MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
DEPARTAMENTO DE COMPUTAÇÃO

Engenharia de Computação
Laboratório de Banco de Dados I – 6º Período – 1º Semestre de 2024
Prof. Evandrino G. Barros (evandrino@cefetmg.br)

Exercício 6 de LBD – 2 pontos em duplas - Processamento de transação e Controle de concorrência – Divulgação: 02/08/2024 – Data de Entrega: 09/08/2024.

Execute, analise e documente os seguintes passos abaixo. Apresente um documento PDF com os comandos executados e seus resultados, bem como a explicação dos mesmos resultados. Basta um relatório por dupla. No entanto, as interações dos participantes do grupo devem ser apresentadas no relatório, sob a perspectiva de somente um deles.

1) Forme uma dupla, na qual cada um deve estar conectado ao banco. Um dos membros será denominado usuário 1 e outro o usuário2.

2) Cada usuário deve criar a seguinte tabela.

```
Create table MEMBRO (  mat number(12)      primary key,
                       nome varchar2(40) not null,
                       cpf varchar2(11)      not null,
                       ende varchar2(40)  not null,
                       data_nasc date         not null);
```

3) Cada usuário deve inserir os seus dados de MEMBRO na tabela. Não confirme a inserção. Ou seja, não dê "*commit*". Para inserir, utilize o comando i*nsert.*

   Exemplo: *insert into MEMBRO values*
   (111, 'Antonio', '11111111', 'Rua A',
   to_date('DD/MM/YYYY','01/12/2001'));

4) Tente consultar a tabela MEMBRO do outro usuário. Para fazer isso, prefixe a tabela com o nome do esquema do outro usuário. Por exemplo, se quiser acessar os dados de *ECBDII00*, emita o seguinte comando.

   *Select * from ECBDII00.MEMBRO;*

5) Não será possível o acesso, pois não foi dada a permissão de consulta. Para dar a permissão, utilize o seguinte comando, que é da *DCL – Data Control Language*.

   Grant select on MEMBRO to ECBDII00;

Substitua, nesse exemplo, o usuário *ECBDII00* pelo usuário a quem você quer dar acesso. Ou seja, o usuário 1 deve dar acesso ao usuário 2 e vice-versa.

Em seguida, tente fazer a mesma consulta do item 4. Será possível ver as alterações, mesmo sem "*commit*" explícito. O comando "*grant*" provoca um "*commit*" implícito para deixar o dicionário de dados persistente e consistente para todos os usuários.

6) Insira dois membros fictícios em sua tabela, mas não confirme (não use "*commit*"). Peça ao outro usuário que tente consultar esses novos dois membros.

7) Agora sim, confirme, emitindo o comando "*commit*" e peça o outro usuário para consultar a sua tabela. O comando "*commit*" finalizou a transação corrente, confirmando para todos as atualizações mantidas na transação.

8) Insira mais dois membros fictícios em sua tabela, sem confirmar. A primeira inserção inicia uma nova transação. Consulte a sua tabela para ver esses novos membros e peça ao outro usuário que tente consultar a sua tabela. Como não houve confirmação com *commit*, só você vê os dois novos membros.

9) Agora desfaça a sua transação corrente, emitindo o comando "*rollback*". Consulte a tabela e veja que somente os dois últimos membros inseridos foram removidos. Na verdade, os *inserts* do item 8 foram desfeitos. Os outros inserts são da transação anterior que foi finalizada com *commit*, portanto não são desfeitos.

10) Tente a alteração (comando update) de endereço em uma linha da tabela MEMBRO do outro usuário. Você não irá conseguir, pois não tem permissão.

Exemplo: *update ECBDII00.MEMBRO set nome='Jose' where mat=111;*

11) Dê ao outro usuário acesso de atualização em sua tabela MEMBRO. Para isso, emita o comando "*grant update on MEMBRO to <outro usuário>*".

12) Tente fazer o item 10 novamente, mas não confirme. Consulte a alteração que você fez agora e peça ao dono da tabela (ou seja, o outro usuário) para consultar a tabela dele para ver a atualização que você fez. Você conseguirá ver a atualização, mas o dono da tabela não, pois você não emitiu *commit*.

13) Agora dê *commit* em sua transação, consulte a tabela do outro para ver a atualização feita no item 10 e peça ao dono da tabela para fazer essa mesma consulta.

14) Combine com o outro usuário de tentar fazer a atualização em uma mesma linha da tabela MEMBRO. Escolha a tabela que será alterada, se a sua ou a do outro usuário. Os dois devem emitir comando update, cada um em sua transação e na mesma linha da mesma tabela. Aquele que emitir por último ficará travado.

15) Para destravar o outro usuário, o usuário destravado deve emitir o comando *commit* ou *rollback*;

16) Agora, vamos testar a concorrência múltipla em um banco de dados. Combine com

o outro usuário de acessar os mesmos dados, conforme a seguinte sequência:

a) No seu usuário, atualize um registro na tabela do outro usuário, mas sem confirmar com *commit*;

b) Agora tente atualizar o registro que o outro usuário atualizou na sua tabela. Haverá uma espera mútua, que corresponde ao travamento mútuo das sessões envolvidas. O banco de dados identifica isso e desfaz uma das atualizações envolvidas nessa situação. Ainda assim, o outro usuário ficará esperando. Essa situação é chamada de *deadlock*.

17) Para revogar o privilégio concedido emita o comando *revoke*, também da DCL. Exemplo de *revoke*, nesse caso do usuário corrente revogando de um usuário que recebeu um privilégio:

revoke select, update on MEMBRO from MEMBRO21;

Comandos da DCL (*grant* e *revoke*) têm *commit* implícito, por confirmar qualquer transação pendente.