**Transação** - Unidade lógica de trabalho, constituída de um conjunto de operações, cujo objetivo é transformar um BD de um estado consistente para outro estado consistente, mesmo que nos passos intermediários o sistema permaneça temporariamente inconsistente. Pode conter diversas operações (inclusão, exclusão, atualização e consulta).

**ACID**

* **Atomicidade -** Uma transação é uma unidade atômica de trabalho;  ela deve ser realizada em sua totalidade ou não der realidade de forma alguma.
* **Consistência -** Uma transação deve preservar a consistência, significando que, se ela for completamente executada do início ao fim sem interferência de outras transações, deve levar o banco de dados de um estado consistente para outro.
* **Isolamento -** Uma transação deve parecer como se fosse executada isoladamente de outras transações, embora muitas delas estejam sendo executadas de maneira simultânea. Ou seja, a execução de uma transação não deve ser interferida por quaisquer outras transações que acontecem simultaneamente. Mecanismo de controle de concorrência garantem às transações a característica de isolamento. O isolamento trata de um conjunto de técnicas que tentam evitar que transações paralelas interfiram umas nas outras, fazendo com que o resultado de várias transações em paralelo seja o mesmo resultado que essas transações teriam caso fossem executadas sequencialmente (uma após a outra).
* **Durabilidade -** As mudanças aplicadas ao banco de dados pela transação confirmada precisam persistir no banco de dados. Essas mudanças não devem ser perdidas por causa de alguma falha.

Além disto, segundo ainda ELMASRI e NAVATHE, a propriedade de atomicidade exige que executemos uma transação até o fim. É de responsabilidade do subsistema de recuperação de transação de um SGBD garantir a atomicidade. Complementarmente, a propriedade de durabilidade é a responsabilidade do subsistema de recuperação do SGBD. Assim, podemos concluir que as propriedades de recuperação do SGBD é ATOMICIDADE e DURABILIDADE.

* ATOMICIDADE >> Subsistema de recuperação.
* ISOLAMENTO >> Subsistema de controle de concorrência.
* DURABILIDADE >> Subsistema de recuperação.
* CONSISTÊNCIA >> Programador ou módulo de restrições de integridade.

**Redundância de Dados**

Há **dois tipos**de redundância de dados, a **redundância *controlada*** de dados e a**redundância *não controlada***de dados.

* **Redundância controlada de dados -** Acontece quando o software tem conhecimento da múltipla representação da informação e garante a sincronização entre as diversas representações. Exemplo: Sistema distribuído, onde uma mesma informação é armazenada em vários computadores, permitindo acesso rápido a partir de qualquer um deles. (utilizada para melhorar a confiabilidade ou o desempenho global do sistema).
* **Redundância não controlada -** Acontece quando a responsabilidade pela manutenção da sincronia entre as diversas representações de uma informação está com o usuário e não com o software.

**Failover** é a capacidade de determinado sistema/serviço migrar automaticamente para um outro servidor, sistema ou rede redundante ou que está em standby quando da ocorrência de falha ou término anormal do servidor, do sistema ou da rede que estava ativo até aquele instante. O Failover acontece sem intervenção humana e geralmente sem aviso prévio, diferente de **switch over**.  
Os projetistas de sistemas normalmente fornecem capacidade de failover em servidores, sistemas ou redes que exigem continua disponibilidade e um alto grau de confiabilidade.

**A replicação de dados** é muito utilizada nos bancos de dados relacionais, principalmente para garantir alta disponibilidade em sistemas críticos e, em segundo lugar, para prover escalabilidade horizontal (onde a carga é distribuída entre outros servidores). A replicação trata do envio dos dados alterados de um servidor para outro e os principais SGBDs (Sistemas Gerenciadores de Bancos de Dados) do mercado possuem esta funcionalidade.

O **espelhamento** do banco de dados pode ser usado junto com a replicação para aprimorar a disponibilidade ao banco de dados de publicação. O espelhamento do banco de dados compreende duas cópias de um único banco de dados que geralmente reside em computadores diferentes. Em determinado momento, apenas uma cópia do banco de dados está atualmente disponível aos clientes. Essa cópia é conhecida como o banco de dados principal. As atualizações realizadas pelos clientes no banco de dados principal são **aplicadas** à outra cópia do banco de dados, conhecida como banco de dados **espelho**. O espelhamento envolve a aplicação do log de transações de cada inserção, atualização ou exclusão efetuada no banco de dados principal, para o banco de dados espelho.

O failover de replicação para um espelho tem o suporte total para os bancos de dados de publicação, com suporte limitado para bancos de dados de assinatura. O espelhamento de banco de dados não tem suporte para bancos de dados de distribuição. Para obter informações sobre como recuperar um banco de dados de distribuição ou banco de dados de assinatura sem precisar reconfigurar a replicação, veja Fazer backup e restaurar bancos de dados replicados.

A replicação MySQL funciona sincronizando os dados entre dois ou mais servidores, em tempo real (ou mais ou menos isso). Em outras palavras, se um usuário criar ou atualizar algum dado em um banco de dados, o outro servidor puxa esse comando e executa nele também. É importante notar que o comando só é feito quando se cria ou atualiza algum dado, pois sincronizar uma leitura não faria o menor sentido.

Em se tratando de segurança em banco de dados, atualmente existem duas abordagens conhecidas como controle **discricionário** e **mandatário**.

* No **controle discricionário**, um usuário terá direitos de acesso (privilégios) diferentes sobre dados diferentes, incluindo a capacidade de acessar arquivos específicos de dados, campos específicos de uma tabela ou mesmo poder utilizar determinadas operações sobre a base de dados (ler, inserir, excluir, alterar, atualizar). O **controle discricionário** é muito flexível, pois, por exemplo, um usuário U1 pode acessar a tabela T1, mas não pode ter acesso a T2, já o usuário U2 pode acessar as tabelas T1 e T2. Segundo Navathe(2011,p.572),"A principal desvantagem dos modelos DAC é sua vulnerabilidade a ataques maliciosos, como cavalos de Troia embutidos nos programas de aplicação."
* No **controle mandatário**, os usuários e os dados são classificados em vários níveis de segurança, de acordo com a política de segurança da empresa. Um determinado usuário somente terá acesso aos dados que estiverem no mesmo nível de segurança ao qual lhe foi concebido. Por exemplo, imaginamos a seguinte situação: se um determinado usuário U1 possui nível de segurança N1 e o usuário U2 possui nível N2, sendo que os dados D1 e D3 possui nível N1 e os dados D2 e D4 possui nível N2. Com base no exemplo acima o usuário U1 terá acesso aos dados D1 e D3, enquanto o usuário U2 terá acesso somente aos dados D2 e D4.

***Concorrência***

* **Timestamp** - Protocolo de controle de concorrência que utiliza ordenação por rótulo de tempo para garantir a serialização das transações.

**4 problemas de consistência** (por ordem crescente de dificuldade para seu tratamento pelos níveis de isolamento nos SGBD):

* **Perda de atualização**: duas transações que ocorrem simultaneamente atualizam o mesmo dado. Isto pode ocorrer em uma sequência segundo a qual uma das atualizações é perdida.
* **Leitura Suja**: leitura de dados não confirmados de uma linha existente, podendo ocasionar a leitura de uma informação nunca confirmada.
* **Leitura Não-Repetida**: duas leituras de dados na mesma transação não se repetem. Na segunda leitura, dados não existem ou foram modificados.
* **Leitura Fantasma**: na leitura de um conjunto de dados (função de agregação), surgem novas informações no conjunto que podem gerar um conflito. Apesar das transações não acessarem tuplas em comum, diz-se que as transações entraram em conflito em uma tupla fantasma.

Só há como se garantir um isolamento total (propriedade de serializabilidade) de transações por meio do mecanismo de bloqueio em duas fases, que tem o seguinte mecanismo de funcionamento:

"Diz-se que uma transação segue o protocolo de bloqueio em duas fases se todas as operações de bloqueio (read\_lock ou write\_lock) precedem a primeira operação de desbloqueio na transação. Essa transação pode ser dividida em duas fases: uma fase de expansão ou crescimento, durante a qual podem-se obter novos bloqueios em itens mas nenhum pode ser liberado; e uma fase de encolhimento ou retração (segunda fase), durante a qual bloqueios existentes podem ser liberados, mas nenhum bloqueio pode ser obtido."

**Técnica de bloqueio em duas fases:** para controle de concorrência é baseado no bloqueio de itens de dados, sendo que, chamamos de bloqueio uma variável que fica atrelada ao item de dados. Este bloqueio pode ser binário (possui dois valores: 1 e 0), logo, o item de dados está bloqueado ou não está bloqueado. Permitindo que o item de dado só esteja acessível para uma transação apenas se a variável não estiver bloqueada (ou estiver com valor 0). São usadas duas operações para o bloqueio binário, são elas: lock(1) e unlock(0), quando o item de dados está sendo usado, o estado da variável é lock(1), assim que a transação encerra a utilização do item é emitida a operação unlock(0), então, o item já está disponível para outra transação. Duas maneiras de bloquear(lock) os dados são:

* **Bloqueio Compartilhado**: quando uma transação recebe este tipo de bloqueio e a instrução é de leitura, então, mais de uma transação poderá acessar o mesmo dado. Se a instrução for de gravação, então ela não poderá participar de um bloqueio compartilhado, ou seja, é permitido que várias transações acessem um mesmo item "A" se todas elas acessarem este item "A" apenas para fins de leitura.
* **Bloqueio Exclusivo**: quando uma transação recebe este tipo de bloqueio, ela fica exclusivamente reservada para a instrução que compõe a transação, não permitindo que outra transação faça uso do dado que está sendo utilizado, logo, um item bloqueado para gravação é chamado de bloqueado exclusivo, pois uma única transação mantém o bloqueio no item.

Outra definição

**Bloqueio Compartilhado ou Bloqueio para leitura:**A transação pode apenas ler o dado. Mais de uma transação podem ter esse bloqueio sobre o mesmo dado ao mesmo tempo.  
  
**Bloqueio Exclusivo ou Bloqueio de escrita**: A transação pode ler e escrever sobre o dado. Apenas uma transação pode ter este bloqueio em um período de tempo.  
  
O nome da técnica de bloqueio pode ser chamado de **bloqueio Compartilhado/Exclusivo**ou **Bloqueio Leitura/gravação**. Em 2 fases (2PL) é dividida em: Fase de expansão ou crescimento (primeira), durante a qual novos bloqueios em itens podem ser adquiridos, mas nenhum pode ser liberado;  
Fase de encolhimento, durante a qual os bloqueios existentes podem ser liberados, mas nenhum novo bloqueio pode ser adquirido.

Com relação ao bloqueio em duas fases...

"Diz-se que uma transação segue o protocolo de **bloqueio em duas fases** se todas as operações de bloqueio (read\_lock, write\_lock) precedem a primeira operação de desbloqueio na transação. Essa transação pode ser dividida em duas fases: uma**fase de expansão ou crescimento**, durante a qual podem se obter novos bloqueios em itens mas nenhum pode ser liberado; e uma **fase de encolhimento ou retrocesso** (segunda fase), durante a qual bloqueios existentes podem ser liberados, mas nenhum bloqueio pode ser obtido."

Uma transação precisa manter o bloqueio do item de dado durante o tempo em que estiver acessando aquele item, até mesmo porque nem sempre o desbloqueio imediato após o acesso final é interessante, pois pode comprometer a serialização em alguns casos.

Sempre haverá a necessidade de bloqueio e desbloqueio dos itens de dados, mas existem algumas situações em que a combinação dessas duas fases pode gerar um problema no banco dados. Caso não seja feito o desbloqueio do item de dado antes da solicitação de um bloqueio a outro item de dado, pode ocorrer um deadlock (ou impasse).

Consideremos que existem duas transações A e B, a transação A está esperando por algum item que está bloqueado na transação B, e a transação B está esperando por algum item que está bloqueado em A, então, ambas ficam na fila de espera, aguardando que seja liberado o bloqueio de um item, como isso não ocorre as duas transações acabam nunca conseguindo ser concluídas.

Outro problema que pode ocorrer é o starverd (ou starvation). Consideremos que existem três transações A, B e C, a transação A faz a requisição de um bloqueio compartilhado de um item de dado, logo em seguida a transação B faz uma requisição de bloqueio exclusivo do mesmo item de dado, como a transação A está usando o bloqueio compartilhado do item, logo a transação B não poderá deter o acesso, o que faz com que ela fique na fila de espera, até a liberação do mesmo. Enquanto a transação B está na fila, chega a transação C com o pedido de compartilhamento do mesmo item, como é possível fazer o compartilhamento do item que A está usando, então, a transação C passa na frente de B e consegue o compartilhamento do item de dado da transação A. Quando a transação A terminar de desocupar, o item de dado continuará ocupado pela transação C, enquanto isso, a transação B continuará aguardando a liberação total do item de dado para que ela possa fazer o bloqueio exclusivo. Caso as próximas transações sejam sempre de acesso compartilhado deste mesmo item, a transação B nunca conseguirá fazer um progresso, e então, ela é considerada **estagnada (sinônimo de starvation)**.

**Protocolo de bloqueio de duas fases (2PL):**

* **Fase de crescimento**: Transação pode obter bloqueios, mas não pode liberar.
* **Fase de encolhimento**: Transação pode liberar bloqueios, mas não pode obter.

Bloqueios de duas fases previnem que transações concorrentes interfiram em outras. Isso para **garantir a seriabilidade**. Entretanto,**isso não garante que não haja deadlock.**

O controle de concorrência para transações de banco de dados que garante que resultados intermediários de uma transação T não afetem outras transações até que T tenha sido totalmente executada, ou rejeitada, é usualmente implementado em bancos de dados por meio de **bloqueio (lock e unlock).**

**Quando diversas transações são executadas de modo concorrente em um banco de dados, é necessário que o sistema controle a interação entre transações concorrentes; esse controle é alcançado por meio de uma larga gama de mecanismos chamados *esquemas de controle de concorrência*. Um meio de garantir a serialização é obrigar que o acesso aos itens de dados seja feito de maneira mutuamente exclusiva; isto é, enquanto uma transação acessa um item de dados, nenhuma outra transação pode modificá-lo. O método mais usado para sua implementação é permitir o acesso a um item de dados somente se ele estiver bloqueado. Infelizmente, o uso de bloqueios pode causar situações indesejáveis, como por exemplo o aparecimento de *deadlocks.***

A questão exige do candidato a compreensão do funcionamento do protocolo 2 Phase Lock no modo Básico. Vamos aos comentários:

O 2PL (2 phase lock) é um protocolo baseado em bloqueio para evitar inconsistências na transação. O funcionamento do protocolo no modo Básico (padrão) pressupõe que itens da mesma transação podem estar no estado LOCK e UNLOCK. É nessa caracteristica de funcionamento que podemos observar semanticamente o comportamento descrito na assertiva, ou seja, um item da transacao inicial que ja foi liberado (UNLOCK) pode ser requisitado por uma nova transacao. Caso a transacao inicial(que ainda nao tinha sido finalizada em todos os seus itens) precise ser desfeita, a transacao subsequente (que estava utilizando o item liberado anteriormente) tambem será impactada pelo desfazimento.

**Níveis de isolamento**

**Isolamento:** Uma transação em andamento, mas ainda não validada, deve permanecer isolada de qualquer outra operação, ou seja, garantimos que a transação não será interferida por nenhuma outra transação concorrente.

* **Leitura Fantasma:** Uma transação “A” pode ler um conjunto de linhas de uma tabela com base em alguma condição Where SQL. Suponhamos que a transação “B” insira uma nova linha que também satisfaz a clausula Where na tabela utilizada pela transação A. Se a transação “A” for repetida ela verá um fantasma, ou seja, uma linha que não existia na primeira leitura utilizando a clausula where.
* **Repeatable Read:** Este nível garante que a mesma leitura de um dado através do SELECT se repita, tendo o mesmo resultado para diferentes execuções na mesma transação. Se neste nível a leitura não fosse repetida, ela estaria aberta a leitura fantasma, que acontece entre um select e outro caso ocorra uma atualização nos dados neste espaço de tempo.
* **Read Commited:** Permite que a transação leia manipule os dados já commitados por outras transações. Caso alguma transação tenha alterado algum dado porém sem efetuar um commit, estes não serão vistos. Está aberto para leitura fantasma.
* **Read Uncommited:** Este nível permite que uma transação possa ver e manipular valores não commitados por outras transações. Também podem ocorrer casos de Leituras fantasmas.
* **Serializable:** Semelhante ao Repeatable Read, porém com a restrição adicional de que as linhas selecionadas por uma transação não podem ser alteradas ou lidas por outra transação, até que a primeira transação seja concluída. Logo este nível isola completamente uma transação da outra, onde a segunda transação aguarda a finalização da primeira, e assim por diante “uma de cada vez”, evitando a leitura fantasma.