Técnicas de control de concurrencia en base de datos: implementación en un sistema de gestión

Guillermo Arduino, Pedro L. Alfonzo

Facultad de Ciencias Exactas y Naturales y Agrimensura, Universidad Nacional del Nordeste, 9 de Julio 1449, Corrientes, Argentina.
{gaarduino@gmail.com, plalfonzo@hotmail.com}

Resumen. Las transacciones concurrentes en una aplicación que se implementa en una estructura cliente servidor, es de vital importancia, dado que mantener la coherencia, integridad y durabilidad de los datos que son consumidos no es una tarea trivial. La elección de una técnica de manejo de concurrencia que permita la serialización de las transacciones y la coherencia de las operaciones realizadas sobre dichos datos, debe ejecutarse de acuerdo a la naturaleza del problema que se esté tratando de resolver. En el presente artículo, se presenta la aplicación técnicas de concurrencia optimista, al modulo "autogestión de compras" en un Sistema de Gestión de Ventas. Los resultados demuestran que con la aplicación de la técnica mencionada y de acuerdo a operaciones que no suponen un bloqueo, o que el bloqueo resultaría costoso en el rendimiento del sistema, es decir, en tiempo y recursos, se consigue mantener la integridad y coherencia de los datos.

Palabras clave: Transacciones concurrentes, ACID, software de gestión, sistema de gestión de base de datos, base de datos.

1 Introduction

Se define Transacción en un sistema de gestión de base de datos (SGBD) como un conjunto de operaciones que se realizan como una sola unidad lógica de trabajo. Dicha unidad debe cumplir con cuatro propiedades básicas, conocidas como atomicidad, coherencia, aislamiento y durabilidad (ACID), para poder ser clasificada como una transacción [1][2].

1.1 Control de concurrencia

El control de las transacciones debe residir en las aplicaciones, primordialmente al especificar el inicio y fin de las mismas, en puntos que aseguren la coherencia lógica de los datos. Por lo tanto, se deben explicitar las secuencias de modificaciones de datos que los dejan en un estado coherente en relación a las reglas de negocios de la organización [3]. Lo expuesto se logra cumpliendo con las propiedades ACID. A continuación se sintetizan cada una de ellas [4]: i) Atomicidad: una transacción debe

ser una unidad atómica de trabajo, es decir, que se realicen todas sus modificaciones en los datos o no se realice ninguna; ii) Coherencia: cuando una transacción finaliza debe dejar todos los datos en un estado coherente. Deberán aplicarse todas las reglas a las modificaciones de la transacción permitiendo de esa manera mantener la integridad de los datos; iii) Aislamiento: las modificaciones realizadas por transacciones simultáneas se deben bloquear de las llevadas a cabo por otras transacciones simultáneas; iv) Durabilidad: el SGBD asegura que perduren los cambios realizados por una transacción que termina con éxito.

Por otra parte, las transacciones pueden encontrarse en los siguientes estados [3]: i) Activa: durante su ejecución; ii) Parcialmente comprometida: después de ejecutar la última instrucción; iii) Fallida: imposible de seguir ejecutándose; iv) Abortada: transacción que retrocedió y dejó la base de datos restaurada a su estado anterior; v) Comprometida: la transacción ha finalizado correctamente.

En ocasiones los usuarios tienen acceso a los datos de manera simultánea, es decir, leen o modifican los mismos datos al mismo tiempo. Cuando esto no se controla pueden suceder algunas de las siguientes situaciones problemáticas: i) Actualizaciones perdidas: cuando dos o más transacciones seleccionan los mismos datos. Con la última actualización se sobrescriben las actualizaciones realizadas por otras transacciones produciendo perdida de datos [4]; ii) Dependencia no confirmada (lectura no actualizada): cuando una transacción selecciona datos que están siendo actualizadas por otra transacción, lo que conlleva a que la segunda transacción acceda a datos que todavía no han sido confirmados [4]; iii) Análisis contradictorios (lectura irrepetible): cuando una transacción obtiene acceso a los mismos datos varias veces y en cada ocasión accede a datos diferentes. Esto se parece a la dependencia confirmada, la diferencia radica en que los datos ya han sido confirmados por la segunda transacción, con lo cual la lectura se hace irrepetible [1], [4]; iv) Lecturas fantasmas: es una situación particular que se produce en un SGBD cuando se ejecutan dos consultas idénticas al mismo tiempo y la recopilación de los datos devueltos por la segunda consulta es diferente a lo que se obtiene con la primera [1]. Por lo tanto, cuando varios usuarios intentan modificar datos en una base de datos al mismo tiempo, debe implementarse un sistema de control de forma que las modificaciones realizadas por un usuario no interfiera en forma negativa a la de otro usuario. Esto se denomina control de simultaneidad [4].

La teoría del control de simultaneidad tiene dos clasificaciones: i) Control de simultaneidad pesimista [1], [4]: un sistema de bloqueos impide que los usuarios modifiquen los datos de forma que afecte a otros usuarios. Cuando un usuario lleve a cabo una acción que da lugar a que se aplique un bloqueo, los demás usuarios no podrán realizar acciones que crearían conflictos con el bloqueo, hasta que sea liberado. Se utiliza principalmente en entornos donde existen conflictos por la obtención de datos, y en los que el costo de la protección de los mismos con bloqueos es menor que el de revertir las transacciones si se producen conflictos de simultaneidad; ii) Control de simultaneidad optimista [2][5][6]: no se bloquean los datos cuando se acceden a ellos para la lectura. Cuando un usuario realiza una actualización de datos, el sistema comprueba si otro usuario los modificó después de la lectura. Si se actualizaron los datos, se produce un error. Normalmente, el usuario que recibe el error revierte la transacción y comienza de nuevo. Se utiliza

principalmente en entornos donde hay pocos problemas de contención por la obtención de datos y en los que el costo de revertir ocasionalmente una transacción es menor que el de bloquear los datos cuando son leídos.

1.2 Técnicas de control de concurrencia

A continuación se mencionan las técnicas con la que se logran evitar problemas de concurrencia aplicadas a la simultaneidad de transacciones en un SGBD. Como se indicó anteriormente entre las técnicas de concurrencia existen dos grandes grupos [1], [3], [4]: i) Control de simultaneidad pesimista y ii) Control de simultaneidad optimista. En cada grupo se incluyen diferentes técnicas que permiten manejar esa simultaneidad de manera eficiente [1], [7]:

a) Control de simultaneidad pesimista:
   - Técnicas basadas en bloqueos (Fig. 1).
   - Técnicas Multi-Granulares (Fig. 2).
   - Técnicas basadas en marcas temporales (Fig. 3)).

b) Control de simultaneidad optimista.
   - Técnicas basadas en Validación.
   - Técnica original de control de concurrencia optimista.

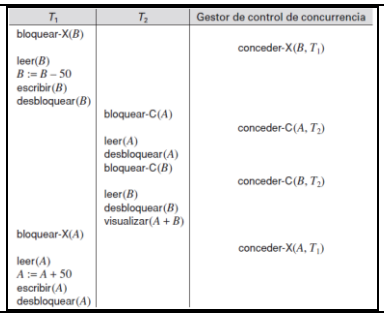


Fig. 1. Técnicas basadas en bloqueos (Fuente: [1])

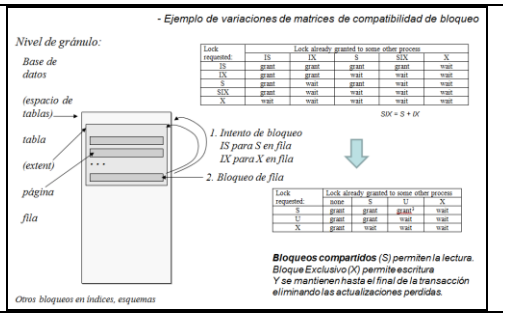


Fig. 2. Técnicas Multi-Granulares (Fuente: [7])

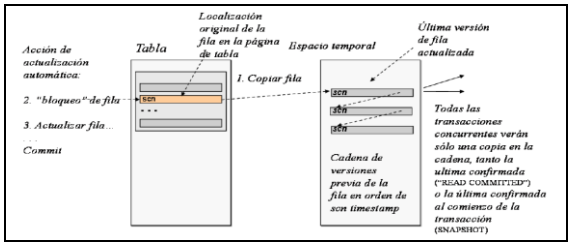


Fig. 3. Técnicas basadas en marcas temporales (Fuente: [7]).

1.3 Las base de datos y las transacciones concurrentes

A continuación se describe la técnica de concurrencia optimista basada en validación utilizado en el presente trabajo. Como se expresó anteriormente se presume que todas las transacciones finalizan, es decir, quedan en un estado transaccional comprometido, pero sujetas a una validación final. Las fases que debe cumplir una transacción son las siguientes [1], [4]: i) Fase de lectura: las transacciones se almacenan de forma temporal sin actualizar la base de datos; ii) Fase de validación: se realiza una prueba para verificar si es factible copiar esas variables locales temporales en la base de datos sin que se produzca alguna transgresión de secuencialidad; iii) Fase de escritura: si no se ha producido ninguna violación de secuencialidad las operaciones se escriben en la base de datos.

Cada transacción puede combinar las tres fases citadas, siempre y cuando respeten ese orden de secuencia en la ejecución de las transacciones concurrentes. Por lo tanto, para validar las transacciones se debe conocer el momento en que tienen lugar las fases. Para ello, se establece el orden de secuencia a través de las técnicas basadas en marcas temporales [1]. En esta técnica la probabilidad de conflictos entre transacciones es muy baja y se obtiene un tiempo de respuesta más rápida y evita los retrocesos en cascada, dado que las escrituras son sólo después de que la transacción que ejecuta la escritura se encuentre comprometida. Sin embargo, existe una posibilidad de que las transacciones extensas nunca sean comprometidas, debido a largas secuencias de transacciones cortas conflictivas que provocan reinicios de aquella transacción. Para evitar esto es necesario bloquear las transacciones conflictivas de forma temporal permitiendo que la transacción afectada sea comprometida [1], [2].

Las aplicaciones prácticas sobre transacciones concurrentes han sido estudiados como temas recientes [1], [2], [4], [5]. Es importante mantener la integridad de un SGBD impidiendo que las transacciones no finalicen en un estado intermedio [2]. En el caso en que se deba cancelarse la transacción, deberá quedar el SGBD en su estado original [4].

Por lo expuesto, en este trabajo se propone implementar transacciones concurrentes en SGBD que permita el acceso de multiples Clientes al Servidor, aplicando control de concurrencia optimista en un módulo de “módulo de autogestión de compras” de un sistema de gestión, garantizando las propiedades ACID.

2 Metodología

La metodología utilizada en la construcción del sistema de gestión se basó en las siguientes etapas:
Etapa 1: Revisión y estudio de técnicas de transacciones concurrentes en base de datos [1], [2], [4], [7].
Etapa 2: Selección de un método de transacción concurrente en base de datos. Se optó por un esquema de validación, que es un método de control de concurrencia optimista, que garantiza el acceso a múltiples usuarios [1], [4], [7].
Etapa 3: Selección y estudio de herramientas [8], [9], [10], [11].

Etapa 4: Desarrollar un sistema de gestión, eligiéndose para tal fin el modelo de prototipo perteneciente al ciclo de vida del modelo Evolutivo [12]. A continuación se menciona algunas actividades desarrolladas:

- Se definieron las funciones principales del sistema: i) Administración y control de usuarios; ii) Administración y ventas y iii) Administración y control de stock.
- Se establecieron los perfiles de usuarios. i) Administrador: gestiona y administra el Sistema; ii) Vendedor: recibe y notifica el pedido terminado y entregado; iii) Cajero: registra los cobros diarios por las ventas; iv) Cliente: realiza el pedido por pantalla táctil de los diferentes tipos de productos.
- Se organizó el sistema bajo el modelo de Cliente-Servidor.
- Se diseño de la base de datos. Se definieron adecuadamente las tablas y sus relaciones.
- Se definieron las funcionalidades del sistema teniendo en cuenta los requerimientos. Se utilizó el Lenguaje de Modelado Unificado (UML) para describirlas, a partir de los requerimientos, utilizándose: i) Casos de Uso; ii) Conversaciones y diagramas de secuencias para detallar la implementación de los Casos de Uso.

Etapa 5: Desarrollo e implementación del módulo de "autogestión de compra" del sistema desarrollado, aplicando un método de transacción concurrente.

Se construyó el prototipo del módulo de "autogestión de compras" con el diseño de interfaces gráficas que permiten interactuar a los actores intervinientes en la concreción de la compra de un producto determinado, como así también la modificación y cancelación del mismo (Fig. 4).

Las conexiones de base datos y las transacciones aplicadas al SGBD se desarrollaron en lenguaje SQL, y se utilizó como gestor Microsoft SQL Server Express 2005. El módulo se construyó en base a los requerimientos y con la modalidad iterativo incremental. En la Tabla 1 se enumeran las funcionalidades desarrolladas en dos etapas iterativas al módulo de autogestión al cual luego se implementó la técnica optimista.

Tabla 1. Funcionalidades incorporadas al módulo de autogestión.

Incrementos	**Funcionalidad**
Primero	a)Seleccionar Productos; b) Seleccionar Gusto; c) Confirmar Compra
Segundo	a)agregar Nuevo producto; b) Agregar Insumos; c) Eliminar Producto

En relación a la evaluación del prototipo se realizaron diversas pruebas que permitieron hacer una evaluación del mismo a fin de detectar y depurar errores que pudieran surgir en el desarrollo del módulo de "autogestión de compras" al cual se aplicó la técnica de transacciones concurrentes. Se aceptó el prototipo siendo implementado el módulo de "autogestión de compra" con la primera iteración y las primeras funcionalidades.

En una segunda iteración se agregaron las funcionalidades restantes que surgieron de la interacción de los usuarios con el módulo mencionado. En esta iteración se implementó la técnica de concurrencia optimista basada en Validación al módulo de "autogestión de compra", para poder controlar los posibles conflictos transaccionales

que generen inconsistencias en la base de datos. Una técnica basada en bloqueos impulsaría una sobrecarga innecesaria del sistema.

En este sentido se implementaron tres fases bien definidas[1],[7]: i) Fase de lectura: donde se leen los productos seleccionados, sin actualizar la base de datos; ii) Fase de validación: al momento de ejecutar la funcionalidad de "Confirmar la Compra" se realizó una prueba de validación, para determinar la modificación de la base de datos en la operación escribir sin alterar ni vulnerar la secuencialidad; iii) Fase de escritura: al confirmarse la compra del producto y la transacción fue correcta en la fase de validación entonces las modificaciones son aplicadas a la base de datos, si la situación explicada anteriormente fuera contraria, se retrocede como si la transacción nunca se hubiese realizado.

Con la aplicación de la técnica mencionada anteriormente y teniendo en cuenta la naturaleza del módulo de autogestión desarrollado, se buscó que las transacciones se ejecuten presumiendo que todas fueron capaces de finalizar la ejecución y ser validadas al final.

3 Resultados

A los efectos de ilustrar la técnica objeto de estudio, en esta sección se describe la aplicación de la técnica de transacciones concurrentes optimistas al módulo "gestión de compras" (Fig. 4), teniendo en cuenta las consideraciones de crear el modelo Cliente Servidor y la forma de poder establecer la conexión a la base de datos, como así también la configuración de la misma para aceptar este tipo de transacciones.

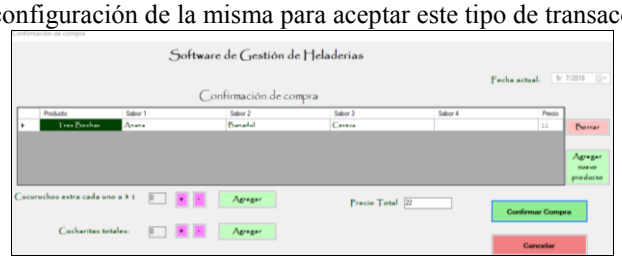


Fig. 4. Pantalla del módulo "Gestión de compras"

Las características del Servidor de Aplicación se observa en la Tabla 2 las cuales también son compartidas por el Servidor de base de datos, los cuales afines demostrativos conviven en el mismo host.

Tabla 2. Características del host servidor

Característica	Descripción
Procesador	AMD C50 DOBLE NÚCLEO 1.0 GHZ
Memoria Ram	DDR 2 4GB
Red Lan de Prueba	a) SSID: Arduino. b) Rango: 192.168.1.1 - 192.168.1.63 c) Asignación de IPs: Manual
Dirección IP	192.168.1.50/24
Sistema Operativo Nativo	Windows 7 32 bits Servipack 1
Nombre de Equipo	Usuario-PC

En la Tabla 3 se visualizan las características correspondientes a los Hosts donde se ejecutó la aplicación cliente para las pruebas de la técnica objeto de estudio. Para lo cual, se utilizaron equipos de características simples ya que al aplicación cliente en si no presenta mayores requerimientos de hardware.

Tabla 3. Características del host cliente

Característica	Descripción
Procesador	AMD ATLHON X II 255 3.01 GHZ
Memoria Ram	DDR 2 4GB
Red Lan de Prueba	a) SSID: Arduino. b). Rango: 192.168.1.1 - 192.168.1.63 c) Asignación de IPs: Manual
Dirección IP Equipo 1	192.168.1.2/24
Dirección IP Equipo 2	192.168.1.10/24
Sistema Operativo Nativo	Windows 7 32 bits Servipack 1
Nombre de Equipo 1	Usuario-PC-1
Nombre de Equipo 2	Usuario-PC-2

Para poder configurar la conexión remota al Servidor SQL Server, se estableció la habilitación de conexión remota al mismo y el modo de Autenticación Mixta con un usuario y contraseña en SQL Server. Posteriormente se establecieron los elementos que componen la cadena de conexión utilizada en la aplicación para enlazar la base de datos con la misma, así como el usuario de SQL Server para el acceso remoto, como ser: i) el Data Source; ii) el InitialCatalog; iii) la Seguridad iV) Mixed Authentication (SQL Authentication) y v) el Network Address.

Establecidos estos parámetros se procedió a habilitar la base de datos con aislamiento Snapshot [13].

Se utilizó este tipo de aislamiento porque representa el uso de la técnica de concurrencia optimista sin utilización de bloqueos. Para ello verifica que la transacción tenga una versión coherente de aquellos datos con la que la misma se inicia. Es como si se tomará una instantánea de cada transacción.

Por lo expuesto se logra que no se bloqueen las escrituras de datos de otras transacciones y tampoco son bloqueadas las lecturas. Para poder utilizarlo se debió elegir la opción ALLOW_SNAPSHOT_ISOLATION establecida en modo ON. Si se realiza una operación UPDATE en una tabla y al instante se emite una operación de selección para esa misma tabla los datos modificados se incluyen en el conjunto de resultados obtenidos.

Para comprender el nivel de aislamiento, en la Fig. 5 se observa donde las versiones de filas actualizadas de cada transacción se mantienen en una tabla temporal. Cada transacción es marcada con un número único de secuencia, los cuales son registrados para cada versión de fila. La transacción trabaja con la versión de fila más reciente, la cual posee un número de secuencia anterior al número de secuencia propio de la transacción ejecutada, y omite cualquier versión de fila más nueva que haya sido creada con posterioridad a su comienzo.

A los efectos de aplicar la técnica de concurrencia optimista, se estableció un acceso restringido como se visualiza en la Fig. 6, basándose en conectividad por protocolo TCP/IP. Lo cual genera que cualquier modulo cliente que intente acceder a la base de datos no podrá hacerlo mientras el Servidor de Aplicación este inactivo.

Cuando este tome actividad se comunicará con el Servidor de base de datos, en este caso SQL Server el cual iniciará las conexiones pertinentes y ejecutará las peticiones de transacciones que le sean requeridas.

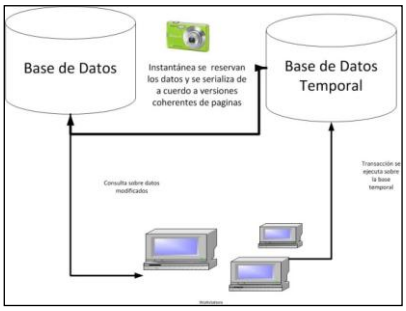


Fig. 5. Aislamiento Snapshot

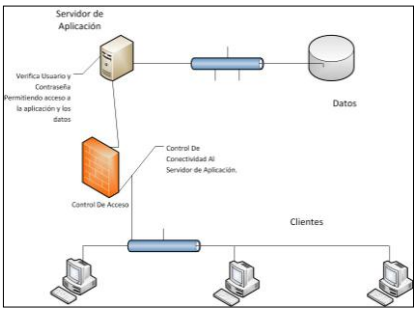
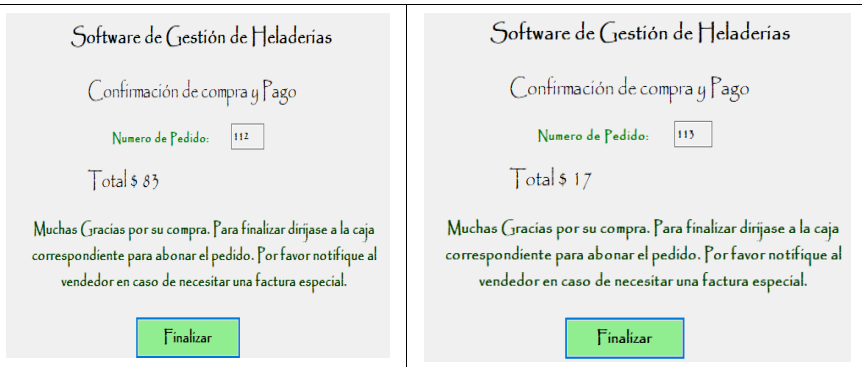


Fig. 6. Control de acceso Cliente Servidor

Una vez que se ingresó al sistema y establecida la conexión se procedió a ejecutar las pruebas de transacciones concurrentes, probando el camino exitoso, es decir, que dos transacciones se pueden ejecutar concurrentemente, superando las validaciones y confirmando ambas transacciones como puede observarse en la Fig. 7. En el caso que una transacción no supere la etapa de validación, la transacción queda anulada y las propiedades ACID se cumplen, como puede observarse en la Fig. 8.

Adicionalmente, se almacenaron en la base de datos información de las transacciones que han cumplimentado las validaciones y han sido confirmadas, como así también aquellas que no lo hicieron. En la Fig. 9 se puede observar a modo de ejemplo las transacciones efectuadas por fecha, disponiendo además, de las opciones de filtrar por dirección IP o por estado (completas o incompletas).



a) Primera terminal remota b) segunda terminal remota

Fig. 7. Mensajes de transacciones exitosas en las terminales.

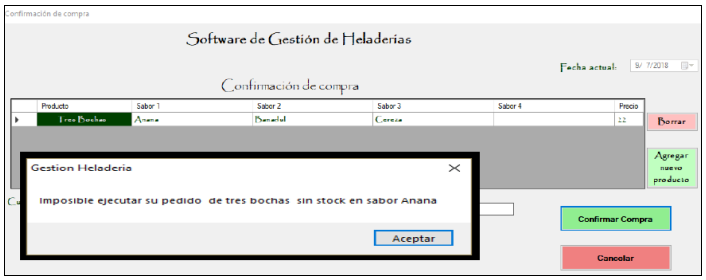


Fig. 8. Transacción fallida con rollback sin superar validación.

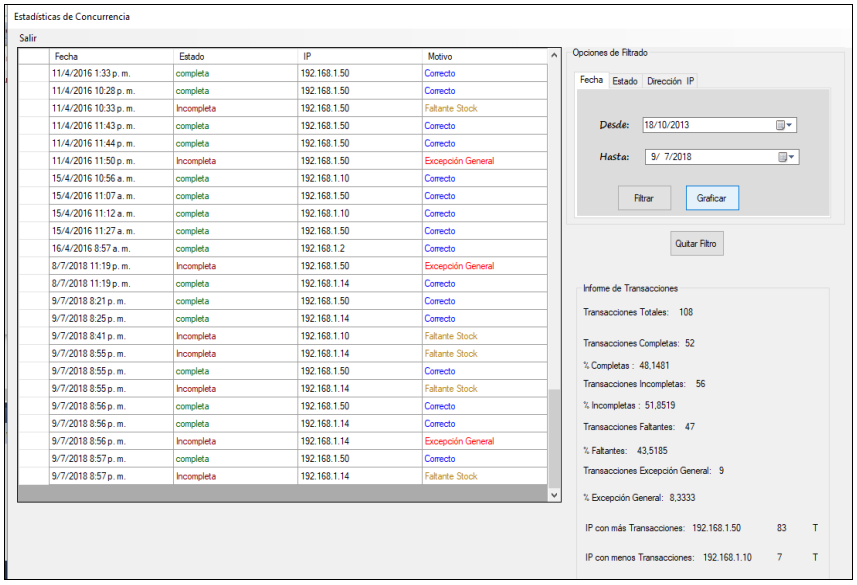


Fig. 9. Lista de transacciones filtradas por fecha.

4 Conclusiones

El estudio de las transacciones concurrentes en una aplicación que se implemente en una estructura cliente servidor, es de vital importancia, dado que mantener la coherencia y durabilidad de los datos que son consumidos desde un SGBD no es una tarea trivial. La elección de una técnica de manejo de concurrencia que permita la serialización de las transacciones y la coherencia de las operaciones realizadas sobre dichos datos, debe realizarse de acuerdo a la naturaleza del problema que se esté tratando de resolver.

En el software desarrollado la naturaleza misma del problema fue la lectura de múltiples datos por las distintas instancias del módulo de “autogestión de compra”, sin necesidad de hacer operaciones de actualización hasta la confirmación y facturación del pedido, lo cual conllevo a que al ser gran parte de las operaciones

aplicadas de lectura, los datos o las tuplas que conforman los registros en la base de datos no fueran necesariamente bloqueadas, sino que simplemente se permitiera su lectura y tratamiento y luego de un proceso de validación las operaciones que modificaran dichos datos fueran aceptadas o si hubiera algún impedimento dichas transacciones se anularan y se volviera la base de datos su estado original. Esto es en definitiva lo que se llama una Técnica de Concurrencia Optimista, siendo lo que se aplicó en este trabajo. Lo que se logra es que las conexiones entre cliente y servidor sean mucho más fluidas con una mayor atención de clientes en menor tiempo, siendo innecesaria una conexión persistente con el servidor, como así también el menor consumo de recursos del mismo.

Lo expuesto, puede observarse en los resultados obtenidos al mostrar como una operación que fue aceptada mucho después que otra, pero que por su tratamiento de los datos que capturó en forma temprana le correspondió insertarse antes, a pesar de que otra transacción haya finalizado primero, lo cual significa que retrasa las actualizaciones permitiendo mantener la coherencia de los datos contenidos en la base de datos (Fig. 9).

Referencias

1. Silberschatz, A., Korth, F., Sudarshan, S.: Fundamentos de Base de Datos, Madrid: Mcgraw-Hill/Interamericana De España, S. A. U., 2002.
2. Simultaneidad optimista. https://msdn.microsoft.com/es-es/library/aa0416cz%28v=vs.110%29.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-4.
3. Neira Cisterna, M.: Métodos de Optimización de Consultas para el Lenguaje SQL, https://docplayer.es/13940333-Metodos-de-optimizacion-de-consultas-para-el-lenguaje-sql.html
4. SQL Server Transaction Locking and Row Versioning Guide, https://technet.microsoft.com/en-us/library/jj856598(v=sql.110).aspx
5. Frati, F. E.: Evaluación de técnicas de detección de errores en programas concurrentes. Trabajo Final presentado para obtener el grado de GRID, Facultad de Informática Universidad Nacional de La Plata, La Plata Pcia de Bs As Argentina, 2014.
6. Buckle, C. E., Urriza, J. M., Paez, F.: Transitando hacia las Bases de Datos de tiempo real, 39JAIIO - JII, Universidad Nacional de La Patagonia San Juan Bosco - Puerto Madryn, Argentina, 2010.
7. Martti, L., Dervos, D. A., Silpiö, K.: SQL Transactions Teoría y ejercicios en la práctica, Universidad de Málaga: DBTech VET, 2014.
8. Enterprise Architect , http://www.sparxsystems.com/products/ea/7.5/
9. Start Uml, http://staruml.io/faq
10. Microsoft Visual Studio, https://www.microsoft.com/en-us/download/details.aspx?id=23507
11. Microsoft SQL Server 2005 Express Edition, http://www.microsoft.com/es-ar/download/details.aspx?id=21844
12. Somerville, I.: Ingeniería del Software, Madrid: Pearson Educación S.A., 2005.
13. Set Transaction Isolation Level, https://msdn.microsoft.com/es-es/library/ms173763%28v=sql.120%29.aspx