Analisis Perbandingan Peforma Beberapa Teknik Hashing pada Bidang Keamanan serta Basis Data

Freddi Yonathan - 13509012
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509012@std.stei.itb.ac.id

***Abstract*—Hash merupakan fungsi yang sangat penting dalam dunia informatika. Dengan fungsi hash, maka pencarian data pada basis data berukuran besar dapat dilakukan dengan cepat. Selain itu, berbagai teknik keamanan pun telah ditemukan dengan melibatkan fungsi hash di dalamnya. Dalam makalah ini, beberapa teknik fungsi hash akan dibahas serta akan dibandingkan kelemahan serta kelebihan masing-masing teknik hashing tersebut.**

***Index Terms*—hashing, hash, hash map, static hash, dynamic hash, double hash, salt**

I. PENDAHULUAN

Fungsi Hash merupakan fungsi yang sudah umum digunakan dalam dunia informatika. Fungsi ini dapat didefinisikan sebagai fungsi yang digunakan untuk menciptakan string dengan ukuran tertentu yang dihasilkan dari string lain berukuran bebas. Fungsi hash diharapkan akan selalu menghasilkan string yang sama untuk masukan string yang sama, dan akan menghasilkan string yang berbeda pada masukan string yang berbeda. Fungsi hash yang baik akan menghasilkan string yang berbeda jauh apabila string masukan hanya berbeda 1 karakter saja

Dalam penggunaannya fungsi hash seringkali digunakan dalam bidang perncarian data pada basis data dan bidang keamanan digital. Maka dari itu, pada makalah ini akan dibahas berbagai kegunaan serta perbandingan dari tiap teknik hash pada kedua bidang tersebut.

II. MANFAAT FUNGSI HASH PADA BASIS DATA

Salah satu kegunaan fungsi hash pada basis data adalah penggunaan fungsi hash dalam penentuan index. Pada penentuan index pada basis data dengan fungsi hash akan dibutuhkan suatu nilai atau kunci yang unik untuk setiap data pada suatu tabel basis data. Biasanya primary key dari suatu tabel yang digunakan sebagai kunci ini.

Penentuan index dilakukan dengan menggunakan kunci tersebut sebagai masukan pada fungsi hash, hasil keluaran dari fungsi hash adalah sebuah angka yang menyatakan index tempat data yang dimaksud disimpan pada tabel.

Fungsi hash yang digunakan untuk penentuan index biasanya dibuat berdasarkan kondisi tabel basis data di mana fungsi hash tersebut diterapkan. Tentu saja hasil keluaran dari fungsi hash ini diharapkan selalu berada di dalam batas index tabel yang ada. Misalkan jika ada 100 tabel, maka hasil keluaran dari fungsi hash yang dipakai berkisar antara 0 hingga 99.

Selain itu, fungsi hash dapat digunakan untuk memperkecil ukuran tabel dengan menggunakan hash key, yaitu nilai hash sebagai primary key dari suatu tabel. Misalkan ada tabel dengan atribut sebagai berikut.

Tempat	Tanggal	Biaya
Varchar(25)	Number(6)	Number(8)

Dengan tempat dan tanggal sebagai primary key, dan penggunaan dari tabel ini hanyalah ingin mengetahui biaya pada tempat dan tanggal tertentu. Maka akan ukuran data yang disimpan dapat dibuat lebih kecil dengan menyimpan nilai hash dari konkatenasi tempat dan tanggal. Misalkan fungsi hash yang digunakan menghasilkan 20 karakter, maka dari 31 karakter yang dibutuhkan untuk menyimpan data tiap tanggal dan tempat, kita dapat menghemat 11 karakter / data. Pada saat ingin dicari biaya pada tempat dan tanggal tertentu maka konkatenasi dari tempat dan tanggal dimasukkan ke fungsi hash baru kemudian dicocokan dengan nilai hash pada tabel.

Kegunaan lainnya dari hash key adalah untuk pencarian cepat pada basis data, terutama basis data yang berukuran besar dan/atau basis data dengan primary key bernilai mirip. Pencarian dengan metode hash key dilakukan dengan mencocokan hasil hash key yang ingin dicari dengan nilai hash yang disimpan dalam tabel. Pencarian menjadi lebih cepat, sebab apabila primary key bernilai mirip, misalkan NIM mahasiswa, sebagai contoh 13509012 dengan 13509010, maka 8 karakter akan dibandingkan barulah dapat diketahui bahwa nilai tersebut berbeda. Apabila yang dicocokan adalah nilai hash, maka pencarian dilakukan dengan lebih cepat sesuai dengan ciri-ciri fungsi hash, yaitu perubahan kecil menghasilkan nilai hash yang berbeda jauh. Hal tersebut dapat dilihat pada tabel di bawah ini.

NIM	Nilai Hash (SHA-1)
13509012	8EC4E9F2021A618F5ACE29BEC64 BE24B7C34BD67
13509010	29A5142E1C0410B79319234131F87 81964A6B0D8

Kedua nilai tersebut sudah dapat diketahui berbeda hanya dengan membandingkan karakter pertama dari nilai hash.

III. Manfaat Fungsi Hash pada Keamanan

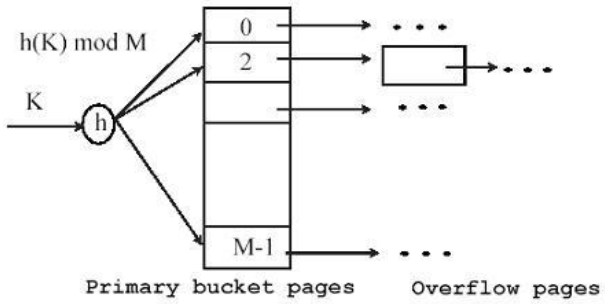
Kegunaan fungsi hash pada keamanan biasa dipakai terutama untuk mengamankan kata kunci. Pada basis data dengan tabel yang berisi username serta password dari anggota-anggota pada suatu situs web, seringkali password tidak disimpan pada tabel, tetapi yang disimpan adalah nilai hash dari password. Hal ini dapat mencegah seseorang yang mungkin dapat mengakses tabel ini diluar wewenangnya untuk dapat mengetahui password sebenarnya, akibat ciri fungsi hash yaitu nilai hash tidak dapat dikembalikan menjadi nilai yang menjadi masukan fungsi hash.

Selain mengamankan kata kunci, fungsi hash pun sering dijadikan alat otentikasi. Beberapa penerapan fungsi hash untuk otetikasi dapat terlihat dari penggunaan fungsi hash pada Digital Signature, Message Authentication Code, ataupun untuk memeriksa keaslian suatu file dengan mencantumkan nilai hash dari suatu file. Apabila nilai hash yang didapat dari file tersebut berbeda dengan nilai hash yang dicantumkan, maka dapat disimpulkan bahwa file tersebut tidak asli atau sudah rusak.

IV. Beberapa Teknik Fungsi Hash pada Basis Data

Ada beberapa macam teknik hashing yang berbeda yang digunakan pada basis data, maupun keamanan. Perbedaan dari kedua bidang tersebut adalah fungsi hash yang digunakan pada basis data akan cenderung lebih diutamakan kecepatannya dalam mencari data, sedangkan pada bidang keamanan lebih diutamakan teknik hash yang hampir tidak ditemukan adanya colission. Colission adalah keadaan di mana 2 string yang berbeda menghasilkan nilai hash yang sama.

Secara umum, teknik hash pada penentuan index basis data dibagi menjadi dua teknik, yaitu static hashing dan dynamic hashing.

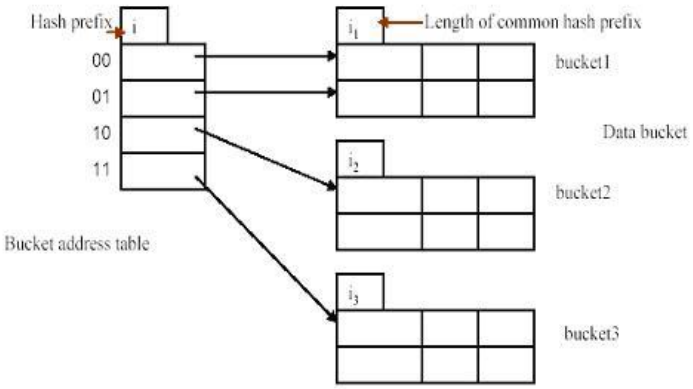


Gambar 1. Struktur Static Hashing
sumber:http://enggedu.com/tamilnadu/university_questions/question_answer/be_mj_2007/5th_sem/cse/CS1301/part_b/14_b_2.html

Static Hashing merupakan salah satu teknik untuk membangun tabel dengan ukuran yang tetap (static). Fungsi hash yang digunakan pada static hashing ini akan sangat bergantung pada ukuran tabel. Biasanya fungsi hash pada static hashing menggunakan modulo dari ukuran tabel dan relatif sederhana. Salah satu contoh fungsi hash pada static hashing adalah :

H(k) = (a*k+b) mod n

H = fungsi Hash
k = nilai key yang dicari
a & b = konstanta
n = ukuran tabel

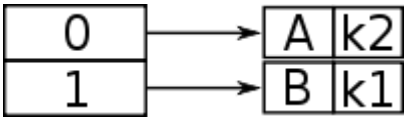


Gambar 2. Struktur Dynamic Hashing
sumber:http://enggedu.com/tamilnadu/university_questions/question_answer/be_mj_2007/5th_sem/cse/CS1301/part_b/14_b_2.html

Sebaliknya, dynamic hashing merupakan teknik membangun tabel dengan index yang ditentukan suatu fungsi hash dengan ukuran tabel yang dapat diperkecil atau diperbesar.

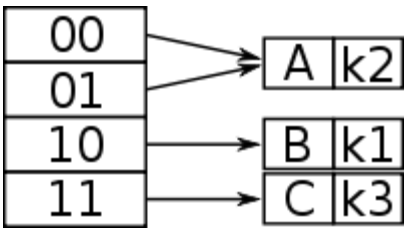
Dynamic Hashing dibagi menjadi 2 cara yang berbeda. Cara yang pertama disebut extendible hashing. Pada teknik ini, ada sebuah bilangan yang menyatakan ukuran maksimal tabel saat itu. Jika bilangan tersebut bernilai n, maka ukuran maksimal tabel saat itu adalah 2^n. Apabila tabel sudah penuh dan ada data baru yang ingin

ditambahkan ke dalam tabel, maka nilai n akan ditambah 1 sehingga ukuran maksimal tabel membesar. Pada proses pencarian data dengan hash, hanya n buah bit dari nilai hash yang dihasilkan yang akan berpengaruh pada pencarian. Contoh berikut akan menjelaskan cara kerja extendible hashing dengan lebih jelas.



Gambar 3. Contoh 1 Extendible Hashing
sumber:http://en.wikipedia.org/wiki/Extendible_hashing

Misalkan pada awalnya nilai n adalah 1. Seperti yang terlihat pada gambar di atas, tabel berukuran 2 dan keduanya telah terisi nilai. Maka ketika ingin ditambahkan data baru ke dalam tabel nilai n ditambah dan bentuk tabel akan menjadi seperti berikut.



Gambar 4. Contoh 2 Extendible Hashing
sumber:http://en.wikipedia.org/wiki/Extendible_hashing

Jumlah bit index akan bertambah 1. Index pada gambar di atas adalah 2 bit pertama dari nilai hash yang dihasilkan. Jadi apabila bit nilai hash yang ingin dicari adalah 01001011 maka data yang dicari adalah data A karena 2 bit pertama dari nilai hash tersebut adalah 01 dan index 01 menunjuk pada data A.

Cara dynamic hashing yang kedua disebut sebagai linear hashing. Sebenarnya cara ini mirip dengan extendible hashing. Perbedaanya pada linear hashing ukuran maksimal tabel bukanlah 2^n melainkan $K*2^n$. Di mana K adalah sebuah konstanta. Linear hashing akan menggunakan rumus :

$$index(level, key) = hash(key) \bmod (K * 2^{level})$$

Rumus tersebut akan memeriksa terlebih dahulu apakah nilai level lebih besar dari nilai n saat itu. Jika lebih besar, maka sebuah tabel baru akan dibuat pada index bernilai index(level, key). Sebagai contoh apabila ada hash(key) menghasilkan nilai 7. Nilai n pada tabel saat itu adalah 1 dan nilai K yang digunakan adalah 1, maka tabel saat itu berukuran maksimal $(1 * 2^1) = 2$. Kemudian untuk menambahkan sebuah data baru maka diakses fungsi index(2, key) = 7 mod $(1 * 2^2) = 3$. Maka data baru tersebut akan ditambahkan pada tabel dengan index 3.

Penggunaan fungsi hash dalam pembangunan tabel pun kadang terjadi masalah. Masalah yang sering terjadi adalah colission, terutama pada penggunaan fungsi hash untuk skala kecil yang digunakan pada basis data berukuran besar. Beberapa teknik untuk mengatasi hal ini pun telah ditemukan. Salah satu teknik untuk mengatasi colission pada saat akan menambahkan data baru pada hash tabel adalah linear probbing. Cara kerja Linear Probbing adalah sebagai berikut.

1. Jika index yang hasilkan fungsi hash kosong, maka taruh data baru pada index tersebut.
2. Jika sudah terisi, maka kembali ke cara no 1 dengan index yang diperiksa adalah (index+i)mod n.

Nilai n merupakan ukuran tabel, dan i adalah suatu konstanta yang biasanya relatif prima dengan n.

Teknik lainnya dalam menangani colission adalah dengan teknik double hashing. Cara kerja Double Hashing mirip dengan cara kerja Linear Probbing, hanya berbeda pada konstanta i yang diganti dengan fungsi hash lain. Secara garis besar cara kerja Double Hashing adalah sebagai berikut.

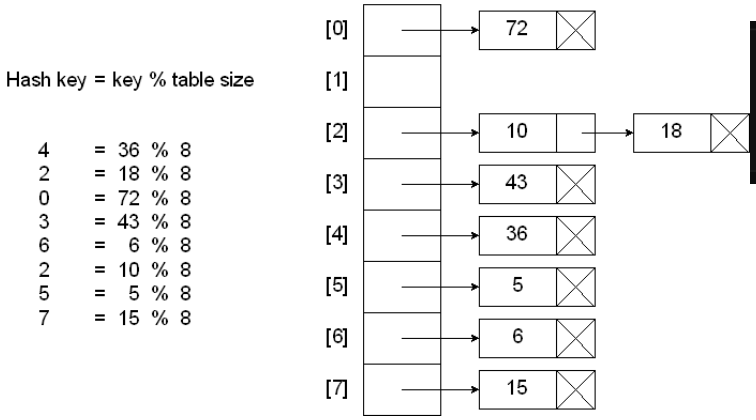
1. Jika index yang hasilkan fungsi hash kosong, maka taruh data baru pada index tersebut.
2. Jika sudah terisi, maka kembali ke cara no 1 dengan index yang diperiksa adalah (index+hash2(key))mod n.

Fungsi hash2 merupkan fungsi hash yang berbeda dengan fungsi hash yang menghasilkan index.

Selain kedua cara tersebut, ada juga cara yang cukup berbeda dalam mengatasi masalah collision yang disebut sebagai teknik Chained Hashing.

Pada teknik ini, diperlukan tabel dengan struktur data list berkait untuk implementasinya. Cara kerja teknik ini cukup sederhana, yaitu apabila terjadi collision pada index tertentu, maka tabel pada index tersebut akan dikaitkan dengan tabel baru dengan index yang bernilai key yang digunakan pada fungsi hash. Jadi, apabila suatu index sering terjadi collision, maka tabel pada index tersebut akan menjadi list berkait yang panjang, sebab apabila suatu index yang menjadi collision ternyata telah dikaitkan dengan tabel lain, maka tabel baru akan dikaitkan menjadi elemen paling terakhir dari list berkait tersebut.

Untuk lebih jelasnya, maka contoh gambar berikut akan lebih merepresentasikan seperti apa Chained Hashing tersebut.



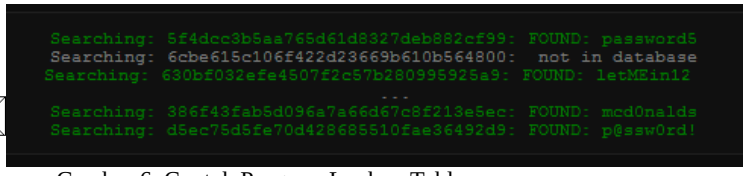
Gambar 5. Chained Hashing
sumber:http://faculty.cs.niu.edu/~freedman/340/340notes/340hash.htm

Pada gambar di atas, dapat terlihat bahwa setiap kali terjadi collision maka list berkait pada index tersebut akan ditambahkan 1 elemen pada bagian akhir list.

V. Beberapa Teknik Fungsi Hash pada Keamanan

Teknik hashing pada bidang keamanan pun beragam. Dari berbagai teknik hash pada bidang keamanan akan dibahas dua teknik diantaranya, yaitu double/multiple hashing serta adding salt.

Salah satu cara metode kriptanalis untuk memecahkan password adalah dictionary attack. Metode ini adalah metode dengan mencoba-coba kata kunci yang tepat yang ada pada suatu kamus kata. Satu per satu kata dicoba untuk dicari kata kunci yang tepat. Namun pada saat ini seringkali sistem sudah diamankan dari dictionary attack ini dengan cara memberi batas berapa kali suatu pengguna dapat mencoba kata kunci yang salah sebelum diblokir. Mengatasi cara tersebut, para cracker pun mempunyai cara tersendiri dalam mencari kata kunci. Apabila cracker dengan suatu cara dapat mengakses tabel tempat nilai hash dari kata kunci disimpan, maka para cracker dapat mendaftarkan beberapa akun pada sistem tersebut, misalkan sistem yang ingin dibobol adalah suatu website. Dengan akun-akun yang didaftarkan oleh cracker, maka cracker bisa mendapatkan fungsi hash yang dipakai pada website tersebut dengan cara mencoba-coba kata kunci yang dia daftarkan dengan membandingkannya dengan nilai hash yang ia dapat dari tabel tempat nilai hash dari kata kunci disimpan. Cracker bisa mencoba-coba beberapa fungsi hash yang terkenal. Apabila cracker mendapatkan fungsi apa yang dipakai pada website tersebut, maka yang perlu ia lakukan adalah melakukan hash pada setiap kata pada dictionary attack, kemudian mencocokkan nilai hash pada tabel tempat nilai hash dari kata kunci disimpan dengan nilai hash pada kata-kata dictionary attack. Cara seperti ini disebut juga sebagai metode Lookup Table.



Gambar 6. Contoh Program Lookup Table
sumber: http://crackstation.net/hashing-security.htm

Untuk mencegah hal tersebut, maka diperkenalkanlah teknik double/multiple hash. Sesuai dengan namanya, cara kerja double/multiple hash adalah dengan melakukan hash berkali-kali pada suatu kata kunci dengan fungsi-fungsi hash yang berbeda-beda. Dengan cara ini, cracker pun akan mengalami kesulitan dalam menentukan fungsi hash yang dipakai pada suatu sistem.

Teknik lainnya yang lebih sederhana adalah dengan adding salt. Yang dimaksud salt di sini adalah suatu string tambahan. Cara kerja adding salt adalah dengan menambahkan string tambahan pada akhir kata kunci sebelum dimasukkan ke dalam fungsi hash.

Salt didapat secara random dengan Cryptographically Secure Pseudo-Random Number Generator. Untuk lebih jelasnya akan diterangkan dengan contoh berikut.

Misalkan suatu situs menggunakan adding salt dalam proses penyimpanan kata kunci dari pengguna. Apabila ada pengguna baru yang mendaftar pada situs tersebut, maka ketika kata kunci ingin disimpan, sistem akan menggenerate salt. Kemudian kata kunci yang disimpan pada tabel adalah hasil dari fungsi hash(kata kunci + salt). Setelah itu salt pun disimpan. Salt akan lebih aman disimpan di tabel yang berbeda dengan kata kunci.

Dengan penambahan salt, maka cracker pun tidak dapat mengetahui fungsi hash yang dipakai pada situs tersebut, sebab cracker akan memerlukan nilai salt untuk mengetahui fungsi hash yang ada pada situs tersebut.

Selain pada saat mendaftar, supaya lebih aman, pada saat pengguna ingin mengganti kata kunci miliknya, maka pada saat menyimpan kata kunci baru akan lebih baik salt yang digunakan pun digenerate lagi.

VI. Analisis Perbandingan Peforma Teknik-Teknik Hashing

A. Static Hashing Vs Dynamic Hashing

Perbandingan dari static hashing dan dynamic hashing sebenarnya sudah dapat terlihat jelas. Jika kedua teknik ini digunakan pada suatu tabel yang berukuran tetap, maka static hashing akan sedikit lebih baik. Hal tersebut disebabkan oleh karena static hashing telah menyiapkan tabel sejumlah dengan data yang diperlukan, sedangkan pada dynamic hashing setiap kali data baru ditambahkan barulah tabel baru dibuat untuk menampung data baru yang ditambahkan tersebut. Sedangkan pada tabel dengan ukuran yang sering berubah-ubah, maka dynamic hashing akan jaul lebih baik. Setiap kali tabel kurang besar, maka

pada static hashing tabel perlu diatur ulang, serta mengubah fungsi hash yang digunakan. Apabila static hashing menyediakan tabel yang besar untuk persiapan apabila banyak data akan ditambahkan tentu akan memakan banyak memory pada apabila ternyata hanya sebagian dari ukuran tabel yang diperlukan.

B. Linear Probing Vs Double Hashing Vs Chained Hashing

Pemecahan masalah collision yang dibahas pada makalah ini adalah Linear Probing dan Double Hashing, serta Chained Hashing. Secara sekilas, cara kerja Linear Probing dan Double Hashing sangatlah mirip. Perbedaan kedua teknik tersebut hanya terletak pada Linear Probing menggunakan suatu konstanta untuk setiap kali penambahan, sedangkan Double Hashing menggunakan nilai fungsi hash lain untuk setiap penambahan nilai index. Sedangkan pada Chained Hashing, setiap kali penambahan dilakukan, maka akan ditambahkan elemen baru pada list berkait. Tidak seperti kedua cara sebelumnya, Chained Hashing tidak mencari tempat kosong, tetapi membuat tabel baru untuk mengisi data yang terjadi collision.

Untuk melihat perbedaan peforma dari ketiga teknik tersebut, tabel berikut akan lebih memperjelas peforma masing-masing teknik.

Load factor (alpha)	Open addressing with linear probing (1+1/(1-alpha))/2	Open addressing with double hashing -ln(1-alpha)/alpha	Chained hashing 1+alpha/2
0.5	1.50	1.39	1.25
0.6	1.75	1.53	1.30
0.7	2.17	1.72	1.35
0.8	3.00	2.01	1.40
0.9	5.50	2.56	1.45
1.0	N/A	N/A	1.50
2.0	N/A	N/A	2.00
4.0	N/A	N/A	3.00

Tabel 1. Perbandingan peforma Linear Probing, Double Hashing, Chained Hashing
sumber:http://www.cs.bu.edu/teaching/cs113/spring-2000/hash/

Pada Tabel di atas, nilai Load factor (alpha) merupakan perbandingan antara jumlah tabel yang telah terisi dibanding ukuran tabel. Nilai-nilai pada kolom Linear Probing dan Double Hashing merupakan perbandingan peforma keduanya dalam segi waktu.

Dapat terlihat bahwa pada nilai Load factor yang kecil peforma kedua teknik pertama tersebut mirp, akan tetapi setelah nilai Load Factor membesar, yaitu saat tabel hampir penuh, dapat terlihat bahwa double hashing hampir bekerja dua kali lebih cepat daripada Linear Probing.

Hal tersebut dapat dijelaskan dengan masalah yang sering terjadi pada Linear Probing yaitu clustering. Masalah Clustering adalah masalah yang terjadi di mana apabila index yang dihasilkan fungsi hash telah terisi, maka akan membutuhkan waktu yang lama untuk menemukan tempat yang kosong. Hal tersebut dapat dijelaskan dengan contoh apabila terjadi collision pada index ke k, maka akan data akan diisi pada index k+i. Nilai i adalah konstanta pada Linear Probing. Jika terjadi colision pada index ke k lagi, maka data akan terisi pada index k+i+i. Dan jika hal ini terus terjadi, tentu pencarian tempat kosong akan menjadi semakin lama. Sedangkan pada double hashing, apabila terjadi collision pada index ke-k data akan diisi pada index k+h1. Nilai h1 adalah nilai dari fungsi hash2(key) pada fungsi Double Hashing. Setelah itu apabila terjadi collision pada index ke-k lagi, index yang digunakan adalah index k+h2. Nilai h1 dan h2 berbeda pada double hashing sebab walaupun sama-sama terjadi collision pada index ke-k namun nilai key yang menyebabkan collision tersebut berbeda, sehingga fungsi hash2(key) akan menghasilkan nilai yang berbeda. Maka dari itu teknik double hashing dapat mengatasi masalah clustering.

Sedangkan pada Chained Hashing dapat terlihat bahwa teknik ini memiliki peforma yang paling baik. Hal tersebut disebabkan oleh karena Chained Hashing akan selalu mendapatkan tempat di mana data baru akan dimasukkan oleh karena Chained Hashing selalu membuat tabel baru untuk setiap data yang mengalam collision.

Pada tabel di atas, nilai peforma Linear Probing serta Double Hashing tidak ada pada load factor di atas nilai 1, sebab ketika tabel penuh, maka kedua teknik tersebut tidak dapat digunakan lagi, sebab kedua teknik tersebut hanya mencari tempat kosong bukan membuat tempat baru untuk data baru yang mengalami collision.

C. Double/Multiple Hashing Vs Adding Salt

Kedua teknik ini sebenarnya digunakan untuk mencegah adanya cracker yang ingin mencoba membobol kata kunci dengan menggunakan metode Lookup Table yang telah dijelaskan di atas.

Hal yang mungkin dapat menjadi masalah pada keamanan dengan penggunaan teknik double/multiple hashing adalah ternyata cracker tetap masih dapat memecahkan fungsi dari double/multiple hashing. Dengan menggunakan program berbagai kombinasi fungsi hash yang tidak terlalu rumit dapat dengan mudah dipecahkan dalam waktu yang tidak terlalu lama. Selain

itu, penggunaan beberapa fungsi hash yang telah diketahui adanya colission pun menyebabkan fungsi double/multiple hashing tidak berguna. Sebagai contoh fungsi MD5 yang telah diketahui memiliki collision. Apabila double hashing yang digunakan adalah MD5(MD5(key)), maka penggunaan string penyebab collision pada MD5 akan menghasilkan nilai yang sama pada fungsi tersebut.

Kelemahan pada teknik double/multiple hashing di atas dapat ditanggulangi dengan teknik adding salt. Pada teknik ini, selama salt yang digunakan pada suatu kata kunci tidak diketahui, maka dapat dikatakan kata kunci tersimpan dengan aman. Namun, apabila cracker berhasil mendapat akses ke tempat di mana setiap nilai salt disimpan, maka cracker dapat segera membobol keamanan sistem tersebut.

Selain perbedaan di atas, kedua teknik di atas sebenarnya dapat dibobol dengan teknik Brute Force, yaitu mencoba satu per satu tiap kemungkinan kata kunci yang ada. Walaupun demikian, teknik Brute Force merupakan metode untuk membobol keamanan yang paling tidak efektif dan memakan waktu yang lama.

VII. KESIMPULAN

Penggunaan setiap fungsi hash memiliki kelebihan dan kekurangan tersendiri. Maka dari itu, jika dipertanyakan manakah teknik hash yang paling baik, maka hal tersebut tidak dapat ditentukan tanpa mempertimbangkan kondisi di mana fungsi hash tersebut akan dipakai.

Pada penggunaan static hashing dan dynamic hashing, dapat disimpulkan akan lebih baik menggunakan static hashing pada tabel yang tidak akan atau jarang berubah ukuran tabelnya dan gunakan dynamic hashing pada tabel yang sering berubah-ubah ukurannya.

Berdasarkan analisis di atas didapatkan bahwa penggunaan double hashing lebih efektif dibandingkan dengan linear probbing, namun kedua cara ini tidak seefektif chained hashing. Namun salah satu kelemahan chained hashing adalah tabel perlu direpresentasikan sebagai list berkait, serta apabila terjadi collision pada suatu index tertentu secara terus menerus, maka peforma teknik ini akan turun secara drastis, sebab akan terbentuk suatu list yang sangat panjang pada index tersebut. Oleh karena itu, dapat disimpulkan bahwa akan lebih baik apabila sebisa mungkin dalam perancangan tabel dengan hashing fungsi hash yang digunakan dapat meminimalis terjadinya collision, sehingga kasus seperti clustering dan list yang panjang pada chained hashing tidak terjadi.

Pada bidang keamanan penggunaan double/multiple hashing serta salt adding memiliki kelemahan tersendiri, namun kedua teknik tersebut ternyata memilki kelebihan yang dapat menutupi kelemahan teknik satunya, sehingga apabila kedua teknik tersebut digunakan bersama-sama, tingkat keamanannya dapat meningkat.

REFERENCES

[1] http://www.simple-talk.com/sql/t-sql-programming/intelligent-database-design-using-hash-keys/.
Diakses tanggal 13 Mei 2012.

[2] http://enggedu.com/tamilnadu/university_questions/question_answer/be_mj_2007/5th_sem/cse/CS1301/part_b/14_b_2.html.
Diakses tanggal 11 Mei 2012.

[3] http://www.cs.bu.edu/teaching/cs113/spring-2000/hash/
Diakses tanggal 11 Mei 2012.

[4] http://crackstation.net/hashing-security.htm
Diakses tanggal 13 Mei 2012.

[5] http://faculty.cs.niu.edu/~freedman/340/340notes/340hash.htm
Diakses tanggal 13 Mei 2012

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 14 Mei 2012

ttd

Freddi Yonathan / 13509012