Explanação das Opções de Transação

Uma transação é uma unidade lógica de trabalho. Desta forma, vários comandos podem ser englobados em uma única transação. Também, um único comando pode se estender por várias unidades físicas de trabalho, por exemplo, um UPDATE com uma clausula WHERE que pode afetar mais que um registro. Quando você confirma (commit) ou descarta (rollback) uma transação, isto afeta todas as ações realizadas na transação (desde de quando a transação foi iniciada, ou após o último commit/rollback com retenção de transação tenha sido executado). Enquanto a transação está em progresso, as mudanças feitas dentro dela não podem ser vistas por outra transação (exceto no caso de se estar utilizando o modo *dirty read* ou “leitura suja (veja a seguir). A transação teoricamente não tem um limite de comandos que possam ser salvos ou cancelados; por outro lado, na prática, existem limites úteis impostos por cada SGDB.

É importante perceber que todo trabalho do Firebird(FB) é feito dentro de uma transação, sendo implícita ou explícita. Isto foi projetado para permitir que múltiplos mecanismos de versionamento trabalhem. Se você não está controlando transações, cada comando é executado em uma transação que se encerra automaticamente quando este termina. Isto é conhecido como Auto-Commit (auto confirmação). Entenda que mesmo nestes casos, cada transação implícita pode iniciar várias operações, por que alterando apenas um registro podem ser disparados vários triggers (gatilhos), e estes podem modificar vários registros em outras tabelas e assim por diante. Então, o modo Auto-commit de um comando e todos os seus efeitos são salvos quando tudo der certo, ou cancelados quando alguma operação falhar.

O FB não suporta transações aninhadas. Isto porque, a linguagem de stored procedure (procedimento armazenado) e trigger não suportam controle de transações nos seus comandos. No FB, cada procedure (procedimento) ou trigger roda no contexto da transação iniciada pela aplicação cliente. Todavia, o FB permite várias transações concorrentes ou paralelas na mesma conexão, mas infelizmente esta capacidade não é oferecida através de produtos de acesso genérico a banco de dados, como BDE ou ODBC, para isto, você conta com a API do FB, em produtos de acesso como a FIBC que é gratuita, IBX ou em produtos comerciais como o IBO.

Para pessoas que usaram bancos de dados relacionais, as opções oferecidas pelo FB podem parecer ocultas, mas isto é mais do que simplesmente nomes diferentes para diferentes possibilidades. De fato, o FB trabalha como qualquer outro banco de dados relacionais trabalha em casos comuns. A partir daqui, eu usarei “txn”como sinônimo para “transação”. Esta é a sintaxe:

SET TRANSACTION [NAME transaction]
[READ WRITE | READ ONLY]
[WAIT | NO WAIT]
[[ISOLATION LEVEL] {SNAPSHOT [TABLE STABILITY]
| READ COMMITTED [[NO] RECORD_VERSION]}]

[RESERVING <reserving_clause>
| USING dbhandle [, dbhandle ...]];
<reserving_clause> = table [, table ...]
[FOR [SHARED | PROTECTED] {READ | WRITE}] [, <reserving_clause>]

Modo de acesso:

É o tipo, ou modo de acesso que a transação terá no banco de dados. Os tipos de acesso pode ser o seguinte:

- READ WRITE (Leitura & Gravação): é o modo padrão. Ele provê ao usuário, que iniciou a txn, o poder de ler informações e alterar o conteúdo do banco de dados;
- READ ONLY (Somente leitura): permite ao usuário apenas ler as informações do banco de dados não permitindo a ele executar comandos que possam vir a alterar o conteúdo do mesmo.

Observe que o modo RW pode executar, dentro do seu contexto, comandos como insert/update/delete e o modo RO permitem somente os comandos select/reference.

Tipo de Bloqueio:

O modo de bloqueio especifica como o FB procederá quando ocorrer algum conflito de leitura ou gravação dos dados. O modo padrão é WAIT (esperar), neste modo o usuário que encontrou o conflito fica aguardando até que a txn anterior que ainda não foi confirmada seja encerrada. O controle não retorna para o cliente até que a operação possa continuar. No modo NO-WAIT(sem esperar), o usuário que gerou o conflito com outra txn não confirmada receberá uma mensagem de erro imediatamente. Saiba que os conflitos podem aparecer não somente em operações de gravação, mas também em operações de leitura. Para ver como as operações de leitura podem se comportar veja a seguir “READ COMMITTED” em “Nível de Isolamento”. Já, para as operações de gravação, a resposta não depende do modo de isolamento, veja a seguir:

- Quando a txn A insere um registro(tupla), mas não dá um commit, se a txn B tentar inserir um registro com a mesma PK(chave primária) que A, e B estiver com Tipo de Bloqueio = WAIT, B terá que esperar até que A dê o commit na operação (e B receberá um erro de violação de PK) ou descarte, não confirmando, (e B poderá continuar); já se o “Blocking Mode” de B for NO-WAIT, ele receberá imediatamente um erro assim que a inserção for enviada.
- Quando a txn A insere um registro, mas dá um commit, se a txn B tentar deletar ou alterar registros da mesma tabela, nenhum problema ocorre, porque os registros da txn A não são vistos por B até que A de um commit na txn, isto irá ocorrer quando o nível de isolamento de B for READ COMMITTED (ler confirmados) ou quando tiver um isolamento superior e der um commit depois de A.
- Quando a txn A atualiza um registro, mas não dá um commit, se a txn B tentar atualizar ou deletar o mesmo registro ou ainda, usar um comando update ou delete, que inclui registros abrangidos por A, e

B estiver com modo de bloqueio igual a WAIT, ele ficará bloqueado até que A de um commit ou rollback. Se A der um commit, B pode receber a mensagem “Deadlock – conflito de atualização com uma atualização concorrente”, mas se A Não der um rollback, B pode continuar. Por outro lado, se B estiver com NO-WAIT, verá imediatamente a mensagem dizendo “Lock conflict on no wait transaction – Deadlock” (Conflito de bloqueio em transação sem espera – Deadlock).

Nível de Isolamento:

Também conhecido como modo de isolamento, ele controla a visibilidade de uma txn com relação às alterações feitas por outras txns concorrentes. Pessoas que vem de bancos de dados desktop e sem conhecimento com transações e conceitos relacionais, podem encontrar problemas tentando entender o porque deles terem que tomar bastante cuidado com esta característica. Aqui estão Quatro níveis conhecidos:

- Read uncommitted: chamado “Dirty Read” (leitura suja) no BDE, ele permite a uma txn ler todas as mudanças feitas por outra txn em qualquer registro, mesmo que estas mudanças não tenham sido confirmadas. Este nível não é recomendado para bancos relacionais por que ele permite ler inconsistências e informações parciais. Os valores que uma txn mudou/inseriu/deletou no banco de dados, depois de confirmados, devem ser vistos pelas outras txns com um novo estado consistente do banco de dados, e o modo de isolamento “dirty read” vai contra esta idéia básica. Isto ocorre tipicamente em Paradox e Dbase. Entre os servidores relacionais, somente DB2 e Informix suportam isto. Mesmo que em teoria o FB possa oferecer a arquitetura Multi Geracional para este nível, ele não suporta isto. Então não procure no IBO ou BDE por este nível, por que o FB não suporta.
- Read committed: chamado pelo mesmo nome no BDE e na maioria dos servidores relacionais, este é o nível típico de qualquer mecanismo. Ele permite a txn ler somente mudanças confirmadas feitas por outras txns. Isto ajuda a garantir que as txns verão apenas estados consistentes do banco (é claro que isto assume que todos os clientes irão fazer um conjunto de mudanças inter-relacionadas dentro de uma txn englobada). Mesmo se uma txn confirmar depois de uma outra txn, usando “read committed” ter iniciado, estas mudanças poderão ser lidas. Este é nível padrão para o BDE e recomendado para usuários interativos. Existem dois sub-modos que se aplicam somente a este nível:
  - *Record version:* a última versão confirmada do registro é lida imediatamente. Esta é o padrão.
  - *No Record version:* se estiver ali uma versão não confirmada de um registro gerado por outra txn, a txn atual fará o seguinte, se o Tipo de Bloqueio for WAIT, ela irá esperar até o registro ser confirmado (commit) ou descartado (rollback), ou se o Tipo de Bloqueio for NO-WAIT um erro será gerado imediatamente. De qualquer forma, a txn sempre tentará ler a versão mais recente de um registro. É claro, que isto se aplica a mudanças feitas por

outras txns, por que uma txn sempre pode ler suas mudanças não confirmadas. Este sub-modo causa um monte de mensagens de "deadlock" que você pode usar com cuidado.

- Snapshot: chamado de "Repeatable Read" (leitura repetitiva) no BDE, ele permite que uma txn possa obter uma imagem completa do banco de dados quando ela for iniciada. Esta txn não consegue ver qualquer mudança feita por outras txns concorrentes. Este nível é ideal para relatórios, porque na mesma txn, um registro pode ser lido muitas vezes, e seu valor deve ser o mesmo até que a sua própria txn mude algum valor. Esta txn pode ver mudanças feitas por ela mesma. Este nível de isolamento não é recomendado para usuários interativos, por que eles precisam de dados atuais, e não de dados congelados. Isto também pode ser usado para operações obrigatórias e, portanto devem ser confirmadas ou descartadas; por que ela previne "garbage collection" (lixos): todas as versões até o momento que a txn foi iniciada se manterão para manter uma visão estável do banco de dados. Até o momento o Delphi 3, que apareceu em 1997, somente o Oracle suportava este nível no modo "read only" e Sybase e MS-SQLServer não suportavam isto. Provavelmente, um servidor relacional "normal" paga um preço caro em performance ou recursos para suportar este modo, enquanto que o FB é projetado para suportar este nível de isolamento em modo read/write degradando pouco o sistema.
- Snapshot table stability: conhecido também pelo nome de "Forced Repeatable Read" (leitura repetitiva forçada), este nível é específico do FB/IB e não existe em outro servidor relacional e também não é suportado pelo BDE. Ele pode ser chamado de escrita de estabilidade "Snapshot", porque ela tem as mesmas propriedades do nível anterior, mas também quando for feita a leitura da tabela a mesma será bloqueada para escrita, para as outras txns. Então, a txn com este nível tem o controle sobre a tabela e todas as outras txns poderão somente ler a partir desta tabela. Este nível não é recomendado para propósitos gerais; ele só deve ser utilizado quando realmente for necessário e em curtos períodos de tempo deve ser confirmada ou descartada, porque cada uma das outras txns que estiverem tentando gravar na mesma tabela receberá uma mensagem de erro ou será colocada em espera até que a txn seja concluída. Isto pode facilmente gerar toneladas de conflitos ou causar um "deadlock" geral no banco de dados. É claro que, se você tentar usar a tabela que está começando a ser modificada por outra txn, um dois terá que esperar o outro terminar ou receberá imediatamente uma mensagem de erro, dependendo do tipo de "blocking mode" que estiver especificado para a sua txn.

Para entender como o acesso de leitura e os níveis de isolamento interagem, eu transformei uma matriz de possibilidades em uma lista de possibilidades:

- Duas txns com "snapshot table stability" só poderão compartilhar uma tabela se ambas tiverem acesso READ (leitura).
- Qualquer número de txns pode usar uma tabela guarnecida, tanto quanto elas tenham acesso READ (leitura), sem levar em consideração os nível de isolamento.

- Nenhuma txn pode usar uma tabela com acesso WRITE (escrita) se uma txn com “snapshot table stability” como mode de acesso WRITE (escrita) estiver usando a mesma tabela.
- Uma txn “snapshot table stability” com modo de acesso WRITE não pode usar uma tabela se qualquer txn com modo de acesso WRITE estiver usando a mesma tabela (é o reverso do caso anterior).
- Combinações de transações “snapshot” e “read committed” com modo de acesso WRITE podem compartilhar tabelas, mas conflitos podem aparecer quando modificarem dados.

Veja o item Tipo de Bloqueio anteriormente para entender os tipos de mensagens que aparecem quando acontecem conflitos no modo WRITE(escrita)

Modo de Pré-alocação:

Ele pode ser chamado de “table reservation” (reserva de tabela) na documentação oficial. Como SET TRANSACTION está disponível em SQL, DSQL e ISQL, ele pode ser embutido em aplicações SQL. Ele permite um fino acabamento através dos recursos requisitados (neste caso, tabelas) quando uma txn inicia no lugar de ficar aguardando, a txn tenta suas operações na tabela. Esta técnica reduz a possibilidade de deadlocks. Eu pularei a cláusula USING (usando) porque ela está disponível apenas para SQL embutido (não em SQL Dinâmico) e especifica manipuladores para bancos de dados completos, então isto limita apartir do início o número de bancos acessados pela txn. Com respeito à cláusula RESERVING (reservando), ela é seguida de uma lista de tabelas separadas por vírgula e depois a palavra reservada FOR que tem as seguintes opções de compartilhamento:

- SHARED: as tabelas podem ser compartilhadas com outras txns concorrentes para leitura.
- PROTECTED: as tabelas não podem ser compartilhadas com outras txns para leitura ou gravação.

e estas opções de modo de acesso:

- READ: a txn irá querer antecipar somente usos do tipo RO, de acordo com uma ou outra opção de compartilhamento mostrada anteriormente.
- WRITE: a txn irá querer antecipar uso de WRITE nas tabelas, de acordo com uma ou outra opção de compartilhamento mostrada anteriormente.

então você tem 4 possibilidades de combinação. Então depois da vírgula você pode colocar uma lista de tabelas separadas por vírgula e seguida da palavra reservada FOR e seu modo de pré alocação desejado. Tente uma dessas quatro combinações:

- Shared, write: permite a qualquer transação com modo de acesso WRITE e nível de isolamento “concurrency” ou “read committed”, atualizar enquanto outras transações com outros níveis de isolamento e acesso READ podem ler dados.
- Shared, read: permite a qualquer transação ler dados, e qualquer transação com acesso WRITE a atualizar. Este é modo de reserva mais liberal.

- Protected, write: previne que outras transações atualizem. Outras transações com níveis de isolamento “concurrency” ou “read committed” podem ler dados, mas somente esta transação pode atualizar.
- Protected, read: previne que todas as transações atualizem, mas permite que todas as transações leiam dados.

Um bom começo para tentar diferentes combinações e ver o que acontece, é rodar duas instâncias do utilitário de linha de comando ISQL.EXE e experimentar. Ele suporta completamente a sintaxe mostrada anteriormente (com exceção do USING que não trabalha em SQL Dinâmico, como explicado antes) então você pode ver conflitos, “deadlocks” e mensagens de erro.

É necessário dizer que se você estiver conectado através do BDE, provavelmente não verá o comportamento padrão dos parâmetros. Provavelmente, o driver do FB, no BDE, usa “NO WAIT” e “READ COMMITTED”, por razões de compatibilidade com outros mecanismos.

Também, não custa enfatizar que “generators” (geradores) ficam de fora das transações. Eles têm um valor único para o banco de dados e apenas por um momento, sem levar em consideração o nível de isolamento, a txn usa ele. Além disso, as mudanças feitas no valor de um “generator” por meio de um GEN_ID não são afetadas tampouco por “commit” ou “rollback”, então GEN_ID é mesmo uma chamada global atômica, não só porque retorno um novo valor para todas as transações, mas também, porque suas chamadas são serializadas, isto para garantir a exclusividade do valor de um “generator” quando usado para incrementar ou decrementar o valor atual.

A tabela a seguir mostra a equivalência dos nomes usados para diferentes níveis de isolamento:

Nível de constantes da API & IBO	Nível de linguagem & ferramentas	Nível do BDE
Não suportado	Não suportado	Dirty Read (leitura suja)
Read Committed	Read Committed	Read Committed
Concurrency	Snapshot	Repeatable Read
Consistency	Snapshot table stability	Não suportado

Artigo Original http://www.cvalde.com/document/TransactionOptions.htm Claudio Valderrama http://www.cvalde.com/index.htm	
Tradução e adaptação: **Jiancarlos Kleinebing** polo@softline.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.	