Transações

Herysson R. Figueiredo
herysson.figueiredo@ufn.edu.br

Sumário

- Transação
- ACID
  - Atomicidade;
  - Consistência;
  - Isolamento;
  - Durabilidade;
- Comando TRANSACTION

Transações

Uma transação em banco de dados é um **conjunto de operações** que são tratadas como uma **única unidade** de trabalho.

Transações

Essas operações podem incluir **inserções, atualizações, exclusões ou consultas,**

Transações

Uma transação deve ser **completamente concluída** ou **completamente revertida** para garantir que o banco de dados permaneça em um **estado consistente.**

Transações

O principal objetivo de uma transação é garantir a **integridade e consistência dos dados**, mesmo diante de falhas, como erros no sistema ou interrupções inesperadas.

ACID

ACID:

- Atomicidade,
- Consistência,
- Isolamento
- Durabilidade.

Atomicidade

A atomicidade garante que uma transação é tratada como uma **única unidade**, o que significa que ela deve ser completamente concluída ou totalmente desfeita. Se qualquer parte da transação falhar, todo o resto também falha.

Consistência

A consistência garante que uma transação leve o banco de dados de um **estado válido** para **outro estado válido**, respeitando todas as regras definidas, como chaves primárias, integridade referencial, etc.

Isolamento

O isolamento garante que as transações sejam **executadas de forma isolada,** sem que as operações de **uma transação afetem as operações de outra.** O nível de isolamento pode variar, afetando a visibilidade das mudanças feitas por transações concorrentes.

Durabilidade

A durabilidade garante que, uma vez que uma transação é **confirmada (committed)**, ela permanecerá no banco de dados, mesmo que ocorra uma falha no sistema. Os dados serão **persistidos** no armazenamento.

O comando TRANSACTION

É utilizado para gerenciar uma **sequência de operações** (transações) no banco de dados, garantindo que essas operações sejam **executadas** de **maneira segura e consistente.**

Para que serve uma transação?

- Garantir integridade e consistência dos dados:
- Reverter alterações em caso de erro:
- Controlar múltiplas operações simultâneas:

Para que serve uma transação?

- **Garantir integridade e consistência dos dados:** Se uma série de operações afeta vários registros ou tabelas, uma transação garante que todas essas operações sejam concluídas corretamente antes de serem confirmadas.
- **Reverter alterações em caso de erro:**
- **Controlar múltiplas operações simultâneas:**

Para que serve uma transação?

- **Garantir integridade e consistência dos dados:**
- **Reverter alterações em caso de erro:** Se ocorrer um erro durante qualquer uma das operações de uma transação, você pode reverter todas as operações feitas até o momento, evitando deixar o banco de dados em um estado inconsistente.
- **Controlar múltiplas operações simultâneas:**

Para que serve uma transação?

- **Garantir integridade e consistência dos dados:**
- **Reverter alterações em caso de erro:**
- **Controlar múltiplas operações simultâneas:** Em ambientes com múltiplos usuários e processos, transações ajudam a isolar as operações e prevenir problemas como "leituras sujas" ou "atualizações perdidas".

Comandos Relacionados a Transações

BEGIN TRANSACTION: Inicia uma nova transação.

COMMIT TRANSACTION: Confirma a transação, aplicando permanentemente todas as operações feitas no banco de dados.

ROLLBACK TRANSACTION: Desfaz todas as operações realizadas desde o início da transação.

SAVEPOINT: Define um ponto dentro de uma transação para permitir um rollback parcial, até esse ponto

Comandos Relacionados a Transações

```
BEGIN TRANSACTION;
“Bloco”
IF @@ERROR <> 0
        ROLLBACK TRANSACTION;
ELSE
        COMMIT TRANSACTION;
```

Comandos Relacionados a Transações

```
BEGIN TRANSACTION;
“Bloco”
IF @@ERROR <> 0
        ROLLBACK TRANSACTION;
ELSE
        COMMIT TRANSACTION;
```

Crie uma transação para realizar duas inserções dentro de uma transação: uma na tabela FUNCIONARIO e outra na tabela DEPARTAMENTO. Se a segunda inserção falhar, a primeira será revertida.

Comandos Relacionados a Transações

```
BEGIN TRANSACTION;
“Bloco”
IF @@ERROR <> 0
        ROLLBACK TRANSACTION;
ELSE
        COMMIT TRANSACTION;
```

Escreva uma transação que tente inserir um funcionário com um Cpf_supervisor que não existe. O banco de dados deve rejeitar a operação, e a transação deve ser revertida.

Comandos Relacionados a Transações

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN TRANSACTION;

“Bloco”

WAITFOR DELAY '00:00:10'

COMMIT TRANSACTION;

BEGIN TRANSACTION;

COMMIT TRANSACTION;
```

Execute duas transações simultaneamente para testar o isolamento de transações no SQL Server. Utilize o nível de isolamento SERIALIZABLE para garantir que uma transação bloqueie a outra até ser finalizada.

Comandos Relacionados a Transações

```
BEGIN TRANSACTION;
“Bloco”
IF @@ERROR <> 0
        ROLLBACK TRANSACTION;
ELSE
        COMMIT TRANSACTION;
```

Escreva uma transação que insira dados em uma tabela, mas deve falhar devido a uma violação de restrição de chave primária. A transação deve ser revertida.

Comandos Relacionados a Transações

BEGIN TRANSACTION;

“Bloco”

IF “condição”

    ROLLBACK TRANSACTION;

ELSE

    COMMIT TRANSACTION;

Crie uma transação que atualize o salário de todos os funcionários de um determinado departamento. Se a atualização de qualquer funcionário falhar, reverta todas as alterações.

Comandos Relacionados a Transações

BEGIN TRANSACTION;

“Bloco”

IF “condição”

    ROLLBACK TRANSACTION;

ELSE

    COMMIT TRANSACTION;

Implemente uma transação que simule uma compra. Verifique o estoque antes de permitir a compra e, se o estoque for insuficiente, reverta a transação. Use uma tabela chamada PRODUTO para registrar o estoque dos produtos. (Crie uma tabela de Produto com os campos ID, Nome e quantidade)

Comandos Relacionados a Transações

BEGIN TRANSACTION;

“Bloco”

IF “condição”

    ROLLBACK TRANSACTION;

ELSE

    COMMIT TRANSACTION;

Implemente uma transação que simule a transferência de fundos entre duas contas bancárias. Se o saldo da conta de origem for insuficiente, a transação deve ser revertida. (Crie uma tabela conta com Id, Nome, Saldo)

Comandos Relacionados a Transações

```
BEGIN TRANSACTION;

“Bloco’’

SAVE TRANSACTION SavePointProjeto;

“Bloco"

IF “condição”

        ROLLBACK TRANSACTION;

ELSE

        COMMIT TRANSACTION;
```

Implemente uma transação que insira dados em três tabelas diferentes. Use um SAVEPOINT para marcar um ponto dentro da transação. Se uma parte falhar, reverta apenas até o ponto de salvamento (savepoint) sem desfazer toda a transação.

Bibliografia

- Christopher John Date. An introduction to database systems. Pearson Education India, 1981 (ver página 27).
- Ramez Elmasri e Sham Navathe. Fundamentals of Database Systems. 7ª edição. Pearson, 2016 (ver páginas 7, 8, 10, 16, 17, 19, 20).
- Nenad Jukic, Susan Vrbsky e Svetlozar Nestorov. Database systems: Introduction to databases and data warehouses. Pearson, 2014 (ver página 23).
- Michael McLaughlin. MySQL Workbench: Data Modeling Development. McGraw Hill Professional, 2013 (ver página 23).
- SQL Tutorial. w3schools, 2022. Disponível em: https://www.w3schools.com/sql/default.asp. Acesso em: 20, abril de 2022