**Manipulación de Datos usando Transacciones y el Manejo de Deadlocks en Bases de Datos**

En los sistemas de gestión de bases de datos (DBMS), la integridad y consistencia de la información es prioritaria. Para lograrlo, se implementan **transacciones**, que son secuencias de operaciones tratadas como una única unidad lógica de trabajo. Una transacción, para ser válida, debe cumplir con las propiedades **ACID**: **Atomicidad, Coherencia, Aislamiento y Durabilidad**.

**La atomicidad** garantiza que todas las operaciones dentro de una transacción se ejecuten por completo o no se ejecute ninguna, evitando estados intermedios inconsistentes.

**La coherencia** asegura que los datos pasen de un estado válido a otro conforme a las reglas definidas, manteniendo así la integridad referencial y las restricciones.

**El aislamiento** evita que los cambios de una transacción interfieran con los de otra en curso, logrando una ejecución ordenada y predecible.

**La durabilidad** implica que una vez confirmados los cambios de una transacción, estos serán permanentes incluso ante fallos del sistema, gracias a mecanismos de registro como bitácoras o archivos de log.

En la práctica, los DBMS como MariaDB, SQL Server u Oracle ofrecen tres modos de transacciones: **automáticas**, **explícitas** e **implícitas**. Las transacciones automáticas se ejecutan por defecto con cada instrucción SQL (INSERT, UPDATE, DELETE). Las transacciones explícitas son controladas directamente por el usuario mediante BEGIN TRANSACTION, COMMIT y ROLLBACK, lo cual permite mayor control sobre los cambios realizados. Por otro lado, las transacciones implícitas comienzan automáticamente después de finalizar una anterior, pero requieren confirmación o reversión explícita.

Sin embargo, en ambientes donde múltiples usuarios acceden simultáneamente a los datos, puede surgir una situación crítica conocida como ***deadlock*** o **interbloqueo**. Este ocurre cuando dos o más transacciones se bloquean mutuamente al esperar recursos que la otra transacción no libera. Por ejemplo, si la Transacción A bloquea la fila 1 y quiere acceder a la fila 2, mientras que la Transacción B ya tiene bloqueada la fila 2 y desea acceder a la fila 1, ambas quedan atrapadas esperando indefinidamente. Este ciclo de espera sin resolución compromete el rendimiento y la estabilidad del sistema.

Los DBMS manejan los deadlocks de forma automática a través de detectores de interbloqueo. Una vez identificado un ciclo de espera, el sistema elige una de las transacciones como víctima y la aborta, liberando así los recursos para que las demás puedan continuar. A pesar de estos mecanismos, es responsabilidad del desarrollador escribir transacciones lo más cortas y eficientes posible, siguiendo buenas prácticas como acceder siempre a los recursos en el mismo orden, y evitando bloqueos prolongados.

En conclusión, el uso correcto de transacciones es fundamental para mantener la coherencia y confiabilidad de los datos. A su vez, comprender y prevenir los **deadlocks** forma parte del diseño eficiente de sistemas multiusuario, donde el acceso concurrente es inevitable. El equilibrio entre control, seguridad y rendimiento en el manejo de transacciones es una habilidad esencial en el desarrollo de aplicaciones robustas que operan sobre bases de datos.