# Integridade Referencial x Triggers

A teoria relacional diz que você deve normalizar os dados para torná-los adequados dentro de um modelo relacional. É claro, entidades devem estar conectadas de alguma forma.

A ligação entre duas entidades normalizadas é um campo em comum para ambas as entidades. O campo em comum age como uma ponte, então você pode passar de uma entidade para outra usando SQL, isto é uma representação física de um relacionamento lógico entre duas entidades. De fato você pode ter uma entidade ligada a ela mesma ou três entidades ligadas ao mesmo tempo também, dependendo da semântica do seu modelo.

Quando um modelo é normalizado, isto geralmente significa que você pega um conceito do mundo real ou alguma coisa decomposta em muitas entidades normalizadas geralmente referenciadas como tabelas. Mas estas tabelas devem ser mantidas em sincronia para ter-se dados consistentes.

Por exemplo, uma Nota Fiscal e seus detalhes são duas tabelas separadas, mas nenhum detalhe deveria ser aceito se referenciasse uma Nota Fiscal que não existe. Por outro lado, uma Nota Fiscal não pode ser excluída a menos que não haja detalhes ou que os detalhes sejam excluídos primeiro. Validar estas regras na aplicação cliente não é uma solução a prova de bala porque pessoas podem usar uma ferramenta para inserir, excluir ou atualizar os dados. Para resolver isso é que veio a integridade referencial.

Esta é uma declaração de metadata que cria um relacionamento de dependência entre duas tabelas através de um ou mais campos. Assim, o mecanismo por si próprio é responsável por verificar a consistência entre as tabelas; você pode pensar que uma tabela refere-se a outra para validação e daí o nome "integridade referencial".

Em FB, uma integridade referencial pode ser definida no nível de campo ou no nível de tabela. Quando mais de um campo é envolvido, uma integridade por nível de tabela é requerida. Em SQL, esta integridade é chamada de "foreign key declaration" (declaração de chave estrangeira). A tabela de consulta (aquela onde os valores são contrastados) deve existir. Três importantes restrições são:

1. Atualmente uma declaração de chave estrangeira só pode ser feita quando uma conexão é feita ao banco de dados. Se mais conexões estão em aberto uma mensagem de erro como "Objeto está em uso" aparecerá.
2. Se a tabela de consulta tem sido usada na sessão corrente mesmo que não haja conexões ao banco de dados, pode ser necessário desconectar e reconectar.
3. A tabela de consulta deve ter já declarado o(s) campo(s) referenciados com um índice único ou o comando não será aceito. Só a tabela de detalhes terá um índice sem ser único gerado automaticamente para manipular o relacionamento.

Agora que as declarações de chave estrangeira tem estabelecido o esquema de banco de dados você vai descobrir que não é muito fácil mudar o metadata de tabelas com integridade referencial. Especialmente, você não pode excluir ou alterar campos que são envolvidos numa declaração de chave estrangeira. Por causa destas limitações, você pode estar tentado definir sua integridade referencial não com comandos padrões de definição de dados (DDL) mas controlar as validações por você mesmo por meio de triggers (gatilhos).

Entretanto alguns inconvenientes existem:

A razão principal pela qual uma tabela (principal, de consulta) deve ser alterada, também é que um registro não pode ser excluído na principal se existirem registros na tabela de detalhes apontando para ele, então você terá que definir uma trigger não apenas na tabela de detalhes (para evitar inserções sem um registro na tabela principal) mas também na tabela principal. Isto pode tornar-se um fardo na medida em que o número de tabelas aumente.

Declarações de chave estrangeira são evidentes pela sua própria leitura. Ter que ler o código de uma trigger para descobrir o que ela faz não é tão atrativo. Também porque uma tabela pode ter muitas triggers, para aquelas definidas por usuários para o propósito de integridade referencial são dados nomes mais significativos. Trigger são mais usadas para criar entradas em tabelas de auditoria, para modificar ou preencher alguns campos antes de inserir ou mudar os dados, para criar a possibilidade de campos auto-incremento e para parar operações no caso de uma condição falhar e esta condição não pode ser manipulada por uma declaração de chave estrangeira por nível de tabela ou por nível de coluna então isso tem que ser feito com instruções de procedure. Também, você não pode excluir registros que são usados por triggers ou stored procedures (procedimentos armazenados).

Geralmente uma declaração de chave estrangeira é otimizada por isso ela faz uma rápida conferência das condições. Se você pretende substituí-la por uma trigger, certifique-se de fazer uma boa procura. Por exemplo, não use "Count(*)" para encontrar uma ocorrência; você está interessado em saber se existe apenas uma ocorrência, logo um teste "EXISTS" com um simples "Select_*" será mais rápido.

A partir do Interbase 5 em diante, declarações de chave estrangeira podem incluir opções que especificam o que deveria acontecer quando um registro numa tabela principal é excluído ou alterado e existem registros dependentes nas tabelas de detalhes. As opções típicas são CASCADE, NULL e DEFAULT. Ter que reproduzir estes recursos numa trigger não é nada interessante.

Talvez o ponto mais importante foi deixado para o final: FB vai esmerar-se para assegurar que as regras de integridade referencial definidas através de declarações de chave estrangeira sejam rodadas em contraste aos dados gravados de outras transações de uma forma independente do estado e nível de isolamento da sua transação.

É claro, dentro da sua própria transação não há problemas de visibilidade. Lembre-se que FB sempre usa uma transação seja implícita ou explícita. Pelo fato do padrão de isolamento ser "snapshot" você corre o risco de perder algumas entradas que foram postadas e gravadas após o início da sua transação, desta forma uma trigger que faça um Select não verá estas novas entradas. Aqui, duas coisas podem ocorrer: ou você não pode inserir numa tabela de detalhes mesmo que outra transação tenha gravado o registro necessário na tabela principal ou pior, você exclui um registro de uma tabela principal e a sua trigger fazendo uma exclusão na tabela de detalhes não apaga alguns registros que correspondem a condição para ser apagado porque eles foram gravados após o início da sua transação, aí você está deixando alguns registros perdidos (registros órfãos) em um problema que parece com ponteiros perdidos em linguagens de programação, a única diferença é que um comando SQL com registros órfãos não faz parar o programa como um ponteiro inválido faz. Estes pesadelos são evitados com declarações de integridade referencial.

Em FB, stored procedures e triggers não podem iniciar transações por si mesmas e não podem mudar o isolamento da transação corrente. Elas rodam dentro da transação iniciada pelo cliente. Você pode dizer que integridade referencial por procedimento (isto é, usando trigger e não declarações) ainda podem ser usadas contanto que o nível de isolamento "read committed" seja usado por todos os clientes. Além disso, outro mecanismo semelhante usou 'read committed" como nível padrão de isolamento e o BDE também usa. No entanto, nada impede seu cliente de usar "snapshot" explicitamente (mesmo que em outros mecanismos) e você não tem meios para detectar isso numa trigger. Também, uma das intenções dos mecanismos semelhantes é passar a execução de todas as regras básicas para o banco, dessa forma não há necessidade de reinventar a roda sem que você realmente precise de uma procura muito complexa em outra tabela. Neste caso, isso pode ser um sinal de um problema de normalização.


| **Artigo Original**<br><br>http://www.cvalde.com/document/declaRefIntegVsTrig.htm<br><br>Claudio Valderrama<br>http://www.cvalde.com/index.htm |  |
|---|---|
| Tradução e adaptação:<br><br>**Rogério Muniz Joras**<br>rogerio@multi-informatica.com.br | **Comunidade Firebird de Língua Portuguesa**<br><br>**Visite a Comunidade em:**<br><br>**http://www.comunidade-firebird.org** |
| A Comunidade Firebird de Língua Portuguesa foi autorizada pelo Autor do Original para elaborar esta tradução. | |