EVALUASI PERFORMA MULTITHREADING PADA PENGOLAHAN LAPORAN PENJUALAN DENGAN APLIKASI DESKTOP

Rosalia Hadi[1)]
Program Studi Sistem Informasi[1)]
Fakultas Informatika dan Komputer, Institut Teknologi dan Bisnis STIKOM Bali[1)]
rosa@stikom-bali.ac.id[(1)]

ABSTRACT

In this day and age, computers have developed functionality in various ways such as recording sales transactions. Multithreading is a small form of parallel processing that allows a job to be done simultaneously. A sales report is one of the mandatory reports on a Point of Sale system. When the system makes a sales report, the system will query the database and then the system will repeat every record generated by the query to perform the printing process. This if done with a conventional method that uses single-thread will take a very long time. The author conducts research in the form of a design that will evaluate the performance of the multithreading method that will be applied in the preparation of financial statements in a point-of-sale system. There are two levels of the threading hierarchy used in the multi-threading method, one main-threading and the rest is child-threading. Multithreading has the advantage of better process speed but has the disadvantage of higher memory consumption.
Keywords*: pararel processing, multithreading, point of sale*

ABSTRAK

Pada jaman sekarang, komputer sudah berkembang fungsionalitasnya dalam berbagai hal seperti pencatatan transaksi penjualan. Multithreading merupakan bentuk kecil dari sebuah pararel processing yang memungkinkan sebuah pekerjaan bisa dilakukan secara bersamaan. Laporan penjualan merupakan salah satu laporan yang wajib ada pada sebuah sistem Point of Sale. Pada saat sistem membuat laporan penjualan, sistem akan melakukan query pada database yang kemudian sistem akan melakukan perulangan pada setiap record yang dihasilkan oleh query untuk melakukan proses pencetakan. Hal ini jika dilakukan dengan metode konvesional yang menggunakan singlethread akan menghabiskan waktu yang sangat lama. Penulis melakukan penelitian berupa rancangan yang akan melakukan evaluasi performa pada metode multithreading yang akan diterapkan dalam pembuatan laporan keuangan pada sistem point-of-sale. Terdapat dua level hirarki threading yang digunakan pada metode multi-threading yaitu satu main-threading dan sisanya adalah child-threading. Multithreading memiliki keuntungan dari sisi kecepatan proses yang lebih baik, namun memiliki kekurangan pada konsumsi memori yang lebih tinggi.
Kata Kunci : pararel processing, multithreading, point of sale

PENDAHULUAN

Pada jaman sekarang, computer sudah tidak lagi sebagai sebuah kebutuhan rekreasi (Underwood, 2015). Namun komputer juga telah berkembang fungsionalitas dalam berbagai hal seperti penggunaan pada dunia medis (Bar et al., 2015), hingga pencatatan transaksi penjualan (Lopez et al., 2018). Hal ini dikarenakan komputer pada saat ini memiliki kemampuan yang cepat dengan harga yang terjangkau.

Teknologi komputer mengalami perkembangan pesat dan tidak hanya dari sektor perangkat keras hingga arsitektur perangkat lunak (Dingsøyr et al., 2012). Munculnya teknologi-teknologi baru dalam arsitektur processor dimulai dari *pipelining* yang memungkinkan proses *fetch* dan *execute* dilakukan secara bersamaan (Jouppi et al., 2017). Hingga saat ini sudah ada teknologi *multithreading* yang memberikan kemampuan pada komputer untuk melakukan tugas secara simultan dan bersamaan (Trott and Olson, 2010).

Multithreading merupakan bentuk kecil dari sebuah *pararel processing* yang memungkinkan sebuah pekerjaan bisa dilakukan secara simultan dan bersamaan (Shen et al., 2016). Pararel processing memungkinkan sebuah perkerjaan dilakukan secara bersamaan dan menurunkan waktu eksekusi yang diperlukan. Antar proses/thread pada pararel processing berdiri sendiri tanpa mempengaruhi proses-proses yang lainnya.

Komputer dalam penggunaan sebagai pencatatan penjual atau Point of Sale System bertugas dalam mencatat proses bisnis yang terjadi pada sebuah toko (Buttenheim et al., 2012). Sistem melakukan pencatatan mulai dari proses datangnya barang, hingga proses penjualan barang ke customer. Tentunya semua proses bisnis yang terjadi akan menghasilkan sebuah laporan yang dapat dibaca dan diolah oleh seorang akuntan pada sebuah toko.

Data penjualan yang biasanya tercatat pada sistem porsinya selalu lebih tinggi jumlah *record*-nya dibandingkan dengan data pembelian. Hal ini dikarenakan biasanya toko melakukan pembelian sekali dalam jumlah yang besar, sehingga sistem akan mencatat pembelian hanya sekali dengan kuantitas barang yang banyak. Berbeda dengan data penjualan yang memiliki jumlah *record* lebih tinggi dari data pembelian. Asumsi dimana toko melakukan pembelian barang sebanyak 100 dalam satu kali pembelian, dimana 100 barang tersebut dijual secara eceran sebanyak 1 barang per transaksi. Asumsi seperti ini menggambarkan bahwa proses bisnis pembelian barang hanya dilakukan sekali dan dicatat sekali, namun penjualan dapat dilakukan lebih dari satu kali dan dicatat sebanyak penjualan yang terjadi.

Laporan penjualan merupakan salah satu laporan yang wajib ada pada sebuah sistem Point of Sale. Pada saat sistem membuat laporan penjualan, sistem akan melakukan *query* pada *database* yang kemudian sistem akan melakukan perulangan pada setiap *record* yang dihasilkan oleh *query* untuk melakukan proses pencetakan, baik *hardcopy* maupun *softcopy*. Hal ini jika dilakukan dengan metode konvesional yang menggunakan *singlethread* akan menghabiskan waktu yang sangat lama.

Permasalahannya adalah ketika program melakukan ektraksi data dari database dengan jumlah data yang banyak. Asumsikan jika data yang tersedia adalah sejumlah 10.000 dan untuk memproses satu data membutuhkan waktu sekitar 1 detik. Maka untuk mencetak data dibutuhkan sekitar 10.000 detik atau sekitar 2.78 jam untuk menghasilkan laporan jika dilakukan secara *singlethread* / metode konvensional. Hal ini menyebabkan program tidak dapat berjalan secara responsive dan tidak dapat diandalkan ketika membutuhkan data dalam hitungan kurang dari 1 jam. Terlebih lagi, munculnya efek samping seperti program mengalami hang/freeze saat membuat laporan tersebut karena pekerjaan dilakukan di *main thread*.

Berdasarkan masalah tersebut, penulis merancang sebuah penelitian yang akan melakukan evaluasi performa pada metode *multithreading* yang akan diterapkan dalam pembuatan laporan keuangan pada sistem point-of-sale. Diharapkan dengan evaluasi performa ini, dapat menghasilkan desain dan algoritma yang lebih baik pada proses

pembuatan laporan penjualan berdasarkan waktu eksekusi yang dibutuhkan dalam menghasilkan satu laporan. Pada uji performanya akan memanfaatkan C# untuk bahasa pemrogramannya, .NET Framework dan MySQL dalam penggunaan databasenya. Penelitian ini diharapkan dapat mengembangkan program yang lebih responsive dengan peningkatan performa yang lebih baik.

TINJAUAN PUSTAKA

Multithreading

Multithreading merupakan kemampuan dari program maupun sistem operasi untuk menjalankan beberapa tugas atau pekerjaan secara bersamaan tanpa menggangu kinerja dari *thread* yang lainnya (Trott and Olson, 2010). *Thread* sendiri merupakan potongan-potongan kecil dari sebuah program yang berjalan yang berasal dari fungsi maupun prosedur yang terdapat dalam suatu program.

Kehadiran *multithreading* semakin menambah *utilisasi* dan multi-core processor yang sudah semakin banyak berada di pasaran. Hal ini disebabkan program memiliki kemampuan untuk menjalankan potongan kecil tugas atau pekerjaan secara bersamaan dengan menyebarnya di seluruh *core* processor yang tersedia. Misalkan sebuah *game* yang melakukan proses *multithreading* untuk melakukan *render* yang terpisah. Satu *thread*, misalkan, bertugas untuk melakukan *render* objek 3D berupa *terrain* yang ditempatkan pada *core* processor pertama dan *thread* lainnya pada core yang lainnya juga.

Thread yang berjalan memiliki kontrol yang baik dalam melakukan tugasnya. Misalkan jika sebuah proses dibatalkan oleh pengguna, maka ada du acara dalam membatalkan proses dalam *thread*. *Asynchronous cancellation* merupakan pembatalan *thread* dengan menghentikan seketika dan *deferred cancellation* merupakan pembatalan *thread* dengan memberikan waktu kepada thread untuk menyelesaikan tugasnya terlebih dahulu.

Multithreading merupakan bentuk kecil dari sebuah *pararel processing* yang memungkinkan sebuah pekerjaan bisa dilakukan secara simultan dan bersamaan (Shen et al., 2016). Pararel processing bekerja dengan cara membagi sebuah masalah menjadi masalah-masalah kecil. Masing-masing masalah kecil ini kemudian di-*assign* dengan masing-masing pekerja dan dikerjakan secara bersamaan. Hal ini dapat mengurangi waktu eksekusi program sehingga program lebih responsive.

Multithreading telah digunakan secara praktikal sebelumnya pada beberapa penelitian salah satunya yang dilakukan oleh (Low et al., 2011). Pada penelitian tersebut *multithreading* dimanfaatkan untuk melakukan *insertion* sejumlah data yang besar kedalam *database*. Hasilnya menunjukan bahwa *multithreading* memberikan dampak positif bagi sistem dengan meningkatkan performa dan respons time.

.NET Framework

.NET Framework merupakan sebuah *framework* atau kerangka kerja yang merupakan kumpulan dari Application Programming Interface (APIs) yang memudahkan seorang programmer dalam membangun sebuah aplikasi (Jailia et al., 2016). Segala fungsi dan prosedur yang telah disediakan oleh .NET Framework menjadikan seorang programmer tidak perlu melakukan pembangunan sebuah program atau aplikasi dari *scratch*.

.NET Framework juga menyediakan *runtime environment* untuk sebuah aplikasi yang dibangun menggunakan framework ini yang dimana disebut dengan Common Language Runtime (CLR). CLR bertindak sebagai host dari sebuah aplikasi yang berfungsi sebagai penghubung antara program dengan system operasi. CLR juga yang bertindak sebagai manajer sumber daya seperti penanganan alokasi dari alamat memory hingga menampilkan hasi keluaran ke layar monitor.

Framework ini hampir digunakan oleh seluruh aplikasi yang berjalan di system operasi Windows dan juga digunakan dijadikan acuan dalam pengajaran programming seperti ditulis oleh (Solis and Schrotenboer, 2018) yang menjelaskan betapa mudahnya penggunaan .NET Framework dalam pembangunan sebuah aplikasi.

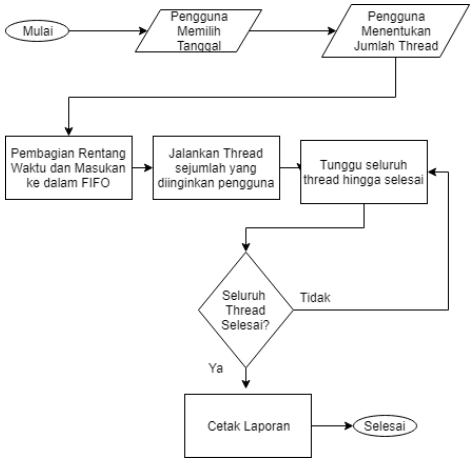
Point of Sales

Point of Sale (POS) merupakan tempat dan waktu dimana sebuah proses transaksi jual beli terjadi (Hines and Youssef, 2018). Proses ini mencakup dari perhitungan kuantitas barang/jasa yang dijual beserta harganya, mencetak bukti pembayaran, hingga pemilihan metode pembayaran yang diinginkan oleh customer. Kemudian seiring meningkatnya kebutuhan data, POS berkembang dan tidak hanya menangani proses transaksi penjualan, namun juga menangani pencatatan stok, harga pokok, hingga keuntungan yang didapatkan dari hasil penjualan.

METODE PENELITIAN

Model Konseptual Penelitian

Secara umum, alur algoritma yang akan diteliti diawali dengan pengguna memilih rentang tanggal laporan yang dikehendaki. Selanjutnya pengguna menentukan jumlah *thread* yang akan digunakan dalam pembuatan laporan. Aplikasi selanjutnya membagi rentan waktu kedalam bagian-bagian kecil kemudian memasukan hasil pembagian tersebut kedalam global variable penyimpanan dengan konsep FIFO (First In First Out). Kemudian aplikasi menjalankan *thread* dan *thread* akan mengambil bagian pekerjaan dari global variable FIFO. Aplikasi kemudian menunggu semua *thread* selesai berjalan dan setelah semua *thread* selesai berjalan, data ditampilkan oleh aplikasi. Gambar 1 merupakan rancangan algoritma secara umum.

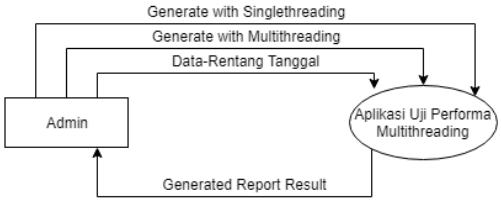


Gambar 1. Diagram alir algoritma secara umum

Diagram Konteks

Aplikasi uji performa ini hanya akan memiliki dua proses utama saat digunakan. Pengguna dapat memilih apakah akan menggunakan metode *pararel-processing* atau menggunakan metode *serial-processing*. Gambar 2 merupakan diagram konteks dari aplikasi uji performa yang akan dibangun.

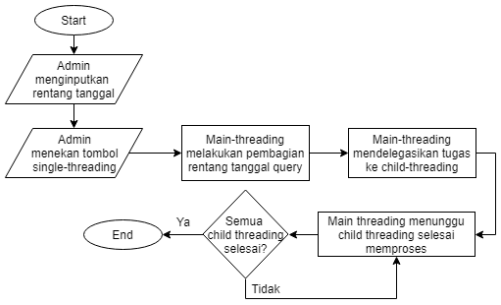
Terdapat 3 nilai masukan yang dapat diterima oleh aplikasi dari admin. Pertama adalah rentang waktu tanggal yang dipilih oleh admin, kedua adalah metode yang dipilih oleh admin apakah memilih metode *single-threading* atau metode *multi-threading*. Kemudian diakhir aplikasi akan memberikan keluaran berupa laporan yang sudah selesai diproses, baik menggunakan metode *single-threading* maupun *multi-threading*.



Gambar 2. Konteks Diagram Aplikasi Uji Performa

Multithreading Flowchart

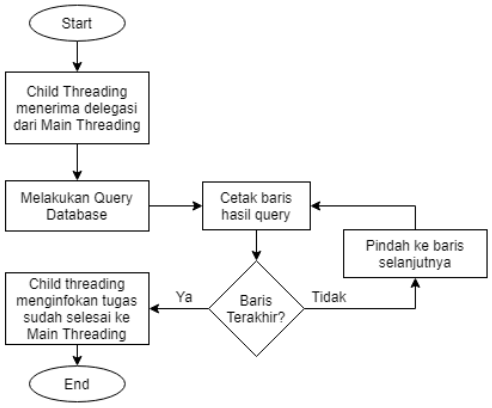
Pada metode *multi-threading* terdapat dua hirarki threading, yaitu yang bertindak sebagai *main threading* dan sisanya sebagai *child threading*. *Main threading* berfungsi untuk mengatur dan mengkordinasikan seluruh *child-threading*. *Child threading* nantinya yang akan melakukan processing data baik *query* maupun mencetak ke layar.



Gambar 3. Proses yang Dirancang pada Main-Threading

Pada Gambar 3 yang merupakan aliran proses yang terjadi pada *main threading*. Proses yang terjadi pada *main threading* adalah yang pertama menerima masuk berupa rentang tanggal yang dipilih oleh admin. Kemudian *main threading* akan membagi rentang waktu dalam beberapa bagian. Masing-masing bagian nantinya akan didelegasikan ke masing-masing *child threading*. Setelah semua bagian telah didelegasikan kepada *child-threading*, *main-threading* akan menunggu semua *child threading* usai menjalankan tugasnya.

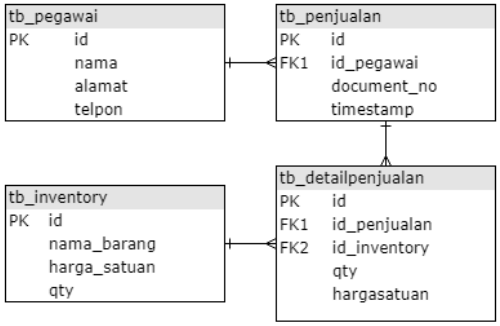
Pada Gambar 4 merupakan proses yang terjadi di *child-threading* dimana pada *child-threading* akan menerima delegasi tugas dari *main-threading*. Proses yang terjadi pada *child-threading* adalah melakukan *query* sesuai dengan delegasi tanggal yang diberikan oleh *main-threading*. Kemudian data yang dihasilkan dari *query* akan dicetak dan ditampilkan ke admin. Setelah usai melakukan tugasnya, *child-threading* akan menginformasikan kepada *main-threading* bahwa tugasnya telah selesai.



Gambar 4. Proses yang terjadi pada *child threading*.

Konseptual Database

Pada aplikasi uji performa *multithreading* ini, semua data penjualan yang digunakan bersifat *dummy* data. Namun meskipun *dummy data*, data tersebut tetap disimpan dalam sebuah *database*. *Database* yang digunakan hanya memiliki 4 tabel yaitu table penjualan, detail penjualan, master inventori dan master pegawai. Gambar 5.5 merupakan konseptual database yang digunakan dalam perancangan aplikasi uji performa ini.



Gambar 5. Konsptual Database yang Digunakan.

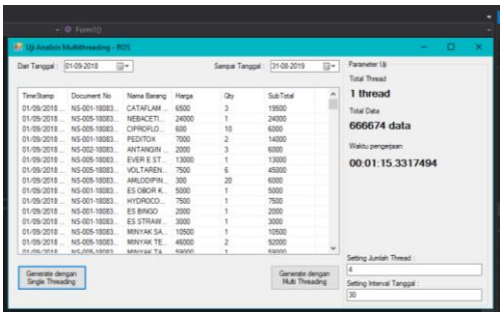
IMPLEMENTASI DAN PENGUJIAN

Pengujian dilakukan dengan melakukan pembuatan laporan untuk beberapa periode laporan. Terdapat beberapa periode laporan yang digunakan yaitu jangka 12 bulan, 6 bulan, 3 bulan, dan 1 bulan. Tabel 1 merupakan jangka pengujian yang digunakan.

Tabel 1. Variabel Pengujian

No	Jangka Laporan	Rentang Tanggal		Total data
		dari Tanggal	sampai Tanggal	
1	1 Bulan	01-01-2019	31-01-2019	51.254
2	3 Bulan	01-01-2019	31-03-2019	159.062
3	6 Bulan	01-01-2019	30-06-2019	344.495
4	12 Bulan	01-09-2018	31-08-2019	666.674

Selama pengujian, peneliti melakukan pengawasan terhadap penggunaan memory yang digunakan dengan bantuan alat debugging dari Visual Studio. Kemudian melihat titik tertinggi capaian penggunaan memory seperti terdapat pada gambar 6.



Gambar 6. Simulasi pengujian yang dilakukan

Pengujian dilakukan dengan menggunakan 1 thread untuk mewakili proses single thread (*serial processing*), dan, 2 dan 4 thread untuk mewakili proses *multithreading* (*pararel processing*). Berdasarkan pengujian yang dilakukan terhadap performa masing-masing jumlah thread yang digunakan dengan kecepatan dan konsumsi memori yang dibutuhkan dituangkan dalam tabel 2.

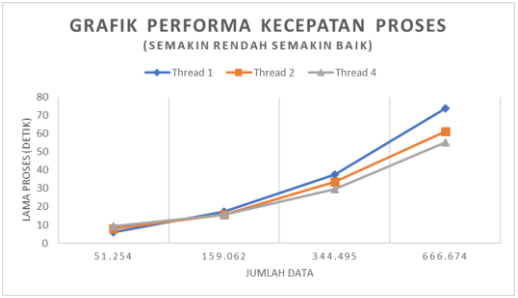
Tabel 2. Rangkuman Pengujian Performa

Jumlah Child Thread	Jangka Pengujian (bulan)	Lama Proses (detik)
1	1	6,035
	3	17,188
	6	37,595
	12	73,997
2	1	8,124
	3	15,645
	6	33,613
	12	61,213
4	1	9,213
	3	15,557
	6	29,546
	12	55,214

Berdasarkan pengujian yang terdapat pada tabel 2 terlihat bahwa terjadi peningkatan kecepatan proses yang terjadi. Namun terlihat pada jangka pengujian 1 bulan, *multithreading* justru lebih lama dibandingkan dengan *singlethreading*. Hal ini disebabkan karena *multithreading* memerlukan waktu untuk melakukan inisialisasi dan delegasi SQL kepada *child threading* terlebih lagi jangka waktu yang dipilih (1 bulan) tidak terlalu panjang.

Pada jangka pengujian 3 bulan keatas, *multithreading* menunjukan performanya dimana kecepatan proses terjadi peningkatan. Namun selisih antara 2 childthread dan 4 childthread tidak terlalu signifikan dan hanya selisih 88 milisecond. Kemampuan 4 childthread terjadi peningkatan kecepatan dibandingankan 2 childthread dengan selisih 4,067 detik pada jangka pengujian 6 bulan dan 5,999 detik pada jangka pengujian 12 bulan.

Berdasarkan hasil analisis tersebut, jumlah childthread yang banyak memberikan keuntungan yang lebih besar jika diaplikasikan pada jumlah data yang semakin besar dan akan memberikan dampak penurunan performa jika digunakan pada jumlah data yang tidak terlalu besar. Gambar 7 merupakan grafik performa dari pengujian kecepatan *multithreading*.



Gambar 7. Grafik Performa Kecepatan Proses

SIMPULAN

Berdasarkan penelitian yang dilakukan, secara umum *multithreading* (*pararel* processing) memiliki kecepatan proses yang lebih baik dibandingkan dengan *singlethreading*. Hal ini disebabkan karena pekerjaan yang dilakukan dibagi dalam bagian-bagian kecil dan dikerjakan secara simultan. Meskipun demikian, performa kecepatan proses *multithreading* hanya terlihat signifikan ketika jumlah data yang diproses semakin banyak. Justru ketika jumlah data yang diproses sedikit, *multithreading* cenderung lebih lambat dibandingkan *singlethreading*.

Berdasarkan pengujian hal tersebut, dapat disimpulkan bahwa *multithreading* sangat

cocok untuk diterapkan pada pemrosesan data dalam jumlah besar karena sifat simultan-nya yang dapat bekerja secara mandiri. Namun perlu diperhatikan juga konsumsi memori dari *multithreading* sendiri karena semakin banyak jumlah *childthreading* semakin cepat waktu proses yang diperlukan dan semakin tinggi pula konsumsi memori yang diperlukan.

DAFTAR PUSTAKA

[1] Bar, Y., Diamant, I., Wolf, L., Lieberman, S., Konen, E. & Greenspan, H. Chest Pathology Detection Using Deep Learning With Non-Medical Training. 2015 Ieee 12th International Symposium On Biomedical Imaging (Isbi), 2015. Ieee, 294-297.

[2] Buttenheim, A. M., Havassy, J., Fang, M., Glyn, J., Karpyn, A. E. J. J. O. T. A. O. N. & Dietetics 2012. Increasing Supplemental Nutrition Assistance Program/Electronic Benefits Transfer Sales At Farmers' Markets With Vendor-Operated Wireless Point-Of-Sale Terminals. 112**,** 636-641.

[3] Dingsøyr, T., Nerur, S., Balijepally, V. & Moe, N. B. 2012. A Decade Of Agile Methodologies: Towards Explaining Agile Software Development. Elsevier.

[4] Hines, C. & Youssef, A. Machine Learning Applied To Point-Of-Sale Fraud Detection. International Conference On Machine Learning And Data Mining In Pattern Recognition, 2018. Springer, 283-295.

[5] Jailia, M., Kumar, A., Agarwal, M. & Sinha, I. Behavior Of Mvc (Model View Controller) Based Web Application Developed In Php And. Net Framework. 2016 International Conference On Ict In Business Industry & Government (Ictbig), 2016. Ieee, 1-5.

[6] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N. & Borchers, A. In-Datacenter Performance Analysis Of A Tensor Processing Unit. 2017 Acm/Ieee 44th Annual International Symposium On Computer Architecture (Isca), 2017. Ieee, 1-12.

[7] Lopez, M. E., Heine, W. A., Mckinstray, K. D. & Keren, G. 2018. System, Method, And Computer Program For Dynamically Reconciling A Distributor Invoice With A Retailer Receiving Invoice For Products Sold Under Multiple Upcs And In Multiple Quantity Units. Google Patents.

[8] Low, B. W., Ooi, B. Y. & Wong, C. S. Scalability Of Database Bulk Insertion With Multi-Threading. International Conference On Software Engineering And Computer Systems, 2011. Springer, 151-162.

[9] Shen, H., Wei, X., Liu, H., Liu, Y. & Zheng, K. Design And Implementation Of An Lte System With Multi-Thread Parallel Processing On Openairinterface Platform. 2016 Ieee 84th Vehicular Technology Conference (Vtc-Fall), 2016. Ieee, 1-5.

[10] Solis, D. & Schrotenboer, C. 2018. C# And. Net Core. *Illustrated C# 7*. Springer.

[11] Trott, O. & Olson, A. J. J. J. O. C. C. 2010. Autodock Vina: Improving The Speed And Accuracy Of Docking With A New Scoring Function, Efficient Optimization, And Multithreading. 31**,** 455-461.

[12] Underwood, C. J. S. 2015. Snapshots Issue 8: Playing Computer Games For Recreation. 8**,** 1.