**Locks**

É um mecanismo de sincronização de processos/threads (transações podem implementar processos/threads), em que processos/threads (transações, por exemplo) devem ser programados de modo que seus efeitos sobre os dados compartilhados sejam **equivalentes serialmente**.

Se for sabido que cada uma de várias threads (transações) tem o mesmo efeito correto quando executada sozinha, então podemos inferir que, se essas threads (transações) forem executadas uma por vez, em alguma ordem, o efeito combinado também será correto.

Uma intercalação das operações das threads (transações) em que o efeito combinado é igual ao que seria se as threads (transações) tivessem sido executadas uma por vez, em alguma ordem, é uma intercalação **equivalente serialmente**.

Quando dizemos que duas threads (transações) distintas tem o mesmo efeito, queremos dizer que as operações de leitura sobre variáveis de instâncias de variáveis (por exemplo, contas bancárias) retornam os mesmos valores e que essas variáveis de instância de objetos tem os mesmos valores finais.

O uso de equivalência serial como critério para execução concorrente correta evita a ocorrência de atualizações perdidas ou recuperações inconsistentes.

**Um exemplo de mecanismo para disposição em série é o caso de locks (travas) exclusivos. Nesse esquema, um servidor tenta impedir o acesso (lock = travar) a qualquer variável que esteja para ser usada por qualquer operação de uma thread (ou transação) de um cliente do servidor. Se um cliente solicitar acesso a uma variável que já está bloqueado (travado) devido a uma thread de outro cliente, o pedido será suspenso e o cliente deverá esperar até que a variável esteja destravada (unlock = destravar).**

A implementação de locks pode ser feita para que o funcionamento de locks (travas) e operação de bloqueio sejam implementadas em separado num servidor, que chamamos de gerenciador de locks (travas).

**O gerenciador de (locks) travas mantém um conjunto de travas** (por exemplo numa tabela hashing). **Cada trava é uma instância da classe** Lock **e é associada a uma variável compartilhada (uma conta bancária) em particular.**

Os métodos de Lock são sincronizados para que as threads que estão tentando adquirir (método aquire, que usa wait) ou liberar (método release, que usa notify) uma trava, não interfiram umas com as outras.

Tentativas de adquirir a trava usam o método wait, quando precisam esperar que outra thread a libere. Todos os pedidos para obter travas (locks) e liberá-las em nomes de threads (transações) são enviadas para uma instância do gerenciador de travas (*class LockManager*).