



Tudo que você precisa saber sobre transações

Carlos H. Cantu
www.firebase.com.br





- Um dos maiores **benefícios** da mudança de bancos desktop para Cliente/Servidor.
- **Modelo** de gerenciamento **varia** entre os SGBDs.
- Visa manter a **consistência** dos dados.
- **Controle e seqüência** de operações são de **responsabilidade do desenvolvedor**.
- “Protegem” os dados no caso de falha no lado cliente, mas **não** do lado servidor (use nobreak, etc!)





- **MGA** – Multi Generational Architecture (conhecida também por MVCC)
- Armazena as alterações diretamente no banco de dados, juntamente com informações para reverter ao estado original (*backversions*).
- Dispensa *log* de transações.





- **A**tomicidade
 - **C**onsistência
 - **I**solamento
 - **D**urabilidade
-
- Firebird é totalmente compatível com ACID





- Todas as operações realizadas dentro de uma única transação, são parte de uma única operação indivisível.





- O banco de dados deve ficar num estado consistente, tão logo a transação seja completada.





- Cada transação deve ficar completamente isolada de outras transações concorrentes, fazendo com que as alterações nos dados não sejam expostas a outras transações antes da transação *commitar*.





- Assim que a transação se completar, as alterações nos dados se tornam permanentes.
- Durante o *commit*, o SGBD deve registrar que a transação foi terminada com sucesso, e realizar outros trabalhos, por exemplo, no caso do *2 phase commit* ou *eventos*.





- Alterações nos dados são escritas diretamente no disco (não ficam na memória*), bem como os dados necessários para reverter para o estado original. Esses dados ocupam espaço temporariamente.
- Necessário uma forma de controlar esse lixo (**G**arbage **C**ollection).

* Não confundir com cache de disco.





- O BD contem sempre a informação da última atualização realizada.
- Para voltar ao estado anterior – é necessário log de transações.
- Implicações:
 - Alto custo de *rollback*
 - É possível voltar o BD em qualquer período de tempo.
 - Fácil limpar os logs de operações (separados do BD)
 - Isolamento entre transações utiliza travas, pois o BD tem somente a última informação escrita.





- O BD contem todas as versões de um registro que estão sendo utilizadas por diferentes transações concorrentes, dispensando o log de transações.
- Implicações:
 - Rollbacks “baratos”.
 - Sujeira no BD.
 - “Impossível” voltar o BD para um determinado momento no tempo.
 - Transações podem ler os registros sem necessidade de travas.





- Um mesmo registro pode ter diferentes versões disponíveis.
- Cada nova versão contém um link apontando para a versão anterior, formando uma cadeia de versões.
- Cada *backversion* possui o número da transação que o criou.
- Leitura começa sempre pela versão mais recente.
- Criados por *updates* ou *deletes*.





- Backversion:
 - Delta – contém apenas os valores dos campos alterados.
 - Full – contém os valores de todos os campos do registro.
- Full – é criado se a mesma transação atualiza o mesmo registro pela segunda vez, ou se o tamanho do Delta ficar maior que a de um full.



Iniciando uma transação no Firebird



- SET TRANSACTION
[READ WRITE | READ ONLY]
[WAIT | NO WAIT [LOCK TIMEOUT]] /* resolução de conflitos */
[ISOLATION LEVEL] /* isolamento */
{SNAPSHOT [TABLE STABILITY] | READ COMMITTED
[[NO] RECORD_VERSION]]}
[RESERVING <reserving-clause> |
USING <db-handle> [, db-handle ...]];
- <reserving-clause> ::= <table> [, <table> ...]
[FOR [SHARED | PROTECTED] {READ | WRITE}]
[, <reserving-clause> [, <reserving-clause>..]]





COMMIT /* Encerrar confirmando as alterações */

ROLLBACK /* Encerrar desfazendo as alterações */

COMMIT RETAIN /* Confirma alterações, mas mantém o contexto da transação */

ROLLBACK RETAIN /* Confirma alterações, mas mantém o contexto da transação */





- O controle transacional é feito pela aplicação cliente.
- Não há como iniciar ou encerrar uma transação de dentro de triggers ou stored procedures.





- Deadlock é a forma que o Firebird tem para avisar que um **conflito** de atualizações foi encontrado.
- A aplicação cliente deve identificar os *deadlocks* e decidir o que fazer.





- **NO WAIT** - Determina se conflitos encontrados devem retornar imediatamente um *deadlock*.
- **WAIT** – Faz com que a transação que encontrou um conflito aguarde que a outra transação faça um *rollback* ou um *commit*.
- A partir do FB 2, é possível determinar um **timeout** para o modo WAIT.





- Determinam como as transações concorrentes interagem e enxergam os dados entre si.
- Isolamentos “comuns”:
 - Dirty Read
 - Read Committed
 - Repeatable Read
 - Serializable





- Não é suportado pelo Firebird (não faz sentido para o FB, pois ele permite que as transações concorrentes enxerguem os dados previamente *commitados*).





- A transação só pode ver os dados que já estão *commitados*.
- Geralmente, nos outros BDs, as transações não conseguem ler registros que estão com alterações pendentes.
- Indicado para “browse”, manutenção de dados.
- FB suporta dois modos neste isolamento:
 - *record_version*
 - *no_record_version* (imita os outros SGBDs que usam travas/log de transações)





- Firebird fornece isolamento similar: **Concurrency/Snapshot**
- “Foto” do BD - a transação não enxerga alterações feitas por outras transações.
- Diferença para o *Repeatable Read*: o Firebird não garante que os registros lidos poderão ser alterados pela própria transação (outra transação pode alterar ele primeiro).
- Indicado para relatórios.





- Também chamado de **Snapshot Table Stability**
- Semelhante ao Snapshot, mas garante que os registros lidos poderão ser atualizados pela transação.
- Bloqueia outras transações de gravar nas tabelas que foram acessadas pela transação.
- **Nota:** A tabela é travada somente depois que alguma instrução SQL a acessou.





- Permite iniciar uma transação e colocar travas em tabelas específicas.
- A trava é colocada assim que a transação é iniciada.
- Sucesso depende do WAIT/NOWAIT e se há escritas pendentes feitas por outras transações.
- Permite travar tabelas "dependentes" (que sofrem alterações por *triggers*, etc)





- Modos suportados:
 - **Shared read.** Todas as transações podem ler e gravar (maior grau de concorrência).
 - **Shared write.** Todas as transações, **exceto** as com isolamento ***consistency*** podem ler e gravar.
 - **Protected read.** Todas as transações (incluindo esta) só podem ler.
 - **Protected write.** Todas as transações, exceto as *consistency*, podem ler, mas somente esta transação pode gravar.





- SQL> set transaction
isolation level read committed
reserving *clientes* for **protected**
write;
- Faz com que a tabela *clientes* só possa ser gravada pela transação atual.





- Permite que uma mesma transação esteja **conectada à diferentes** bancos de dados.
- Útil, por exemplo, em replicações síncronas.
- Pode gerar transações em **limbo**.





- Faz a limpeza do lixo.
- No FB ≥ 2.0 pode ser *cooperativa* ou *background*.
- Apaga as versões dos registros anteriores à OST, diminuindo o tamanho da “cadeia” de *backversions*.
- Libera o espaço ocupado pelos *backversions* para ser reaproveitado.





- Limpeza do banco de dados.
- Pode ser disparado manualmente ou automaticamente (OAT – OIT).
- Percorre todas as tabelas do BD, descartando as *backversions*, liberando o espaço para ser reaproveitado.
- Incrementa a OIT. Se o acesso for exclusivo, o número será o da Next Transaction menos 1.





Database "employee.fdb"

Database header page information:

Flags	0
Checksum	12345
Generation	199
Page size	4096
ODS version	11.1
Oldest transaction	194 -- OIT
Oldest active	195 -- OAT
Oldest snapshot	192 -- OST
Next transaction	197 -- NT
Bumped transaction	1
Sequence number	0
Next attachment ID	8
Implementation ID	16
Shadow count	0
Page buffers	0
Next header page	0
Database dialect	3
Creation date	Feb 27, 2009 23:24:29
Attributes	force write

Variable header data:

END





- **OAT** – **O**ldest **A**ctive **T**ransaction
- **OIT** – **O**ldest **I**nteresting **T**ransaction - transações com estado diferente de *commit*, ex: *active*, *limbo*, *rolled back*.
- **OST** = **O**ldest **S**napshot **T**ransaction.
- Os valores da OAT, OST, OIT e Next são atualizados sempre que uma nova transação é iniciada.





- Deadlock gerado com um *select*.
 - Transações com isolamento **read committed no record_version** e **no wait**;
- Transações **Read Committed READ ONLY** não bloqueiam a coleta de lixo.
- **90%** das reclamações de lentidão com o Firebird estão relacionadas com o controle incorreto das transações (ou falta dele).



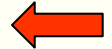


- Commit Retaining

Operação	Trans. ID	OAT
Start Transaction	1	1
Commit	1	1
Start Transaction	2	2
Commit Retaining	2	2
Commit Retaining	2	2
Commit Retaining	2	2
Commit Retaining	2	2
Commit	2	2
Start Transaction	3	3



Operação	Trans. ID	OAT
Start Transaction	1	1
Commit	1	1
Start Transaction	2	2
Update...		2
Commit Retaining	→ 3	2
Commit Retaining	3	2
Update...		2
Commit Retaining	→ 4	2
Commit Retaining	4	2
Commit	4	2
Start Transaction	5	5





- **FB Scanner**
- **FB DataGuard**

- Versões de avaliação e Free no CD do FDD.





Dúvidas?

