Planos de Consulta

Otimizador de consultas

```
teste=# explain (analyse,timing)  select * from pgbench_accounts as a join pgbench_branches as  b on a.bid=b.bid;
                                                                 QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.12..3928.14 rows=100000 width=461) (actual time=0.034..61.786 rows=100000 loops=1)
   Join Filter: (a.bid = b.bid)
   ->  Index Scan using pgbench_branches_pkey on pgbench_branches b  (cost=0.12..8.14 rows=1 width=364) (actual time=0.
   ->  Seq Scan on pgbench_accounts a  (cost=0.00..2670.00 rows=100000 width=97) (actual time=0.009..21.723 rows=100000
 Planning Time: 0.181 ms
 Execution Time: 69.287 ms
(6 rows)
```

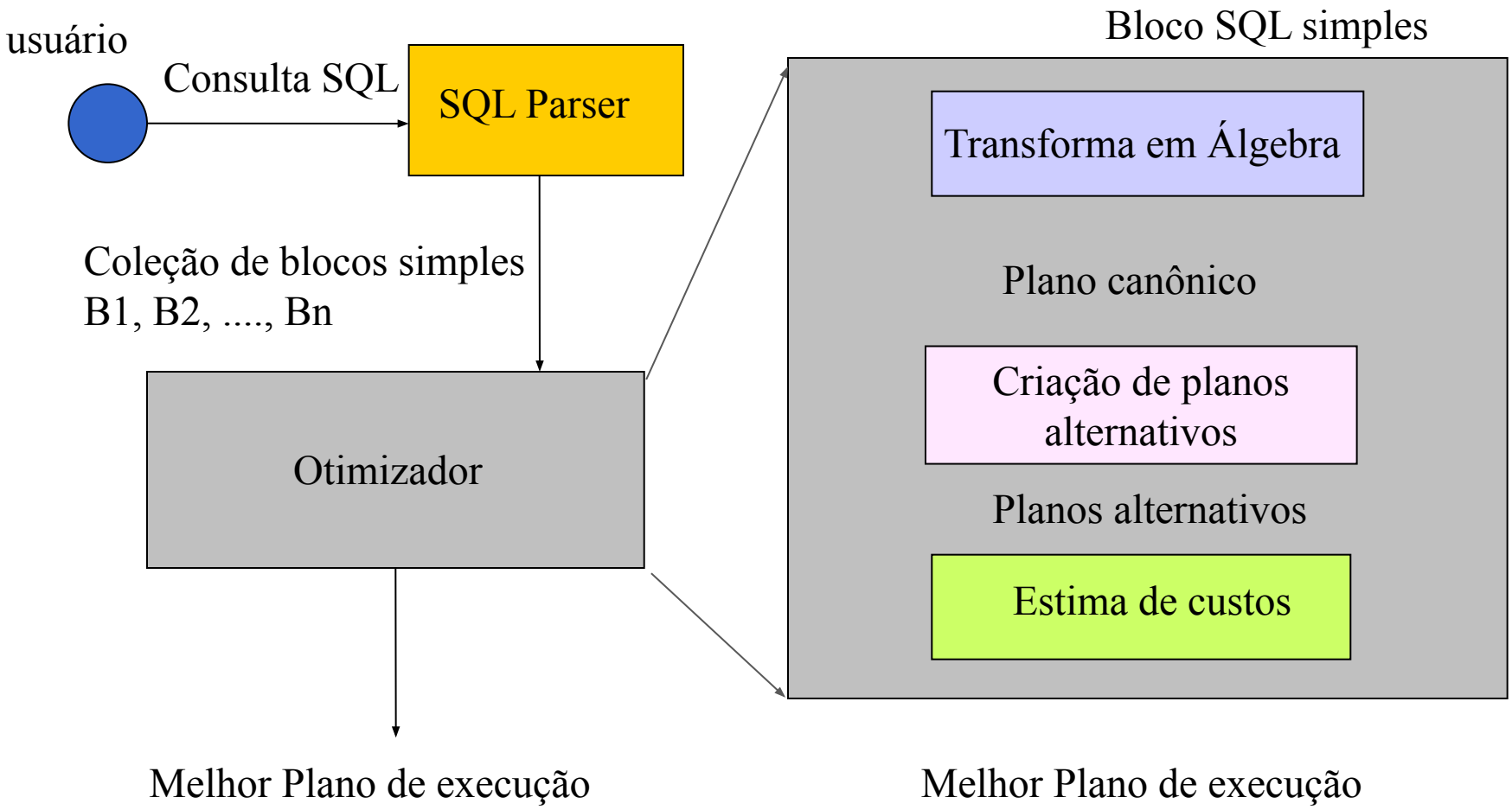
```
teste=# explain (analyse, timing)  select * from pgbench_accounts as A, pgbench_branches as  B where A.bid=B.bid;
                                                                 QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.12..3928.14 rows=100000 width=461) (actual time=0.033..50.932 rows=100000 loops=1)
   Join Filter: (a.bid = b.bid)
   ->  Index Scan using pgbench_branches_pkey on pgbench_branches b  (cost=0.12..8.14 rows=1 width=364) (actual time=0.
   ->  Seq Scan on pgbench_accounts a  (cost=0.00..2670.00 rows=100000 width=97) (actual time=0.009..17.533 rows=100000
 Planning Time: 0.178 ms
 Execution Time: 57.131 ms
(6 rows)
```

Otimizador de consultas

Problema: Uma query SQL é declarativa - não especifica o plano de execução.

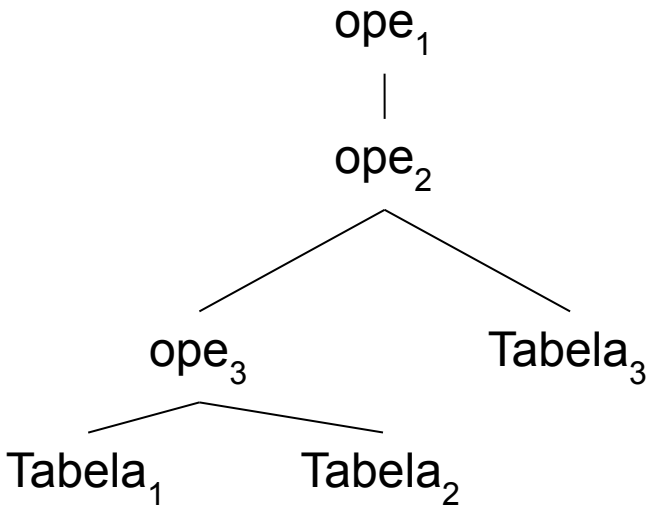
Solução: Converter a query para um plano equivalente da álgebra relacional.

Esquema Geral do Otimizador



Plano de Consulta

- Estratégia que o banco utiliza para executar uma consulta
- Normalmente, vários planos são proposto e um deles é escolhido
- O plano é um pseudocódigo em forma de árvore e álgebra relacional

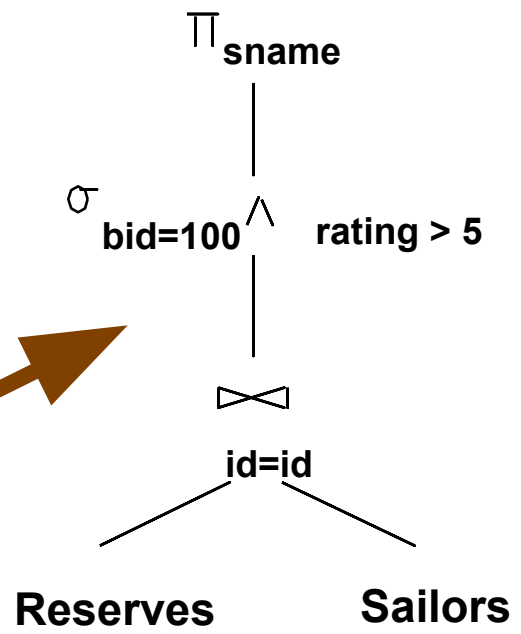


Otimização de consultas

- A consulta é convertida em álgebra relacional
- Álgebra relacional é convertida em uma árvore
- Cada operador pode ser alterado
- Operadores podem ser aplicados em diferentes ordens

```
SELECT  S.sname
FROM  Reserves R, Sailors S
WHERE  R.id=S.id AND
   R.id=100 AND S.rating>5
```

$\pi_{(sname)}\sigma_{(id=100 \wedge rating > 5)}$ (Reserves $\bowtie$ Sailors)



Plano de Consulta

- Estratégias para processar consulta
  - Qual tabela processar primeiro
    - Mais ou menos volumosa
  - Utilizar índice
  - Ordenar tabela
  - Tratamento junção
  - Melhor decomposição
  - Quantos planos propor
  - Como escolher o melhor plano
  - Algoritmo de tratamento dos operadores

Esquema de exemplo

Sailors (*id*: integer, *sname*: string, *rating*: integer, *age*: real)
Reserves (*id*: integer, *bid*: integer, *day*: dates, *rname*: string)

Reserves:
- Cada tupla com 40 bytes, 100 tuplas por página , 1000 páginas.
- Assumir que existe 100 barcos diferentes, distribuídos uniformemente;

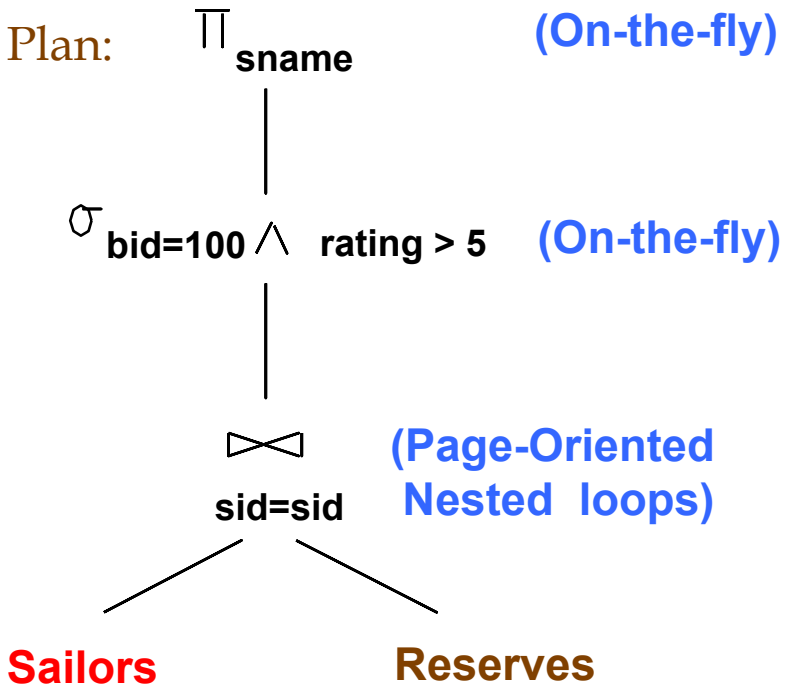
Sailors:
- Cada tupla com 50 bytes, 80 tuplas por página, 500 páginas;
- Assumir que existem 10 diferentes ratings, distribuídos uniformemente;

- Assumir que temos 5 páginas no buffer

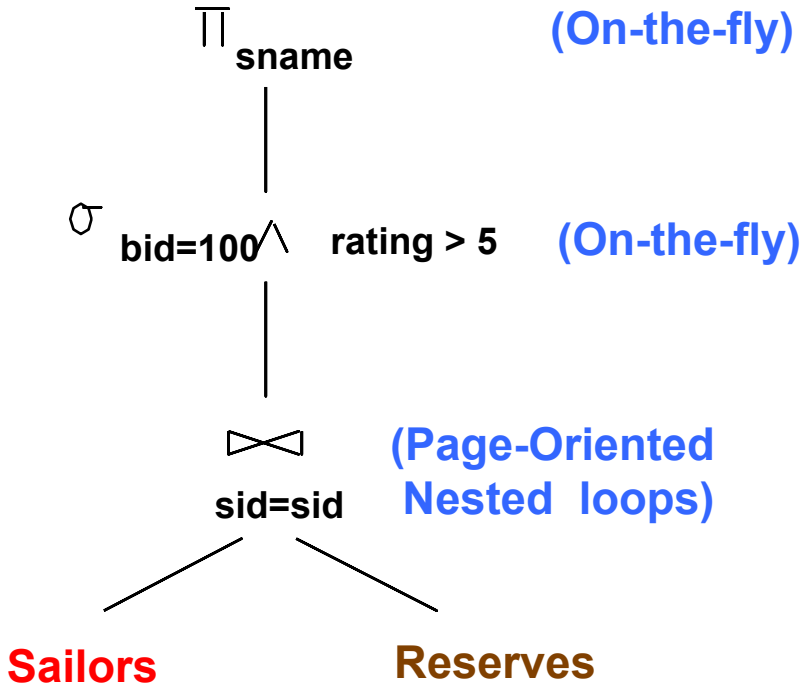
Exemplo

```
SELECT  S.sname
FROM  Reserves R, Sailors S
WHERE  R.id=S.id AND
   R.id=100 AND S.rating>5
```

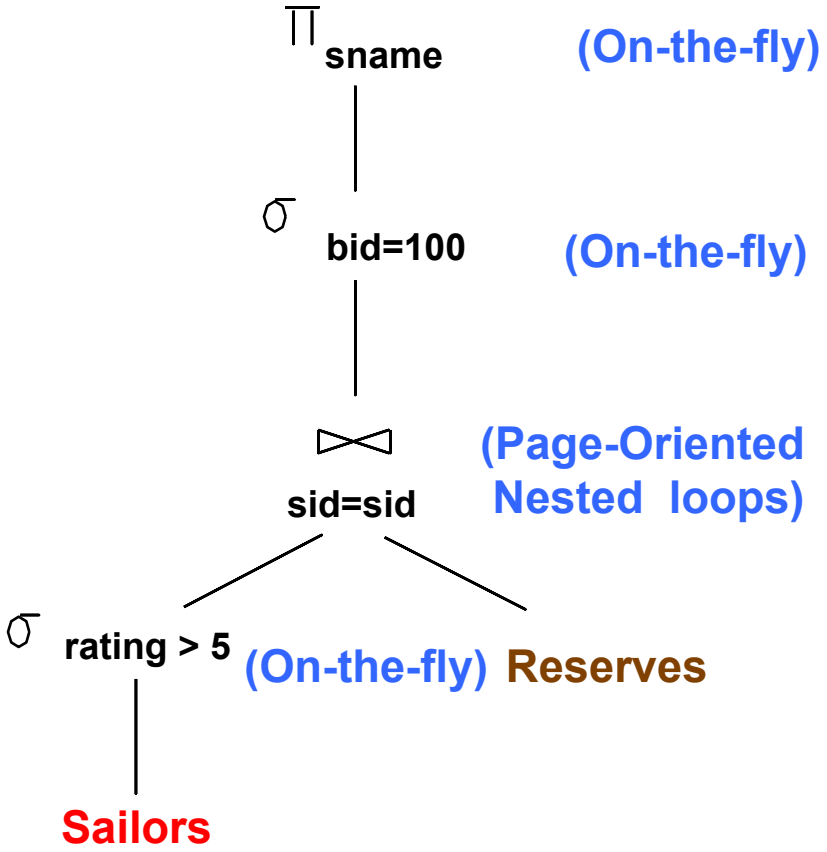
- Custo:



Planos alternativos

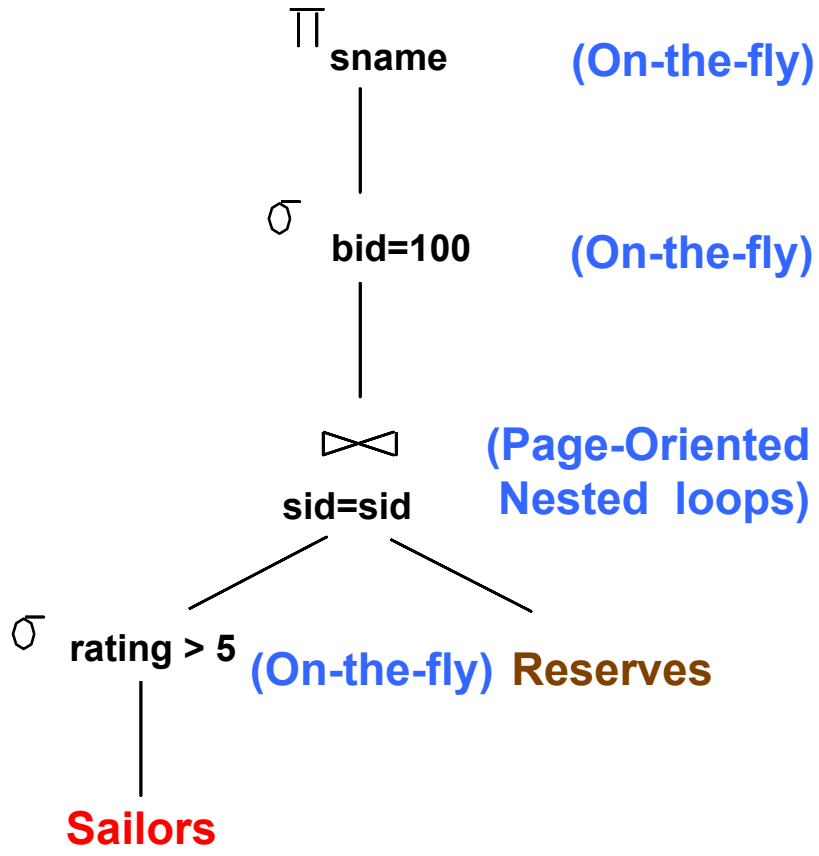


500.500 IOs

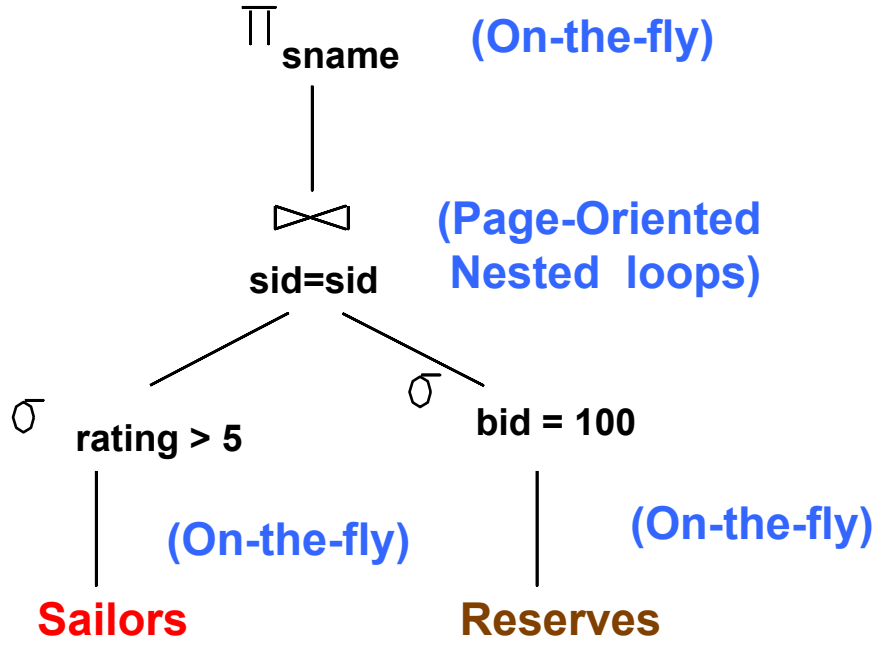


250.500 IOs

Planos alternativos

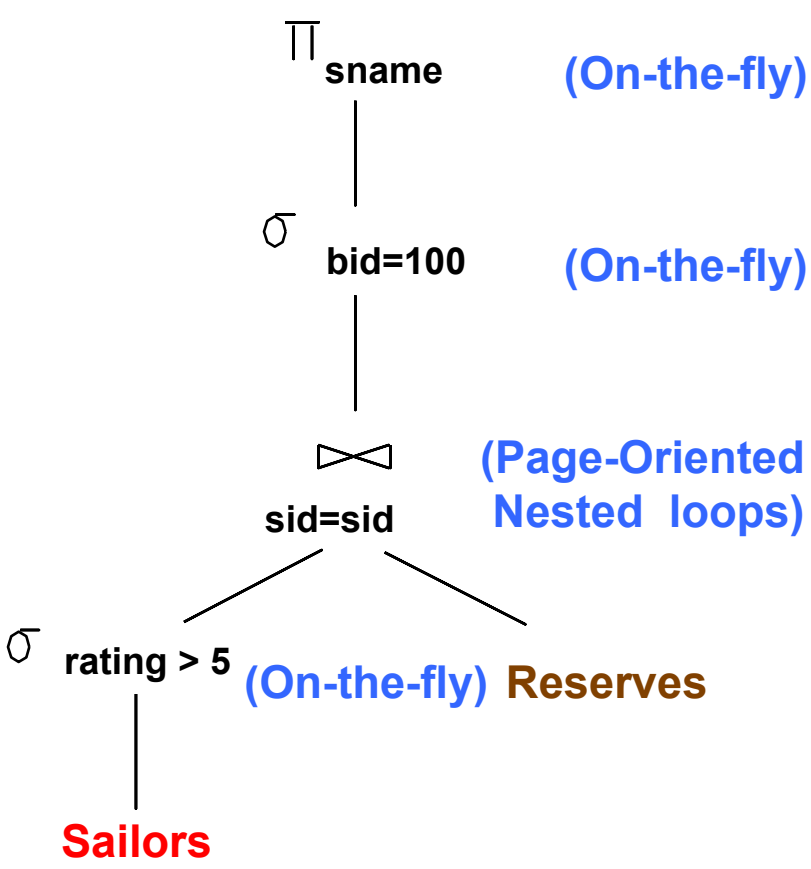


250,500 IOs

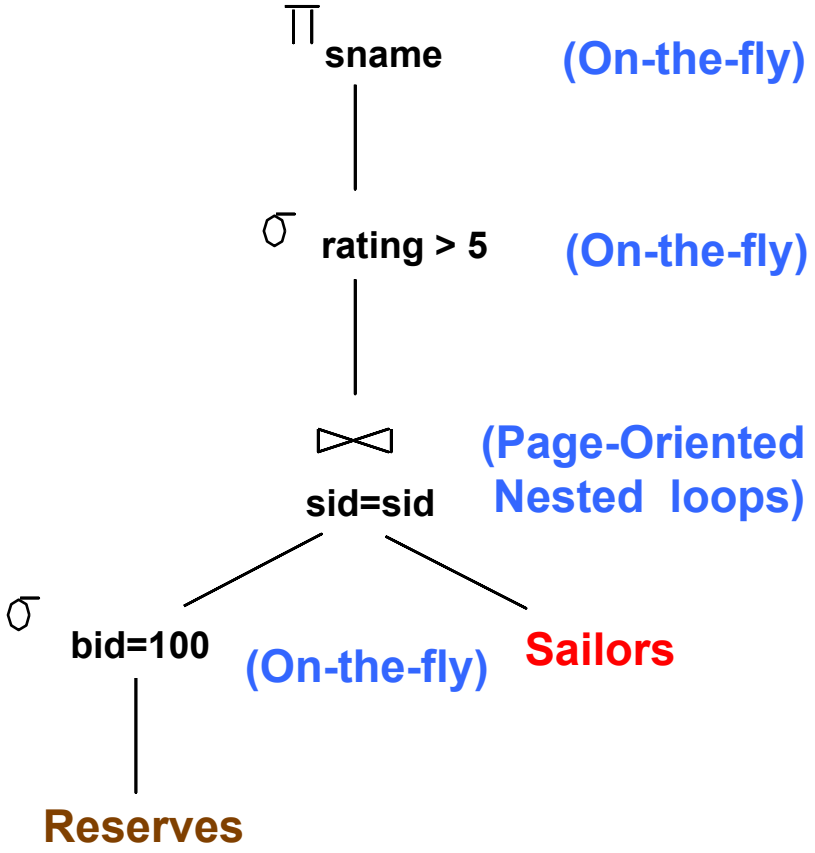


250,500 IOs

Planos alternativos

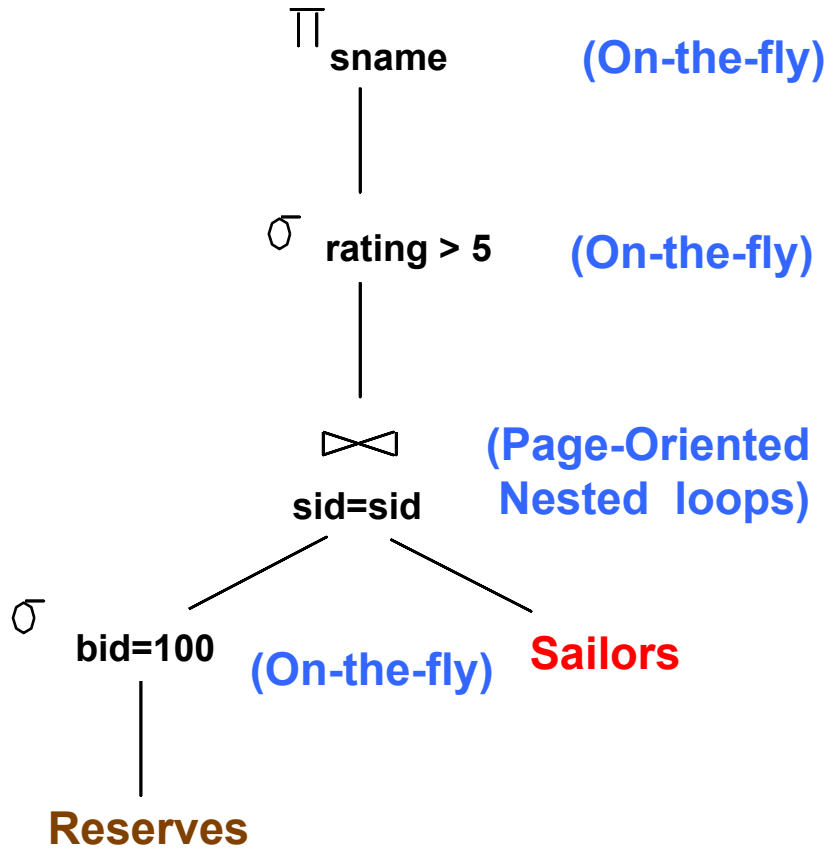


250,500 IOs

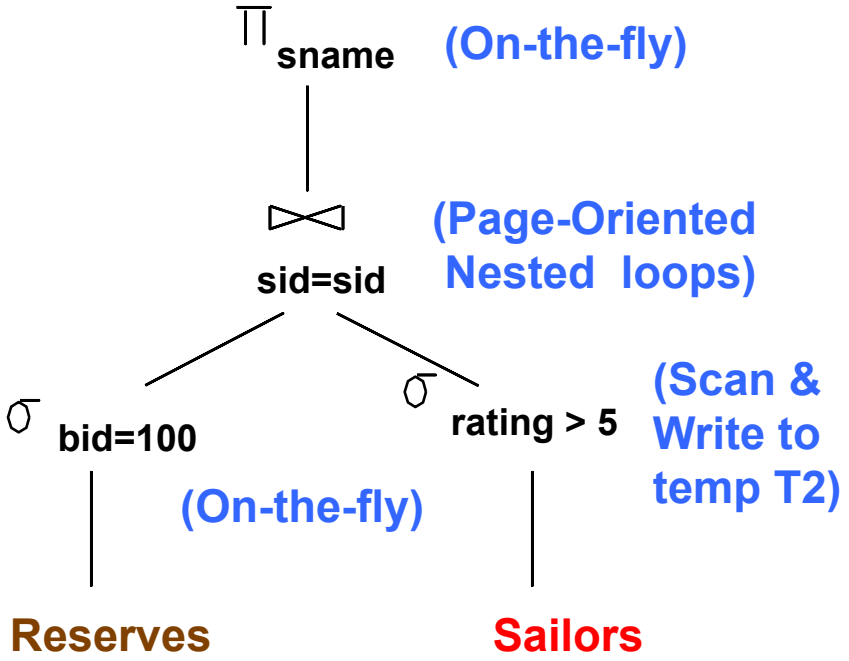


6000 IOs

Planos alternativos

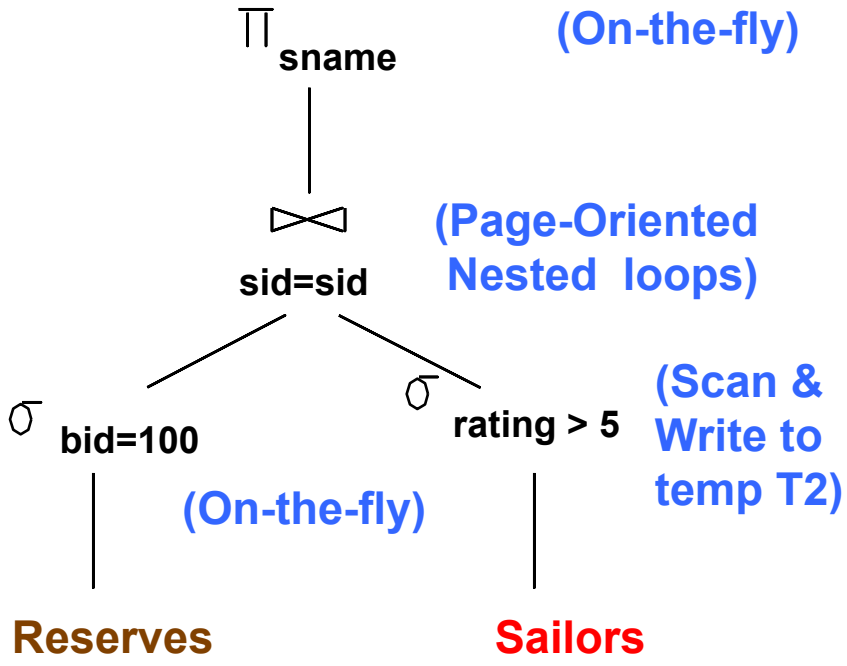


6000 IOs

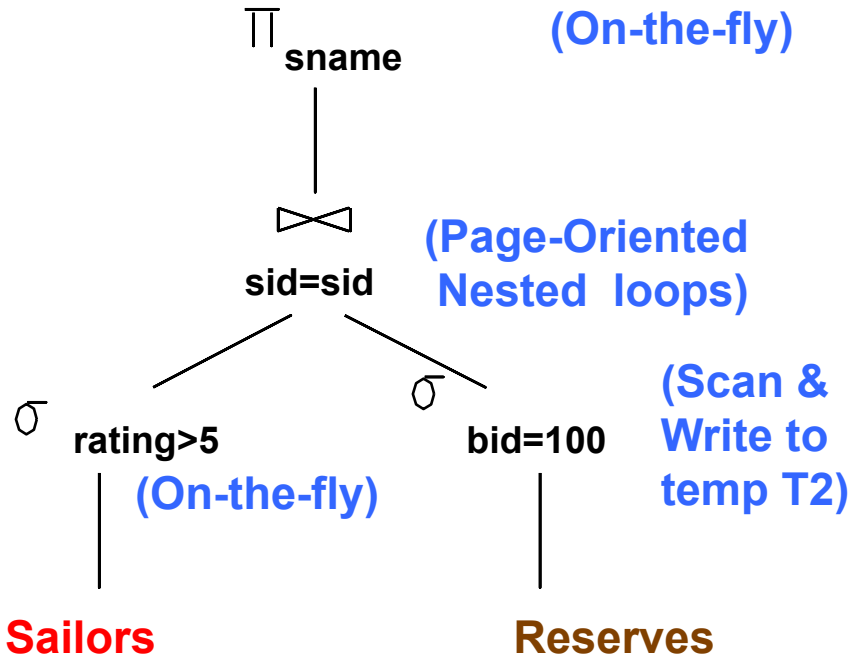


4250 IOs
1000 + 500+ 250
+ (10 * 250)

Planos alternativos



4250 IOs



4010 IOs
500 + 1000 +10
+(250 *10)

O que é preciso para melhorar uma consulta ?

1- Plano de consulta:
   - Baseado na equivalência relacional
2- Estimativa de custo:
   - Fórmulas
   - Estimativa de tamanho, baseado no catálogo e na Seletividade
3- Algoritmo de busca:
   - Busca no plano de consulta com base nos custos

Plano de Consulta

- Equivalências na álgebra relacional
  - Seleções
    - $\sigma_{c_1 \wedge c_2 \wedge \cdots \wedge c_n}(R) \equiv \sigma_{c_1}(\sigma_{c_2}(\cdots(\sigma_{c_n}(R)\cdots)))$
    - $\sigma_{c_1}(\sigma_{c_2}(R)) \equiv \sigma_{c_2}(\sigma_{c_1}(R))$
  - Projeção
    - $\pi_{a_1}(R) \equiv \pi_{a_1}(\pi_{a_2}(\cdots(\pi_{a_n}(R)\cdots)))$
  - Produtos cartesianos e junções
    - $R \times S \equiv S \times R$
    - $R \bowtie S \equiv S \bowtie R$
    - $R \times (S \times T) \equiv (R \times S) \times T$
    - $R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$

Plano de Consulta

- Equivalências na álgebra relacional
  - Seleções, projeções e junções
    - $\pi_a(\sigma_c(R)) \equiv \sigma_c(\pi_a(R))$
    - $R \bowtie_c S \equiv \sigma_c(R \times S)$
    - $\sigma_c(R \times S) \equiv \sigma_c(R) \times S$
    - $\sigma_c(R \bowtie S) \equiv \sigma_c(R) \bowtie S$

Plano de Consulta

- Heurística
  - Algumas regras que diminuem o custo da consulta
    - Quando possível empurra-se as projeções para a parte baixo da árvore
      - **Diminui o tamanho da resposta**
        - Relação R(a, b, c) com 20.000 tuplas
        - Cada tupla 190 bytes (header = 24 bytes, a = 8 bytes, b = 8 bytes, **c = 150 bytes**)
        - Bloco 1024b
        - 1 bloco = 5 tuplas (5 * 190 = 950)
        - Para 20.000 tuplas: **4.000 blocos**
        - **Projeção eliminando o atributo c:**
        - Tupla 40 b, cabem 25 em um bloco
        - Necessários **800 blocos** (fator de redução 5)

Plano de Consulta- baseado em heurística

- Heurística
  - Algumas regras que diminuem o custo da consulta
    - Quando possível deixa-se as **seleções** próximas as tabelas que são aplicadas
      - Diminui o número de tuplas para serem processadas mais adiante
    - Aplicar os **joins** por último (quando possível)

Plano de Consulta- baseado em heurística

Algoritmo básico

Passo 1: Quebre as seleção com condições conjuntivas (and) em uma cascata de operações de seleção

Passo 2: Mova cada operação de seleção o mais baixo possível na árvore de consulta

Passo 3: Reordene os nós folhas de forma que as seleções mais restritivas sejam executadas primeiro

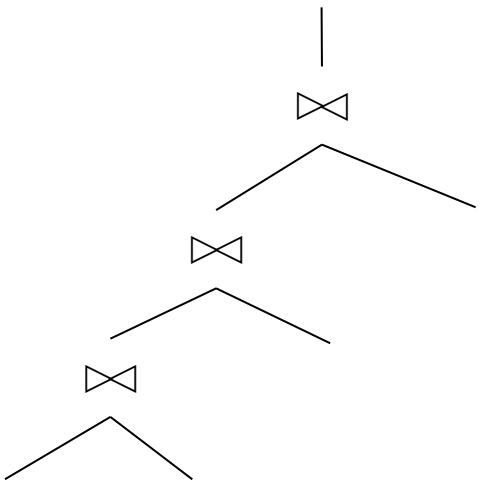
Passo 4: Combine as operações de produto cartesiano com seleções subseqüentes (formando junções)

Passo 5: Quebre e mova as projeção o mais baixo possível na árvore e crie novas projeção quando necessário

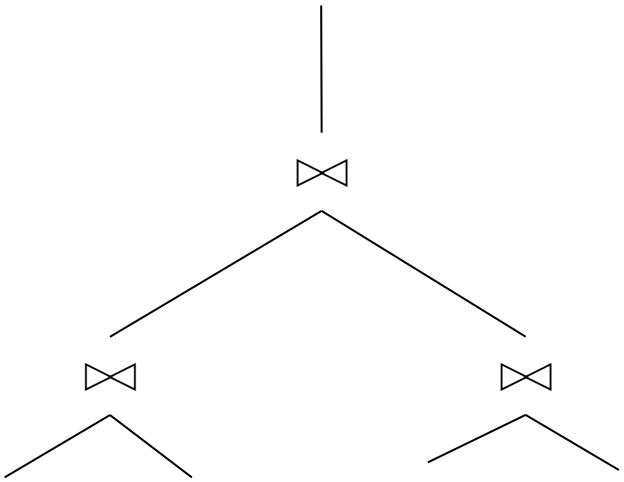
Passo 6: Identifique subárvore que representam grupos de operações que podem ser executadas em um único algoritmo

Plano de Consulta

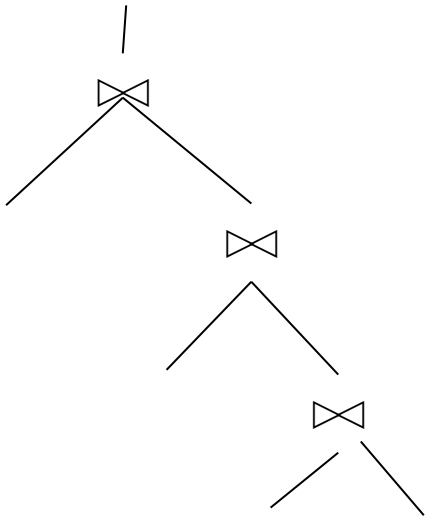
Dada a consulta: `select * from R natural join S natural join T natural join V;`



Left-deep tree



Bushy-deep tree



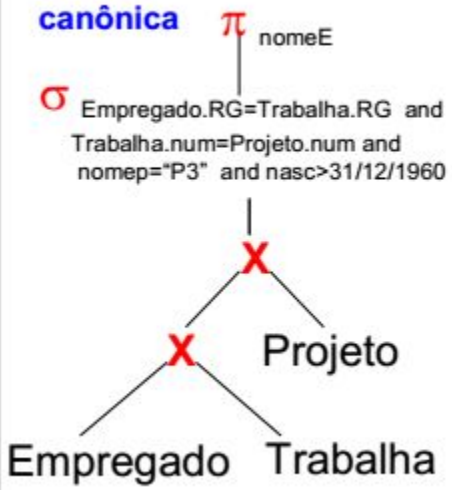
Right-deep tree

Plano de Consulta- baseado em heurística

Exemplo:
EMPREGADO (nomeE, RG, nasc, sexo, salário, RG-sup, Dept)
TRABALHA (RG, num, horas)
PROJETO (nomep, num, Dept)

$$\pi_{nomeE}((\sigma_{Nasc > 31/12/1960}(Empregado)) * (Trabalha)) * \sigma_{Nomep="P3"}(Projeto))$$

Árvore canônica



Plano de Consulta- baseado em heurística

Desenhe a árvore canônica e a corresponde otimizada, utilizando as regras heurísticas e equivalência algébrica

Exemplo:

EMPREGADO (nomeE, RG, ender, nasc, sexo, salário, RG-sup, Dept)

DEPARTAMENTO (nomeD, Dept, RG-gerente, Início-ger)

TRABALHA (RG, num,Horas)

PROJETO (nome, núm, Dept)

a) $\pi_{nomeE, ender}$ (($\sigma_{nomeD="pesquisa"}$ (Departamento)) $\bowtie$ (Empregado))

b) $\pi_{nomeE, nomeE}$ (($\sigma_{RG\text{-}sup=RG}$ (Empregado)) $\bowtie$ (Empregado))

c) π_{nomeE} ($\sigma_{nomeD="pesqueisa"}$ (Departamento) $\bowtie$ $\sigma_{salário\ >3000}$ (Empregado) $\bowtie$ $\sigma_{nproj=4}$ (Trabalha))

Dicionário de Dados

- Grupos de tabelas que armazenam dados sobre os dados (metadados)
  - Tabelas do banco
    - Nome lógico, organização
    - Atributos (nome, tipo, tamanho)
    - Chaves e Restrições
  - Índices
    - Estrutura (B+, hash...)
    - Atributos envolvidos
  - Visões
    - Nome
    - Construção

Dicionário de Dados

- Grupos de tabelas que armazenam dados sobre os dados (metadados)
  - Estatísticas
    - Cardinalidade da tabela: *NTuplas(T)*
    - Tamanho ocupado em páginas/blocos : *NPages(T)*
    - Cardinalidade de índices: *NKeys(I_T)*
    - Tamanho do índice: *INPages(I_T)* – para B+ só as folhas
    - Altura do índice: número de nós de uma B+ (*IHeight(I_T)*)
    - Faixa de valores dos índices: menor *ILow(I_T)* e maior *IHigh(I_T)*.

Estimativa do Tamanho do Resultado

SELECT <lista de atributos>

FROM < lista de relações R1,...,Rk >

WHERE <cond1 ^ cond2 ^....^condn>

Número máximo de tuplas no resultado =

M1.M2....Mk, onde Mi = tamanho de Ri

Cláusula WHERE atua como um redutor desta estimativa

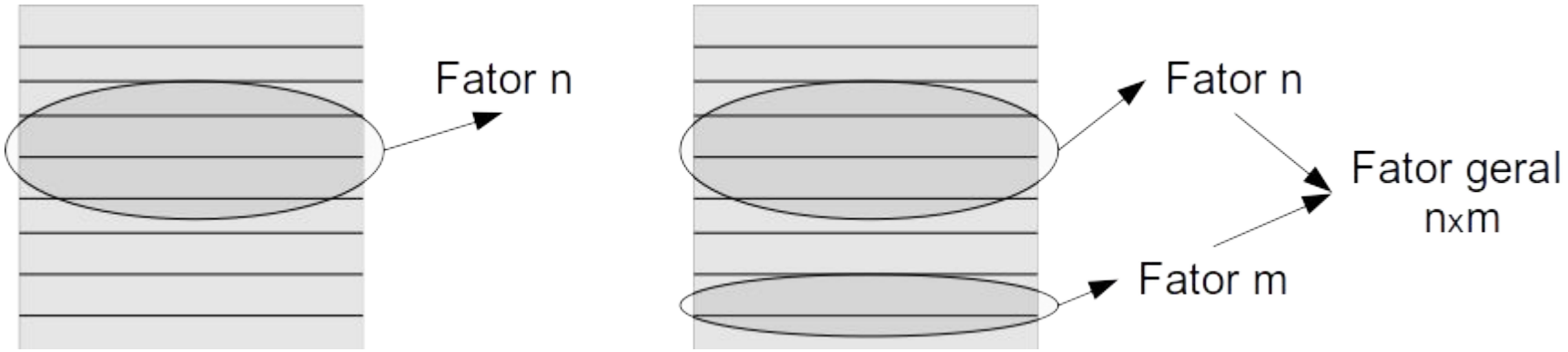
Cada condição do WHERE tem o seu fator de redução próprio

Seletividade de Caminhos

Fator de redução

Filtro que retorna apenas uma fração das tuplas de uma determinada tabela

Para várias condições, a fração de tuplas que satisfaçam todas as condições é aproximadamente a multiplicação de seus fatores de redução.



Fatores de Redução

R.A = valor

Fator de redução = 1/NKeys(I), caso exista um índice I com chave A para a relação R

Fator de redução = 1/10, caso contrário (ou utiliza-se estatísticas mantidas no catálogo sobre a distribuição dos valores dos atributos)

R.A = R.B

Fator de redução = 1/Max(NKeys(IA),NKeys(IB)) se existe índices IA e IB com chave A e B respectivamente.

Fator de redução = 1/NKeys(I) se somente um dos atributos é chave de um índice I

Fator de redução = 1/10 caso contrário.

Fatores de Redução

R.A > valor

Fator de redução = (High(I) – valor) / High(I) – Low(I) caso exista um índice I com chave A para a relação R

Fator de redução = fração < ½ caso não exista índice ou se o valor não é aritmético

Seletividade de Caminhos

- Exemplo
  - Índice hashing H (desc,bid,vid)
  - Predicado `desc='Verao' and bid=5 and vid=3`
  - NKeys(H)=150
  - NPages(Reserva)=100
  - Fórmula

$$NPages(Reserva) \times \frac{1}{NKeys(H)}$$

  - O fator de redução é 0,667.
  - Utilizando o índice para Reserva com, por exemplo, 100.000 tuplas, teríamos 66.700 descartadas com o índice

Seletividade de Caminhos

- Exemplo
  - Se tem-se um Índice (bid,vid) com a condição `bid=5 and vid=3`
  - Se é conhecido o número de valores distintos de `bid`, pode-se estimar o fator de redução para esta coluna.
    - Por exemplo, se existem 30 valores distintos, a redução seria 1/30.
  - Fazendo o mesmos para `vid`, tem-se o fator de redução de `vid`.
  - Multiplicando os dois temos o fator de redução total
  -

Planos de consultas aninhadas

```
SELECT S.sname
FROM Sailors S
WHERE S.rating = (SELECT MAX (S2.rating)
                  FROM Sailors S2)
```

- Subconsulta interna: **SELECT MAX (S2.rating)**
  FROM Sailors S2

  Executada uma única vez, produzindo um número **X**

- ```
  SELECT S.sname
  FROM Sailors S
  WHERE S.rating = X
  ```
  ```

O Que Faz Um Plano Ser Bom

- O planejador escolhe um plano (entre vários) baseado no seu custo estimado
- Assume: o I/O domina o custo de uma consulta (assim, seleciona o plano que requer menos I/O)
  - I/O randômico é mais caro que I/O sequencial nos hardwares modernos
- O I/O é estimado tentando predizer o tamanho dos resultados intermediários, usando as estatísticas do SGBD
  - Isso é uma ciência imperfeita (para não dizer algo pior)
- Distinguir entre custo de saída (I/O para a primeira tupla) e custo total

Princípios Gerais de Otimização

- O custo de um nó é em função de sua entrada: o número de linhas produzidas pelos nós filhos e a distribuição de seus valores:
  - Reorganizar os nós pode mudar todo o custo
  - Uma escolha pobre perto das folhas pode levar a um desastre
  - Aplicar os predicados mais cedo para reduzir o número de tuplas em resultados intermediários
- Mantenha em mente a ordenação: uma entrada ordenada pode ter planos mais baratos
- Planejar os joins de forma eficiente é fundamental

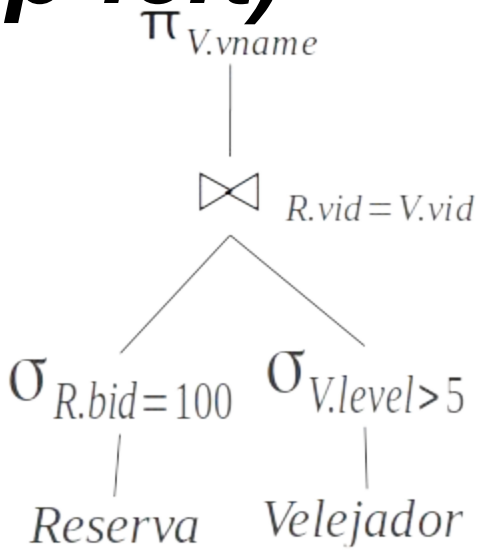
Algoritmo Planejador (PG)

- Conceitualmente, três fases:
  - Enumerar todos os planos disponíveis
  - Avaliar o custo de cada plano
  - Escolher o mais barato
- Naturalmente, executar essas fases não seria muito eficiente
- O algoritmo do projeto Sistema R é normalmente utilizado (algoritmo PD criado pela IBM em 1970s)
- Ideia básica: encontre bons planos para uma consulta simplificada com n joins
  - Encontre bons planos para n+1 joins, junte os planos (Repita)

Fases de um Plano (deep-left)

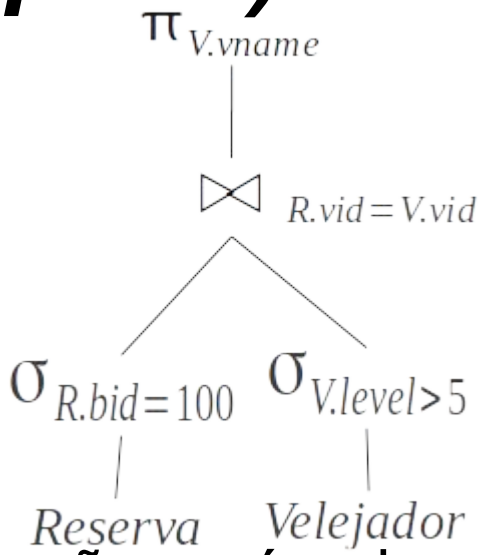
- Join com n tabelas na junção:
  - Passo 1: encontre o melhor plano para 1-relação para cada relação
  - Passo 2: encontre a melhor forma de fazer a junção para o plano 1-relação com outra relação (planos para 2-relações)
  - Passo 3: encontre a melhor forma de fazer a junção para o plano (n-1)-relação para a n-ésima relação.
- Para cada sub-conjunto de relações, guarde:
  - O plano mais barato para um scan completo, e
  - O plano mais barato considerando índices ou ordenação externas

Fases de um Plano (deep-left)



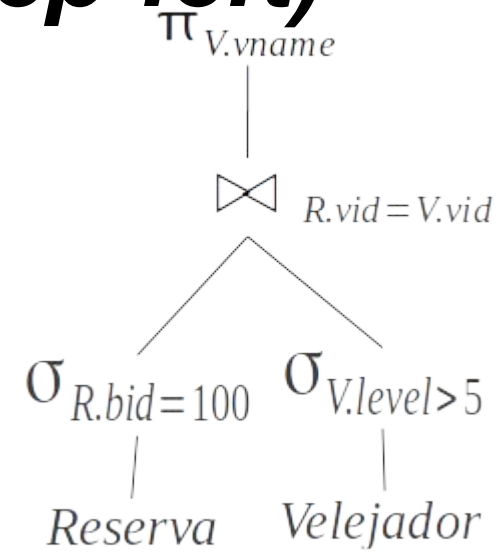
- Exemplos dos passos
  - Velejador B+ em level e Hash em vid
  - Reserva B+ em bid
- Passo 1:
  - Velejador
    - Escolhas: B+ para level > 5 ou scan na tabela
    - (se a seleção espera retornar várias tuplas ou índice não clusterizado, talvez scan seja mais barato)
    - Mantém o plano com B+ e o scan
  - Reserva
    - B+ em bid para bid=100 (+barato) e o scan
    -

Fases de um Plano (deep-left)



- Exemplos dos passos
  - Velejador B+ em level e Hash em vid
  - Reserva B+ em bid
- Passo 2:
  - Cada plano do passo 1 verificar qual relação será a da esquerda do Join
    - Reserva à esquerda: índice Hash pode ser utilizado para recuperar as tuplas de velejador
      - Satisfaz r.vid=valor de vid da tupla da relação da direita
    - Velejador à esquerda: utilizar um algoritmo de sort-merge
    - Escolha: reserva ⋈ velejador

Fases de um Plano (deep-left)



- Exemplos dos passos
  - Velejador B+ em level e Hash em vid
  - Reserva B+ em bid
- Resultado
  - Os predicados de seleção selecionados são B+ em level e B+ em bid
  - A junção é feita lendo-se primeiro reserva e, utilizando Hash em vid, buscando as tuplas de velejador

Algoritmo do Sistema R

- Agrupamentos e ordenações são feitos em etapas finais
- Considere planos left, right ou bushy trees (left é preferível)
- O número de planos considerados explode em relação ao crescimento do número de joins.
  - Para consultas com mais de 11 joins, o PostgreSQL utiliza um algoritmo genético (GEQO – Genetic Query Optimization)
    - Busca não exaustiva e não determinística para possíveis ordenações de joins utilizando left-deep tree.

Planejamento subconsultas

- FROM-lista:

```
select * from filme f, (select * from diretor d where
d.city='NYC') d where f.codd=d.codd;
--- otimizador pode converter
select * from filme f, diretor d
where f.codd=d.codd and d.city='NYC';
```

- A subquery foi puxada permitindo ao planejador todas as abordagens para otimizar join
- Integrando os qualificadores da subquery na consulta pai pode fazer com o otimizador otimize melhor a consulta pai

Planejamento subqueries

- Expressões
  - Produz planos aninhados através de chamadas recursivas do planejador
  - Subqueries sem relação com o pai necessitam ser executadas apenas um vez (melhor para o plano)

```
select * from filme f where f.codd=(select codd from
diretor where lname='Spielberg');
--- A subquery é independente
$var= select codd from diretor where lname='Spielberg';
select * from filme f where f.codd=$var;
```

- Se a subquery utiliza valores do pai, a mesma deva ser avaliada durante a execução do pai

Planejamento operadores de conjunto

- O planejamento é primitivo (trivial)
- Gera planos para as subqueries filhos, então adicione um nó para concatenar os conjuntos resultantes
- Alguns operadores exigem mais trabalhos
  - Union: ordena e remove duplicatas
  - Except, Intersect: ordena e remove duplicatas, produz o conjunto resultante através de um scan linear
- Não são consideradas nenhuma outra alternativa, assim os planos são simples

Passos Otimizador Consultas

Step	Entrada	Componente	Saída
1	String da consulta	Parser	Query Tree
2	Query Tree	Checker	Valid Query Tree
3	Valid Query Tree	View Expander	Valid Query Tree wo/ Views
4	Valid Query Tree wo/ Views	Gerador de plano lógico	Plano lógico
5	Plano lógico	Rewriter (heurística)	Plano lógico otimizado
6	Plano lógico	Gerador de plano físico (custo)	Plano físico
7	Plano físico	Gerador de código	Código executável
8	Código executável	Executor	Resposta

Melhorias Potenciais

- Difíceis
  - Estatísticas do banco para correlação entre colunas
  - Funções de otimização
  - Reescrever o GEQO
- Quase impossíveis:
  - Recolher as estatística online
  - O executor receber online o feedback do otimizador
  - Processamento paralelo em um servidor (uma consulta em múltiplos processadores concorrentes)
  - Distribuição do processamento da consulta (via rede)

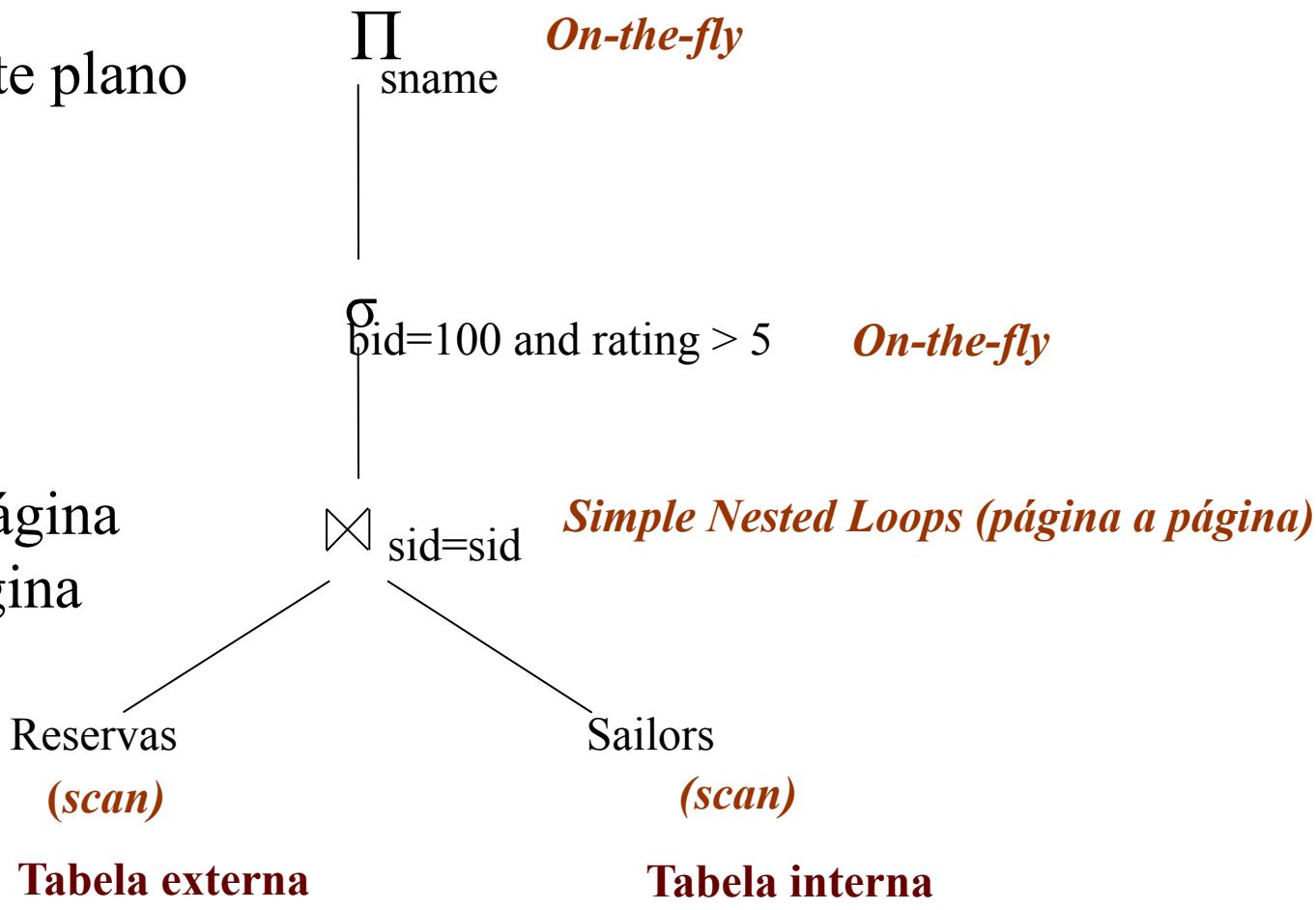
Cálculo de Custos de Planos de Execução

Exercício 1 :
Calcule o custo deste plano

R : 1000 páginas

S : 500 páginas

R: 100 tuplas por página
S: 80 tuplas por página



“Empurrando” **seleções** para baixo na árvore de execução

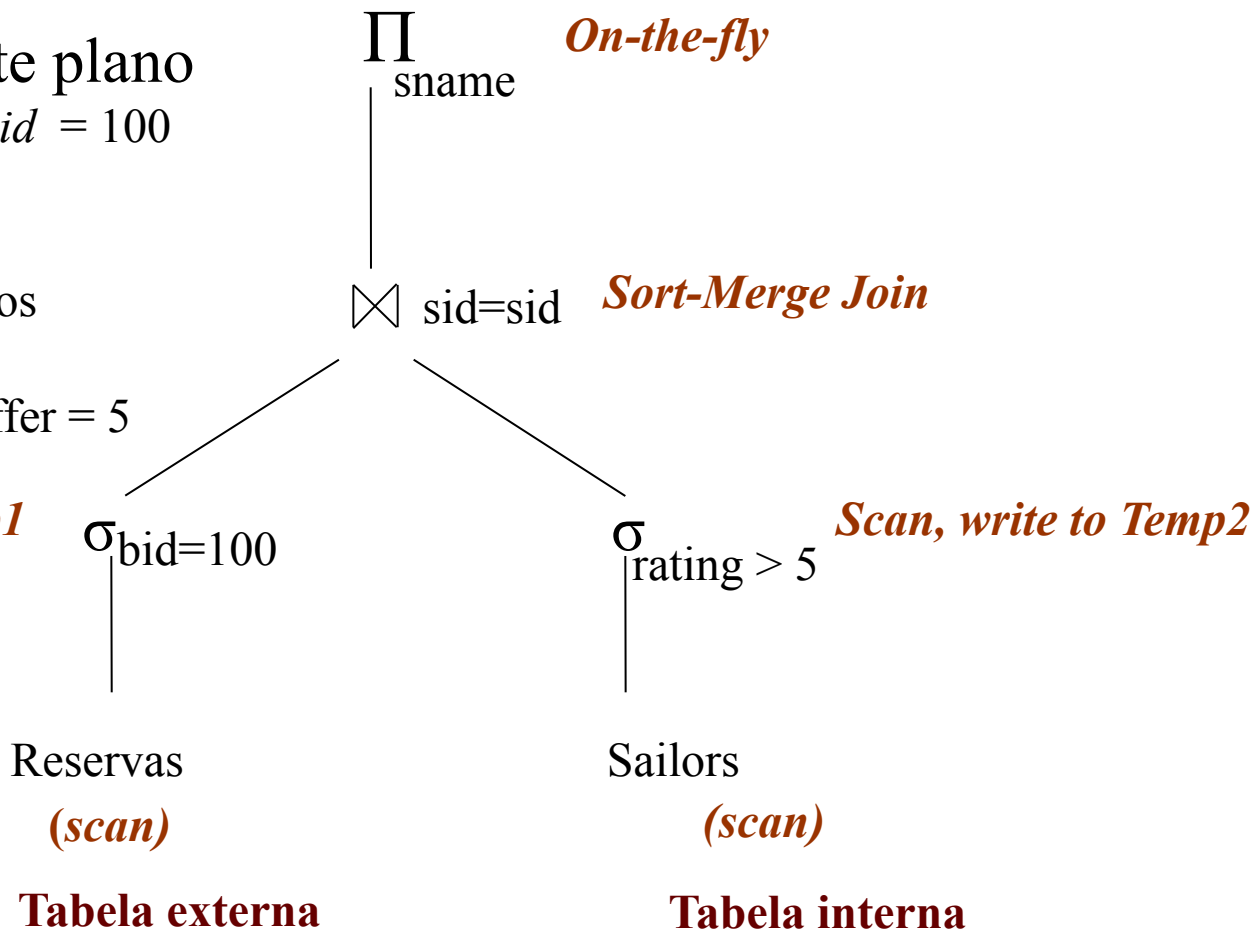
Exercício 2:

Calcule o custo deste plano

Número de valores *para bid* = 100

Rating varia de 1 a 10
Uniformemente distribuídos

Número de páginas no buffer = 5



“Empurrando” **seleções** para baixo na árvore de execução

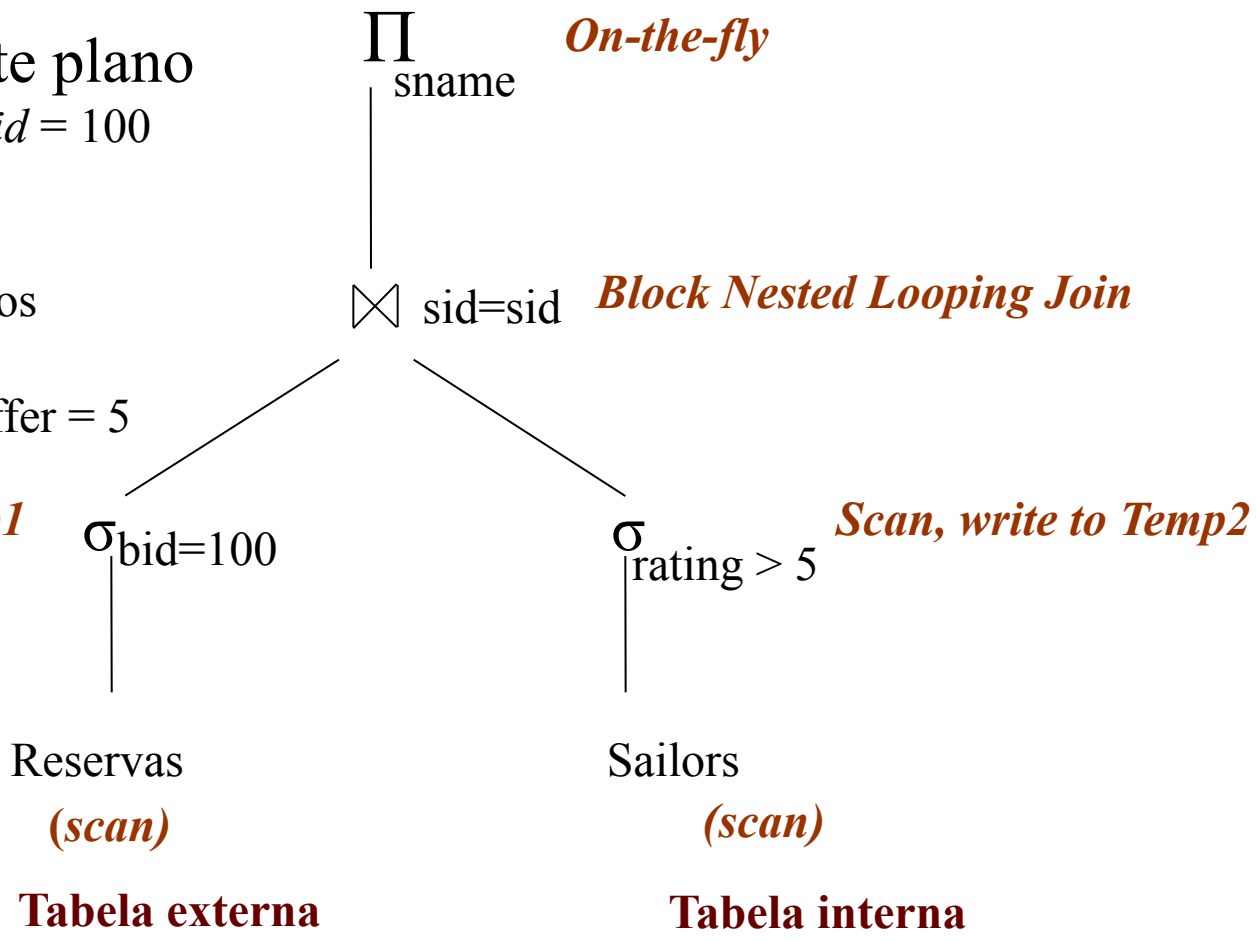
Exercício 3 :

Calcule o custo deste plano

Número de valores para *bid* = 100

Rating varia de 1 a 10
Uniformemente distribuídos

Número de páginas no buffer = 5



“Empurrando” **projeções** para baixo na árvore de execução

Exercício 4 :

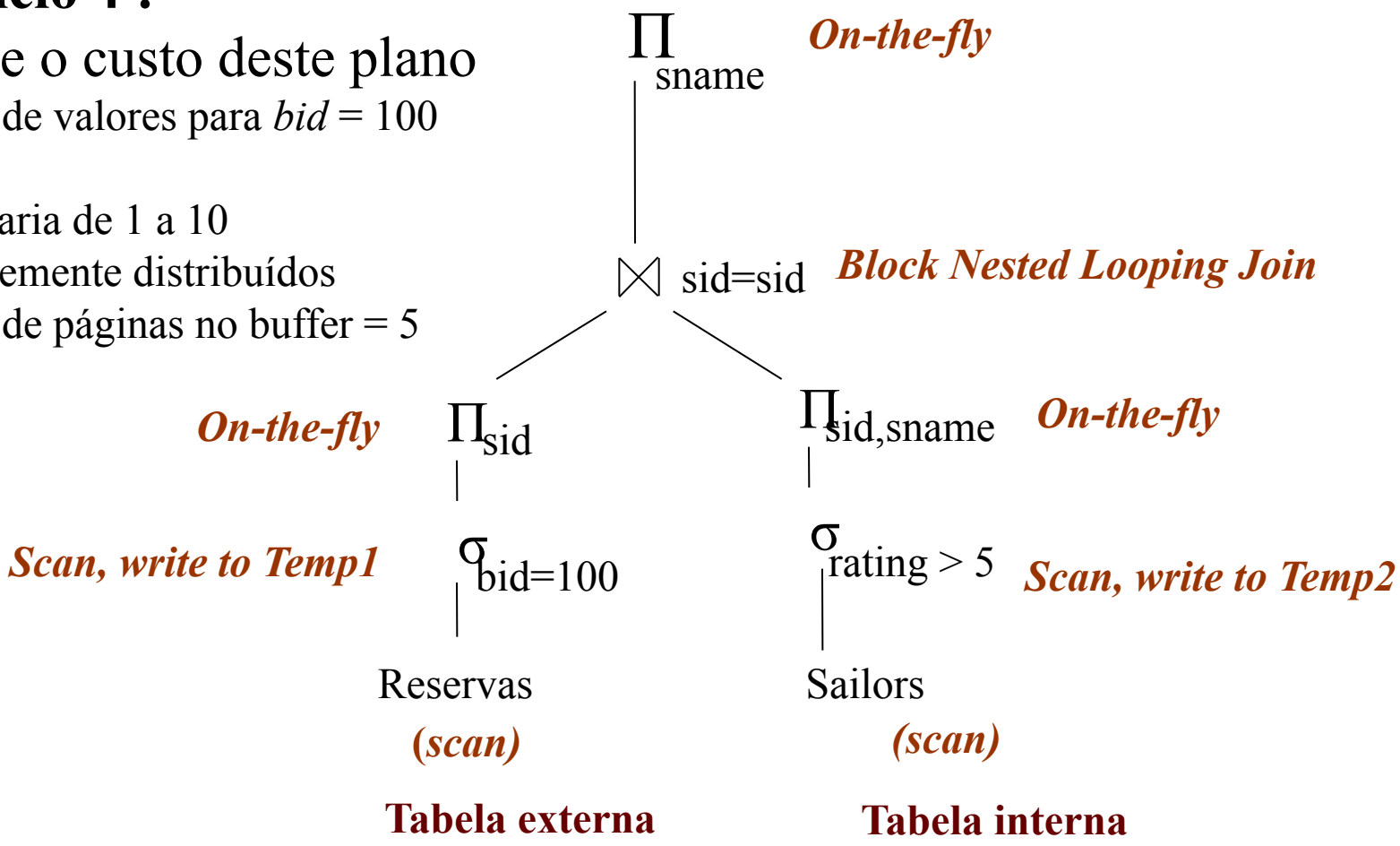
Calcule o custo deste plano

Número de valores para *bid* = 100

Rating varia de 1 a 10

Uniformemente distribuídos

Número de páginas no buffer = 5



Nem sempre “empurrar seleções abaixo do join é vantajoso”

Exercício 5 :

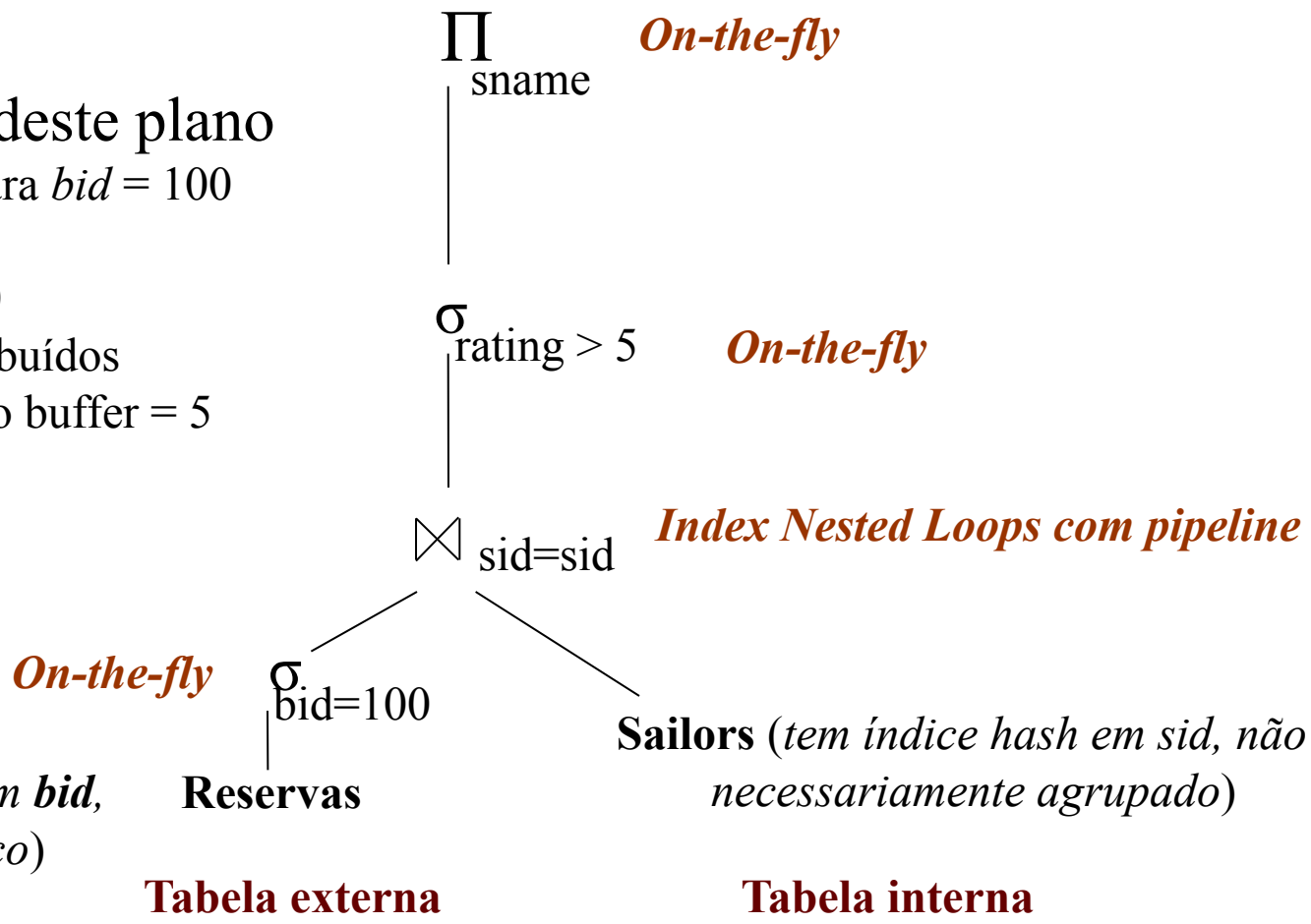
Calcule o custo deste plano

Número de valores para *bid* = 100

Rating varia de 1 a 10

Uniformemente distribuídos

Número de páginas no buffer = 5

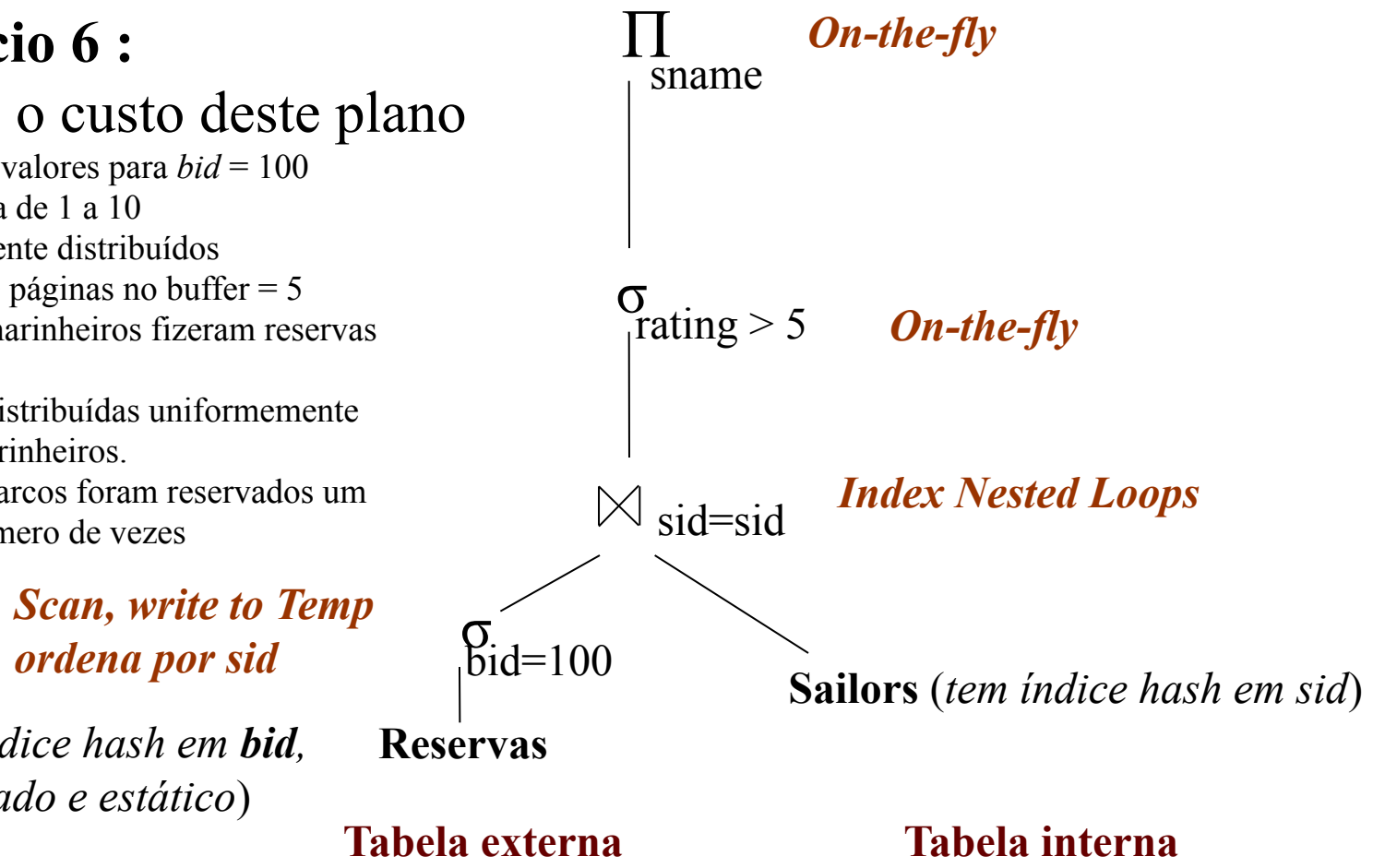


Nem sempre execuções em pipeline são mais vantajosas que as materializadas

Exercício 6 :

Calcule o custo deste plano

- Número de valores para *bid* = 100
- *Rating* varia de 1 a 10 uniformemente distribuídos
- Número de páginas no buffer = 5
- Todos os marinheiros fizeram reservas de barcos.
- Reservas distribuídas uniformemente entre os marinheiros.
- Todos os barcos foram reservados um mesmo número de vezes

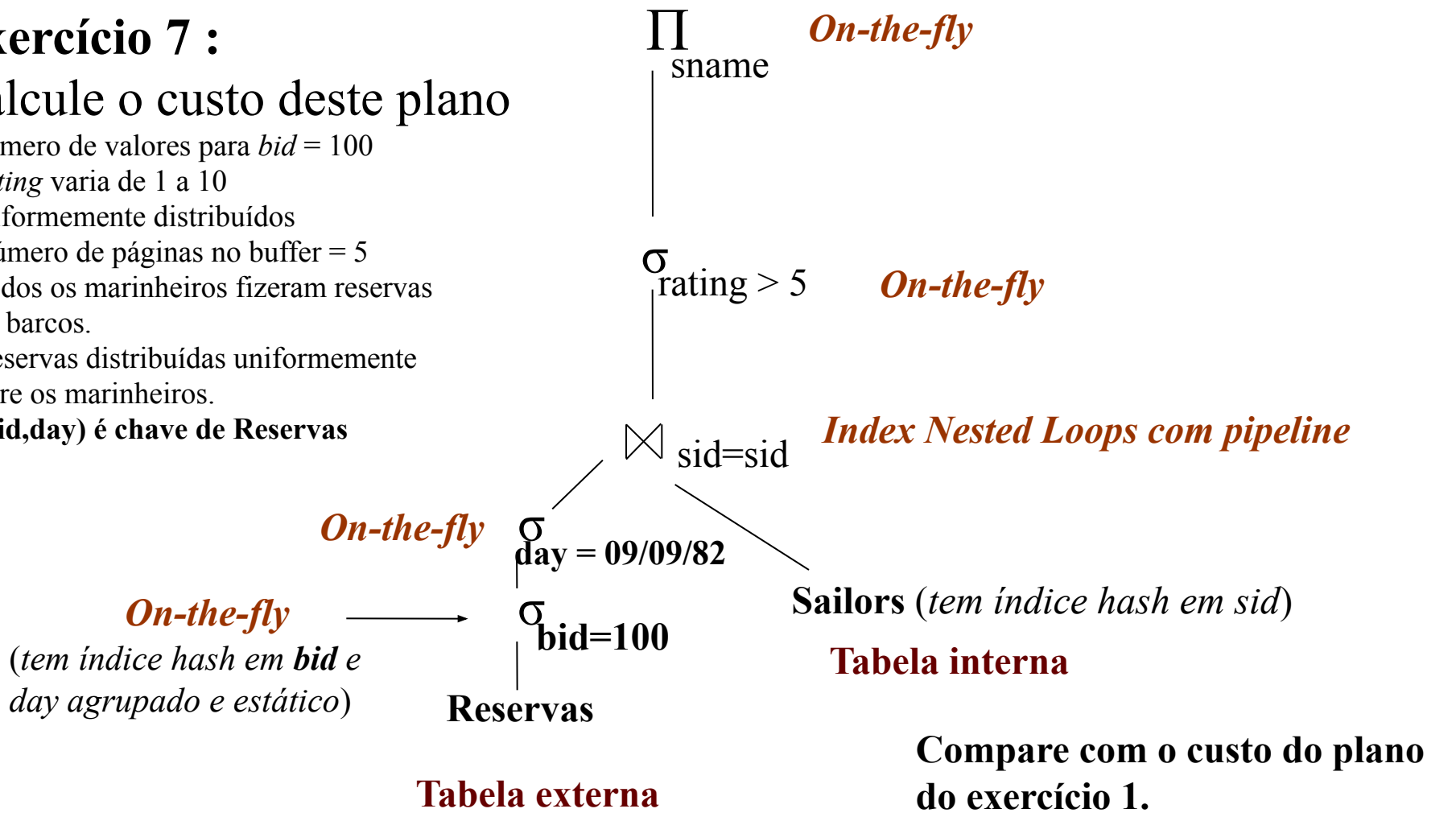


Seleção por atributo chave "empurrada" abaixo do Join é muito vantajosa.

Exercício 7 :

Calcule o custo deste plano

- Número de valores para *bid* = 100
- *Rating* varia de 1 a 10 uniformemente distribuídos
- Número de páginas no buffer = 5
- Todos os marinheiros fizeram reservas de barcos.
- Reservas distribuídas uniformemente entre os marinheiros.
- **(bid,day) é chave de Reservas**



Compare com o custo do plano do exercício 1.

Conclusões

- Consultas são as operações mais caras do SGBD
- Encontrar a melhor forma de executar uma consulta é um problema intratável em computação
- O dicionário de dados tem um papel importante no processamento de consultas
- Índices são essenciais
- O SGBD tenta resolver os problemas de otimização de forma automática