Controle de Concorrência

Banco de Dados II

Profa. Késsia R. C. Marchi

Transação

- Transação é uma unidade lógica de trabalho, envolvendo diversas operações de bancos dados.“

C. J. Date

- Uma transação inicia-se, basicamente, com uma instrução BEGIN e termina com um COMMIT ou ROLLBACK;

Vantagens no uso de transações

- Não carregar com processamento no lado cliente;
- Ter uma estrutura modular;
- Evitar perda de informações;
- Diminuir o tráfego na rede já que estaremos trafegando parâmetros e os entregando ao banco de dados;
- Ter controle sobre os erros que possam acontecer com as lógicas de definição de fluxo de dados;
- Controle de atualizações de registros através de bloqueios.

Commit e Rollback

- COMMIT: indica a conclusão de uma transação bem-sucedida (Parcialmente confirmado). Informando ao gerenciador de transações do SGBD que uma unidade lógica de trabalho foi concluída com sucesso, com essa ação, o banco retoma o seu estado consistente e todas as atualizações feitas por essa unidade já podem se tornar permanentes (Confirmado).

- ROLLBACK: indica a conclusão de uma transação malsucedida (Em falha). Ela informa ao gerenciador a ocorrência de uma falha, logo, nesse momento, o banco pode estar incosistente, portanto, todas as atualizações realizadas pela transação devem ser desfeitas (Abortada).

Técnicas de Controle de Concorrência

- Uma técnica tradicional é o Bloqueio (lock);
  - MySql
  - PostGreSql
- MVCC – Multiversion Concurrence Control;
  - PostGreSql

MVCC

- Cada transação enxerga uma versão do banco de dados (SNAPSHOT);
  - Não considera o estado concorrente dos dados provocados pelas demais transações.
  - Impede que seja visualizado dados inconsistêntes.

Diferença

- MVCC x Locks
  - no MVCC, os bloqueios obtidos para consultar (ler) os dados não conflitam com os bloqueios obtidos para escrever os dados e, portanto, a leitura nunca bloqueia a escrita, e a escrita nunca bloqueia a leitura.

Isolamento

- Nível de isolamento determina o grau em que as operações de uma sessão podem afetar os dados observados ou acessados por outra sessão.
- Os 4 níveis de isolamento representam um balanceamento entre Isolamento e Concorrência da transação.
  - Nível alto → Menor capacidade de executar concorrência
  - Nível baixo → Maior capacidade de executar concorrência. Logo, maior chance de obter conflitos.

Objetivos do Isolamento

- Evitar 3 fenômenos em transações concorrêntes:
  1. **dirty read (leitura suja)**
     - A transação lê dados não efetivados (uncommitted) escritos por uma transação concorrente.

UM EXEMPLO DE *DIRT READ*

Transação 1	Dados vistos pela Transação 1	O que o *Dirt Read* em outra transação veria	O que outras transações veriam se o *Dirt Read* não ocorresse
postgres=# BEGIN WORK; BEGIN	Juliano	Juliano	Juliano
postgres=# UPDATE tbl_usuario SET nome='Marcio' WHERE usuid=1; UPDATE 1	Marcio	Marcio	Juliano
postgres=# COMMIT WORK; COMMIT	Marcio	Marcio	Marcio
postgres=# BEGIN WORK; BEGIN	Marcio	Marcio	Marcio
postgres=# UPDATE tbl_usuario SET nome='Andrea' WHERE usuid=1; UPDATE 1	Andrea	Andrea	Marcio
postgres=# ROLLBACK WORK; ROLLBACK	Marcio	Marcio	Marcio

Objetivos do Isolamento

- Evitar 3 fenômenos em transações concorrêntes:
  2. **nonrepeatable read (leitura que não pode ser repetida)**
     - A transação lê uma segunda vez os dados, e descobre que os dados foram modificados por outra transação (que os efetivou após ter sido feita a leitura anterior).

UM EXEMPLO DE *UNREPEATABLE READ*

Transação 1	Dados vistos pela Transação 1	O que o *Unrepeatable Read* em outra transação veria	O que outras transações veriam se o *Unrepeatable Read* não ocorresse
postgres=# BEGIN WORK; BEGIN	Marcio	Marcio	Marcio
postgres=# UPDATE tbl_usuario SET nome='Mauricio' WHERE usuid=1; UPDATE 1	Mauricio	Marcio	Marcio
postgres=# COMMIT WORK; COMMIT	Mauricio	Mauricio	Marcio
postgres=# SELECT nome FROM tbl_usuario WHERE usuid=1; (1 row)	Mauricio	Mauricio	Mauricio

Objetivos do Isolamento

- Evitar 3 fenômenos em transações concorrêntes:

3. phantom read (leitura fantasma)

- A transação executa uma segunda vez uma consulta que retorna um conjunto de linhas que satisfaz uma determinada condição de procura, e descobre que o conjunto de linhas que satisfaz a condição é diferente devido a uma outra transação efetivada recentemente.

UM EXEMPLO DE *PHANTOM READ*

Transação 1	Transação 2
postgres=# BEGIN WORK; BEGIN	postgres=# BEGIN WORK; BEGIN
postgres=# UPDATE tbl_usuario SET nome = nome || ','; UPDATE 1	
	postgres=# INSERT INTO tbl_usuario (nome) VALUES ('Adriana'); COMMIT
postgres=# COMMIT WORK; COMMIT	
	postgres=# COMMIT WORK; COMMIT

O INSERT começou antes da atualização ser confirmada, então,
esperamos que seja razoável que o nome inserido esteja com a v írgula adicionada.

Se um *Phantom Read (leitura fantasma)* ocorre, então a nova tupla que aparece depois da Transação 1 determina quais tuplas serão atualizadas, e a v írgula do novo item não é concatenada.

Níveis de Isolamento

- Read Uncommitted
  - Nível mais baixo de isolamento
  - Também conhecido como leitura suja.
  - Permite que sejam lidas linhas que não tenham sido commitadas.
  - Utilizado para melhorar performance

Níveis de Isolamento

- Read Commited
  - São visualizadas apenas, linhas já commitadas por outras transações.
  - Exemplo:
    - Sessão A realiza um select na tabela cliente;
    - Ao mesmo tempo a sessão B realiza um insert na mesma tabela;
    - Esse cliente inserido pela sessão B não será visível no select da visão A.

Níveis de Isolamento

- Repeatable read
  - Nível de isolamento padrão
  - No banco, todas as leituras são consistentes.

Níveis de Isolamento

- Serializable
  - A transação fica completamente isolada.
  - Comparta-se como se tivesse executada serialmente.

Isolamento

Nível de isolamento	Dirty Read	Nonrepeatable Read	Phantom Read
Read uncommitted	Possível	Possível	Possível
Read committed	Impossível	Possível	Possível
Repeatable read	Impossível	Impossível	Possível
Serializable	Impossível	Impossível	Impossível

Verificando o nível de isolamento

- Para verificar o nível de isolamento que está aplicado utilize:
- MySQL
  - SELECT @@tx_isolation;
  - SELECT @@global.tx_isolation;

Definindo o nível de isolamento

- Global:
  - Estabelece o nível de isolamento globalmente para todas as novas conexões criadas a partir deste ponto.

SET [SESSION | GLOBAL] TRANSACTION ISOLATION LEVEL {READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE}

AUTOCOMMIT
(Modo Encadeado ou Não Encadeado)

- Recurso que auxilia no controle da efetividade das transações no servidor

- Transações implícitas (Não Encadeado)
  - Já possuem commit interno.
  - Padrão para SGBDs como o MySQL e o PostgreSQL
- Transações explícitas (Encadeado)
  - Possuem início e fim indicado pelo desenvolvedor.

Autocommit

- MySQL
  - AUTOCOMMIT = 1 indica que a transação terá um commit interno
  - AUTOCOMMIT = 0 indica que a transação esperará por um commit explícito.

Autocommit

- PostgreSQL
  - Não há uma instrução explícita para indicar o modo de Autocommit.
  - Ao indicar o início de uma transação com BEGIN COMMIT já altera o modo para encadeado.

SAVEPOINT

- Permite criar pontos de salvamento para recuperação através do ROLLBACK TO SAVEPOINT
- É obrigatoriamente vinculado a uma transação.
- Sintaxe:
  - SAVEPOINT *nomeponto;*
  - ROLLBACK SAVEPOINT *nomeponto;*

Bloqueios

- Utilizados para isolar diferentes transações entre usuários;
  - Compartilhado (SHARE)
    - Permite bloquear transações para leitura de tuplas.
    - Se transações distintas solicitarem bloqueio compartilhado, todos são concedidos de imediato.
    - Havendo um bloqueio compartilhado, não é possível conseguir um bloqueio exclusivo até que o bloqueio compartilhado conclua.
  - Exclusivo (EXCLUSIVE)
    - Permite bloquear transações para alteração ou exclusão de tuplas.

Bloqueios em demais instruções SQL

- Delete e Update
  - Estabelece um bloqueio exclusivo de next-key em todos os registros da pesquisa.
    - Bloqueio de next-key
      - Significa que além dos registros de índices , também são bloqueadas as "lacunas" anteriores ao registro de índice. Isto é feito para bloquear alterações por outros usuários imediatamente antes do registro de índice.

Bloqueios em demais instruções SQL

- Insert e Replace
  - Estabelece um bloqueio exclusivo na linha inserida.

Bloqueios em demais instruções SQL

- Auto-increment
  - Bloqueio exclusivo no fim do índice no qual está armazenado o auto-increment.
  - Esse bloqueio ocorre apenas até o final da inserção e não até o final da transação.
- Chave Estrangeira
  - Aplica o bloqueio compartilhado, e é validado as condições aplicada através do constraints.

Bloqueios

- PostgreSQL
  - Além dos mecanismos convencional de bloqueios, o postgre também utiliza o modelo de MVCC.

Bloqueios

- Atualmente, há a necessidade de preocupar-se com bloqueios em somente 2 situações:
  - Deadlocks (Abraço Mortal)
  - Bloqueio Explícito

Bloqueios

- Deadlock
  - O que acontece quando duas diferentes aplicações tentam e alteram os mesmos dados na mesma hora?

Bloqueios Explícitos

- Usado quando há a necessidade de controlar explicitamente as transações.
- O Padrão SQL não possui definição de modo de bloqueio para uma tabelas, essa atividade é extensões dos SGBDs.

Bloqueios Explícitos

- Bloqueando Registros
  - Select .... For Update;

- Bloqueando Tabelas
  - LOCK [TABLE] table

Referências

- **Korth,** Henry F. e Silbershcatz, Abraham; **Sistemas de Banco de Dados**; Elsevier; 5ª Ed., 2006.
- **Elmasri**, Ramez; **Sistemas de Banco de Dados**; Addison Wesley, 2005.
- Batista, Lígia; Transações com PostgreSQL. Disponível em http://inf.cp.cefetpr.br/ligia/material/bd2/transacoes/transacoes_imasters3.pdf