Processamento de Consultas

Simone Dominico
Orientador: Dr. Eduardo Cunha de Almeida
PPGINF - UFPR

Sumário

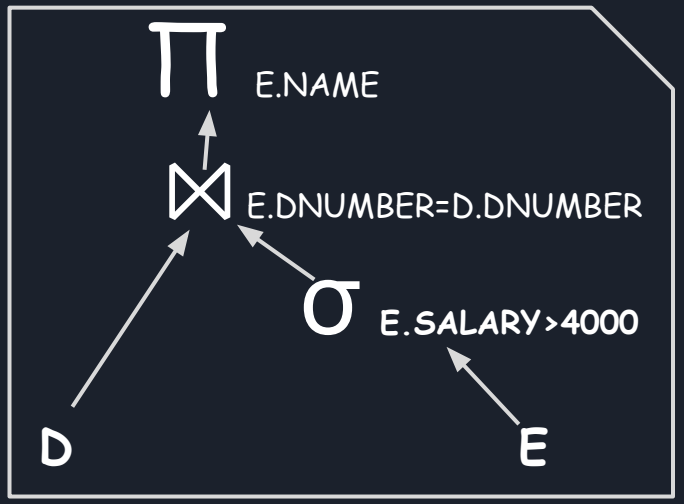
- Plano de Consulta;
- Modelos de processamento de Consultas;
- Metódos de Acesso;
  - Algoritmos de seleção;
  - Algoritmos de Junção;

Plano de Consulta

Plano de Consulta

- Os operadores são organizados em árvores;
- O fluxo de dados acontece das folhas para a raiz;
- A saída o nó raiz é o resultado da consulta;

```
SELECT E.NAME
FROM EMPLOYEE E, DEPARTMENT D
WHERE D.DNUMBER=E.DNUMBER
AND E.SALARY > 4000;
```



Modelos de Processamento

Modelos de Processamento

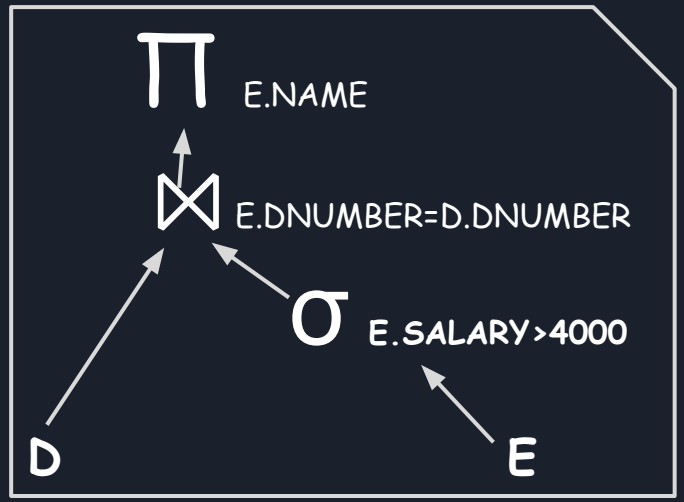
- Os modelos de processamento definem como o SGBD executa o plano de consulta.
- Três abordagens são:
  - Modelo de iteração;
  - Materialização;
  - Vetorização

Modelo de Iteração/Pipeline

- Modelo de execução mais comum usado por quase todos os SGBDs;
- Todo operador do plano de consulta implementa uma próxima função;
- Cada chamada para a próxima função, o operador retorna uma única tupla ou um marcador nulo se não tem mais tuplas;
- O operador implementa um loop que chama o próximo em seus filhos para recuperar as tuplas e depois processa-lás;
- Operadores como: join, subconsultas e order by são bloqueados até seus filhos retornarem todas as suas tuplas;
- Processamento top-down;

Modelo de Iteração/Pipeline - Exemplo

```
SELECT E.NAME
FROM EMPLOYEE E, DEPARTMENT D
WHERE D.DNUMBER=E.DNUMBER
AND E.SALARY > 4000;
```



For t in **child.Next()**:
emit(**projection**(t))

For t_1 in **left.Next()**:
buildHashTable(t_1)
For t_2 in **right.next()**:
If **probe**(t_2): **emit** ($t_1 \bowtie t_2$)

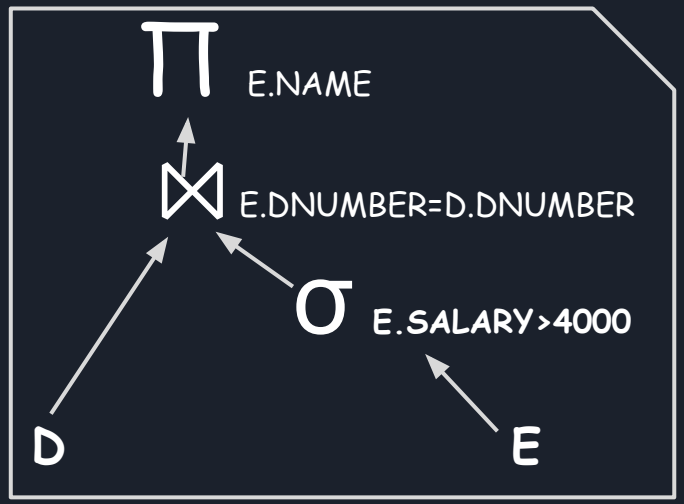
For t in **child.Next()**:
If **evalPred**(t): **emit**(t)

For t in **E**:
Emit (t)

For t in **D**:
Emit (t)

Modelo de Iteração/Pipeline - Exemplo

```
SELECT E.NAME
FROM EMPLOYEE E, DEPARTMENT D
WHERE D.DNUMBER=E.DNUMBER
AND E.SALARY > 4000;
```



For t in **child.Next()**:
emit(projection(t))

1

For t_1 in **left.Next()**:
buildHashTable(t_1)
For t_2 in **right.next()**:
If **probe**(t_2): **emit** ($t_1 \bowtie t_2$)

2

For t in **child.Next()**:
If **evalPred**(t): **emit**(t)

4

3

For t in **E**:
Emit (t)

For t in **D**:
Emit (t)

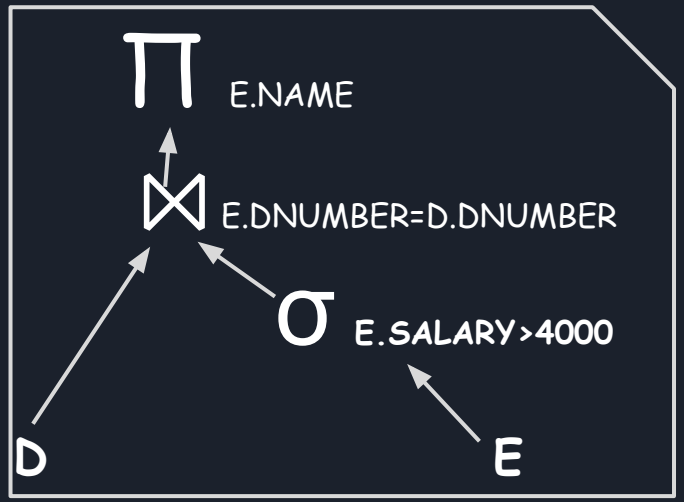
5

Materialização

- Cada operador processa sua entrada de uma só vez e emite a saída de uma só vez;
- O operador “materializa” a saída como resultado único;
- Processamento de baixo para cima (bottom-Up);
- Melhor para OLTP.

Materialização - Exemplo

```
SELECT E.NAME
FROM EMPLOYEE E, DEPARTMENT D
WHERE D.DNUMBER=E.DNUMBER
AND E.SALARY > 4000;
```




```
Out = { }
For t in child.Output:
out.add(projection(t))
```

```
For t1 in  left.Output():
        buildHashTable(t1)
For t2 in right.Output():
     If probe(t2): out.add (t1 ⋈t2)
```

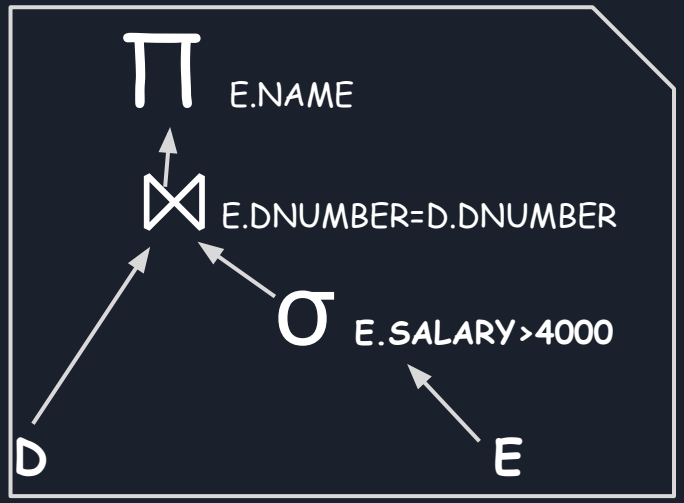
```
For t in child.Next():
If evalPred(t): out.add(t)
```

```
Out = { }
For t in E:
out.add (t)
```

```
Out = { }
For t in D:
out.add (t)
```

Materialização - Exemplo

```
SELECT E.NAME
FROM EMPLOYEE E, DEPARTMENT D
WHERE D.DNUMBER=E.DNUMBER
AND E.SALARY > 4000;
```



5

```
Out = { }
For t in child.Output:
out.add(projection(t))
```

```
For t1 in  left.Output():
        buildHashTable(t1)
For t2 in right.Output():
     If probe(t2): out.add (t1 ⋈t2)
```

4

3

```
For t in child.Next():
If evalPred(t): out.add(t)
```

```
Out = { }
For t in E:
out.add (t)
```

1

2

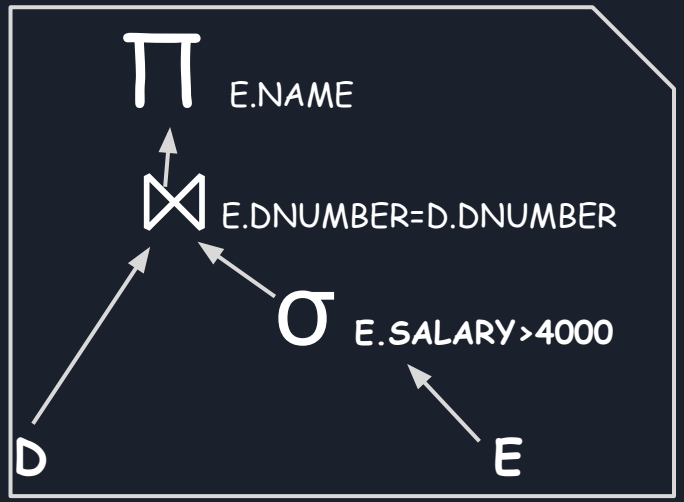
```
Out = { }
For t in D:
out.add (t)
```

Vetorização

- Semelhante ao iterador, cada operador implementa uma próxima função;
- Cada operador emite um lote de dados, não apenas uma única tupla;
- Reduz o número de invocações por operador;
- Bom para consultas OLAP;

Vectorização - Exemplo

```
SELECT E.NAME
FROM EMPLOYEE E, DEPARTMENT D
WHERE D.DNUMBER=E.DNUMBER
AND E.SALARY > 4000;
```




```
Out = { }
For t in child.Output:
out.add(projection(t))
If |out|>n: emit(out)
```

```
For t1 in left.Output():
      buildHashTable(t1)
For t2 in right.Output():
    If probe(t2): out.add (t1 ⋈t2)
    If |out|>n: emit(out)
```

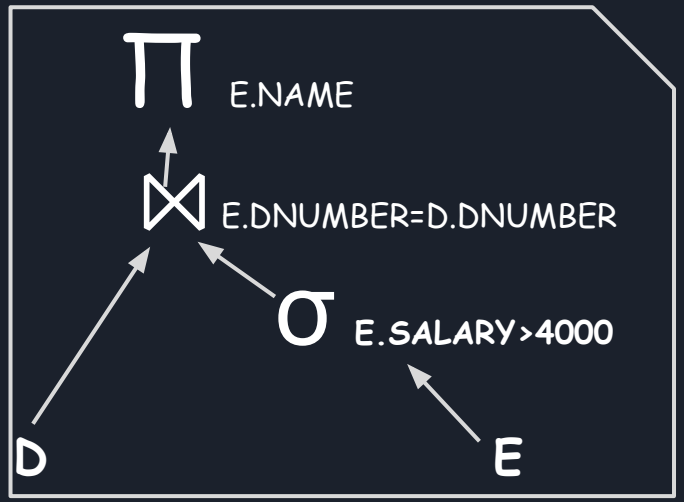
```
For t in child.Next():
If evalPred(t): out.add(t)
If |out|>n: emit(out)
```

```
Out = { }
For t in E:
out.add (t)
If |out|>n: emit(out)
```

```
Out = { }
For t in D:
out.add (t)
If |out|>n: emit(out)
```

Vectorização - Exemplo

```
SELECT E.NAME
FROM EMPLOYEE E, DEPARTMENT D
WHERE D.DNUMBER=E.DNUMBER
AND E.SALARY > 4000;
```




```
Out = { }
For t in child.Output:
out.add(projection(t))
If |out|>n: emit(out)
```

1

```
For t1 in left.Output():
      buildHashTable(t1)
For t2 in right.Output():
    If probe(t2): out.add (t1 ⋈t2)
    If |out|>n: emit(out)
```

2

```
Out = { }
For t in E:
out.add (t)
If |out|>n: emit(out)
```

3

```
For t in child.Next():
If evalPred(t): out.add(t)
If |out|>n: emit(out)
```

4

```
Out = { }
For t in D:
out.add (t)
If |out|>n: emit(out)
```

5

Métodos de Acesso

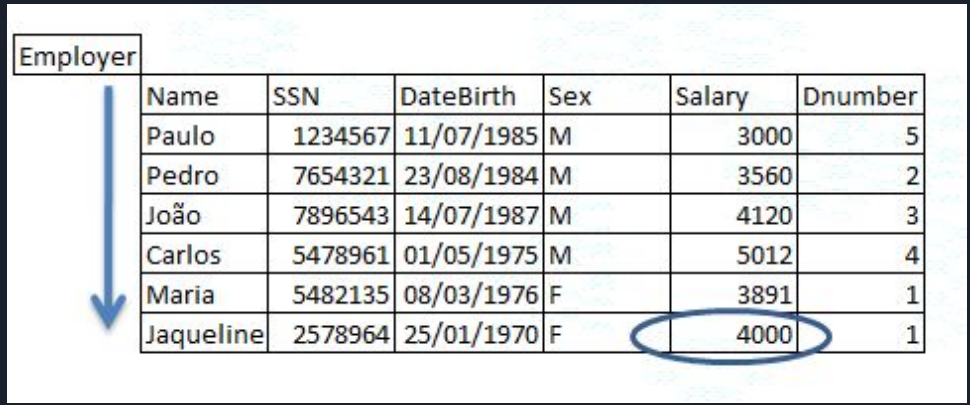
Métodos de Acesso

- Um método de acesso é como o SGBD consegue acessar os dados em uma tabela;
- Algoritmos dependem
  - da existência de índices;
  - das condições de seleção;
- • Métodos para seleção simples
  - varredura de arquivos (i.e., file scan);
- busca linear e binária
  - varredura de índices (i.e., index scan);
- busca baseada em índice primário, secundário e cluster

Seleção

Busca Linear

- Faz a iteração sobre todas as páginas que contém dados da tabela, verificando o predicado;



Name	SSN	DateBirth	Sex	Salary	Dnumber
Paulo	1234567	11/07/1985	M	3000	5
Pedro	7654321	23/08/1984	M	3560	2
João	7896543	14/07/1987	M	4120	3
Carlos	5478961	01/05/1975	M	5012	4
Maria	5482135	08/03/1976	F	3891	1
Jaqueline	2578964	25/01/1970	F	4000	1

Busca Binária

- Se o arquivo é ordenado em um atributo e a condição de seleção é uma comparação de igualdade neste atributo, podemos usar uma busca binária para localizar os registros que satisfazem a seleção.

```
SELECT NAME
FROM EMPLOYEE
WHERE
SALARY=4000;
```

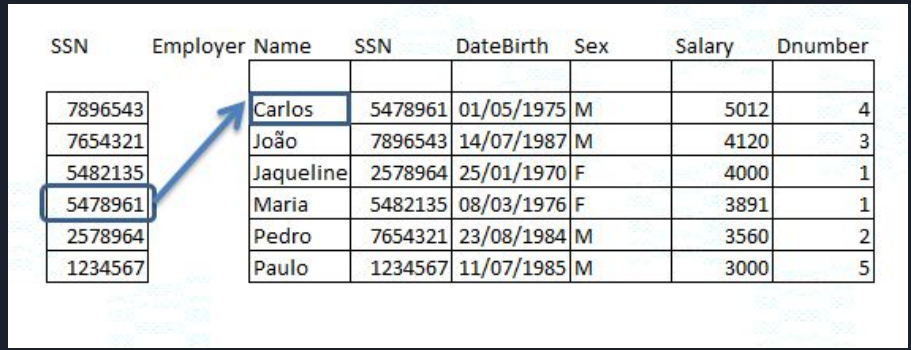
Employer

Name	SSN	DateBirth	Sex	Salary	Dnumber
Paulo	1234567	11/07/1985	M	3000	5
Pedro	7654321	23/08/1984	M	3560	2
Maria	5482135	08/03/1976	F	3891	1
Jaqueline	2578964	25/01/1970	F	4000	1
João	7896543	14/07/1987	M	4120	3
Carlos	5478961	01/05/1975	M	5012	4

Varredura de índice

- algoritmos de busca que usam um índice (condição de seleção precisa ser sobre chave de busca do índice).
  - Primário, se a condição de seleção envolve uma comparação de igualdade em um índice chave com um índice primário:

```
SELECT NAME
FROM EMPLOYEE
WHERE
SSN=5478961;
```



SSN
7896543
7654321
5482135
5478961
2578964
1234567

Employer Name	SSN	DateBirth	Sex	Salary	Dnumber
Carlos	5478961	01/05/1975	M	5012	4
João	7896543	14/07/1987	M	4120	3
Jaqueline	2578964	25/01/1970	F	4000	1
Maria	5482135	08/03/1976	F	3891	1
Pedro	7654321	23/08/1984	M	3560	2
Paulo	1234567	11/07/1985	M	3000	5

Varredura de índice

- algoritmos de busca que usam um índice (condição de seleção precisa ser sobre chave de busca do índice).
  - Primário, para recuperar múltiplos registros;
  - Comparação >, < , <=, >=

```
SELECT E.NAME
FROM EMPLOYEE E
WHERE E.DNUMBER>3;
```

Varredura de índice

- algoritmos de busca que usam um índice (condição de seleção precisa ser sobre chave de busca do índice).
  - Utilização de um índice de cluster para recuperar múltiplos registros;
  - Envolve uma comparação de igualdade em um atributo não chave;

```
SELECT E.NAME
FROM EMPLOYEE E
WHERE
E.SALARY = 4000;
```

Varredura de índice

- algoritmos de busca que usam um índice (condição de seleção precisa ser sobre chave de busca do índice).
  - Utilização de um índice secundário em uma comparação de igualdade;
  - Pode recuperar um único registro se o campo de indexação é chave ou múltiplos registros se o campo de indexação não for chave;

Varredura de índice

- Seleção conjuntiva usando índice individual:
  - Se um atributo possui um caminho de acesso que permita o uso de busca binária ou índice secundário;
  - Usa a condição para recuperar os registros;
  - Depois verifica se cada atributo recuperado satisfaz a condição;

```
SELECT E.NAME
FROM EMPLOYEE E
WHERE E.DNUMBER=3
AND
E.SALARY > 4000 AND
E.sex= "F";
```

Varredura de índice

- Seleção conjuntiva usando índice composto:
  - Se dois ou mais atributos estiverem envolvidos em condições de igualdade na condição conjuntiva e houver um índice composto para combinação dos campos então usa-se o índice diretamente;

```
SELECT E.NAME
FROM EMPLOYEE E
WHERE E.DNUMBER=3
AND
E.SALARY > 4000 AND
E.sex= "F";
```

Varredura de índice

- Seleção conjuntiva por meio da interseção de registros:
  - se índices secundários (ou outros caminhos de acesso) estiverem disponíveis para mais de um dos campos envolvidos nas condições simples de uma condição conjuntiva;
  - se apenas algumas das condições possuir índices secundários, cada registro recuperado será posteriormente testado para determinar se ele satisfaz as condições restantes.

Junções

Algoritmos de Junção

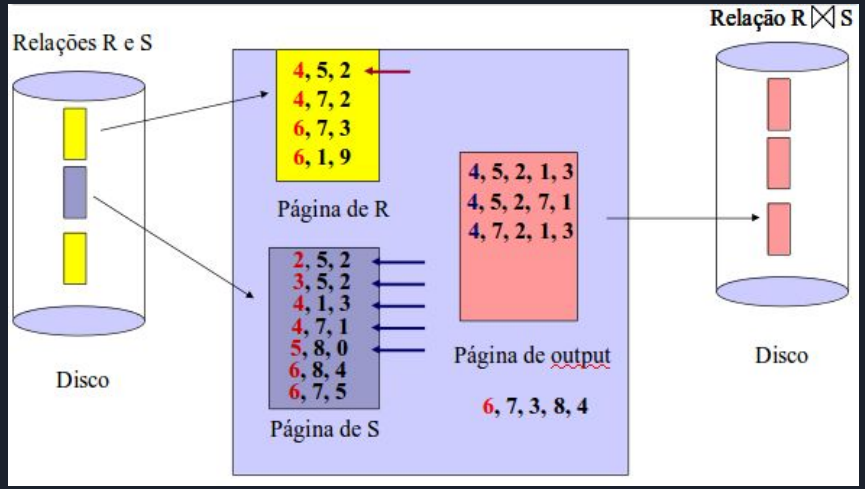
- Junção de laço aninhado (nested loop) -força bruta
  - Para cada registro t em R (laço externo):
    - Recupere cada registro s em S
    - Teste se os dois registros satisfazem a condição de junção;

Algoritmos de Junção

- Junção de laço único (single loop)
  - Usando uma estrutura de acesso para recuperar os registros correspondentes à junção:
    - Se existir um índice para um dos dois atributos de junção (por ex, B de S)
      - recupere cada registro t em R (um por vez)
        - Use a estrutura de acesso para recuperar os registros em S que satisfaçam a condição de junção

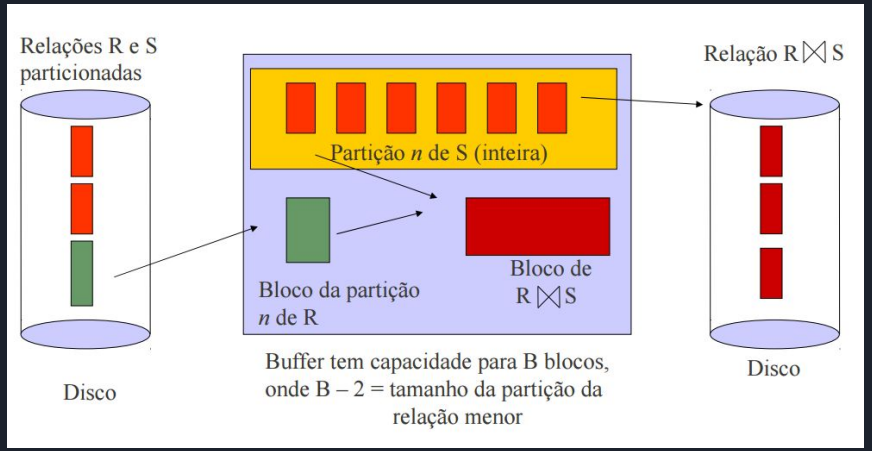
Algoritmos de junção

- Ordenação e intercalação:
  - Se a primeira é a segunda tabela estão fisicamente ordenadas, pode percorrer os registros simultaneamente e recuperar os dados que atendem a junção.



Algoritmos de junção

- Hash:
  - Na primeira fase os registros das duas tabelas são separados em arquivos menores utilizando a mesma função hash.
  - Na segunda fase junta os registros correspondentes.



Referências e Material de Apoio

Pavlo, Andy. Lecture 10 - Query Processing. Computer Science Dept. Carnegie Mellon Univ. Disponível em: http://15445.courses.cs.cmu.edu/fall2017/slides/10-queryprocessing.pdf

Silberschatz, A. Sistema de Banco de Dados. 6 ed.,Elsevier, 2012. Capítulo 13 – Processamento da Consulta;