ISSN: 2087-1716
Jurnal Ilmiah ILKOM Volume 8 Nomor 3 (Desember 2016)

ANALISIS EMPIRIS PERBANDINGAN KINERJA METODE HASHING PROGRESSIVE OVERFLOW DAN LINEAR QUOTIENT DALAM STUDI PEMBUATAN APLIKASI DEKSTOP ADMINISTRASI KEPEGAWAIAN

Eko Prianto[1], Anton Yudhana[2], Abdul Fadlil[3]
[1]ekoprianto05@gmail.com, [2]eyudhana@mti.uad.ac.id, [3]fadlil@mti.uad.ac.id
[123]Program Studi Magister Teknik Informatika Universitas Ahmad Dahlan Yogyakarta, Indonesia

Abstrak

Administrasi merupakan kegiatan sehari-hari yaitu berupa kegiatan mencatat, mengumpulkan dan menyimpan suatu kegiatan atau hasil kegiatan sehingga membutuhkan memori guna menyimpan itu semua. Untuk memudahkan proses pencarian data yang telah disimpan dengan cepat, maka diperlukan suatu metode, yaitu menggunakan metode *hashing.* Penerapan manajemen memory bertujuan, agar mencegah adanya perulangan data yang sama, manajemen basis data yang baik, manajemen memori dan memberikan kemudahan untuk mengakses data dengan cepat walaupun data yang ada sangat banyak, dengan memasukkan kata kunci yang telah ditentukan dapat mencari data dan menampilkan dengan akurasi waktu yang dibutuhkan sebentar. Dengan metode *progressive overflow* dan *linear quotient* akan dianalisis nilai rata-rata dari setiap metode, nilai yang paling kecil dari metode tersebut adalah nilai yang terbaik akan digunakan sebagai pedoman seberapa besar nilai proses *hashing* dalam sebuat database suatu instansi dan sebagai studi kasus penentuan penyediaan memori yang dibutuhkan.

Kata kunci: *Hashing*, *Collision, key*, Administrasi.

1. Pendahuluan

Dalam melaksanakan pengelolaan suatu Instansi, diperlukan kebijakan-kebijakan yang tepat dilakukan oleh karyawan atau orang-orang dalam instansi tersebut dan juga memerlukan suatu cara yang dapat memberikan informasi sebagai bahan pertimbangan dalam pengambilan keputusan, yaitu adanya perencanaan.

Administrasi merupakan kegiatan sehari-hari yaitu berupa kegiatan mencatat, mengumpulkan dan menyimpan suatu kegiatan atau hasil kegiatan untuk membantu pimpinan dalam mengambil keputusan. Dalam setiap kegiatan baru masih sering juga kita membutuhkan *file* yang telah tersimpan untuk kita ambil, dan membutuhkan waktu untuk mencarinya, hal demikian dapat menghambat kelancaran proses administrasi.

Dengan penerapan manajemen memory, agar mencegah adanya perulangan data yang sama, manajemen basis data yang baik, manajemen memori dan memberikan kemudahan untuk mengakses data dengan cepat walaupun data yang ada sangat banyak, dengan memasukkan kata kunci yang telah ditentukan dapat mencari data dan menampilkan dengan akurasi waktu yang dibutuhkan sebentar.

Permasalahan di atas dapat diselesaikan dengan adanya sebuah metode yaitu metode *hashing.* Dalam metode *hashing* yang akan digunakan adalah *hashing statis.* Ada beberapa turunan metode *hashing statis* yaitu *progressive overflow, linear quotient,* dan *Coalesced* Hash (dan Variannya : LISCH, EISCH, LICH dan EICH), dalam journal ini penulis mencoba membandingkan kinerja metode *hashing progressive overflow dengan linear quotient,* sebagai batasan masalah dalam penelitian ini [2].

Hashing merupakan metode pengaksesan data yang dilakukan dengan cara memetakan/mengkonversikan himpunan kunci *record* menjadi himpunan alamat memori (posisi *subscript* dalam larik), sehingga range *address* menjadi kecil. Fungsi untuk mengkoversikan nilai kunci aktual menjadi lokasi alamat disebut fungsi hash, sedangkan metode pencarian yang memanfaatkan fungsi *hash* disebut sebagai *hashing* atau hash *addressing*. Tujuan utama fungsi *hash* adalah agar dua buah nilai kunci (=r) mempunyai nilai *hash* yang berbeda. Jika tidak, maka akan terjadi *collision*/*mash clash* [2]. Untuk menentukan suatu fungsi *hash* H, diperlukan asumsi sebagai berikut:

1. Kunci yang diletakkan adalah *primary key*
2. Tidak ada *altemate key*

3. Satu alamat digunakan untuk satu *primary key*
4. *Primary key* diasumsikan tersusun atas sebuah atribut (*simple key*)
5. Memori terbatas 1 blok dengan alamat dinormalkan, yaitu 0 sampai maksimum blok (virtual address). Nilai fungsi hash terletak pada range hash: 0..P-1, dimana P adalah bilangan prima terkecil yang lebih besar dari cacah kunci.

Dengan demikian perlu adanya alternatif media untuk menunjang administrasi. Permasalahan di atas dapat diselsaikan dengan adanya sebuah perencanaan, agar dalam proses pencarian suatu data dapat dilakukan dengan cepat. Dari metode yang dihasilkan nantinya dapat digunakan sebagai: Manajemen basis data, sehingga mengurangi tingkat *collision* data base, penumpukan ataupun pengulangan database dan manajemen memory, pada hal ini memory yang akan terpakai pada saat jumlah data yang ada saja, sehingga dapat mengurangi tingkat pemborosan penggunaan memory.

2. Landasan Teori

2.1 Empiris

Empiris adalah ilmu pengetahuan yang di dasarkan pada observasi kenyataan akal sehat, serta hasilnya tidak spekulatif. Jadi data empiris merupakan data yang ditemukan atau disimpulkan dari sebuah eksperimen atau penelitian. Seperti yang kita ketahui pada umumnya bahwa ilmu dan penelitian pada zaman modern sekarang ini memiliki kiblat empiris sebagai peneitian [5].

2.2 Hashing

Berdasarkan distribusi seragam nilai diberbagai kotak, yang nilainya diberikan dan ditentukan oleh fungsi, disebut fungsi hash. Dari pertimbangan beberapa teknik hashing, tidak ada satu teknik yang terbaik. Sebaliknya, setiap teknik yang akan cocok untuk aplikasi database tertentu dan dievaluasi dari beberapa faktor antara lain [6]:

1. Jenis akses
   Jenis akses yang digunakan secara efisien, dapat mencakup dan menemukan catatan dengan nilai atribut tertentu serta menemukan catatan atribut jatuh dalam kisaran yang telah ditetapkan.
2. Waktu akses
   Waktu yang diperlukan untuk menemukan item data tertentu, atau kumpulan item, menggunakan teknik yang bersangkutan.
3. Waktu *insertion*
   Waktu yang diperlukan untuk memasukkan item data baru. Nilai ini termasuk waktu yang dibutuhkan untuk menemukan tempat yang tepat untuk memasukkan item data baru, serta sebagai waktu yang dibutuhkan untuk memperbarui struktur indeks.
4. Waktu penghapusan
   Waktu yang diperlukan untuk menghapus item data. Nilai ini termasuk waktu yang dibutuhkan untuk menemukan item yang akan dihapus, serta waktu yang dubutuhkan untuk memperbarui struktur indeks.
5. Ruang *overhead*
   Ruang tambahan ditempati oleh struktur indeks. Asalkan jumlah ruang tambahan cukup, biasanya bernilai sementara dikorbankan untuk mencapai peningkatan kinerja.

2.3 Progressive Overflow

Progressive Overflow merupakan salah satu metode yang paling sederhana yaitu Metode Progressive Overflow Metode ini jika didesain dengan salah bisa menimbulkan infinite loop, dimana alamat alternatif tidak dapat ditemukan karena pencarian hanya memutar-mutar di tempat yang sama Kelemahan ini dapat diatasi dengan cara : Jarak pencarian diambil suatu nilai yang bukan merupakan faktor dari jumlah ruang alamat (n) Banyaknya ruang alamat diambil suatu bilangan prima, misalkan ada 10 data, maka disediakan ruang alamat sebanyak 11 Maju Mundur [4].

2.4 Linear Quotient

Linear Quotient disebut juga doube hashing yaitu mirip dengan progressive overflow, tetapi menggunakan kenaikan variabel dari pada satu untuk pergi kelokasi berikutnya. Ketika memiliki sinomin, hash mendapatkan lagi lokasi berikutnya (menggunakan fungsi yang berbeda). Sinonim selanjutnya dapat menggunakan fungsi yang sama lagi atau satu dimodifikasi dengan jumlah sinonim. Potensi masalah untuk keluar, untuk mencari seluruh file dan menemukan ketika benar-benar penuh hanya melompati ruang yang tersedia, bisa dengan menghindari dan melacak jumlah lokasi yang dicari [3].

2.5 Basis Data

Perancangan basis data merupakan bagian dari kegiatan besar dalam rangka pengembangan sistem informasi manajemen (SIM), khususnya pada tahap perancangan. Perancangan basis data dilakukan dengan menentukan kebutuhan file-file dalam basis data berdasarkan model sistem tersebut ditunjukkan oleh diagram airan data/DAD (*data flow diagram*/DFD)sistem baru yang telah dibuat pada tahap sebelumnya [8].

2.6 Adminitrasi Kepegawaian

Administrasi kepegawaian adalah seluruh aktivitas atau kegiatan yang berkaitan dengan masalah penggunaan pegawai (tenaga kerja) untuk mencapai tujuan. Sedangkan administrator bertujuan untuk menyusun dan mengendalikan seluruh aktivitas untuk memelihara, mengembangkan, mendapatkan maupun menggunakan para pegawai sesuai dengan beban kerja untuk mencapai tujuan organisasi atau perusahaan yang telah di tentukan sebelumnya [7].

2.7 Delphi

Merupakan membangun sebuah aplikasi menggunakan delphi berarti bekerja denga satu lingkungan terpadu yang dinamakan Delphi. IDE (*Integrated Development Environment).* IDE ini meliputi beberapa *form* yang berguna dalam mendukung aktifitas pemrograman yang kita lakukan [1].

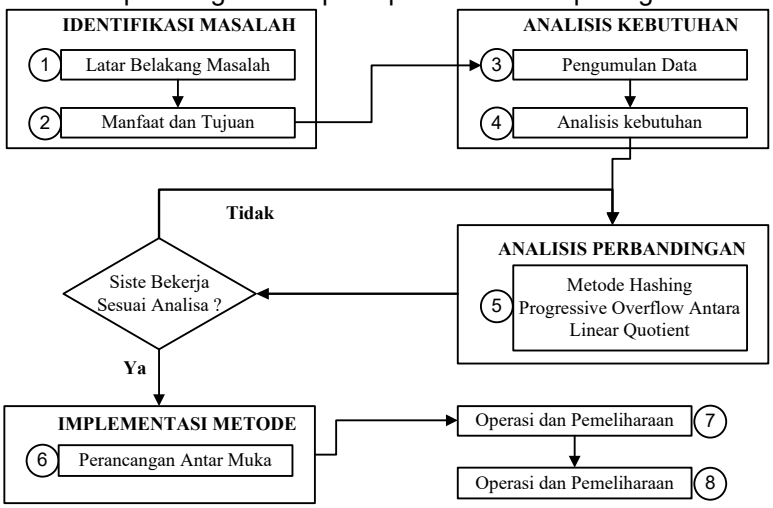
3. Metode

Dalam metodologi penelitian ini, teknik pengumpulan data yang digunakan dalam penelitian ini yaitu:

1. Studi Literatur
   Pengambilan data dengan menggunakan buku, paper dan sumber ilmiah lain, seperti jurnal internasional, situs internet ataupun artikel teks dokumen yang berhubungan dengan penilitian ini.
2. Metode Observasi
   Mengadakan pengamatan langsung terhadap obejek penelitian, yaitu pelaksanaan kegiatan dan administrasi Puskesmas Jatisrono I.
3. Metode Wawancara
   Metode ini dilakukan dengan mengadakan Tanya jawab langsung kepada karyawan Puskesmas Jatisrono I sehingga data yang diperoleh bersifat objektif dan data dipertanggungjawabkan.

3.1 Perancangan Pengembangan Sistem

Dari bebrapa metode penelitian yang digunakan, maka penelitian dilakukan berdasarkan diagram alir penelitian dibawah ini. Adapun diagram alir pada penelitian ini dapat digambarkan sebagai berikut:



Gambar 1. FlowChart Perbandingan Metode

Sebelum melakukan pengujian metode, maka ditentukan terlebuh dahulu input data yang akan diuji yaitu berupa *key* (kata kunci), dalam hal ini *key* yang di tentukan berupa *number* berjumah 10 gidit angkat, yang akan dikelompokkan menjadi 5 kelompok.

Pengujian metode *progressive overflow*, yang mana pada metode ini akan diterapkan aturan-aturan yang ada pada metode tersebut, dan akan menghasilkan nilai *collision* dan nilai rata-rata yang akan dibandingan dengan metode berikutnya. Sedangkan metode *linear quotient,* demikian halnya pada metode sebelumnya, metode ini juga terdapat aturan-aturan untuk menganalisa sejauh mana tingkat keberhasilan dalam implimentasi manajemen memori dan kecepatan *searching* seperti pada gambar 1. Dari hasilnya analisis metode ini juga akan mempunyai nilai *collision* yang akan menunjukkan seberapa efektifk untuk implementasi studi kasus nantinya. Jika metode *linear quotient* lebih baik maka yang akan digunakan, jika tidak maka akan kembali pada metode *progressive overflow,* sebagai implementasi studi kasus berapa jumlah memori yang di butuhkan dalam sebuah database.

a) Metode Progressive Overflow

Metode coaleced hashing membutuhkan media penyimpan tambahan untuk menyimpan link field. Pada saat media penyimpan tidak mungkin untuk menerima penambahan lagi, maka penggunaan

metode tersebut bukan merupakan pilihan yang tepat. Metode konvensional sederhana untuk mengatasi masalah tersebut adalah menggunakan metode *progressive overflow/liniear probing*. Dalam metode ini, jika suatu kunci mengalami *collision*, maka nilai kunci tersebut akan diletakkan pada indeks selanjutnya yang masih kosong dengan fungsi hash:
$(K) = K \ MOD \ P$
Jika hingga indeks terakhir sudah penuh, maka pencarian lokasi dilakukan mulai dari awal tabel, sehingga membentuk struktur indeks *circular*.
Algoritma metode progressive overflow/liniear probing adalah sebagai berikut:
1) Konversikan nilai kunci untuk mendapatkan home address
2) Jika home address kosong, maka sisipkan kunci pada lokasi tersebut. Jika isi, maka lakukan proses berikut:
   a. Tentukan nilai penambahan (*increment*) dengan cara mencari sisa bagi nilai kunci dengan ukuran tabel, jika hasilnya 0 (nol) set nilai variabel *increment* menjadi 1
   b. Set nilai pencacah (*counter*) untuk pencarian lokasi kosong menjadi 1
   c. Selama nilai pencacah (*counter*) kurang dari ukuran tabel, maka lakukan proses berikut:
      Hitung lokasi selanjutnya dengan cara menambahkan nilai/indeks alamat sekarang dengan nilai increment dan kemudian hasilnya di modulo dengan ukuran tabel
      Jika alamat kosong, sisipkan kunci pada lokasi tersebut. Jika tidak kosong, tambahkan nilai pencacah (*counter*) dengan 1 dan kembali ke langkah-1)
   d. Akhiri dengan komentar "Tabel penuh….."

Contoh:
Jika diketahui nilai kunci sebagai berikut:
14, 15, 08, 12, 25
Penempatan nilai-nilai kunci tersebut dengan metode *progressive overflow* adalah sebagai berikut ini.
Penyelesaian:
N = 5, P = 7, Alamat = 0 s/d 6
Perhitungan:
H(14) = 14 MOD 7 => 0
H(15) = 15 MOD 7 => 1
H(08) = 08 MOD 7 => 1 (collision)
Ditempatkan pada lokasi berikutnya yang kosong: 2
H(12) = 12 MOD 7 => 5
H(25) = 25 MOD 7 => 4
Hasil perhitungan tersebut ditampilkan dalam Tabel 1. Rata-rata untuk akses suatu nilai kunci adalah:
= (5 + 1)/5 = 8/5 = 1,2
Keterangan:
5: Langkah penempatan setiap kunci pada home *address*
1: Langkah penempatan kunci 08 (mengalami *collision*)

Tabel 1: Penempatan kunci dengan metode *progressive overflow* (1)

Record	Kunci
0	14
1	15
2	08
3	25
4	
5	12
6	

b) Metode Linear Quotient
Metode linear quotient mirip dengan metode progressive overflow. Dalam metode progressive overflow nilai increment adalah tetap, yaitu 1 (satu). Sedangkan dalam metode linear quotient nilai increment dapat dipilih di antara dua pilihan berikut:
1) Menggunakan hasil quotient (hasil bagi (div) nilai kunci dengan ukuran tabel)
2) Menggunakan fungsi hash yang lain, sehingga metode ini sering pula disebut sebagai metode double hashing. Fungsi hash kedua ini berbeda (H1 untuk fungsi hash pertama dan H2 untuk fungsi hash kedua)

Metode quotient ini akan menghasilkan distribusi yang lebih baik, sehingga akan meminimalkan terjadinya collision secara berulang sebagaimana terjadi pada metode progressive overflow. Fungsi yang digunakan untuk menentukan penambahan nilai pada variabel increment, adalah sebagai berikut:

1) $H1 = K \ MOD \ P$
2) $H2 = K \ DIV \ P \Longrightarrow Hash(K) = \big(H1 + (K \ DIV \ P)\big) \ MOD \ P$

Jika penambahan nilai pada variabel increment, yaitu fungsi hash kedua dalam metode ini menggunakan fungsi hash pertama (secara linier), yaitu:
$H2 = Quotient \ (K \ DIV \ P) \ MOD \ P$

Maka lokasi baru untuk nilai kunci yang mengalami collision dihitung dengan formula sebagai berikut:

1. Konversikan nilai kunci/hashing untuk mendapatkan home address kunci.
2. Jika home address kosong, maka sisipkan kunci pada lokasi tersebut. Jika tidak, maka lakukan proses berikut:
   a. Tentukan nilai increment dengan cara mencari sisa bagi kunci dengan ukuran tabel, jika hasilnya 0 (nol) set nilai variabel increment menjadi 1
   b. Set nilai pencacah (counter) untuk pencarian lokasi kosong menjadi 1
   c. Selama nilai pencacah kurang dari ukuran tabel, lakukan:

- Hitung pencarian lokasi selanjutnya dengan cara menambahkan nilai indeks alamat sekarang dengan nilai increment dan kemudian hasilnya dimodulo dengan ukuran tabel.
- Jika alamat kosong, sisipkan kunci pada lokasi tersebut. Jika tidak kosong, tambahkan nilai pencacah (counter) dengan 1 dan kembali ke langkah -1)

Diakhiri dengan komentar: “Tabel Penuh ...”
Contoh:
Jika diketahui nilai-nilai kunci sebagai berikut:
14, 15, 08, 12, 25
Maka penempatan nilai-nilai kunci tersebut dengan metode linear quotient adalah sebagai berikut ini.
Penyelesaian:
N = 5, P = 7, Alamat indeks = 0 s/d 6
H2 = Quotient (K DIV P) MOD P
Perhitungan:
H(14) = 14 MOD 7 => 0
H(15) = 15 MOD 7 => 1
H(08) = 08 MOD 7 => 1 (collision)
Ditempatkan pada lokasi baru
Increment => (08 DIV 7) MOD 7 => 1
Lokasi baru => 1 + 1 => 2
H(12) = 12 MOD 7 => 5
H(25) = 25 MOD 7 => 4
Hasil perhitungan tersebut ditampilkan pada Tabel 2. Rata-rata untuk akses suatu nilai kunci adalah:
= (5 + 1)/5 = 6/7 = 1,2
Keterangan:
5: Langkah penempatan setiap kunci pada home address
1: Langkah penempatan kunci 08 (mengalami collision)

Tabel 2: Penempatan kunci dengan metode linear quotient

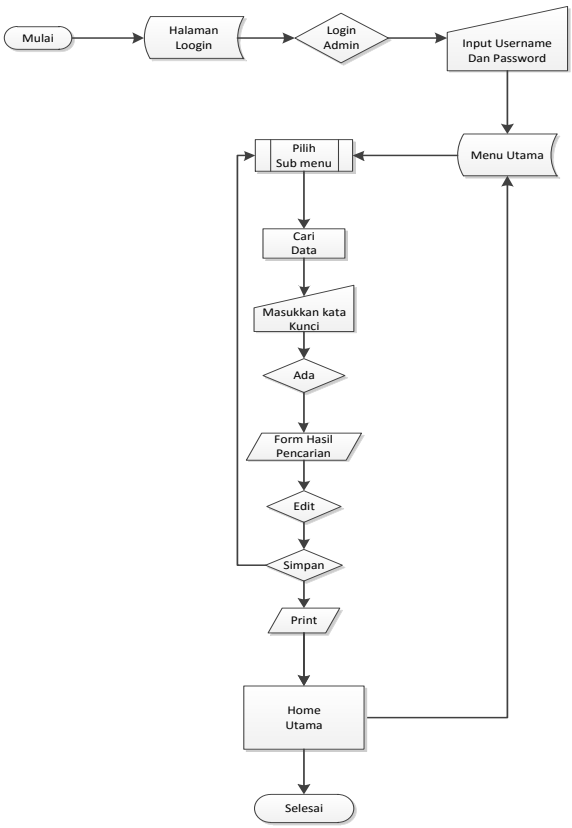
Record	Kunci
0	14
1	15
2	08
3	
4	25
5	12
6	

3.2 Sistem yang Diusulkan Adminitrasi Kepegawaian

Perancangan Basis Data Langkah yang dilakukan untuk perancangan basis data adalah sebagai berikut:

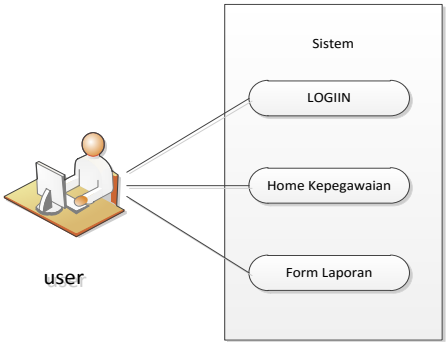
a. Menentukan sebuah kebutuhan file basis data untuk sistem baru yang dibutuhkan ditunjukkan dalam DAD, yaitu pada simbol *data store.*
b. Menentukan parameter file basis data meliputi file, nama atribut, tipe dan ukuran, serta kunci relasi.
c. Normalisasi file basis data, langkah ini dimaksudkan untuk menguji pada setiap file dengan harapan dapat menghindari permasalahan-permasalahan yang mungkin terjadi.

Optimalisasi file basis data, dapat dilakukan dengan tujuan memperoleh unjuk kerja basis data yang efisien. Perancangan administrasi pada puskesmas digunakan hanya sebatas basis data seperti pada gambar 2, yang meliputi:



Gambar 2.*User Case Diagram*

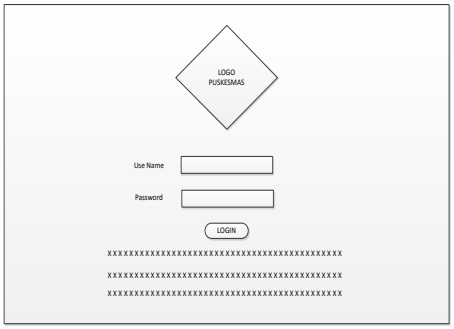
User Case Diagram dibawah ini menggambarkan bagaimana cara pengguna berinteraksi dengan sistem yang akan dibuat. Pada aplikasi ini pengguna dapat melakukan 3 interaksi *login*, *Home* kepegawaian, *Form* laporan. Untuk lebih jelas dapat dilihat pada Gambar 3 *User Case Diagram* dibawah ini:



Gambar 3. *User Case Diagram*

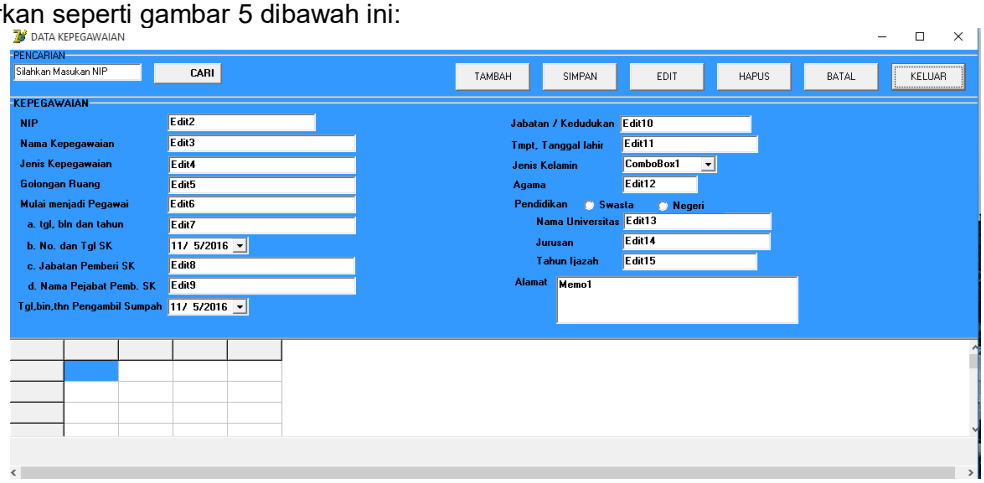
3.3 Activitas Diagram

Form Login diagram dibawah ini menggambarkan sebelum *user* masuk ke prorgam utama diharapkan untuk login terlebih dahulu. Untuk lebih jelas dapat dilihat pada Gambar 4 *Login* Aplikasi Administrasi dibawa ini.



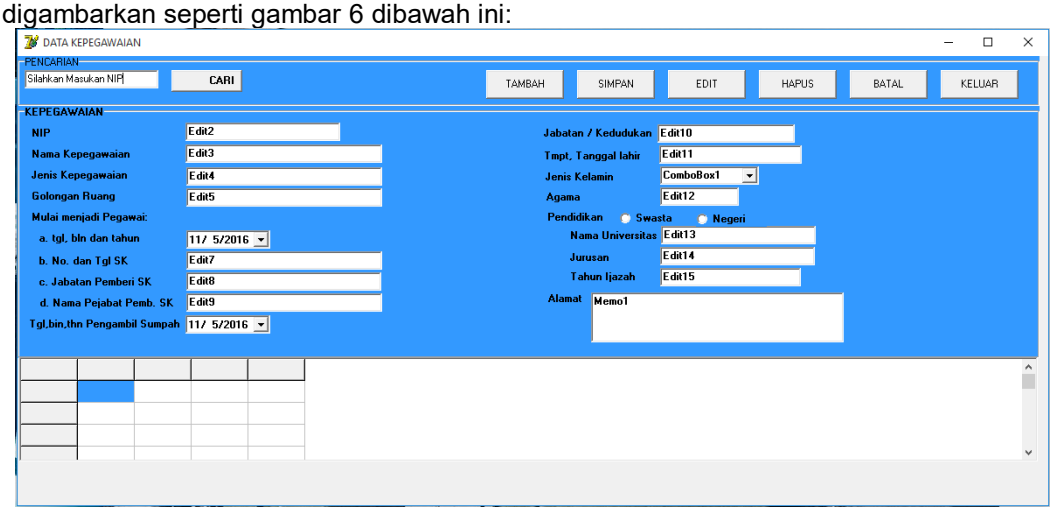
Gambar 4. *Login* Aplikasi Administrasi

Pada gambar 4 dijelaskan pengguna menu kepegawaian dan sebelumnya *login* terlebih dahulu. Pengguna dapat memilih Home kepegawaian, lalu ada pilihan data kepegawaian, data keluarga, laporan dan ajuan surat serta pengguna dapat meng-input data karyawan. Pada menu data keluarga, pengguna dapat memilih menu suami, istri, dan anak. Pada menu laporan pengguna dapat memilih mutasi dan lain-lainya. Pada Ajuan Surat Pengguna dapat memilih menu berkala, kenaikan pangkat, izin keluar, izin cuti. pada menu rekap data pengguna dapat memilih menu kepegawaian dan buku mutasi. Pada form ini juga dilengkapi dengan tombol logout untuk user. Penjelasan diatas dapat digambarkan seperti gambar 5 dibawah ini:



Gambar 5. *Home* Utama

Pada dasarnya form pada setiap home kepegawaian, susunana keluarga, riwayat kepegawaian dan rekap data, menu yang dipilih akan sama persis tampilannya, dengan ini pengguna dapat memilih menu cari, tambah, edit, simpan, hapus, batal dan keluar/kembali kemenu utama. Penjelasan diatas dapat digambarkan seperti gambar 6 dibawah ini:



Gambar 6. Form Sub Menu

4. Hasil

Pada sampel uji metode ini, mengambil 5 sampel dengan 10 digit dikelompokkan menjadi 5 kelompok, dan hasil dari analisis perbandinagn dari kedua metode tersebut adalah:

Tabel 3. Experimen analisis metode PO dan LQ

Exp.	Kunci	Metode *Hashing*	
		Progressive Overflow	***Linear Quotient***
1	1415081225	1.2	1.2
2	1215090822	2	1.6
3	1015100919	1.2	1.2
4	9515110716	1	1
5	9915091020	1.6	1.6
6	1308101786	1.8	1.4
7	0908121671	1.8	1.6
8	1010131665	1.6	1.6
9	1310111769	1.6	1.4
10	1112121867	2.4	1.8

Dari hasil 10 experiment pada tabel di atas dapat dilihat pada no 1,3,4,5,dan 8, menghasilkan nilai yang saya antara metode *progressive overflow* dengan *linear quotient,* selain itu pada no, 2, 6, 7, 9, dan 10 menghasilkan nilai yang berbeda, dan nilai pada metode *linear quotient* yang lebih kecil. Sehingga*,* dengan menggunakan aturan-aturan pada metode tersebut disimpulkan bahwa metode *linear quotient* yang lebih baik untuk digunakan sebagai implementasi pembuatan aplikasi desktop administrasi kepegawaian dari kedua metode tersebut.

5. Kesimpulan

Dari uji coba kedua mentode diatas dapat kita simpulkan bahwa metode yang lebih efektif sebagai implementasi penentuan jumlah memori yang akan disiapkan sebagai database, antara *progressive overflow* dan *linear quotient* yang dapat dilihat dari nilai rata *collision.* Maka metode *linear quotient* yang paling efektif, karena nilai-nilai *collision*-nya lebih kecil dari metode *progressive overflow* dan dapat disimpulkan:

1. Fungsi hash yang memberikan kemungkinan semakin kecil terjadinya collision berarti fungsi hash tersebut semakin baik. Secara alamiah tidak ada jaminan bahwa aspek ke-2 dapat dipenuhi tanpa terlebih dahulu mengetahui kunci-kunci yang ada.
2. Secara umum, metode *linear quotient* memiliki unjuk kerja yang sangat baik, dan sesuai digunakan jika didukung oleh ketersediaan ruang yang cukup, sedangkan kelemahannya adalah memerlukan ruang tambahan untuk menyimpan link field.
3. Secara umum, metode *progressive overflow* / *liniear probing* memiliki kelebihan karena tidak memerlukan ruang tambahan dan algoritma yang sederhana, sedangkan kelemahannya adalah unjuk kerjanya sangat buruk, sehingga jarang diterapkan.

Daftar Pustaka

[1] Abdia Away, Gunaidi. 2011. The Shourtcut of Delphi 2010 – Firebird. Infromatika: Bandung.
[2] *Austing, R.H., 1988, File Organzation and Acces, DC Heat & Co., USA*
[3] Becker,Katrin., (2002) http://www.minkhollow.ca/Courses/461/Notes/Hashing/HashingPartIII.html. Diagses 15 Juni 2016: Jam 01.31
[4] *Riyanto, http://slideplayer.info/slide/2806021/.* Diakses 22 November 2016, jam 11.45
[5] Santika, Gin. http://fiveteen.heck.in/pengertian-empiristeoritiskumulatif-dan.xhtml. Diakses 2 Juni 2016, Jam 22.39.
[6] Silberschatz, Abraham,. Henry F.K,. Sudarshan, S,*. (2011). Database System Concepts, Sixth Edition.* McGraw-Hill: Amerika
[7] Sora N. (2015). http://www.pengertianku.net/2015/05/pengertian-administrasi-kepegawaian-secara-umum.html Diakses 21 November 2016, jam 11.30
[8] Sutanta, E., 2011. *Sistem Informasi Manajemen.* Graha Ilmu Yogyakarta.