Computer Programming Solutions
DOCUMENTO DE INFORMAÇÃO TÉCNICA

Mantendo uma seqüência sujeita a Auditoria

Resumo: *Algumas vezes, os programadores têm a missão de fornecer meios fáceis de manter seqüências ininterruptas de números seriais para certos tipos de documentos ou para identificar univocamente artigos produzidos por suas companhias. Este documento de TI examina alguns dos problemas com sistemas de números seriais e apresenta uma demonstração de um sistema como tal que tira vantagem de algumas características especiais do InterBase e do IB Objects.*

É recomendado que você acompanhe o estudo deste documento com o projeto Test_Series localizado no diretório ../TechTopics. Em virtude dos controles usados, o projeto requer o Delphi 4 ou mais recente. Uma versão executável, o qual você pode linkar ao banco de dados de teste, está incluso para propósitos demonstrativos.

Sobre os "generators"
Os "generators" do interbase são ideais para serem utilizados na criação de valores para "primary keys" porque, uma vez que determinado número é gerado, ele não pode ser gerado novamente. Diferentemente das seqüências que você mesmo mantêm nas tabelas, os "generators" estão fora do controle de transações. Não é possível para duas transações concorrentes obterem o mesmo número. As "primary keys" devem ser atômicas, ou seja, elas não devem ter nenhum significado como dado, senão satisfazer uma "unique constraint". Assim, a integridade da "unique constraint" em uma "primary key" é garantida, enquanto sua "metadata" estiver protegida de acessos de um SYSDBA não autorizado ou usuário proprietário que decida resetar um "generator" e arruinar a integridade de sua "primary key"!

"Generators" para números seriais?
Os desenvolvedores algumas vezes perguntam-se sobre usar "generators" para criar seqüências de números seriais, tais como números de fatura. Em alguns países, a regulamentação contábil ainda requer que todo número de documento gerado seja calculado dentro de uma seqüência ininterrupta ou "pura". A desvantagem dos "generators" para essa finalidade é que uma transação pode obter um número, desta forma este é "consumido", e então, se um rollback ocorre, o número cai em desuso.

Resetar o "generator" NUNCA é uma boa opção, uma vez que o número mais recente gerado seja talvez maior que o número perdido, e de modo algum resetar o "generator" irá garantir que uma duplicação subseqüente não ocorrerá.

Uma Seqüência Ininterrupta
Manter uma seqüência ininterrupta, pura é um pouco complicado, pois requer que você mantenha uma tabela para guardar não apenas os números gerados para o seu número serial, mas também para aqueles que, por várias razões, você deseja reutilizar. Entretanto, um número que aparece para um usuário como "disponível" pode não estar realmente, porque outro usuário "pegou-o" e está atualmente usando dentro de uma transação que ainda não foi comitada(commited).

Uma abordagem para resolver este problema é preterir a designação de números seriais até um ponto na existência do documento no qual a anulação do número não pode ocorrer.

> Por exemplo, se a numeração das faturas é preterida até esta ser impressa, você terá passado pela possibilidade de alocar um número que poderia se perder em um rollback da inserção do registro cabeçalho. As faturas são impressas em lotes envolvendo cabeçalhos comitados e o número da fatura é aplicado em seqüência precisa para cada fatura que seja impressa.

Realizando a Auditoria
Sistemas que requerem seqüências puras de números de faturas geralmente requerem uma contabilidade adequada para faturas deletadas ou canceladas. O único objetivo das seqüências puras é expor lacunas inexplicáveis que o auditor pode com toda a segurança reconhecer como uma irregularidade no sistema de contabilidade. Um caminho de auditoria para todos os números de fatura é, portanto, uma exigência para seqüências puras – senão não há finalidade para elas.

Numeração imediata de faturas
Muitas vezes a abordagem de números preteridos não é praticável – há uma exigência que os números de faturas devem ser designados um por um e estar disponíveis no momento que a fatura por si só é gerada. Isso geralmente ocorre em e-commerce (comércio eletrônico) ou transações de encomendas via correio com cartão de crédito, vendas cash-on-delivery (pronto pagamento no ato da entrega) e em regimes que possuem impostos adicionais em vendas e/ou serviços.

Você pode envolver "generators" em seqüências puras lidando cuidadosamente com números perdidos. Apenas NUNCA use a chave primária de sua tabela para um número de documento sujeito a auditoria.

Exigências para um sistema de numeração serial seguro
Aqui está um conjunto de exigências para um sistema de numeração de faturas que combina um generator com uma tabela de logging de números de fatura que seguramente reaproveita os números desperdiçados e grava quaisquer faturas que são canceladas após terem sido comitadas.

Nessa situação, os usuários são normalmente impossibilitados de deletar faturas uma vez que elas são comitadas. Entretanto, uma fatura pode ser "clean-deleted" por um usuário autorizado se um usuário acidentalmente digitou a mesma fatura ou criou uma fatura para o cliente errado. Faturas Clean-deleted na existem mais e apenas os números designados a elas podem ser reaproveitados.

O processo é também exigido para reaproveitar quaisquer números de fatura que foram gerados, mas tornaram-se "órfãos" porque a transação não foi comitada.

Em todas as outras situações, uma fatura comitada deve ser cancelada e o sistema criará o diário apropriado. Nesse caso o número da fatura não estará disponível para ser reutilizado – ele deve ser guardado para fins de auditoria.

Dica – Faça validações Primeiro !
Para minimizar o número de lacunas que podem ocorrer, tente codificar seus programas de entrada e manutenção de dados de modo que a nova fatura tenha passado por todas as outras validações ANTES de você aplicar a ela o número serial gerado.

> *Em uma aplicação de e-Commerce, por exemplo, faça a verificação do cartão de crédito antes de posta-lo. Se você preferir criar uma fatura e reservar um número sem levar em conta o resultado da verificação do cartão de crédito. Esteja preparado pra registrar um cancelamento.*

Implementando as Exigências

Você pode explorar este contexto no projeto demo "Series". Este é um script de demonstração (test_series.sql) para você mesmo criar este pequeno banco de dados de teste.

Nós temos estas duas tabelas:

```
CREATE TABLE INV_HEADER(
  INV_ID INTEGER NOT NULL,
  CUST_ID INTEGER NOT NULL,
```

```
  INV_DATE DATE NOT NULL,
  INV_NUMBER INTEGER,
  INV_OPERATOR VARCHAR(128),
  STATUS VARCHAR(20),
  CONSTRAINT PK_INV_HEADER PRIMARY KEY(INV_ID));

CREATE TABLE INV_LOG(
  LOG_ID INTEGER NOT NULL,
  INV_NUMBER INTEGER,
  INV_ID INTEGER,
  STATUS VARCHAR(20),
  REASON VARCHAR(50),
  OPERATOR VARCHAR(128),
  GEN_DATE DATE,
  LOGDATE DATE,
  CONSTRAINT PK_INV_LOG PRIMARY KEY(LOG_ID));

CREATE GENERATOR GEN_INV_HEADER_ID; /* para a primary key  do cabeçalho da fatura */
CREATE GENERATOR GEN_INV_LOG_ID; /* para a primary key da tablea de log da fatura */
CREATE GENERATOR GEN_INV_NUMBER; /* para os números de fatura */
```

Nós adicionamos alguns "unique indexes" para auxiliarem na exibição de auditoria e também garantir que um número de fatura será sempre único:

```
CREATE UNIQUE INDEX UN_INV_HDR_INV_NUMBER
ON INV_HEADER(INV_NUMBER);
CREATE UNIQUE INDEX UN_INV_LOG_INV_NUMBER
ON INV_LOG(INV_NUMBER);
```

Forneça outros números de posição para os "Before Insert triggers" das tabelas de cabeçalho de fatura e log de números de fatura que sejam menores que os "triggers" que fazem o trabalho com o número da fatura, e.g.

```
CREATE TRIGGER BI_INSERT_INV_HDR
FOR INV_HEADER
ACTIVE BEFORE INSERT POSITION 9 AS
BEGIN
  IF (NEW.INV_ID IS NULL) THEN
    NEW.INV_ID = GEN_ID(GEN_INV_HEADER_ID, 1);
  IF (NEW.INV_DATE IS NULL) THEN
    NEW.INV_DATE='NOW';
  NEW.INV_OPERATOR = USER;
END

CREATE TRIGGER BI_INSERT_INV_LOG
FOR INV_LOG
ACTIVE BEFORE INSERT POSITION 9 AS
BEGIN
  IF (NEW.LOG_ID IS NULL) THEN
    NEW.LOG_ID = GEN_ID(GEN_INV_LOG_ID, 1);
  NEW.OPERATOR = USER;
END
```

O último "Before Insert trigger" é usado para pegar o número da fatura para um novo cabeçalho de fatura. Primeiro ele testa se algum número reciclado está disponível e, se houver, ele trava o menor deles. Enquanto uma transação está mantendo este "lock", nenhuma outra transação pode efeturar o SELECT MIN() assim esteja certo que a transação está configurada com

"LockWait" em true para evitar "deadlocks".

Se não houver números reciclados disponíveis, ele obtém um do "generator".

```
CREATE TRIGGER BI_INSERT_INV_NUM
FOR INV_HEADER
ACTIVE BEFORE INSERT POSITION 9 AS
DECLARE VARIABLE INV_NO INTEGER;
BEGIN
  INV_NO = 0;
  SELECT MIN(INV_NUMBER) FROM INV_LOG
  WHERE STATUS='AVAILABLE'
  INTO INV_NO;
  IF INV_NO > 0 THEN
    UPDATE INV_LOG SET STATUS=STATUS /* locks the record */
    WHERE  INV_NUMBER= :INV_NO;
  ELSE
  BEGIN
    INV_NO = GEN_ID(GEN_INV_NUMBER, 1);
    INSERT INTO INV_LOG(INV_NUMBER, GEN_DATE)
    VALUES(:INV_NO, 'NOW');
  END
  NEW.INV_NUMBER = :INV_NO;
END
```

Em um "Äfter Insert trigger", o registro de log gerado é atualizado com o valor da "primary key" do Cabeçalho da Fatura e tem o seu "status" alterado para torna-lo indisponível. Quando a transação se completa, o "lock" no número da fatura "reutilizável" é liberado sem trabalhos de programação adicionais.

```
CREATE TRIGGER AI_INSERT_INV_NUM
FOR INV_HEADER
ACTIVE AFTER INSERT 1O AS
BEGIN
  UPDATE INV_LOG SET STATUS='USED',
  INV_ID = NEW.INV_ID,
  REASON ='',
  OPERATOR = USER,
  LOGDATE = 'NOW'
  WHERE INV_NUMBER = NEW.INV_NUMBER;
END
```

Duas "procedures" mais (ou sentenças SQL para executar a partir de sua aplicação, se preferir) e um "trigger" são necessários para lidar com faturas que são canceladas, abortadas e deletadas respectivamente. Primeiro, uma "stored procedure" para cancelar uma fatura:

```
CREATE PROCEDURE SP_CANCEL_INVOICE (INV_NO INTEGER, RSN VARCHAR(50))
AS
BEGIN
  UPDATE INV_LOG
  SET STATUS = 'CANCELLED',
     REASON = :RSN
  WHERE INV_NUMBER = :INV_NO;
  UPDATE INV_HEADER
  SET STATUS = 'CANCELLED'
  WHERE INV_NUMBER = :INV_NO;
END
```

Chame esta "stored procedure" (ou submeta a sentença SQL apartir de sua aplicação) para liberar o número da fatura quando a inserção do Cabeçalho da Fatura é abortado:

```
CREATE PROCEDURE SP_FREE_INV_NUMBER(INV_NO INTEGER)
AS
BEGIN
  UPDATE INV_LOG
  SET INV_ID = NULL,  /* it probably is already NULL but we want to make sure */
       STATUS = 'AVAILABLE',
       REASON = 'INVOICE ABORTED',
       OPERATOR = USER,
       LOGDATE = 'NOW'
  WHERE INV_NUMBER = :INV_NO;
END
```

Finalmente, um "After Delete trigger" na tabela de Cabeçalho da Fatura faz exatamente a mesma coisa se um registro é deletado:

```
CREATE TRIGGER AD_DELETE_INV_HDR
FOR INV_HEADER
ACTIVE AFTER DELETE POSITION 9 AS
BEGIN
  UPDATE INV_LOG
    SET INV_ID = NULL,
       STATUS='AVAILABLE',
       REASON='INVOICE DELETED',
       OPERATOR = USER,
       LOGDATE='NOW'
    WHERE INV_ID=OLD.INV_ID;
END
```

Experimentado com a Aplicação demo Test_Series

Crie o Bando de Dados de teste

Abra a toolbox IB_SQL e carregue o script Teste_Series.sql para dentro do utilitário de script. Edite a sentença CREATE DATABASE para adaptar ao local do arquivo de banco de dados que você deseja e, se necessário, edite o nome do usuário e a senha.

Se você está criando o banco de dados no InterBase versão 6/Firebird Dialeto 3, edite o tipo do domínio D_IDENTIFIER e mude-o para NUMERIC(18,0). Declarações INTEGER nos "stored procedures" e "triggers" estão marcados no script e devem também ser alterados.

Durante o processo, alguns registros de log "órfãos" serão criados para simular registros de log que foram criados para cabeçalhos de fatura que nunca fora comitados no banco de dados. Não tente deletá-los – eles são necessários para a demonstração. Se você precisa criar mais órfãos, delete alguns registros da INV_HEADER, anotando seus INV_NUMBERs e então atualize o campo "STATUS" do registro de log correspondente para "USED" interativamente, usando o IB_SQL.

Configurando a IDE do Delphi

Uma vez que você tem seu banco de dados, abra o projeto Test_Series no Delphi 4 ou 5 e configure a string de conecção corretamente nas propriedades do componente IB_Connection.

> *Você pode querer abrir o script na IDE do Delphi, usando o filtro *.sql para ter um layout bom, legível, code-sensitive no editor. Você pode configurar as opções do seu editor para mostrar o Editor do Delphi enquanto a aplicação estiver rodando.*

Execute a aplicação.

Inserindo Cabeçalhos de Fatura
Coloque a tabela de Cabeçalho de Fatura em modo de inserção clicando no símbolo de inserção (verde + sinal) na barra de ferramentas. Simplesmente digite qualquer numero inteiro na caixa de edição rotulada "Customer ID" e clique no símbolo de Post (símbolo de "tick"' ou "check").

Observe o trabalho dos "triggers"quando as tabelas se atualizam. Neste estágio, a aplicação está gerando novos valores de INV_NUMBER para cada nova fatura criada. Os números INV_ID e LOG_ID devem estar em sincronia neste ponto.

Cancelando Faturas
Para cancelar uma fatura, selecione a Razão da "radio box", selecione uma fatura e clique no botão Cancelar Fatura Comitada da barra de ferramentas. Este é o símbolo "X" com um "i" preto no centro.

Observe o que acontece quando ambas as tabelas se atualizam. Observe que o Status de Log do número de fatura mudou de "USED" para "CANCELLED" e a razão de cancelamento é exibida. O Status da fatura também muda para "CANCELLED".

Testando Números de Fatura Reaproveitados
Nessa aplicação, você pode deletar qualquer cabeçalho de fatura. Nós adquirimos algumas liberdades em virtude da necessidade do propósito da demonstração, é claro!Sua aplicação real precisará ser muito mais exigente sobre quem pode deletar faturas.

Selecione um cabeçalho de fatura dentre aquelas que não estão marcadas como "'CANCELLED". (Perdão, não há multi-select!) Clique no botão Delete da barra de ferramentas (o símbolo '-' vermelho) e responda a mensagem de confirmação.

Observe que o Cabeçalho da Fatura se foi, mas o registro de log permanece, o qual seu status mudou para "AVAILABLE" e a mensagem na coluna Reason.

Delete dois ou mais cabeçalhos.

Agora adicione alguns novos Cabeçalhos de Fatura. Observe como o "Before Insert trigger" (position 9) reaproveita os números de fatura "liberados" em uma rigorosa seqüência FIFO, para assegurar que os números estejam dentro do limite de qualquer novo número que venha depois. Observe como o status de cada número reaproveitado muda quando um novo Cabeçalho de Fatura os utiliza.

Envolva-se em um pequeno conflito...

Libere mais alguns números de fatura deletando mais alguns cabeçalhos.

Agora, abra uma outra instância da aplicação executando o EXE a partir do diretório do projeto. Alinhe as duas instâncias assim você pode observar os grids do Cabeçalho e do Log de ambas. Em uma instância, comece a inserir um novo cabeçalho, mas não confirme-o.

Faça a mesma coisa na segunda instância, mas, dessa vez, tente posta-la. Observe que você recebe uma mensagem de Deadlock. Isto não é um problema – isso mostra que o lock aplicado no "Before Insert trigger" pela primeira instância está funcionando.

Confirme a inserção na primeira instância. Agora confirme aquele na segunda instância. O Deadlock se vai.

> *Em sua aplicação na vida real, você desejará colocar um loop temporizador na confirmação/repetição da inserção e capturar o erro de conflito silenciosamente, assim o*

usuário será desconhecedor de qualquer conflito existente. Atrasos de Deadlock serão imperceptíveis para o usuário porque sua duração é extremamente curta, provavelmente apenas uma fração de milisegundos.

Controle de concorrência de Multi-usuário
Para aqueles que são novos no InterBase, vocês estão observando um importante recurso em programação para InterBase – o controle de concorrência multi-usuário. Outros bancos de dados cliente/servidor exigem que a aplicação cliente requisite locks explícitos onde o conflito é mais comum aparecer. Com o InterBase, você pode ser tão pessimista quanto precisar ser – O IB procura por locks explícitos internamente e os aplica se o conflito surge.

Neste exemplo, a transação que insere novos cabeçalhos de fatura está configurada com o nível de isolamento tiCommitted, significando que a transação pode apenas considerar (e serem mostrados) registros que já tenham sido comitados no momento em que o novo registro é postado. Se outra transação já modificou um registro mas ainda não comitou a alteração para o banco de dados, a nova transação não pode vê-lo. Assim a primeira transação tem a garanta de ser a única candidata para reaproveitar o menor número de fatura disponível.

Fazendo uso dos Deadlocks Planejados
Quando a transação vai postar o novo registro, ela primeiro percorre seus "Before Insert triggers" e chega no "trigger" de posição 9. Neste ponto, ela procura pela existência de algum registro de log contendo números de fatura reaproveitados. Se sim, ela "pega" o menor número e realiza um falso update - SET STATUS=STATUS. Isso faz com que o InterBase cause um lock no conjunto de registros diponíveis. Devido cada transação do usuário também ter LokWait configurado para true, isto causa a condição de Deadlock para todos os outros competidores para o menor número. O LockWait fala para o motor do banco de dados que, se um lock é encontrado, a transação quer esperar até que o conflito acabe.

Num ambiente multi-usuário normal, você não solicitaria um LockWait porque uma transação do usuário com um longo tempo de duração podia travar grandes blocos de registros indefinidamente. Para esta tarefa particular, isto é ideal, porque o conflito tem a garantia de ser apenas momentâneo. Uma vez que os "After Insert triggers" são executados e a alteração é escrita no banco de dados, o lock desaparece. A transação em espera (sua segunda instância) pode agora ir à frente e seguramente testar e adquirir o próximo número reaproveitado.

Normalmente, você não programa deliberadamente para usar deadlocks – de fato, você programa arduamente para evitá-los e manter o trabalho multi-usuário fluindo! Entretanto, esta é uma situação para qual os deadlocks são destinados – para impor serializações se ela é uma exigência absoluta. Ele trata uma falha comum de design em sistemas de geração de números que alguns de vocês podem ter herdado com o legado Paradox, Access ou outra aplicação single-user. Aqui está como ele acontece.

Como ponderar ajudas de How deliberate deadlocking helps
Em bancos de dados baseados em arquivos e até mesmo em alguns sistemas RDBM cliente/servidor de "grande nome", todo o trabalho é estritamente seqüenciado. O Conflito de usuário deve ser tratado por travamento estritamente pessimista. Devido a este poder congestionar um ambiente multi-usuário, os desenvolvedores ignoram o travamento explicito. Em procedimentos de geração de números, violações a restrições de unicidade são inevitáveis. Como resultado, um desenvolvedor deve remover as restrições de unicidade a fim de prevenir violações – e resultem na corrupção dos dados! O apropriado uso das capacidades de deadlocking do InterBase evita tudo isso.

Liberando números "orfãos" para Reciclagem
No grid, no canto inferior esquerdo do formulário, está um grid listando todos os números de fatura que estão presentes na tabela de log. Eles são trazidos usando esta sentença SQL:

```
SELECT LOG_ID, INV_NUMBER FROM INV_LOG
WHERE (NOT (INV_NUMBER IN (SELECT INV_NUMBER FROM INV_HEADER)))
AND STATUS <> 'AVAILABLE'
```

Em outras palavras, eles são como registros de log deixados para trás pelos números de fatura deletados, exceto que seus status estão ainda como eram quando a transação tentou (e falhou) comitar o novo cabeçalho para o banco de dados. Como você pode ver no mostrador de Log, eles estão carregando a chave INV_ID dos cabeçalhos "pais" que "morreram" mas eles não estão prontos ainda para serem dados a um novo ancestral. Devido ao seu status estar ainda "USED", o "insert trigger" da tabela INV_HEADER não pode vê-lo.

Uma "stored procedure" - SP_FREE_INV_NUMBER – está esperando para resgata-lo. Para chamar esta SP, apenas selecione um número órfão do grid e clique no botão "Reciclar"'. Ele simplesmente passa o INV_NUMBER selecionado para a stored procedure, que atualiza o Status para "AVAILABLE", nula o não-existente INV_ID e adiciona uma mensagem à coluna Razão.

> *Em sua aplicação, se você tem sido impossibilitado de eliminar totalmente a possibilidade de falha na confirmação de cabeçalhos de fatura, você pode facilmente implementar um tratador de exceção que realiza uma atualização similar a que a SP SP_FREE_INV_NUMBER faz nessa demonstração, invisivelmente para os usuários.*

Como sempre, se você tem quaisquer comentários ou questões sinta-se à vontade para contatar a mim ou a lista pública dedicada ao IBO.

Jason Wharton
http://www.ibobjects.com
mailto:jwharton@ibobjects.com

Artigo Original: http://www.ibobjects.com **Jason Wharton** jwharton@ibobjects.com	
Tradução e adaptação: **George Hugo** georgehg@uol.com.br	**Comunidade Firebird de Língua Portuguesa** **Visite a Comunidade em:** **http://www.comunidade-firebird.org**
A Comunidade Firebird de Língua Portuguesa foi autorizada pelo Autor do Original para elaborar esta tradução.	