
# Recuperação Baseada em Log e Controle de Concorrência

O objetivo do tutorial é demonstrar como funciona Recuperação Baseada em Log e Controle de Concorrência no MySQL.

Carlos Henrique Reis - 30415
Mateus Henrique Toledo - 34849
Victor Rodrigues da Silva - 31054


**Questão 1:** Responda as seguintes perguntas:

**a)** Explique as técnicas para Tratamento de Deadlock.

Existem três técnicas para o tratamento do Deadlock. Elas são “Ignorar a situação”, “Detectar o deadlock e recuperar o sistema”, ou ainda evitar o deadlock.

**Ignorar a situação:** Essa técnica é apelidada de “Algoritmo do avestruz”. Nessa técnica, caso haja algum deadlock, ele simplesmente é ignorado.

**Detectar o Deadlock e recuperar o sistema:** Nessa, o procedimento de recuperação é executado após a ocorrência do deadlock.

Na detecção do deadlock, uma estrutura de dados é implementada para o armazenamento das informações sobre os processos e os recursos alocados a eles. A estrutura de dados é atualizada dinamicamente para que a mesma reflita a real situação de cada processo/recurso no sistema. Porém, essa atualização dinâmica gera uma sobrecarga no sistema, pois as estruturas precisam ser atualizadas todas as vezes que um processo aloca, libera ou requisita um recurso. Também há a possibilidade de uma outra sobrecarga, resultada das verificações de esperar circular nas estruturas de dados para a efetiva detecção do deadlock, feita pelo banco. Assim, quando um deadlock é detectado, um procedimento de recuperação é executado.

**b)** O que é bloqueio por granularidade múltipla?

O bloqueio permite a uma transação impedir que outras acessem ou atualizem registros de forma a evitar os problemas de concorrência. No bloqueio por granularidade múltipla, ao invés do bloqueio de um item de dados, é possível bloquear tuplas, tabelas, blocos de disco ou bancos de dados. Para a escolha da granularidade do bloqueio é possível a utilização de árvores, sendo que cada nó pode ser bloqueado individualmente e cada nível corresponde a uma granularidade de bloqueio.

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

**a.** O desempenho do sistema quando não ocorre uma falha


Quanto maior a quantidade de checkpoints maior será o tempo de execução das transações quando não ocorre uma falha, pois a cada checkpoint o SGBD pára a execução das transações, verifica as operações de write declaradas antes do checkpoint, persiste os dados das declarações write e retoma a execução das transações.

**b.** O tempo gasto para a recuperação de uma falha

Quanto maior a quantidade de checkpoints mais rápido será a recuperação de uma falha, pois o checkpoint garante que os dados estão consistentes e já estão persistidos sem nenhum erro.

**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 alterar essas configurações? Como?

O MySQL tem vários arquivos de log diferentes que podem ajudá-lo a descobrir o que está acontecendo dentro do mysqld:

| **Log file** | **Description** |
|---|---|
| O log de erros | Problemas encontrados iniciando, executando ou parando o processo do MySQL |
| O log isam | Documenta todas alterações a tabelas ISAM. Usado somente para depuração do código isam. |
| O log de consultas | Conexões estabelecidas e consultas executadas. |
| O log de atualizações | Desatualizado: Armazena todas as instruções que alteram dados. |

| | |
|---|---|
| O log binário | Armazena todas as instruções que alteram qualquer coisa. Usada também para replicação. |
| O log para consultas lentas | Armazena todas queries que levaram mais de long_query_time segundos para executar ou que não usaram índices. |

Todos logs podem ser encontrados no diretório de dados do mysqld. Você pode forçar o mysqld a reabrir os arquivos de log (ou em alguns casos trocar para um novo log) executando `FLUSH LOGS`.

Se o mysqld finaliza inesperadamente e o mysqld_safe precisar reiniciar o mysqld, mysqld_safe gravará uma linha restarted mysqld neste arquivo. Este log também guarda um aviso se o mysqld notificar uma tabela que precisa ser automaticamente verificada ou reparada.

Em alguns sistemas operacionais, o log de erro irá conter registros de pilha de onde o mysqld finalizou. Isto pode ser usado para saber onde e como o mysqld morreu.

É possível especificar onde o mysqld armazena o arquivo de log de erro com a opção --log-error[=filename]. Se nenhum nome de arquivo for dado, o mysqld usará mysql-data-dir/'maquina'.err no Unix e \mysql\data\mysql.err no Windows.i Se você executar flush logs o arquivo antigo terá o prefixo --old e o mysqld criará um novo arquivo de log vazio.

Se você não especificar –log-error ou se você utilizar a opção --console, o erro será escrito em stderr (o terminal).

Se forem utilizadas as opções --log ou -l, o mysqld escreve um log geral com o nome de arquivo nome_máquina.log, e o reinicio e a recarga não geram um novo arquivo de log (embora ele seja fechado e reaberto). Neste caso você pode copiá-lo (no Unix) usando:

```
mv nome_máquina.log nome_máquina-antigo.log
mysqladmin flush-logs
```

```
cp nome_máquina-antigo.log para-diretório-backup
rm nome_máquina-antigo.log
```

O log de atualização é inteligente pois registra somente instruções que realmente alteram dados. Portanto, um `UPDATE` ou um `DELETE` com uma cláusula `WHERE` que não encontre nenhum registro não é escrito no log. Ele salta até instruções `UPDATE` que atribui a uma coluna o mesmo valor que ela possuia.

O registro da atualização é feito imediatamente após uma consulta estar completa mas antes que as bloqueios sejam liberados ou que algum commit seja feito. Isto garante que o log seja escrito na ordem de execução.

Se você desejar atualizar um banco de dados a partir de arquivos de logs de atualização, você pode fazer o seguinte (assumindo que seus logs de atualização estejam nomeados na forma nome_arquivo.###):

```
shell> ls -1 -t -r nome_arquivo.[0-9]* | xargs cat | mysql
```

ls é utilizado para obter todos os arquivos de log na ordem correta.

Isto pode ser útil se você tiver que recorrer a arquivos de backup depois de uma falha e desejar refazer as atualizações que ocorreram entre a hora do backup e a falha.

Quando iniciado com a opção --log-bin[=nome_arquivo], o mysqld escreve um arquivo de log contendo todos comandos SQL que atualizam dados. Se nenhum arquivo for fornecido, ele aponta para o nome da máquina seguido de -bin. Se for fornecido o nome do arquivo, mas ele não tiver o caminho, o arquivo é escrito no diretório de dados.

Se você fornecer uma extensão à --log-bin=nome_arquivo.extensão, a extensão será removida sem aviso.

O mysqld irá acrescentar uma extensão ao nome de arquivo do log binário que é um número que é incrementado cada vez que mysqladmin refresh, mysqladmin flush-logs, a instrução `FLUSH LOGS` forem executados ou o servidor

for reiniciado. Um novo log binário também será automaticamente criado quando o tamanho do log atual alcançar `max_binlog_size`. Nota se você estiver usando transações: uma transação é escrita em um bloco no arquivo de log binário, já que ele nunca é separado entre diversos logs binários. Desta forma, se você tiver grnades transações, você pode ter logs binários maiores que `max_binlog_size`.

**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 MySQL é iniciado com todos os logs desabilitados por padrão. O usuário deve definir que tipos de log ele deseja habilitar, o caminho dos logs e até o nome do arquivo de determinado tipo de log, podendo selecionar entre os tipos de log listados na **questão 2**.

→ **Habilitar logs:**

O MySQL disponibiliza em sua documentação alternativas para habilitar os logs no SGDB. As duas maneiras mais simples são setar variáveis de sistema com 1 e 0, ON e OFF ou iniciar o mysqld com a opção de determinado log habilitado.

→ **Executar logs:**

É possível forçar o servidor a abrir ou fechar os arquivos de log com o seguinte comando:

```
FLUSH LOGS;
```

Também é possível iniciar o mysqladmin com a opção flush-logs, da seguinte forma:

```
mysqladmin -u root -p flush-logs;
```

**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?

O InnoDB implementa um mecanismo de ponto de verificação chamado fuzzy checkpoint. O InnoDB descarregará páginas de banco de dados modificados da área de buffer em pequenos grupos. Não há necessidade de descarregar a área de buffer em um único grupo, o que iria, na prática, para o processamento da instrução SQL do utilizador por um instante.

A declaração FLUSH limpa ou recarrega várias caches internas usadas pelo MySQL. Para executar FLUSH, você deve ter o privilégio de RELOAD. A declaração RESET é semelhante a FLUSH. Você não pode emitir uma instrução FLUSH de dentro de uma função armazenada ou um gatilho. Ao fazê-lo dentro de um procedimento armazenado é permitida, desde que ele não seja chamado por uma função armazenado ou gatilho. Por padrão, as declarações FLUSH são escritas no log binário e será replicado. A palavra-chave NO_WRITE_TO_BINLOG (LOCAL é um apelido) irá garantir que a instrução não está escrita no log binário.

**Questão 5:** Como o SGBD implementa o controle de concorrência?

A estratégia do MySQL com a engine InnoDB é o chamado bloqueio de chave seguinte. Basicamente o SGBD bloqueia os registros de índice e a lacuna antes dos registros de índices para bloquear que outros usuários realizem inserções antes do registro de índice.

O isolamento no innoDB segue o padrão `REPEATABLE READ`. Isso faz com que se houver mais de uma leitura sobre uma mesma tabela em determinada transação essa retorne resultados iguais aos da primeira consulta. Apenas a

instrução de `INSERT` (inclusão de novos registros) altera os dados na segunda consulta.

**Questão 6:** A SQL Padrão define quatro tipos de níveis de isolamento:
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:

**a)** Quais são as consequências de optar por cada um dos níveis de isolamento acima citados?

- **Read uncommitted**

  Também é chamada de leitura suja. `SELECTs` sem bloqueio são realizados de forma a não procurar uma versão mais nova do registro. Esse nível de isolamento faz com que as leituras sejam inconsistentes.

- **Read committed**

  Nível de isolamento parecido com o que o SGBD Oracle utiliza. Todas as instruções `SELECT ... FOR UPDATE` e `SELECT ... LOCK IN SHARE MODE` só travam o registro de índice, não a lacuna antes dele, e assim permitem livremente a inserção de novos registros próximo ao registro travado.

  Em instruções `UPDATE` e `DELETE`, o InnoDB deve definir o bloqueio da chave seguinte ou da lacuna e bloquear inserções feitas por outros usuários nas lacunas cobertas pela faixa. Isto é necessário já que deve se bloquear "linhas fantasmas" para a replicação e recuperação no MySQL funcionarem. Leituras consistentes (Consistent reads) se comportam como no Oracle: cada leitura consistente, mesmo dentro da mesma transação, configura e lê a sua própria cópia recente.

- **Repeatable read**

  Este é o nível de isolamento padrão do InnoDB. `SELECT ... FOR UPDATE`, `SELECT ... LOCK IN SHARE MODE`, `UPDATE`, e `DELETE` que utilizam um índice único com uma condição de busca única, travam apenas o registro de

índice encontrado, e não a lacuna antes dele.

De outra forma estas operações empregam bloqueio de registro seguinte, bloqueando a faixa de índice varrida com trava de chave seguinte ou de lacuna e bloqueando novas inserções feitas por outros usuários. Em leituras consistentes (consistent reads) existe uma diferença importante do nível de isolamento anterior: neste nível todas as leituras consistentes dentro da mesma transação lêem a mesma cópia estabelecida pela primeira leitura. Esta conversão significa que se você executa diversas `SELECTs` dentro da mesma transação, elas também são consistentes entre elas.

- **Serializable**

Segue o mesmo padrão que as transações Repeatable read, exceto que todos os `SELECTs` são convertidos de forma implícita para `SELECT ... LOCK IN SHARE MODE`.

**b)** Como é feita essa configuração no seu SGBD?

É possível definir os níveis de isolamento padrão para todas as conexões da seção do mysql pelo mysql.cnf, da seguinte forma:

```
transaction-isolation = {READ-UNCOMMITTED | READ-COMMITTED |
REPEATABLE-READ | SERIALIZABLE}
```

É possível alterar o nível de isolamento de uma única seção ou de todas as próximas seções com a instrução sql, da seguinte forma:

```
SET [SESSION | GLOBAL] TRANSACTION ISOLATION LEVEL {READ UNCOMMITTED | READ COMMITTED
| REPEATABLE READ | SERIALIZABLE}
```

Para definir o nível de isolamento é necessário o privilégio SUPER. É possível consultar o nível de isolamento das transações da seguinte forma:

```
SELECT @@global.tx_isolation;SELECT @@tx_isolation;
```

**Questão Desafio 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.

**Questão Desafio 2 (valendo nota extra):** Alterar as configurações de flush do SGBD e avaliar o tempo de execução.