EL PROBLEMA DE CONTROL DE CONCURRENCIA EN LAS BASES DE DATOS

Rodrigo Guzmán Maldonado

Universidad de Guanajuato

iscrodgm@gmail.com

Geovanni Hernández Gómez

Universidad de Guanajuato, Campus Irapuato Salamanca, Sede Yuriria

geov.hernandez@ugto.mx

Hugo Armando Aguilera García

Universidad de Guanajuato, Campus Irapuato Salamanca, Sede Yuriria

kakistos741@gmail.com

Resumen

Una de las problemáticas más populares dentro del área de Base de Datos es el Control de Concurrencia, el cual consiste en la implementación de algoritmos de acuerdo con distintos enfoques, para resolver cuellos de botella. A través de una transacción los usuarios modifican el estado de una Base de Datos. Cuando existen múltiples usuarios intentando acceder a los diferentes objetos de una Base de Datos, el problema de Control de Concurrencia es más complejo. El presente artículo tiene como finalidad, mostrar una semblanza sobre algunos enfoques de control de concurrencia que han sido propuestos y analizados con anterioridad con la intención de formar una guía de consulta para quienes se encuentren trabajando en el tema.

Palabras Claves: Bases de datos, Control de concurrencia, SGBD.

Abstract

Nowadays, Databases represent not just a set of necessary information for the decision-making process, but rather represent entities themselves in the real world.

One of the most popular issues in Databases is the concurrency control, which is intimately related with the implementation of algorithms to solve bottleneck. A transaction permits a user to modify Database`s state. When multiple users are trying to access different Database`s objects, the control concurrency problem emerges. The objective of this paper is to show the different concurrency control approaches in order to facilitate the understanding of this subject.
Keywords: *Databases, Concurrency control, DBMS.*

1. Introducción

En la actualidad las Bases de Datos (BD) no sólo son un conjunto de información que permite a las distintas entidades realizar un estudio para tomar decisiones, sino que los valores representan entidades en el mundo real.

Mientras que las tecnologías Multi-Core evolucionan a pasos agigantados en relación a los avances en el área de hardware en su poder de procesamiento, los sistemas convencionales de Gestión de Bases de Datos (SGBD) escalan lentamente con apenas unas cuantas transacciones Multi-Core.

La problemática de CC (control de concurrencia) de datos, dependerá del enfoque que esté siendo empleado en un SGBD en particular. El estado de la BD cambia al momento de que un usuario ejecuta una transacción. Si se hablara de una transacción individual, ejecutándose de manera aislada, podemos suponer que las transacciones se realizarán sin contratiempos. Sin embargo, cuando el sistema debe atender múltiples solicitudes de distintos usuarios, intentando varios accesos a los diferentes objetos de una BD que puede o no estar distribuida; el problema de CC emerge. Se expone el problema de CC en una BD y se enuncian los algoritmos básicos para afrontarlo. De igual manera se menciona esta problemática en las BDD (Bases de Datos Distribuidas) y además en un ambiente que incluye dispositivos móviles.

2. Método

Cuando hablamos de coordinar las ejecuciones de transacciones simultaneas en una BD multiusuario, estamos hablando de control de concurrencia [Rob,

2009]. Cada vez que dos o más transacciones se colapsan, se considera como inconsistente a la Base de Datos. El control de concurrencia es el proceso por medio del cual le damos un estado de consistencia a cualquier Base de Datos [Mhatre, 2014]. Qasim et al, definen una transacción en un SGBD como una secuencia de operaciones de escritura y lectura. Existen cuatro propiedades para las transacciones y se definen como ACID, por sus siglas en inglés, (*Atomicity, Consistency, Isolation and Durability*), atomicidad, consistencia, aislamiento y durabilidad [Qasim, 2016].

Para el diseño de algoritmos de control de concurrencia, existen básicamente tres enfoques genéricos que pueden ser: *Wait, Timestamp y Rollback* [Bhargava, 1999].

Un algoritmo basado en el mecanismo *Wait*, propone como solución en los casos donde hay transacciones en conflicto, que una de las transacciones espere hasta que la otra haya desocupado la entidad sobre la que ambas quieren actuar [Mandeep, 2013]. Para poder implementar una acción como esta, el sistema provee bloqueos en las entidades de la base de datos. Si una transacción solicita un bloqueo para una entidad, y ya se encuentra bloqueada por otra, la transacción solicitante debe esperar [Bhargava, 1999]. Para reducir el tiempo de espera cuando una transacción quiere hacer una lectura, existen dos tipos de bloqueos que pueden emplearse, basados en sí la transacción quiere realizar una operación de lectura o escritura:

- Readlock: La transacción bloquea la entidad en modo colaborativo o compartido. Cualquier otra transacción en espera de leer la misma entidad también obtiene un bloqueo de lectura [Mandeep, 2013].
- Writelock: El bloqueo de una entidad se hace en modo exclusivo. Si alguna transacción desea escribir en una entidad, ninguna otra puede obtener un bloqueo de lectura o escritura [Mandeep, 2013].

Dentro de los criterios involucrados en el control de concurrencia, resalta por su importancia la serialización de las transacciones [Rob, 2009]. En los mecanismos de asignación de estampas de tiempo (*Timestamping*) el orden de serialización es prioritaria, las transacciones que se van a ejecutar están obligadas a obedecer

dicho orden. En el orden de estampas de tiempo, cada transacción tiene asignada una estampa de tiempo única y exclusiva que se asigna a través de un planificador o controlador de concurrencia. Es obvio que para lograr estampas de tiempo únicas en transacciones que provienen de diferentes nodos de sistemas distribuidos, los relojes de todos los nodos deben estar sincronizados, ya que si no lo están, habría que resolver transacciones con estampas de tiempo idénticas [Bhargava, 1999].

Los algoritmos basados en mecanismos de reversión (*Rollback*), también conocidos como algoritmos de no-bloqueo, o bien, algoritmos optimistas de CC. En este enfoque la idea es validar una transacción contra un grupo de transacciones previamente comprometidas. Si la validación falla, el valor de lectura de la transacción es actualizado y dicha transacción repite su cómputo para volver a realizar el proceso de validación. La fase de validación utilizará los conflictos entre los valores de lectura y de escritura usando alguna información de estampas de tiempo. El procedimiento de validación comienza cuando una transacción ha concluido su ejecución dentro de la suposición optimista de que otras transacciones no se conflictuarán con ésta [Bhargava, 1999]. En un enfoque optimista, cada transacción se mueve a través de dos o tres fases, conocidas como lectura, validación y escritura [Rob, 2009].

Durante la fase de lectura, la transacción lee la BD, ejecuta los cómputos necesarios y realiza las actualizaciones en una copia privada de los valores de la BD. Todas las operaciones actualizadas de la transacción son grabadas en un archivo temporal actualizado, el cual no pueden acceder las transacciones remanentes. En la fase de validación, la transacción es validada para asegurar que los cambios hechos no afectarán la integridad y consistencia de la BD. Si la validación resulta positiva, la transacción pasa a la fase de escritura, de lo contrario, la transacción reinicia y se descarta cualquier cambio. Finalmente, en la fase de escritura, los cambios se aplican de manera permanente a la base de datos. Cabe mencionar que los enfoques optimistas son aceptables para la mayoría de las BDs que requieren pocas actualizaciones de transacciones [Rob, 2009].

El planificador es un proceso especial del SGBD que establece el orden en el que las operaciones dentro de transacciones concurrentes serán ejecutadas. El planificador intercala la ejecución de las operaciones de la BD para asegurar la seriabilidad y el aislamiento de las transacciones. Para determinar el orden apropiado, el planificador determina sus acciones de acuerdo con algoritmos de CC como los ya antes mencionados. Sin embargo, es importante mencionar que no todas las transacciones se pueden serializar. El SGBD determina qué transacciones se pueden serializar y procede a intercalar la ejecución de las operaciones de estas transacciones. Por lo general, aquellas transacciones que no quedan sujetas a un proceso de serialización, se ejecutan de acuerdo con los parámetros de primeras en llegar, primeras en ser atendidas (*first-come, first served*), establecidos por el SGBD para darles salida [Rob, 2009].

Bases de Datos Distribuidas

En esta sección revisaremos el problema de control de concurrencia en un ambiente distribuido, comenzaremos definiendo una base de datos distribuida como un conjunto de datos compartidos y distribuidos sobre una red de computadoras. Los datos pueden ser accedidos a través de una red de alta velocidad [Qasim, 2016].

En un sistema de BD distribuida, normalmente la utilizan muchos usuarios. Estos sistemas regularmente permiten múltiples transacciones ejecutarse concurrentemente al mismo tiempo. El control de concurrencia es la actividad de coordinar los accesos concurrentes a una base de datos en un sistema administrador multiusuarios. El control de concurrencia permite a los usuarios acceder a una BD distribuida, mientras conserva la ilusión de que se está ejecutando en sistema dedicado. Cuando una transacción actualiza datos concurrentes, podría generar varios problemas con la consistencia de los mismos. [Qasim, 2016].

Mandeep y Kaur (2013) en su trabajo, identifican que los sistemas de BD distribuidas son necesarios para aplicaciones donde los datos y sus accesos son inherentemente distribuidos para incrementar la disponibilidad durante las fallas.

Accediendo a los datos en donde sean requeridos y estén disponibles de manera local para los usuarios.

Qasim et al. [Qasim, 2016] en su artículo, identifica un par de tipos de bases de datos distribuidas, utilizando como criterio la uniformidad de recursos: · Homogéneos: Datos distribuidos, pero todos los sitios tienen el mismo sistema gestor de bases de datos. · Heterogéneos: Los datos están distribuidos, y cada sitio utiliza un sistema gestor de bases de datos diferente.

Sin importar el tipo de base de datos distribuida, la metodología para diseñarla es la misma que es usada para las BDs centralizadas [Mandeep, 2013]. Sin embargo, algunos factores adicionales han de ser considerados tales como la Fragmentación y Replicación de los datos. La fragmentación identifica las unidades de distribución y localización de los datos separándolas en unidades llamadas fragmentos. Los fragmentos son almacenados en las diferentes ubicaciones involucradas en el sistema. Existen tres tipos de fragmentación de datos: Horizontal, Vertical e Hibrida. La replicación es una opción utilizada para manejar la tolerancia a fallos. Así como las copias de la BD son guardadas en dos o más sitios para su resguardo. Además, es posible crear una copia de cada fragmento en distintos sitios.

Transacciones en una base de datos distribuida

Una transacción distribuida es capaz de ejecutar múltiples procesos, y normalmente esta se realiza con la interacción de varias máquinas. Los sistemas de procesamiento de transacciones distribuidas están diseñados para facilitar transacciones de extensión heterogénea. Una transacción puede incluir varios administradores encargados de recursos. La ejecución de transacciones distribuidas requiere de la existencia de una correcta coordinación entre el sistema administrador global de transacciones y todos los administradores de recursos de los sistemas individuales involucrados [Mandeep, 2013].

El control de concurrencia en una base de datos distribuida. El mecanismo de CC de los Sistemas de BDD asegura que la consistencia de la BD sea mantenida en un ambiente distribuido multiusuarios. Si una transacción es internamente

consistente (esto es que no viola ninguna de las restricciones de consistencia), su ejecución aislada sería más sencilla, una vez terminada, se ejecuta la siguiente y así sucesivamente. Es obvio que tal alternativa solo es posible teóricamente, esto no podría ser implementado en ningún sistema práctico, debido a que se minimiza el rendimiento. El nivel de concurrencia (el número de transacciones concurrentes) es probablemente el parámetro más importante en un sistema distribuido [Ozsu, 2011].

Clasificación de los algoritmos de control de concurrencia distribuidos

El criterio más común de clasificación para el control de concurrencia es la primitiva de sincronización. La correspondiente separación de los algoritmos de control de concurrencia resulta en dos clases [Bernstein, 1981]: aquellos algoritmos que son basados en accesos de exclusión mutua para compartir datos (bloqueos), y aquellos que intentan ordenar la ejecución de las transacciones de acuerdo a un conjunto de reglas (protocolos). Estas primitivas pueden ser usadas en algoritmos con dos puntos de vista diferentes: la perspectiva pesimista en la que muchas transacciones generarán conflictos entre la una y la otra, o la perspectiva optimista en la cual no muchas transacciones entrarán en conflicto con alguna otra. Los mecanismos de control de concurrencia se separan en dos clases: Métodos de control de concurrencia pesimista y métodos de control optimista.

Los algoritmos pesimistas sincronizan la ejecución concurrente de las transacciones tarde en sus ciclos de vida de ejecución, mientras que los algoritmos optimistas retardan la sincronización de la transacción hasta su terminación. El grupo pesimista consiste de los algoritmos basados en bloqueo, algoritmos basados en ordenamiento y algoritmos híbridos. El grupo de los optimistas, pueden de manera parecida ser clasificados como basados en bloqueo y basados en estampas de tiempo ordenadas. La clasificación de acuerdo con [Ozsu, 2011] se muestra a continuación.

La idea principal en el control de concurrencia basado en bloqueos es el de asegurar que un elemento de datos compartido por operaciones en conflicto sea

accedido por una operación a la vez. Esto se logra asociando un "seguro" a cada unidad bloqueada. Este seguro es asignado a una transacción antes de acceder y liberado al final del uso del dato [Ozsu, 2011].

El algoritmo de Bloqueo de Dos Fases Centralizado delega la responsabilidad de administración los seguros a un único sitio. Esto significa que solamente uno de los sitios tiene la administración de los seguros; los administradores de la transacción de otros sitios se comunican con él, en lugar de administrar sus propios seguros. Este también es conocido como algoritmo de bloqueo de dos fases de Sitio primario. Los algoritmos de Bloqueo de Dos Fases Distribuidos requieren la disponibilidad de administradores de seguros en cada sitio. En él los mensajes son enviados a el sitio central de administración de bloqueos, quién los reenvía a los otros administradores de seguros. A diferencia de los algoritmos basados en bloqueo, los algoritmos basados en estampas de tiempo, no intenta mantener la seriabilidad por exclusión mutual. En su lugar seleccionan un orden de serialización y ejecución de las transacciones. Para establecer este orden, los administradores de transacciones asignan a cada transacción una estampa de tiempo única en su inicialización. Una estampa de tiempo es un simple identificador que sirve para identificar cada transacción como única y es usado para ordenarlas [Ozsu & Valduriez, 2011]. El algoritmo básico de Estampas de Tiempo es la implementación directa del mecanismo de Estampas de Tiempo. El coordinador de transacciones administra las estampas de tiempo asignadas a cada transacción, determina los sitios donde cada elemento de datos está almacenado y envía las operaciones relevantes a estos sitios.

La versión de naturaleza conservadora de estos algoritmos relaciona la manera en que se ejecutan cada operación. El algoritmo básico de Estampas de Tiempo trata de ejecutar una operación tan pronto como sea aceptada; esto una forma "agresiva" o "progresiva". Los algoritmos conversadores, por otro lado, retardan cada operación hasta que se asegura que ninguna operación con menor estampada de tiempo puede entrar en el organizador.

En la multiversión de Estampas de Tiempo, las actualizaciones no modifican la BD; cada operación de escritura crea una nueva versión de cada elemento de

datos. Cada versión es marcada por la estampada de tiempo de la transacción que la creo. Como consecuencia el algoritmo multiversión de Estampas de Tiempo ocupa espacio de almacenamiento por algún tiempo. Al hacer eso, los procesos de cada transacción sobre el estado de la base de datos podrán ser visibles si las transacciones son ejecutadas serialmente en el orden de las estampas de tiempo [Ozsu, 2011].

Los algoritmos optimistas, por otro lado, retardan la fase de validación hasta la escritura. Como consecuencia una operación enviada al organizador optimista nunca es retardada. Las operaciones de lectura, cálculo y escritura de cada transacción son procesadas libremente sin actualizar la BD actual. Cada transacción inicialmente realiza sus actualizaciones sobre copias locales de los elementos de datos. La fase de validación consiste de la revisión si estas actualizaciones podrán mantener la consistencia de las BD. Si la respuesta es afirmativa, los cambios son hechos globalmente. En caso contrario, la transacción es abortada y reinicializada.

Interbloques distribuidos

Cualquier algoritmo de control de concurrencia basado en bloqueo puede resultar en interbloqueos, dada su exclusión mutua de acceso a los recursos compartidos (datos) y transacciones esperando bloqueos. Además, los algoritmos basados en estampas de tiempo que requieren la espera de una transacción, puede causar interbloqueos. Por esta razón, los sistemas gestores de bases de datos distribuidos requieren procedimientos especiales para manéjalos. Un interbloqueo puede ocurrir debido a una transacción que se encuentra esperando por otra. Informalmente, una situación de interbloqueo es un conjunto de peticiones que nunca puede ser resuelta por el mecanismo de control de concurrencia [Ozsu, 2011].

Prevención de Interbloqueos

Los métodos para la prevención de interbloqueos garantizan que los interbloqueos no puedan ocurrir en primer lugar. Como resultado, el administrador

de transacciones revisa una transacción cuando es inicializada y no permite su proceder si puede causar un interbloqueo. Para ejecutar esta revisión, requiere que todos los elementos de datos que serán accedidos por una transacción serán pre-declarados. A la transacción se le permitirá proceder, si el administrador de transacciones identifica que los elementos de datos están disponibles. En caso contrario, a la transacción no se le permite proceder. El administrador de transacción dispone de todos los elementos de datos que son pre declarados por una transacción para permitir su proceder.

Evasión de Interbloqueos

Los esquemas para evitar interbloqueos emplean técnicas de CC que nunca resultan en interbloqueos o demanda que situaciones potenciales de interbloqueos sean detectadas y que los pasos apropiados sean tomados para que tales situaciones no ocurran [Ozsu, 2011].

Concurrencia móvil

Con el avance de las tecnologías de comunicación, hoy en día es más simple realizar una conexión a una base de datos desde distintos sistemas de cómputo. Gracias la red tan grande que es internet, los dispositivos portátiles son capases de realizar una conexión a una base de datos para realizar transacciones fundamentales como lo son la inserción, eliminación, actualización o el lanzamiento de una rutina que modifique el estado de una base de datos, ya sea en su versión única o distribuida.

Como lo hemos hablado a lo largo de este trabajo, al momento de realizar una conexión hacia una base de datos, es necesario aplicar un mecanismo de control de concurrencia. En el caso de la conexión a una base de datos en donde tienen acceso distintos tipos de dispositivos móviles, es necesario implementar mecanismos más agiles, debido a que los datos pueden cambiar entre una conexión y otra. Esto es, cuando un dispositivo se conecta a un nodo de conexión a la base de datos es, ya que, este nodo es el que se encuentra más cercano al dispositivo portátil o móvil; pueden existir cambios en la base de datos al momento

de que el dispositivo cambia de posición realizando una nueva conexión. Como podemos notar, para realizar el cambio de conexión es necesario un tiempo de petición y respuesta a la conexión. Durante este tiempo, es posible que distintos sistemas hayan realizado variaciones en la base de datos generando así un problema de inconsistencia de datos. Además durante estos intervalos de tiempo, se pueden llegar a tener múltiples peticiones a un mismo elemento, creando con esto el problema de control de concurrencia.

Para contemplar este tipo situaciones es la que Mohana et al [Mohana, 2016] en su trabajo *Hierarchical replication and multiversion concurrency control model for mobile database systems*, trabajan el concepto de modelo jerárquico de replicación y concurrencia para tratar las multi-versiones en los sistemas de datos móviles. Para abordar esta problemática, se hace uso de Encabezados de Cluster, los cuales se encuentra basados en la distancia en donde se encuentran los nodos en conexión en una etapa anterior. La movilidad entre los dos nodos de conexión puede ser estimada a partir de la relación RSS obtenida entre la trasmisión de paquetes consecutivos de un nodo vecino. Por lo tanto, la métrica de movilidad definida como Moj (i) del nodo j con respecto a i se calcula conforme a la ecuación 1.

$$Mo_j(i) = 10\log_{10}\frac{Rss_{i\to j}^{new}}{Rss_{i\to j}^{old}} \quad (1)$$

Donde la intensidad de la señal recibida RSS es calculada de acuerdo a la ecuación 2.

$$RSS = \delta \times \phi \times P_{tx} \quad (2)$$

Donde delta es una constante dependiente de longitud de onda de la antena, phi es el canal de ganancia y P_{tx} es el poder de la señal de trasmisión. Con esto nuestra distancia de nodo ND es calculada como se muestra en la ecuación 3.

$$ND = f(Mo_j(i),\ Rss) \quad (3)$$

Para llevar el control de concurrencia, cada tabla CH mantiene una tabla de concurrencia para cada uno de los objetos duplicados. Dicha tabla incluye parámetros como id, numero de versión, fecha y hora de actualización y número de clientes que accedieron al objeto.

Con esta metodología, se propone un mecanismo de replicación y control de concurrencia utilizando multi-versiones jerárquicas en sistemas de Base de Datos Móviles. Utilizando técnicas de localización de nodos de conexión para llevar una tabla CH correspondiente a cada una de las conexiones.

Soluciones patentadas

Debido a la creciente generación de información y a la necesidad de utilizar los objetos insertados en los sistemas de base de datos, se han creado protocolos patentados para ser utilizados como mecanismos de acción ante el problema de concurrencia. Tal es el caso del trabajo patentado por Amarnagh Sai Eluri et al. En su trabajo titulado *Database Lockless Index For Accessing Multi-Version Concurrency Control Data* [Bhattacharjee, 2016a] proponen nuevamente la idea de utilizar multi-versiones a través de su sistema denominado *multi-version concurrency control* (MVCC). Como ya lo mencionamos, este sistema se basa en la localización de objetos asociados para determinar los cambios que deberá tener una base de datos al momento de alterar uno de sus elementos. Como podemos ver, también es necesario el uso de estados de tiempo para llevar el control de versiones durante múltiples acceso a los elementos de la base de datos.

El uso de multi-versiones para el control de concurrencia es utilizado por autores como Bishwaranjan Bhattacharjee, Mustafa CANIM, Mohammad Sadoghi Hamedani, Fabian Nagel y Kenneth A. Ross en su trabajo *Reducing database locking contention using multi-version data record concurrency control* [Bhattacharjee, 2016b] patentado en octubre del 2016. En este trabajo se realiza el mediante dos tipos de registros, uno lógico y uno físico de las cuales se mantiene un mapeo de sus estados. Las entradas a la base de datos se actualizan dentro de un índice para señalar el identificador de registros lógicos en lugar de los identificadores de registro físicas para reaccionar sobre los elementos comprometidos y los no comprometidos. El identificador de registro físico que

corresponde a un registro de datos es leído de la tabla de asignación de direccionamiento indirecto. Este es utilizado para acceder a una versión comprometida de registro de datos. Mientras que un elemento se encontrará modificando el registro de datos a manera de evitar el bloqueo del escritor de un lector de los elementos de la base. Un identificador de registro físico no comprometida correspondiente al registro de datos escribe en la tabla de asignación de direccionamiento indirecto para insertar una nueva versión no confirmada del registro de datos dentro de una tabla Con esto, el lector se encontrará leyendo la versión comprometida del registro de datos en un estado particular para evitar que el lector sea bloqueado por el escritor.

El uso de mecanismos para ser efectuados sobre múltiples accesos en el control de concurrencia es un problema que sigue afectando a cualquier sistema de base de datos en la actualidad. A medida que crece la tecnología, crece la información y la manera de trabajar con ella. Por ello, los mecanismos de múltiples versiones son una muy buena opción, implementada en la vida real para solventar situaciones catastróficas de accesos recurrentes y múltiples en situaciones de tiempo relativamente cortos.

Otro mecanismo muy utilizado en control de concurrencia es la utilización de bloqueos. El principal objetivo de los bloques, es simplemente como su nombre lo indica, restringe el acceso a un recurso a través de un mecanismo de prioridades. EL trabajo de Eluri et al [Eluri, 2016] patentado en estados unidos en 2016 realiza una combinación de la teoría de multi-versiones con ayuda de un índice de bloqueos. Al momento de querer acceder a un objeto, el mecanismo de múltiples versiones de control de concurrencia (MVCC) se inicia para identificar los bloques asociados a la posición de la fila mediante operaciones bit a bit. Posteriormente, se genera un bloque de estado fila calculada en base a la posición de la fila para determinar un estado de fila. Con esto, es posible acceder a al menos otro bloque por diferentes usuarios, basado en parte en el estado fila, para obtener al menos una marca de tiempo de la computado debido al desplazamiento basado en la fila. Es importante resaltar que esta técnica no es la ideal y que es necesario realizar más trabajos en el área para minorar el problema.

3. Resultados

El control de concurrencia en los sistemas de bases de datos es un tema muy analizado. Podemos identificar tres enfoques generales: Protocolos de bloqueo, Protocolos de Estampa de Tiempo y Protocolos Optimistas. Cada uno de ellos sugiere diferentes algoritmos para bloquear, actualizar y liberar los recursos de la bases de datos.

Se han propuesto e implementado gran diversidad de variaciones de estos algoritmos dependiendo del contexto: bases de datos centralizadas, distribuidas o móviles.

Para la elaboración de este trabajo, se realizaron los siguientes esfuerzos. Primero, se recopilaron artículos y bibliografía referentes a esta problemática. Una vez analizados los contenidos, se incluyeron en el trabajo aquellos que se consideraron apropiados para formar una guía útil y rápida de consulta, dirigida a las personas involucradas con la problemática del control de concurrencia.

Se recomienda a los interesados en el tema, profundizar consultando los artículos y bibliografía citados en este artículo, o bien, relacionados con el CC en las BDs, además de otras fuentes como revistas tecnológicas y artículos científicos que presenten innovaciones en la materia. También se recomienda participar en eventos como congresos, talleres, cursos-talleres, diplomados, conferencias, etc., en donde se aborden aspectos relacionados al tema de la CC en las BDs.

4. Discusión

A lo largo de este documento, se mencionaron varios trabajos que abordan el tema de Control de Concurrencia en las Bases de Datos. Cada uno de estos documentos aborda el problema de control de concurrencia en un caso específico, generando soluciones similares a través de enfoques distintos. Una de las soluciones más populares respecto de este problema, es el uso de multi-versiones de la BD, combinado con estampas de tiempo; sin embargo, dependerá de las necesidades y recursos del administrador de la BD, el enfoque que empleará para darle solución al problema.

5. Bibliografía y Referencias

[1] Bhargava B. (1999) Concurrency control in database systems. (1999). IEEE Trans Knowl Data Eng. 11(1):3-16. doi:10.1109/69.755610.

[2] Bhattacharjee, B., Canim, M., & Hamedani, M. S. (2016a). U.S. Patent No. 9,442,837. Washington, DC: U.S. Patent and Trademark Office.

[3] Bhattacharjee, B., Canim, M., Hamedani, M. S., Nagel, F., & Ross, K. A. (2016b). U.S. Patent No. 9,336,258. Washington, DC: U.S. Patent and Trademark Office.

[4] Eluri, A. S., Schreter, I., & Tonder, A. (2016). U.S. Patent No. 20,160,147,811. Washington, DC: U.S. Patent and Trademark Office.

[5] Mandeep Kaur and Kaur, H. (2013). "Concurrency Control in Distributed Database System". International Journal of Advanced Researchin Computer Science and Software Engineering, 3(7), 1443-1447.

[6] Mhatre A, Shedge R. (2014). Comparative Study of Concurrency Control Techniques in Distributed Databases. Fourth Int Conf Commun Syst Netw Technol. 2014:378-382. doi:10.1109/CSNT.2014.81.

[7] Mohana, M., & Jaykumar, C (2016). Hierarchical replication and multiversion concurrency control model for mobile database systems (MDS). Wireless Networks, 1-11.

[8] Özsu, M., & Valduriez, P. (2011). Principles of distributed database systems. Springer. https://doi.org/10.1007/978-1-4419-8834-8.

[9] Qasim, A., Hammand S., Imran A., Sridevi T. (2016). Concurrency Control in Distributed Database System. Int Conf Comp Commun Infor (ICCCI - 2016), Jan. 07/09.

[10] Rob P, Coronel C. (2009). Database Systems: Design, Implementation, and Management. Eighth Edition. United States.