# Implementasi Fungsi *Hash* Satu Arah pada Optimasi Data Graf

Aya Aurora Rimbamorani/13515098
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*aya.aurora25@gmail.com*

**Abstrak—Fungsi *hash* umum digunakan dalam memudahkan pencarian data pada basis data. Fungsi *hash* yang digunakan pada pencarian data dan fungsi *hash* yang digunakan pada kriptografi memiliki perbedaan. Fungsi *hash* pada data tentu dibuat sedemikian rupa sehingga *indexing* pencarian data menjadi lebih mudah. Namun, penggunaan fungsi *hash* pada data ini kerap kali terjadi kolisi atau terdapat lebih dari satu data yang memiliki nilai fungsi *hash* yang sama. Berbeda dengan data umumnya, data graf tidak perlu memperhatikan *indexing* nilai fungsi *hash* tersebut dalam memudahkan pencarian. Oleh karena itu, pada makalah ini akan dibahas mengenai penerapan fungsi *hash* satu arah pada kriptografi pada optimasi data graf baik dari segi penyimpanan data maupun pencarian data.**

**Kata Kunci—Data graf, Fungsi *hash* MD5, Fungsi *hash* SHA-256.**

## I. Pendahuluan

Penyimpanan data dalam bentuk graf merupakan suatu hal yang tidak lagi asing pada era *Big Data* ini. Penggunaan data graf ini umumnya digunakan untuk menyimpan data yang memiliki keterhubungan tertentu. Sebagai contoh, data hubungan pertemanan pengguna sosial media, data sitasi suatu jurnal, dan data jaringan pada suatu wilayah tertentu.

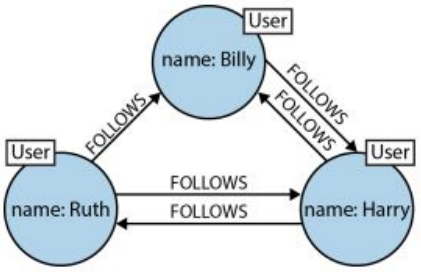
Penyimpanan dan pencarian data graf besar dapat memakan waktu yang cukup lama dan membutuhkan memori yang besar. Hal tersebut dikarenakan banyaknya data yang harus disimpan pada basis data yang mengakibatkan proses penyimpanan dan pencarian membutuhkan waktu yang relatif tidak sebentar. Dengan adanya tantangan tersebut, terdapat banyak algoritma yang diciptakan guna mempercepat proses penyimpanan dan pencarian data graf.

Pada makalah ini, akan diulas penggunaan fungsi *hash* satu arah guna mendapatkan performa yang baik pada proses penyimpanan dan pencarian data graf. Dengan memanfaatkan nilai fungsi *hash* dari setiap data yang akan disimpan dalam bentuk data graf ini, diharapkan besarnya data yang perlu disimpan dan lamanya waktu pencarian pada data menjadi lebih sedikit dan lebih cepat. Beberapa fungsi *hash* yang akan digunakan antaranya adalah fungsi *hash* MD5 dan SHA-256. Dari penerapan kedua fungsi *hash* tersebut, akan dilihat perbandingan kecepatan untuk melakukan penyimpanan dan pencarian data.

## II. Dasar Teori

### *A. Basis Data Graf*

Berbeda dengan basis data relasional yang menyimpan data-datanya dalam bentuk tabular, basis data graf merupakan basis data NoSQL yang merepresentasikan suatu data menjadi suatu simpul dan keterhubungan antar data tersebut menjadi sisi hubung. Sebagai contoh [1], data pada suatu media sosial yang direpresentasikan pada gambar II.1, entitas data user bernama Billy memiliki hubungan pertemanan (*follows*) dengan user bernama Harry yang juga melakukan hubungan pertemanan balik dengan Billy. Berbeda dengan hubungan pertemanan Billy dengan Harry, Ruth memiliki hubungan pertemanan satu arah dengan Billy.




Gambar II.1 Graf Pertemanan Media Sosial
Sumber : Robinson *et al,* 2015

### *B. Fungsi Hash Satu Arah*

Fungsi *hash* satu arah merupakan fungsi matematika yang digunakan untuk mendapatkan nilai tetap (nilai fungsi *hash*) suatu teks yang memiliki panjang beragam. Nilai fungsi *hash* dari setiap teks yang memiliki panjang beragam tersebut berbeda untuk setiap teks yang berbeda dengan panjang yang berbeda pula. Oleh karena itu, pergantian satu bit saja pada teks, akan menghasilkan nilai fungsi *hash* yang berbeda

(*avalanche effect*).

Fungsi *hash* satu arah yang baik tidak akan menghasilkan nilai fungsi *hash* yang sama untuk setiap teks berbeda atau dengan kata lain tidak terjadi *collision*. Selain itu, nilai fungsi *hash* yang didapat dengan menerapkan fungsi *hash* satu arah ini tidak dapat digunakan untuk mengembalikan teks asli atau bersifat *irreversible.*

### *C. Fungsi Hash MD5*

MD5 merupakan fungsi *hash* kriptografi yang memiliki besar nilai fungsi *hash* 128 bit [3]. Untuk mendapatkan nilai fungsi *hash*, masukan teks harus ditambahkan (*padding*) terlebih dahulu sehingga hasil modulo panjang bit teks tersebut dengan 512 adalah 448. *Padding* dilakukan dengan menambahkan bit-bit dengan nilai penambahan pertama adalah 1 dan bit setelahnya adalah 0.

```
word A: 01 23 45 67
word B: 89 ab cd ef
word C: fe dc ba 98
word D: 76 54 32 10
```

Tabel II.1 *Word-buffer*

Pada algoritma fungsi *hash* MD5 ini, terdapat 4 *word-buffer* yang digunakan yaitu A, B, C, dan D dengan masing-masing *word-buffer* memiliki panjang 32 bit. Tabel II.1 menampilkan inisialisasi dari 4 *word-buffer* tersebut. Selain *word-buffer*, fungsi *hash* MD5 juga memiliki 4 fungsi tambahan yang dieksekusi untuk mendapatkan nilai fungsi *hash.* Empat fungsi tambahan tersebut dapat dilihat pada persamaan di bawah ini dengan masukan berupa teks sepanjang 32 bit.

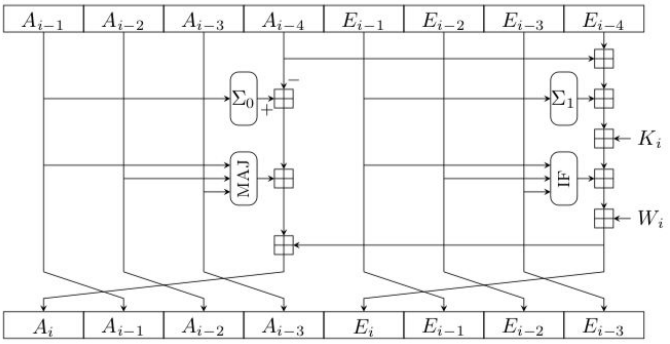
$$F(x,y,z) = (x \wedge y) \vee (\neg x \wedge z)$$
$$G(x,y,z) = (x \wedge y) \vee (y \wedge \neg z)$$
$$H(x,y,z) = x \oplus y \oplus z$$
$$I(x,y,z) = y \oplus (x \vee \neg z)$$

### *D. Fungsi Hash SHA-256*

SHA-256 merupakan fungsi *hash* kriptografi yang termasuk ke dalam kelompok fungsi *hash* SHA-2. Nilai fungsi *hash* yang dihasilkan dari SHA-256 adalah 256 bit dengan menggunakan fungsi kompresi [4]. Pada fungsi *hash* SHA-256 ini juga terdapat mekanisme *message expansion* dan *state update transformation* (gambar II.3). Pada *message expansion*, blok pesan (*message block*) sebesar 512 bit akan dibagi menjadi 16 kata (*words*) dan dari ke 16 kata tersebut kemudian akan diekspansi untuk mendapatkan 64 kata lain (*expanded message words*) *Wi* (gambar II.2). Pada kedua mekanisme *message expansion* dan *state update transformation* tersebut terdapat beberapa fungsi yang harus dijalankan yang dapat dipahami lebih lanjut pada [4].

$$W_i = \begin{cases} M_i & 0 \le i < 16 \\ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} & 16 \le i < 64 \end{cases}$$

Gambar II.2 *Expanded Message Words Wi*
Sumber : F. Mendel *et al*, 2013




Gambar II.3 *State Update Transformation*
Sumber : F. Mendel *et al*, 2013

## III. Rancangan Implementasi

### *A. Persiapan Data Graf*

Sebelum menjadi data graf, data merupakan data relasional yang memiliki atribut hubungan antar satu *tuple* data dengan *tuple* data yang lain. Dari data relasional tersebut, kemudian dibangun struktur data graf pada sistem basis data graf. Struktur data graf tersebut terdiri dari simpul dan sisi hubung dimana baik setiap simpul maupun sisi hubung dapat memiliki properti atau atribut tertentu. Setiap simpul akan menyimpan informasi dari setiap *tuple* dan setiap sisi hubung akan menyimpan informasi keterhubungan antar *tuple.*

| *id* | *title* | *author* | *year* | *ref* |
|---|---|---|---|---|
| 1 | Implementasi Fungsi Hash Sa.. | Aya A R | 2019 | 2, 100, 1020 |
| 2 | *Graph Database* | Robinson | 2015 | ... |
| ... | ... | ... | ... | ... |

Tabel III.1 Data Relasional Sitasi Artikel, Buku, dan Jurnal

Sebagai contoh pada tabel III.1, *tuple* dengan *id* 1 memiliki referensi terhadap *tuple* dengan *id* 2, 100, dan 1020. *Tuple* dengan *id* 1, 2, 100, dan 1020 tersebut kemudian akan direpresentasikan sebagai simpul-simpul. Berdasarkan tabel III.1 tersebut pula dapat dilihat bahwa *tuple* dengan *id* 1 merujuk artikel atau buku atau jurnal lain yang memiliki *id* 2, 100, dan 1020. Oleh karena itu, dari simpul dengan *id* 1 tersebut terdapat sisi hubung yang menghubung simpul dengan *id* 1 dengan simpul *id* 2, simpul *id* 100, dan simpul *id* 1020.

### *B. Implementasi Fungsi Hash*

Fungsi *hash* yang digunakan adalah fungsi *hash* MD5 dan fungsi *hash* SHA-256. Kedua fungsi tersebut akan menghitung nilai fungsi *hash* dari setiap *tuple* pada data relasional yang kemudian setiap nilai fungsi *hash* tersebut akan dijadikan simpul pada data graf. Pembentukan suatu simpul dengan fungsi *hash* tersebut dilakukan dengan menggabungkan nilai *string* dari setiap atribut pada data relasional dan kemudian dihitung fungsi *hash*-nya.

Sebagai contoh, pada tabel III.1, setiap atribut pada *tuple* dengan *id* 1 akan digabungkan dan membentuk teks sebagai berikut.

| |
|---|
| "1Implementasi Fungsi Hash Satu Arah pada Optimasi Data GrafAya A R201921001020" |

Tabel III.2 Teks Hasil Gabungan *tuple* dengan *id* 1

Dari teks tersebut kemudian dihitung nilai fungsi *hash* nya. Berikut merupakan nilai fungsi *hash* untuk teks pada tabel III.2.

| *MD5* | *SHA-256* |
|---|---|
| 7d473961c4e6b57fe1e3c753a79ecab2 | b82e54b8e31607a450a541d7761a60949c36dc2a35b6b97193ed6be18ee40085 |

Tabel III.3 Nilai Fungsi *Hash* dari Teks *tuple* dengan *id* 1

## IV. Eksperimen dan Analisis

### *A. Eksperimen dan Pra Proses Data Graf*

Eksperimen dilakukan menggunakan bahasa Python 3.6 dan basis data graf neo4j pada komputer bersistem operasi UNIX dengan spesifikasi memiliki besar RAM 4GB dan prosesor i5. Eksperimen pertama dilakukan untuk mengetahui lama waktu yang diperlukan untuk menyimpan *tuple* pada basis data graf dan eksperimen kedua dilakukan untuk mengetahui lama waktu pencarian data graf. Pencarian data graf dilakukan untuk mencari suatu simpul dengan atribut tertentu dan mencari keterhubungan suatu simpul dengan simpul lain.

Data yang digunakan merupakan data sitasi ACM yang berupa artikel, jurnal, dan buku di bidang ilmu komputer. Data tersebut merupakan *file* dengan format .txt. Selain itu, pada data tersebut pula terdapat struktur tersendiri untuk menyatakan setiap atribut (tabel IV.1). Pada *file* tersebut kemudian dilakukan pra proses berupa *parsing* untuk mendapatkan data dengan format seperti data relasional sehingga lebih mudah untuk diproses.

| |
|---|
| #*Approximating fluid schedules in crossbar<br>packet-switches and Banyan networks<br>#@Michael Rosenblum,Constantine Caramanis,Michel X.<br>Goemans,Vahid Tarokh<br>#t2006<br>#index17<br>#%357875<br>#%214023<br>#%... |

Tabel IV.1 Format Satu *tuple* pada Data ACM

Pada data ACM tersebut, setiap atribut memiliki awalan yang berbeda-beda. Atribut berupa judul diawali dengan '#*', atribut berupa penulis diawali dengan '#@', atribut berupa tahun diawali dengan '#t', atribut berupa index diawali dengan '#index', dan atribut berupa referensi ke *tuple* lain diawali dengan '#%'. Setelah dilakukan *parsing*, didapat data relasional dengan format seperti pada tabel III.1.

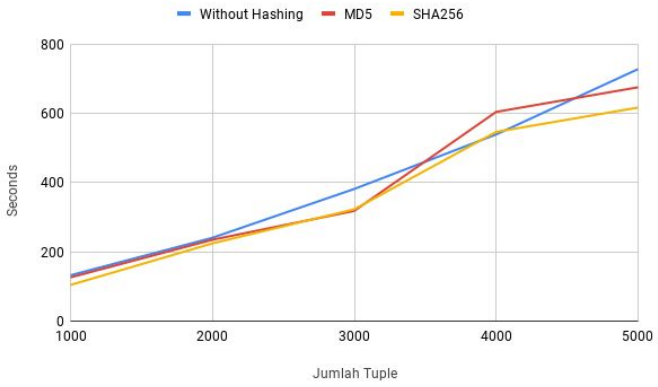
Pada penyimpanan data graf berdasarkan *tuple* tidak hanya menghasilkan simpul sejumlah data *tuple,* namun juga simpul-simpul referensi yang dirujuk oleh suatu *tuple.* Pada tabel IV.2 ditampilkan jumlah simpul dan sisi hubung (relasi) yang terbentuk dari eksperimen yang dilakukan terhadap data *tuple* sejumlah 1000, 2000, 3000, 4000, dan 5000.

| *Jumlah tuple* | *Jumlah simpul terbentuk* | *Jumlah relasi terbentuk* |
|---|---|---|
| 1000 | 5128 | 4983 |
| 2000 | 9165 | 9416 |
| 3000 | 12843 | 13338 |
| 4000 | 16431 | 17518 |
| 5000 | 19717 | 21782 |

Tabel IV.2 Jumlah *Tuple* dan Jumlah Simpul serta Sisi Hubung yang terbentuk

### *B. Analisis Kecepatan Penyimpanan Data Graf*

Berdasarkan hasil eksperimen, didapat hasil bahwa penggunaan nilai fungsi *hash* kriptografi sebagai atribut yang disimpan pada data graf memerlukan waktu yang lebih singkat dibandingkan dengan tidak menggunakan nilai fungsi *hash* kriptografi. Selain itu, dari gambar IV.1 dan tabel IV.3, dapat dilihat pula bahwa penggunaan fungsi *hash* SHA-256 memiliki waktu penyimpanan yang paling singkat.



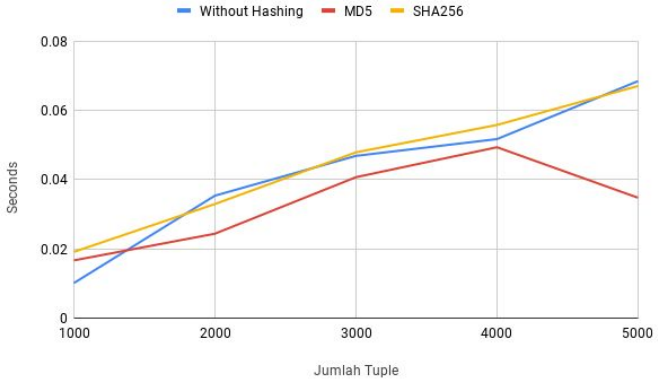

Gambar IV.1 Grafik Lama Waktu Penyimpanan Data Graf

| *Jumlah tuple* | *Tanpa Fungsi hash* | *MD5* | *SHA-256* |
|---|---|---|---|
| 1000 | 132.6425068 | 126.475522 | 104.5685902 |
| 2000 | 240.9423208 | 235.2694991 | 224.7323639 |
| 3000 | 382.079618 | 318.728714 | 323.2962413 |
| 4000 | 539.5782082 | 604.8726609 | 547.0740578 |
| 5000 | 728.4052439 | 675.9648228 | 617.3541431 |

Tabel IV.3 Lama Waktu Penyimpanan Data Graf (dalam *seconds*)

### *C. Analisis Kecepatan Pencarian Data Graf*

Pada eksperimen terhadap kecepatan pencarian data graf, hasil eksperimen terhadap pencarian simpul maupun keterhubungan simpul tidak memberikan hasil yang signifikan. Hal tersebut dapat dilihat pada hasil (gambar IV.2 dan gambar IV.3) ketika data *tuple* yang digunakan adalah 5000, hasil pencarian simpul maupun keterhubungan simpul dengan data graf nilai fungsi *hash* MD5 bekerja jauh lebih cepat.




Gambar IV.2 Grafik Lama Waktu Pencarian Simpul Data Graf


Gambar IV.3 Grafik Lama Waktu Pencarian Relasi antar Simpul Data Graf

## V. Kesimpulan

Berdasarkan hasil implementasi dan eksperimen yang dilakukan, dapat disimpulkan bahwa dengan menyimpan nilai fungsi *hash* kriptografi, terutama fungsi *hash* SHA-256, sebagai data graf, waktu penyimpanan data graf menjadi lebih cepat. Hal ini tentu dapat dipertimbangkan apabila jumlah simpul dan relasi yang hendak disimpan sangat banyak. Namun, penggunaan nilai fungsi *hash* kriptografi sebagai data simpul pada data graf tidak memberikan dampak yang signifikan pada proses pencarian data graf baik pencarian simpul maupun pencarian keterhubungan simpul (relasi).

## VI. Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena dengan bantuan dan rahmat-Nya penulis dapat menyelesaikan makalah ini. Tidak lupa penulis juga menyampaikan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir selaku dosen pengampu IF4020 Kriptografi karena dengan ilmu yang diberikan beliau penulis mampu memahami ilmu kriptografi. Selain itu, penulis juga ingin menyampaikan terima kasih kepada orang tua serta kerabat yang selalu mendukung penulis.

## Referensi

[1] Robinson, I., Webber, J., & Eifrem, E. (2015). *Graph Databases*. USA: O'Reilly Media, Inc.

[2] Anonymous. 2010. *Chapter-3: One-way Hash Function*. Retrieved from http://www.aspencrypt.com/crypto101_hash.html

[3] R. Rivest. "RFC 1321 - The MD5 Message-Digest Algorithm", MIT Laboratory for Computer Science and RSA Data Security, Inc., April 1992.

[4] F. Mendel, T. Nad, M. Schläffer, "Improving Local Collisions: New Attacks on Reduced SHA-256", Athens, Advances Cryptography EUROCRYPT, 2013, pp. 262-278.

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, 10 Mei 2019

Aya Aurora Rimbamorani/13515098