Transações

Última atualização a *16 de maio de 2020*

Um SGBD é um intermediário entre a aplicação e a BD, sobre a qual realiza operações, solicitadas por um ou vários utilizadores. Às unidades lógicas de trabalho compostas pelo conjunto de uma ou mais operações, chamamos **transações**.

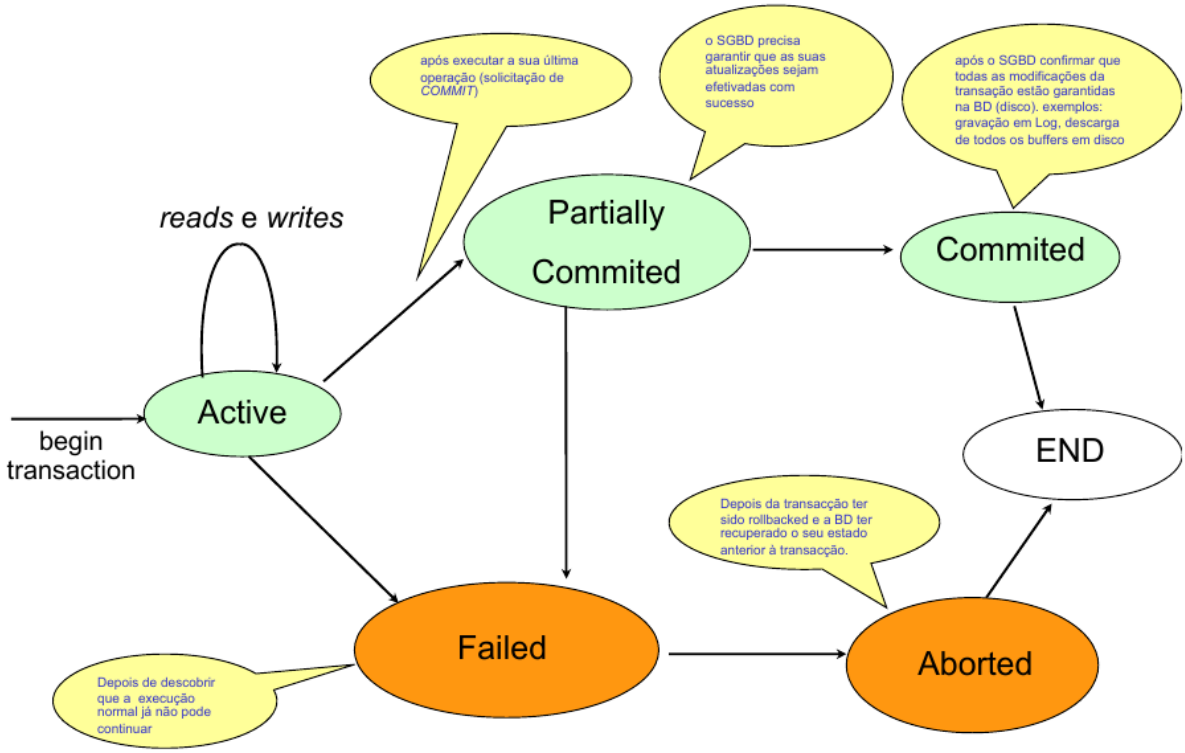
Podemos considerar que estas operações podem assumir dois tipos:

- **Leitura** transfere um elemento da BD para a área de memória volátil associada à transação que executou a operação de leitura
- **Escrita** transfere um elemento da área de memória associada à transação para a BD

> Para transferir 50€ entre duas contas, há que **ler** o saldo da primeira, subtrair 50 e **escrever** o novo valor e o mesmo na segunda, mas desta vez somando.

Estados de uma transação

Ao longo da sua execução, uma transação pode assumir vários estados, entre os quais **Active, Partially Commited, Commited, Failed, Aborted e Concluded**.



Propriedades da uma transação (ACID)

Atomicidade as operações ocorrem de forma indivisível (atómica), ocorrendo todas ou nenhuma

> Não faz sentido separar as operações de remoção do dinheiro de uma conta e depósito noutra durante uma transferência, pois se uma falhar, devem falhar as duas

Consistência a integridade da BD é sempre mantida

> Durante pode ser violada, mas no final da execução deve voltar

Isolamento as transações concorrentes não interferem entre si (recorrendo ao escalonamento)

> Enquanto uma transação manipula um valor de uma BD, este não deve ser manipulado em simultâneo por outra, devendo ser escalonada para executar depois da primeira ter terminado

Durabilidade os efeitos de uma transação *commited* na BD são permanentes e visíveis para as restantes

> Nenhuma falha posterior deve afetar as modificações, podendo-se recorrer a Logs, p.e.

Controlo de concorrências

Para garantir o isolamento das transações e de forma a garantir a integridade da BD é necessária a implementação de **escalonamentos**, que podem ser de dois tipos:

Escalonamento serializado caso seja executada uma instrução de cada vez e de forma sequencial

> Esta solução é pouco eficiente, pois os recursos não são aproveitados em pleno e os tempos de espera podem ser elevados

Escalonamento concorrente serializado quando é permitida a execução concorrente de transações, mas de forma a preservar o isolamento

Cenários de erro

No entanto, este isolamento não é fácil de manter, havendo dois **cenários de erro** possíveis, que ocorrem quando operações pertencentes a transações diferentes acedem ao mesmo elemento sobre o qual é feita pelo menos uma operação de escrita.

> **Atualização perdida** + ocorre quando uma entidade atualiza um valor que é de seguida atualizada pela segunda, que nesta atualização não teve em conta o valor entretanto atualizado pela primeira, mas o valor antes da atualização que esta fez
>
> **Leitura suja** + ocorre quando uma transação lê um valor atualizado por outra transação que ainda não foi *commited*, o que viola a integridade da BD pois como ainda não foi *commited*, a segunda transação pode ser cancelada e assim o valor lido pela primeira não vai ser o correto

Escalonamento

Por estes motivos temos de recorrer a **escalonadores**, que definem a ordem de execução (intercalada) das operações de várias transações, através de um de vários mecanismos, que podem ser **preventivos**, evitando a execução de transações que possam gerar conflito, ou **otimistas**, permitindo a execução de todas as transações e no caso de erros, fazer *rollback* e

reiniciá-las.

Dentro dos preventivos, há dois métodos que se destacam:

Mecanismos de locking Como que um semáforo que garante exclusão mútua a todos os processos ou acesso exclusivo aos escritores, que é libertado no final de cada transação (seja esta *commited* ou *rolled back*).

> Neste mecanismo é importante implementar regras que evitem o **deadlock**

Mecanismos de etiquetagem Consiste em atribuir uma etiqueta incremental a cada transação, com a qual marca todos os elementos a que acede. Se uma transação tentar aceder a um elemento com uma etiqueta maior que a sua, é cancelada e reiniciada.