OPTIMALISASI ALGORITMA INSERT MEMANFAATKAN MEMORY PRIMER DAN BULK INSERT STUDI KASUS: PENGEMBANGAN SISTEM PENGELOHAN DATA PERPAJAKAN PNS

Antonius Bima Murti Wijaya[1)], Irya Wisnubhadra[2)], Benyamin L. Sinaga[3)]
[1 2 3)]*Program Pasca Sarjana Teknik Informatika*
Universitas Atma Jaya Yogyakarta
Jl babarsari no 22 Tambakbayan 55281, Sleman, Yogyakarta
bimamurti@gmail.com[1)], irya@mail.uajy.ac.id[2)], blsinaga@mail.uajy.ac.id[3)]

Abstrak
Pada setiap daerah/kabupaten ada kurang lebih 8000 orang yang bekerja sebagai pegawai negri sipil(PNS). Dengan jumlah orang yang tidak sedikit itu, maka diperlukan suatu sistem pengelolaan data yang berkualitas, baik itu dari sisi kecepatan maupun ketepatannya. Studi kasus ini memiliki ciri yang khas yaitu dengan pengelolaan data yang banyak dalam satu waktu, seperti pada fungsi untuk memasukkan data yang dalam satu proses memasukkan kurang lebih 100.000 data gaji.Penggunaan algoritma insert ke basis data biasa akan memakan waktu yang lama, dikarenakan faktor interaksi dengan memori sekunder yang tinggi. Dalam mengembangkan sistem ini akan digunakan teknologi bulk insert dalam penyesuaian algoritma untuk proses insert data baik yang menggunakan primary key maupun tidak. Penelitian ini mengarahkan proses pengolahan datanya dalam memory primer agar tidak terhambat kecepatan membaca file pada memori sekunder. Sumber data sistem ini berupa data excel dalam bentuk tabel mentah dengan format bermacam-macam, sehingga proses optimalisasi juga dilakukan untuk algoritma penyesuaian format. Dari hasil penelitian waktu eksekusi dapat dipangkas secara signifikan, dan penggunaan RAM juga menjadi bertambah.

Kata Kunci: *Optimalisasi, Memori Primer, Bulk Insert.*

1. PENDAHULUAN

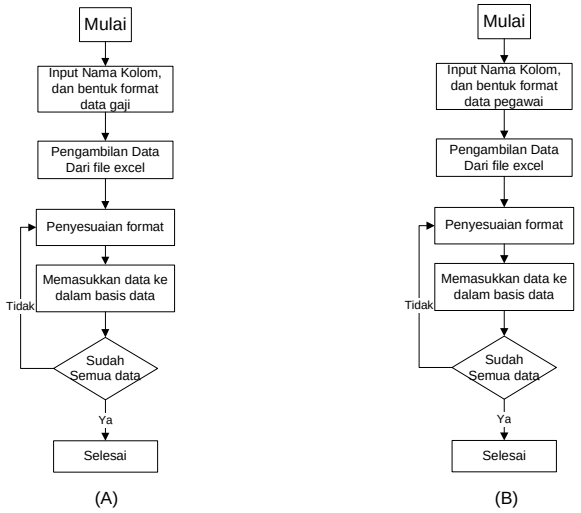
Berdasarkan survei Situs CIA (Central Intellegent Agency) mengenai perbandingan populasi jumlah penduduk didunia, Indonesia merupakan negara dengan jumlah penduduk terbesar ke 4 di dunia dengan jumlah penduduk berkisar 248.216.193 jiwa, hal ini pun mempengaruhi jumlah karyawan pemerintahaan pada setiap daerahnya. Pada setiap daerah/kabupaten ada kurang lebih 8000 orang yang bekerja sebagai pegawai negri sipil (PNS). Dengan jumlah pegawai yang tidak sedikit itu maka diperlukan suatu sistem pengelolaan yang berkualitas, baik itu dari sisi kecepatan maupun ketepatannya. Sistem pengolahan data perpajakan akan mengolah data PNS pada setiap tahun yang mencangkup 12 bulan ditambah 1 gaji bulan ke 13. Jadi jika dijumlahkan dengan perkiraan PNS dikabupaten data yang harus diolah mencapai kira kira 13 x 8000 data atau 104000 data. Data tersebut untuk Kabupaten kecil namun untuk data ibukota kabupaten jumlah PNS bisa mencapai 11000 data perbulannya atau 143000 data untuk 13 bulan. Sebagai contoh pada data gaji kabupaten yang memiliki data gaji sebanyak 198096 dan membutuhkan waktu sekitar 35 menit untuk memasukkan data gaji dan 13 menit untuk memasukkan data gaji pegawainya. Karena jumlah data yang relatif besar diperlukan suatu penanganan kusus untuk meningkatkan kecepatan pemrosesan data untuk jumlah data sebesar itu. Hal ini bertambah kompleks karena adanya perbedaan format sumber data yang berupa *file excel*. Sebagai contoh pada data gaji yang terpisah kedalam 12 file perbulan dengan nama kolom nip adalah f_nip, sementara pada kabupaten lain, data gaji dipisah ke dalam 2 bagian satuan kerja dan masing masing satuan kerja dibagi kedalam 12 bulan jadi ada 24 file excel, dengan nama kolom nip adalah pnip dan tidak memiliki tabel jenis kelamin.

Studi kasus ini memiliki ciri yang khas yaitu dengan pengelolaan data yang banyak dalam satu waktu. kecepatan dan ketepatan suatu algoritma dan logika dalam melakukan pemrograman suatu sistem sangatlah penting terutama untuk sistem yang menuntut kedua hal tersebut. Apalagi pada jaman sekarang dimana kecepatan lebih dituntut untuk meningkatkan produktifitas. Peningkatan kecepatan pada suatu sistem dapat dilakukan melalui 2 cara yaitu melalui sisi perangkat kerasnya maupun dari sisi perangkat lunaknya. Dalam studi kasus ini kecepatan pemrosesan adalah hal yang penting, karena adanya proses koreksi yang memungkinkan suatu proses dalam sistem harus diulang. Hal ini bisa diulang mulai dari hal yang salah atau diulang dari awal proses. Jika proses lambat maka hal yang terjadi adalah akan banyak waktu terbuang untuk mengulang proses yang lama.

Oleh karena itu untuk proses *insert*nya datanya akan menggunakan *bulk insert* yang disesuaikan dengan kasus sumber data yang *multi format* dan bentuk tabel yang memiliki *primary key* maupun yang tidak memiliki primary key. Untuk kasus algoritma-algoritma nya akan dilakukan pemodelan fungsi waktu secara matematis

bagi fungsi-fungsi yang menyebabkan program berjalan lambat, yang difokuskan pada algoritma input data gaji dan data pegawai tidak termasuk pengubahan data atas data gaji ke 13.

Berikut merupakan gambaran umum proses input data yang terjadi pada sistem pengolah data perpajakan:



Gambar 1. (A) proses memasukkan data gaji dan (B) proses memasukkan data pegawai

Dalam penelitian ini akan dilakukan proses pengujian kecepatan proses, dengan metode yang dikembangkan dengan metode yang dipakai sebelumnya. Selain itu juga pengujian daya prosesor dan penggunaan RAM.

2. TINJAUAN PUSTAKA

Optimalisasi adalah proses untuk memaksimalkan atau meminimalkan fungsi objektif yang diinginkan sementara memenuhi batasan-batasan yang berlaku. Algoritma optimalisasi dapat didefinisikan sebagai algoritma atau metode numerik untuk menemukan nilai x sedemikian hingga menghasilkan f(x) yang bernilai sekecil/sebesar mungkin untuk suatu fungsi f yang diberikan, yang mungkin disertai dengan beberapa batasan pada x(Suyanto,2010). Optimalisasi dalam suatu sistem bisa dilakukan dengan dua cara yang pertama adalah dengan meningkatkan kinerja dari sisi hardwarenya atau dengan meningkatkan kinerja dari sisi softwarenya(Jananto,2006). Langkah untuk melakukan optimalisasi bisa juga dilakukan dengan bantuan algoritma pihak ke 3 seperti yang dilakukan oleh Wang(2010), dimana ia membuat suatu algoritma untuk melakukan optimalisasi algoritma Powell direct dan Algoritma Genetik sederhana. Masalah optimalisasi lainnya juga dilakukan oleh Niknam dkk(2011) yang melakukan penelitian mengenai penggabungan dari fuzzy adaptive partikel *swarm* dan *differtial evolution* untuk mengkonfigurasi ulang sistem distribusi radial secara optimal. Dalam penilitian lainnya Abo-Hamad dkk(2011) melakukan review dari 100 lebih tulisan mengenai optimalisasi kususnya pada kasus supply chain. Dia menyatakan bahwa dalam melakukan klasifikasi teknik optimalisasi dalam rangka pengambilan keputusan untuk supply chain perlu dipertimbangkan mekanisme optimalisasi, jenis variabel keputusan dan ruang untuk pencarian. Dalam pemetaan yang dia buat, teknik optimalisasi dibagi menjadi 2 bagian yaitu metode pencarian langsung dan metode matematis.

Proses Optimalisasi dalam prakteknya bisa menggunakan model matematika salah satunya adalah dengan menggunakan linear programming. McAllister dkk(2008) melakukan penelitian mengenai penggunaan integer linear programming untuk mengatasi masalah optimasi pada komputasi biologi sehingga menghasilkan program yang lebih efisien. Linear programming juga digunakan untuk masalah multi obyektif dengan mengubah permasalahan multikriteria ke masalah global optimum(Maimos dkk,2009). Pada Perkembangannya teknik linear programming ini sering dikombinasikan dengan teknik lain seperti dengan menggunakan Algoritma Genetik atau partikel swarm(Hagh dan Galvani, 2009)(Cisty,2011). Menggabungkan ke dua buah algoritma ini bukan berarti mengubahnya namun lebih ke mengkombinasikannya, dimana kedua algoritma ini akan saling melengkapi untuk memecahkan masalah suatu kasus.

Optimalisasi-optimalisasi yang ada tidak hanya terjadi pada algoritma suatu program namun pada query pada basis data juga. Optimalisasi Query adalah suatu proses untuk menganalisa query untuk menentukan sumber-sumber apa saja yang digunakan oleh query tersebut dan apakah penggunaan dari sumber tersebut dapat dikurangi tanpa mengubah output. Atau bisa juga dikatakan bahwa optimalisasi query adalah sebuah prosedur

untuk meningkatkan strategi evaluasi dari suatu query untuk membuat evaluasi tersebut menjadi lebih efektif. Optimalisasi query mencakup beberapa teknik seperti transformasi query ke dalam bentuk logika yang sama, memilih jalan akses yang optimal dan mengoptimumkan penyimpanan data. Tujuan dari optimalisasi query adalah menemukan jalan akses yang termurah untuk meminimumkan total waktu pada saat proses sebuah query. Untuk mencapai tujuan tersebut, maka diperlukan *optimizer* untuk melakukan analisa query dan untuk melakukan pencarian jalan akses (Siallagan dkk, 2008). Jananto(2006), melakukan penelitian mengenai optimalisasi basis data dengan menggunakan teknik optimalisasi Rushmore. Algoritma genetik pun bisa digunakan untuk melakukan optimalisasi terhadap query. Penelitian pada jalur ini pernah dilakukan oleh Siallagan dkk(2008), dia membandingkan 2 buah algoritma genetik yaitu *M2S crossover* dan *CHUNK crossover*. Hasilnya M2S crossover memiliki hasil yang lebih optimal dibanding chunk crossover. Wahyu dkk(2008) juga melakukan pembangan teknik optimalisasi query antara cross product dengan subset query, dimana dari hasil percobaannya query dengan menggunakan subset query lebih cepat dari cross product ketika data yang diolah semakin banyak.

Penggunaan *cache* memory untuk menampung sementara hasil query dilakukan karena ada keadaaan mengirim query yang sama pada satu atau banyak pengguna pada waktu yang berbeda(Gour dkk, 2010) (Selvara dkk, 2006). Suatu teknik optimalisasi untuk basis data tidak seterusnya bisa berperforma baik pada seluruh lingkungan. Jadi lebih baik mengembangkan teknik adaptif untuk suatu konteks dari pada mengembangakan secara umum(Deshpande dkk, 2007)(Mahajan dkk, 2010).

Dalam penelitian ini akan memanfaatkan teknologi dari Bulk Insert yang sudah ada pada sistem manajemen basis data skala besar seperti Sql server dan Oracle. Berdasarkan buku *Maintaining a Microsoft SQL Server 2008 R2 Database*, *Bulk Insert* merupakan sebuah peralatan atau *tools* pada suatu basis data yang digunakan untuk melakukan transfer data secara langsung dari file data sistem pada sistem operasi ke sebuah tabel basis data. Untuk mempercepat proses transfer data diperlukan beberapa hal yang perlu diperhatikan yaitu dengan adanya konstrain, pengindexan dan *trigers* pada basis data akan memperlama proses transfer data, untuk itu saat proses transfer data akan dilakukan perlu menon-aktifkan kesemuanya. Sebagai contoh dengan adanya foreign key, suatu data akan dicek ketersediaan data itu pada tabel yang bersangkutan, jika ada ribuan lebih data yang akan dimasukkan maka data ini akan dicek satu persatu setiap kali akan masuk.

Selain basis data penelitian ini juga akan memanfaatkan daya dari memori primer. Memori primer merupakan tempat penyimpanan data dan program sebelum data dieksekusi. Memori utama ini bersifat volatile atau tidak dapat mempertahankan data dan program yang disimpan bila sumber daya dihentikan. Prosesor akan mengambil instruksi dari memori(*fetch*) dan mengeksekusi instruksi yang diambil(*execute*) (Harianto,2008). Jika dibandingkan dengan memori sekunder tentu memori utama memiliki kecepatan akses yang tinggi, namun memiliki kapasistas penyimpanan yang lebih kecil. Memori utama terdiri dari *Random Access Memory*(RAM) yang digunakan untuk menyimpan program dan data yang akan dieksekusi oleh prosesor dan *Read Only Memory*(ROM) yang hanya bisa digunakan untuk membaca saja berupa bootstrap program dan BIOS(Basic Input Ouput System)(Jogiyanto,2003).

3. METODE PENELITIAN

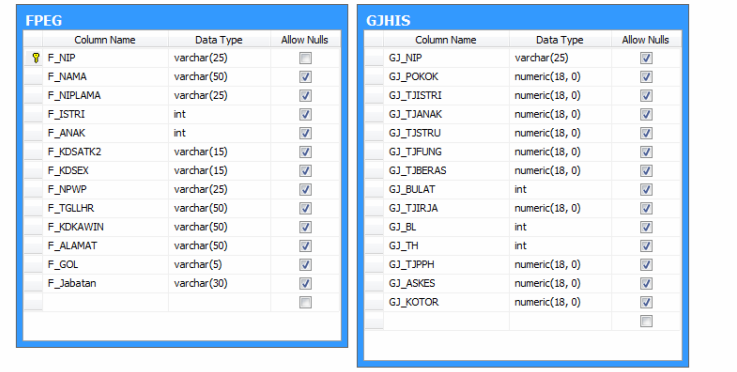
Dalam penelitian ini akan digunakan komputer dengan spesifikasi prosesor dengan clock 3.1 Ghz yang memiliki 2 inti prosesor secara fisik dan memiliki teknologi *hyperthread* yang memungkinkan 1 inti prosesor dianggap menjadi 2 prosesor oleh sistem operasi dan RAM sebesar 4 GB, menggunakan sistem operasi 64 bit dan menggunakan framework C# .Net untuk sistemnya. Untuk proses mendapatkan kinerja prosesor dan RAM dilakukan dengan pengamatan pada task manager pada bagian performa. Pengamatan dilakukan di 1 menit pertama ketika sistem sudah mulai melakukan sesuatu yang berulang. Berikut merupakan 2 langkah besar dalam penelitian dalam penelitian ini:

Langkah 1 : Melakukan analisis untuk algoritma input pada sistem
Analisis yang dilakukan meliputi analisis matematis dengan menggunakan big O dan menggunakan analisis empiris dengan mengambil sample waktu dari algoritma

Langkah 2 : Melakukan optimalisasi algoritma input pada sistem
Mengubah proses penyimpanan data dari memori sekunder ke memori primer
Langkah ini termasuk proses-proses pengadaptasian algoritma untuk menanggapi perubahan penyimpanan data ke memori primer dan menggunakan teknologi bulk insert sebagai alat untuk transfer data dari memori primer untuk disimpan pada basis data. mencatat waktu eksekusi dari algoritma yang sudah diadaptasikan prosesnya
Sama seperti sebelumnya langkah ini akan mencatat perubahan waktu yang diperoleh dari proses sebelum memanfaatkan memori primer dan setelahnya.

3. HASIL DAN PEMBAHASAN

Berikut merupakan hasil dan pembahasan berdasarkan langkah langkah pada metodologi penelitian:
Langkah 1: Melakukan analisis untuk algoritma input pada sistem
Dalam penelitian ini seperti dapat dilihat pada gambar 1 terdapat 2 macam input yang pertama merupakan input data gaji dan pegawai. Dalam sistem ini basis data dapat dibagi kedalam 2 buah tabel yang bersesuaian dengan data yang akan diolah yaitu tabel gaji dan tabel pegawai seperti yang tampak dalam gambar berikut:



Gambar 2. Tabel FPEG dan GJHIS padabasis data

Tabel FPEG merupakan tabel pegawai dan GJHIS merupakan tabel gaji. Kedua tabel ini tidak memiliki keterikatan satu sama lain untuk mempercepat proses penyimpanan. Relasi tabel yang berkurang tentunya mengurangi beban dari query saat dijalankan dibanding dengan tabel yang memiliki banyak relasi. Dapat dilihat pada tabel GJHIS tidak memiliki *primary key* dan tabel pada FPEG memiliki. Hal ini yang akan menjadi pembeda nantinya untuk proses optimalisasi dengan menggunakan memori primer dan bulk insert. Pada tabel FPEG memiliki dua buah NIP (Nomor Induk Pegawai) karena ada nip baru dan ada nip lama, dimana dalam sistem ini salah satunya saja yang menjadi primary key. Sehingga untuk pengecekan NIP dilakukan pada 2 kolom ini.

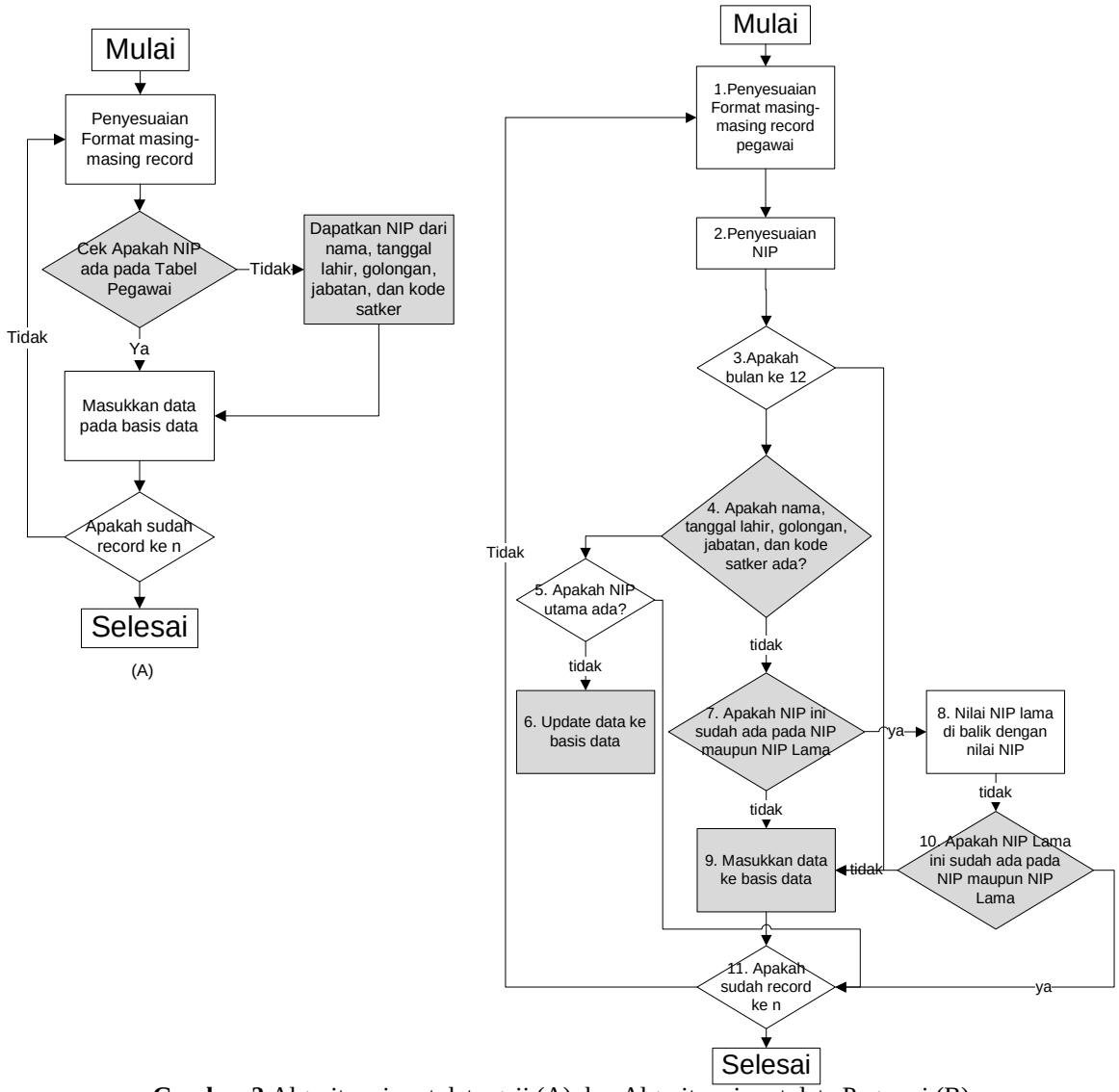
Untuk yang pertama akan dibahas analisis mengenai algoritma pada proses input pada tabel GJHIS. Algoritma input ini tanpa memperhitungkan nilai query memiliki nilai kompleksitas O(n) atau berbentuk persamaan linear. Nilai n disini adalah jumlah data gaji dalam satu tahun. Namun jika ditambah dengan proses query untuk melakukan pengecekan dan untuk melakukan input data menjadi $O(n\ (q_1+q_2))$. Nilai q_1 adalah nilai kompleksitas big O pada query untuk cek Nip dan q_2 adalah nilai kompleksitas query untuk mendapatkan nip berdasarkan nama, tanggal lahir, golongan, jabatan, dan kode satker. Sementara untuk query insert ke dalam basis data, DBMS(Database Management System) tidak melakukan pengecekan ke unikan data sebab tidak menggunakan primary key maupun index.

Untuk yang kedua adalah pembahasan analisis mengenai algoritma pada proses input pada tabel FPEG yang memiliki primary key. Algoritma input tanpa memperhitungkan query memiliki nilai kompleksitas O(n) atau berbentuk persamaan liniear, dengan nilai n adalah jumlah data pegawai dalam satu bulan. Namun jika proses query dipertimbangkan kompleksitasnya dalam kondisi terbaik (langkah 1,2,4,6,11) menjadi $O(n\ q_4)$. nilai q4 disini adalah kompleksitas untuk query pada langkah no 4. Dalam kondisi terburuk (langkah 1,2,3,4,7,8,10,9,11) nilai kompleksitasnya menjadi $O(n(q_4+q_7+q_{10}+ q_9+ q_{11})$. Dimana nilai q adalah kompleksitas untuk masing-masing query. Selain dari sisi kompleksitas algoritma kecepatan suatu algoritma untuk menjalankan suatu instruksi dipengaruhi juga oleh kemampuan perangkat keras seperti prosesor dan RAM (Random Access Memory). Berikut merupakan gambar flow chart algoritma, warna gelap pada latar belakang diagram adalah proses query.

Pada algoritma ini didapatkan data empiris mengenai waktu eksekusi algoritma sebagai berikut:

Tabel 1 Tabel performa algoritma input gaji

No	**Jumlah Data**	Waktu
1	198096	35 menit 13 detik
2	198096	34 menit 57 detik
3	198096	34 menit 55 detik
Prosesor 23%-27% RAM 0.06Gb		



Gambar 2 Algoritma input data gaji (A) dan Algoritma input data Pegawai (B)

Tabel 2 Tabel performa algoritma input pegawai tanpa optimalisasi

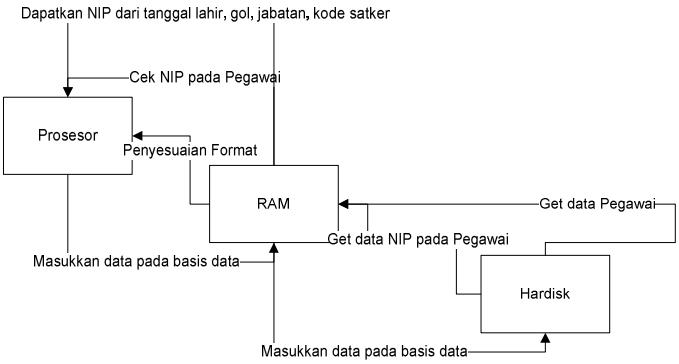
No	Jumlah Data	Dengan Format memiliki 2 NIP	Dengan Format 1 NIP
1	198096	13 menit 5 detik	12 menit 12 detik
2	198096	13 menit 10 detik	12 menit 10 detik
3	198096	13 menit 3 detik	12 menit 17 detik
prosesor 23%-26%, RAM 0.06Gb			

Dalam percobaan empiris tabel no 2 data dengan format 2 NIP memiliki kondisi kompleksitas yang bervariasi dari kondisi terbaik hingga kondisi terburuk, namun kondisi terburuk lebih sering terjadi pada kasus format yang memiliki 2 NIP, namun dalam kasus ini format 1 NIP lebih sering berada dalam kondisi $O(n(q_7+q_9+q_{11}))$.

Langkah 2 : Melakukan optimalisasi algoritma input pada sistem

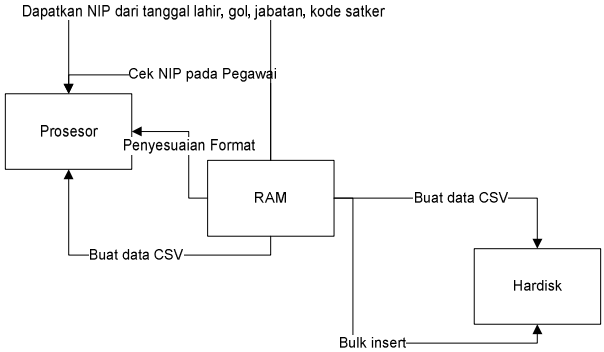
Dari gambar 1 dapat dilihat bahwa proses yang berwarna abu-abu dan proses memasukkan data ke basis data adalah proses yang memerlukan koneksi suatu sistem terhadap basis data atau dengan kata lain dengan memori sekunder, dimana ketika perintah dieksekusi sistem akan membaca maupun menulis data ke memori sekunder yang dimana kecepatannya bergantung pada RPM(revolution per minutes) hardisk yang dimana kecepatanya jauh dibawah kecepatan membaca data yang terletak di memori primer RAM. Dalam optimalisasi ini akan meminimalisir akses data pada algoritma ke memori primer. Jika memori sekunder atau koneksi ke

perangkat lain seperti *harddisk* dilibatkan dalam algoritma tentunya fungsi untuk waktu eksekusi tidak hanya diukur menggunakan kompleksitas algoritma saja namun juga melibatkan banyak algoritma melakukan akses koneksi ke perangkat lain tersebut. Pada pengolahan data satuan atau kecil efek dari koneksi ini sangatlah kecil, namun ketika melibatkan data dalam skala besar tentunya, waktu kecil ini akan terakumulasi dengan banyaknya data dan menyebabkan program berjalan lambat. Berikut merupakan gambar dari proses algoritma input gaji jika dilihat dari koneksi antar perangkatnya.



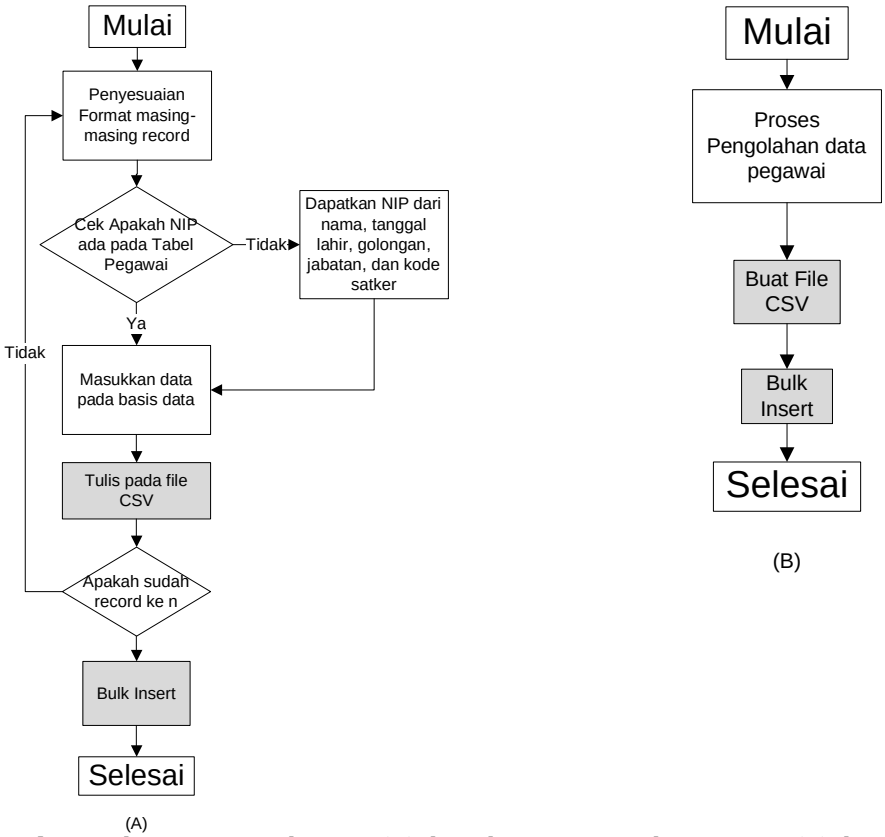
Gambar 3 Proses algoritma input gaji dilihat dari koneksi antar perangkatnya

Dari gambar terlihat bahwa koneksi ke hardisk dilakukan sebanyak 3 kali pada setiap perulangan tiap record sehingga jumlah koneksinya menjadi 3n dengan n adalah banyak data. Namun pada proses optimalisasi ini koneksi ke hardisk akan dilakukan hanya 1 kali yang dilakukan pada perulangan yaitu ketika membuat text dengan format CSV. Untuk melakukannya hanya mungkin dilakukan dengan memanfaatkan RAM sebagai penampung data dan menggunakan bulk insert untuk transfer data ke basis data berikut proses algoritma yang akan terjadi:



Gambar 4 Proses algoritma optimalisasi input gaji dilihat dari koneksi antar perangkatnya

Dengan adanya proses seperti itu maka proses algoritmanya akan menjadi seperti gambar 5
Pada gambar B proses pengolahan data pegawai adalah proses dari no 1-10 pada gambar 2 dimana proses-proses basis data dialihkan ke memori primer. Jadi dapat dilihat pada proses input data gaji koneksi ke hardisk dilakukan hanya 1 kali pada tiap perulangan, dan satu kali diluar perulangan. Sementara pada input data pegawai tidak ada koneksi ke hardisk pada perulangan utama, dan ada 2 kali koneksi diluar perulangan. Proses bulk insert pada basis data adalah proses transfer data skala besar dari suatu data ke basis data secara cepat yang dikarenakan melewati proses penyimpanan pada log. Data yang akan ditransfer disiapkan oleh sistem dalam bentuk text dalam format CSV sehingga sistem bulk insert tinggal mengambil data dari file text ini. Berikut merupakan hasil percobaan empiris yang diperoleh data dalam kondisi yang sama dengan saat percobaan pada tabel 1 dan tabel 2.



Gambar 5 Algoritma input data gaji (A) dan Algoritma input data Pegawai (B) dengan Memori Primer

Tabel 1 Tabel performa algoritma input gaji dengan optimalisasi

No	**Jumlah Data**	Waktu
1	198096	0 menit 49 detik
2	198096	0 menit 49 detik
3	198096	0 menit 48 detik
Prosesor 25-27% RAM 0.12Gb		

Tabel 2 Tabel performa algoritma input pegawai dengan optimalisasi

No	**Jumlah Data**	**Dengan Format memiliki 2 NIP**	**Dengan Format 1 NIP**
1	198096	1 menit 47 detik	1 menit 46 detik
2	198096	1 menit 47 detik	1 menit 45 detik
3	198096	1 menit 47 detik	1 menit 45 detik
Prosesor 23-26%, RAM 0.11Gb			

Dari data dapat dilihat bahwa terjadi peningkatan kecepatan yang sangat drastis dari metode lama dibanding dengan penerapan metode optimalisasi ini. Pada ke proses algoritma tidak terlihat perubahan yang mencolok dari pendayagunaan prosesor, namun terlihat ada perubahan pada sisi memorinya dimana algoritma optimalisasi memiliki daya RAM yang cenderung lebih tinggi. Pada input pegawai penggunaan daya ram memiliki selisih 0.05 Gb atau setara dengan 50 Mb sementara itu pada input data gaji terdapat selisih 0.06Gb atau setara dengan 60Mb, hal ini tentu dikarenakan RAM digunakan untuk penyimpanan data pegawai sementaranya. Dalam kasus pengolahan data skala besar dan memanfaatkan RAM ini diperlukan perhatian untuk beberapa hal kususnya untuk terjadinya eksepsi. Eksepsi memiliki dampak yang cukup besar terhadap waktu, ketika suatu proses terjadi eksepsi yang biasanya dilakukan dengan menggunakan try dan catch maka sistem akan menghasilkan suatu laporan-laporan eksepsi yang cukup memakan waktu. Dari hasil pengembangan metode ini, permasalahan eksepsi terutama yang berkaitan dengan tiap record memberikan delay waktu sehingga waktu keseluruhan menjadi hampir sama dengan metode tanpa optimalisasi dengan perbedaan waktu yang tidak mencolok. Selain itu hal-hal yang berkaitan dengan pengecekan terhadap sesuatu yang tidak ada datanya akan

memperlama waktu pula. Dapat dilihat pada algoritma input pegawai pada gambar 2 B langkah no 3, merupakan langkah antisipasi untuk mengatasi permasalahan ini, pada algoritma input pegawai, data yang dimasukkan adalah data pegawai pada bulan 12 terlebih dulu agar data terbaru yang terpakai, sehingga saat terjadi pengecekan tabel pegawai telah memiliki isi.

5. KESIMPULAN DAN SARAN

Dalam Penelitian mengenai optimasi algoritma query pada kasus eksploitasi sumber daya *multi-core* ini dapat ditarik beberapa kesimpulan dan saran:

1. Sistem Pengolah data perpajakan PNS perkabupaten dieksekusi dengan waktu yang lebih cepat, dengan perbedaan yang signifikan.
2. Dalam mengembangkan algoritma optimalisasi yang melibatkan basis data, variabel banyak koneksi ke basis data juga perlu diperhatikan, tidak hanya kompleksitas dari algoritma saja.
3. Penggunaan RAM lebih banyak dalam penggunaan metode ini, hal ini dikarenakan metode ini memanfaatkan RAM sebagai memori penyimpan data keseluruhan.
4. Faktor-faktor eksepsi yang berlaku pada tiap record dan pencarian data pada sumber kosong akan menghambat kecepatan pada metode ini dalam kasus data skala besar.

DAFTAR PUSTAKA

--, 2008, 6231B Maintaining a Microsoft SQL Server 2008 R2 Database, Microsoft,pp8-6:8-8

--, 2012, Country Comparison Population, https://www.cia.gov/library/publications/the-worldfactbook/rankorder/2119rank.html, diakses pada tanggal 23-03-2012.

Abo-Hamad , Waleed, Amr Arisha, *Simulation–Optimisation Methods in Supply Chain Applications: A Review*,2011,irish journal management,pp97-124

Cisty, Milan, 2009, *Hybrid Genetic Algorithm and Linear Programming Method for Least-Cost Design of Water Distribution Systems*, Water Resour Manage,No 24,pp 1–24

Deshpande, Amol, Zachary Ives, Vijayshankar Raman,Adaptive Query Processing,2007 Foundations and TrendsR_ in Databases Vol. 1,pp 1-140

Gour,Vishal, S.S.Sarangdevot, Anand Sharma, Vinod Choudhary, 2010, *Improve Performance of Data Warehouse by Query Cache*, International Conference on M ethods and M odels in Science and Technology, pp 198-200

Hagh, Mehrdad Taraf Dar, Sadjad Galvani, 2011,, *Minimization of load shedding by sequential use of linear programming and particle swarm optimization* Turk J Elec Eng & Comp Sci, Vol.19, No.4, pp551-563

Jananto, Arif, 2006, *Meningkatkan Kecepatan Akses Data dengan Teknologi Optimalisasi Query Rushmore*, Jurnal Teknologi Informasi DINAMIK Volume XI, No. 1, pp 47-56 ,

Maimos, Mahamat, Yves Cherruault, Balira O. Konfe, Ange-gar S. Nkokolo Massamba, 2009, *Alienor method to solve multi-objective linear programming (MOLP)*, Kybernetes, Vol. 38, No. 5, pp. 789-799

McAllister , S.R., R. Rajgaria , C.A. Floudas, 2008, *A path selection approach to global pairwise sequence alignment using integer linear optimization*, Taylor and Francis Vol. 57, No. 1, pp 101–111

Prabha, Selvaraj,Arphutaraj Kannan, Palaniappan Anandhakumar, 2006, *An Optimization Query Processor With An Efficient Caching Mechanism For Distributed Databases*, International Arab Journal, vol 3 no 3, pp 231-236

Siallagan ,Manahan, Mira Kania Sabariah, Malanita Sontya , 2008, *Optimalisasi Query Database Menggunakan Algoritma Genetik*, Seminar Nasional Aplikasi Teknologi Informasi,pp53-57

Wahyu W, Tri, 2008 , *Pengoptimalisasian Pencarian Data Dengan Algoritma Subset Query*, Jurnal Artificial, ICT Research Center UNAS, Vol.2, pp66-69

Wang, Yuping, 2010, *A Uniform Enhancement Approach For Optimization Algorithms: Smoothing Function Method*, World Scienti¯c Publishing Company, Vol. 24, No. 7, 1111_1131

Niknam, Taher, Ehsan Azad Farsani, Majid Nayeripour,Bahman Bahmani Firouzi, 2011, *Hybrid Fuzzy Adaptive Particle Swarm Optimization and Differential Evolution Algorithm for Distribution Feeder Reconfiguration*, Electric Power Components and Systems, 39: p p 158–175