**MYSQL – RECUPERAÇÃO BASEADA EM LOG E CONTROLE DE CONCORRÊNCIA**

LUCAS LAMOUNIER GONCALVES DUARTE - 2016012688

KEVIN VIEIRA PEREIRA - 2016015385

RODRIGO APARECIDO SILVA MAIA - 2016013095

Itajubá, 9 de maio de 2018

Sumário

[**QUESTÃO 1: RESPONDA AS SEGUINTES PERGUNTAS:** 2](#_Toc513324549)

[a) Explique as técnicas para Tratamento de Deadlock. 2](#_Toc513324550)

[b) O que é bloqueio por granularidade múltipla? 2](#_Toc513324551)

[c) Como a frequência dos pontos de verificação(checkpoint) afeta: 2](#_Toc513324552)

[**QUESTÃO 2: QUAIS SÃO OS TIPOS DE LOG UTILIZADOS NO SGBD? DETALHE A FUNCIONALIDADE DE CADA UM E DESCREVA TAMBÉM INFORMAÇÕES COMO TAMANHO MÁXIMO, LOCAL DE ARMAZENAMENTO. É POSSÍVEL ALTERA ESSAS CONFIGURACÕES? COMO?** 2](#_Toc513324553)

[**QUESTÃO 3: VIMOS EM SALA QUE EXISTEM TRÊS ABORDAGENS DE RECUPERAÇÃO DE TRANSAÇÕES APÓS UMA FALHA NO SISTEMA. COMO O SGBD REALIZA ESSA RECUPERAÇÃO? É POSSÍVEL MUDAR?** 2](#_Toc513324554)

[**QUESTÃO 4: SOBRE O LOG DE RECUPERAÇÃO, É POSSÍVEL ALTERAR A FREQUÊNCIA COM QUE OS CHECKPOINTS SÃO REALIZADOS? O QUE É A OPERAÇÃO DE FLUSHING?** 3](#_Toc513324555)

[**QUESTÃO 5: COMO O SGBD IMPLEMENTA O CONTROLE DE CONCORRÊNCIA?** 5](#_Toc513324556)

[**QUESTÃO 6: AVALIE POR MEIO DE CONSULTAS E DESCREVA A IMPRESSÃO DO GRUPO SOBRE:** 6](#_Toc513324557)

[a) Quais são as consequências de optar por cada um dos níveis de isolamento acima citados? 6](#_Toc513324558)

[b) Como é feita essa configuração no seu SGBD? 6](#_Toc513324559)

[**QUESTÃO DESFIO 1 (VALENDO NOTA EXTRA): EXECUTAR TRANSAÇÕES COM OS DIFERENTES NÍVEIS DE ISOLAMENTO, EM SITUAÇÕES ONDE AS DIFERENTES TRANSAÇÕES ACESSEM O MESMO DADO. VERIFICAR A OCORRÊNCIA DE ERROS.** 7](#_Toc513324560)

[**QUESTÃO DESAFIO 2 (VALENDO NOTA EXTRA): ALTERAR AS CONFIGURAÇÕES DE FLUSH DO SGBD E AVALIAR O TEMPO DE EXECUÇÃO.** 7](#_Toc513324561)

[**REFERÊNCIA:** 8](#_Toc513324562)

# 

# **QUESTÃO 1: RESPONDA AS SEGUINTES PERGUNTAS:**

## Explique as técnicas para Tratamento de Deadlock.

## O que é bloqueio por granularidade múltipla?

## Como a frequência dos pontos de verificação(checkpoint) afeta:

# **QUESTÃO 2: QUAIS SÃO OS TIPOS DE LOG UTILIZADOS NO SGBD? DETALHE A FUNCIONALIDADE DE CADA UM E DESCREVA TAMBÉM INFORMAÇÕES COMO TAMANHO MÁXIMO, LOCAL DE ARMAZENAMENTO. É POSSÍVEL ALTERA ESSAS CONFIGURACÕES? COMO?**

Para se listar os índices de uma tabela

# **QUESTÃO 3: VIMOS EM SALA QUE EXISTEM TRÊS ABORDAGENS DE RECUPERAÇÃO DE TRANSAÇÕES APÓS UMA FALHA NO SISTEMA. COMO O SGBD REALIZA ESSA RECUPERAÇÃO? É POSSÍVEL MUDAR?**

O sistema de recuperação do MySQL é baseado no Log com modificações adiadas, com a utilização de checkpoints, em que todas as transações que estiverem antes dele já foram passadas para a memória secundaria. Quando uma transação foi realizada na memória principal, mas não chegou a atualizar os dados na memória secundária por motivo de falha, então o **Redo** **Log**, quando o sistema foi reiniciado irá executar as transações que atingiram o estado commited. Já as transações que estão com o estado uncommited passaram pelo **Undo Log**, para serem desfeitas.

Depois de uma falha, o sistema primeiro irá realizar o **Redo Log**, e depois iniciará a conexão, garantindo assim que as propriedades de Atomicidade e Durabilidade sejam mantidas intactas mesmo após a falha. Depois disso irá desfazer as transações que não foram completadas, o tempo para desfazer uma transação uncommited pode ser três ou quatro vezes o tempo para realizar a mesma transação caso ela estivesse no estado ativo.

É possível modificar a abordagem de recuperação através do comando **SET GLOBAL** **innodb\_flush\_log\_at\_trx\_commit = 0**, caso seja executado, para cada modificação de dados será atualizado no disco de armazenamento, ou seja, a abordagem de recuperação de Log como modificações adiantadas. Se for utilizado **SET GLOBAL** **innodb\_flush\_log\_at\_trx\_commit = 1**, para cada transação concluída será atualizado no disco de armazenamento. Também pode ser usado o comando **SET GLOBAL** **innodb\_flush\_log\_at\_trx\_commit = 2**, será armazenado na memória principal, as transações que alcançaram o estado commited, e a cada segundo será realizado a passagem das transações para o disco de armazenamento.

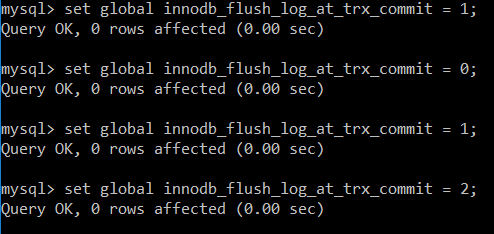


Figura y – que mostra a execução do comando descrito acima

# **QUESTÃO 4: SOBRE O LOG DE RECUPERAÇÃO, É POSSÍVEL ALTERAR A FREQUÊNCIA COM QUE OS CHECKPOINTS SÃO REALIZADOS? O QUE É A OPERAÇÃO DE FLUSHING?**

É possível realizar a alteração da frequência através dos comandos:

* **SET GLOBAL** **innodb\_flush\_log\_at\_trx\_commit = 2,** como foi apresentado anteriormente, nessa configuração será realizado um checkpoint a cada segundo as operações que foram feitas;
* **SET GLOBAL innodb\_flush\_log\_at\_timeout = X,** sendo que **X** pode variar entre 1 e 2700 segundos, ou seja, realizará um checkpoint a cada intervalo de tempo determinado. Caso haja uma falha poderá haver a perda de dados pela quantidade igual de segundos descritos pelo valor escolhido. Por padrão vem definido como 1. Entende-se que este comando funciona em conjunto com o anterior, visto que o comando anterior declara que será baseado no tempo os checkpoints e esse comando o intervalo para cada um.

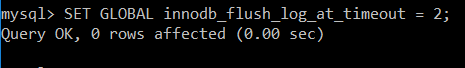


Figura x – que mostra a execução do comando descrito acima

**Flushing** é a operação de sincronizar os dados que estão contidos na memória principal para o banco de dados, ou seja, é a atualização da base de dados para que ela esteja de acordo com o que se encontra na memória principal, em que está pode estar diferente devido a transações. Pode ocorrer de maneira implícita através de de um commit caso a abordagem de modificações adiadas ou após cada modificação caso seja modificação imediata, ou também pode ocorrer de forma explicita através do comando **FLUSH TABLE nomeDaTabela;**



Figura x – que mostra a execução do comando Flush, as tabelas city, country e countrylanguage do banco de dados world

# **QUESTÃO 5: COMO O SGBD IMPLEMENTA O CONTROLE DE CONCORRÊNCIA?**

O MySQL implementa o controle de concorrência baseado no protocolo de bloqueio em duas fases, em que se adquire todos os Locks necessários no começo de cada transação e conforme o Lock já foi usado será realizado o Unlock da tabela. Se a transação necessitar aplicar Locks para mais de uma tabela, basta apenas colocar os locks na mesma ordem que a transação utilizará as tabelas.

Os comandos que o MySQL permite para a utilização de locks estão listados abaixo:

**Lock Table nomeDaTabela Read** - comando do Lock usado para apenas para leitura, múltiplas transações podem acessar a mesma tabela para a operação de leitura.

**Lock Table nomeDaTabela Write** – comando para se utilizar o lock de escrita (também pode se realizar a leitura com esse lock), apenas uma transação poderá ter esse lock para uma dada tabela por vez(caso a tabela tenha foreign keys, a tabela referenciada também receberá o lock), sendo necessário as outras transações que desejem escrever nessa tabela esperem até o termino dessa transação.



Figura x – que mostra a execução dos comandos de lock de escrita e do lock de leitura, para o banco de dados world

**Unlock tables** - é o comando responsável por liberar as tabelas, que receberam previamente um lock.



Figura x – que mostra a execução do comando unlock

As transações executadas no MySQL podem sofrer de deadlock, sendo que ele afeta principalmente a performance, mas o InnoDB consegue detectar automaticamente e lidar com eles, aplicando um rollback em uma das transações afetadas. É possível desabilitar o sistema que identifica deadlocks para que se ganhe performance, é recomendado para um sistema com alta concorrência, é possível através do comando SET GLOBAL innodb\_deadlock\_detect = OFF. Caso seja aplicado o comando anterior, o sistema contará com o innodb\_lock\_wait\_timeout, em que caso se passe 50 segundos e o deadlock continuar, será executado um rollback em uma das transações que estão no deadlock.

O SGBD possui uma política para diminuir a incidência de deadlocks. Primeiro ele classifica as tabelas em uma ordem para ser aplicados os locks. Depois se a tabela for receber um lock de leitura e um de escrita, o lock de escrita será passado à frente do lock de leitura. E por fim será aplicado os locks de cada tabela, até a sessão ter todos os locks necessário.

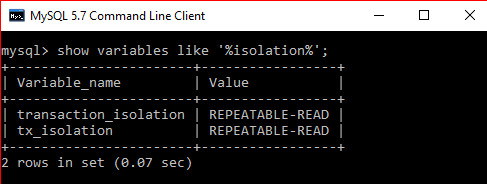
# **QUESTÃO 6: EM GERAL, OS SGBDS SÃO INSTALADOS COM UM NÍVEL DE ISOLAMENTO PADRÃO, MAS DISPONIBILIZAM UMA VARIÁVEL DE AMBIENTE ONDE É POSSÍVEL CONFIGURAR ESSE NÍVEL DE ISOLAMENTO. ASSIM SENDO, RESPONDA:**

## Quais são as consequências de optar por cada um dos níveis de isolamento acima citados?

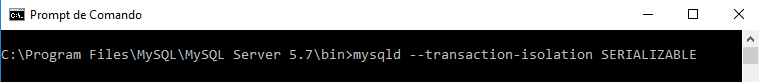
* **Repeatable read** – Este nível de isolamento é o padrão da engine InnoDB e garante que a mesma leitura através do SELECT se repita, ou seja, exibe o mesmo resultado para diferentes execuções em uma mesma transação, dessa forma impede a leitura fantasma e a leitura suja e garante também uma leitura repetitiva. A leitura fantasma nesse nível também é evitada diferindo do padrão SQL.
* Read Commited – Este nível de isolamento permite que uma dada transação possa ler e utilizar dados já commitados por outras transações, no entanto não será possível ver dados ainda não commitados. No nível de Read Commited as leituras sujas ainda são evitadas, porém pode ocorre leituras fantasmas e ela não garante uma leitura repetitiva.
* Read Uncommited – Este nível permite que as transações possam ler e manipular dados não commitados pelas demais transações, permitido que ocorra leituras suja, leituras fantasmas e leituras não repetitivas.
* Serializable – Este nível isola completamente as transações uma das outras evitando a leitura fantasma e a leitura suja e garantido também uma leitura repetitiva.

## Como é feita essa configuração no seu SGBD?

No MySQL o nível de isolamento pode ser configurado através do arquivo “my.ini” ou pelo executável “mysqld”. Para ver o nível que o banco está operado pode ser usado na interface do MySQL o comando “SHOW variables LIKE ‘%isolation%’”



Para alterar via “mysqld” o nível de isolamento apenas é necessário executar o comando “mysqld --transaction-isolation SERIALIZABLE”



Pode ser feita alteração adicionando ao arquivo “my.ini” a linha transaction\_isolation = {READ-UNCOMMITTED | READ-COMMITTED | REPEATABLE-READ | SERIALIZABLE}, caso a linha já exista no arquivo altere apenas o nível de isolamento.

# **QUESTÃO DESFIO 1 (VALENDO NOTA EXTRA): EXECUTAR TRANSAÇÕES COM OS DIFERENTES NÍVEIS DE ISOLAMENTO, EM SITUAÇÕES ONDE AS DIFERENTES TRANSAÇÕES ACESSEM O MESMO DADO. VERIFICAR A OCORRÊNCIA DE ERROS.**

Uma transação pode ser necessária quando um conjunto de instruções SQL devem ser considerados como uma única unidade de trabalho. Ou seja, se toda aquela transição ocorrer você pode persistir a mudança no banco, caso uma delas falhe, você deve refazer todos os passos já realizados e recomeçar o conjunto de instruções.

# **QUESTÃO DESAFIO 2 (VALENDO NOTA EXTRA): ALTERAR AS CONFIGURAÇÕES DE FLUSH DO SGBD E AVALIAR O TEMPO DE EXECUÇÃO.**

O chamado SQL Injection é um tipo de ataque que se baseia na manipulação do código SQL. Quando uma aplicação recebe um valor digitado pelo usuário e concatena isso a uma string com código SQL para uma consulta em um banco de dados, a maneira que esta string é concatenada pode gerar uma abertura para que partes de código SQL sejam enviadas para a variável que coleta os dados digitados pelo usuário e enviem para manipular a consulta pré-determinada no momento em que a concatenação é feita.

# **REFERÊNCIA:**

ORACLE: **MySQL 5.7 Reference Manual**, 2018. Disponível em: < https://dev.mysql.com/doc/refman/5.7/en/storage-engines.html >. Acesso em 05/05/2018.