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 a **busca binária**. Considerando que o tempo de seek seja **50 vezes** mais lento que o tempo de transferência, indique para qual **quantidade de blocos** do arquivo passa a **valer a pena** usar a **busca binária**. **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 > (ts+tt) * log (n)*
*50tt + (n/2)tt > 51tt * log (n)*

Com n = 256
*50tt + 128tt > 51tt *8*
178tt > 408tt ?? (falso)

Com n = 512
*50tt + 256tt > 51tt *9*
306tt > 459tt ?? (falso)

Com n = 1024
*50tt + 512tt > 51tt * 10*
562tt > 510tt ?? (verdadeiro) Letra e

2) (1,5 pontos) Suponha que se deseja **ordenar** um conjunto de registros divididos em **4096 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. Dadas essas características, **quantos** estágios de intercalação são necessários **ao todo**?

Resposta:

log (4096) = 12

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 espalhados em **5 blocos**. Nesse cenário, indique o **menor** custo para realizar um **block nested loop join** entre essas duas tabelas, **somando** o número de **seeks** e o número de **transferências**. Considere que o seek seja **50 vezes** mais lento que a transferência e que só caiba **um bloco** de cada relação na **memória**.

Resposta:

N(aloc) = 10.000
N(proj) = 50
B(aloc) = 500
B(proj) = 5
Ts = 50tt

Aloc relação externa
transferências
*B(aloc) + B(aloc) * B(proj) = 500 + 500 * 5 = 3.000tt*
seeks
B(aloc) + b(aloc) = 1.000ts = 50.000 tt
Ao todo = 53.000 tt

Proj relação externa
transferências
*B(proj) + B(proj) * B(aloc) = 50 + 50 * 500 = 2.550tt*
seeks
B(proj) + b(proj) = 10ts = 500tt
Ao todo = 3005tt
Menor custo = 3005 tt

4) (1,5 pontos) Marque V para a 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 salario from func union select custo from projeto **(V)**
b) Select salario from func union all select custo from projeto **(F)**
c) Select salario from func where salario < 1000 union select salario from func where salario >2000 **(V)**
d) Select idFunc from func where salario < 1000 union select idFunc from func where salario >2000 **(F)**

Obs. A claúsula **all** indica que todos registros devem ser considerados, independente da quantidade de vezes que cada um deles aparece.

A c) é verdadeira porque pode ter salários duplicados vindos de cada uma das subconsultas.

Resposta: *V-F-V-F*

5) (1,5 pontos) Em álgebra relacional, a operação de **diferença** não é **comutativa**. Demonstre um caso que prove essa afirmação usando **consultas SQL** 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)

Consulta 1:
select distinct salario from func where idDepto = 1
except
select distinct salario from func where idDepto = 2
Resposta:
{2000,3000} - {3000,4000}
{2000}

Consulta 2:
select distinct salario from func where idDepto = 2
except
select distinct salario from func where idDepto = 1
Resposta:
{3000,4000} - {2000,3000}
{4000}

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 nomeDepto, avg(salario)
from depto join func using(idDepto)
group by nomeDepto
```

Resposta:

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

rel1 <- (Π idDepto, nomeDepto (depto)) |X| (Π idDepto, salario (func))
resp <- nomeDepto G avg(salario) (rel1)

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

```
select *
from func natural join aloc natural join proj
where funcao = 'coordenador' or aloc.duração = 2
```

Suponha que essa consulta foi transformada em um plano de execução usando as **regras de otimização** vistas em aula. Nesse caso, qual seria a **primeira junção** a ser feita? **Quantos registros** seriam retornados por essa junção?

Resposta:

n(func) = 1.000
n(aloc) = 10.000
n(proj) = 50
v(func,aloc) = 10
v(duração,aloc) = 20

*n(aloc) = 10.000 * (1 – ((1- 1/10) * (1-1/20)))*
*n(aloc) = 10.000 * (1 – (9/10 * 19/20))*
*n(aloc) = 10.000 * (1 – (171/200))*
*n(aloc) = 10.000 * (29/200)*
n(aloc) = 1.450

func |X| proj = 50.000 registros
func |X| aloc = 1.450 registros
proj |X| aloc = 1.450 registros

melhor junção:
proj |X| aloc, trazendo 1.450 registros
ou
func |X| aloc, trazendo 1.450 registros

depto (idDepto, nomeDepto, setor)
func (idFunc, idDepto, nomeFunc, salario)
    idDepto referencia depto
proj (idProj, nomeProj, custo, duracao)
aloc (idFunc, idProj, função, duracao)
    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(função, aloc)	10
V(duração,aloc)	20

Figura 2 – Estatísticas das tabelas