Nomes: Samuel H. Dalmas e Maria Carolina **Professora:** Helena Grazziotin
Disciplina: Aspectos de Implementação de Banco de Dados **Horário:** 28-29

Documentação Escalonador

1) Protocolo e Características do Funcionamento

O escalonador implementado utiliza o 2PL Strict, seguindo suas características de maneira fidedigna.

O escalonador analisa a história inicial (informada pelo usuário ou gerada automaticamente) até chegar na última operação. Após isso, verificará se há deadlock. Para considerar que houve deadlock, a lista de bloqueios exclusivos ou compartilhados deve possuir algum dado ao chegar no final da história.

Se esse requisito acontecer, escolherá uma vítima para sofrer abort (transação há mais tempo ativa no escalonador), reverterá suas modificações e mandará suas operações novamente para o escalonador.

Em relação à liberação de bloqueios, é feita logo após o commit, liberando todos os bloqueios exclusivos e compartilhados relacionados à transação.

2) Características Implementadas

Desenvolvido com a linguagem Java.

O funcionamento da aplicação é controlado por um menu principal, em que são apresentadas as opções disponíveis:

```
############## ESCALONADOR 2PL STRICT ##############


Escolha uma Opcao:

1- Ler Transacoes do Arquivo
2- Cadastrar Transacoes
3- Digitar Historia Inicial
4- Mostrar Transacoes Cadastradas
5- Mostrar Transacoes Armazenadas
6- Gerar Historia de Execucao
7- Executar Escalonador
Opcao:
```

Descrição das opções:

1 - Ler Transações do arquivo:

Lê arquivo txt incluso no diretório do projeto. Nesse arquivo são incluídas transações (devem começar da transação número 0) que o programa armazenará durante a execução.

2 - Cadastrar Transações:

Permite ao usuário inserir transações em tempo de execução. Necessário começar pela número 0.

3 - Digitar História Inicial:

O usuário pode informar uma história independentemente da execução atual, para verificar histórias já conhecidas, podendo ter outras transações envolvidas. Não há necessidade de começar com a transação 0, podendo assim começar pela 1, diferentemente dos outros módulos do programa.

4 - Mostrar Transações Cadastradas:

Exibe as transações cadastradas na opção 2 do menu.

5 - Mostrar Transações Armazenadas:

Exibe as transações lidas do arquivo txt.

6 - Gerar História de Execução:

A aplicação irá gerar uma história com as operações entrelaçadas. Para essa opção, pode-se utilizar as transações lidas do arquivo ou as informadas no menu.

7 - Executar Escalonador:

Manda as operações para o escalonador, podendo escolher a história gerada pela aplicação ou a informada pelo usuário.

3) Entrada e Saída de Dados:

Os dados informados necessitam possuir apenas um caracter.
Exemplo: p, z, x, y,...

- Através da opção 1 do menu, o arquivo txt deve estar formatado da seguinte maneira:

```
r0[p] w0[p] r0[p] c0
r1[z] w1[z] r1[z] c1
```

- Através da opção 2 do menu, transações devem ser cadastradas com a seguinte formatação:

```
Opcao Escolhida: 2

Digite Enter para proxima transacao e -1 se todas as transacoes foram informadas
Digite a transacao T0: r0[p] w0[p] r0[p] c0
Digite a transacao T1: r1[z] w1[z] r1[z] c1
Digite a transacao T2: -1

Voce informou 2 transacoes
```

- Através da opção 3 do menu, a história inicial deve ser cadastradas com a seguinte formatação:

```
Opcao Escolhida: 3
Digite a historia: w1[x] w2[y] r2[x] r1[y] c1 c2
```

A saída de dados final apresenta, além das operações da história inicial, os locks (exclusivos e compartilhados) além da liberação dos mesmos após o commit.

Exemplo de Execução 01:

Para a história: w1[x] w2[y] r2[x] r1[y] c1 c2 (com deadlock)

Tem-se a saída:

```
Historia atualizada:
lx2[y] w2[y] ls2[x] r2[x] c2 ux2[y] us2[x] lx1[x] w1[x] ls1[y] r1[y] c1 ux1[x] us1[y]
```

Exemplo de Execução 02:

Para a história: w1[x] r1[x] w1[y] r2[y] r2[z] c1 c2

Tem-se a saída:

```
Historia atualizada:
lx1[x] w1[x] r1[x] lx1[y] w1[y] c1 ux1[x] ux1[y] ls2[y] r2[y] ls2[z] r2[z] c2 us2[y] us2[z]
```

Exemplo de Execução 03:

Para a história: r0[p] w0[p] r1[z] w1[z] r0[p] r1[z] c1 c0

Tem-se a saída:

```
Historia atualizada:
ls0[p] r0[p] lx0[p] w0[p] ls1[z] r1[z] lx1[z] w1[z] r0[p] r1[z] c1 ux1[z] c0 ux0[p]
```

4) Estruturas de Dados Utilizadas:

Armazenamento de transações: ArrayList de objetos do tipo Transacao.

Armazenamento de histórias: Para a história inicial, é utilizado ArrayList do tipo operação. Quando a história é entrelaçada, passa para vetor de objetos Operacao. A história final é mantida num ArrayList de Strings.

Armazenamento de locks: Os locks exclusivos e compartilhados são mantidos em dois ArrayLists de objetos do tipo Operacao.

Controle de Execução: Vários dados como opções escolhidas do menu e flags são mantidas para manter o correto fluxo de execução. Esses dados são do tipo inteiro e as flags, do tipo boolean.

Dados: A aplicação possui 5 classes, sendo:

Main: inicia a aplicação

Escalonador: possui a lógica principal do funcionamento e controle de execução

Historia: refere-se às histórias presentes durante o funcionamento

Transacao: representa as transações das quais a história é constituída

Operacao: relaciona-se às operações presentes nas transações e histórias