PROPIEDADES ACID EN SISTEMAS
DE GESTIÓN DE BASES DE DATOS

Garantía de integridad, consistencia y persistencia de datos corporativos.

ASIGNATURA: Bases de Datos I

FECHA: Mayo de 2026

INSTITUCIÓN: Facultad de Ingeniería

INTREGRANTES: Barreto Facundo, Dylan Pagliaccetti, Agustin Carricart, Ramiro Martino,Tomas Delio

1. Introducción al Concepto de Transacción

En el ámbito de las ciencias de la computación, y más específicamente en el diseño e implementación de Sistemas de Gestión de Bases de Datos (DBMS), una transacción constituye la unidad lógica fundamental de trabajo. No se trata simplemente de una única instrucción aislada (como un comando de inserción o actualización), sino de un conjunto estructurado de operaciones que se ejecutan de manera secuencial para cumplir un objetivo de negocio unificado.

El propósito primordial de un sistema de transacciones es mantener la integridad de la información almacenada frente a un entorno intrínsecamente hostil y propenso a fallas. Los entornos informáticos modernos operan en redes concurrentes donde múltiples usuarios e hilos de ejecución modifican recursos compartidos en tiempo real. Adicionalmente, el hardware subyacente y las conexiones de red están expuestos a cortes energéticos, fallas de memoria, sobrecargas de procesamiento y desconexiones físicas abruptas.

Ante este escenario, surge la necesidad absoluta de un paradigma de diseño que asegure que el estado de la base de datos sea siempre predecible, correcto y confiable. Si un proceso de actualización múltiple se interrumpe de forma inesperada a mitad de su ejecución, la base de datos corre el riesgo severo de quedar en un estado corrupto, inconsistente o asimétrico. Para evitar esta degradación del sistema, la comunidad científica e informática formalizó el estándar clásico de propiedades denominado ACID

2. La 'A' de Atomicidad (Atomicity)

La propiedad de Atomicidad toma su nombre del concepto clásico de la física sobre el átomo como la partícula más pequeña, indivisible e indestructible de la materia. Trasladado al universo de los datos, la atomicidad dicta que una transacción debe ser tratada como una unidad operativa elemental y monolítica. Esto implica una regla binaria inquebrantable: o se ejecutan todas y cada una de las operaciones contenidas en la transacción con éxito absoluto, o el sistema simula que no se ha ejecutado ninguna.

En el desarrollo de software y administración de datos, no existen los estados parciales ni las transacciones completadas 'a medias'. Si una transacción consta de diez pasos secuenciales y el sistema falla catastróficamente en el noveno paso debido a un corte de energía, una violación de acceso o un desbordamiento de memoria, el motor de la base de datos tiene la obligación estricta de revertir (operación conocida como Rollback) los ocho pasos anteriores. La base de datos debe regresar con precisión matemática al estado exacto en el que se encontraba justo antes de iniciar la transacción.

2.1 Mecanismos Técnicos: El Log de Transacciones y Write-Ahead Logging (WAL)

Para lograr que este comportamiento 'todo o nada' sea posible sin penalizar masivamente el rendimiento del hardware, las bases de datos modernas implementan un mecanismo interno denominado Registro de Transacciones o Log de Cambios. El principio fundamental es el Write-Ahead Logging (WAL), el cual establece que cualquier modificación en los datos debe registrarse primero de forma secuencial en este archivo de bitácora seguro antes de que los cambios reales se apliquen a las tablas en el disco principal.

Si ocurre un fallo a mitad de camino, el sistema lee este log durante el proceso de recuperación y ejecuta un 'Undo' (deshacer), borrando los rastros de la transacción inacabada. Por el contrario, si la transacción alcanza su instrucción final con éxito, se ejecuta un 'Commit' (confirmación), consolidando permanentemente las operaciones. Gracias a la atomicidad, los desarrolladores de aplicaciones pueden programar flujos complejos con la tranquilidad de saber que el sistema purgará automáticamente los errores intermedios.

3. La 'C' de Consistencia (Consistency)

La propiedad de Consistencia establece que una transacción solo puede transformar la base de datos de un estado válido a otro estado válido. Esto significa que cualquier dato escrito en el sistema debe respetar rigurosamente todas las reglas de validación, restricciones de integridad, claves primarias y foráneas, disparadores (triggers) y cualquier otra norma lógica definida explícitamente en el esquema del sistema.

A diferencia de las otras tres propiedades de ACID que son gestionadas e impuestas de forma casi exclusiva por el motor interno de la base de datos, la consistencia es una responsabilidad compartida entre el diseño del motor y el desarrollador de la aplicación de software. El sistema provee las herramientas (constraints) para evitar datos absurdos, pero el modelo de negocio conceptual debe estar bien diseñado para que estas restricciones reflejan la realidad operativa de la organización.

3.1 Invariantes de Estado y Preservación del Modelo de Negocio

Un concepto clave dentro de la consistencia es la noción de 'invariante'. Un invariante es una condición o una afirmación lógica sobre los datos que siempre debe ser verdadera. Por ejemplo, en una base de datos contable corporativa, un invariante fundamental es que la suma total de los activos debe ser perfectamente igual a la suma de los pasivos más el patrimonio neto. Si una transacción de balance rompe esta igualdad, el estado de la base de datos se vuelve inconsistente.

Cuando una transacción viola una restricción de integridad o un invariante a mitad de su ejecución, el motor de la base de datos detecta inmediatamente la anomalía, aborta la operación completa e invoca las rutinas de atomicidad descritas previamente para limpiar el progreso. De esta forma, se previene activamente la existencia de 'datos huérfanos' (como un registro de compra asociado a un cliente que no existe) o balances desfasados que arruinarían las auditorías.

4. La 'I' de Aislamiento (Isolation)

La propiedad de Aislamiento (comúnmente denominada Isolation en la literatura técnica) aborda el problema de la concurrencia masiva. En un entorno real, una base de datos corporativa no procesa una sola transacción a la vez. Por el contrario, decenas, cientos o miles de usuarios e interfaces de programación de aplicaciones (APIs) leen y escriben datos de forma simultánea sobre las mismas tablas y registros exactos.

El aislamiento garantiza que la ejecución concurrente de múltiples transacciones resulte en un estado idéntico al que se obtendría si dichas transacciones se hubiesen ejecutado de manera estrictamente secuencial, una detrás de la otra. En términos prácticos, cada transacción debe operar de forma totalmente aislada dentro de su propia 'burbuja lógica', actuando como si fuera la única transacción activa en todo el universo del sistema y desconociendo los cambios temporales de los demás hasta que estén finalizados.

4.1 Fenómenos de Lectura Anómalos y Niveles de Aislamiento

Cuando el aislamiento no se implementa o se relaja demasiado en favor de la velocidad extrema, ocurren fenómenos de lectura anómalos que destruyen la lógica de las aplicaciones. Los tres problemas clásicos de concurrencia son:

• Lectura Sucia (Dirty Read): Ocurre cuando una transacción 'A' modifica un registro y una transacción 'B' lee ese registro modificado antes de que 'A' haga Commit. Si 'A' luego decide hacer Rollback debido a un error, la transacción 'B' habrá operado con datos fantasmas que nunca existieron formalmente en el almacenamiento permanente.
• Lectura No Repetible (Non-repeatable Read): Pasa cuando una transacción 'A' lee un registro, luego una transacción 'B' modifica o elimina ese mismo registro y hace Commit. Si la transacción 'A' vuelve a leer el registro dentro del mismo flujo,

descubrirá con sorpresa que el dato cambió o desapareció a mitad de su proceso.

• Lectura Fantasma (Phantom Read): Sucede cuando una transacción 'A' ejecuta una consulta para contar filas bajo un criterio, y simultáneamente una transacción 'B' inserta nuevas filas que cumplen ese criterio. Al repetir la consulta, 'A' obtiene un número diferente de filas.

Para mitigar esto, el estándar SQL define cuatro Niveles de Aislamiento ajustables por el administrador del sistema según el equilibrio requerido entre seguridad absoluta y

5. La 'D' de Durabilidad (Durability)

La propiedad de Durabilidad es la garantía definitiva de permanencia en el tiempo. Establece de forma tajante que una vez que una transacción ha completado su ejecución con éxito y el sistema ha retornado un mensaje de confirmación (Commit) al usuario o aplicación, los cambios realizados por dicha transacción se vuelven definitivos, inmutables y persistentes ante cualquier catástrofe posterior.

Si un milisegundo después de recibir la confirmación de una transacción el servidor sufre un corte total de energía eléctrica, el sistema operativo colapsa por un pantallazo azul, o los discos duros experimentan un fallo crítico parcial, el estado modificado no debe perderse bajo ninguna circunstancia. Al reiniciar el hardware y levantar nuevamente el motor de la base de datos, el sistema iniciará una rutina automática de recuperación que leerá los diarios de transacciones y se asegurará de consolidar físicamente los cambios que hubieran quedado pendientes en la memoria RAM.

5.1 Almacenamiento no Volátil, Caché de Disco y Respaldos Automatizados

Para cumplir la promesa de la durabilidad, los sistemas DBMS no pueden confiar únicamente en la memoria de acceso aleatorio (RAM), ya que esta es volátil y se borra por completo al interrumpirse la corriente eléctrica. Los motores interactúan de forma directa con los subsistemas de almacenamiento no volátil, tales como unidades de estado sólido (SSD) y discos duros magnéticos de nivel empresarial.

Además, se emplean técnicas avanzadas como el vaciado forzado de búfer (Buffer Flushing / fsync) para obligar al sistema operativo a escribir físicamente los datos del caché de memoria al medio físico antes de considerar el Commit cómo finalizado. En infraestructuras profesionales de alta disponibilidad, la durabilidad se extiende mediante la replicación síncrona en servidores secundarios geográficamente distantes y la generación constante de copias de seguridad en caliente.

7. Alternativas Modernas: El Paradigma NoSQL y el Teorema de CAP

Durante décadas, las propiedades ACID dominaron de forma absoluta el desarrollo de sistemas de información. No obstante, con la llegada de la era de Internet a escala masiva, la aparición de redes sociales globales (como Facebook, Instagram o Twitter) y la necesidad de gestionar volúmenes de Big Data distribuidos en miles de servidores globales, el modelo ACID tradicional comenzó a mostrar limitaciones severas de escalabilidad vertical.

El científico Eric Brewer formuló el célebre Teorema de CAP, el cual demuestra matemáticamente que en un sistema de datos distribuido es físicamente imposible garantizar de forma simultánea tres características fundamentales: Consistencia inmediata (C), Disponibilidad constante (A - Availability) y Tolerancia a la partición de red (P). Dado que en redes globales los cortes y retrasos de fibra óptica (particiones) son inevitables, los ingenieros debieron elegir entre Consistencia Estricta o Disponibilidad de Respuesta.

Esto dio origen al paradigma opuesto a ACID, denominado BASE (Basically Available, Soft state, Eventual consistency). En una base de datos tipo BASE (común en motores NoSQL como Cassandra o MongoDB), se prioriza la velocidad y la disponibilidad. Por ejemplo, si das 'Me Gusta' a una foto, no importa si tus amigos en Japón tardan tres segundos en ver actualizado ese dato (Consistencia Eventual). Sin embargo, para datos transaccionales duros, ACID sigue siendo irreemplazable.

8. Conclusiones Generales del Informe

Las propiedades ACID no representan una moda tecnológica pasajera, sino un pilar teórico y práctico fundamental que sostiene la infraestructura digital del mundo moderno. Garantizar la atomicidad, la consistencia, el aislamiento y la durabilidad es la diferencia directa entre un sistema de software profesional y confiable, y una base de datos frágil propensa a la pérdida catastrófica de activos e información crítica.

Para un ingeniero o profesional de tecnología, comprender a fondo estos mecanismos internos permite tomar decisiones arquitectónicas maduras y responsables. Al equilibrar de manera inteligente la rigidez estructural de los entornos ACID con la flexibilidad escalable de los nuevos entornos NoSQL, se pueden construir ecosistemas de información balanceados, eficientes y, por encima de todo, seguros para los usuarios finales.