

# Arquitetura Geral de um Sistema de Gerenciamento de Bases de Dados Relacional

**Prof. Dr. Ives Renê V. Pola**

[ivesr@utfpr.edu.br](mailto:ivesr@utfpr.edu.br)

Departamento Acadêmico de Informática – DAINF  
UTFPR – Pato Branco DAINF  
UTFPR  
Pato Branco - PR

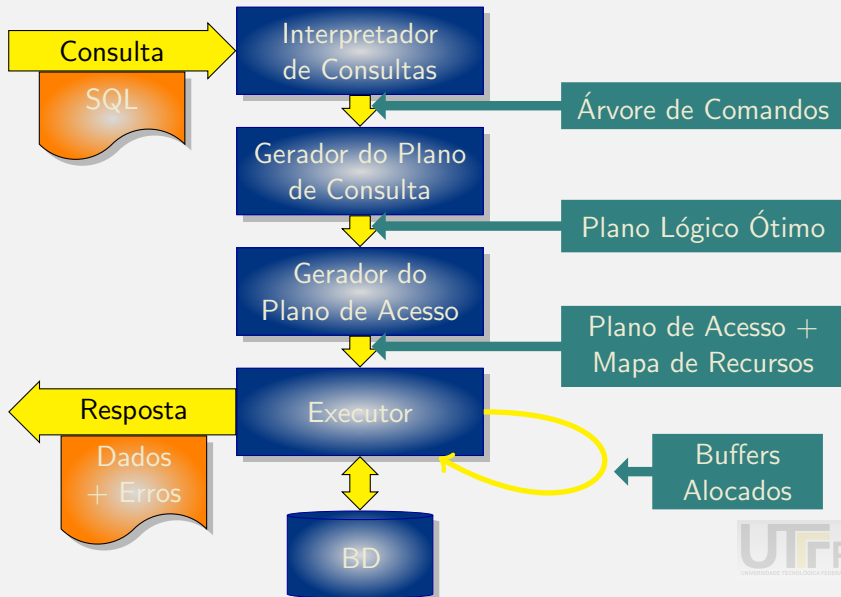
Esta apresentação mostra os principais conceitos da arquitetura de um servidor de SGBD Relacional.



# Roteiro

- 1 Conceitos gerais
- 2 O Interpretador de Consultas
- 3 O Otimizador Lógico
- 4 O Otimizador do Plano de Acesso Físico
- 5 O Executor
- 6 O que e como deve/pode ser automatizado

# Arquitetura de um SGBDR

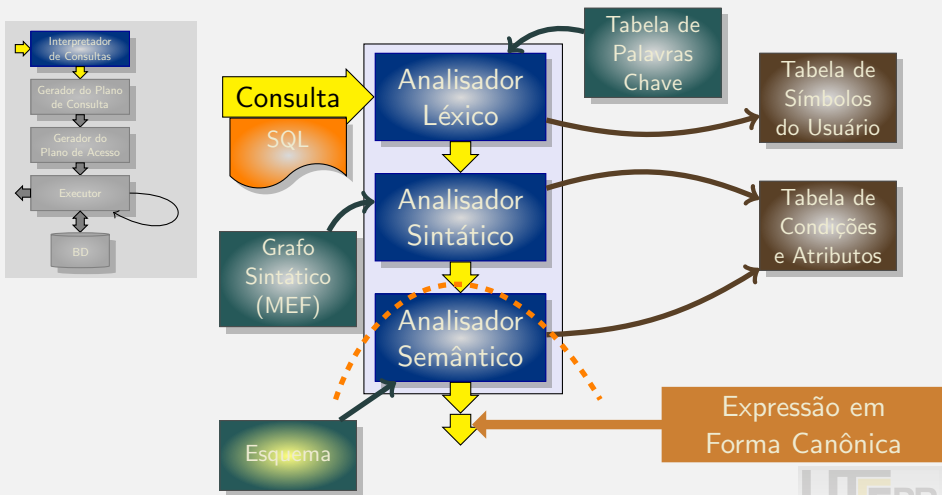


# Classes de comandos 'que devem ser executados

- Comandos da DCL
  - CREATE DATABASE, SET TRANSACTION, etc.  
São apenas compilados, mas têm execução sequencial
- Comandos da DDL
  - CREATE TABLE, ALTER TABLE, DROP INDEX, etc.  
São 'traduzidos' para comandos da DML
- Comandos da DML
  - INSERT INTO
  - DELETE
  - UPDATE
  - SELECT
- A essência do processamento otimizado é o tratamento das cláusulas dos comandos SELECT, UPDATE e DELETE,
  - e o processamento é concentrado na execução das operações de busca das tuplas que devem ser mostradas/atualizadas/apagadas pelo comando.
- O problema a ser resolvido é **Recuperar** as tuplas a serem afetadas, e daí fazer com elas o que o comando indica: mostrar/atualizar/apagar.

# Interpretador de Consultas

...tal e qual um compilador para linguagens de programação



# Interpretador de Consultas

```
SELECT <list_atr1>
FROM <relações>
WHERE <cond1>
GROUP BY <list_atr2>
HAVING <cond2>
ORDER BY <list_atr3>
```

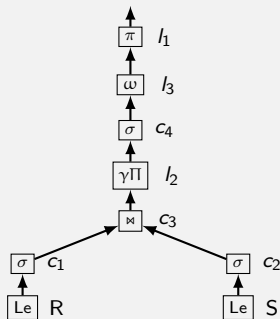
- Ler todas as <relações>, gera  $\text{Le}_{(\text{relacao})}$
- Processar as <cond<sub>1</sub>>
  - Se  $\langle r_i.\text{atr}_1 \theta \text{cte} \rangle$ , gera  $\sigma_{(\text{cond})}$
  - Se  $\langle r_i.\text{atr}_1 \theta r_j.\text{atr}_1 \rangle$  e  $i=j$ , gera  $\sigma_{(\text{cond})}$
  - Se  $\langle r_i.\text{atr}_1 \theta r_j.\text{atr}_1 \rangle$  e  $i \neq j$ , gera  $r_i \bowtie_{(\text{cond})} r_j$
- Se houver mais de uma tabela, gera  $r_i \times r_j$
- Processar atrib. <list\_atr<sub>2</sub>> agregados, gera  $\gamma \rightarrow \Pi_{\{\langle \text{list\_atr1} \rangle \cup \langle \text{list\_atr2} \rangle \cup \langle \text{cond2.atr} \rangle\}}$
- Processar as <cond<sub>2</sub>>, gera  $\sigma_{(\text{condicao})}$
- Processar a <list\_atr<sub>3</sub>>, gera  $\omega_{(\langle \text{lista\_atr} \rangle)}$
- Processar a <list\_atr<sub>1</sub>>, gera  $\pi_{\{\langle \text{lista\_atr} \rangle\}}$

# Interpretador de Consultas

```

SELECT R.a, S.b
FROM R, S
WHERE R.a = 10
AND S.b > S.c
AND R.d = S.e
GROUP BY R.a, S.b, R.x
HAVING count(R.y) > 5
ORDER BY count(R.y)

```



Ler todas as <relações>, gera  $Le_{(relacao)}$ .  
 Processar as <cond<sub>1</sub>>  
 Se  $\langle r_j.ATR_1 \theta cte \rangle$ , gera  $\sigma_{(condicao)}$ .  
 Se  $\langle r_j.ATR_1 \theta r_j.ATR_1 \rangle$  e  $i=j$ , gera  $\sigma_{(condicao)}$ .  
 Se  $\langle r_j.ATR_1 \theta r_j.ATR_1 \rangle$  e  $i \neq j$ , gera  $r_j \bowtie_{(condicao)} r_j$ .  
 Se houver mais de uma tabela, gera  $r_i \times r_j$ .  
 Processar os atributos <list\_atr<sub>2</sub>> agregados, gera  
 $\gamma \rightarrow \Pi \{ \langle listatr1 \rangle \cup \langle listatr2 \rangle \cup \langle cond2.ATR \rangle \}$ .  
 Processar as <cond<sub>2</sub>>, gera  $\sigma_{(condicao)}$ .  
 Processar a <list\_atr<sub>3</sub>>, gera  $\omega_{\langle lista\_atr \rangle}$ .  
 Processar a <list\_atr<sub>1</sub>>, gera  $\pi_{\langle lista\_atr \rangle}$ .

TCA

rótulo	condição
c <sub>1</sub>	R.a = 10
c <sub>2</sub>	S.b > S.c
c <sub>3</sub>	R.d = S.e
l <sub>2</sub>	R.a, S.b, R.x
c <sub>4</sub>	count(R.y) > 5
l <sub>3</sub>	count(R.y)
l <sub>1</sub>	R.a, S.b

TSU

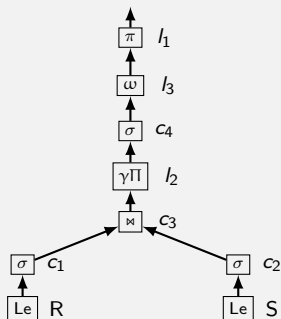
1	R.a	atr	R
2	R	rel	R
3	S.b	atr	S
	...		

# Interpretador de Consultas

```

SELECT R.a, S.b
FROM R, S
WHERE R.a = 10
AND S.b > S.c
AND R.d = S.e
GROUP BY R.a, S.b, R.x
HAVING count(R.y) > 5
ORDER BY count(R.y)

```



Expressão em Forma Canônica:

1	Le	R		
2	Le	S		
3	$\sigma$	1		$c_1$
4	$\sigma$	2		$c_2$
5	$\bowtie$	3	4	$c_3$
6	$\gamma\Pi$	5		$l_2 \cup \dots$
7	$\sigma$	6		$c_4$
8	$\omega$	7		$l_3$
9	$\pi$	8		$l_1$

TCA

rótulo	condição
$c_1$	$R.a = 10$
$c_2$	$S.b > S.c$
$c_3$	$R.d = S.e$
$l_2$	$R.a, S.b, R.x$
$c_4$	$\text{count}(R.y) > 5$
$l_3$	$\text{count}(R.y)$
$l_1$	$R.a, S.b$

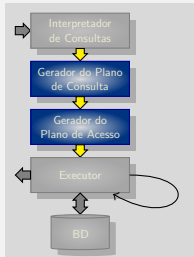
TSU

1	R.a	atr	R
2	R	rel	R
3	S.b	atr	S
	...		



# Otimizadores

## Otimizadores de Expressão e de Caminho de Acesso



- Gerador do Plano de Consulta



Otimizador da expressão algébrica  
(Otimizador Lógico, ou *Rule Based Optimizer* (RBO))

- Gerador de Planos
- Avaliador de Planos (lógicos)

- Gerador do Plano de Acesso

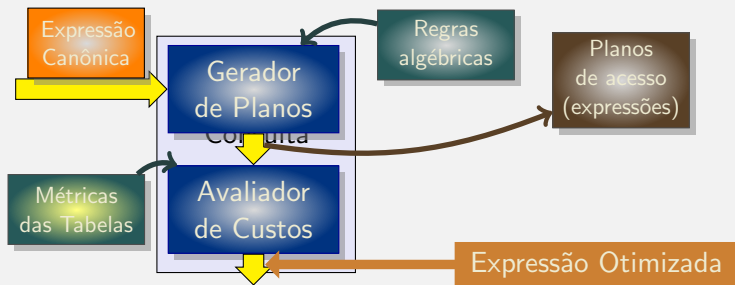
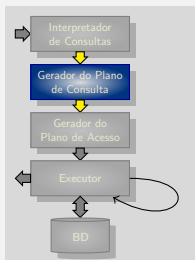


Otimizador do caminho de acesso  
(Otimizador Físico, ou *Cost Based Optimizer* (CBO))

- Controle de Concorrência
- Otimizador de Recursos.

# Gerador do Plano de Consulta

## Otimizador Lógico



O Gerador de Planos de Consulta é baseado nas Regras Algébricas que existem para os operadores relacionais.

Revisão de  
Álgebra Relacional

# Regras da álgebra relacional

Seja  $R$  uma relação e  $c_1$  e  $c_2$  condições definidas sobre  $R$ . Então:

Operador SELECT -  $\sigma$

$\sigma_{(c_1)}(\sigma_{(c_2)}R) \Leftrightarrow \sigma_{(c_2)}(\sigma_{(c_1)}R)$  (Comutativa) 

$\sigma_{(c_1)}(\sigma_{(c_1)}R) \Leftrightarrow \sigma_{(c_1)}R$  (Idempotente)

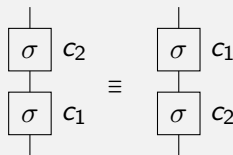
$\sigma_{(c_1)}(\sigma_{(c_2)}R) \Leftrightarrow \sigma_{(c_1 \wedge c_2)}R$

$\sigma_{(c_1 \wedge c_2)}R \Leftrightarrow \sigma_{(c_1)}R \cap \sigma_{(c_2)}R$

$\sigma_{(c_1 \vee c_2)}R \Leftrightarrow \sigma_{(c_1)}R \cup \sigma_{(c_2)}R$

$\sigma_{(\neg c_1)}R \Leftrightarrow R - \sigma_{(c_1)}R$

...



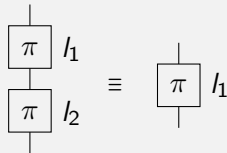
i	$\sigma$	k		$c_1$
j	$\sigma$	i		$c_2$
	...			

$\Leftrightarrow$

i	$\sigma$	j		$c_1$
j	$\sigma$	k		$c_2$
	...			

# Regras da álgebra relacional

Seja  $R$  uma relação e  $l_1$  e  $l_2$  sub-conjuntos de atributos de  $R$ . Então:



Operador PROJECT -  $\pi$

se  $l_1 \subseteq l_2$ , então:  $\pi_{l_1}(\pi_{l_2} R) \Leftrightarrow (\pi_{l_1} R)$

i	$\pi$	k		$l_2$
j	$\pi$	i		$l_1$
	...			

$\Leftrightarrow$

i	$\pi$	k		$l_1$
~~~~~				
	...			

# Regras da álgebra relacional

Sejam  $R_1$ ,  $R_2$  e  $R_3$  relações e  $c_1$  e  $c_2$  condições definidas sobre as relações às quais se aplicam. Então:

Operador JOIN -  $\bowtie$

$$R_1 \bowtie_{c_1} R_2 \Leftrightarrow R_2 \bowtie_{c_1} R_1 \text{ (Comutativa)}$$

Se  $c_1$  envolver apenas atributos de  $R_1$  e de  $R_2$ , e  $c_2$  envolver apenas atributos de  $R_2$  e  $R_3$ , então:

$$(R_1 \bowtie_{c_1} R_2) \bowtie_{c_2} R_3 \Leftrightarrow R_1 \bowtie_{c_1} (R_2 \bowtie_{c_2} R_3) \text{ (Associativa)}$$

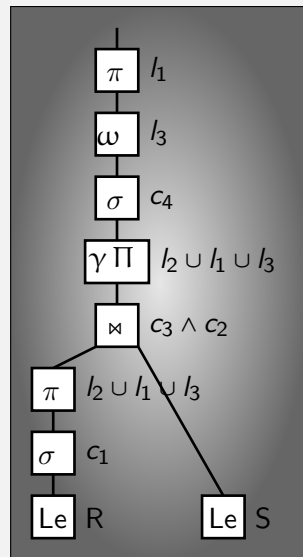
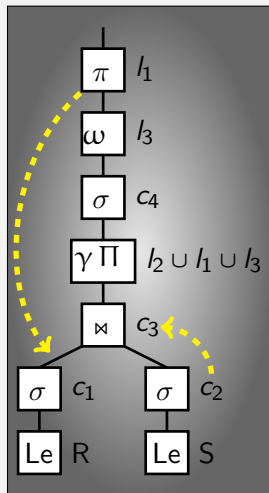
... (operadores União, Intersecção, Diferença, Produto Cartesiano e as respectivas operações distributivas).

## Gerador de Planos de Consulta

```

SELECT R.a, S.b
FROM R, S
WHERE R.a = 10
AND S.b > S.c
AND R.d = S.e
GROUP BY R.a, S.b, R.x
HAVING count(R.y) > 5
ORDER BY count(R.y)

```



# Gerador de Planos de Consulta

E se houvessem diversas tabelas e operadores de junção?  
(várias condições em seqüência)

```
SELECT *  
FROM R, S, T, U  
WHERE R.a = S.b  
AND S.c = T.d  
AND T.e = U.f  
...
```

Quais junções executar antes?

Depende da seletividade das condições.

# Seletividade das Condições

## Operadores de Seleção e Junção

$$Seletividade(C_i) = 1 - \frac{\text{Numero de tuplas no resultado}}{\text{Numero de tuplas na entrada}}$$

Como prever a seletividade das condições?



# Previsão de Seletividade

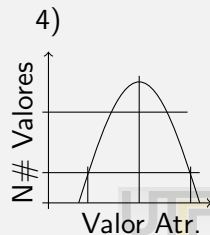
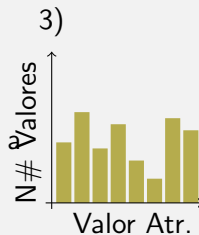
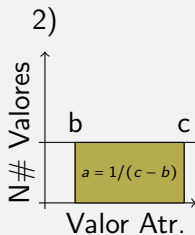
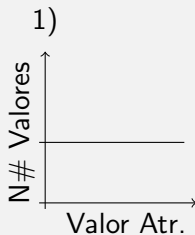
- A previsão de seletividade depende de métricas mantidas pelo Gerenciador e inclui informações sobre:
  - Medidas sobre a [Base de Dados](#),
  - Medidas sobre as [Tabelas](#),
  - Medidas sobre cada [Atributo](#) em cada tabela,
  - Medidas sobre cada [Chave](#) em cada tabela...
- As Métricas (estatísticas) são obtidas:
  - durante cada operação de consulta,
  - pela execução de comandos específicos para coletá-las
    - automaticamente (periodicamente, quando algum evento pré-definido ocorre, etc.),
    - por solicitação do DBA.

# Previsão de Seletividade

- Existem Métricas sobre:
  - A base de dados:
    - Tamanho dos registros físicos
  - Cada tabela:
    - Número de registros físicos,
    - Tamanho de tupla em disco,
    - Número de tuplas gravadas,
    - Número máximo de tuplas já existentes
  - Cada atributo em cada tabela:
    - Tamanho do atributo em disco,
    - Tamanho do domínio ativo,
    - Distribuição de valores,
    - ...
  - ...

# Previsão de Seletividade


- Métricas sobre a distribuição de valores de cada atributo em cada tabela:
  - Constante.** Assume um valor constante para a seletividade;
  - Distribuição Uniforme.** Cada valor contribui  $1/|\text{Domínio ativo}|$ ;
  - Histograma.** Deve ser mantido um contador para cada valor do domínio ativo (arquivo invertido);
  - Distribuição estatística.** Armazena menor e maior valor do domínio ativo, mais dados sobre a distribuição estatística assumida (normal, etc.).



# Gerador de Planos de Consulta


Quando houverem diversas tabelas e diversos operadores de junção, e/ou diversas condições de seleção, e/ou subconsultas aninhadas (possivelmente envolvendo agregações e junções externas), etc... em uma mesma consulta:

```
SELECT *  
  FROM R, S, T, U Outer Join V on U.g=V.h  
 WHERE R.a = S.b  
       AND S.c = T.d  
       AND T.e ≤ U.f  
       AND T.f IN (SELECT ...)  
       . . .
```

 a quantidade de expressões equivalente é muito grande e torna-se impraticável avaliar todas antes de se escolher a melhor opção.

# Gerador de Planos de Consulta

- Quando quantidade de expressões equivalente é muito grande e torna-se impraticável avaliar todas antes de se escolher a melhor opção,

 O melhor plano de execução depende da seletividade das condições envolvidas nos diversos operadores, para reduzir os recursos necessários (memória, processador, rede), sempre priorizando minimizar o **tempo total** de execução (o que depende fundamentalmente da quantidade de acessos a disco).

# O Avaliador de Planos Lógicos

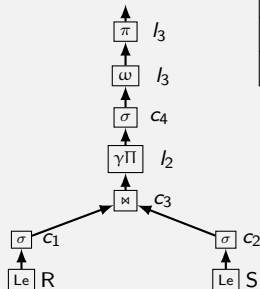
- A otimização visa reduzir:
  - O número de acessos a disco,
  - O uso de memória,
  - O uso do processador (ou dos processadores).
- Para a previsão de custo de cada operador, considera-se apenas o  
Número de acessos a disco para leitura +  
Número de acessos a disco para escrita
  - pois quanto mais registros são necessários, mais
    - Acessos a disco,
    - Memória e
    - Tempo de processamento.
  - são, proporcionalmente, necessários. Assim, a previsão considera apenas números de acesso a disco, o que é mais facilmente calculado.
  - Em particular, considera-se apenas o número de acessos a disco para escrita e leitura (abstraindo-se que quanto mais se lê, mais se escreve).

# O Avaliador de Planos Lógicos

```

SELECT R.a, S.b
FROM R, S
WHERE R.a = 10
AND S.b > S.c
AND R.d = S.e
GROUP BY R.a, S.b, R.x
HAVING count(R.y) > 5
ORDER BY count(R.y)

```



Expressão do plano:

1	Le	R			v <sub>1</sub>
2	Le	S			v <sub>2</sub>
3	$\sigma$	1		c <sub>1</sub>	v <sub>3</sub>
4	$\sigma$	2		c <sub>2</sub>	v <sub>4</sub>
5	$\bowtie$	3	4	c <sub>3</sub>	v <sub>5</sub>
6	$\gamma\Pi$	5		l <sub>2</sub> ..	v <sub>6</sub>
7	$\sigma$	6		c <sub>4</sub>	v <sub>7</sub>
8	$\omega$	7		l <sub>3</sub>	v <sub>8</sub>
9	$\pi$	8		l <sub>1</sub>	v <sub>9</sub>

Nova coluna,  
indicando o numero  
de acessos a disco  
necessários para  
escrever o resultado.

Por exemplo,

- operador “Le”: lê e escreve no *cache*. Total de escritas:

$$v_i = \# \text{Páginas na tabela lida (NPag)}$$

- operador  $\sigma$ :

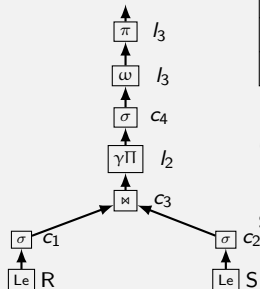
$$v_i = \# \text{Págs escritas pelo op. anterior} * (1 - \text{Sel}(\text{cond}))$$

# O Avaliador de Planos Lógicos

```

SELECT R.a, S.b
FROM R, S
WHERE R.a = 10
AND S.b > S.c
AND R.d = S.e
GROUP BY R.a, S.b, R.x
HAVING count(R.y) > 5
ORDER BY count(R.y)

```



Expressão do plano:

1	Le	R			v <sub>1</sub>
2	Le	S			v <sub>2</sub>
3	σ	1		c <sub>1</sub>	v <sub>3</sub>
4	σ	2		c <sub>2</sub>	v <sub>4</sub>
5	⋈	3	4	c <sub>3</sub>	v <sub>5</sub>
6	γΠ	5		l <sub>2</sub> ..	v <sub>6</sub>
7	σ	6		c <sub>4</sub>	v <sub>7</sub>
8	ω	7		l <sub>3</sub>	v <sub>8</sub>
9	π	8		l <sub>1</sub>	v <sub>9</sub>

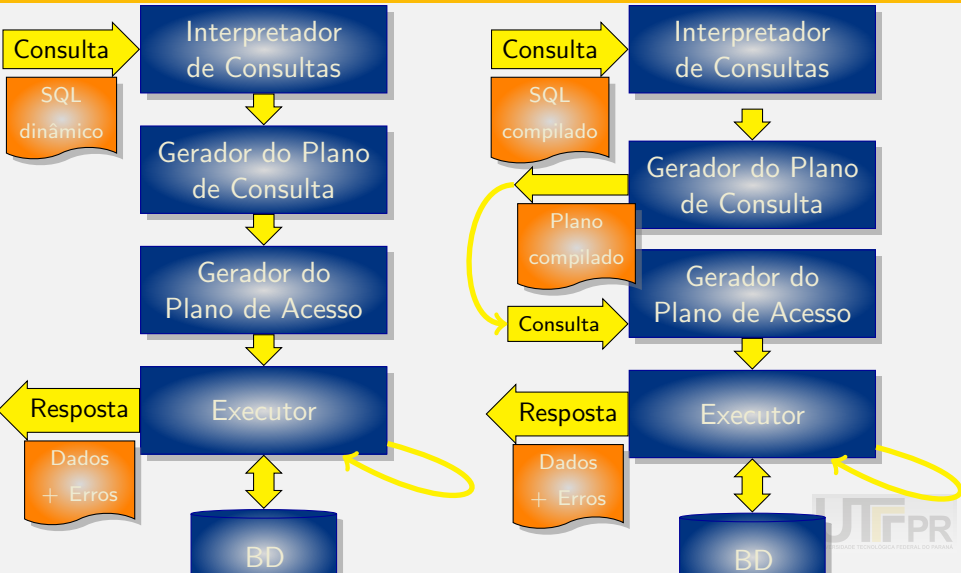
Total

O custo estimado de cada plano é o total de páginas previstas para serem escritas em todos os seus operadores algébricos.



# Arquitetura de um SGBDR

SELECT + UPDATE + DELETE



# Arquitetura de um SGBDR

INSERT + DDL + DCL

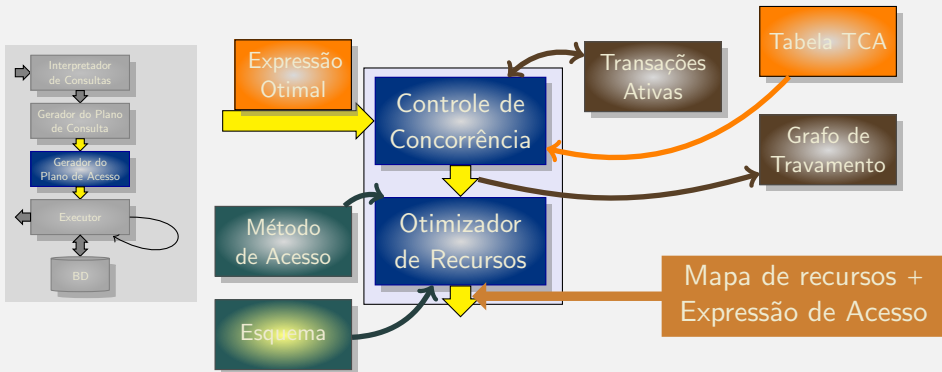


Acessar o esquema é um **HOT SPOT!**

- Compilar um comando requer **ler** o esquema.
- Executar comandos DDL (e alguns DCL) requer **escrever** o esquema.

# Gerador de Plano de Acesso Físico

## Otimizador Físico



# Controle de Concorrência

- O controle de concorrência é executado pela obtenção de “locks”
  - de escrita:  $W(x) \Rightarrow lock(w, X)$ , ou
  - de leitura:  $R(x) \Rightarrow lock(r, X)$ .
- O *lock* é gerado automaticamente quando se solicita uma operação de escrita (update, delete, insert) ou leitura (select).

```
UPDATE Customer  
SET LastPurchase = 100  
WHERE CId = 12321;
```



$W((t_v : Customer.LastPurchase, t_{id} : CId = 12321) = 100)$



$W(x = 100) \Rightarrow Lock(w, x)$

# Granularidade do travamento

- Controle de concorrência: travamento em duas fases.
  - Uma transação deve ter um *lock* antes de poder acessar um dado;
  - Uma transação não pode solicitar *locks* depois de liberar qualquer *lock*.
- Cada operação de *lock* “reserva” parte dos dados, dependendo de sua Granularidade:
  - Travar cada tupla: **inexequível!**
  - Travar a base de dados: **inviabiliza concorrência.**
  - Travar a relação: **concorrência muitíssimo baixa.**
  - Travar o atributo: **concorrência muito baixa.**
  - Travamento por predicado: **maximiza concorrência a um custo aceitável.**

# Travamento por predicado

$T_1$

$r_1$  R R.b=10

$T_2$

```
SELECT R.a
FROM R
WHERE R.b=10;
```

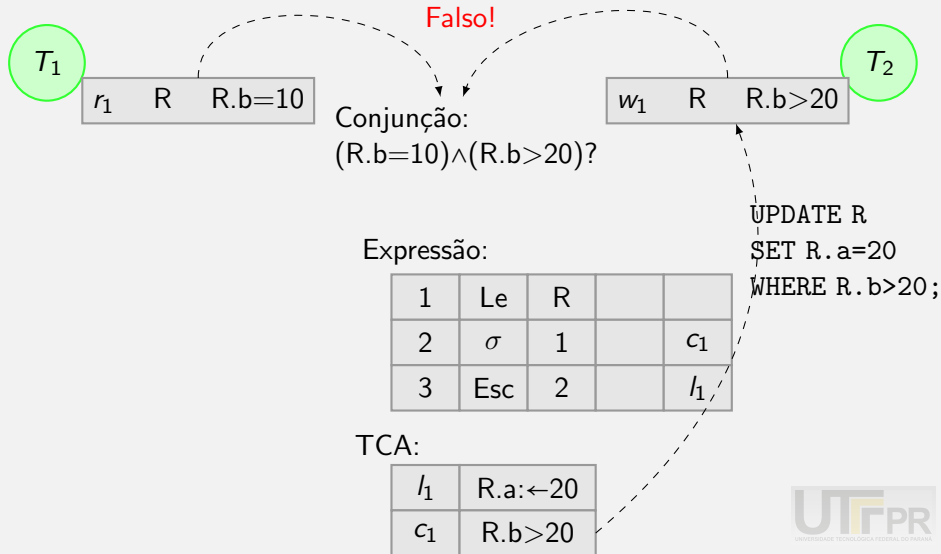
Expressão:

1	Le	R		
2	$\sigma$	1		$c_1$
3	$\pi$	2		$l_1$

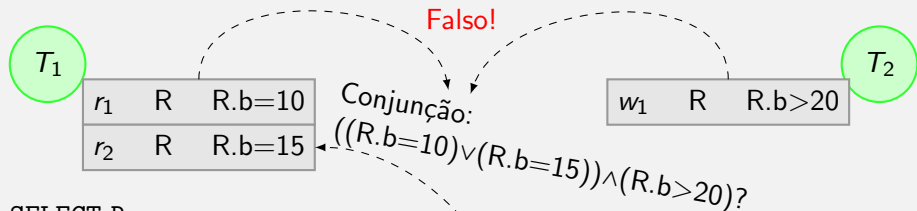
TCA:

$l_1$	R.a
$c_1$	R.b=10

# Travamento por predicado



# Travamento por predicado



SELECT R.c  
 FROM R  
 WHERE R.b=15;

Expressão:

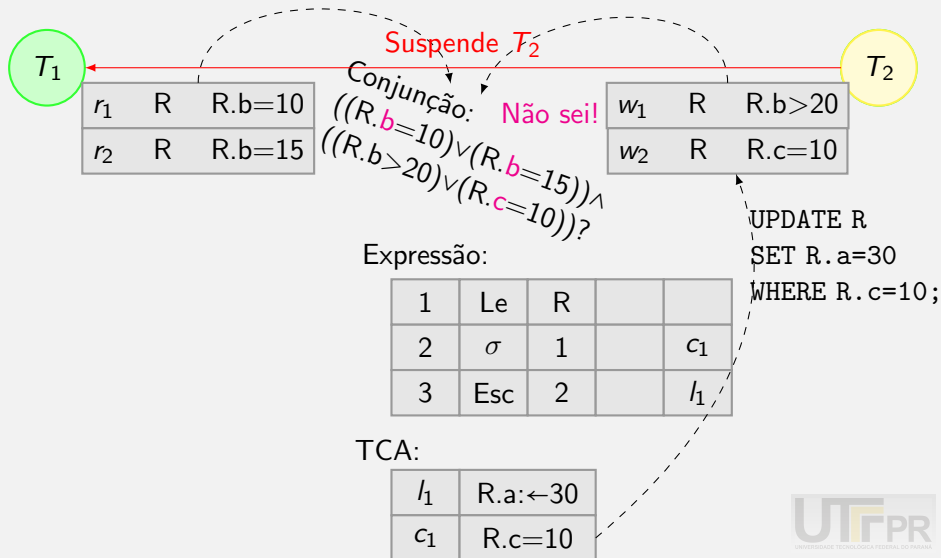
1	Le	R		
2	$\sigma$	1		$c_1$
3	$\pi$	2		$l_1$

TCA:

$l_1$	R.a
$c_1$	R.b=15



# Travamento por predicado

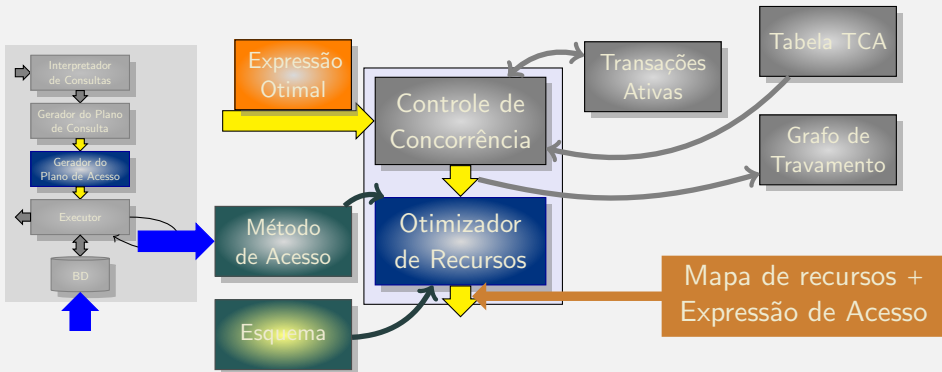


# Controle de Concorrência

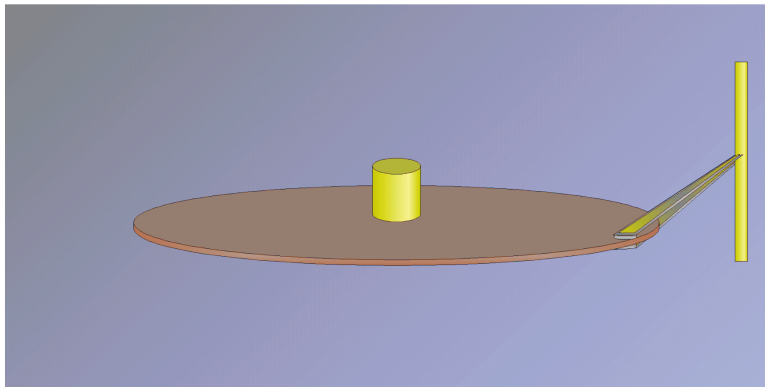
- Controle total implica o uso do Travamento em duas fases:
  - Fase 1: obtém os travamentos necessários (não libera nada);
  - Fase 2: libera todos os travamentos (não solicita mais nenhum).
- O travamento pode ser para:
  - Leitura: permite leitura compartilhada;
  - Escrita: permite leitura e escrita compartilhada.
- Cria o **Dígrafo de Dependência**
- Se houver bloqueio cíclico (*deadlock*), escolhe arestas para eliminar os ciclos e aborta os processos correspondentes aos nós onde cada aresta se origina.
- Se não há bloqueio, a expressão é
  - escrita no *log*,
  - e liberada para execução.
- Mas antes alocam-se os recursos necessários...

# Gerador de Plano de Acesso

## Otimizador Físico



# Arquitetura física das unidades de disco

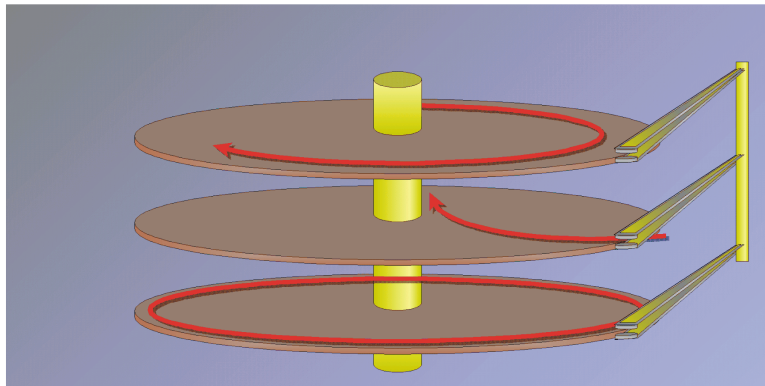


# Arquitetura física das unidades de disco

Velocidade angular (RPM)

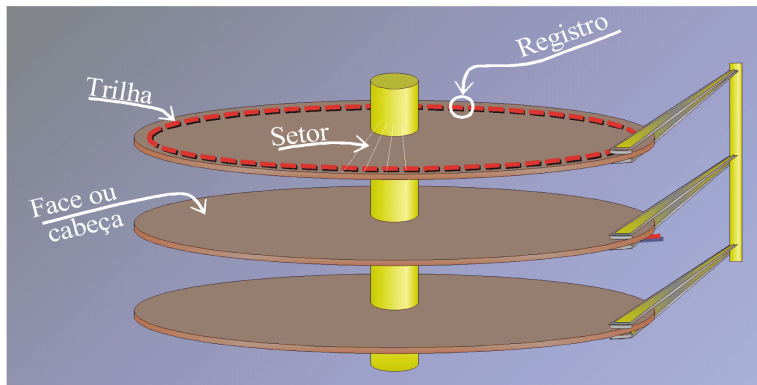
Tempo médio para posicionar as cabeças (ms)

Taxa de transferência (MBytes/s)



# Arquitetura física das unidades de disco

Registro = 512 Bytes (em geral)





# Disposição física dos dados em disco

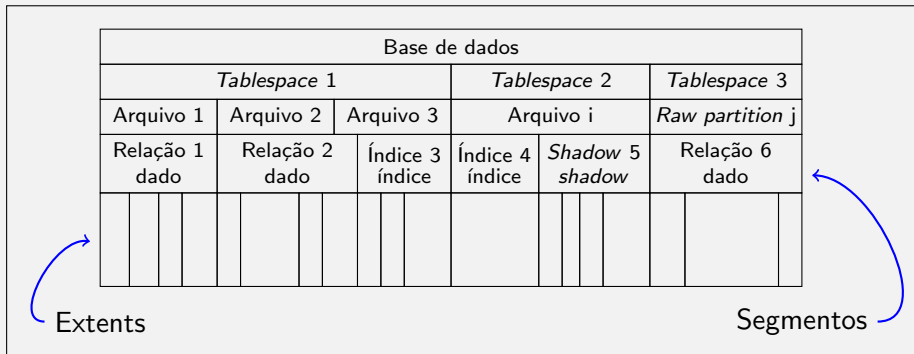
- Cada relação é guardada como uma seqüência de tuplas, de tamanho fixo ou aproximadamente fixo (média).
- Podem haver espaços em branco.
- Agrupam-se as tuplas em páginas de tamanho fixo,
  - múltiplo do tamanho dos registros em disco.
- Uma página é sempre lida inteira (mas podem ser vários registros).
- Existem páginas de dados, páginas de índices, páginas de sombra, etc.
  - Uma tabela pode ter tantos índices quanto necessários;
  - Páginas de dados são básicas;
  - Páginas de índices são secundárias;
  - Páginas de sombra (*shadow*) são mantidas por transação;
  - Páginas temporárias duram no máximo até o final do *thread* corrente.



# Arquitetura dos 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,
  - sobra de uma relação por transação, etc.
- Segmento
  - Composto por um *extent*
    - Sempre tem um *initial extent*.

# Arquitetura dos dados em disco



# Arquitetura dos 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 !?)

# Tempos de acesso a disco

- Tempo para acessar o primeiro registro
  - Posicionar as cabeças (*seek time*): 6 *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:  $\approx 9$  *ms*.
- Tempo para acessar o próximo registro na mesma operação:
  - Tempo de transferência (4.000 KBytes, 100 MBytes/s): 0,05 *ms*
  - Tempo do *gap*  $\ll 0,01$  *ms*
  - Portanto:  $\approx 0,05$  *ms* (se estiver no mesmo cilindro)
- Tempo para acessar o próximo registro em outra operação:
  - 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:  $\approx 3$  *ms*
- Para acessar 20 KBytes em 5 *extents* de 4 KBytes:  
$$9,05 + 4 * (3,05) = 21,25 \text{ ms}$$
- Para acessar 20 KBytes em 1 *extents* de 20 KBytes: 9,25 *ms*

# Métodos de acesso

- Existem operadores de acesso (operadores físicos) para todos os operadores da álgebra relacional, bem como diversas das combinações de uso mais freqüente,
- E diversos métodos de acesso
  - Acesso sequencial (*SequentialScan*)
  - *Index-Sequential Access Method* (ISAM ou  $B^+$ -tree)
  - *Hash*
  - Métodos de acesso multidimensionais (*R-tree*)

# Métodos de acesso

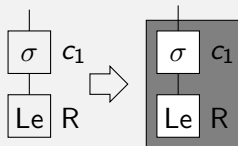
- Leitura simples
  - Leitura de dados do *extent*
- Leitura com seleção do *extent*
  - Sequencial  $\Rightarrow |extent_i R| + |extent - 1 R|$
  - ISAM  $\Rightarrow H_R * |extent idxR| + 1 extent R$  (para valores em uma tupla)
  - Hash  $\Rightarrow |extent idxR| + 1 extent R$  (para valores em uma tupla)
- Leitura com seleção do cache
  - Sequencial  $\Rightarrow |extent R|$
- Leitura com seleção seguida de junção
  - *sequentialscan* x *sequentialscan*  $\Rightarrow |extent R| * |extent S|$
  - ISAM x *sequentialscan*  $\Rightarrow H_R + 1 + |extent S|$  (para um valor)
  - ISAM x ISAM  $\Rightarrow H_R + 1 + H_S + 1$  (para um valor)
  - *sequentialscan* x Hash  $\Rightarrow |extent idxR| + |extent R|$  (para valores em uma tupla)
  - ISAM x Hash  $\Rightarrow H_R + 1 + 1$  (para um valor)

# Métodos de acesso

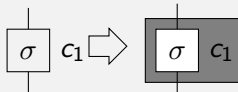
- Leitura simples:



- Leitura com seleção do *extent*:

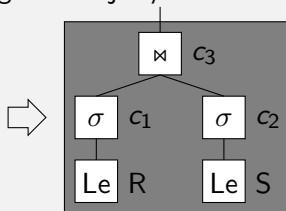


- Leitura com seleção do cache:

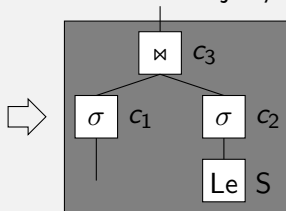


## Métodos de acesso (cont.)

- Leitura com seleção seguida de junção:



- Seleção no cache seguida de leitura com junção:



- Seleção no cache seguida de junção no cache.

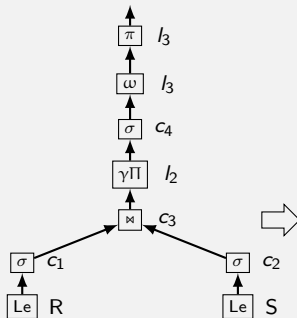


# Otimizador de recursos

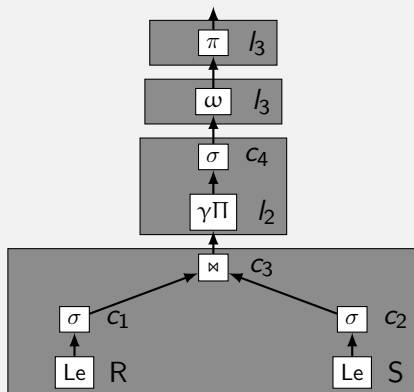
```

SELECT R.a, S.b
FROM R, S
WHERE R.a = 10
AND S.b > S.c
AND R.d = S.e
GROUP BY R.a, S.b, R.x
HAVING count(R.y) > 5
ORDER BY count(R.y)

```



Nove operadores algébricos, mas apenas quatro métodos de acesso.



# Otimizador de recursos

- Recurso: espaço no Cache
  - Cada operador de acesso requer espaço no cache, para duas ou três atividades:
    - Ramo esquerdo
    - Ramo direito
    - Resultado
- O otimizador deve levar em conta a disponibilidade de cache para a transação e alocar o espaço de acordo com a necessidade de acesso a disco de cada operador de acesso.
- O cache de um gerenciador não é alocado sob demanda como nos S.O., mas tem os espaços necessários reservados pelo otimizador.

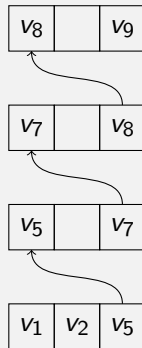
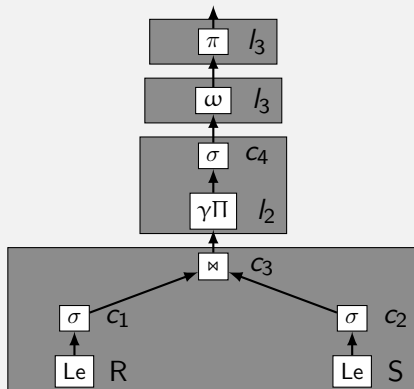
# Otimizador de recursos

```
SELECT R.a, S.b
FROM R, S
WHERE R.a = 10
AND S.b > S.c
AND R.d = S.e
GROUP BY R.a, S.b, R.x
HAVING count(R.y) > 5
ORDER BY count(R.y)
```

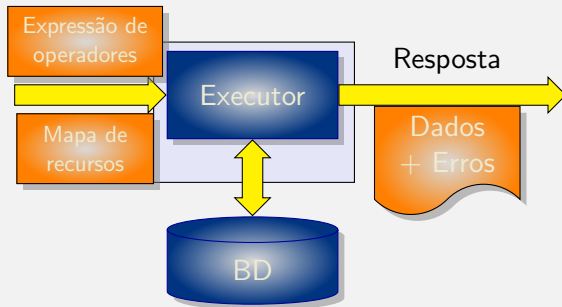
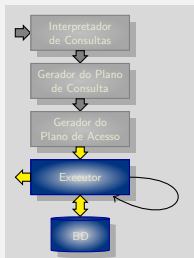
1	Le	R			$v_1$
2	Le	S			$v_2$
3	$\sigma$	1		$c_1$	$v_3$
4	$\sigma$	2		$c_2$	$v_4$
5	$\bowtie$	3	4	$c_3$	$v_5$
6	$\gamma\Pi$	5		$l_2..$	$v_6$
7	$\sigma$	6		$c_4$	$v_7$
8	$\omega$	7		$l_3$	$v_8$
9	$\pi$	8		$l_1$	$v_9$

Seja  $M$  o total de registros do cache para a transação. Cada operador  $i$  terá  $n_i$  registros no cache:

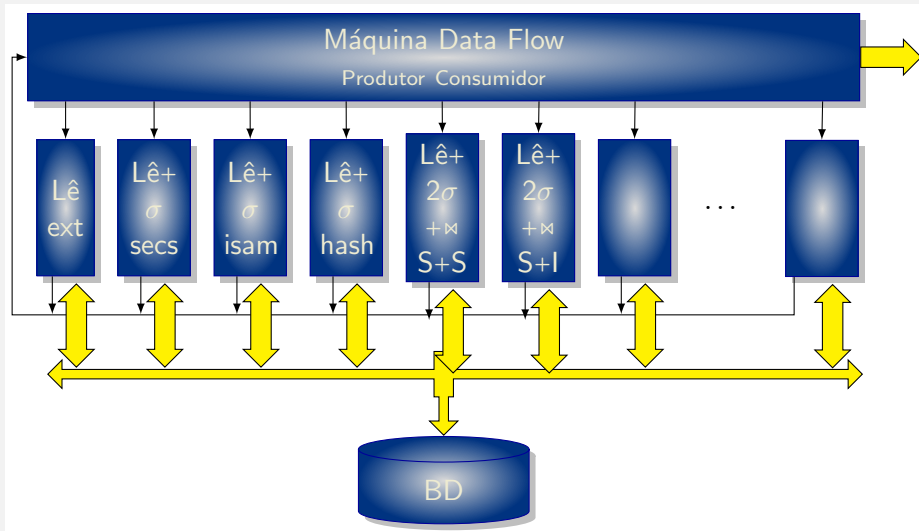
$$n_i = \frac{M \cdot v_i}{(v_1 + v_2 + v_5 + v_7 + v_8 + v_9)}$$



# Executor



# Executor



# Otimizações que o gerenciador faz

- Otimizador de consultas
  - Escolha das operações algébricas;
  - Escolha da ordem das operações.
- Ambas as escolhas dependem das métricas de
  - previsão de seletividade,
  - previsão de custo de acesso a disco.
- Otimizador de planos de acesso
  - Escolha do método de acesso. Depende da previsão de seletividade.
  - Escolha da execução em *Multi-thread*.
- Onde o DBA pode atuar:
  - Manutenção das métricas
  - Criação de chaves para os métodos de acesso
  - Alocação de memória cache e *extents*
  - Manutenção de regras para dependências funcionais e *triggers*.

# Manutenção das métricas

- Quando as métricas são calculadas/atualizadas?
  - Nas operações de atualização
  - Na execução de consultas
  - Em operações específicas
    - Automaticamente
    - Comando específico

# Criação de chaves para os métodos de acesso

- Regra geral:

As relações de uma base de dados podem ser divididas em estáticas e dinâmicas.

- As **estáticas** sofrem muito pouca atualização e, portanto, devem ter todas suas dependências funcionais, *triggers*, etc. completamente definidas e continuamente ligadas;
- As **dinâmicas** sofrem muitas atualizações. Nessas tabelas é aconselhável dividir as operações em dois estágios:
  - **Base acordada**: ela está sob solicitação de consultas externas. Nesse estágio todas as dependências funcionais, *triggers*, etc. devem estar ligadas, mas as atualizações devem ser dirigidas (sempre que possível) para tabelas de trabalho, mantidas idealmente pequenas, com acesso apenas de inclusão.
  - **Base dormindo**: ela não recebe consultas externas. Nesse estágio ela posterga todas as dependências funcionais, *triggers*, etc. para avaliação no final da transação e executa uma transação para processar cada tabela de trabalho, de maneira o menos concorrente possível.



## Criação de chaves para os métodos de acesso

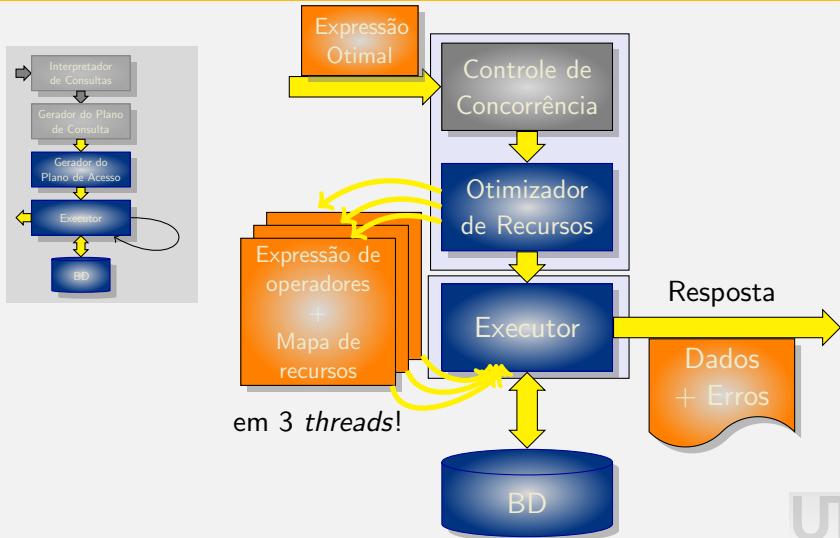
- Escolha dos métodos de acesso que melhor se adaptam para os tipos de consulta mais frequentes em que cada relação possa participar.
  - Junções e chave estrangeira - ISAM
  - Tabelas de definição de atributos discretos (pequenas) - *HASH*
- Tabelas pouco atualizadas devem ter todas as chaves e atributos de junção sempre ligados em ISAM.
- Considerar se é possível não usar índices e chaves estrangeiras em tabelas com atualização frequente.
  - É possível garantir a consistência por projeto da base ou no aplicativo?
  - Cuidados com a normalização
    - Excessiva: diminui o desempenho;
    - Insuficiente: leva a inconsistência dos dados.

# Alocação de memória cache e *extents*

- A alocação de memória cache, áreas de trabalho e *extents* devem ser feitas prevendo a carga típica de trabalho e concorrência.
  - Alocar as áreas de trabalho, *logs* e *extents* em unidades de disco/controladores separados, prevendo a possibilidade de acesso concorrente do *hardware*.

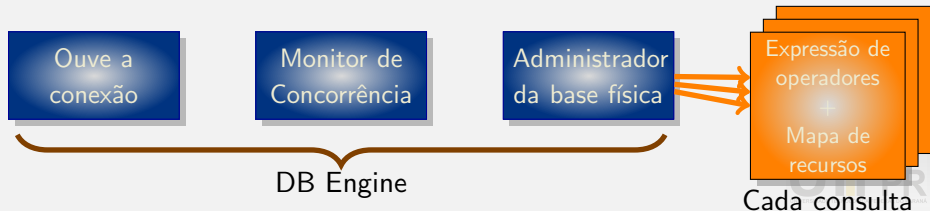
# Execução em *multi-thread*

Manutenção de regras para dependências funcionais e *triggers*.



## Execução em *multi-thread*

- O Gerenciador é constituído por 3 threads que ficam em execução permanente:
  - Principal (atende os clientes);
  - Monitor de concorrência;
  - Administrador da base física.
- Cada comando SQL pode gerar até 3 *threads* para execução em:
  - Resposta imediata;
  - Validação de dependências de chave estrangeira e *triggers* de pós-execução;
  - Final da transação.



# Roteiro

- 1 Conceitos gerais
- 2 O Interpretador de Consultas
- 3 O Otimizador Lógico
- 4 O Otimizador do Plano de Acesso Físico
- 5 O Executor
- 6 O que e como deve/pode ser automatizado

# Arquitetura Geral de um Sistema de Gerenciamento de Bases de Dados Relacional

**Prof. Dr. Ives Renê V. Pola**

[ivesr@utfpr.edu.br](mailto:ivesr@utfpr.edu.br)

Departamento Acadêmico de Informática – DAINF  
UTFPR – Pato Branco DAINF  
UTFPR  
Pato Branco - PR

FIM

