UFSM – Universidade Federal de Santa Maria DLSC
Prova 1 –Implementação de BD **Prof:** Sérgio Mergen

Nome do aluno:_____ **Matrícula do aluno:** _____

1) (1,5 pontos) Suponha que se deseja fazer uma **seleção por igualdade** sobre um atributo que é **chave primária e chave de ordenação** de um arquivo. Duas das possíveis estratégias de busca são a **busca linear** e o **uso de índice**. Indique para **qual quantidade de blocos** do arquivo passa a **valer a pena** usar o **índice**. O cálculo deve levar em consideração a soma dos custos de *seek* e transferência, sendo que um *seek* é 50 vezes mais lento que uma transferência. Para o uso de índice, considere que a árvore do índice tenha altura igual a **três**, para qualquer um dos tamanhos de arquivo considerados, e que também é necessário acessar o arquivo de dados. **Justifique sua resposta.**

a) 64 blocos
b) 128 blocos
c) 256 blocos
d) 512 blocos <--
e) 1024 blocos

Resposta:

1 ts = 50tt
*1ts + n/2 * tt > (h+1)*(ts+tt)*
*50tt + (n/2)tt > 4*51tt*
50tt + (n/2)tt > 204tt

Com n = 256
50tt + 128tt > 204tt
178tt > 204tt ?? (falso)

Com n = 512
50tt + 256tt > 204tt
306tt > 204tt ?? (verdadeiro) Letra D

2) (1,5 pontos) Suponha que se deseja **ordenar** um conjunto de registros divididos em **1024 partições** com **dois registros** cada. Ainda, suponha que, por limitação do espaço em memória, seja usado um mecanismo de **intercalação** que mescle partições, gerando partições maiores e em menor número. O mecanismo escolhido consegue diminuir o número de partições pela **metade** após cada estágio de intercalação. Ainda, em **cada estágio** de intercalação, **cada registro** sofre **uma operação de entrada e uma de saída**. Dadas essas características**, quantas operações** de entrada e saída são necessárias **ao todo**?
Resposta:

Número de estágios = log(1024) = 10
*Número de registros = 1024 * 2 = 2048*
*Estágios * registros = 10 * 2048 =* ***20.480***

Considere o esquema relacional da Figura 1 e as estatísticas da Figura 2 para responder as questões de 3 até 7.

3) (1,5 pontos) Considere que os registros da tabela **aloc** estão espalhados em 500 blocos, e os registros da tabela **proj** estão concentrados em um único bloco. Nesse cenário, indique o custo para realizar um **merge join** entre essas duas tabelas, em **número de seeks** e **número de transferências**, separadamente. **Ignore** o custo pare ordenar as relações, e considere o pior caso, em que só cabe **um bloco** de cada relação na **memória**.

Resposta:

N(aloc) = 10.000
N(proj) = 50
B(aloc) = 500
B(proj) = 1
Ts = 50tt

transferências
B(aloc) + B(proj) = 501tt
seeks
B(aloc) + B(proj) = 501ts

Mas.... só é necessário 1 bloco de projeto, então faz-se um seek para localizar esse bloco e outro seek para localizar o primeiro bloco da outra relação

seeks
1 + 1 = 2ts

De qualquer forma, a resposta anterior (501) também é válida.

4) (1,5 pontos) Marque V para as consultas que se **beneficiam pela ordenação** dos registros em alguma ordem específica, e F caso contrário? Para acertar a questão, **todas** marcações precisam estar corretas. Responda na **folha de respostas**.

a) Select distinct idFunc, salario from func (F)
b) Select salario, count(*) from func group by salario (V)
c) Select * from func order by salario (V)
d) Select salario from func (F)

Resposta: *F-V-V-F*

Apesar do uso do distinct na letra a), ela é falsa porque a consulta não gera duplicatas, uma vez que os registros são diferenciados entre si pelo idFunc.

5) (1,5 pontos) Em álgebra relacional, a operação de **diferença** não é **associativa**. Demonstre um caso que prove essa afirmação usando consultas em **álgebra relacional** sobre a tabela **func**. Para auxiliar na explicação, mostre os **registros da tabela**, bem como os **registros retornados pelas consultas**.

Resposta:

Registros de func (idFunc, idDepto, nomeFunc, salario):
(1, 1, 'Joao', 2000)
(2, 1, 'Ana', 3000)
(3, 2, 'Pedro', 3000)
(4, 2, 'Cesar', 4000)
(5, 3, 'Carla', 4000)
(6, 3, 'Paulo', 2000)

Consultas por extenso e usando relações temporárias, mas poderia ser em forma de árvore

rel1 <= Π salario (σ idDepto = 1 (func))
rel2 <= Π salario (σ idDepto = 2 (func))
rel3 <= Π salario (σ idDepto = 3 (func))

Consulta 1:
rel1 - (rel2-rel3)

*Resposta: {**2000,3000} - ({3000,4000}- {4000,2000})*
*{**2000,3000} - {3000}*
*{**2000}*

Consulta 2:
rel2 - (rel1-rel3)

*Resposta: {3**000,4000} - ({2000,3000}- {4000,2000})*
*{3**000,4000} - {3000}*
*{4**000}*

As duas consultas geram resultados diferentes

6) (1,5 pontos) Transforme a consulta SQL abaixo em uma expressão em **álgebra relacional otimizada**. Use as **três regras** de otimização vistas em aula.

Select f.*
from depto natural join func
where setor = ‘sul’ and salario > 2.000

Resposta:
rel1 <=Π idDepto (σ setor = sul (depto))
rel2 <=(σ salario > 2000 (func))
resp <= rel1 |X| rel2

7) (1,5 pontos) Considere a consulta abaixo:

```
Select *
from func natural join aloc natural join proj
where salario = 2.000 and custo = 2000
```

Existem **duas** possíveis ordens de junção que respondem essa consulta usando as relações de chave estrangeira. Das duas, a **mais eficiente** envolve combinar primeiro **aloc** e **proj**. Qual **deveria ser** o valor de **V(salario, func)** para que **valesse a pena** realizar primeiro a junção entre **aloc** e **func**?.

Resposta:

Estatísticas:
n(func) = 1.000
n(aloc) = 10.000
n(proj) = 50
v(salario,func) = 3
v(custo,proj) = 5

Após filtros:
n(func) = 1.000 / 3
n(proj) = 50 / 5

Comparação
1000/x < 50/5
1000/x < 10
x > 100
V(salario,func) > deveria ser maior do que 100

depto (idDepto, nomeDepto, setor)
func (idFunc, idDepto, nomeFunc, salario)
    idDepto referencia depto
proj (idProj, nomeProj, custo, duracao)
aloc (idFunc, idProj, função)
    idFunc referencia func
    idProj referencia proj

Figura 1 – Esquema Relacional

Chave	Valor
N. registros func	1.000
N. registros aloc	10.000
N. registros proj	50
V(salario, func)	3
V(custo, proj)	5

Figura 2 – Estatísticas das tabelas