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

Caetano Traina Jr. — Versão Beamer: Mônica Ribeiro Porto Ferreira

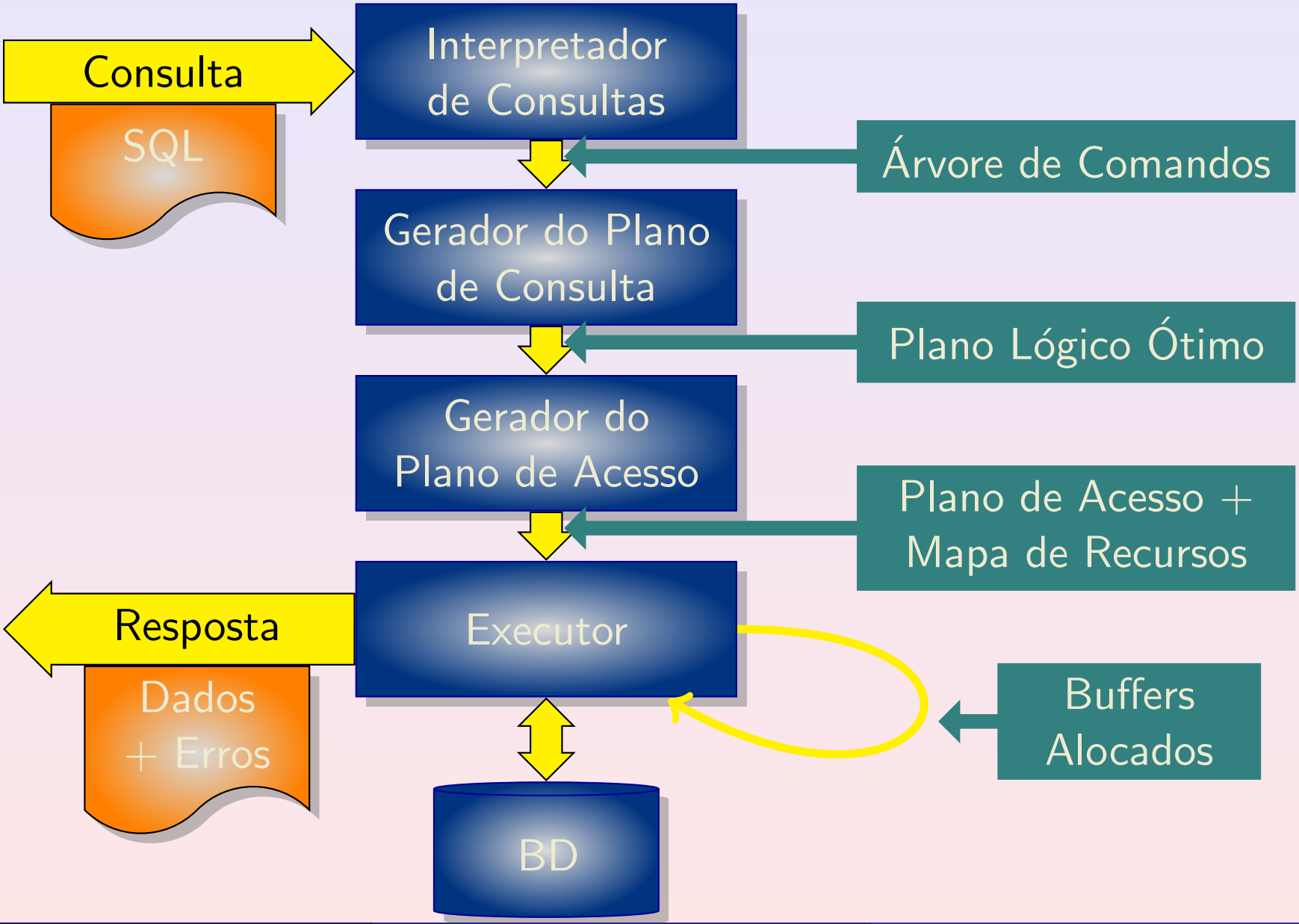
Grupo de Bases de Dados e Imagens
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - São Carlos
caetano@icmc.usp.br

7 de março de 2013
São Carlos, SP - Brasil

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

Outline

Arquitetura de um SGBDR

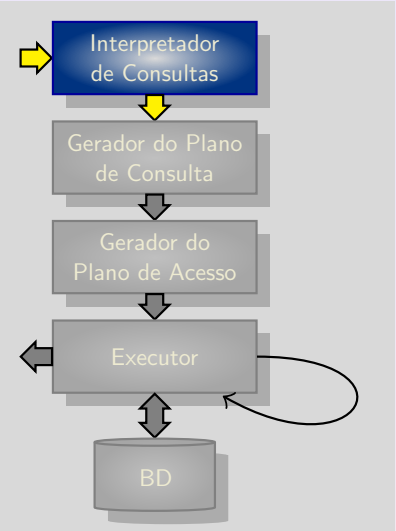


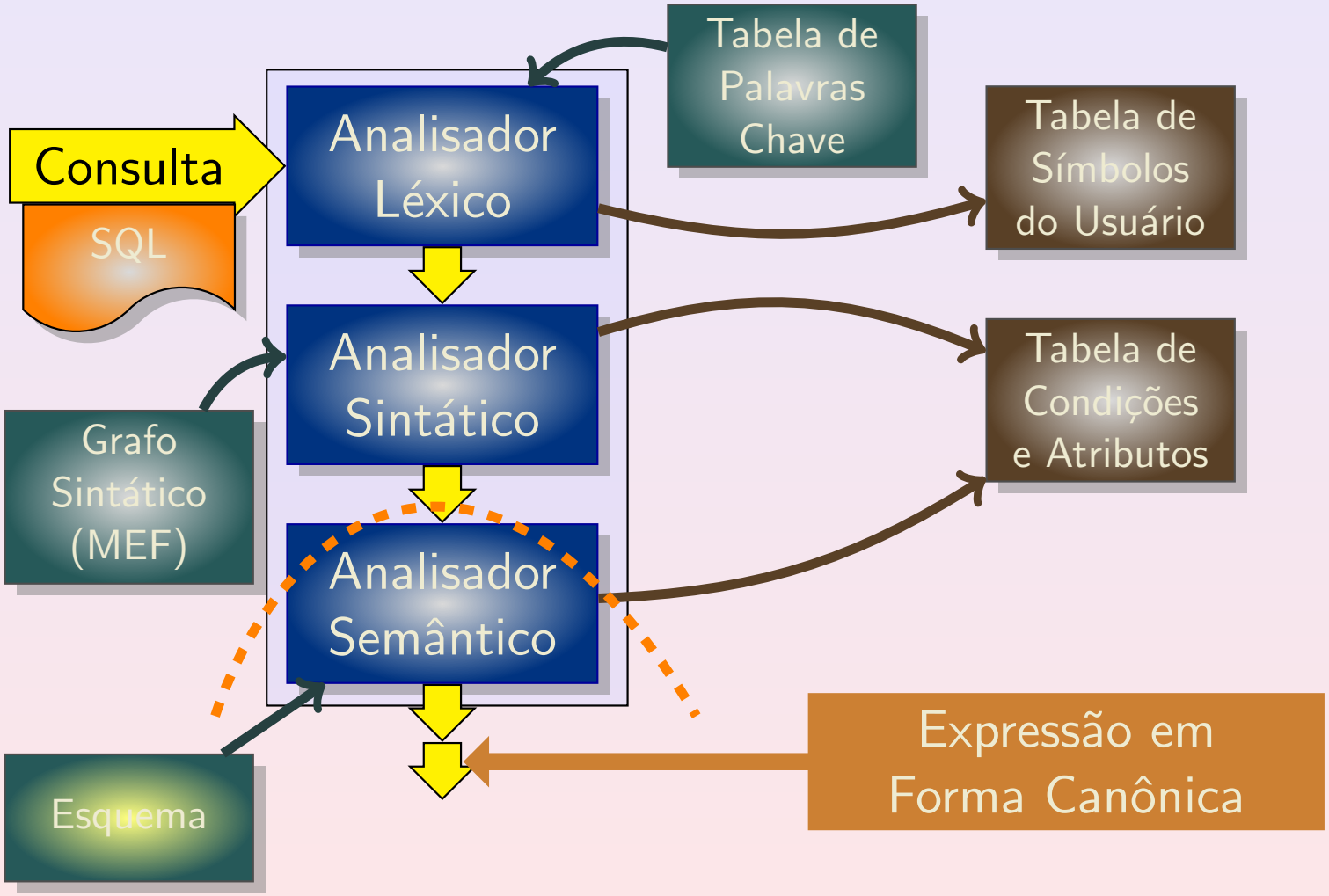
Classes de comandos ‘que devem ser executados

- Comandos da DCL
  - CREATE DATABASE, SET TRANSACTION, etc.
    São apenas compilados, mas têm execução seqüencial
- 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 dai fazer com elas o que o comando indica: mostrar/atualizar/apagar.

Interpretador de Consultas

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





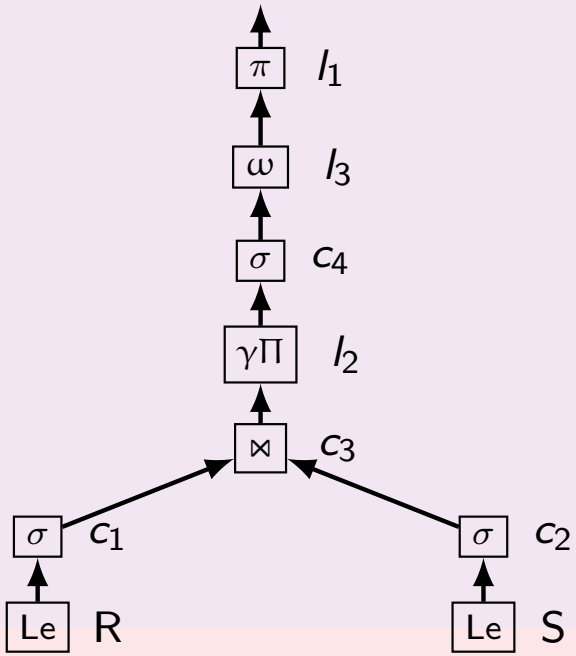
Interpretador de Consultas

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

- Ler todas as <relações>, gera $\boxed{\text{Le}_{(relacao)}}$
- Processar as $\texttt{<cond}_1\texttt{>}$
  - Se $\texttt{<r}_i\texttt{.atr}_1\ \theta\ \texttt{cte>}$, gera $\boxed{\sigma_{(cond)}}$
  - Se $\texttt{<r}_i\texttt{.atr}_1\ \theta\ \texttt{r}_j\texttt{.atr}_1\texttt{>}$ e $\texttt{i}=\texttt{j}$, gera $\boxed{\sigma_{(cond)}}$
  - Se $\texttt{<r}_i\texttt{.atr}_1\ \theta\ \texttt{r}_j\texttt{.atr}_1\texttt{>}$ e $\texttt{i}\neq\texttt{j}$, gera $\boxed{\texttt{r}_i \bowtie_{(cond)} \texttt{r}_j}$
- Se houver mais de uma tabela, gera $\boxed{\texttt{r}_i \times \texttt{r}_j}$
- Processar atrib. $\texttt{<list\_atr}_2\texttt{>}$ agregados, gera $\boxed{\gamma \rightarrow \prod_{\{<list\_atr1> \cup <list\_atr2> \cup <cond2.atr>\}}}$
- Processar as $\texttt{<cond}_2\texttt{>}$, gera $\boxed{\sigma_{(condicao)}}$
- Processar a $\texttt{<list\_atr}_3\texttt{>}$, gera $\boxed{\omega_{(<lista\_atr>)}}$
- Processar a $\texttt{<list\_atr}_1\texttt{>}$, gera $\boxed{\pi_{\{<lista\_atr>\}}}$

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 $\underline{Le_{(relacao)}}$
Processar as <$cond_1$>
Se <$r_i.atr_1$ θ cte>, gera $\underline{\sigma_{(condicao)}}$
Se <$r_i.atr_1$ θ $r_j.atr_1$> e i=j, gera $\underline{\sigma_{(condicao)}}$
Se <$r_i.atr_1$ θ $r_j.atr_1$> e i≠j, gera $\underline{r_i \bowtie_{(condicao)} r_j}$
Se houver mais de uma tabela, gera $\underline{r_i \times r_j}$
Processar os atributos <$list\_atr_2$> agregados, gera
$\underline{\gamma \rightarrow \Pi_{\{<listatr1> \cup <listatr2> \cup <cond2.atr>\}}}$
Processar as <$cond_2$>, gera $\underline{\sigma_{(condicao)}}$
Processar a <$list\_atr_3$>, gera $\underline{\omega_{(<lista\ atr>)}}$
Processar a <$list\_atr_1$>, gera $\underline{\pi_{\{<lista\ atr>\}}}$

TCA

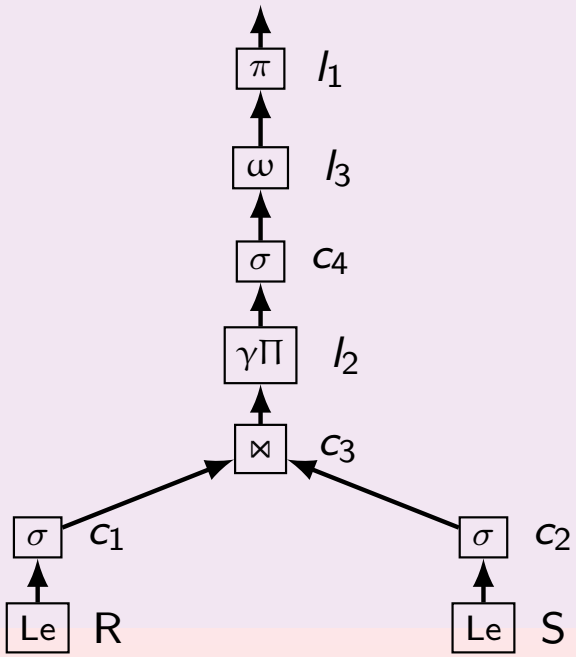
rótulo	condição
c_1	R.a = 10
c_2	S.b > S.c
c_3	R.d = S.e
l_2	R.a, S.b, R.x
c_4	count(R.y)>5
l_3	count(R.y)
l_1	R.a, S.b

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	σ	1		c_1
4	σ	2		c_2
5	$\bowtie$	3	4	c_3
6	$\gamma\Pi$	5		$l_2 \cup \ldots$
7	σ	6		c_4
8	ω	7		l_3
9	π	8		l_1

TCA

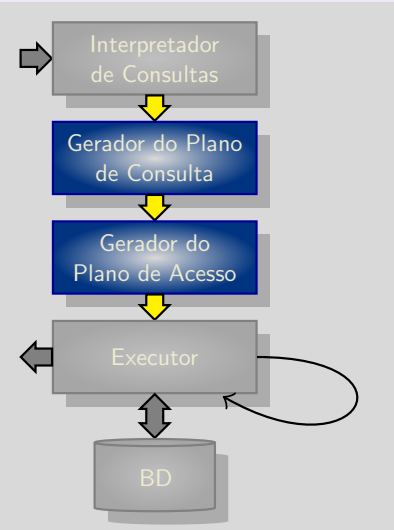
rótulo	condição
c_1	R.a = 10
c_2	S.b > S.c
c_3	R.d = S.e
l_2	R.a, S.b, R.x
c_4	count(R.y)>5
l_3	count(R.y)
l_1	R.a, S.b

TSU

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

Otimizadores

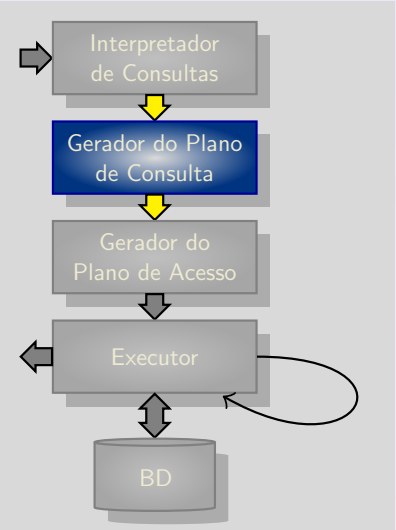
Otimizadores de Expressão e de Caminho de Acesso

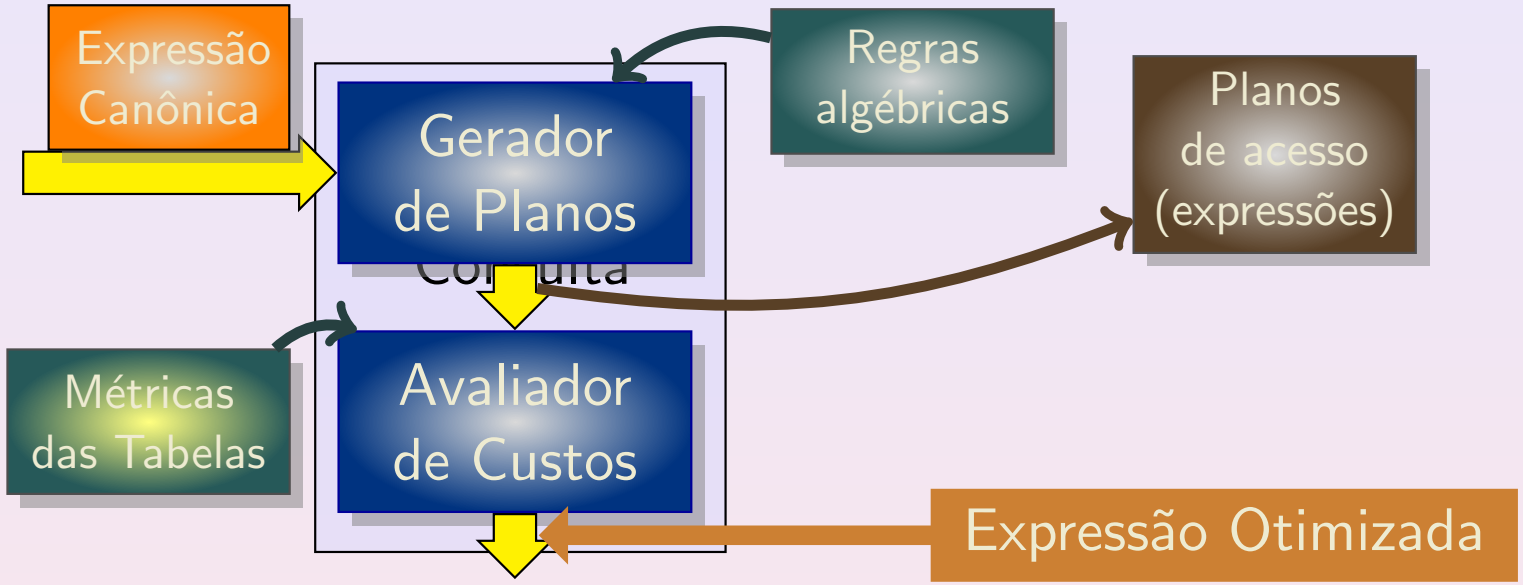


- Gerador do Plano de Consulta
  - ☞ Otimizador da expressão algébrica (Otimizador Lógico)
    - Gerador de Planos
    - Avaliador de Planos (lógicos)
- Gerador do Plano de Acesso
  - ☞ Otimizador do caminho de acesso (Otimizador Físico)
    - Controle de Concorrência
    - Otimizador de Recursos.

Gerador do Plano de Consulta

Otimizador Lógico



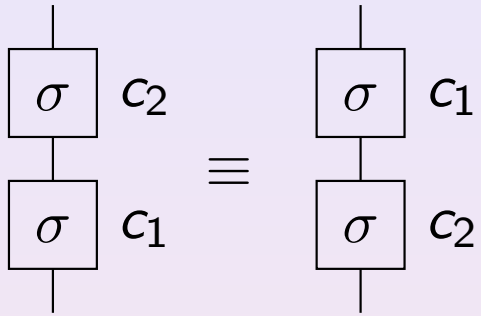


O Gerador de Planos de Consulta é baseado nas Regras Algébricas que existem para os operadores relacionais.

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 - σ

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

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

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

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

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

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

...

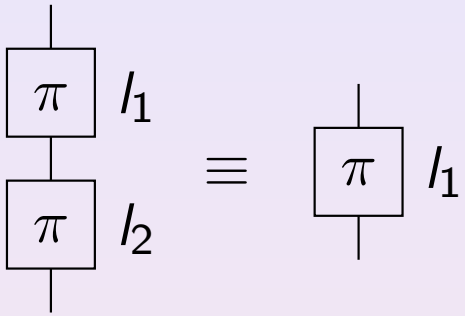
i	σ	k		c_1
j	σ	i		c_2
	...			

$\Leftrightarrow$

i	σ	j		c_1
j	σ	k		c_2
	...			

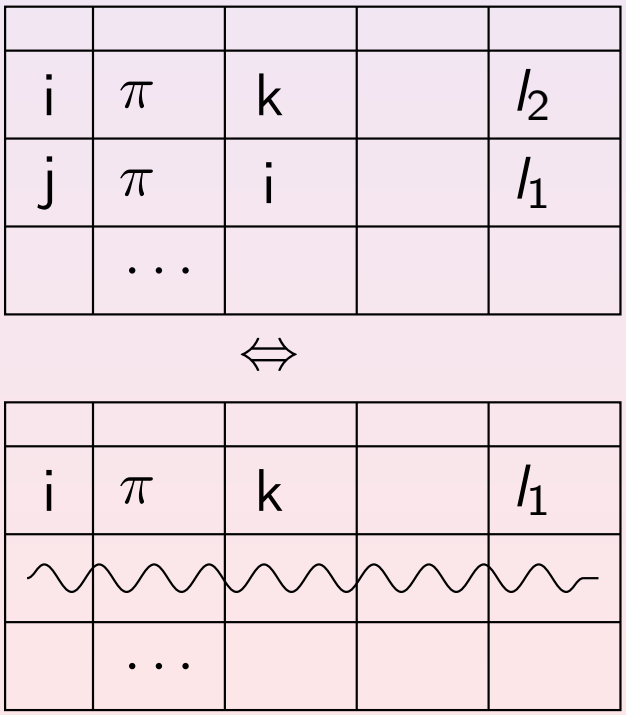
Regras da álgebra relacional

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



Operador PROJECT - π

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



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` - $\bowtie$

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

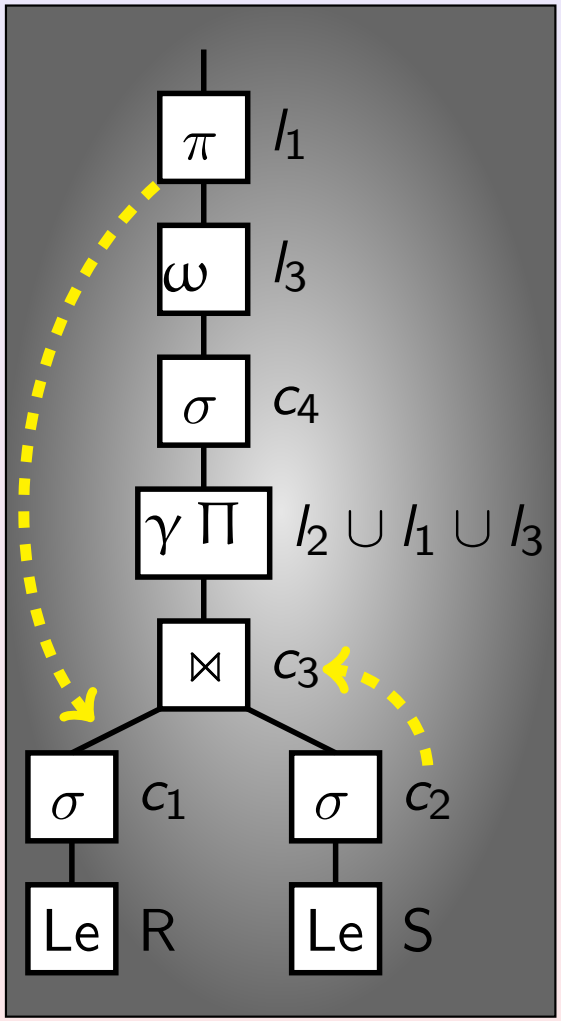
Se c_1 envolver apenas atributos de R_1 e de R_2, e c_2 envolver apenas atributos de R_2 e R_3, então:

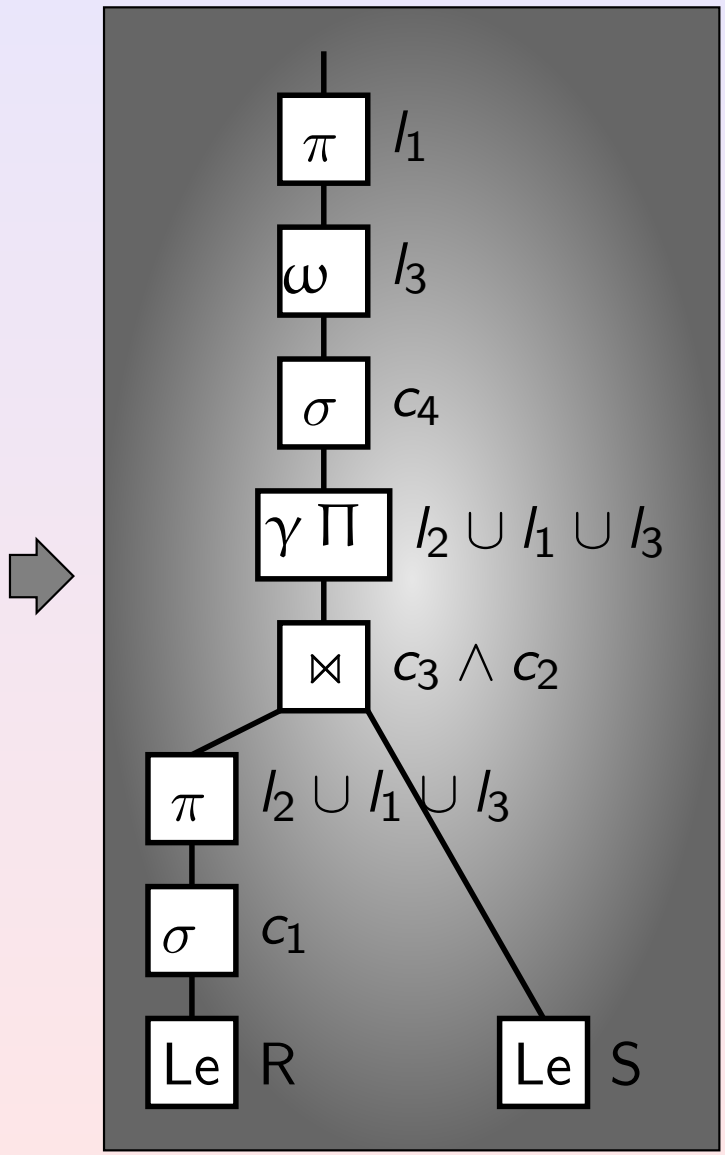
$$(R_1 \bowtie_{c_1} R_2) \bowtie_{c_2} R_3 \Leftrightarrow R_1 \bowtie_{c_1} (R_2 \bowtie_{c_2} R_3) \text{ (Associativa)}$$

. . . (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

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

Como prever a seletividade das condições?

Previsão de Seletividade

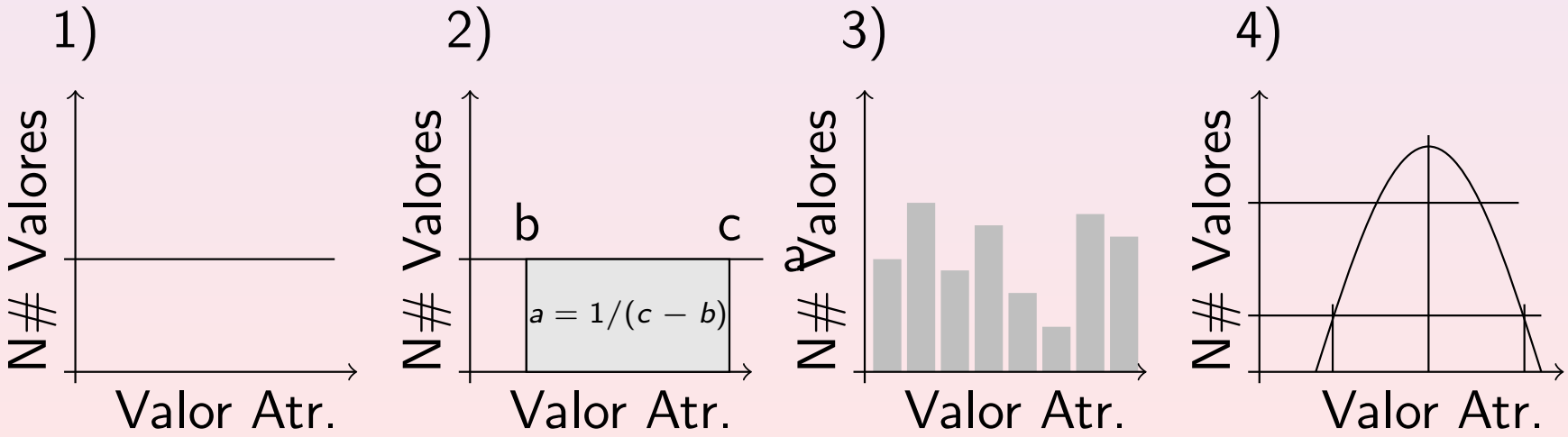
- A previsão de seletividade depende de métricas mantidas pelo Gerenciador e inclui informações sobre:
  - Medidas sobre a Base de Dados,
  - Medidas sobre as Tabelas,
  - Medidas sobre cada Atributo em cada tabela,
  - Medidas sobre cada Chave em cada tabela...
- As Métricas (estatísticas) são obtidas:
  - 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:
  1. Constante. Assume um valor constante para a seletividade;
  2. Distribuição Uniforme. Cada valor contribui $1/|Dominio\ ativo|$;
  3. Histograma. Deve ser mantido um contador para cada valor do domínio ativo (arquivo invertido);
  4. 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 diversão condições de seleção, e/ou subconsultas aninhadas (possivelmente envolvendo agregações e junções externas), etc...

```
SELECT *
 FROM R, S, T, U
 WHERE R.a = S.b
 AND S.c = T.d
 AND T.e = U.f
 AND T.f IN (Select /ldots)
    .  .  .
```

a melhor ordem de execução depende da seletividade das condições envolvidas nos diversos operadores, para minimizar:

- o Número de acessos a disco para leitura +
- o Número de acessos a disco para escrita

O Espaço de busca do plano ótimo

- A quantidade de planos de execução equivalentes pode ser muito grande!
- É necessário que a escolha seja feita rapidamente, então apenas alguns dos planos possíveis, aqueles que reconhecidamente tem maiores chances de serem ótimos, são de fato avaliados.
- O otimizador **estima** o custo de cada plano:
  - Procurando dentre os planos de um **Espaço de Busca**,
  - usando um **Modelo de custo**
  - e uma **estratégia de enumeração** para convergir rapidamente para o plano ótimo.

O Espaço de busca do plano ótimo

O Espaço de busca

- O **Espaço de Busca** de um otimizador corresponde ao conjunto de planos de execução que são avaliados na busca do plano ótimo.
- Ele é definido por uma **Heurística**, baseada na experiência do fabricante do SGBD, que inclui apenas planos com chances reais de ser ótimo para alguma consulta.
- Mesmo dentre os que têm chance, apenas alguns são considerados — não se pode gastar muito tempo avaliando opções em demasia!
- Por exemplo, não existe uma heurística para prever qual ordem de junção é a melhor para uma consulta que envolve múltiplas junções...
- e a quantidade de alternativas pode ser grande: para n junções, a quantidade de opções é maior do que $n!$ (se houver 6 junções, são mais de 720 alternativas).

O Espaço de busca do plano ótimo

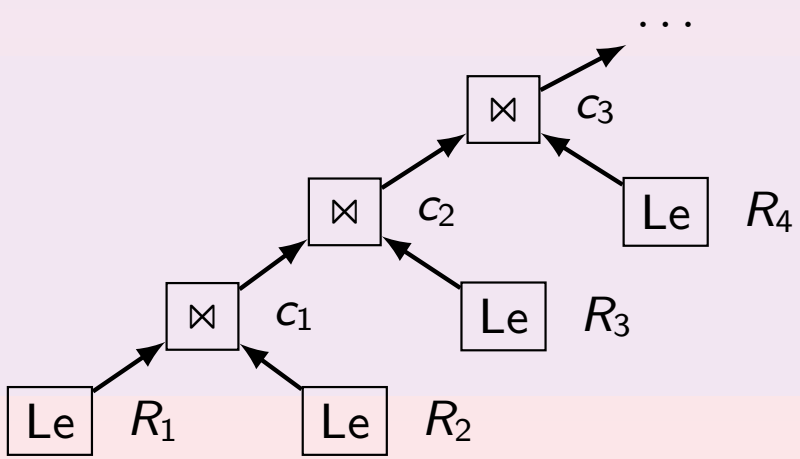
O Espaço de busca

- Em geral, o espaço de busca é definido quebrando as consultas em blocos e classificando as consultas e os blocos em tipos, e abordando o subespaço que tem chances de conter o plano ótimo para aquele bloco/consulta.
- As principais restrições que são usadas para diminuir o espaço de busca abordam:
  - Reordenação de Junções;
  - Reordenação de Junções e agrupamentos;
  - Reordenação de Junções internas e externas;
  - Reordenação de Junções e ordenações;

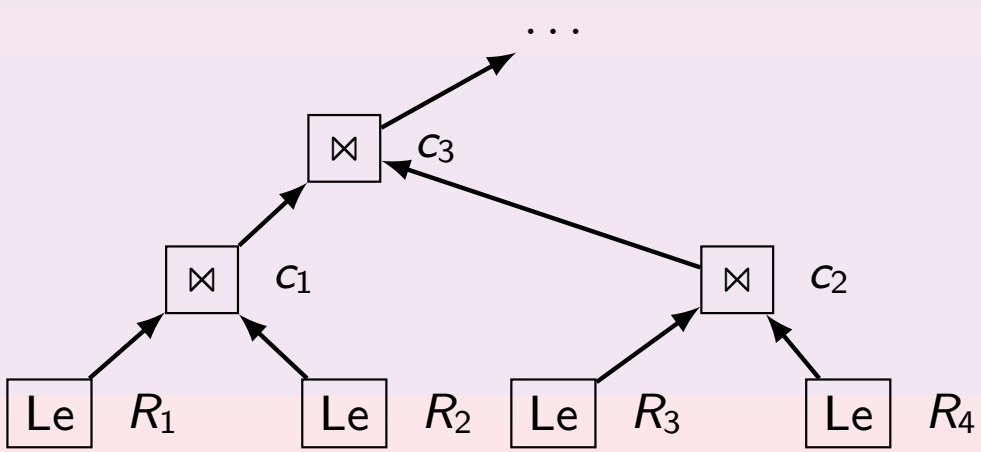
O Espaço de busca do plano ótimo

Reordenação de Junções

- Como o espaço de reordenação de junções é muito elevado, procuram-se apenas algumas delas.
- O mais comum é que o otimizador use apenas:
  - As sub-árvores de comando chamadas lineares: $(((R_1 \bowtie R_2) \bowtie R_3) \bowtie \ldots) \bowtie R_n)$
  - As sub-árvores de comando chamadas *bushy* (tentanto balancear a sub-árvore de junções):$((R_1 \bowtie R_2) \bowtie \ldots \bowtie (R_{n-1} \bowtie R_n))$



Linear



Bushy

O Espaço de busca do plano ótimo

Reordenação de Junções

- Uma consulta com n junções leva a $n!$ árvores lineares distintas mais outro tanto de árvores *Bushy*, mais as demais árvores...
- Somente as árvores lineares e *Bushy* são avaliadas pelos otimizadores,
- Nem todo otimizador considera árvores *bushy* no espaço de busca:
  - Elas podem levar a um plano de execução mais barato;
  - mas requerem materializar resultados intermediários;
  - e, principalmente, dobram o espaço de busca.

O Espaço de busca do plano ótimo

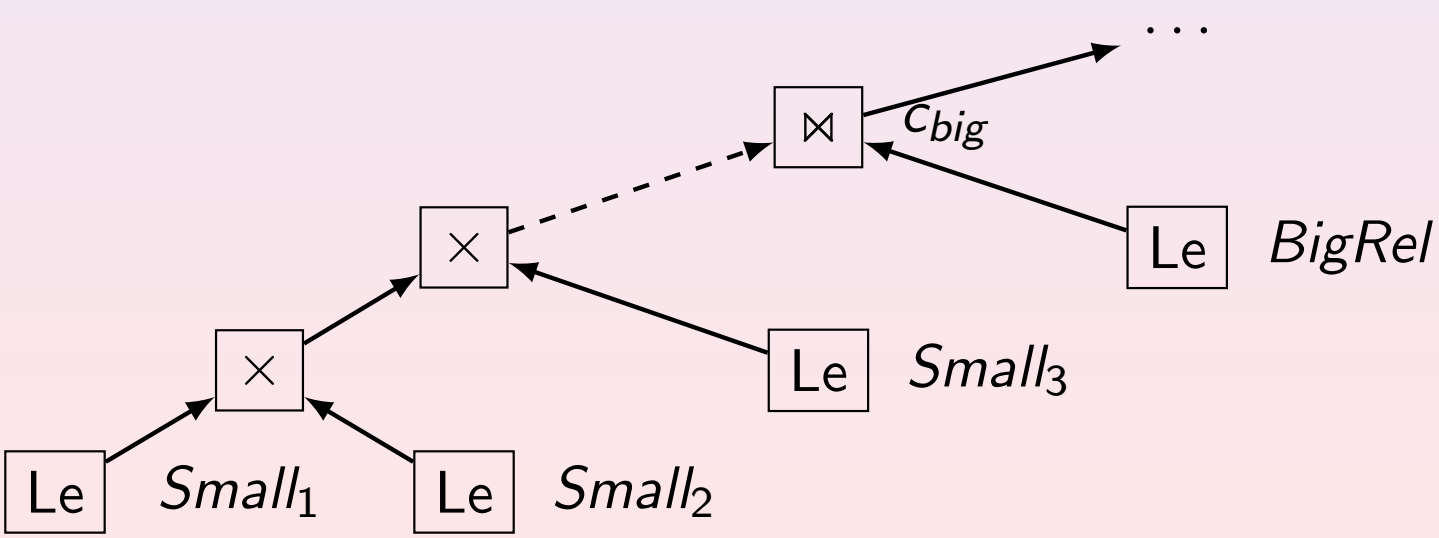
Reordenação de Junções

- Veja que produtos cartesianos comutam e são distributivos em relação à junção;
- Portanto podem ser colocados também entre os operadores de junção:
- Mas considerá-los junto com as junções aumenta bastante o espaço de busca: $n!$ árvores lineares, onde n é o total de junções internas mais o total de produtos cartesianos da consulta;
- Por outro lado, são poucas as consultas em que seja melhor executá-los depois de todas as junções;
- A árvore canônica sempre tem os operadores de produto cartesiano depois de todas as junções;
- Os otimizadores somente avaliam antecipar os cartesianos para alguns tipos específicos de consulta.

O Espaço de busca do plano ótimo

Reordenação de Junções

- Uma oportunidade de otimização por executar produtos cartesianos antes de algumas junções ocorre em consultas analíticas em que os produtos cartesianos incorporam ao plano de execução várias tabelas pequenas a uma grande tabela "fato";
- Executar os produtos cartesianos entre as tabelas pequenas antes pode levar à um bom ganho de desempenho.



O Espaço de busca do plano ótimo

Reordenação de Junções

- Esse é um exemplo de que dividir o espaço de busca em regiões que são percorridas de maneira diferente em consultas de tipos diferentes permite que se obtenha bom ganho de desempenho em algumas consultas daquele tipo, sem aumentar exageradamente o espaço de busca em geral.

O Espaço de busca do plano ótimo

Reordenação de Junções e Agrupamentos

Reordenação de Junções e Agrupamentos

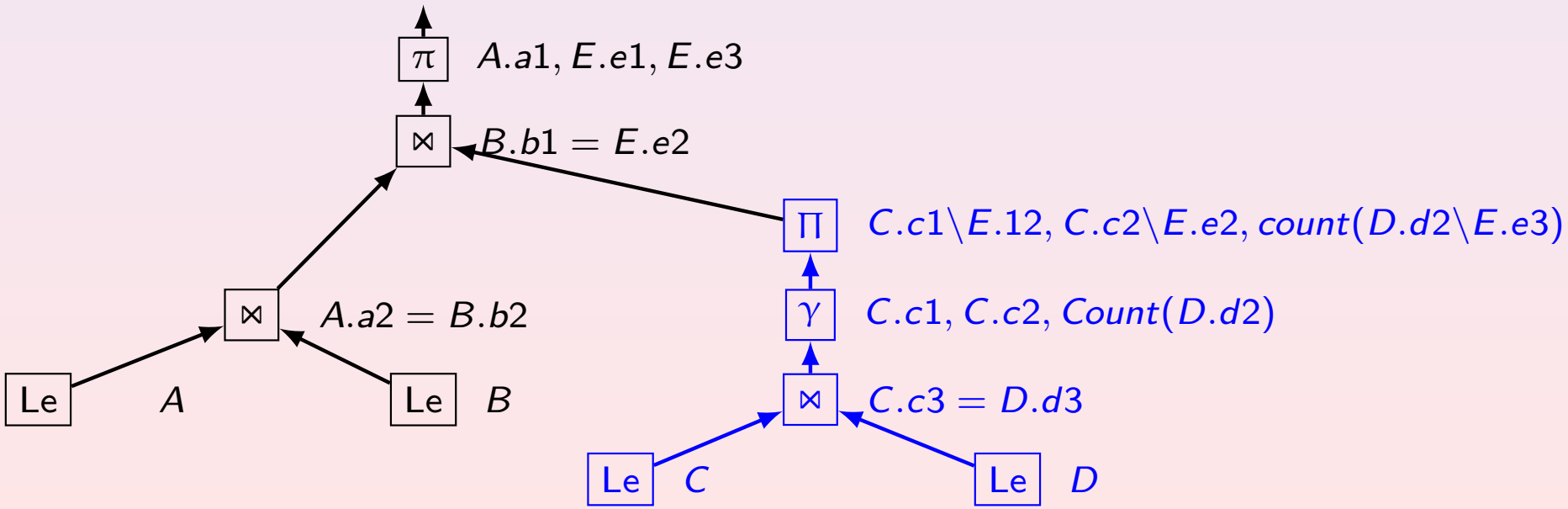
- O plano canônico sempre coloca a operação de agrupamento da cláusula `GROUP BY` depois de todas as junções e seleções da cláusula `WHERE`;
- No entanto, veja que quando a consulta inclui uma sub-consulta que tenha a cláusula `GROUP BY`, então a operação de agrupamento correspondendo à sub-consulta interna irá preceder a todas as junções e seleções da sub-consulta externa;
- O mesmo ocorre se pelo menos uma das relações da cláusula `FROM` for uma visão que inclui a cláusula `GROUP BY`.

O Espaço de busca do plano ótimo

Reordenação de Junções e Agrupamentos

Por exemplo, considere a seguinte consulta:

```
SELECT A.a1, E.e1, E.e3
    FROM A, B, (
        SELECT C.c1, C.c2, D.Count(D.d2) AS e1
            FROM C,D
            WHERE C.c3=d.d3
            GROUP BY C.c1, D.d2 ) AS E
    WHERE A.a2= B.b2 AND B.b1=E.e2
```



O Espaço de busca do plano ótimo

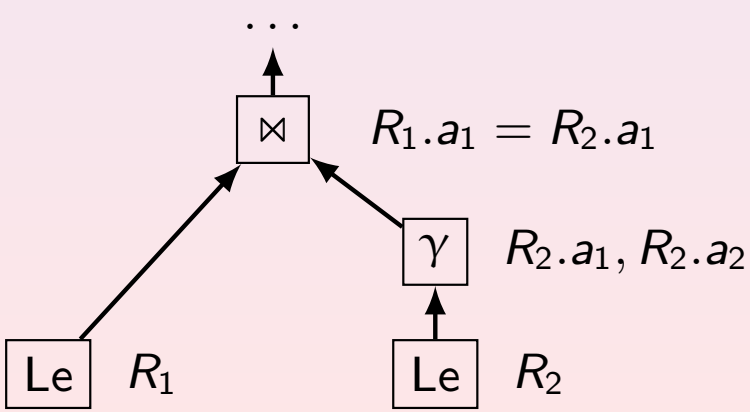
Reordenação de Junções e Agrupamentos

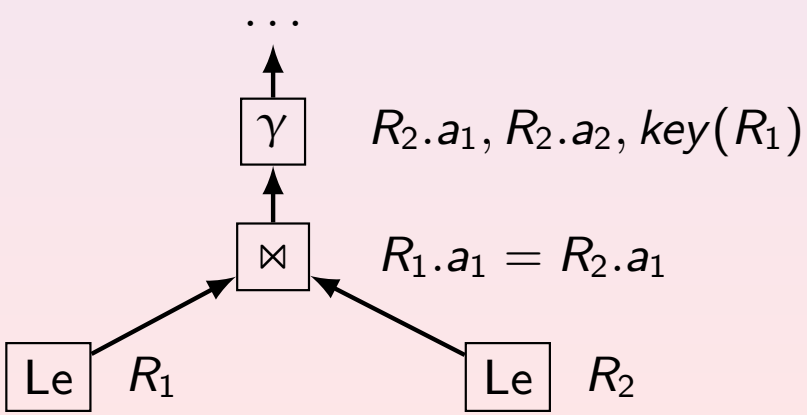
- A ordem de execução de uma junção e de um agrupamento pode levar a ganhos significativos de desempenho, dependendo das cardinalidades das relações de entrada e saída de cada operador:
  - Executar o agrupamento antes é eficiente quando o resultado consiste de poucos grupos agrupados, o que leva a existirem poucas tuplas para a junção operar;
  - Executar a junção antes é eficiente quando ela puder eliminar grande quantidade de tuplas das relações de entrada (por exemplo, se a junção é uma semijunção de alta seletividade).
- Portanto, a decisão de reordenar junções e agrupamentos pode levar a ganhos de desempenho em consultas que envolvam sub-consultas com `GROUP BY`.

O Espaço de busca do plano ótimo

Reordenação de Junções e Agrupamentos

- Os operadores algébricos de junção e agrupamento podem comutar, dependendo de restrições que se apliquem ou não aos seus parâmetros.
- Por exemplo, um agrupamento pode ser postergado a uma junção quando:
  - O predicado da junção não envolve o resultado de um operador de agregação do agrupamento; e
  - inclui-se a chave da outra relação na lista de atributos do agrupamento.

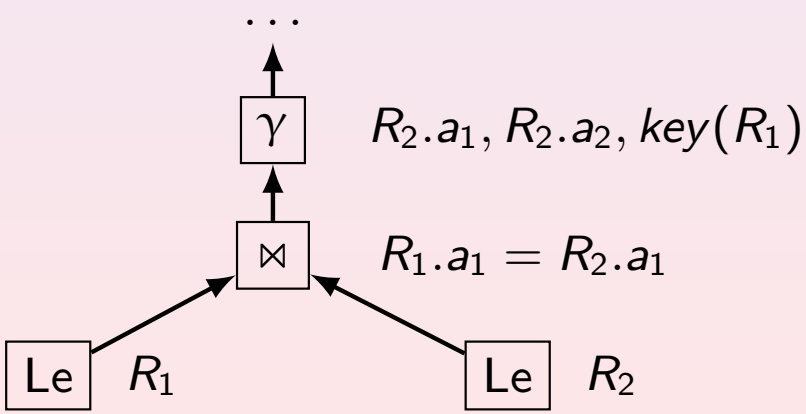


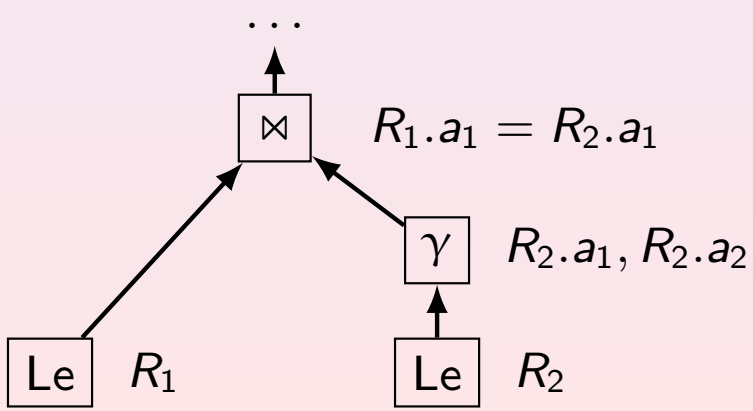


O Espaço de busca do plano ótimo

Reordenação de Junções e Agrupamentos

- Por outro lado, um agrupamento pode ser adiantado para um dos operandos de uma junção (digamos à direita, sobre a relação R_2) quando:
  - Os atributos do agrupamento incluem os atributos de R_2 necessários à junção;
  - Os atributos de agrupamento incluem uma chave de R_1; e
  - Os atributos de agregação são definidos apenas sobre atributos da relação R_2.





O Espaço de busca do plano ótimo

Reordenação de Junções e Agrupamentos

- As vezes é possível também "distribuir" um agrupamento sobre os dois operandos de uma junção, antecipando a junção também quando os atributos da agregação são definidos sobre atributos de ambas as relações de entrada;
- No entanto, para isso, é necessário que a função de agregação possua determinadas propriedades específicas, em particular alguma forma de distributividade.
- Por exemplo isso aconteçe com a soma:

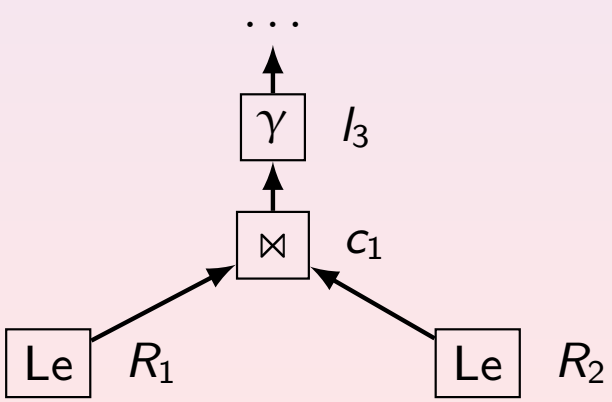
☞ $Sum(R_1 \bigcup R_2) \equiv Sum\,(Sum(R_1) \bigcup Sum(R_2))$
(note-se que $\bigcup$ aqui representa a operação de união de multiconjuntos tal como definido em SQL, não a união de conjuntos)

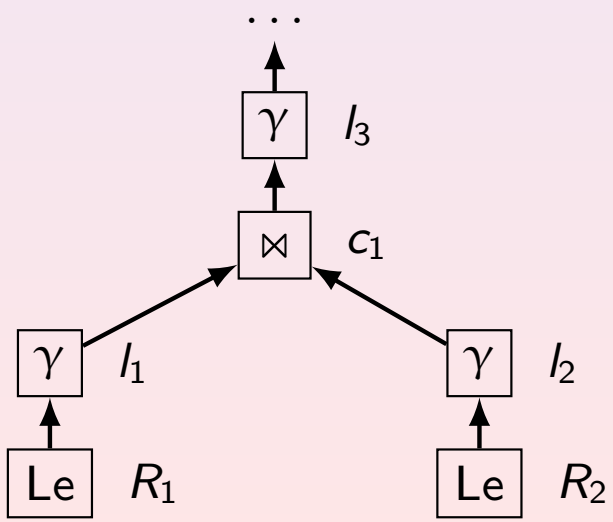
- Já a função de agregação média (*avg*) não possui essa propriedade.

O Espaço de busca do plano ótimo

Reordenação de Junções e Agrupamentos

- Note-se que uma propriedade assim "quebra" um agrupamento em 3 outros.
- No entanto, os agrupamentos que são antecipados à junção trabalham com relações menores, de maneira que o plano de consulta final pode ser mais eficiente do que realizar um agrupamento só (de novo, dependendo das cardinalidades das relações e seletividades das condições envolvidas).





$l_1, l_2 \subset l_3$

O Espaço de busca do plano ótimo

Reordenação de Junções internas e externas

- Junções externas não preservam a propriedade da comutatividade, nem entre junções externas, e nem combinando junções externas e internas;
- No entanto, as seguintes propriedades existem entre elas:

☞ $R_1 \bowtie (R_2 \mathbin{⟗} R_3) \equiv (R_1 \bowtie R_2) \mathbin{⟗} R_3$
$R_1 \bowtie (R_2 \mathbin{⟕} R_3) \equiv (R_1 \bowtie R_2) \mathbin{⟕} R_3$
$R_1 \bowtie (R_2 \mathbin{⟖} R_3) \equiv (R_1 \bowtie R_2) \mathbin{⟖} R_3$

- Assim, é possível postergar as junções externas para depois de todas as junções internas, o que permite otimizar pelo menos as junções externas,
- ou ao contrário, realizar primeiro todas as junções externas na base da árvore de comandos, e integrar os resultados parciais com junções internas que podem ser otimizadas no topo da árvore.

O Espaço de busca do plano ótimo

Reordenação de Junções e ordenações

- Um cuidado especial precisa ser tomado para preservar a ordem das tuplas quando se sabe que elas estão ordenadas;
- Os SGBDs atuais têm poucos recursos para tratar ordem. Somente se garante que a ordem é preservada na projeção final da cláusula `SELECT`, preservando a ordem resultante do último operador anterior a ela: o operador de ordenação ω quando há a clásula `ORDER BY`;
- Junções realizadas usando a técnica de *merge-join* gera um resultado ordenado pela chave de junção. Se operações posteriores puderem utilizar essa ordem, é importante preservar a ordem:
  - Operadores de projeção e seleção podem preservar a ordem de entrada;
  - Operadores de Junção, Agrupamento, Ordenação e remoção de tuplas repetidas podem aproveitar essa ordem;
  - Operadores de Junção, Agrupamento e Ordenação podem modificar a ordem se a chave de junção, agrupamento ou ordenação requer ordem distinta.
- Atualmente, praticamente não existe suporte para a preservação de ordem parcial.

O Espaço de busca do plano ótimo

Reordenação de Junções e ordenações

Portanto:

- Os SGBDR mais usados atualmente não tratam de relações que preservem ordem, nem de relações-base e nem de visões;
- O único lugar seguro onde o operador de ordenação (ω) pode ser usado é como o penúltimo operador da árvore de comandos, logo antes da projeção final;
- Em geral não se pode antecipar o operador de ordenação para antes de alguma junção;
- Alguns operadores físicos, como o "*merge-join*" podem se beneficiar de relações que estejam pré-ordenadas;
- Esse tipo de otimização é completamente dependente da heurística do otimizador de cada fabricante e dos operadores físicos implementados no SGBD.

O Espaço de busca do plano ótimo

Outras transformações de expressão

- **Uso de visões materializadas** - sub-consultas oriundas de visões materializadas ou mesmo que estejam atualizadas em razão de consultas anteriores recentes, podem ajudar a otimizar a consulta corrente;

☞ Esse tipo de transformação é muito complexo pois introduz o desafio de comparar pedaços da consulta corrente com outras expressões. Mas pode agilizar MUITO uma consulta. Os SGBDs estão começando a incluir algumas formas de otimização nesse sentido.

- **Sub-expressões repetidas** - muitas vezes uma consulta precisa repetir uma mesma sub-expressão mais de uma vez no mesmo comando de consulta.

☞ Quando essas sub-expressão ocorrem por restrições da sintaxe do SQL, pode ser que a mesma computação também se repita. Identificar e reaproveitar esses casos pode ser também complexo, mas casos especiais (frequentes) têm sido tratados pelos otimizadores.

O Espaço de busca do plano ótimo

- Como o otimizador não considera todos os planos de execução possíveis, é necessário ter uma estratégia de enumeração daqueles que são considerados.
- Existem basicamente três alternativas que estão disponíveis:
  - **Programação Dinâmica** (por exemplo: *System-R*);
    - ☞ Descobre o plano ótimo, mas pode ser muito demorada.
    - ☞ Hoje está disponível como opção do DB2.
  - **Representação Estrutural** (por exemplo: *Starburst*);
    - ☞ É essencialmente empírica, dependendo do traquejo do fabricante do SGBD.
    - ☞ É usada pela maioria dos SGBDR atuais.
  - **Transformação de Sub-comandos** (por exemplo: *Cascade*);
    - ☞ Permite tratar diferentes porções da consulta usando um meio termo entre a programação dinâmica e a Representação estrutural.
    - ☞ É usada por alguns poucos SGBDR, incluindo o *MS SQL Server*.

O Espaço de busca do plano ótimo

Otimização por Transformação de Sub-comandos

- Nem todas as consultas se beneficiam da exploração de todas as regras possíveis para a transformação de planos de execução.
- Uma consulta típica, tem " Sub-comandos" da árvore de busca em que determinadas propriedades são proveitosas, mas estender todas as regras para toda a árvore não necessariamente traz benefícios, mas com certeza aumenta o espaço de busca;
- Então a alternativa é classificar as consultas em Tipos, e tratar cada tipo como composto por sub-comandos, sendo que em cada sub-comando se usa apenas um sub-conjunto das regras de transformação existentes, portanto reduzindo o espaço de busca a algumas regiões do universo de busca possível.

O Espaço de busca do plano ótimo

Tipos de consultas

- Os Tipos de consultas em geral tratados são:
  - Consultas **S***elect*-**P***roject*-**J***oin* — SPJ;
  - Consultas com sub-consultas (usando visões, ou com operadores do cálculo de relações (incluindo os operadores {`IN`, `EXISTS`, *thetat*-`ANY`, *thetat*-`ALL`}), ou com sub-consulta definindo relações na cláusula `FROM`);
  - Consultas com junções externas;
  - Consultas com agregação sem agrupamento;
  - Consultas com agrupamento (e possivelmente com agregações);
  - Consultas com operadores de conjunto (`UNION`, `INTERSECT` e `EXCEPT`).

O Espaço de busca do plano ótimo

Tipos de consultas

- As consultas SPJ são divididas em três sub-tipos, dependendo dos sub-comandos que ela possui:
  - *Star query* – Dois ou mais operadores de junção com um atributo de ligação em comum, junto com os respectivos operadores de seleção e/ou projeção;
  - *Select query* – Cinco ou mais operadores de seleção;
  - *Default query* – quando a consulta não é nem *star* nem *select*.
- Note-se que "sub-comando" significa tanto a sub-árvore de comandos que compõe a consulta, quanto o conjunto de operadores algébricos que devem ter tratamento específico para compor o espaço de busca de planos de execução avaliado para otimizar essa sub-árvore.

O Espaço de busca do plano ótimo

Tipos de consultas

- As transformações algébricas que são aplicadas a cada sub-árvore depende dos operadores algébricos usados no sub-comando:
  - *Star query* – Tipicamente, essas consultas são processadas avaliando apenas sub-árvores lineares, ordenando a sequência começando pelas junções menos custosas;
  - *Select query* – Tipicamente, essas consultas são processadas usando uma sequência definida heuristicamente;
  - *Default query* – Aplicam-se todas as regras de transformação possíveis.

Gerador de Planos de Consulta

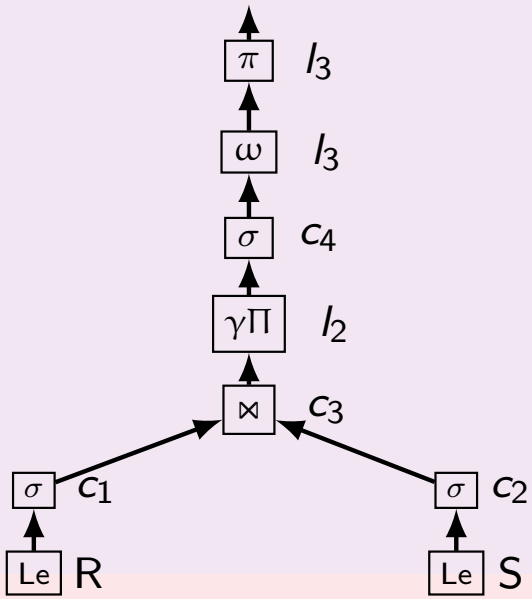
- A otimização visa reduzir:
  - O número de acessos a disco,
  - O uso de memória,
  - O uso do processador (ou dos processadores).
- Para a previsão de custo de cada operador, considera-se apenas o

  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).

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)
```



Expressão do plano:

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

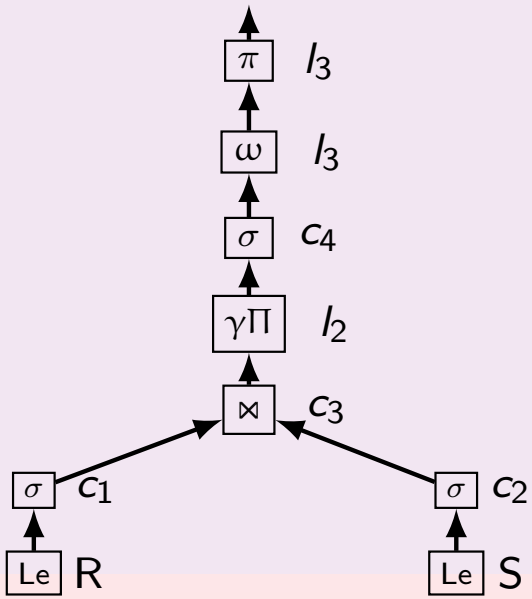
Nova coluna, indicando o numero de acessos a disco necessários para escrever o resultado.

Por exemplo,

- operador "Le": lê e escreve no *cache*. Total de escritas:
  $v_i = \#$Págs tabela
- operador σ:
  $v_i = \#$Págs op.anterior*(1-Sel(cond))

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)
```



Expressão do plano:

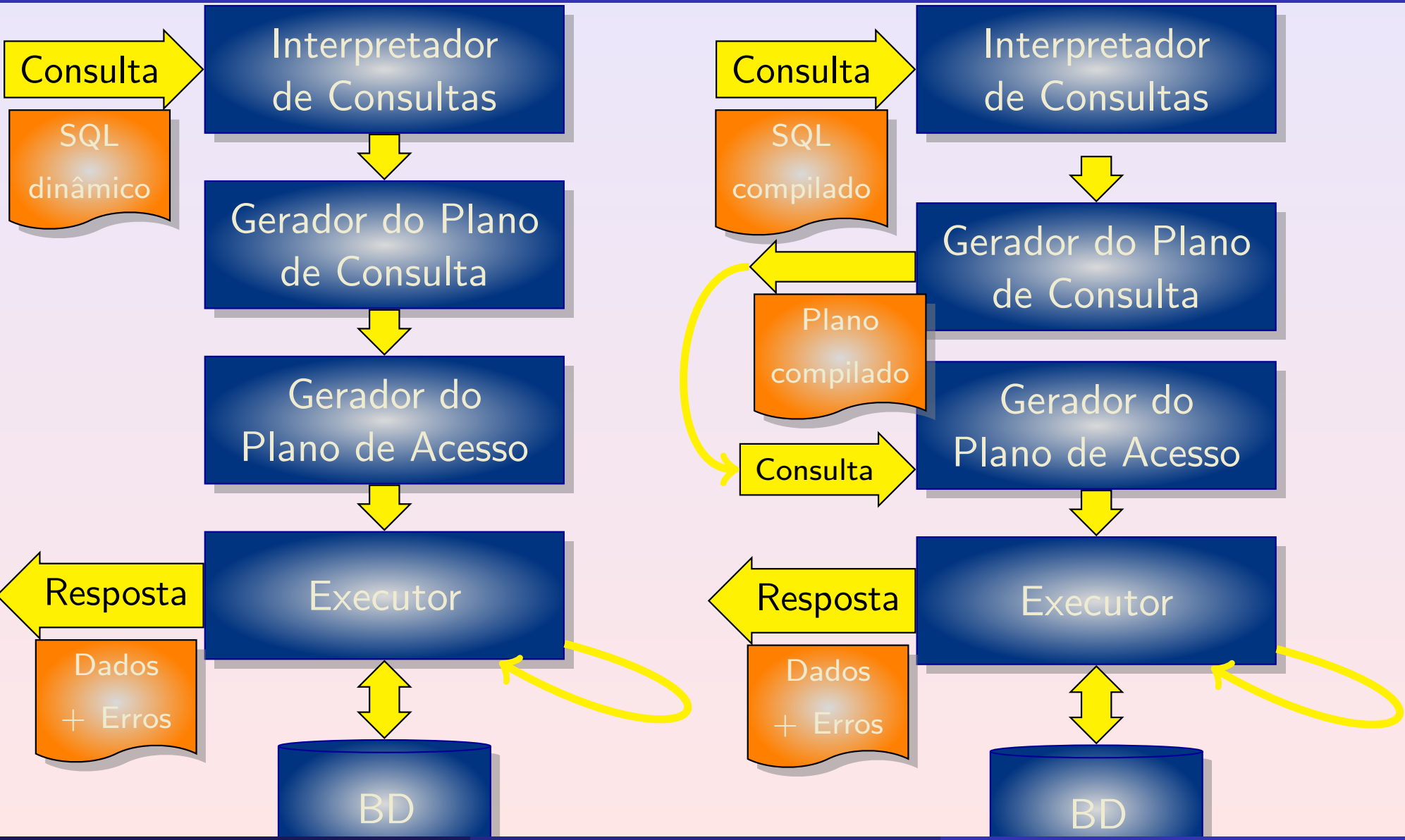
1	Le	R			v_1
2	Le	S			v_2
3	σ	1		c_1	v_3
4	σ	2		c_2	v_4
5	$\bowtie$	3	4	c_3	v_5
6	$\gamma\Pi$	5		$l_2..$	v_6
7	σ	6		c_4	v_7
8	ω	7		l_3	v_8
9	π	8		l_1	v_9

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

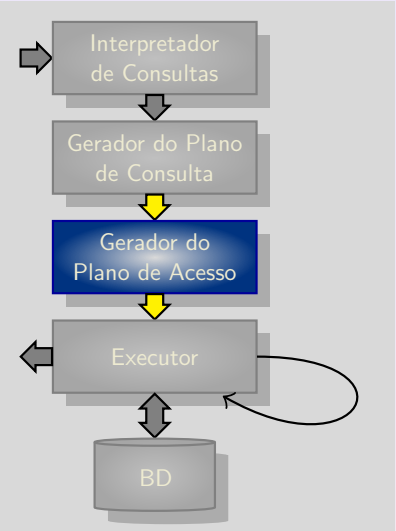
`INSERT` + `DDL` + `DCL`

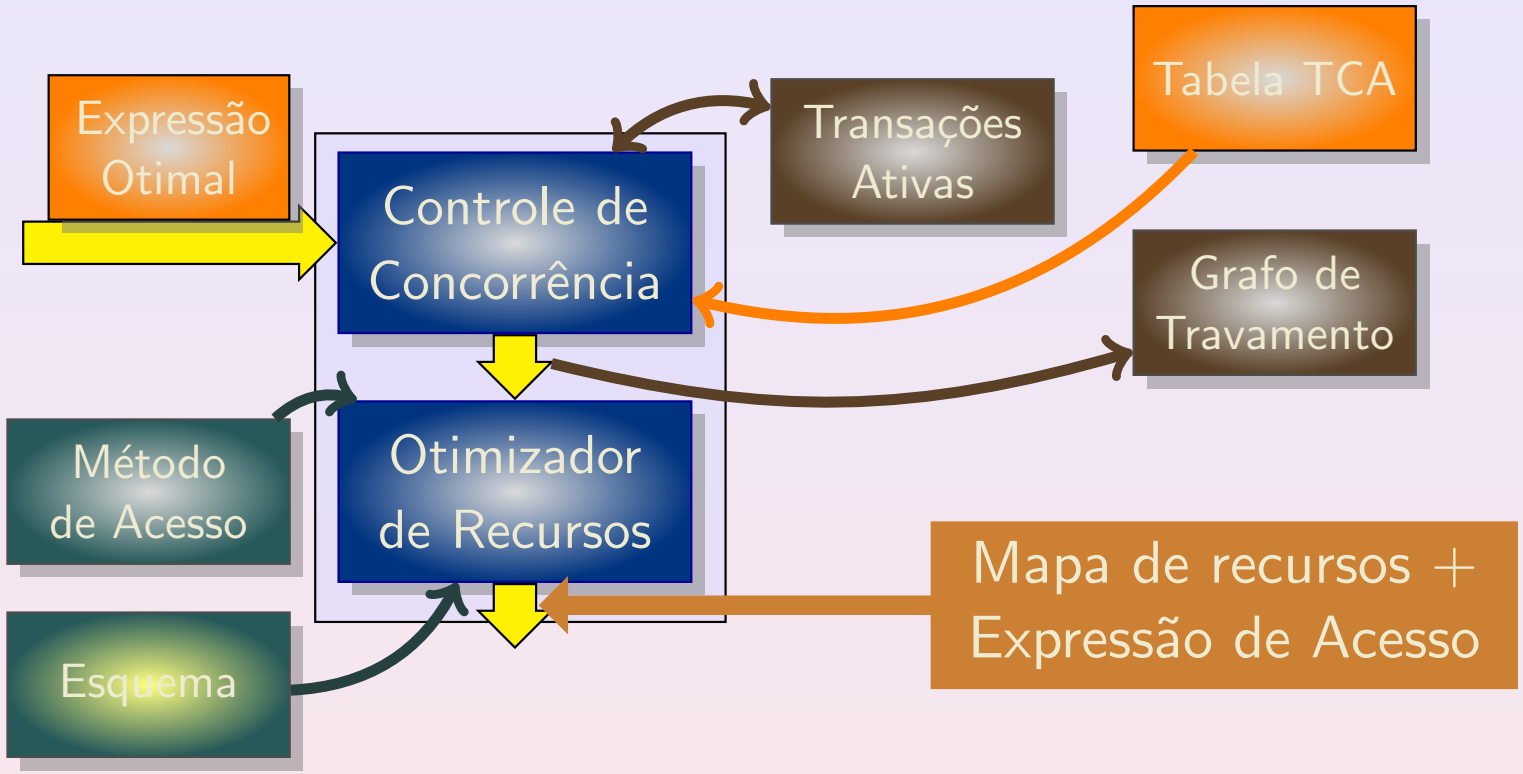
Acessar o esquema é um HOT SPOT!

- 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;
```

$$\Downarrow$$

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

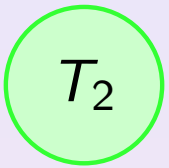
$$\Downarrow$$

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

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 muitissimo baixa.
  - Travar o atributo: concorrência muito baixa.
  - Travamento por predicado: maximiza concorrência a um custo aceitável.

Travamento por predicado




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

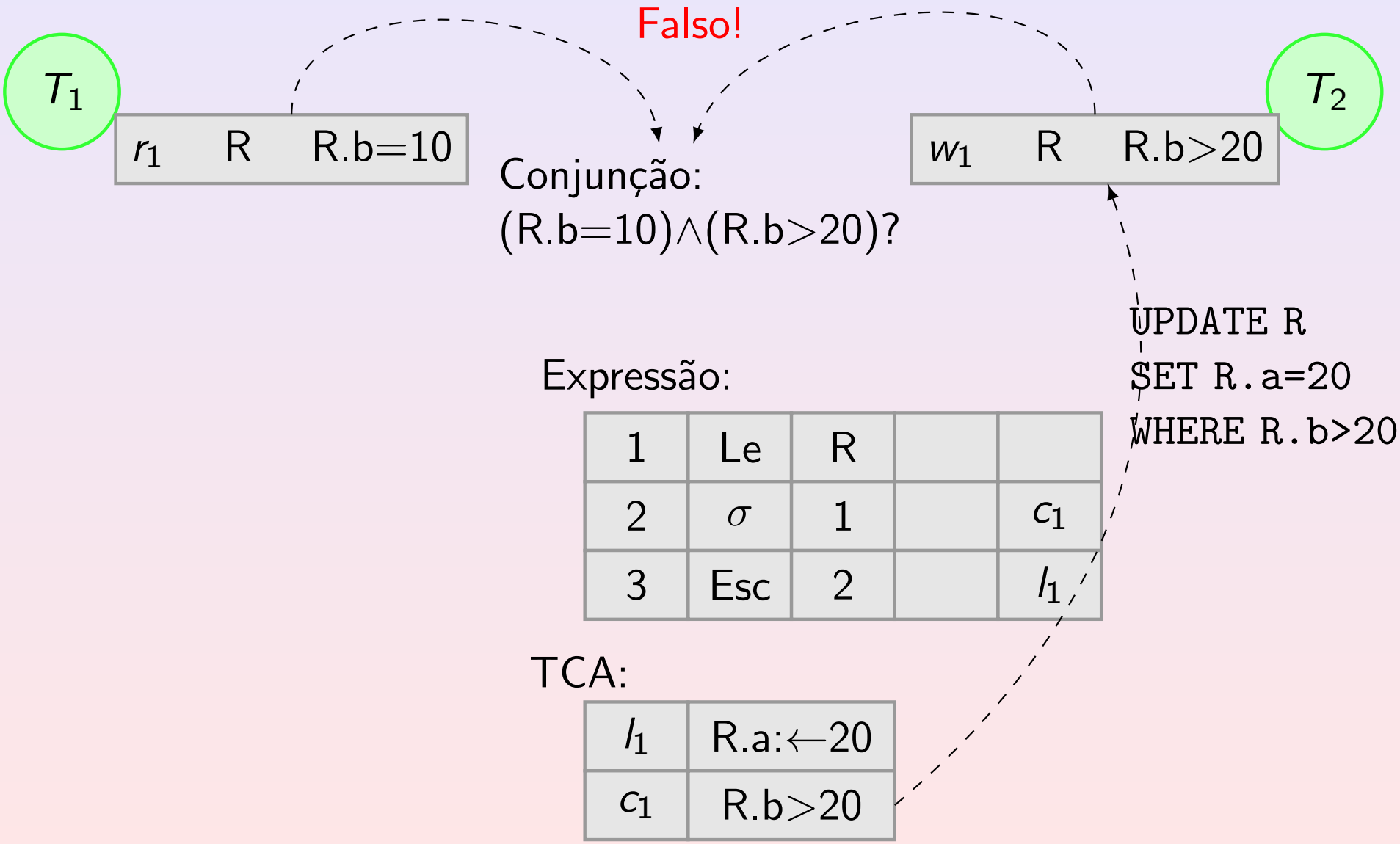
Expressão:

1	Le	R		
2	σ	1		c_1
3	π	2		l_1

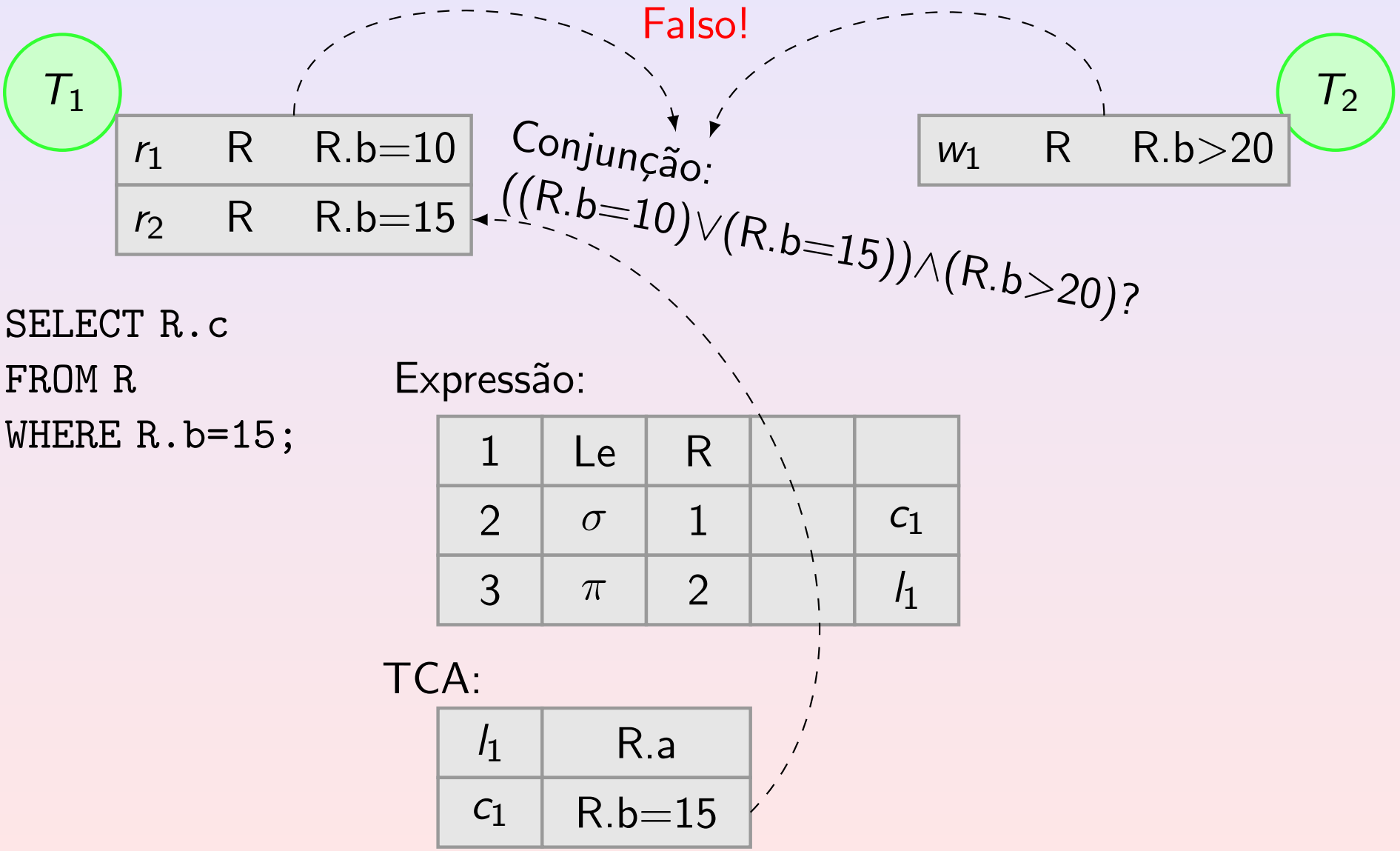
TCA:

l_1	R.a
c_1	R.b=10

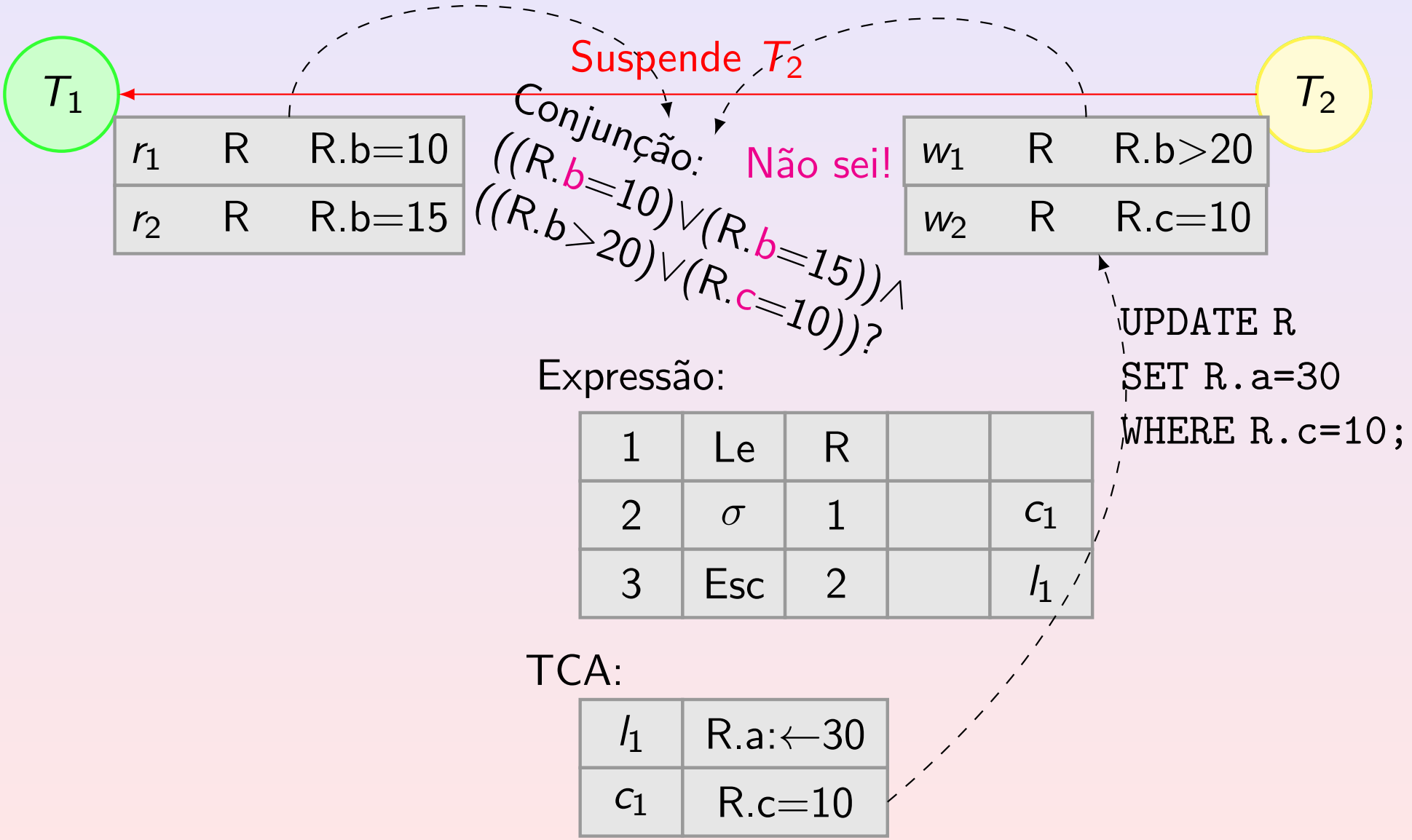
Travamento por predicado



Travamento por predicado



Travamento por predicado

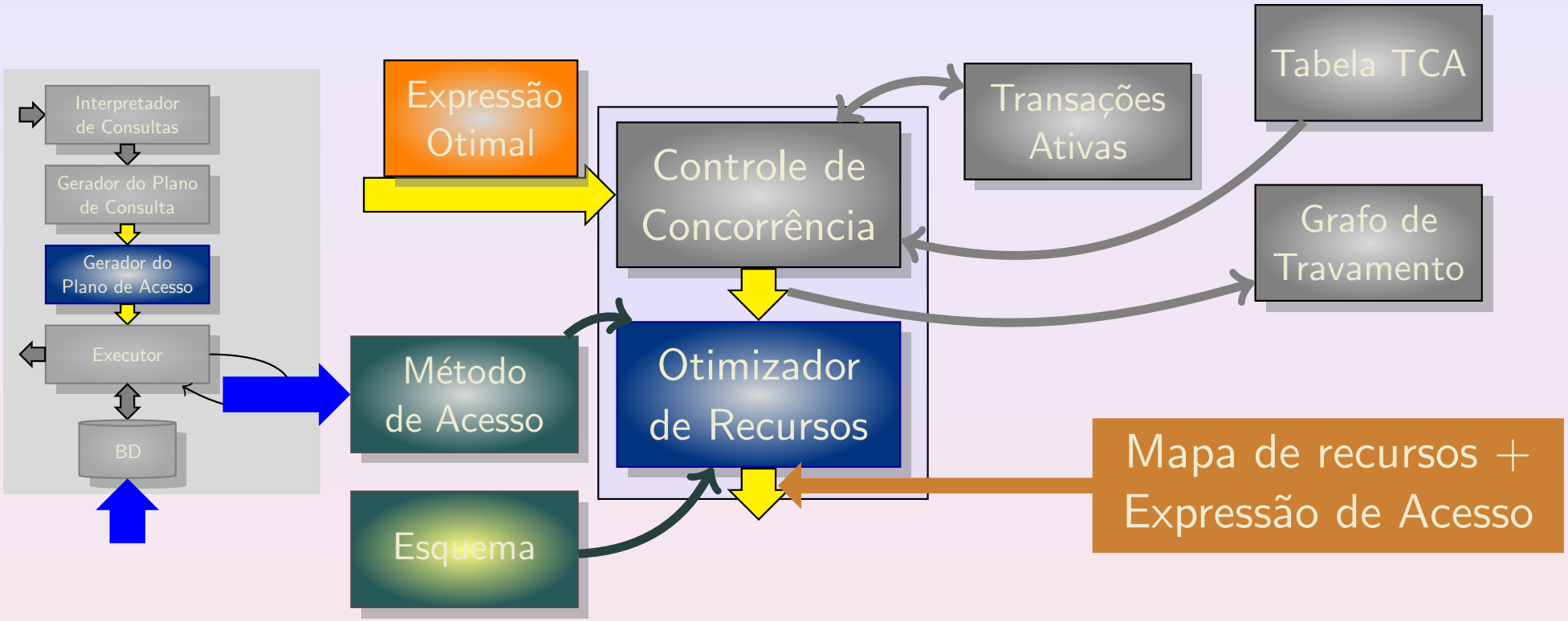


Controle de Concorrência

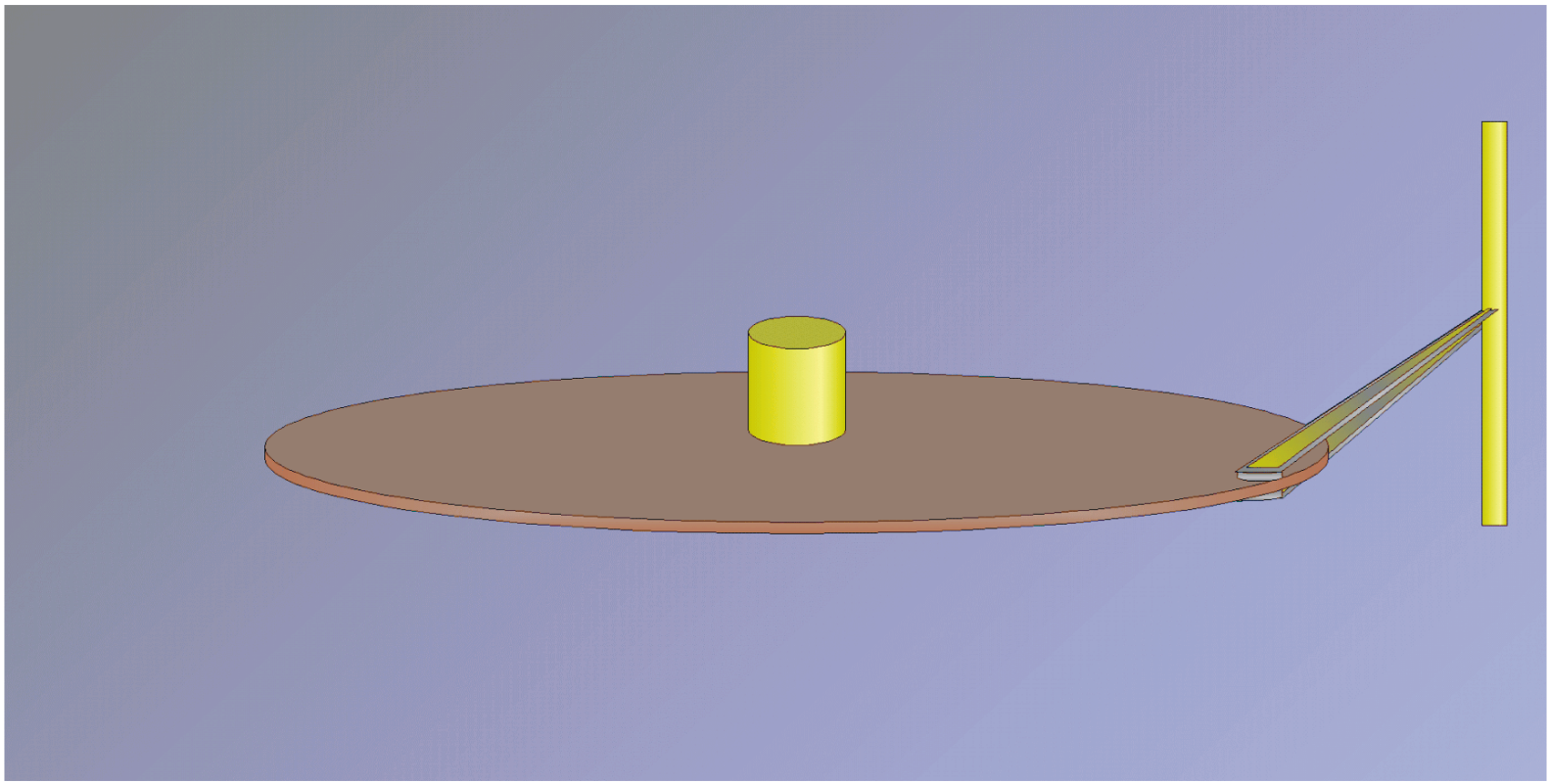
- Controle total implica o uso do Travamento em duas fases:
  - Fase 1: obtem os travamentos necessários (não libera nada);
  - Fase 2: libera todos os travamentos (não solicita mais nenhum).
- O travamento pode ser para:
  - Leitura: permite leitura compartilhada;
  - Escrita: permite leitura e escrita 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

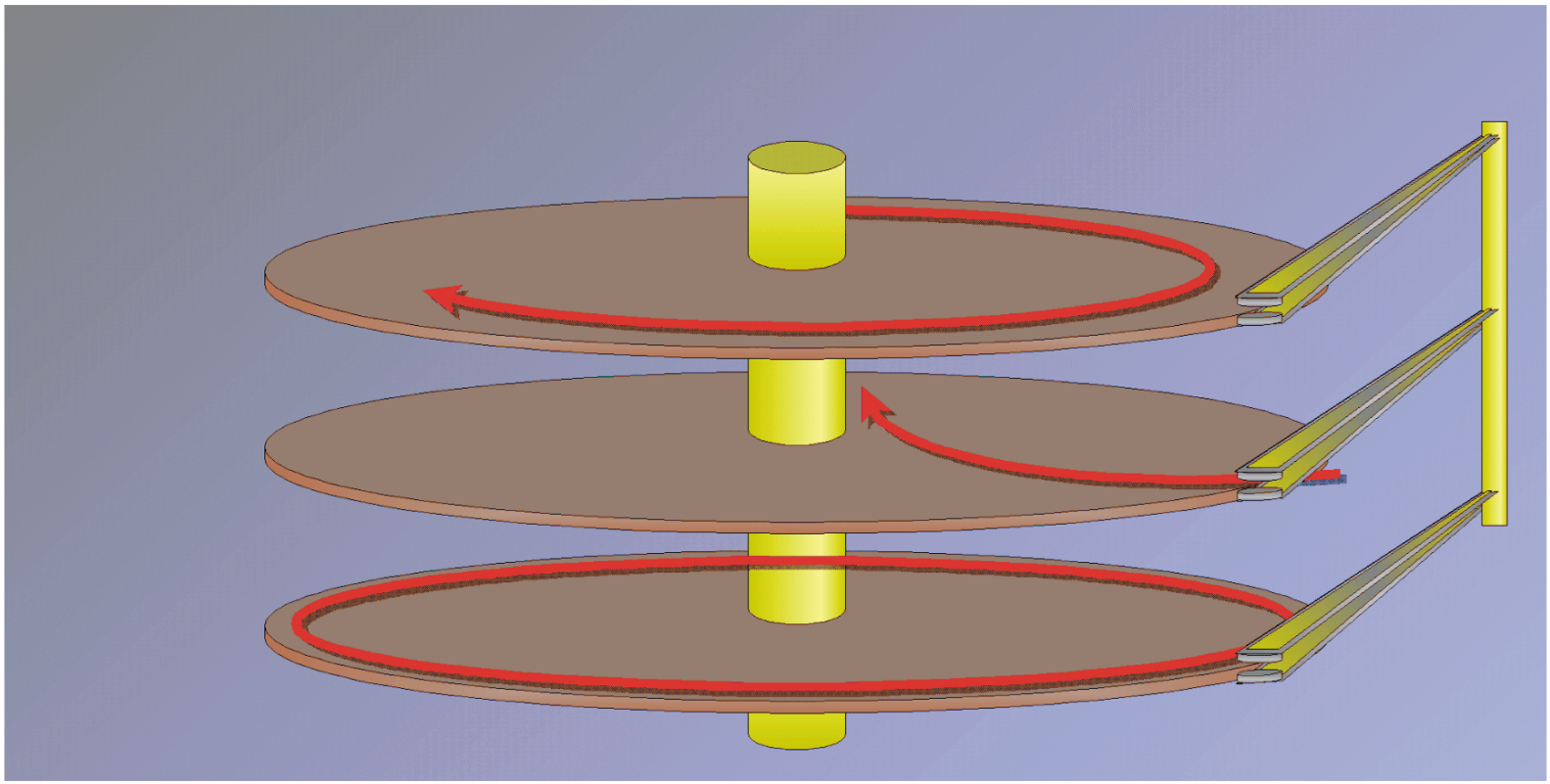


Arquitetura física das unidades de disco



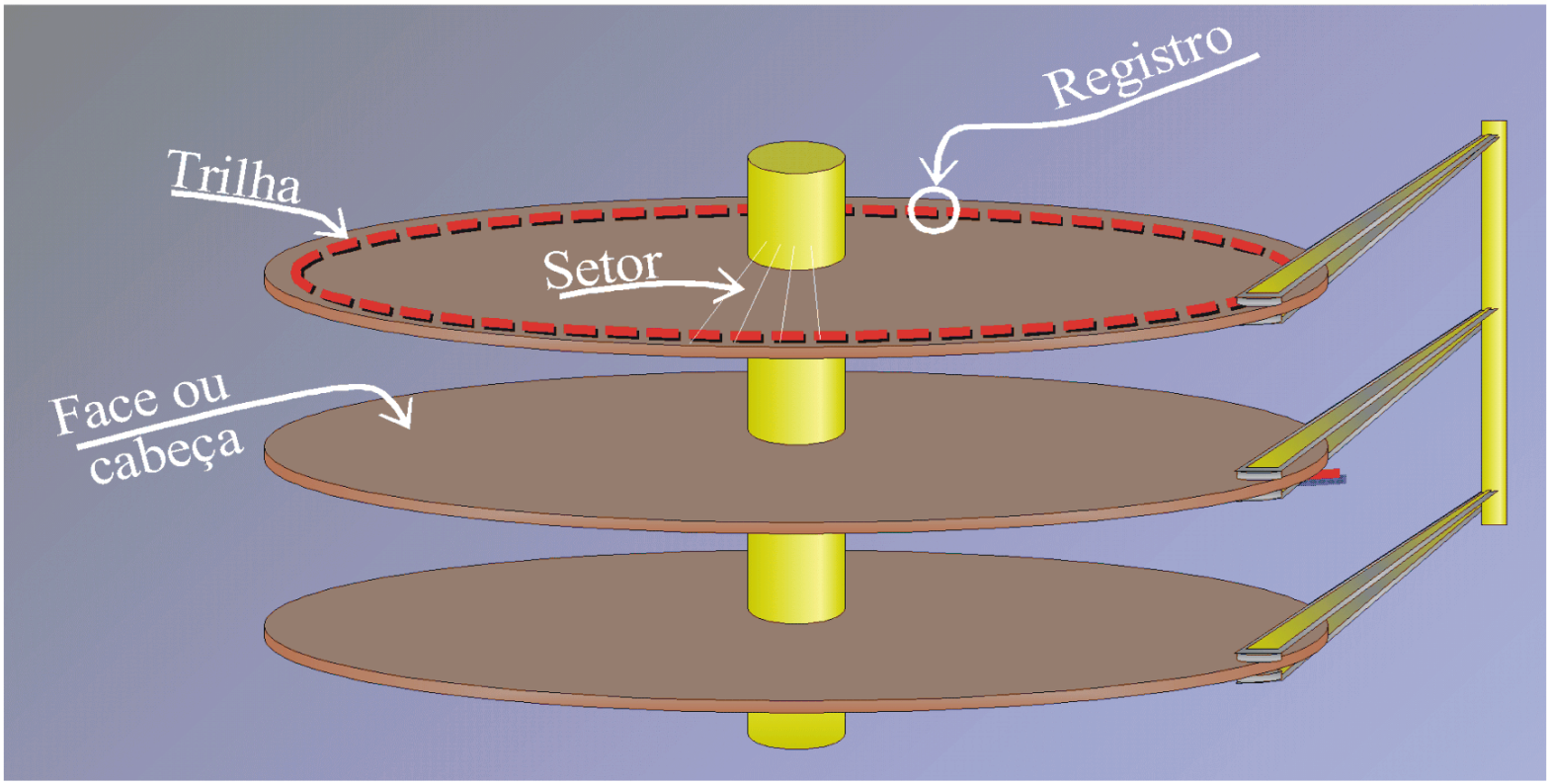
Arquitetura física das unidades de disco

Velocidade angular (RPM)
Tempo médio para posicionar as cabeças (ms)
Taxa de transferência (MBytes/s)



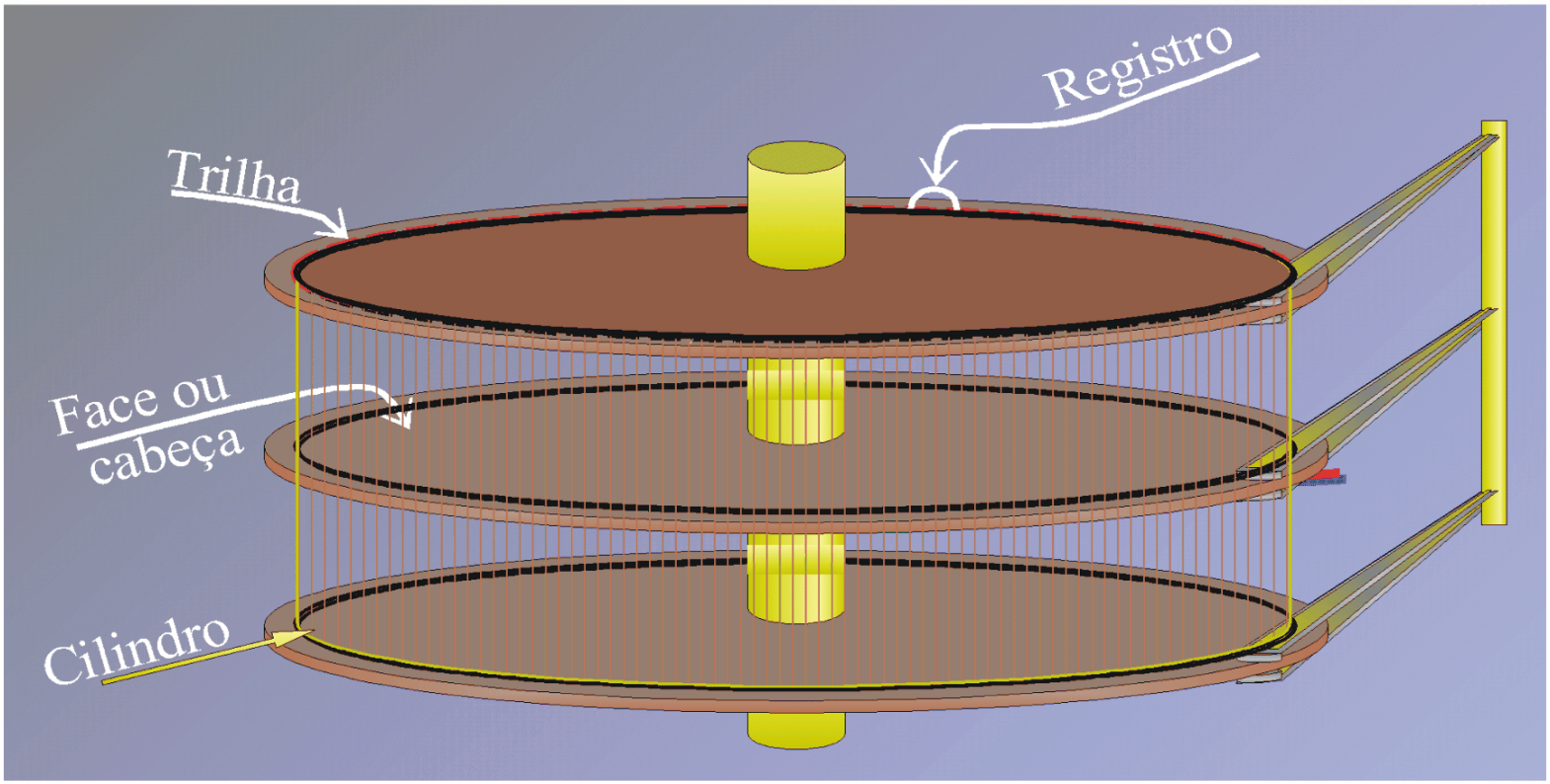
Arquitetura física das unidades de disco

Registro = 512 Bytes (em geral)



Arquitetura física das unidades de disco

Registro = 512 Bytes (em geral)



Disposição física dos dados em disco

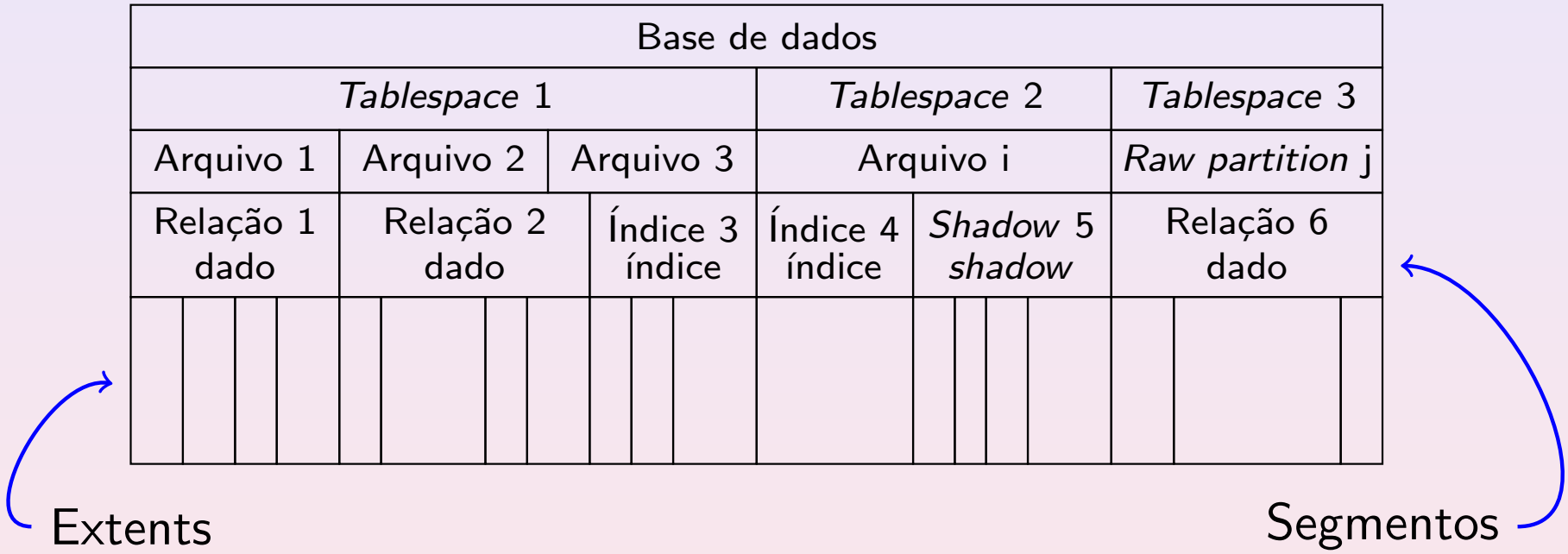
- Cada relação é guardada como uma seqüência de tuplas, de tamanho fixo ou aproximadamente fixo (média).
- Podem haver espaços em branco.
- Agrupam-se as tuplas em páginas de tamanho fixo,
  - múltiplo do tamanho dos registros em disco.
- Uma página é sempre lida inteira (mas podem ser vários registros).
- 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 multiplo 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: ≈ 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* $<<$ $0,01$ *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: ≈ 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

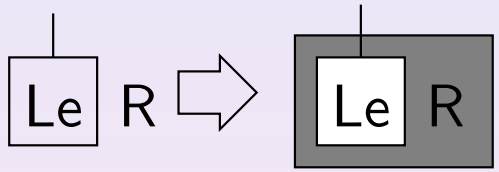
- Existem operadores de acesso (operadores físicos) para todos os operadores da álgebra relacional, bem como diversas das combinações de uso mais freqüente,
- E diversos métodos de acesso
  - Acesso sequencial (*sequentialscan*)
  - *Index-sequential access method* (ISAM ou B^+-*tree*)
  - *Hash*
  -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

Métodos de acesso

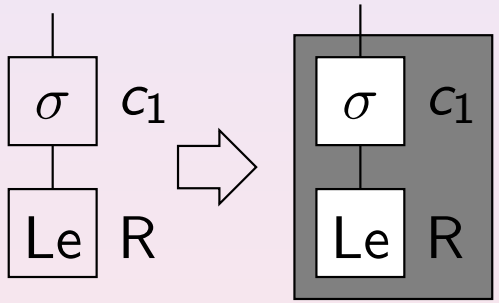
- Leitura simples
  - Leitura de dados do *extent*
- Leitura com seleção do *extent*
  - Sequencial $\Rightarrow |extent_i\ R| + |extent - 1\ R|$
  - ISAM $\Rightarrow H_R * |extent\ idxR| + 1\ extent\ R$ (para valores em uma tupla)
  - *Hash* $\Rightarrow |extent\ idxR| + 1\ extent\ R$ (para valores 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

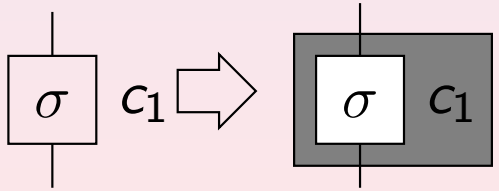
- Leitura simples:



- Leitura com seleção do *extent*:

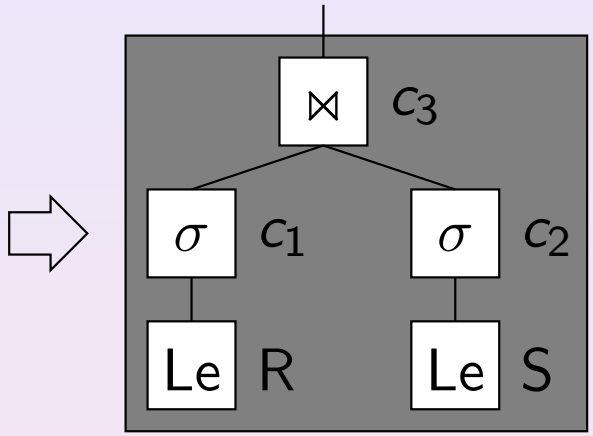


- Leitura com seleção do cache:

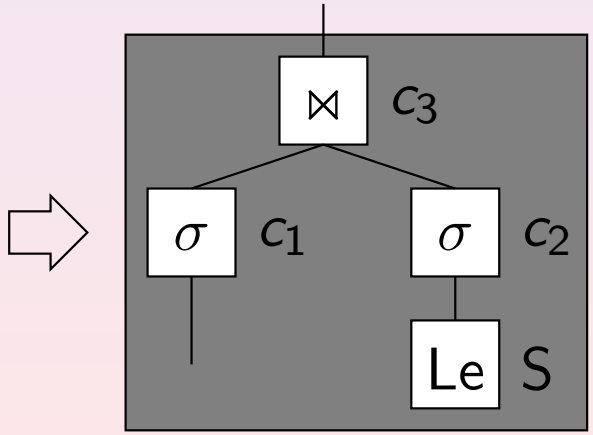


Métodos de acesso (cont.)

- Leitura com seleção seguida de junção:



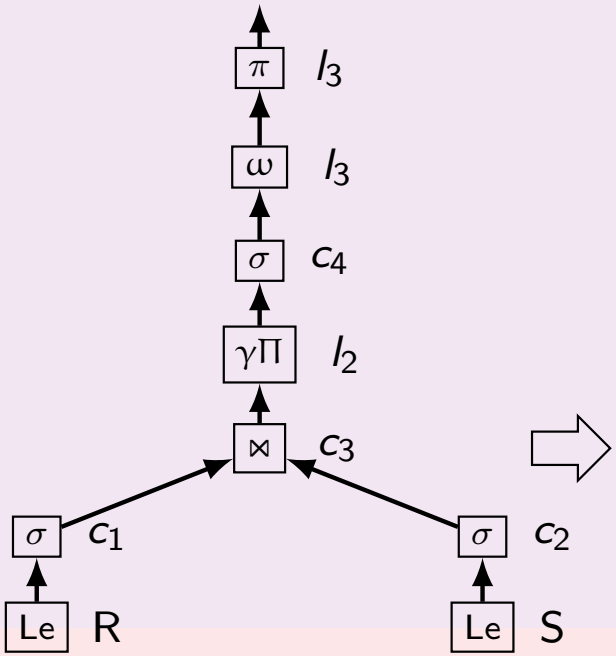
- Seleção no cache seguida de leitura com junção:



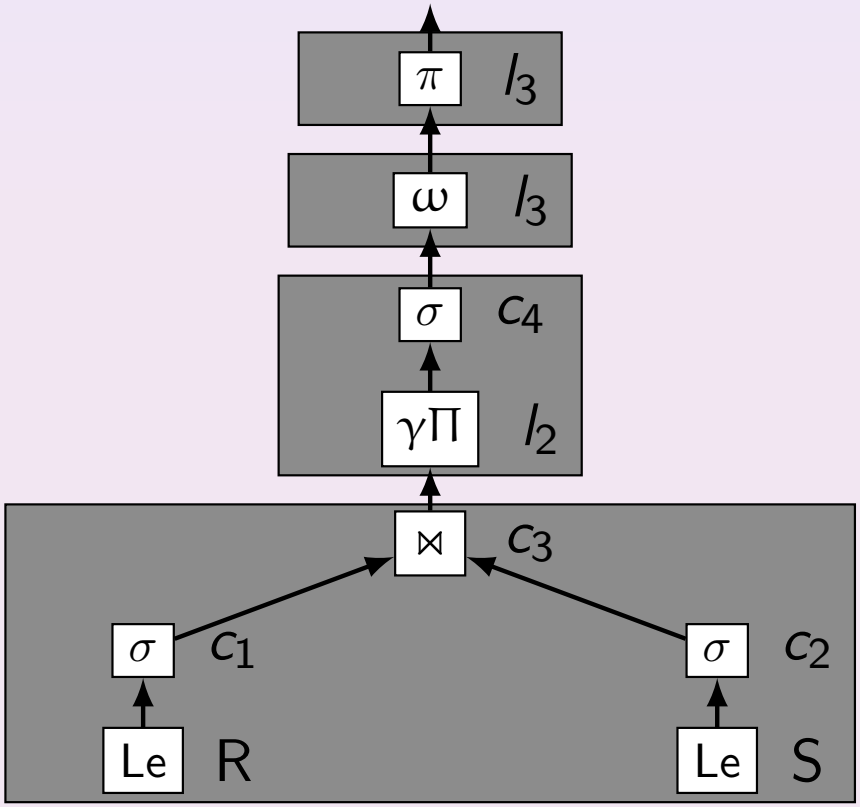
- Seleção no cache seguida de junção no cache.

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.

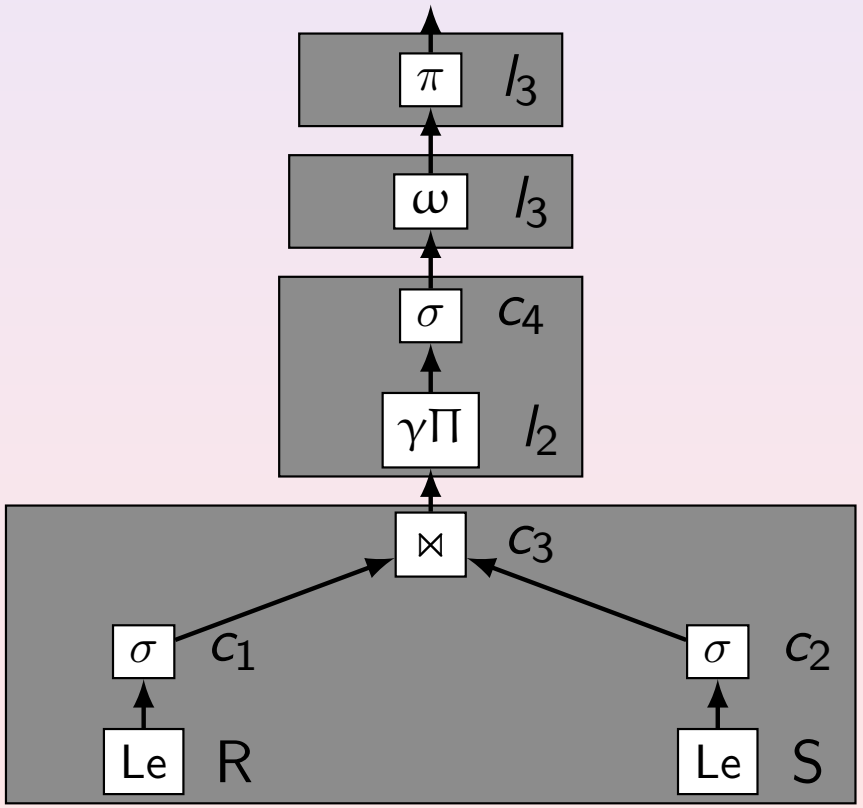
Otimizador de recursos

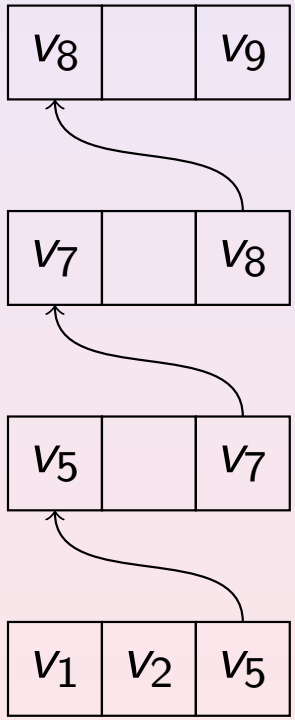
```
SELECT R.a, S.b
FROM R, S
WHERE R.a = 10
AND S.b > S.c
AND R.d = S.e
GROUP BY R.a, S.b, R.x
HAVING count(R.y) > 5
ORDER BY count(R.y)
```

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

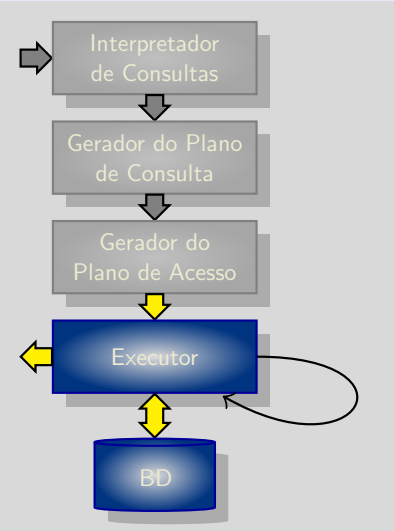
Seja M o total de registros do cache para a transação. Cada operador i terá n_i registros no cache:

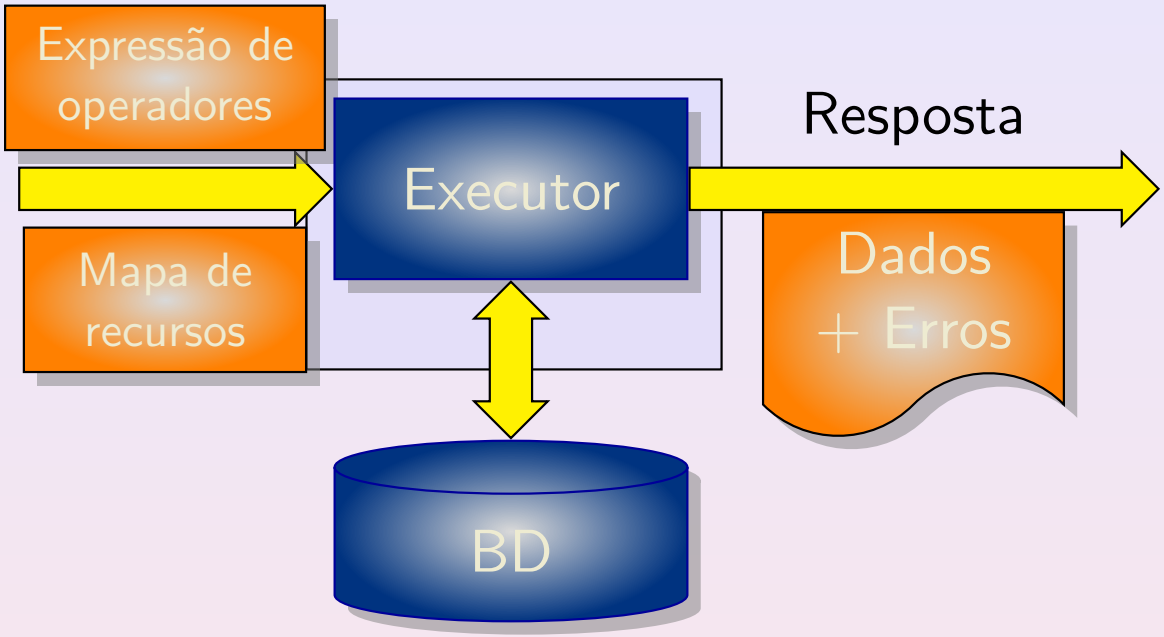
$$n_i = \frac{M * v_i}{(v_1 + v_2 + v_5 + v_7 + v_8 + v_9)}$$



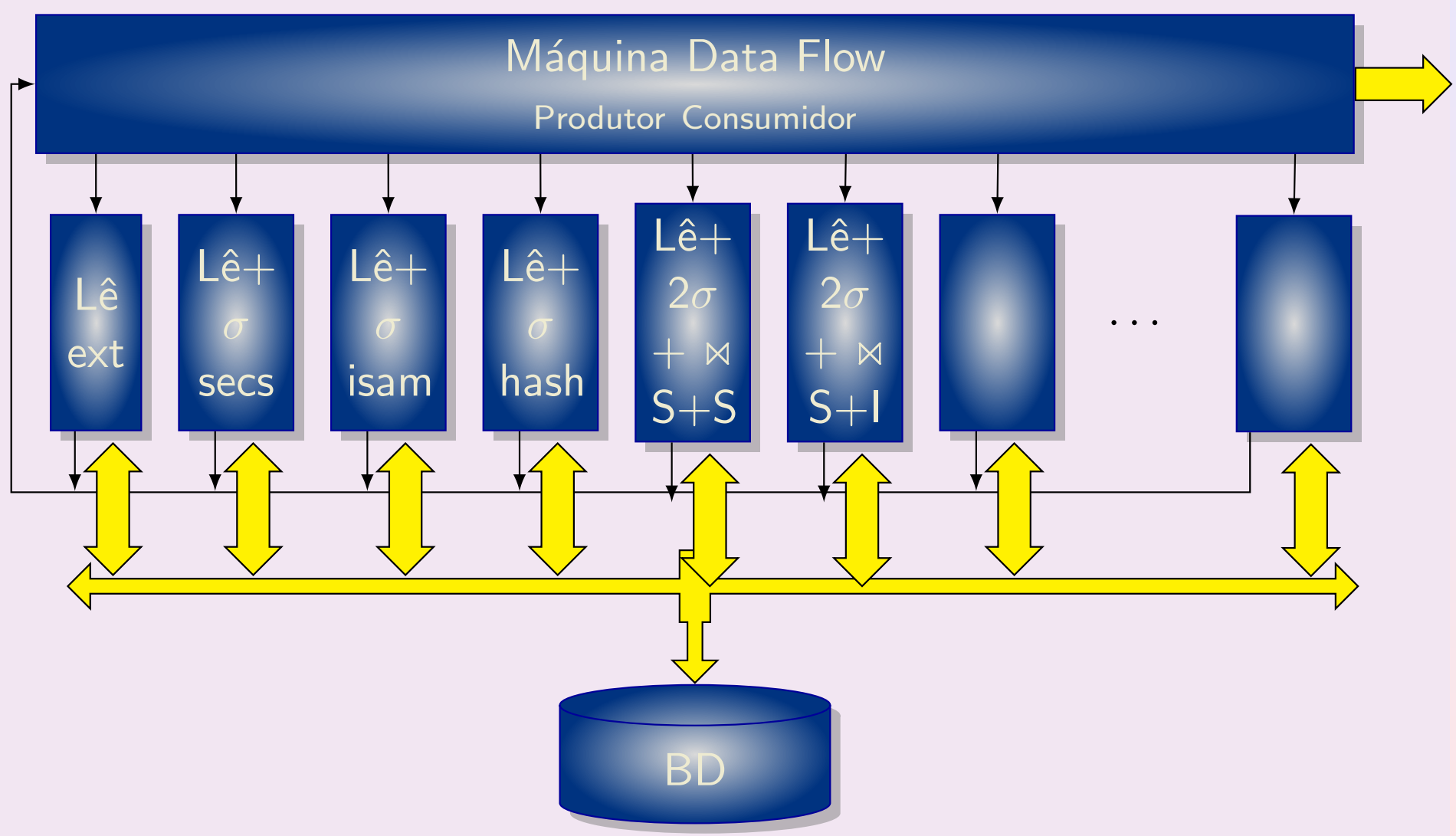


Executor


Interpretador de Consultas
Gerador do Plano de Consulta
Gerador do Plano de Acesso
Executor
BD


Expressão de operadores
Mapa de recursos
Executor
Resposta
Dados + Erros
BD

Executor


Máquina Data Flow
Produtor Consumidor
Lê ext
Lê+ σ secs
Lê+ σ isam
Lê+ σ hash
Lê+ 2σ + ⋈ S+S
Lê+ 2σ + ⋈ S+I
...
BD

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

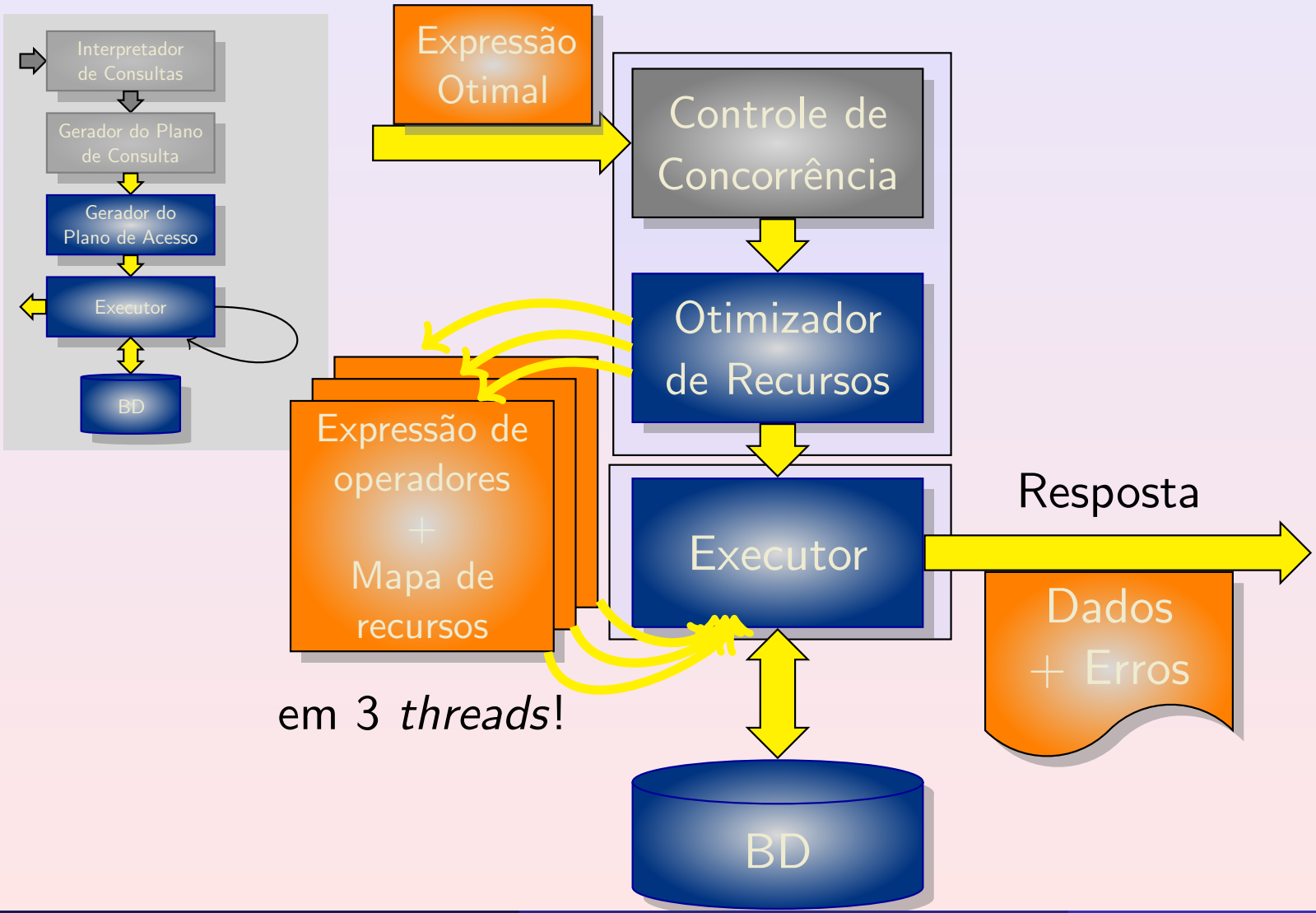
- Escolha dos métodos de acesso que melhor se adaptam para os tipos de consulta mais freqüentes em que cada relação possa participar.
  - Junções e chave estrangeira - ISAM
  - Tabelas de definição de atributos discretos (pequenas) - *HASH*
- Tabelas pouco atualizadas devem ter todas as chaves e atributos de junção sempre ligados em ISAM.
- Considerar se é possivel não usar índices e chaves estrangeiras em tabelas com atualização freqüente.
  - 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*.

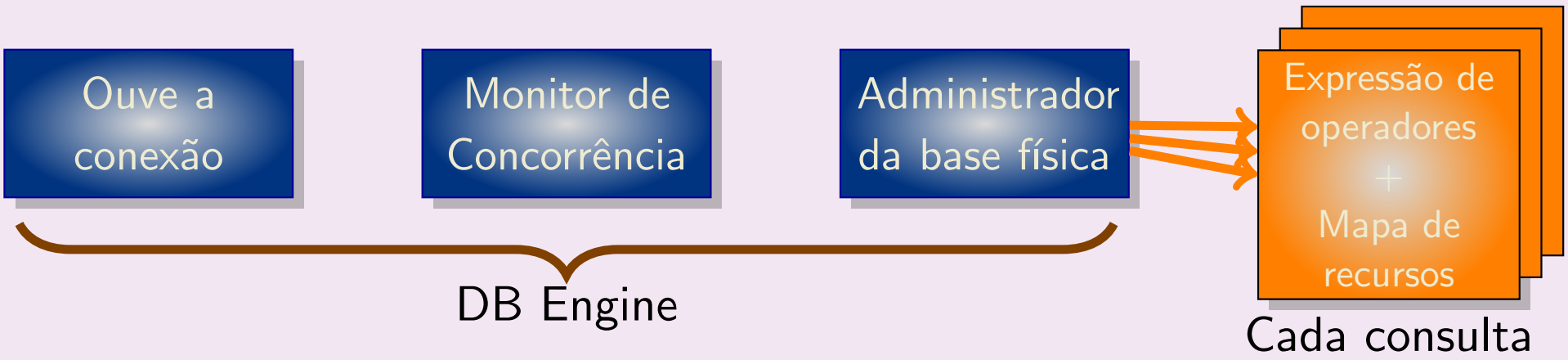
Execução em *multi-thread*

Manutenção de regras para dependências funcionais e *triggers*.



Execução em *multi-thread*

- O Gerenciador é constituido 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.



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

Caetano Traina Jr. — Versão Beamer: Mônica Ribeiro Porto Ferreira

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

7 de março de 2013
São Carlos, SP - Brasil

FIM