (definição de atributos e restrições)
    [WITH (fillfactor (integer))
    [TABLESPACE tablespace]
```

as restrições de chave primária e chave candidata podem ter as cláusulas

Cláusulas UNIQUE e PRIMARY KEY – Postgres

```
    [WITH ((fillfactor (integer))
    [USING INDEX TABLESPACE tablespace]
```

Comando CREATE TABLE

Postgres

`TABLESPACE tablespace`

- Indica em qual *tablespace* será criado o segmento de dados da tabela. Se não for indicado, é usado o segmento default.

`USING INDEX TABLESPACE tablespace` nas cláusulas `{UNIQUE` e `PRIMARY KEY`

- Indica em qual *tablespace* será criado o segmento de índice desse índice. Se não for indicado, é usado o mesmo *tablespace* onde está o segmento de dados da tabela.

Comando CREATE TABLE

Postgres

`WITH (fillfactor (integer))`

- Indica a taxa máxima de inserção em comandos `INSERT`, para blocos de segmentos de dados ou de índices respectivamente.
- Pode variar de 10 a 100 (100 é bloco cheio). O default é 100.
- O default de 100 é o ideal para tabelas que não sofrem atualização.
- Para tabelas que podem ser atualizadas, deve ser dado um valor usualmente entre 70 e 90.

Comando CREATE TABLE

Oracle

CREATE TABLE – Oracle

```
CREATE TABLE tbl-name (definição de atributos e restrições)
    [TABLESPACE tablespace]
    [STORAGE ([INITIAL init-size] [NEXT next-size]
           [MINEXTENTS n1] MAXEXTENTS {n2 | UNLIMITED}]
           [PCTINCREASE n3])
     [PCTFREE n4] [PCTUSED n5]
```

onde a cláusula STORAGE tem o mesmo significado que tem no comando CREATE TABLESPACE.

Comando CREATE TABLE

Oracle

`[PCTFREE n4] [PCTUSED n5]`

- A declaração `[PCTFREE n4]` tem significado parecido ao da declaração `WITH (fillfactor (integer))` do Postgres, mas com o valor invertido (`[PCTFREE n4]` indica bloco cheio). O default é 10 e o máximo é 99.
- Só que no Postgres, se algum bloco ficar com utilização inferior à especificada devido a comandos `DELETE` ou `UPDATE`, novos comandos `INSERT` podem voltar a inserir nesse bloco.
- Em Oracle, as novas inserções só voltam a usar um bloco se sua taxa de ocupação cair abaixo do valor dado na na declaração `[PCTUSED n5]`.
- O default de `[PCTUSED n5]` é 40.
- Note-se que o `n4+n5` não pode ser maior do que 100.

Comando `CREATE INDEX`

- O comando de declaração de índices tem diversas variações entre os diversos produtos, mas elas são equivalentes às variações que cada produto tem para os comandos `CREATE TABLESPACE` e `CREATE TABLE`.
- Assim, apenas relembramos aqui a sintaxe do comando `CREATE INDEX` já mostrada anteriormente, especializada nas construções específicas dos gerenciadores:

Comando CREATE INDEX

Postgres

CREATE INDEX – Postgres

```
CREATE [UNIQUE] INDEX idx-name
     ON table [USING method]
    ({column | (expression)} [ASC | DESC][, ...])
    [WITH ((fillfactor (integer))
    [TABLESPACE tablespace]
    [WHERE predicate]
       <outras...>
```

Comando CREATE INDEX

Oracle

CREATE INDEX – Oracle

```
CREATE [UNIQUE] [BITMAP] INDEX idx-name ON table
    ({column | (expression)} [ASC | DESC][, ...])
    [TABLESPACE tablespace]
    [STORAGE ([INITIAL init-size] [NEXT next-size]
       [MINEXTENTS n1] MAXEXTENTS {n2 | UNLIMITED}]
       [PCTINCREASE n3])
[PCTFREE n4]
    [WHERE predicate]
       <outras...>
```

Estrutura de uma base de dados em um disco

Todos os Métodos de Acesso a Disco são construídos para utilizar a estrutura da base de dados como uma sequencia de extents iniciais e de extensão agrupados em segmentos, os quais são armazenados nos discos, de preferência em *raw partitions*, visando tornar o mais eficiente possível o acesso dos dados em disco.

Arquitetura de SGBD Relacionais
— Organização da Memória em Disco para SGBD Relacionais —

Caetano Traina Jr.

Grupo de Bases de Dados e Imagens
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - São Carlos
caetano@icmc.usp.br

5 de maio de 2010
São Carlos, SP - Brasil

FIM