Prof. Taciano Balardin

www.taciano.pro.br

taciano@ulbra.edu.br

2015-2

Banco de Dados II

E-MAIL DE CONTATO:
taciano@ulbra.edu.br

SITE DA DISCIPLINA:
http://www.taciano.pro.br/

Transações em Banco de Dados

BANCO DE DADOS II
AULA 15

Prof. Taciano Balardin

Transações

Etapas para um saque de R$ 100,00

1. Ler o saldo atual
2. Saldo = Saldo Atual - Valor do Saque
3. Gravar Saldo
4. Entregar o Valor

Etapas para uma transferência de R$ 100,00 (Poupança → CC)

1. Ler o saldo atual da PP
2. Saldo PP = Saldo Atual PP - Valor Transferido
3. Ler o saldo atual da CC
4. Saldo CC = Saldo Atual CC + Valor Transferido
5. Gravar Saldo CC
6. Gravar Saldo PP

Se você perceber que há **4 maneiras** de uma coisa **dar errada**, e confiante dribla **as quatro**, de repente... a **Lei de Murphy**, sempre atuante, faz uma **quinta maneira** surgir do nada!

Transações

- Sequência de ações (operações de banco de dados) que são executadas como um bloco único e indivisível.

  Ou todas são executadas com sucesso

  Ou nenhuma delas é executada

- Exemplo:
  - Um banco transfere dinheiro entre duas contas (saque + depósito);
  - Se **qualquer uma das operações falhar**, a transferência será **cancelada**.

Transações

- Transações podem ser **implícitas** ou **explícitas**.
  - Todo comando **DML** ou **DDL** que é enviado ao SGBD é considerado uma **transação implícita**, ou seja, após dispararmos o comando, via terminal, por exemplo, o SGBD se encarregará de enviar um ***commit*** ao final.
  - No MySQL temos uma variável de ambiente que controla tal característica, denominada ***autocommit***, que pode ser acessada através do seguinte comando:

```
mysql> SELECT @@autocommit;
```

  - O padrão do ***autocommit*** é 1, implicando diretamente no comportamento do SGBD para cada comando que é enviado pelo usuário de um banco de dados, pois, após cada comando, a declaração ***commit*** é enviada automaticamente para tornar permanente aquela transação.

Transações

- Transações podem ser **implícitas** ou **explícitas**.
  - Caso configuremos o ***autocommit*** como 0 (**SET AUTOCOMMIT=0;**), teremos uma situação em que, caso não enviemos um ***commit*** explicitamente após uma transação implícita, a última transação não se tornará permanente no arquivo de dados, pois, o SGBD considerará uma transação ainda aberta.

```
# comando 1
UPDATE tabela1 SET valor = 100 WHERE id = 14;
# comando 2
SELECT valor FROM tabela1 WHERE id = 14;
COMMIT;
```

Transações

- Transações podem ser **implícitas** ou **explícitas**.
  - Caso configuremos o ***autocommit*** como 0 (**SET AUTOCOMMIT=0;**), teremos uma situação em que, caso não enviemos um ***commit*** explicitamente após uma transação implícita, a última transação não se tornará permanente no arquivo de dados, pois, o SGBD considerará uma transação ainda aberta.

```
# comando 1
UPDATE tabela1 SET valor = 100 WHERE id = 14;
# comando 2
SELECT valor FROM tabela1 WHERE id = 14;
ROLLBACK;
```

Transações

Garantem a **consistência** e **integridade** do banco de dados, pois todas as operações são executadas como uma unidade (ou nenhuma será).

- Características:
  - Todas as modificações da transação são temporárias;
  - Modificações serão “persistidas” apenas após o ***commit***;
  - A qualquer momento (antes do ***commit***) as modificações podem ser canceladas através de um ***rollback***.

Propriedades ACID

Todo SGBD **aplica os conceitos de ACID**, pois, caso isto **não ocorra** ele **não pode ser considerado** um SGBD.

- ACID é a união das iniciais de:

  Atomicidade

  Consistência

  Isolamento

  Durabilidade


MITO
VERDADE

MySQL Storage Engine

- Storage Engine é o motor de armazenamento do MySQL.
  - Até a **versão 5.5 do MySQL**, o Storage Engine padrão do MySQL era o **MyISAM**, onde o bloqueio de transação era realizado em nível de tabela, ou seja, a tabela trava por completo, enquanto a transação atual não terminar.
  - Outro problema, as tabelas **MyISAM** não possuem suporte a chave estrangeira e suportam somente transações implícitas;

MySQL Storage Engine

- Storage Engine é o motor de armazenamento do MySQL.
  - A partir da **versão 5.5 do MySQL**, o **InnoDB** tornou-se o Storage Engine padrão do MySQL e tem bloqueio em nível de linha, ou seja, caso uma transação *A* esteja atualizando uma linha de uma tabela, a transação *B* terá que esperar até que *A* termine suas atividades para atualizar esta mesma linha, nessa mesma tabela;
  - Este Engine tem suporte à transações e **contempla em 100% o modelo de transação ACID**.

MySQL™

Atende as propriedades ACID.

É um sistema gerenciador de banco de dados!

Propriedades ACID

- **Atomicidade**
- Toda transação deverá ser atômica, o verdadeiro “tudo ou nada”.
  - Exemplo: Uma transferência bancária entre contas de mesmo banco. Imagine que subtraímos o saldo da *conta A* e checamos a existência da *conta B*. Caso a *conta B* exista, somamos o saldo à *conta B*;
  - Agora imaginemos que, ao checar a existência da *conta B*, esta conta não existe! A quantia subtraída da *conta A* não poderá simplesmente desaparecer, afinal, esse dinheiro tem dono;
  - O comando seguinte falhará e haverá um **ROLLBACK**, restaurando o valor antes subtraído à conta de origem, no caso, a *conta A*;

Propriedades ACID

- **Consistência**
  - Transações devem preservar a consistência do banco de dados, ou seja, transforma um estado consistente do banco de dados em outro estado consistente, sem necessariamente preservar o estado de consistência em todos os pontos intermediários;
  - Exemplo 1: O ponto de inconsistência plena do estado do banco de dados no caso da transferência bancária é exatamente no momento em que o saldo é subtraído de uma conta para ser adicionado na outra.
  - Exemplo 2: Consistência de relacionamento entre chaves.

Propriedades ACID

- **Isolamento**
  - Múltiplas transações simultâneas não afetam umas as outras. O nível de isolamento define o quanto uma transação “enxerga” alterações das outras.
    - **Read committed**: permite que a transação atual leia/manipule somente os dados já permanentes ou “comitados” por outras transações.
    - **Read uncommitted**: permite que uma transação veja manipulações não “comitadas” de outras transações.

Propriedades ACID

- **Isolamento**
  - Múltiplas transações simultâneas não afetam umas as outras. O nível de isolamento define o quanto uma transação “enxerga” alterações das outras.
    - **Repeatable read**: todas as leituras dentro de uma transação mostram os mesmos valores de dados. Isto é válido mesmo se uma segunda transação tenha alterado os seus dados e os tenha confirmado, enquanto a primeira continua a correr.
    - **Serializable**: isola completamente uma transação de outra, ou seja, enquanto uma trabalha a outra aguarda para poder iniciar o seu trabalho.

Propriedades ACID

- **Durabilidade**
  - Se uma transação é confirmada (***commit***) ela será persistente e não pode ser perdida nem desfeita.
  - Exemplo: Se após concluída a transferência a energia falhar, ao retornar, os dados continuam íntegros e registrados conforme o momento imediatamente anterior à falha.