

# Ficha 8 – Transações

Bases de Dados, FCT-NOVA

Ano letivo 2020/21

**Grupo 1.** Durante os exercícios deste grupo serão testadas transações concorrentes. Para isso, deverá ter duas consolas, ou terminais (SQL Developer, SQL Plus, mas *não em APEX*) abertos sobre a mesma base de dados.

Alguns detalhes sobre o início e fim das transações em Oracle:

- Uma transação começa com o primeiro comando depois de um `commit` ou um `rollback`, ou depois de se estabelecer uma ligação ao servidor.
- Uma transação termina com um `commit` ou com um `rollback`, ou quando se termina a ligação ao servidor.
- Se uma ligação termina de forma normal, o Oracle implicitamente executa um `commit`.
- Se uma ligação termina de forma anormal, devido a um erro, o Oracle executa um `rollback`.
- Sempre que um comando DDL é executado (e.g. `create table`, etc...) o Oracle implicitamente adiciona um `commit` antes e depois.
- Para Informação adicional sobre os mecanismos fornecido pelo Oracle para suportar concorrência, consulte a parte III “Oracle Transaction Management” do manual “Oracle Database Concepts 18c” (disponível em <https://docs.oracle.com/en/database/oracle/oracle-database/18/cncpt/oracle-transaction-management.html>)

1. Crie uma tabela `t` e uma tabela `s`, ambas com um atributo `a` do tipo inteiro e um atributo `b` do tipo `varchar2(30)`, com os comandos:

```
create table t (a int primary key, b varchar2(30));
create table s (a int primary key, b int);
```

Teste que o Oracle implicitamente adiciona um `commit` após a criação da tabela, inserindo na tabela `t`, **no terminal diferente daquele onde a tabela foi criada**, os tuplos (1, 'um') e (2, 'dois'), com os comandos

```
insert into t values (1, 'um');
insert into t values (2, 'dois');
```

Faça um `commit`.

2. Execute o seguinte escalonamento:

T <sub>1</sub>	T <sub>2</sub>
<code>select * from t;</code>	<code>insert into t values (3, 'tres');</code>
<code>select * from t;</code>	<code>commit;</code>

Apenas com este teste, o que pode concluir sobre o nível de Isolamento garantido por omissão pelo Oracle? Porquê?

3. O Oracle tem três níveis de isolamento,:

```
SERIALIZABLE  
READ ONLY  
READ COMMITTED
```

Para alterar o nível de isolamento, deve ser executado o seguinte comando no início de uma transação:

```
SET TRANSACTION ISOLATION LEVEL {SERIALIZABLE | READ COMMITTED};
```

ou

```
SET TRANSACTION READ ONLY;
```

Considere o seguinte escalonamento:

T <sub>1</sub>	T <sub>2</sub>
<pre>select * from s; select * from t; insert into t values (4,'quatro'); commit;</pre>	<pre>select * from s;       select * from t; commit;</pre>

- (a) Este escalonamento é serializável?
- (b) Execute o escalonamento em modo serializável e verifique o comportamento do Oracle.

4. Considere agora o seguinte escalonamento:

T <sub>1</sub>	T <sub>2</sub>
<pre>select * from t;   update t set b = 'a' where a = 1; commit;</pre>	<pre>select * from t;    update t set b = 'c' where a = 1; commit;</pre>

- (a) Este escalonamento é serializável?
- (b) Execute o escalonamento em modo serializável e verifique o comportamento do Oracle.

5. Execute primeiro os seguintes comandos:

```
insert into s values(1,10);  
insert into s values(2,10);
```

Depois execute em modo serializável o escalonamento:

T <sub>1</sub>	T <sub>2</sub>
select sum(b) from s;	select sum(b) from s;
update s set b = 0 where a = 1;	update s set b = 0 where a = 2;
select sum(b) from s;	select sum(b) from s;
commit;	commit;

Como justifica o comportamento do Oracle neste exemplo?

6. Apresente um escalonamento de duas transações concorrentes em que o comportamento (e resultado final na base de dados) se executado em modo `READ COMMITTED` seja diferente do comportamento em modo `SERIALIZABLE`.

**Grupo 2.** Em transacções ACID, a consistência só é imposta no fim de uma transacção. Aliás, se tal não fosse assim, algumas operações válidas não poderiam ser efectuadas: por exemplo, não seria possível inserir tuplos em relações muitos para muitos, totais em ambos os lados.

Apesar de não ser o comportamento por omissão, o Oracle permite que restrições de integridade sejam verificadas apenas no fim de uma transacção. Por omissão, a consistência (chaves candidatas, restrições de domínio e chaves estrangeiras) é verificada após cada operação de manipulação de dados (insert, delete e update).

Para verificar as restrições apenas no fim das transacções, é necessário:

- declarar como `DEFERRABLE` todas as restrições que se pretende que sejam verificadas apenas no fim da transacção; isto pode ser feito adicionando `DEFERRABLE` no fim do comando que cria a restrição.
- para cada transacção, impor que a verificação deve ser adiada, através do comando:  
`SET CONSTRAINTS ALL DEFERRED`.

Também é possível adiar apenas algumas restrições, substituindo o `ALL` por uma lista com os nomes das restrições que pretendemos adiar.

1. Crie as seguintes tabelas:
  - `mulheres`, com atributos `nome` e `marido`, ambos `not null`, onde `nome` é chave primária;
  - `maridos`, com atributos `nome` e `mulher`, ambos `not null`, onde `nome` é chave primária;
2. Adicione restrições de integridade que garantam que, em cada momento, uma mulher tem que ter um marido e um marido tem que ter uma mulher.
3. Verifique que, sem o mecanismo para adiar a verificação de restrições, não é possível adicionar qualquer tuplo às tabelas criadas.
4. Use o mecanismo descrito para adiar a verificação das restrições, e verifique que já é possível adicionar tuplos às tabelas criadas.