UNIVERSIDAD DEL SURESTE DE LA FRONTERA COMALAPA

ASIGNATURA: Bases de Datos II

DOCENTE: Berning Eduardo Aguilar Cordova

ALUMNO: Josué Roberto Pérez López

CUATRIMESTRE: Octavo

GRUPO: A

CARRERA: Ingeniería en sistemas computacionales.

PARCIAL: Segundo

TRABAJO: Ensayo unidad 1 a 3

FECHA: 9 de Abril de 2023.

INTRODUCCION

En el presente documento abordaremos la creación de una Base de Datos, cual es el proceso de la recuperación que dicha base debe tener, veremos las condiciones que determinan una transacción, sus propiedades y los tipos que existen. Que hacer cuando se presenten las fallas en una transacción, fallas en el sistema y fallas en el medio, veremos también el DML o (lenguaje de manipulación de datos).

Analizaremos la concurrencia y en que momento se puede presentar, ¿es esto bueno o malo para nuestra base de datos?, ¿Qué es un bloqueo mortal?, ¿se puede evitar en nuestra base de datos?, abordaremos este tema a continuación.

DESARROLLO

Una transacción es una colección de acciones que hacen transformaciones consistentes de los estados de un sistema preservando la consistencia del sistema. Sin embargo, durante la ejecución de una transacción, la base de datos puede estar temporalmente en un estado inconsistente. El punto importante aquí es asegurar que la base de datos regresa a un estado consistente al fin de la ejecución de una transacción. Una transacción siempre termina, aun en la presencia de fallas. Sin embargo, aún no se ha dado ninguna justificación para afirmar que las transacciones son unidades de procesamiento consistentes y confiables. La atomicidad requiere que, si una transacción se interrumpe por una falla, sus resultados parciales deben ser deshechos. La actividad referente a preservar la atomicidad de transacciones en presencia de abortos debido a errores de entrada, sobrecarga del sistema o interbloqueos se le llama recuperación de transacciones. La actividad de asegurar la atomicidad en presencia de caídas del sistema se le llama recuperación de caídas. En otras palabras, una transacción es un programa correcto que lleva la base de datos de un estado consistente a otro con la misma característica. Debido a esto, las transacciones no violan las restricciones de integridad de una base de datos. Una transacción en ejecución no puede revelar sus resultados a otras transacciones concurrentes antes de su commit. Es la propiedad de las transacciones que asegura que una vez que una transacción hace su commit, sus resultados son permanentes y no pueden ser borrados de la base de datos. Por lo tanto, los DBMS aseguran que los resultados de una transacción sobrevivirán a fallas del sistema. Esta propiedad motiva el aspecto de recuperación de bases de datos, el cual trata sobre como recuperar la base de datos a un estado consistente en donde todas las acciones que han hecho un commit queden reflejadas Aun cuando los problemas fundamentales son los mismos para las diferentes clases, los algoritmos y técnicas que se usan para tratarlas pueden ser considerablemente diferentes. Las transacciones pueden ser agrupadas a lo largo de las siguientes dimensiones: Las transacciones que operan en datos distribuidos se les conoce como transacciones distribuidas. Por otro lado, dado que los resultados de una transacción que realiza un commit son durables, la única forma de deshacer los efectos de una transacción con commit es mediante otra transacción. Tomando en cuenta el tiempo que transcurre desde que se inicia una transacción hasta que se realiza un commit o se aborta, las transacciones pueden ser de tipo batch o en línea. Las transacciones en línea se caracterizan por tiempos de respuesta muy cortos y por accesar una porción relativamente pequeña de la base de datos. Por otro lado, las transacciones de tipo batch toman tiempos

relativamente largos y accesan grandes porciones de la base de datos. El sistema debe estar preparado para recuperarse no sólo de fallas puramente locales, como la aparición de una condición de desborde dentro de una transacción, sino también de fallas globales, como podría ser la interrupción del suministro eléctrico al CPU Las fallas locales son las que afectan sólo a la transacción en donde ocurrió. Las fallas de sistema se conocen también como caídas (crash) suaves. Una falla de los medios de almacenamiento es un percance en el cual se destruye físicamente alguna porción de la DB. Un sistema de gestión de base de datos consta de varios componentes, todos los cuales contribuyen al buen funcionamiento del software. La atomicidad o integridad describe la propiedad de “todo o nada” de los SGBD, por la que todas las fases de una transacción deben finalizarse por completo y en el orden correcto para que esta sea válida. La permanencia implica que todos los datos queden almacenados permanentemente en el SGBD, no solo después de una transacción correcta, sino también o especialmente en caso de error o caída del sistema. Otros enfoques para organizar los datos son el modelo de base de datos de red, donde los datos, como el nombre indica, se estructuran en forma de red, o el modelo de bases de datos orientada a objetos, en el que no solo importa la relación entre los registros de datos, sino también el concepto de la herencia: esto significa que los objetos pueden transferir algunos de sus atributos a otros objetos, lo que se regula a través del SGBD. Existen tres formas en las que una transacción, aunque sea correcta por sí misma, puede producir una respuesta incorrecta si alguna otra transacción interfiere con ella en alguna forma. Los tres problemas son: El problema de la Actualización Perdida, El problema de la Dependencia No Confirmada, El problema del Análisis Inconsistente. El control de transacciones concurrentes en una base de datos brinda un eficiente desempeño del Sistema de Administración de Base de Datos, puesto que permite controlar la ejecución de transacciones que operan en paralelo, accediendo a información compartida y, por lo tanto, interfiriendo potencialmente unas con otras.

En sistemas operativos, el bloqueo mutuo (también conocido como interbloqueo, traba mortal, deadlock, abrazo mortal) es el bloqueo permanente de un conjunto de procesos o hilos de ejecución en un sistema concurrente que compiten por recursos del sistema o bien se comunican entre ellos. A diferencia de otros problemas de concurrencia de procesos, no existe una solución general para los interbloqueos. El control de concurrencia y detección y manejo de bloqueos es un área de mucho studio en las bases de datos distribuidas, a pesar de esto no hay ningún algoritmo aceptado para solucionar el problema. Para el control de bloqueos mutuos no se ha desarrollado ninguna solución viable y la forma más simple y

que la mayoría de productos utilizan es la implementación de un tiempo máximo de espera en las peticiones de bloqueos. El algoritmo 2PL utiliza bloqueos de lectura y escritura para prevenir conflictos entre operaciones. as reglas básicas para manejar los bloqueos son: transacciones distintas no pueden tener acceso simultáneamente a un elemento (lectura-escritura o escritura- scritura), y una vez se libere un bloqueo no se puede pedir otro, es decir, los bloqueos de la transacción crecerán mientras no libere ninguno y luego de liberar alguno solo puede liberar los demás.

Cada transacción realizada se le asigna un timestamp (literalmente: sello de tiempo) único en el nodo que se originó. Una forma de evitar los problemas de interferencia es no permitir que las transacciones se intercalen. Una ejecución en la cual ninguna de dos transacciones se intercala se conoce como serial. Si el DBS procesara transacciones serialmente, significaría que no podría ejecutar transacciones concurrentemente, si entendemos concurrencia como ejecuciones intercaladas. Una solución puede ser ampliar el rango de ejecuciones permisibles para incluir aquellas que tienen los mismos efectos que las seriales. Dichas ejecuciones se conocen como serializables. Las ejecuciones que ilustran la actualización perdida y el análisis inconsistente no son serializables.

El deadlock, conocido también como abrazo mortal o bloqueo mutuo en programación, se refiere a la situación en la que dos o más tareas, como procesos o hilos, se pausan esperándose la una a la otra para poder llevar a cabo la continuación o finalización de su trabajo. Cabe destacar que estos procesos o labores implicadas en el Deadlock se encuentran a la espera de un recurso determinado que no se les será asignado, como puede ser un componente de CPU, de memoria o de entrada/salida. Dentro de las características del deadlock en programación, se encuentra que este puede presentarse por diversas situaciones, como puede ser que varios procesos compitan entre sí por la asignación de uno o más recursos determinados. Esta destaca como la causa más común de los bloqueos mutuos. La estrategia de prevención del interbloqueo consiste, a grandes rasgos, en diseñar un sistema de manera que esté excluida, a priori, la posibilidad de interbloqueo. Los métodos para prevenir el interbloqueo son de dos tipos. Los métodos indirectos consisten en impedir la aparición de alguna de las tres condiciones necesarias, antes mencionadas (condiciones 1 a 3). Los métodos directos consisten en evitar la aparición del círculo vicioso de espera (condición 4).

Las estrategias de prevención del interbloqueo son muy conservadoras; solucionan el problema del interbloqueo limitando el acceso a los recursos e imponiendo restricciones a

los procesos. En el lado opuesto, las estrategias de detección del interbloqueo no limitan el acceso a los recursos ni restringen las acciones de los procesos. Un sistema que pretenda recuperarse del interbloqueo, debe invocar a un algoritmo de detección cuando lo considere oportuno.

Pensar en seguridad de datos y construir defensas desde el primer momento es de vital importancia. Los ingenieros de seguridad tienen como objetivo proteger la red de las amenazas desde su inicio hasta que son confiables y seguras. La ingeniería de seguridad cubre mucho terreno e incluye muchas medidas, desde pruebas de seguridad y revisiones de código regulares hasta la creación de arquitecturas de seguridad y modelos de amenazas para mantener una red bloqueada y segura desde un punto de vista holístico. En el caso de que los datos sean interceptados, la encriptación dificulta que los hackers hagan algo con ellos. Esto se debe a que los datos encriptados son ilegibles para usuarios no autorizados sin la clave de encriptación. La encriptación no se debe dejar para el final, y debe ser cuidadosamente integrada en la red y el flujo de trabajo existente para que sea más exitosa. Si en la red ocurren acciones de aspecto sospechoso, como alguien o algo que intenta entrar, la detección de intrusos se activará. Los sistemas de detección de intrusos de red (NIDS) supervisan de forma continua y pasiva el tráfico de la red en busca de un comportamiento que parezca ilícito o anómalo y lo marcan para su revision.

Los hackers suelen analizar las redes de forma activa o pasiva en busca de agujeros y vulnerabilidades. Los analistas de seguridad de datos y los profesionales de la evaluación de vulnerabilidades son elementos clave en la identificación de posibles agujeros y en cerrarlos. El análisis de vulnerabilidad (que identifica amenazas potenciales) también puede incluir deliberadamente investigar una red o un sistema para detectar fallos o hacer pruebas de intrusion. El software anti-malware y anti-spyware también es importante. Está diseñado para supervisar el tráfico de Internet entrante o el malware como spyware, adware o virus troyanos. Se pueden prevenir ataques de ransomware siguiendo buenas prácticas de seguridad, como tener software antivirus, el último sistema operativo y copias de seguridad de datos en la nube y en un dispositivo local.

La matriz de autorización ayuda a proteger el acceso a los datos en el sistema SAP mediante los objetos de autorización. La matriz concede autorización sólo después de completer verificaciones complejas con múltiples condiciones. La matriz concede autorización basada en el papel del individuo en la organización. SAP desarrolló base para organizaciones con muchos empleados. La configuración de seguridad y administración de

base utiliza un proceso de múltiples fases, y este proceso ayuda a garantizar la adecuada seguridad, integridad de la información y privacidad de las personas dentro de la organización.

CONCLUSIÓN

La seguridad ha sido muy importante para la preservación de los datos, sin embargo, este ha pasado de utilizarse para datos clasificados de ciertos sectores privados, a tener una aplicación en diversas áreas personales. Por ello, se hace impredecible que las necesidades de seguridad potenciales sean tomadas en cuenta y se determinen para todo tipo de aplicaciones.

Aun cuando los problemas fundamentales son los mismos para las diferentes clases, los algoritmos y técnicas que se usan para tratarlas pueden ser considerablemente diferentes. Tomando en cuenta el tiempo que transcurre desde que se inicia una transacción hasta que se realiza un commit o se aborta, las transacciones pueden ser de tipo batch o en línea. Por otro lado, las transacciones de tipo batch toman tiempos relativamente largos y accesan grandes porciones de la base de datos. La atomicidad o integridad describe la propiedad de “todo o nada” de los SGBD, por la que todas las fases de una transacción deben finalizarse por completo y en el orden correcto para que esta sea válida. La permanencia implica que todos los datos queden almacenados permanentemente en el SGBD, no solo después de una transacción correcta, sino también o especialmente en caso de error o caída del sistema. Otros enfoques para organizar los datos son el modelo de base de datos de red, donde los datos, como el nombre indica, se estructuran en forma de red, o el modelo de bases de datos orientada a objetos, en el que no solo importa la relación entre los registros de datos, sino también el concepto de la herencia: esto significa que los objetos pueden transferir algunos de sus atributos a otros objetos, lo que se regula a través del SGBD.