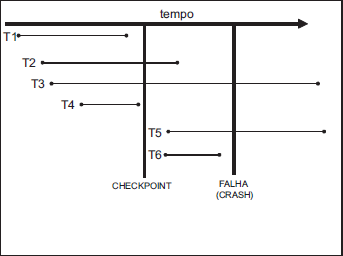
|  |  |
| --- | --- |
| LogoEfeitrans | **Banco de Dados II**  Profa. Vanessa Souza |

**Questão 2:** Considerando as técnicas e os procedimentos de recuperação em caso de falhas, o que acontece com as transações T1, T2, T3, T4, T5 e T6, que estão sendo executadas em um Sistema Gerenciador de Banco de Dados, conforme apresentado na figura, após a falha, se:

1. a técnica de recuperação usada for a de **atualização imediata**

- T1, T2, T4, e T6: Não faz nada: Como essas transações já foram concluídas, suas atualizações foram confirmadas e persistidas no banco de dados. Portanto, não haverá necessidade de desfazer suas operações durante a recuperação imediata.

- T3 e T5: Faz **Undo:** Como essa transação estava em execução durante a falha na recuperação imediata será necessário desfazer as operações realizadas por essas transações com o UNDO, retornando o banco de dados ao seu estado anterior à execução de T3 e T5.

b) Se a técnica de recuperação usada for a de **atualização adiada**:

- T1 e T4 : Não faz nada: Como essas transações já foram concluídas antes do checkpoint, suas atualizações foram confirmadas e persistidas no banco de dados. Durante a recuperação adiada, não será necessário refazer suas operações.

- T5 E T3: Não faz nada: Como essa transação estavam em execução durante a falha, nao foram finalizadas, então não ha como refaze-las durante a recuperação adiada.

-T2 e T6: Faz **Redo**: Como essas transações estavam **parcialmente compromissadas** durante a falha, na recuperação adiada, será necessário REFAZER estas transações com o REDO, poissuas atualizações podem não ter sido confirmadas ou persistidas no banco de dados.

O Undo é usado para desfazer as alterações das transações não confirmadas, enquanto o Redo é usado para refazer as transações parcialmente compromissadas durante a recuperação do banco de dados após uma falha.

**Questão 3:** Julgue e justifique os itens abaixo:

1. A escalabilidade dos bancos de dados NoSQL é garantida pela ausência de um esquema (scheme free). Falso – A escabilidade do NoSQL nao depende do esquema é garantida pela arquitetura própria desses sistemas. Os bancos de dados NoSQL são projetados com uma arquitetura distribuída e escalável desde o início, permitindo que eles lidem com grandes volumes de dados e tráfego intenso.
2. Bancos de dados NOSQL orientados a documentos são apropriados para o armazenamento de dados semiestruturados. Correta. Os bancos de dados NoSQL orientados a documentos são especialmente adequados para armazenar dados semiestruturados. Esses bancos permitem que os documentos sejam armazenados sem a necessidade de seguir um esquema rígido, o que os torna flexíveis o suficiente para acomodar dados sem uma estrutura fixa definida antecipadamente.
3. Uma característica de bancos de dados NoSQL é o suporte à replicação de dados. Entre as abordagens utilizadas para replicação, inclui-se a mestre-escravo. Correta. A replicação de dados é uma característica comum nos bancos de dados NoSQL. A abordagem mestre-escravo é uma das estratégias utilizadas para replicação, na qual um nó mestre recebe todas as operações de escrita e replica esses dados para os nós escravos, que são responsáveis por atender as operações de leitura.
4. Os bancos de dados NoSQL são imunes a ataques de injeção SQL. Falso. Os bancos não são imunes aos ataques de injeção, estão sujeitos a ataques, por isso é preciso aplicar práticas de segurança adequadas ao usar bancos de dados NoSQL,como validar e sanitizar dados de entrada, limitar privilégios de acesso, configurar firewalls, etc.
5. Em um banco de dados NoSQL orientado a documentos, a inexistência de um esquema impossibilita a definição de índices. incorreta. Em um banco de dados NoSQL orientado a documentos, é possível definir índices, mesmo que não exista um esquema rígido. Os índices podem ser criados para otimizar consultas e melhorar o desempenho das operações de busca. A diferença é que, em vez de um esquema pré-definido, os índices são normalmente criados com base nas necessidades específicas das consultas realizadas no banco de dados.

**Questão 4:** Bancos de dados conhecidos como NoSQL podem ser particionados em diferentes servidores, o que introduz o problema de processar consultas que envolvem múltiplos nós de processamento. Um modelo usualmente empregado nessas circunstâncias é conhecido como: a-CAP Theorem; b- Map/Reduce; c-Hash tables; d-Clustered columns; e- Data Thread. resposta: b- O modelo usualmente empregado nessas circunstâncias é o Map/Reduce, que é um modelo de processamento de dados distribuído usado em bancos de dados NoSQL para lidar com consultas que envolvem múltiplos nós de processamento. No modelo Map/Reduce, as consultas são divididas em duas fases principais: a fase de mapeamento (Map) e a fase de redução (Reduce). Na fase de mapeamento, os dados são processados em paralelo em cada nó de processamento, onde cada nó executa uma função de mapeamento para extrair e transformar os dados relevantes.Na fase de redução, os resultados parciais gerados na fase de mapeamento são combinados e consolidados para produzir o resultado final da consulta. Essa etapa envolve a execução de uma função de redução nos resultados parciais para obter o resultado agregado desejado. O modelo Map/Reduce permite que as consultas sejam executadas em paralelo em vários nós de processamento, tornando-o adequado para lidar com a distribuição e o processamento de dados em ambientes particionados de bancos de dados NoSQL.

**Questão 5:** Big Data é:

1. volume + variedade + agilidade + efetividade, tudo agregando + valor + atualidade.
2. volume + oportunidade + segurança + veracidade, tudo agregando + valor.
3. dimensão + variedade + otimização + veracidade, tudo agregando + agilidade.
4. volume + variedade + velocidade + veracidade, tudo agregando + valor. Big Data é um termo que se refere a conjuntos de dados extremamente grandes e complexos, que não podem ser facilmente processados e gerenciados por métodos tradicionais. Esses conjuntos de dados são caracterizados pelos "5 V's" do Big Data: volume, variedade, velocidade, veracidade e valor. **- Volume:** Refere-se à grande quantidade de dados gerados e armazenados. O Big Data lida com a necessidade de processar e analisar conjuntos de dados que são muito grandes para serem tratados com ferramentas e técnicas convencionais. - **Variedade**: Refere-se à diversidade dos tipos e formatos dos dados. No contexto do Big Data, os dados podem ser estruturados, semiestruturados e não estruturados, provenientes de várias fontes, como textos, imagens, vídeos, logs, dados de sensores, entre outros**. - Velocidade:** Refere-se à velocidade com que os dados são gerados e precisam ser processados e analisados. Com o Big Data, muitas vezes é necessário lidar com dados em tempo real ou próximo disso, onde a análise e tomada de decisões devem ocorrer rapidamente. - **Veracidade:** Refere-se à qualidade e confiabilidade dos dados. No contexto do Big Data, é crucial garantir que os dados sejam precisos, confiáveis e livres de erros ou distorções. **Valor:** refere à capacidade de extrair informações úteis e insights significativos dos dados coletados. É o processo de transformar os dados brutos em conhecimento acionável, que pode ser utilizado para tomar decisões informadas, identificar padrões, prever tendências, obter vantagens competitivas, gerar insights valiosos e informar decisões estratégicas.

**Questão 8:** Quais são os níveis de isolamento permitidos no PostgreSQL? Qual o nível *default*? Como alterar o nível de isolamento de uma transação no PostgreSQL? No PostgreSQL, os níveis de isolamento permitidos são 3: 1-**Read Committed:** Neste nível, uma transação só pode ler dados confirmados de outras transações. Isso elimina as leituras sujas, mas ainda pode resultar em leituras repetíveis e leituras fantasmas.  **[**O Read Uncommitted não é implementado pelo postgres, este modo comporta-se como Read Committed no postegres, mas este nível, em teoria, permite que uma transação possa ler dados não confirmados de outras transações, o que pode resultar em leituras sujas (dirty reads), leituras repetíveis (non-repeatable reads) e leituras fantasmas (phantom reads)**]**. 2-**Repeatable Read:** Neste nível, uma transação garante que todas as leituras repetidas retornem o mesmo conjunto de dados, mesmo que outras transações tenham realizado modificações.O postgres não permite leitura fantasma neste nível. 3-**Serializable:** Neste nível, as transações são isoladas umas das outras e os efeitos das transações concorrentes são invisíveis. Isso garante a consistência e evita leituras sujas, leituras repetíveis e leituras fantasmas. **O nível de isolamento padrão no PostgreSQL é o "Read Committed".** No entanto, o nível de isolamento pode ser alterado em uma transação usando o seguinte comando: BEGIN (ou START) TRANSACTION ISOLATION LEVEL SERIALIZABLE (ou repeatable Read) ; mas é importante observar que a alteração do nível de isolamento afeta apenas a transação atual e não afeta outras transações concorrentes. Cada transação pode definir seu próprio nível de isolamento de acordo com suas necessidades específicas.

**Questão 9:** Qual o log gerado para a transação T2, se A = B = 1000 ?

|  |  |
| --- | --- |
| T2: read(A);  A := A – 100;  Read(B);  B := B + 100;  **Write(B);** | **<T2,start>**  **<T2, B, 1000, 1100>**  **<T2, commit>** |

**Questão 10:** Descreva as transações que originaram o log abaixo.

|  |  |  |  |
| --- | --- | --- | --- |
| <T0, start>  <T0, A, 100,950>  <T1, start>  <T0,B, 2000,2050>  <T1, C, 700, 600>  <T1, commit>  <T2, start>  <T0, commit>  <T2, k, ‘casa’, NULL>  <T2, commit> | T0: READ(A);  T0: A:= A + 850;  T0: Write(A);  T0: READ(B);  T0: B:= B + 50;  T0: Write(B); | T1: READ(C);  T1: C:= C - 100;  T1: Write(C); | T2: READ(k);  T2: k:= NULL;  T2: Write(K); |

**Questão 11:** Dada a sequencia do registro de log:

<T, start>

<T, A, 10, 11>

**<U, start>**

<U, B, 20, 21>

<T, C, 30, 31>

<U, D, 40, 41>

**<U, commit>**

<T, E, 50, 51>

**<T, commit>**

Considerando a recuperação adiada e a recuperação imediata, o que acontece se a última instrução gravada no arquivo de log for:

a) <U, start>:

- Recuperação adiada: Não faz nada com T e U, pois foram abortadas por estarem incompletas.

- Recuperação imediata: Desfaz com **undo** as transações T e com U nao faz nada, pois não ha operações a serem desfeitas em U.

b) <U, commit>:

- Recuperação adiada: Refaz com **rendo** a transação U e com T nao faz nada,pois T foi abortada por estar incompleta.

- Recuperação imediata: Desfaz com **undo** a transação T e com U nao faz nada, pois U ja foi salva no banco.

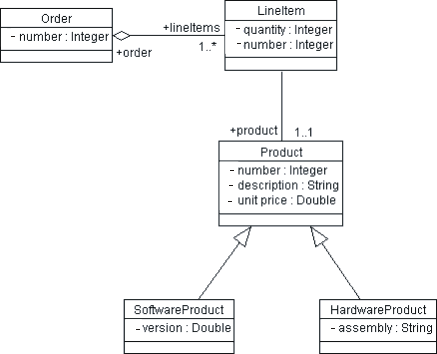
c) <T, E, 50, 51>: mesma resposta de b)

d) <T, commit>:

- Recuperação adiada: Refaz com **rendo** as transações T e U para comita-las completamente no banco.

- Recuperação imediata: Não faz nada. T e U já foram salvas no banco.

**Questão 1:** Dado o modelo relacional abaixo, modelá-lo no contexto de BDOR e implementar seu modelo físico.T



Colocaria software e hardware como um novo atributo denominado tipo \_pro dentro de product e os seus respetivos atributos passam a ser de product.

* **Em product:** tipo\_pro varchar (15); CHECK ( tipo\_pro = 'software' or tipo\_pro = 'hardware'); Version real; Assembly varchar (100); ...

Depois faria product ser um tipo de LineItem. create type tipo\_product as (“atributo de produto”);

* **Em line item:** number: integer; produto tipo \_product;quantity: integer;

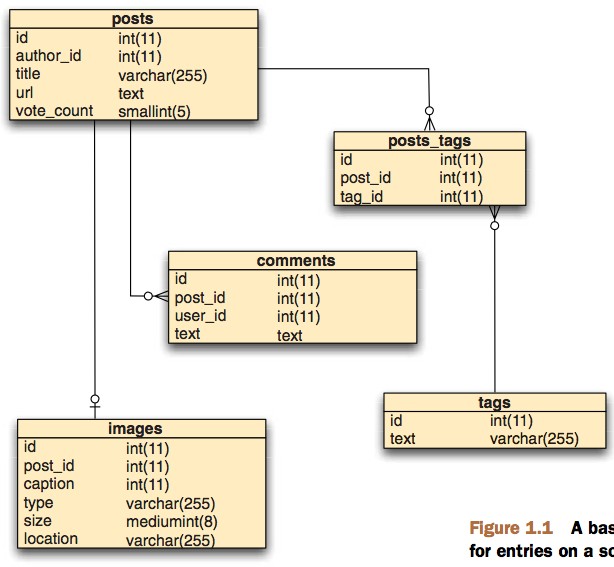
Por fim faria LineItem ser um tipo array de order. create type tipo\_lineItem as (“atributo de line\_item”);

* Em order: number integer; item tipoItem[100];

Ficando apenas no fim a tabela order que possui um atributo item que é do tipo array de lineItem, que por sua vez possui como variavel produto que é do tipo product

**Questão 6:** Dado o modelo relacional abaixo, crie um modelo de agregado para utilizar no MongoDB. Justifique sua modelagem.

Pensando que posts seria a entidade mais consultada do banco , utilizaria o posts como agregado, assim iria incorporar todos as demais dentidade dentro dele. Onde images seria um objeto, comments um array, tags um array de post\_tags e, por fim post\_tags seria um array incorporado a posts.



**Questão 7:** Dado o modelo agregado gerado na questão 9, faça:

* 1. Um documento JSON para inserir no MongoDB

Exemplo da estrutura de json: rsrs

{

"name": "OpenAI Cafe",

"location": "123 Main Street",

"city": "San Francisco",

"country": "United States",

"rating": 4.5,

"reviews": [

{

"user": "John",

"comment": "Great food and atmosphere!"

},

{

"user": "Lisa",

"comment": "The service was excellent."

}

]

}

* 1. Código para adicionar o campo ‘data’ para um post

Para um post especifico de id =1

db.posts.updateOne(

{ **\_id:** 1) }, // Filtro para identificar o post específico

{ **$set:** { data: new Date() } } // Define o campo 'data' com o valor atual da data

)

Para o primeiro post do documento:

db.posts.findOne(

{ },

{ **$set:** { data: new Date() } } // Define o campo 'data' com o valor atual da data

)

* 1. Código para retornar os comentários de **um post**

Para um post especifico de id =1

db.posts.findOne(

{ \_id: 1 }, // Filtro para encontrar o post específico

{ comments: 1 } // Projeção para incluir todos os 'comentarios'

);

Para o primeiro post do documento:

db.posts.findOne(

{ },

{ comments: 1 } // Projeção para incluir todos os 'comentarios'

);