

Nome: Harrison Caetano Candido
RA: 156264

1 - TESTE - BLOQUEIO POR TABELA NO MODO READ

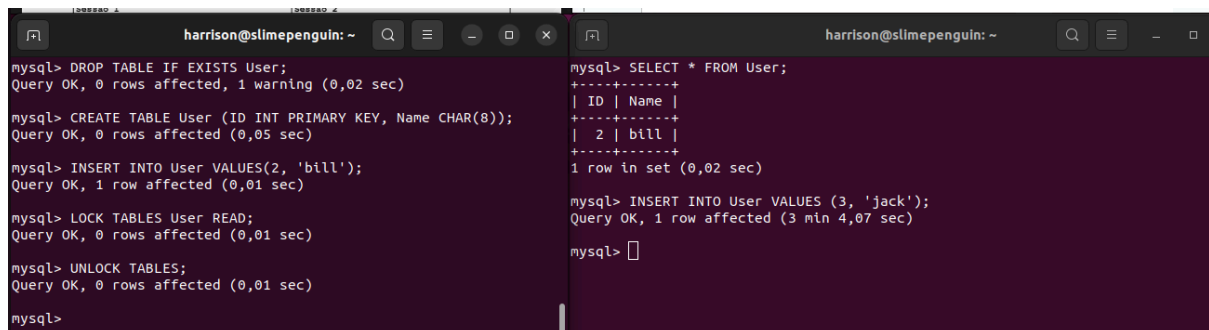
Sessão 2: O que ocorre após o unlock?

Após o uso de “LOCK TABLES User READ;” é feito um bloqueio no modo READ LOCK, em que o dado é reservado para leitura na sessão atual, mas permite que outra sessão também realize a operação de leitura. É o que chamamos de SHARED LOCK.

Visto que apenas operação de leitura é permitida na tabela, qualquer outra transação que tentar escrever ficará em espera ociosa até que uma operação “UNLOCK TABLES;” seja aplicada.

Sessão 1: O que ocorre após o unlock?

Após o “UNLOCK TABLES;” qualquer operação de escrita bloqueada em outra transação poderá ser finalizada com sucesso.

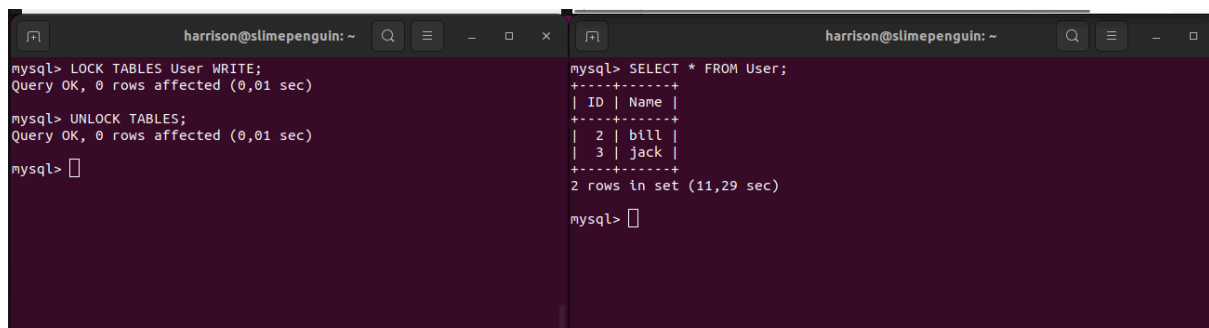


```
harrison@slimepenguin: ~  
mysql> DROP TABLE IF EXISTS User;  
Query OK, 0 rows affected, 1 warning (0,02 sec)  
  
mysql> CREATE TABLE User (ID INT PRIMARY KEY, Name CHAR(8));  
Query OK, 0 rows affected (0,05 sec)  
  
mysql> INSERT INTO User VALUES(2, 'bill');  
Query OK, 1 row affected (0,01 sec)  
  
mysql> LOCK TABLES User READ;  
Query OK, 0 rows affected (0,01 sec)  
  
mysql> UNLOCK TABLES;  
Query OK, 0 rows affected (0,01 sec)  
  
mysql>  
  
harrison@slimepenguin: ~  
mysql> SELECT * FROM User;  
+-----+  
| ID | Name |  
+-----+  
| 2 | bill |  
+-----+  
1 row in set (0,02 sec)  
  
mysql> INSERT INTO User VALUES (3, 'jack');  
Query OK, 1 row affected (3 min 4,07 sec)  
  
mysql>
```

TESTE 2 - BLOQUEIO POR TABELA NO MODO WRITE

O que ocorre?

Nesse caso, diferentemente do TESTE 1, nós estamos realizando uma operação de bloqueio no modo EXCLUSIVE LOCK com o uso de “LOCK TABLES User WRITE;”, que impede a leitura e a escrita. Logo, a operação de LEITURA da segunda Transação falha, por que qualquer operação seja de leitura ou escrita só é permitida para a Transação que iniciou o bloqueio.



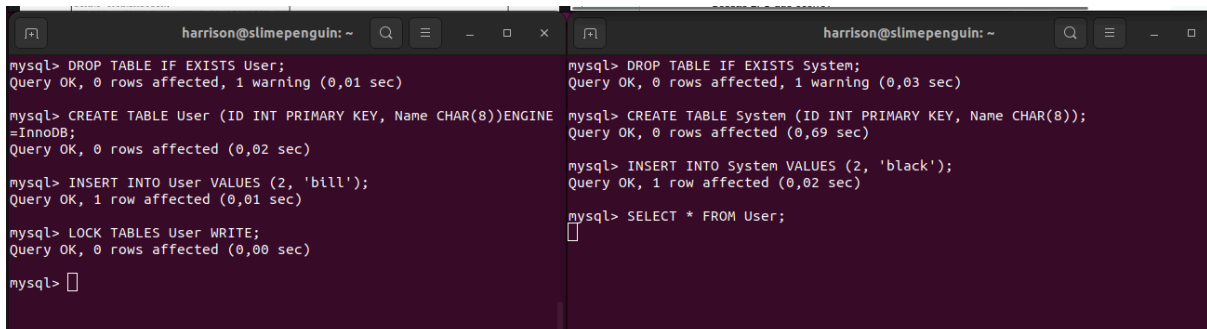
```
harrison@slimepenguin: ~  
mysql> LOCK TABLES User WRITE;  
Query OK, 0 rows affected (0,01 sec)  
  
mysql> UNLOCK TABLES;  
Query OK, 0 rows affected (0,01 sec)  
  
mysql>  
  
harrison@slimepenguin: ~  
mysql> SELECT * FROM User;  
+-----+  
| ID | Name |  
+-----+  
| 2 | bill |  
| 3 | jack |  
+-----+  
2 rows in set (11,29 sec)  
  
mysql>
```

TESTE 3 - BLOQUEIO POR TABELA NO MODO WRITE

O que ocorre? Justifique

As operações da transação da sessão 1 levam ao bloqueio da tabela User no modo EXCLUSIVE LOCK ao utilizar o parâmetro WRITE, ou seja, apenas a transação da sessão 1 é capaz de ler ou escrever na tabela User. Podemos ter um exemplo disso na transação

da sessão 2, em que consulta SELECT entra em espera ociosa pelo desbloqueio exclusivo da tabela User.



```
harrison@slimepenguin: ~  
mysql> DROP TABLE IF EXISTS User;  
Query OK, 0 rows affected, 1 warning (0,01 sec)  
  
mysql> CREATE TABLE User (ID INT PRIMARY KEY, Name CHAR(8))ENGINE  
=InnoDB;  
Query OK, 0 rows affected (0,02 sec)  
  
mysql> INSERT INTO User VALUES (2, 'bill');  
Query OK, 1 row affected (0,01 sec)  
  
mysql> LOCK TABLES User WRITE;  
Query OK, 0 rows affected (0,00 sec)  
  
mysql>   
  
harrison@slimepenguin: ~  
mysql> DROP TABLE IF EXISTS System;  
Query OK, 0 rows affected, 1 warning (0,03 sec)  
  
mysql> CREATE TABLE System (ID INT PRIMARY KEY, Name CHAR(8));  
Query OK, 0 rows affected (0,69 sec)  
  
mysql> INSERT INTO System VALUES (2, 'black');  
Query OK, 1 row affected (0,02 sec)  
  
mysql> SELECT * FROM User;  
□
```

TESTE 4 – BLOQUEIO POR LINHA MODO READ

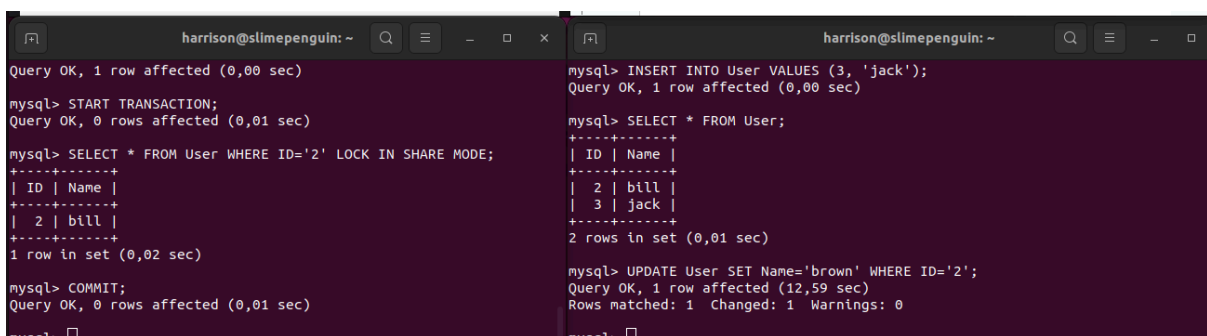
Observação: Utilizamos a linha START TRANSACTION, pois ela desabilita o commit automático.

Sessão 2: O que ocorre?

A transação da sessão 2 fica em espera ociosa quando tenta dar um UPDATE (basicamente uma operação de escrita) em uma linha da tabela User que está bloqueada apenas para operações de escrita para qualquer transação que não seja a da sessão 1.

Sessão 1: O que ocorre após o commit?

Após o Commit, o SGBD entende que a transação finalizou e por isso qualquer bloqueio feito pela transação da sessão 1 é desabilitado, permitindo que a transação da sessão 2 realize sua operação de ESCRITA na linha anteriormente bloqueada. Uma operação “LOCK IN SHARE MODE” permite a leitura de qualquer transação que não utilize um “LOCK IN SHARE MODE” e “FOR UPDATE”, bem como não permite a escrita de qualquer transação que não seja a atual.



```
harrison@slimepenguin: ~  
Query OK, 1 row affected (0,00 sec)  
  
mysql> START TRANSACTION;  
Query OK, 0 rows affected (0,01 sec)  
  
mysql> SELECT * FROM User WHERE ID='2' LOCK IN SHARE MODE;  
+-----+  
| ID | Name |  
+-----+  
| 2 | bill |  
+-----+  
1 row in set (0,02 sec)  
  
mysql> COMMIT;  
Query OK, 0 rows affected (0,01 sec)  
  
mysql>   
  
harrison@slimepenguin: ~  
mysql> INSERT INTO User VALUES (3, 'jack');  
Query OK, 1 row affected (0,00 sec)  
  
mysql> SELECT * FROM User;  
+-----+  
| ID | Name |  
+-----+  
| 2 | bill |  
| 3 | jack |  
+-----+  
2 rows in set (0,01 sec)  
  
mysql> UPDATE User SET Name='brown' WHERE ID='2';  
Query OK, 1 row affected (12,59 sec)  
Rows matched: 1 Changed: 1 Warnings: 0  
  
mysql>   

```

TESTE 5 – BLOQUEIO POR LINHA MODO WRITE

O que ocorre antes e depois do commit?

Antes do commit a consulta “UPDATE User SET Name='brown' WHERE ID='2';” na transação da sessão 2 fica em espera ociosa por que ela tenta dar escrever em uma linha com EXCLUSIVE LOCK, ou seja, nenhuma operação com “LOCK IN SHARE MODE”, “FOR UPDATE”, “UPDATE”, “WRITE”, ou qualquer outro tipo que escreva ou tente dar criar bloqueio será capaz de performar. O uso de “SELECT * FROM User;” na transação da

sessão 2 só foi possível por que a operação de leitura é possível de ser realizada externamente a transação que iniciou o bloqueio com “FOR UPDATE”. Porém após o COMMIT na transação da sessão 1, a operação “UPDATE User SET Name='brown' WHERE ID='2';” sai da espera ociosa e é finalizada.

TESTE 6

Sessão 2: O que ocorre?

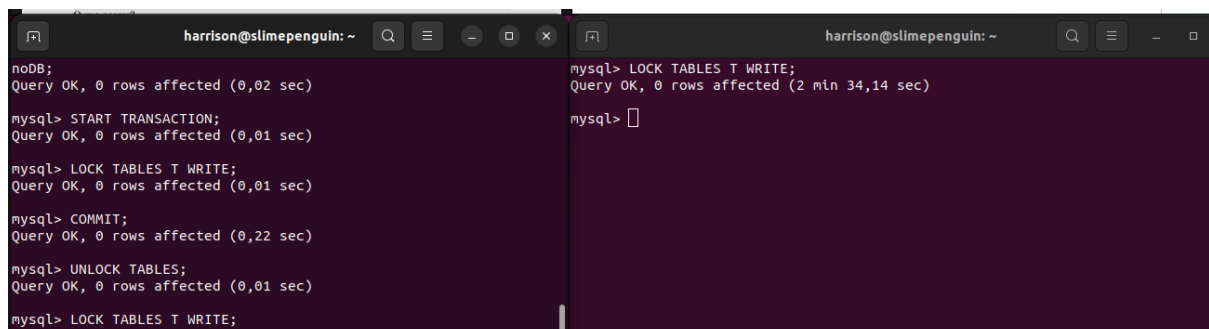
Nesse caso em que uma transação bloqueia a tabela T antes de outra transação que realizaria a mesma operação, ocorre o bloqueio da transação da sessão 2, pois o bloqueio de tabela do tipo EXCLUSIVE LOCK impede que até mesmo outras transações realizem operação de LOCK na tabela bloqueada.

Sessão 2: Ocorreu algo depois do commit?

Nada aconteceu após o commit, pois não estamos executando nenhuma operação de bloqueio em linhas, mas sim em toda a tabela. Logo o commit não é capaz de desabilitar o efeito dos comandos de bloqueio passados através das linhas da transação da sessão 1.

Sessão 1: Desbloqueie a tabela T e o que ocorre na sessão 2?

Quando utilizamos o comando “UNLOCK TABLES;” nós realmente permitimos que qualquer tabela seja desbloqueada, o que faz com que a operação de bloqueio que estava em espera ociosa na transação da sessão 2 seja capaz de performar seu bloqueio na tabela T.

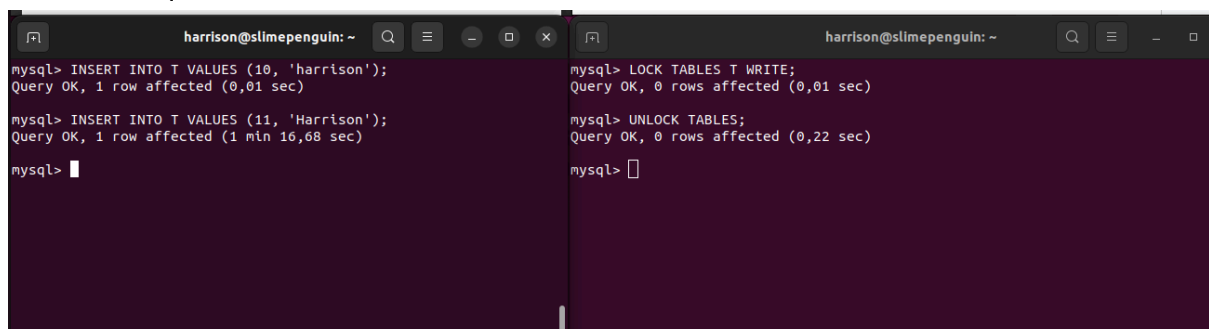


```
harrison@slimepenguin: ~  
noDB;  
Query OK, 0 rows affected (0,02 sec)  
  
mysql> START TRANSACTION;  
Query OK, 0 rows affected (0,01 sec)  
  
mysql> LOCK TABLES T WRITE;  
Query OK, 0 rows affected (0,01 sec)  
  
mysql> COMMIT;  
Query OK, 0 rows affected (0,22 sec)  
  
mysql> UNLOCK TABLES;  
Query OK, 0 rows affected (0,01 sec)  
  
mysql> LOCK TABLES T WRITE;  
  
harrison@slimepenguin: ~  
mysql> LOCK TABLES T WRITE;  
Query OK, 0 rows affected (2 min 34,14 sec)  
  
mysql>
```

TESTE 7

O que ocorre?

Como utilizamos uma operação de bloqueio do tipo EXCLUSIVE LOCK na tabela T, nenhuma operação de escrita é permitida, a não ser que esta seja feita na transação que iniciou o bloqueio da tabela.

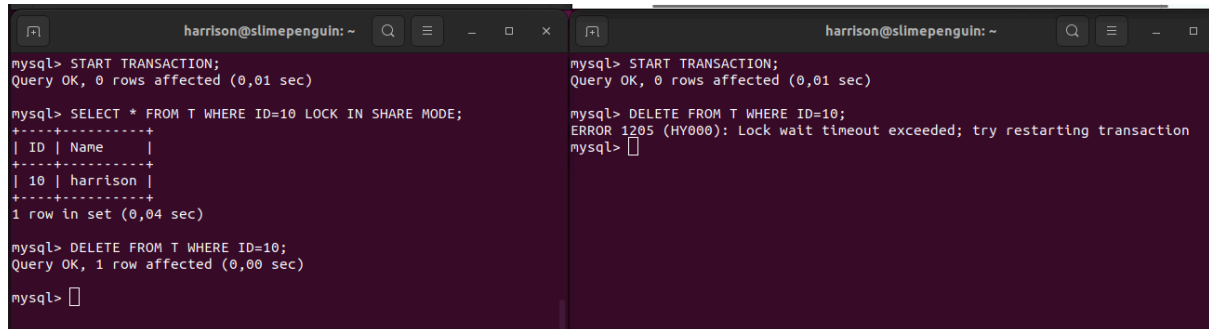


```
harrison@slimepenguin: ~  
mysql> INSERT INTO T VALUES (10, 'harrison');  
Query OK, 1 row affected (0,01 sec)  
  
mysql> INSERT INTO T VALUES (11, 'Harrison');  
Query OK, 1 row affected (1 min 16,68 sec)  
  
mysql>  
  
harrison@slimepenguin: ~  
mysql> LOCK TABLES T WRITE;  
Query OK, 0 rows affected (0,01 sec)  
  
mysql> UNLOCK TABLES;  
Query OK, 0 rows affected (0,22 sec)  
  
mysql>
```

TESTE 8

O que ocorre?

Ocorre que, como a transação da sessão 1 criou um bloqueio do tipo compartilhado em uma linha específica, qualquer transação que não seja ela mesmo não será capaz de realizar nenhum tipo de operação bloqueante ou de escrita naquela linha. Logo, a operação de DELETE da transação da sessão 2 não é capaz de realizar enquanto a transação da sessão 1 não desbloquear a linha de ID=10.



```
harrison@slimepenguin: ~  
mysql> START TRANSACTION;  
Query OK, 0 rows affected (0,01 sec)  
  
mysql> SELECT * FROM T WHERE ID=10 LOCK IN SHARE MODE;  
+-----+  
| ID | Name |  
+-----+  
| 10 | harrison |  
+-----+  
1 row in set (0,04 sec)  
  
mysql> DELETE FROM T WHERE ID=10;  
Query OK, 1 row affected (0,00 sec)  
  
mysql>   
  
harrison@slimepenguin: ~  
mysql> START TRANSACTION;  
Query OK, 0 rows affected (0,01 sec)  
  
mysql> DELETE FROM T WHERE ID=10;  
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction  
mysql> 
```