Fernanda Simões de Almeida nUSP 3286502
Rodrigo Simões de Almeida nUSP 3132218

Nome – Atualização otimizada

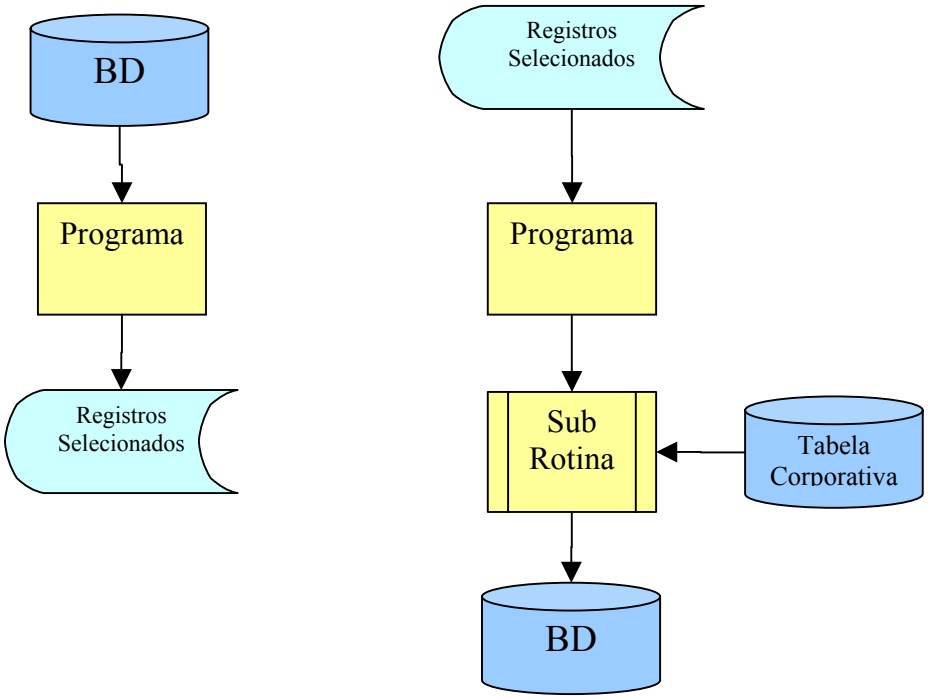
Objetivo

Atualizar uma base de dados, em um ambiente de grande porte (*mainframes*), de forma a otimizar o acesso ao banco de dados, garantindo que o sistema seja tolerante a falhas.

Motivação

Considere uma aplicação implementada para plataforma de grande porte que precisa efetuar atualizações em um banco de dados relacional após pesquisa no mesmo. A atualização pode ou não estar sendo feita nos mesmos registros selecionados.
Precisamos que o sistema seja tolerante à falhas, ou seja, se o programa for interrompido durante a atualização, não pode perder as informações dos registros já atualizados. Além disso, gostaríamos de otimizar o acesso ao banco de dados, reduzindo a quantidade de acessos a disco sem aumentar a contenção da base, para tanto desejamos efetuar "commits" parciais ao invés de "commit" a cada registro atualizado.

Estrutura



Implementação

Fatorar o programa, subdividindo-o em dois. O primeiro deve selecionar os registros desejados e efetuar toda o processamento dos dados. O resultado obtido e os dados do registro são gravados em um arquivo seqüencial.
O segundo programa somente efetua a leitura do arquivo gerado e faz as atualizações determinadas no arquivo efetuando o "commit" depois de um determinado número de registros. A cada "commit" efetuado é gravada, em uma tabela de controle, a chave do último registro atualizado anteriormente ao "commit". Após todas as atualizações a tabela de controle é atualizada com a indicação de fim do processamento. Para garantir tolerância à falhas, logo no início do segundo programa, é feita uma

consulta à tabela de controle e, se o processamento não foi concluído, o programa deve desprezar os registros do arquivo de entrada até que o arquivo esteja posicionado após o último registro já processado. A partir de então o processamento segue normalmente.
É recomendável que todo o processo de acessos à tabela de controle seja feita por uma sub-rotina (um programa chamado dentro do segundo programa) para garantir uniformidade e reutilização de código. Outra importante recomendação é que a tabela de controle possua a quantidade de registros que deverão ser atualizados a cada "commit", pois permite que ajustes possam ser feitos pelos DBA´s (Adiministradores do Banco de Dados) do sistema, sempre que necessário melhorar a performance do acesso.

Exemplo de Código

```
      ****************************************************************
      *                        00000-LOGICA-PRINCIPAL                *
      ****************************************************************
       LP-LOGICA-PRINCIPAL   SECTION.

          PERFORM A-INICIALIZA.

          PERFORM B-PROCESSA    UNTIL    FL-FIM     EQUAL   'S'.

          PERFORM C-FINALIZA.

       LP-LOGICA-PRINCIPAL-FIM.
          EXIT.

      ****************************************************************
      *                          INICIALIZACAO                       *
      ****************************************************************
       A-INICIALIZA          SECTION.

          PERFORM AA-VALIDA-PARM.

          OPEN    INPUT    TH27501I.

          PERFORM RA-LER-TH27501I.

          PERFORM AB-INICIALIZA-TB-CTRL.

          PERFORM AB-RESTART.

       A-INICIALIZA-FIM.
          EXIT.

      ****************************************************************
      *                             RESTART                          *
      ****************************************************************
       AB-RESTART           SECTION.

          IF  DB996-QT-RGO-PCE  NOT EQUAL  ZEROS
             PERFORM RA-LER-TH27501I
                   UNTIL  WS-CHAVE-LIDA  EQUAL DB996-CHAVE-GRAV
                     OR  FL-FIM-ARQ     EQUAL  'S'
             IF  FL-FIM        EQUAL      'S'
                DISPLAY '*************************************'
                DISPLAY '* TH275 - ERRO NO RESTART                *'
                DISPLAY '* REGISTRO PARA RESTART NAO ENCONTRADO *'
```

```
            DISPLAY '****************************************'
            PERFORM YA-ACESSA-SB002
         END-IF
         PERFORM RA-LER-TH27501I
      END-IF.

   AB-RESTART-FIM.
      EXIT.

  ****************************************************************
  *                      PROCESSAMENTO PRINCIPAL                 *
  ****************************************************************
   B-PROCESSA          SECTION.

      PERFORM BA-ATUALIZA-V1TH0004.

      ADD   +1     TO    AC-QTD-TRAN-COMMIT

      IF    AC-QTD-TRAN-COMMIT   EQUAL   DB996-QTD-REG-CMI
            PERFORM BB-REGISTRO-DB996
            EXEC   SQL  COMMIT   END-EXEC
            MOVE   ZEROS   TO   AC-QTD-TRAN-COMMIT
      END-IF.

      PERFORM RA-LER-TH27501I.

   B-PROCESSA-FIM.
      EXIT.
  ****************************************************************
  *                          FINALIZA PROGRAMA                   *
  ****************************************************************
   C-FINALIZA          SECTION.

      EXEC   SQL  COMMIT   END-EXEC

      CLOSE   TH27501I.

      PERFORM YC-ACESSA-SB030.

      STOP   RUN.

   C-FINALIZA-FIM.
      EXIT.
```

Conseqüências

Vantagens:

Tolerância à falhas - O uso do padrão permite que o sistema seja tolerante a falhas, pois ele consegue identificar se o processamento foi concluído normalmente ou não e dar o devido tratamento para cada condição;

Otimização do acesso ao banco de dados – As atualizações do banco de dados são feitas em momento distinto das consultas, além disso, os "commits" podem ser feitos de forma otimizada e com controle flexível, uma vez que a liberação do recurso do banco de dados está parametrizada e sob a tutela de um DBA.

Reutilização de código – O uso da sub-rotina permite reutilizar código tornando a construção mais simples e eficiente, mantendo um tratamento uniforme das retomadas em situações de término anormal de um processamento.

Desvantagens:

Uso de arquivo seqüencial e acesso a uma nova base de dados (base de controle).

Usos Conhecidos

- Sistema de administração de produtos Voucher (como: Vale Alimentação, Refeição, Farmácia, etc), em diversos momentos a citar:

  Agendamentos de transações para posterior postagem em conta;
  Efetivação de solicitações on-line em processamento batch;
  Renovação do Limite de Crédito das Contas;
  Atualizações cadastrais.

- Sistema de atendimento de reclamações de clientes de cartões de créditos;
- Sistema de premiação e fidelização de usuários de cartão de crédito;