Ficha Prática 7

Transações e Controlo de Concorrência

Bases de Dados

Departamento de Engenharia Informática

1 2 9 0

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

1. Abra duas sessões SQL (*psql* ou *pgadmin*) usando a mesma conta (*aulaspl*). Com uma delas, crie uma tabela DEP1 com os mesmos dados que a tabela DEP (`create table dep1 as select * from dep`). Se a tabela DEP1 já existir, remova-a (`drop table dep1`).

2. Chamemos a uma das sessões Sessão A e à outra a Sessão B. Na Sessão A, execute:

```
select * from dep1;
```

e de seguida faça o mesmo na Sessão B. Ambas produzem o mesmo resultado, certo?

Execute o seguinte código em A:

```
begin transaction;

update dep1 set local='VISEU' where ndep = 10;

select * from dep1;
```

E agora em B:

```
select * from dep1;
```

O que aconteceu? Em A viu-se o valor da localização do departamento 10 alterada enquanto que em B ainda se vê a localização antiga. Como justifica isso sabendo que tanto A como B estão ligadas ao mesmo utilizador?

3. Ainda em B execute:

```
begin transaction;

update dep1 set local='LISBOA' where ndep = 20;
```

Em A e em B execute:

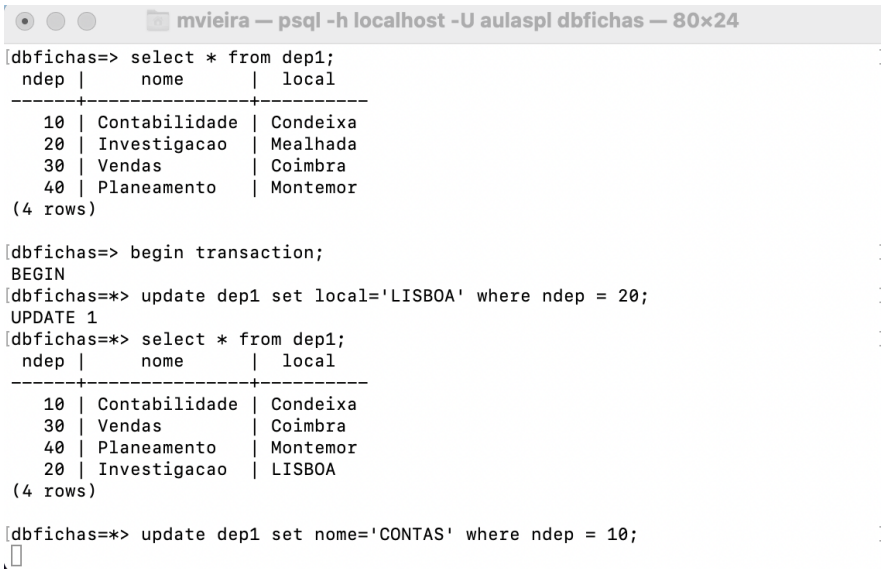
```
select * from dep1;
```

Tanto A como B só vêm as suas próprias alterações.

Continuando em B faça:

```
update dep1 set nome='CONTAS' where ndep = 10;
```

O resultado é o da figura seguinte (o facto de a janela ter bloqueado na última linha é que é importante).



Justifique a situação. Como é que se pode desbloquear B?

4. Em A execute `commit`. Quais as consequências da execução deste comando? Existe mais do que uma e algumas não são visíveis!…

Faça `rollback` em B. Verifique que as alterações em B foram desfeitas.

5. Se em vez de fazer `commit` no passo anterior, tivesse feito `rollback`, também desbloquearia? Qual seria a diferença em termos do resultado obtido?

6. E se no `update` o valor a alterar fosse rigorosamente igual ao valor que já lá estava (por exemplo mudar a localização de "Coimbra" para "Coimbra")? Será que produziria bloqueio do registo "alterado"? Experimente e responda.

7. Inicie duas novas transações em A e em B:

```
begin transaction;
```

Agora em A execute:

```
update dep1 set local='VISEU' where ndep = 10;
```

E em B execute:

```
update dep1 set local='LISBOA' where ndep = 20;
```

```
update dep1 set nome='CONTAS' where ndep = 10;
```

B bloqueou porque está a tentar alterar um registo bloqueado por A.

Agora em A tente alterar um registo bloqueado por B fazendo:

```
update dep1 set nome='PESCAS' where ndep = 20;
```

O PostgreSQL deteta o impasse (*deadlock* em inglês) e passado uns segundos quebra-o. Quando um impasse é detetado e quebrado pelo sistema existe sempre informação que se perde. O que é que se perdeu aqui?

8. Desfaça todas as alterações fazendo `rollback` nas duas sessões. Abra agora uma terceira sessão, C, com o mesmo *login* e *password* e tente criar uma situação de impasse com 3 sessões.

9. Faça `rollback` de todas as transações. Na janela A inicie uma transação só de leitura fazendo:

```
begin transaction;

set transaction read only;
```

Tente alterar o valor de um registo em A. Execute, por exemplo, o seguinte comando:

```
update dep1 set local = 'VISEU' where ndep = 10;
```

Como pode verificar, não pode alterar dados numa transação só de leitura.

Em A execute:

```
rollback;

begin transaction;

set transaction isolation level repeatable read read only;
```

Em B execute:

```
begin transaction;

update dep1 set local='LISBOA' where ndep = 20;
```

Verifique o que as sessões A e C vêm fazendo em ambas:

```
select * from dep1;
```

Agora volte a B e execute `commit`.

Verifique novamente o que as sessões A e C vêm através de:

```
select * from dep1;
```

i. Como justifica?
ii. Quais as consequências do *commit* em B?
iii. Como é que A ainda vê os valores antigos?
iv. O que é que acontece se A não fizer *commit* (ou *rollback*) ou demorar muito tempo a fazê-lo?

10. Em A, execute:

```
rollback;

begin transaction;

update dep1 set local = 'VISEU' where ndep = 10;
```

Execute o seguinte código em B:

```
begin transaction;

select * from dep1 where ndep in (10, 20) for update;
```

Porque é que B ficou bloqueado?

11. Faça `commit` em A. B desbloqueia. Agora, ainda em A tente alterar novamente o registo fazendo:

```
begin transaction;

update dep1 set nome = 'CONTAS' where ndep = 10;
```

Justifique o sucedido.

12. Em B execute `commit`. Ao fazer isto A é libertado. Agora, em B, execute:

```
begin transaction;

select * from dep1 where ndep in (10, 20) for update nowait;
```

O que aconteceu e porquê?

13. Faça `commit` em A e repita o comando em B:

```
rollback;

begin transaction;

select * from dep1 where ndep in (10, 20) for update nowait;
```

Quais as diferenças de resultados? Os registos estão bloqueados? Por quem?