| Disciplina: Banco de Dados II  Professor: Dejan M. Conceição | |
| --- | --- |
|  | Universidade Federal do Pará  Instituto de Ciências Exatas e Naturais  Faculdade de Computação |

**Projeto de Banco de Dados: Teste de Controle de Concorrência e Processamento de Transações**

## **Objetivo:** Este projeto tem como objetivo permitir que os alunos testem diferentes conceitos de controle de concorrência e processamento de transações em bancos de dados relacionais, utilizando **MySQL**. Os alunos trabalharão em **equipes de até 3 integrantes**, realizando experimentos para compreender os desafios e as soluções para a concorrência de transações.

## **Estrutura do Projeto:** O projeto consiste em **10 questões práticas**, cada uma abordando um conceito fundamental sobre transações em banco de dados. Os alunos deverão criar e executar transações, analisar os resultados e registrar suas respostas.

Para isso, foi desenvolvido um **banco de dados específico**, contendo as seguintes tabelas:

* **Alunos**: Cadastro dos participantes.
* **Equipes**: Grupos de até 3 alunos.
* **Questões**: Lista das 10 questões que devem ser resolvidas.
* **Tentativas**: Registros das respostas enviadas pelas equipes.
* **Logs de Testes**: Histórico de transações executadas para cada questão.

## **Passos do Projeto**

### **Cadastro dos Alunos e Formação de Equipes**

### Cada aluno será cadastrado no banco de dados com nome, e-mail e matrícula. As equipes serão formadas e associadas a até 3 integrantes.

1. **Acesso às Questões**

As 10 questões estarão disponíveis no banco de dados. Cada equipe deverá acessar a descrição de cada questão e planejar os testes a serem executados.

1. **Execução dos Testes no Banco de Dados**

As equipes deverão executar transações SQL para cada questão, observando os comportamentos e analisando os resultados.

### **Registro das Respostas**

Cada equipe deve registrar no banco de dados uma resposta descrevendo os resultados obtidos no teste e justificando sua resposta.

### **Correção e Feedback**

O professor analisará as respostas registradas e poderá corrigir e dar feedbacks, aprovando ou reprovando cada tentativa.

## **Questões do Projeto:** Cada questão representa um experimento prático a ser realizado no banco de dados.

### **Estados e Propriedades das Transações (ACID)**

Explique os estados de uma transação no modelo ACID (Active, Partially Committed, Committed, Failed e Aborted) e como cada um impacta a consistência do banco de dados. Testar a atomicidade e a consistência ao realizar uma transação e um rollback.

**Active (Ativa):** A transação está em andamento, mas suas alterações ainda não são visíveis e podem ser desfeitas. O banco ainda não está consistente.

**Partially Committed (Parcialmente Cometida):** A transação fez todas as operações, mas ainda não foi confirmada. O banco está temporariamente inconsistente.

**Committed (Cometida):** A transação foi confirmada, e as alterações são permanentes. O banco de dados agora está consistente.

**Failed (Falha):** Ocorreu um erro e a transação não pode continuar. O banco de dados precisa ser revertido para o estado anterior, garantindo a consistência.

**Aborted (Abortada):** A transação foi cancelada e todas as mudanças são desfeitas, retornando o banco ao seu estado anterior, garantindo consistência.

**Impacto na consistência:** Durante os estados *Active* e *Partially Committed*, o banco pode não ser consistente, enquanto nos estados *Committed*, *Failed* e *Aborted*, o banco volta a estar consistente.

1. **Tipos de Escalonamento de Transações**

Qual a diferença entre escalonamento serial e escalonamento equivalente a serial? Dê exemplos práticos de cada caso. Executar transações concorrentes e verificar a ordem dos commits.

Escalonamento Serial: É quando as transações são executadas uma após a outra, sem intercalar operações de diferentes transações. Ou seja, uma transação só começa após a outra ser completamente finalizada.

Escalonamento Equivalente a Serial: São escalonamentos onde, embora as transações sejam intercaladas, o resultado final seria o mesmo de um escalonamento serial, ou seja, a ordem das operações pode mudar, mas o banco de dados ainda permanece consistente como se as transações fossem executadas uma após a outra.

No escalonamento serial, as transações são executadas de forma sequencial. No equivalente a serial, as transações podem ser intercaladas, mas a ordem final das operações pode ser reorganizada sem afetar a consistência.

### **Conflito de Transações**

Em um sistema bancário, duas transações tentam atualizar o saldo da mesma conta ao mesmo tempo. Quais tipos de conflito podem ocorrer e como eles afetam a integridade dos dados? Simular conflitos entre transações que tentam modificar a mesma informação simultaneamente.

Em um sistema bancário, quando duas transações tentam atualizar o saldo da mesma conta ao mesmo tempo, podem ocorrer:

**Conflito de Leitura (Lost Update)**: As transações leem o saldo e atualizam com base na mesma informação, perdendo uma atualização.  
 *Impacto*: Saldo incorreto devido à perda de uma atualização.  
**Conflito de Escrita (Write Skew)**: Duas transações escrevem dados inconsistentes.  
 *Impacto*: Saldo final inconsistente.  
**Conflito de Leitura Suja (Dirty Read)**: Uma transação lê dados não confirmados, podendo usar dados inválidos.  
 *Impacto*: A transação pode cometer com dados incorretos.

### **Seriabilidade de Escalonamento de Transações**

Dado o seguinte escalonamento de operações em duas transações 𝑇1 e 𝑇2, determine se ele é serializável usando o grafo de precedência:

T1: R(A), W(A), R(B), W(B)

T2: R(B), W(B), R(A), W(A)

Analisar um escalonamento e determinar se ele é serializável.

O escalonamento não é serializável devido ao ciclo no grafo de precedência. Isso ocorre porque as transações dependem uma da outra de forma circular, tornando impossível definir uma ordem sequencial válida.

### **Técnicas de Bloqueio de Transações**

O que são bloqueios exclusivos (X) e compartilhados (S)? Explique como eles ajudam a garantir a integridade dos dados em sistemas concorrentes. Utilizar bloqueios em registros para evitar acessos simultâneos indesejados.

**Bloqueios Exclusivos (X)**:  
 São usados quando uma transação precisa de acesso exclusivo a um registro, impedindo que outras transações leiam ou escrevam nesse registro enquanto o bloqueio estiver ativo.  
 **Exemplo**: Uma transação que atualiza o saldo de uma conta utiliza um bloqueio exclusivo para evitar que outras transações acessem ou modifiquem o mesmo registro.

**Bloqueios Compartilhados (S)**:  
 São usados quando uma transação precisa apenas ler um registro, permitindo que outras transações também leiam o mesmo registro, mas impedindo que ele seja modificado.  
 **Exemplo**: Uma transação que consulta o saldo de uma conta pode usar um bloqueio compartilhado, permitindo que outras transações também façam consultas, mas sem alterar o saldo.

**Como ajudam na integridade dos dados**:

* **Bloqueios Exclusivos (X)** evitam que transações concorrentes modifiquem dados simultaneamente, prevenindo conflitos de escrita (como o *Lost Update*).
* **Bloqueios Compartilhados (S)** permitem leituras simultâneas sem risco de dados sujos, garantindo que as transações leiam dados consistentes sem alterá-los.

Esses bloqueios garantem que transações sejam executadas de maneira ordenada e evitam acessos simultâneos indesejados que poderiam comprometer a integridade dos dados.

### **Conversão de Bloqueios de Transações**

Em que situações uma transação pode converter um bloqueio compartilhado em exclusivo? Como essa conversão impacta o desempenho e a concorrência no banco de dados? Testar a conversão de bloqueios compartilhados para exclusivos.

Uma transação converte um **bloqueio compartilhado (S)** em **bloqueio exclusivo (X)** quando precisa modificar um dado que estava apenas sendo lido. Isso é necessário para garantir que ninguém mais acesse ou modifique o dado durante a atualização. A conversão pode **reduzir a concorrência** e aumentar o tempo de espera, pois outras transações ficam bloqueadas até que o bloqueio exclusivo seja liberado. A concorrência diminui, pois outros processos não podem acessar os dados enquanto o bloqueio exclusivo estiver ativo.

### **Bloqueios em Duas Fases (Two-Phase Locking - 2PL)**

Explique o funcionamento do protocolo 2PL e como ele garante a serializabilidade de escalonamentos. Quais são as desvantagens desse método? Observar a fase de crescimento e a fase de liberação de bloqueios.

O **protocolo 2PL (Two-Phase Locking)** é um método que garante a **serializabilidade** de transações em sistemas concorrentes. Ele divide a execução de uma transação em duas fases:

1. **Fase de Crescimento**: A transação pode **adquirir bloqueios** em dados, mas **não pode liberar** nenhum bloqueio. Isso garante que ela tenha controle total sobre os dados necessários para sua execução.
2. **Fase de Liberação**: Depois de liberar o primeiro bloqueio, a transação entra nesta fase, onde **não pode mais adquirir bloqueios**. Apenas liberações podem ocorrer daqui para frente.

O 2PL assegura que as transações sejam executadas de forma ordenada, evitando conflitos e garantindo que o resultado seja o mesmo que se as transações fossem feitas de forma sequencial. No entanto, ele pode causar **deadlocks** (quando transações ficam esperando umas pelas outras) e reduzir a **concorrência**, já que o bloqueio prolongado de recursos pode diminuir o desempenho do sistema.

### **Deadlock e Starvation de Transações**

Qual a diferença entre deadlock e starvation em um banco de dados transacional? Quais estratégias podem ser usadas para preveni-los ou resolvê-los? Criar um deadlock intencional e verificar como o banco lida com a situação.

Deadlock e starvation são problemas comuns em sistemas com várias transações simultâneas:

* Deadlock acontece quando duas ou mais transações ficam presas esperando umas pelas outras. Cada uma está esperando um recurso que a outra já bloqueou, criando um ciclo sem fim onde nenhuma transação pode ser concluída.
* Starvation ocorre quando uma transação nunca consegue acessar os recursos necessários para ser executada. Isso pode acontecer quando transações de maior prioridade continuam sendo processadas, deixando as de menor prioridade esperando indefinidamente.

### Como prevenir ou resolver:

1. Deadlock:  
   * Detecção e Recuperação: O sistema detecta o deadlock e cancela uma das transações para liberar os recursos.
   * Prevenção: Impõe regras que forçam as transações a pegarem os bloqueios de recursos na mesma ordem, evitando ciclos.
   * Evitação: Usa algoritmos que preveem se uma transação pode causar deadlock e bloqueiam a operação antes que o problema aconteça.
2. Starvation:

Escalonamento Justo: Técnicas de escalonamento garantem que todas as transações, independentemente da prioridade, tenham uma chance de serem executadas.  
Ajuste de Prioridade: Mudando as prioridades das transações ao longo do tempo, é possível garantir que transações de baixa prioridade não fiquem esperando por muito tempo.

### **Protocolos Baseados em Timestamps**

Como os protocolos baseados em timestamps garantem a serializabilidade? Explique a diferença entre timestamps mais antigos e mais recentes em uma transação. Testar o funcionamento de transações usando timestamps para escalonamento.

Os protocolos baseados em timestamps garantem a serializabilidade de escalonamentos de transações por meio da atribuição de um timestamp único a cada transação no momento de sua criação. Esse timestamp reflete a ordem em que as transações foram iniciadas, com transações mais antigas sendo priorizadas sobre as mais recentes. Quando duas transações entram em conflito, o protocolo verifica seus timestamps. Se uma transação tenta ler ou escrever dados de uma transação anterior, ela será aceita sem problemas, se duas transações tentam acessar os mesmos dados simultaneamente, a transação com o timestamp mais recente será abortada e reexecutada para garantir que a ordem das transações seja respeitada e que não ocorra violação de integridade. Esse mecanismo assegura que as transações sejam executadas de forma que o resultado final seja equivalente à execução serial, ou seja, uma execução onde as transações são feitas uma após a outra, sem intercalamentos conflitantes. Em outras palavras, garante que o banco de dados mantenha a consistência, sem problemas como leitura ou escrita perdida, sem precisar de bloqueios explícitos.

### **Protocolos Multiversão (MVCC)**

Como o controle de concorrência multiversão (MVCC) melhora o desempenho do banco de dados em relação a técnicas baseadas em bloqueios? Quais são as vantagens e desvantagens desse método? Observar como o banco de dados gerencia múltiplas versões de um mesmo dado.

O **Controle de Concorrência Multiversão (MVCC)** melhora o desempenho do banco de dados permitindo que várias transações acessem dados simultaneamente sem interferir entre si. Ao invés de usar bloqueios, o MVCC mantém múltiplas versões dos dados, garantindo que cada transação acesse a versão correspondente ao seu timestamp de início. Isso resulta em maior concorrência, já que as transações podem ser executadas paralelamente sem precisar esperar que outras liberem bloqueios, aumentando assim a eficiência do sistema. A contenção é reduzida, pois as transações não precisam aguardar o desbloqueio de dados, o que melhora o desempenho, especialmente em sistemas com alta carga de leitura. Além disso, o MVCC assegura que cada transação veja os dados conforme seu timestamp, evitando problemas como leituras inconsistentes ou "leituras fantasmas".

Por outro lado, o MVCC também apresenta algumas desvantagens. A gestão de múltiplas versões de dados pode aumentar a complexidade do sistema e o uso de espaço em disco, já que cada transação precisa ter acesso à versão correta dos dados. Além disso, é necessário realizar um processo de limpeza (garbage collection) para remover versões antigas que não são mais necessárias, o que pode impactar o desempenho do banco de dados, especialmente em sistemas de alta carga.

No MVCC, o banco de dados usa ponteiros para cada versão dos dados, garantindo que cada transação acesse a versão correta sem interferir nas demais. Esse método é particularmente útil em sistemas com alta concorrência, como os bancos de dados usados em Processamento de Transações Online (OLTP).

## **Critérios de Avaliação:** Os alunos serão avaliados com base nos seguintes critérios:

* **Execução correta das transações** – As consultas e transações devem ser bem estruturadas e funcionalmente corretas.
* **Análise dos resultados** – A equipe deve apresentar uma justificativa clara para os resultados obtidos.
* **Registro de respostas** – Cada equipe deve registrar suas respostas no banco de dados corretamente.
* **Trabalho em equipe** – A participação de todos os integrantes será levada em conta.

## **Entrega do Projeto - 03/04/2025**

* Cada equipe deve responder todas as questões no banco de dados e registrar suas tentativas na tabela **“Tentativas”**.
* O professor irá corrigir e validar as respostas diretamente no banco de dados.
* As equipes que concluírem todas as questões corretamente receberão um *feedback* final e a nota correspondente.

**Equipes - Turma 2ª e 4ª**

01 - Geovana Saldanha, Leonardo Tamasauskas

02 - Abraão da Cunha Ribeiro Silva, Adriano Marinho, Willian Felicidade

03 - Leandro Souza, Walfredo Dos Santos, Rodrigo Almeida

04 - Emanoel Felipe dos Santos, Joao Batista Barbosa, Murilo das Chagas

05 - Ponciano Matheus Lima, Rodrigo Teixeira

06 - Fabrício assunção

07 - Daniele Veras Linhares

08 - Andreya Paiva, Luan Farias, Heloísa Silva

09 -

10 -

**Equipes - Turma 3ª e 5ª**

01 - Raphael Pinho, Guilherme Leite e Christian de Jesus;

02 - Andreya Paiva, Luan Farias e Heloisa Silva;

03 - Enzo Santos, Jessé Barros, Ramon Mendes;

04 - Lucas Ferreira, Lucas Moreno, Luis Carlos:

05 - Marcos Thiago S. Horsford,

06 - José Lucas S. Cantão, Luan Souza

07 - Thiago Calado

08 -

09 -

10 -