GERENCIAMENTO E CONTROLE DE CONCORRÊNCIA EM TRANSACTIONS

ANTONIO DAVID VINISKI
antonio.david@pucpr.br
PUCPR

TRANSACTIONS

O QUE SÃO TRANSAÇÕES?

- Uma transação no MySQL é um grupo sequencial de instruções, consultas ou operações como **SELECT**, **INSERT**, **UPDATE** ou **DELETE** para executar como uma única unidade de trabalho que pode ser confirmada ou revertida.
- Se a transação fizer várias modificações no banco de dados, duas coisas acontecem:
  - Todas as modificações são bem-sucedidas quando a transação é confirmada.
  - Ou todas as modificações são desfeitas quando a transação é revertida.

Como funciona?

- A transação MySQL permite que você execute um conjunto de operações do MySQL para garantir que o banco de dados nunca contenha o resultado parcial de operações.
  - Em um conjunto de operações, se uma delas falhar, ocorre a reversão para restaurar o banco de dados ao seu estado original.
  - Se nenhum erro ocorrer, todo o conjunto de instruções será confirmado no banco de dados.

Como funciona?

- Em outras palavras, uma transação não pode ser bem-sucedida sem concluir cada operação disponível no conjunto.
  - Isso significa que se alguma instrução falhar, a operação de transação não poderá produzir resultados.
- Uma transação no MySQL começa com a primeira instrução SQL executável e termina quando encontra um commit ou rollback explicitamente ou implicitamente.
- Ele usa explicitamente a instrução **COMMIT** ou **ROLLBACK** e implicitamente quando uma instrução **DDL** é usada.

Propriedades das Transações

- A transação contém quatro propriedades principais, conhecidas como propriedades ACID.
  - Atomicidade.
  - Consistência.
  - Isolamento.
  - Durabilidade.

ACID - ATOMICIDADE

- **Atomicidade**: Esta propriedade garante que todas as instruções ou operações dentro da unidade de transação devem ser executadas com sucesso. Caso contrário, se alguma operação falhar, toda a transação será abortada e será revertida para seu estado anterior.
- Inclui recursos:
  - declaração **COMMIT**.
  - instrução **ROLLBACK**.
  - Configuração de confirmação automática.
  - Dados operacionais das tabelas INFORMATION_SCHEMA.

ACID - CONSISTÊNCIA

- **Consistência**: Esta propriedade garante que o banco de dados mude de estado somente quando uma transação for confirmada com sucesso. Também é responsável por proteger os dados de falhas.
- Inclui recursos:
  - InnoDB doublewrite buffer.
  - InnoDB crash recovery.

ACID - ISOLAMENTO

- **Isolamento**: Esta propriedade garante que cada operação na unidade de transação opere de forma independente. Também garante que as declarações sejam transparentes entre si.
- Inclui recursos:
  - instrução **SET ISOLATION LEVEL**.
  - Configuração de confirmação automática.
  - Os detalhes de baixo nível do bloqueio do InnoDB.

ACID - DURABILIDADE

- **Durabilidade**: Esta propriedade garante que o resultado das transações confirmadas persista permanentemente mesmo se o sistema travar ou falhar.
- Inclui recursos:
  - Buffer de gravação em um dispositivo de armazenamento.
  - Cache com bateria em um dispositivo de armazenamento.
  - Opção de configuração **innodb_file_per_table**.
  - Opção de configuração **innodb_flush_log_at_trx_commit**.
  - Opção de configuração **sync_binlog**.

Instruções de transação do MySQL

- O MySQL nos fornece a seguinte declaração importante para controlar transações:
  - Para iniciar uma transação, você usa a instrução **START TRANSACTION**. O **BEGIN** ou **BEGIN WORK** são os *aliases* da **START TRANSACTION**.
  - Para confirmar a transação atual e tornar suas alterações permanentes, use a instrução **COMMIT**.
  - Para reverter a transação atual e cancelar suas alterações, use a instrução **ROLLBACK**.
  - Para desabilitar ou habilitar o modo de autocommit para a transação atual, use a instrução **SET** autocommit.

EXEMPLO COMMIT

```
SET autocommit = FALSE;

START TRANSACTION; #BEGIN; #BEGIN WORK;

INSERT INTO restaurante.pessoa VALUES (NULL, "Jessica Rocha", "12341234544", "F", "1986-09-23");

SET @pessoa_id := (SELECT LAST_INSERT_ID());

INSERT INTO restaurante.cliente VALUES (@pessoa_id, NOW());

INSERT INTO restaurante.comanda VALUES (NULL, NOW(), 0.0, 0, @pessoa_id, NULL);

SET @comanda_id := (SELECT LAST_INSERT_ID());

COMMIT;
```

INSTRUÇÃO SAVEPOINT

- Um **SAVEPOINT** é um ponto de reversão lógico dentro de uma transação.
- Quando você define um ponto de salvamento, sempre que ocorrer um erro após um ponto de salvamento, você pode desfazer os eventos que você fez até o ponto de salvamento usando o **ROLLBACK**.
- MySQL InnoDB fornece suporte para as instruções **SAVEPOINT**, **ROLLBACK TO SAVEPOINT**, **RELEASE SAVEPOINT**.
- A instrução **SAVEPOINT** é usada para definir um ponto de salvamento para a transação com o nome especificado.
  - Se já existir um ponto de salvamento com o nome fornecido, o antigo será excluído.;

EXEMPLO SAVEPOINT

```
SET autocommit = FALSE;
START TRANSACTION;
SELECT * FROM garcom g JOIN funcionario f ON g.id = f.id JOIN pessoa p ON p.id = f.id;
UPDATE garcom SET valor_hora = valor_hora + 1;
SAVEPOINT insert_garcom;
INSERT INTO restaurante.pessoa VALUES (NULL, "Mariana Fernandes", "34341234544", "F", "1991-04-05");
SET @pessoa_id := (SELECT LAST_INSERT_ID());
INSERT INTO restaurante.funcionario VALUES (@pessoa_id, "mariana@restaurante.com","41999090900",
NOW());
 INSERT INTO restaurante.garcom VALUES (@pessoa_id, 90.00);
SELECT * FROM garcom g JOIN funcionario f ON g.id = f.id JOIN pessoa p ON p.id = f.id;
ROLLBACK TO SAVEPOINT insert_garcom;
COMMIT;
SELECT * FROM garcom g JOIN funcionario f ON g.id = f.id JOIN pessoa p ON p.id = f.id;
```