Resumo: Capítulo 18 - Recuperação de Falhas

Andréia Kondo, Cléo Billa

8 de junho de 2005

O gerenciador de recuperação de falhas de um SGBD é responsável por assegurar duas importantes propriedades das transações: **atomicidade** e **durabilidade**. Ele garante atomicidade desfazendo as ações das transações que não foram validadas (*commit*). E garante a durabilidade assegurando que todas as ações de uma transação validada sobreviverão a uma falha no sistema.

A seção 1 começa introduzindo o algoritmo de recuperação ARIES. A seção 2 dicute sobre o log, que é uma estrutura de dados importante na recuperação de falhas. Na seção 3 são abordadas outras estruturas de dados relacionadas à recuperação. A seção 4 apresenta o protocolo *Write-Ahead Logging* (WAL). A seção 5 aborda o *checkpoint*. Na seção 6 é mostrada como é feita a recuperação de falhas. A seção 7 discute sobre as falhas na mídia.

1 Introdução ao ARIES

ARIES é um algoritmo desenvolvido para trabalhar com uma abordagem **roubar**, **não-forçar**. Ele é dividido em três passos:

1. **Análise:** Identifica as páginas sujas do *buffer-pool* e as transições ativas no momento da falha.
2. **Refazer:** Refaz todas as ações a partir de um ponto específico do log.
3. **Desfazer:** Desfaz as ações das transações que não foram validadas.

Existem três princípios básicos por trás do ARIES:

1. **WAL(** *Write-Ahead Logging*)**:** Significa escrita antecipada no log, ou seja, qualquer mudança em um objeto do banco de dados é primeiro gravada no log.
2. **Repetição da História:** ARIES refaz todas as ações necessárias para deixar o sistema exatamente igual ao momento que houve a falha. Então desfaz as ações das transações não validadas, abortando-as assim de fato.

3. **Muda o log durante o desfazer:** Mudanças feitas no banco de dados por operações de desfazer são gravadas no log para assegurar que uma ação não se repita no caso de haver uma nova falha.

O segundo ponto acima diferencia o ARIES de outros algoritmos de recuperação de falhas e é a base de sua simplicidade e flexibilidade. O segundo e o terceiro ponto também são importantes para lidar com operações de refazer e desfazer onde elas não são exatamente o inverso uma da outra.

2 Log

O log é o histórico das ações executadas pelo SGBD. Na verdade, o log é um arquivo com diversas cópias que são mantidas em diversos discos, o que reduz a chance de ele ser perdido. A parte mais recente do log é mantida na memória principal e é periodicamente forçada a ser gravada no disco.

Cada registro de log contém um campo denominado LSN - *Log Sequence Number*. Normalmente o log é um arquivo seqüencial, que cresce indefinidamente, nesse caso o LSN pode ser simplesmente o endereço do primeiro *byte* do registro.

Para auxiliar na recuperação de falhas, cada página do banco de dados contém o LSN do registro de log mais recente que alterou a página, essa informação é armazenada em `pageLSN`.

Um registro de log é gravado para cada ação descrita a seguir:

Atualização: Após modificar a página, um registro de log do tipo `atualização` é adicionado ao final do log. O `pageLSN` da página é modificado para o LSN do novo registro do log.

Validação: Quando uma transição é validada, um registro de log do tipo `validação` com a identificação da transação é gravado. Quando esse tipo de registro de log é identificado, força-se o log a ser gravado em disco. A transação é considerada válida no momento que esse registro de log é gravado. Alguns passos extras devem ser feitos, como retirar a transação da tabela de transações, etc.

Abortar: Quando uma transação é abortada, é gravado no log um registro do tipo `abortar` que contém a identificação da transação. Os passos para desfazer as ações da transação são feitas logo depois o registro de log ser gravado.

Fim: Como foi mostrado acima, quando uma transação é abortada ou validada, ainda existem algumas ações que o SGBD deve realizar após o registro de log ser gravado. O registro de log do tipo `fim` é gravado após essas ações serem completadas.

CLR: Um registro do tipo `clr` - *Compesation Log Record* é adicionado ao log quando alguma ação de atualização é desfeita.

Todos os registros de log contém alguns campos: `prevLSN`, `transID` e `type`. O conjunto de ações de uma transação é mantido através de uma lista encadeada que volta no tempo, através do campo `prevLSN`. O campo `transID` identifica a transação e o campo `type` indica qual o tipo do registro de log. Esses tipos foram os discutidos acima.

Dependendo do tipo do registro de log alguns campos são adicionados. Alguns deles serão discutidos a seguir.

2.1 Registros de Log de Atualização

Um registro de atualização possui alguns campos, além dos já citados. O campo `pageID` é a identificação da página que foi alterada; o tamanho em *bytes* e o *offset* também são armazenados. `before-image` contém o valor dos *bytes* alterados antes da alteração, já `after-image` é o valor após a mudança. Esses campos são necessários para que as ações de refazer e desfazer possam ser realizadas.

2.2 CLR - *Compesation Log Records*

Um CLR é adicionado ao log um pouco antes que uma operação de desfazer é realizada. O CLR descreve qual a operação feita para desfazer uma ação de atualização e é adicionada ao final do log assim como qualquer outro registro de log. O CLR contém o campo `undoNextLSN` que é o LSN do próximo registro de log que deve ser desfeito para abortar a transação.

Ao contrário de um registro de atualização, um CLR define uma ação que nunca será desfeita, ou seja, nunca se desfaz uma ação de desfazer. Isso acontece porque um registro de atualização descreve uma mudança feita por uma transação durante uma execução normal e a transação pode ser abortada mais tarde, por outro lado um CLR descreve uma ação para que a transação seja desfeita e a decisão da transação ser abortada já foi feita.

Um CLR pode ter sido gravado em disco, mas a ação que ele descreve não foi gravada em disco ainda, quando o sistema falha novamente. Nesse caso a ação descrita no CLR é realizada na fase de "refazer", assim como as ações descritas em um registro de atualização. Por isso, o CLR necessita de informações para refazer as ações, mas não para desfazê-la.

3 Outras Estruturas Relativas A Recuperação de Falhas

Além do log, as duas tabelas descritas a seguir contém informações importantes para a recuperação de falhas:

Tabela de Transações: Essa tabela contém uma linha para cada transação ativa, cada linha é composta pela identificação da transação, o seu *status* e um campo chamado `lastLSN` que aponta para o último registro de log da transação correspondente. O *status* pode ser executando, validada ou abortada.

Tabela de Páginas Sujas: Essa tabela contém uma linha para cada página suja do *buffer-pool*. Cada linha contém, além de uma identificação da página, um campo `recLSN` que aponta para primeiro registro no log que altera a página.

Durante uma execução normal, essas tabelas são mantidas pelo gerenciador de transações e pelo gerenciador de *buffer* e após uma falha essas tabelas são reconstruídas durante a fase de análise.

4 WAL - *Write-Ahead Logging*

Antes de uma página ser escrita em disco, todos os registros de atualização devem ser escritos em memória não volátil.

WAL é regra fundamental que assegura que o registro de todas as mudanças no banco de dados estão disponíveis durante uma recuperação de falha. Se uma transação é validada não há garantias de que ela foi realmente salva em disco antes de uma falha. Sem o registro dessas alterações algumas operações podem ser perdidas.

Assim que uma transação é validada o log é escrito em disco, mesmo que uma abordagem `não-forçada` esteja sendo utilizada. Isso é melhor do que uma abordagem `forçada`, porque o log é muito menor do que o tamanho de uma página.

5 Checkpointing

Um *checkpoint* é como uma fotografia do sistema, e ao fazer *checkpoints* periodicamente, o SGBD reduz a quantidade necessária de trabalho durante a recuperação de uma falha.

No ARIES o *checkpointing* tem três passos. Primeiro, um `begin_checkpoint` é gravado no log. Segundo, um `end_checkpoint` é construído, incluindo a tabela de transações e a tabela de páginas sujas, e depois é gravado no log. O terceiro passo é armazenar em um lugar conhecido o `master record` contendo o LSN do `begin_checkpoint`. Enquanto o `end_checkpoint` está sendo construído, o SGBD continua executando transações e escrevendo outros registros de log, portanto as tabelas descritas nele retratam o estado do SGBD quando o `begin_checkpoint` foi gravado.

Esse tipo de `checkpoint`, chamada de `fuzzy checkpoint`, tem custo praticamente zero porque ele não congela o sistema ou fica gravando páginas em disco. Por outro lado, sua efetividade é limitada pelo `recLSN` mais novo salvo, porque durante a fase de refazer, o sistema irá começar exatamente pelo registro de log apontado por esse `recLSN`. Um processo que roda em segundo plano e fica gravando páginas sujas em disco pode ajudar a limitar esse problema.

Quando o sistema volta a executar após uma falha, ele começa a partir do último `checkpoint` gravado. Por motivos de uniformidade, o sistema sempre começa uma execução normal a partir de um `checkpoint`, onde a tabela de transações e a tabela de página suja estão vazias.

6 Recuperação de uma Falha no Sistema

Quando o sistema é reiniciado após uma falha no sistema, o gerenciador de recuperação procede em 3 fases, conforme é ilustrado na Figura 1:

- Fase de Análise: inicia no mais recente *checkpoint*, em que na Figura 1 é denotado por C, e segue examinando até o último registro de log.
- Fase de Refazer: segue a fase de análise e refaz todas as alterações realizadas em alguma determinada página. Esta fase inicia do menor *recLSN* (ilustrado na Figura 1 como B) de uma determinada página suja, que é determinado durante a fase de análise, e segue até o fim dos registros de log.
- Fase de Desfazer: defaz todas as ações das transações ativas no momento da falha. Na Figura 1 pode-se notar que esta fase segue o fluxo na ordem inversa, ou seja, de trás pra frente no registro de log.

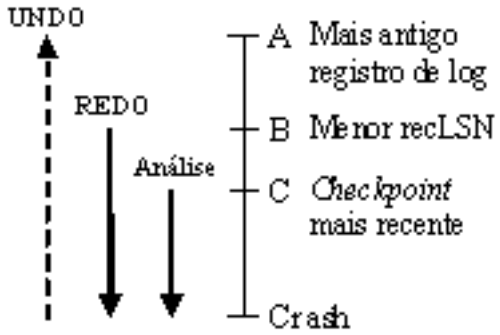

Figura 1: Fases do Algoritmo ARIES.

Nas seções a seguir cada fase do Algoritmo de recuperação será abordada com mais detalhes. Além disso, um exemplo de recuperação de falha será apresentado na seção 6.4.

6.1 Fase de Análise

A fase de análise possui três tarefas:

1. Determina o ponto no log em que a fase de Refazer inicia.

2. Determina as páginas que estavam sujas no momento em que ocorreu a falha.

3. Identifica as transações que estavam ativas no momento da falha e que devem ser desfeitas.

A análise começa no *checkpoint* mais recente e percorre o registro de log, criando a tabela de transações e a tabela de páginas sujas, até encontrar o fim do log. Esta fase ocorre da seguinte forma:

- se um registro de "fim"para uma determinada transação T for encontrado, então T deve ser removida da tabela de transações.

- se um registro de log, exceto um registro de "fim", para uma transação T for encontrado, então T deve ser adicionada à tabela de transações. Com esta adição, é necessário atualizar o campo *lastLSN* atribuindo-o com o *LSN* desse registro de log. Além disso, se esse registro for de "commit", então o estado de T é C, senão é atribuído U, indicando que essa transação é para ser desfeita.

- se uma página foi modificada e ela não pertencente à tabela de páginas sujas, então esta página deve ser adicionada a esta tabela. Além disso, o *recLSN* deve ser igual ao *LSN* do registro de log. O *recLSN* identifica o registro de log que fez com que essa página tornasse suja.

6.2 Fase Refazer

Durante a fase Refazer, o algoritmo de ARIES reaplica as atualizações de todas as transações. Além disso, se antes da falha no sistema uma determinada transação abortou e seus *updates* foram desfeitos, como indicados pelos CLRs, as ações descritas nos CLRs também serão reaplicadas. Esse paradigma de repetição de histórico é o que diferencia o ARIES dos outros algoritmos de recuperação baseados no WAL.

A fase Refazer inicia no menor recLSN de todas as páginas na tabela de páginas sujas, construída na fase de Análise, devido a esse registro de log identificar a atualização mais antiga que pode não ter sido gravada no disco antes da falha. A partir desse registro, a fase Refazer percorre até o fim do log verificando as ações que devem ser refeitas. Cada LSN de CLR ou registro de update deve ser refeito exceto quando ocorre as seguintes condições:

- A página afetada não está na tabela de páginas sujas;

- A página afetada está na tabela de páginas sujas, mas o *recLSN* é maior do que o *LSN* do registro verificado;

- O *pageLSN* é maior ou igual ao *LSN* do registro de log verificado.

A primeira condição significa que todas as modificações para essa página foram gravadas no disco. A segunda quer dizer que as atualizações verificadas foram realmente propagadas no disco. A terceira condição garante que a atualização verificada foi gravada no disco, devido a última atualização para uma página ter sido gravada em disco.

Se uma ação deve ser refeita então:

- As ações gravadas no log devem ser reaplicadas.
- O campo *pageLSN* da página é atualizado com o *LSN* do registro de log refeito. Nenhum registro de log adicional é escrito neste momento.

6.3 Fase Desfazer

A fase Desfazer, diferentemente das outras fases, percorre o log de trás para frente. O objetivo desta fase é desfazer as ações de todas as transações ativas no momento da falha. Esse conjunto de transações ativas é gerado pela fase de Análise. Estas transações são denominadas transações perdedoras, e todas as ações de perdedoras devem ser desfeitas.

Considere que o conjunto dos campos *lastLSN* de todas as transações perdedoras é chamado de *ToUndo*. O algoritmo de Desfazer repete os passos mencionados abaixo até que o conjunto *ToUndo* esteja vazio.

1. Escolhe o maior dos LSNs que estão no conjunto *ToUndo*.
2. Se este LSN é um CLR e *undoNextLSN* é igual a *NULL*, isso significa que essa transação já foi desfeita então deve ser gravado um registro de fim e o CLR descartado.
3. Se este LSN é um CLR e *undoNextLSN* é diferente de *NULL* então é necessário adicionar o valor de *undoNextLSN* ao conjunto *ToUndo*.
4. Se é um registro de atualização, então um CLR deve ser escrito e a ação correspondente deve ser desfeita. Além disso, o valor de *prevLSN* deve ser adicionado ao conjunto *ToUndo*.

6.4 Um Exemplo de Recuperação de Falha

Nesta seção, será discutido um exemplo de recuperação de falhas, ilustrado na Figura 2. Esse exemplo mostra como abortar uma transação é um caso especial de Desfazer e como o uso de CLRs garante que a ação Desfazer não seja aplicada duas vezes em um registro de atualização.

O log, ilustrado na Figura 2, identifica a ordem em que as ações são executadas, observe que os LSNs estão em ordem crescente e cada registro de log para uma transação possui o campo *prevLSN* que aponta para os registros de log anteriores.

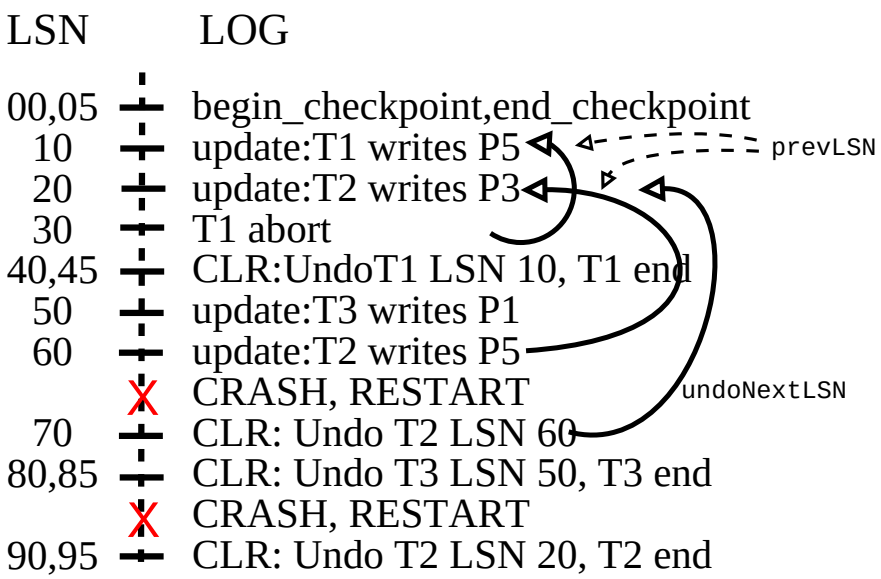

Figura 2: Exemplo de Recuperação de Falha.

Na Figura 2, o registro de log 30 indica que T1 abortou. Quando isso acontece todas as ações referentes a esta transação devem ser desfeitas, na ordem inversa. Neste exemplo, a única ação de T1 que será desfeita é o registro de atualização 10. Isto é indicado pelo CLR no LSN 40.

Na primeira falha, o sistema reinicia e começa o processo de recuperação partindo da fase de análise. A fase de análise identifica as páginas sujas P1 (com *recLSN* 50), P3 (com *recLSN* 20) e P5 (com *recLSN* 10). Esta fase também identifica as trasanções que estavam ativas no momento da falha, T2 (com *lastLSN* 60) e T3 (com *recLSN* 50). A transação T1 já tinha sido completada no registro 45, dessa forma, ela não faz parte da tabela de transações.

A fase Refazer inicia com o registro de log 10, que é o menor *recLSN* presente na tabela de páginas sujas, e percorre até fim do registro de log reaplicando todas as ações (para os registros de atualização e CLR).

A fase Desfazer inicia gerando o conjunto *ToUndo*, que consiste dos LSNs 60, para T2, e 50, para T3. O algoritmo começa sua execução com o maior LSN desse conjunto, que é o LSN 60. Esse registro é desfeito e um CLR (com LSN 70) é escrito no log. Como este CLR tem *undoNextLSN* igual a 20 (como pode ser visto na Figura 2), que é o campo *prevLSN* do registro de log 60, esta transação não pode ser finalizada pois o registro 20 ainda deve ser desfeito. Agora, o maior *LSN* no conjunto *ToUndo* é o 50. Então a escrita correspondente ao registro 50 é desfeita e um CLR (com *LSN* 80) é escrito descrevendo esta modificação. Como o *undoNextLSN* de 50 é igual a *null* então também é escrito um registro de fim para esta transação (*LSN* 85) no log.

Quando ocorre a segunda falha no sistema, como ilustrada na Figura 2, o sistema reinicia e começa a fase de análise novamente. Nesta fase, a tabela de páginas sujas

será idêntica a anterior ao reinício e a tabela de transações terá apenas a transação T2, devido a T3 ter finalizado no registro 85.

A fase Refazer refaz as ações desde o registro 10 até o 85.

A fase Desfazer considera apenas o LSN 70, como presente no conjunto *ToUndo*, processa ele e insere o valor de *undoNextLSN* (20) no conjunto *ToUndo*. Depois disso, o registro 20 é desfeito e escrito um CLR (com LSN 90). Como não há mais nenhuma ação de T2, um registro de fim é gravado no log com LSN 95.

Agora, o conjunto *ToUndo* está vazio, então a recuperação está completa e a execução normal pode continuar com a escrita de um registro de *checkpoint*.

Este exemplo ilustrou uma falha durante a fase de Desfazer. Para completar, vamos considerar o que acontece se ocorrer uma falha na fase de Análise ou na fase de Refazer. Se uma falha ocorre durante a fase de análise, todo o trabalho feito nesta fase é perdido. Se uma falha ocorre na fase de Refazer, algumas modificações podem ter sido gravadas no disco antes da falha, então quando o sistema reinicia, ele começa novamente a partir da fase de análise. Depois segue para a fase de refazer, em que alguns registros de atualização que foram refeitos na primeira vez, não serão refeitos novamente, pois o *pageLSN* é igual ao *LSN* do registo de atualização.

7 Recuperação de Mídia

Recuperação de mídia consiste em fazer cópias periódicas do banco de dados. Copiar um grande objeto de banco de dados, tal como um arquivo, pode levar muito tempo. Enquanto isso o sistema de gerenciamento do banco de dados deve permitir continuar com suas operações. Desse modo, a criação de uma cópia é manipulada de maneira similar ao *fuzzy checkpoint*.

Quando um objeto do banco de dados tal como um arquivo ou uma página está corrompido, a cópia daquele objeto é atualizada utilizando o log para identificar e reaplicar as modificações de transações *commited* e desfazer as modificações de transações *uncommited*.

Finalmente, as atualizações de transações que estão incompletas no momento da recuperação da mídia ou que foram abortadas depois que a cópia *fuzzy* foi completada, precisam ser desfeitas para garantir que a página reflete somente as ações de transações *commited*.

8 Referência

Ramakrishnan and Gehrke, Database Management Systems, McGraw-Hill, 3rd. Edition, 2003.