# Processamento e Otimização de Consulta
# Implementando a Operação SELECT

Profa. Dra. Maria Madalena Dias



## Processamento e Otimização de Consulta
## Implementando a Operação SELECT

- Operações

(OP1): $\sigma_{SSN = '123456789'}$ (EMPLOYEE)

(OP2): $\sigma_{DNUMBER > 5}$ (DEPARTMENT)

(OP3): $\sigma_{DNO = 5}$ (EMPLOYEE)

(OP4): $\sigma_{DNO= 5 \text{ AND } SALARY>3000 \text{ AND } SEX= 'F'}$ (EMPLOYEE)

(OP5): $\sigma_{ESSN= '123456789' \text{ AND } PNO= 10}$ (WORK_ON)



## Implementando a Operação SELECT

- Métodos de Busca para Seleção Simples
  - S1. Busca Linear (força bruta): Recupera todo registro no arquivo e testa quando seus valores de atributo satisfazem a condição de seleção.
  - S2. Busca Binária: Se a condição de seleção envolve uma comparação de igualdade sobre um atributo chave no qual o arquivo é ordenado, a busca binária - que é mais eficiente do que busca linear - pode ser usada. Um exemplo é OP1 se SSN é o atributo de ordenação para o arquivo EMPLOYEE.



## Implementando a Operação SELECT
### Métodos de Busca para Seleção Simples

- S3. Usando um Índice Primário (ou chave hash): Se a condição de seleção envolve uma comparação de igualdade sobre um atributo chave com um índice primário (ou chave hash) - por ex., SSN = '123456789' em OP1 - usa o índice primário (ou chave hash) para recuperar o registro. Note que esta condição recupera um único registro (quando muito).



## Implementando a Operação SELECT
### Métodos de Busca para Seleção Simples

- S4. Usando um Índice Primário para Recuperar Múltiplos Registros: Se a condição de comparação é >, >=, < ou <= sobre um campo chave com um índice primário - por ex., DNUMBER > 5 em OP2 - use o índice para encontrar o registro satisfazendo a condição de igualdade correspondente (DNUMBER = 5), então recupere todos os registros subseqüentes no arquivo (ordenado). Para a condição DNUMBER < 5, recupere todos os registros precedentes.



## Implementando a Operação SELECT
### Métodos de Busca para Seleção Simples

- S5. Usando um Índice de Agrupamento para Recuperar Múltiplos Registros: Se a condição de seleção envolve uma comparação de igualdade sobre um atributo não chave com um índice de agrupamento - por ex., DNO = 5 em OP3 - use o índice para recuperar todos os registros satisfazendo a condição.

## Implementando a Operação SELECT
### Métodos de Busca para Seleção Simples

- S6. Usando um Índice Secundário (árvore $B^+$) sobre uma Comparação de Igualdade: Este método de busca pode ser usado para recuperar um único registro se o campo de indexação é uma chave (tem valores únicos) ou para recuperar múltiplos registros se o campo de indexação não é uma chave. Isto pode também ser usado para comparações envolvendo >, >=, < ou <=.



## Implementando a Operação SELECT
### Métodos de Busca para Seleção Complexa

- S7. Seleção Conjuntiva Usando um Índice Individual: Se um atributo, envolvido em qualquer condição única simples na condição conjuntiva, tem um caminho de acesso que permite o uso de um dos Métodos S2 a S6, use aquela condição para recuperar os registros e então cheque quando cada registro recuperado satisfaz as condições simples restantes na condição conjuntiva.



## Implementando a Operação SELECT
### Métodos de Busca para Seleção Complexa

- S8. Seleção Conjuntiva Usando um Índice Composto: Se dois ou mais atributos estão envolvidos em condições de igualdade na condição conjuntiva e um índice composto (ou estrutura hash) existe sobre os campos combinados - por ex., se um índice foi criado sobre a chave composta (ESSN, PNO) do arquivo WORK_ON para OP5 - nós podemos usar o índice diretamente.



## Implementando a Operação SELECT
### Métodos de Busca para Seleção Complexa

- S9. Seleção Conjuntiva por Interseção de Ponteiros de Registro: Se índices secundários (ou outros caminhos de acesso) estão disponíveis sobre mais de um dos campos envolvidos em condições simples na condição conjuntiva, e se os índices incluem ponteiros de registro, então cada índice pode ser usado para recuperar o conjunto de ponteiros de registro que satisfazem a condição individual.



## Implementando a Operação SELECT
### Métodos de Busca para Seleção Complexa

- Cont. S9. A interseção destes conjuntos de ponteiros de registro dá os ponteiros de registro que satisfazem a condição conjuntiva, que são então usadas para recuperar aqueles registros diretamente. Se somente algumas das condições têm índices secundários, cada registro recuperado é adicionalmente testado para determinar quando ele satisfaz as condições restantes.



## Implementando a Operação SELECT

- Estratégias para a seleção de uma condição única:
  - Checar se existe um caminho de acesso sobre um atributo envolvido na condição;
  - Se existe um caminho, então o método correspondente àquele caminho de acesso é usado.
  - Senão, a abordagem de busca linear força bruta do método S1 pode ser usada.

## Implementando a Operação SELECT

- Otimização de Consulta e SELECT
  - Otimização de consulta para uma operação de SELECT é necessária, principalmente, para condição de seleção conjuntiva sempre que mais de um dos atributos envolvidos nas condições tenham um caminho de acesso.
  - O otimizador deveria escolher o caminho de acesso que recupera a minoria dos registros de forma mais eficiente, estimando os diferentes custos e escolhendo o método com o menor custo estimado.



## Implementando a Operação SELECT

- Escolha entre múltiplas condições simples:
  - Considerar a seletividade de cada condição. A seletividade é definida como a proporção do número de registros (tuplas) que satisfazem a condição em relação ao total do número de registros do arquivo (relação), e portanto, é um número entre zero e 1.
    - Zero seletividade significa nenhum registro satisfaz a condição e 1 significa que todos os registros satisfazem a condição



## Implementando a Operação SELECT

  - Estimativas de seletividades são mantidas freqüentemente no catálogo do SGBD e são usadas pelo otimizador, embora não sejam exatas e possam estar incompletas.
- Condição de Disjunção:
  - São mais difíceis de processar e otimizar do que condição de seleção conjuntiva, porque os registros que satisfazem a condição de disjunção são a união dos registros que satisfazem as condições individuais.



## Implementando a Operação JOIN

- Operações:

  (OP6): EMPLOYEE $|\text{X}|_{\text{DNO=DNUMBER}}$ DEPARTMENT

  (OP7): DEPARTMENT $|\text{X}|_{\text{MGRSSN=SSN}}$ EMPLOYEE
- Métodos para Implementar Junções:
  - J1. Junção *Nested-loop* (força bruta): Para cada registro *t* em *R* (loop externo), recupere todo registro *s* de *S* (loop interno) e teste quando os dois registros satisfazem a condição de junção *t*[A] = *s*[B].



## Implementando a Operação JOIN
### Métodos para Implementar Junções

  - J2. Junção Loop Único (usando uma estrutura de acesso para recuperar registros semelhantes): Se um índice (ou chave hash) existe para um dos dois atributos de junção - *B* de *S* - recupere cada registro *t* em *R* (loop único), e então use a estrutura de acesso para recuperar diretamente todos os registros semelhantes *s* de *S* que satisfaçam *s*[B] = *t*[A].



## Implementando a Operação JOIN
### Métodos para Implementar Junções

  - J3. Junção *Sort-merge*: Se os registros de *R* e *S* estão fisicamente classificados (ordenados) pelo valor dos atributos de junção A e B, respectiva-mente, nós podemos implementar a junção na maneira mais eficiente possível. Ambos os arquivos são explorados concorrentemente na ordem dos atributos de junção, casando os registros que têm os mesmos valores para A e B. Se os arquivos não estão classificados, eles podem ser classificados primeiro usando classificação externa.

## Implementando a Operação JOIN
### Métodos para Implementar Junções

- Cont. J3. Neste método, pares de blocos de arquivos são copiados nos *buffers* da memória em ordem e os registros de cada arquivo são explorados somente uma vez para casar com o outro arquivo, a menos que A e B não sejam atributos chave, no qual o método precisa ser ligeiramente modificado.



## Implementando a Operação JOIN
### Métodos para Implementar Junções

- J4. Junção *Hash*: Os registros dos arquivos *R* e *S* são copiados para um único arquivo *hash*, usando a mesma função *hashing* sobre os atributos de junção *A* de *R* e *B* de *S* como chaves *hash*. Na primeira fase, fase de particionamento, os registros do arquivo com menos registros (*R*) são copiados para partições do arquivo *hash*, após serem calculadas suas chaves *hash*.



## Implementando a Operação JOIN
### Métodos para Implementar Junções

- Cont. J4. A segunda fase, chamada fase de exploração, calcula as chaves *hash* para os registros do outro arquivo (*S*) para sondar qual a partição apropriada e qual registro é combinado com todos os registros semelhantes de *R* na partição. Esta descrição simplificada da junção *hash* assume que o menor dos dois arquivos ajusta-se inteiramente nas partições de memória após a primeira fase.



## Estimativa de Custo

- Informações de Catálogo Usadas em Funções Custo:
  - Número de registros (tuplas) (*r*);
  - Tamanho (médio) do registro (*R*);
  - Número de blocos (*b*);
  - Fator de bloco (*bfr*);
  - Método de acesso primário e atributos de acesso primário para cada arquivo;



## Estimativa de Custo
### Informações de Catálogo

- Número de níveis (*x*) para cada índice multinível (primário, secundário ou agrupamento);
- Número de blocos de índices de primeiro nível ($b_{I1}$);
- Número de valores distintos (*d*) de um atributo;
- Seletividade de um atributo (*sl*), que permite estimar a cardinalidade de seleção ($s = sl * r$) de um atributo, que é o número médio de registros que satisfarão uma condição de seleção de igualdade sobre aquele atributo.



## Estimativa de Custo

- Para um atributo chave:
  $d = r$, $sl = 1/r$ e $s = 1$
- Para um atributo não chave, considerando distribuição uniforme de valores distintos entre os registros:
  $sl = (1/d)$ e $s = (r/d)$

## Estimativa de Custo

- Exemplos de Funções Custo para SELECT
  - S1. Busca Linear (força bruta): Nós buscamos todos os blocos de arquivo para recuperar todos os registros satisfazendo a condição de seleção; portanto, $C_{S1a} = b$. Para uma condição de igualdade sobre uma chave, somente metade dos blocos de arquivo são alcançados em média antes de encontrar o registro, então $C_{S1b} = (b/2)$ se o registro é encontrado; se nenhum registro satisfaz a condição, $C_{S1b} = b$.



## Estimativa de Custo
### Exemplos de Funções Custo para SELECT

- S2. Busca Binária: Esta busca acessa aproximadamente $C_{S2} = \log_2 b + \lceil (s/bfr) \rceil - 1$ blocos de arquivo. Isto reduz para $\log_2 b$ se a condição de igualdade está sobre um único atributo chave, porque $s = 1$ neste caso.



## Estimativa de Custo
### Exemplos de Funções Custo para SELECT

- S3. Usando um Índice Primário (S3a) ou Chave Hash (S3b) para Recuperar um Único Registro: Para índice primário, recupera um bloco a mais do número de níveis de índice; portanto, $C_{S3a} = x+1$. Para hashing, a função custo é aproximadamente $C_{S3b} = 1$ para hashing estatístico ou hashing linear e ela é 2 para hashing extensível.



## Estimativa de Custo
### Exemplos de Funções Custo para SELECT

- S4. Usando um Índice de Ordenação para Recuperar Múltiplos Registros: Se a condição de comparação é >, >=, < ou <= sobre um campo chave com um índice de ordenação, aproximadamente metade dos registros satisfarão a condição. Isto dá uma função custo de $C_{S4} = x + (b/2)$. Isto é uma estimativa muito bruta e, embora ela possa estar correta em média, ela pode ser muito imprecisa em casos individuais.



## Estimativa de Custo
### Exemplos de Funções Custo para SELECT

- S5. Usando um Índice de Agrupamento para Recuperar Múltiplos Registros: Dada uma condição de igualdade, $s$ registros satisfarão a condição, quando $s$ é cardinalidade de seleção do atributo de indexação. Isto significa que $\lceil (s/bfr) \rceil$ blocos de arquivo serão acessados, dando $C_{S5} = x + \lceil (s/bfr) \rceil$.



## Estimativa de Custo
### Exemplos de Funções Custo para SELECT

- S6. Usando um Índice Secundário (árvore $B^+$): Para uma comparação de igualdade, $s$ registros satisfarão a condição, onde $s$ é a cardinalidade de seleção do atributo de indexação. Entretanto, por não ser um índice de agrupamento, cada um dos registros pode residir num bloco diferente, então a estimativa de custo (pior caso) é $C_{S6a} = x + s$. Isto reduz para x + 1 para um atributo de indexação.

## Estimativa de Custo
### Exemplos de Funções Custo para SELECT

- Cont. S6. Se a condição de comparação é >, >=, < ou <= e assumindo que metade dos registros do arquivo satisfazem a condição, então (muito aproximadamente) metade dos blocos de índice de primeiro nível são acessados, mais metade dos registros do arquivo via o índice. A estimativa de custo para este caso é, aproximadamente, $C_{S6b} = x + (b_{I1}/2) + (r/2)$. O fator $r/2$ pode ser refinado se melhores estimati-vas de seletividade estiverem disponíveis.

31

## Estimativa de Custo
### Exemplos de Funções Custo para SELECT

- S7. Seleção Conjuntiva: Nós podemos usar S1 ou um dos métodos S2 a S6 discutidos acima. No último caso, nós usamos uma condição para recuperar os registros e checamos no buffer de memória quando cada registro recuperado satisfaz as condições restantes na conjunção.
- S8. Seleção Conjuntiva Usando um Índice Composto: Pode ser usado S3a, S5 ou S6a, dependendo do tipo de índice.

32