Negative Database Technique for Generic Database Security

Yudhistira Qasthari Putra, 18217003, Sistem Teknologi Informasi

Makalah Keamanan Informasi

***Abstract*—Saat ini, Data merupakan sebuah aset yang berharga. Mulai banyak tren tentang data tersebar di masyarakat seperti *Data science* dan *Big data*. Untuk menyimpan data, digunakan sebuah penyimpanan yang disebut *database*. *Database* digunakan untuk menyimpan data dalam format tertentu yang sudah memiliki standar. Salah satu *database* yang ada adalah *generic database*. *Generic* database adalah sebuah *database* yang menyimpan data dalam sebuah tabel yang terdiri dari *entity, attribute,* dan *value.* Keamanan dari *database* sangat diperlukan untuk menjaga data yang ada pada *database*. Keamanan data yang tidak baik dapat menyebabkan pengambilan data oleh pihak yang tidak bertanggung jawab. Salah satu teknik yang dapat digunakan adalah *negative database. Negative database* adalah *database* yang menyimpan data asli dan data palsu untuk menjaga keamanan dari data asli. Penggunaan *negative database* berguna untuk menjaga pencurian data dari *database*.**

Keywords*—makalah, keamanan informasi, *database security, negative database, generic database

I. PENDAHULUAN

A. *Introduction*

Akhir-akhir ini, data yang tersebar semakin banyak. Akses dan modifikasi terhadap data tersebut juga lebih sering dilakukan. Perlu adanya struktur data yang lebih baik untuk mendukung kebutuhan ini. Oleh karena itu, diperkenalkan sebuah model *database* yang fleksibel terhadap akses, modifikasi, maupun penambahan data. *Database* ini disebut *generic database. Generic database* menggunakan sebuah model struktur data yang disebut *entity-attribute-value model*. Alih-alih menyimpan data sebuah entitas dalam sebuah tabel sendiri, data disimpan dalam sebuah tabel dengan format yang standar yaitu *entity, attribute,* dan *value*.

Keamanan data pada sebuah *database* merupakan suatu hal yang penting. Banyak teknik yang dapat digunakan untuk meningkatkan keamanan data pada *database.* Teknik yang paling umum adalah enkripsi data. Selain itu, ada satu teknik yang dapat digunakan untuk meningkatkan keamanan *database,* yaitu *negative database*.

B. *Entity-Attribute-Value (EAV) Model*

Terdapat beberapa macam model yang dapat digunakan untuk sebuah *database*. Salah satu model yang dapat digunakan adalah *entity-attribute-value model* atau lebih singkatnya *EAV model*. *EAV model* merupakan sebuah model yang terdiri dari *entity, attribute,* dan *value.* Ide yang membentuk *EAV model* berasal dari konsep *association list* yang juga dikenal dengan istilah LISP. Pada sebuah *association list,* informasi disimpan dalam bentuk pasangan *key* dan *value*. Konsep ini dapat dibandingkan dengan komponen *attribute* dan *value* yang ada pada *EAV model* [1].

EAV model merupakan sebuah model yang sederhana dengan tabel yang terdiri dari tiga kolom. Informasi yang disimpan pada masing-masing kolom adalah sebagai berikut [2].

- *Entity* : Pada komponen *entity*, disimpan sebuah informasi yang menjelaskan tentang *item* data yang sedang digambarkan. Informasi yang disimpan pada entity adalah sebuah *foreign key* yang mengacu pada *primary key* dari sebuah tabel. Contoh : *Student-ID* yang merupakan *primary key* dari tabel *Student*
- *Attribute* : Pada komponen *attribute,* disimpan sebuah informasi tentang atribut atau ciri yang sedang digambarkan dari *entity*. Informasi pada *attribute* adalah *foreign key* yang mengacu pada atribut dari sebuah tabel. Contoh : *StudentName, Year, FatherName, MotherName,* dan *Telephone* dari tabel *Student*
- *Value* : Pada *value* disimpan nilai atau data dari *entity* dan *attribute* yang dijelaskan dalam sebuah *tuple*. Nilai dari *value* adalah nilai yang ada pada sebuah *tuple* untuk atribut yang sesuai dalam sebuah tabel. Contoh : “Andi” untuk *attribute StudentName,* “2017” untuk *attribute Year*, “Doni” untuk *attribute FatherName*, “Ani” untuk *attribute MotherName*, dan “081299999999” untuk *attribute Telephone* pada tabel *Student*.

EAV model adalah model yang cocok digunakan jika data yang disimpan bervariasi. Model ini cocok digunakan pada sistem aplikasi *e-commerce* dan bank [1].

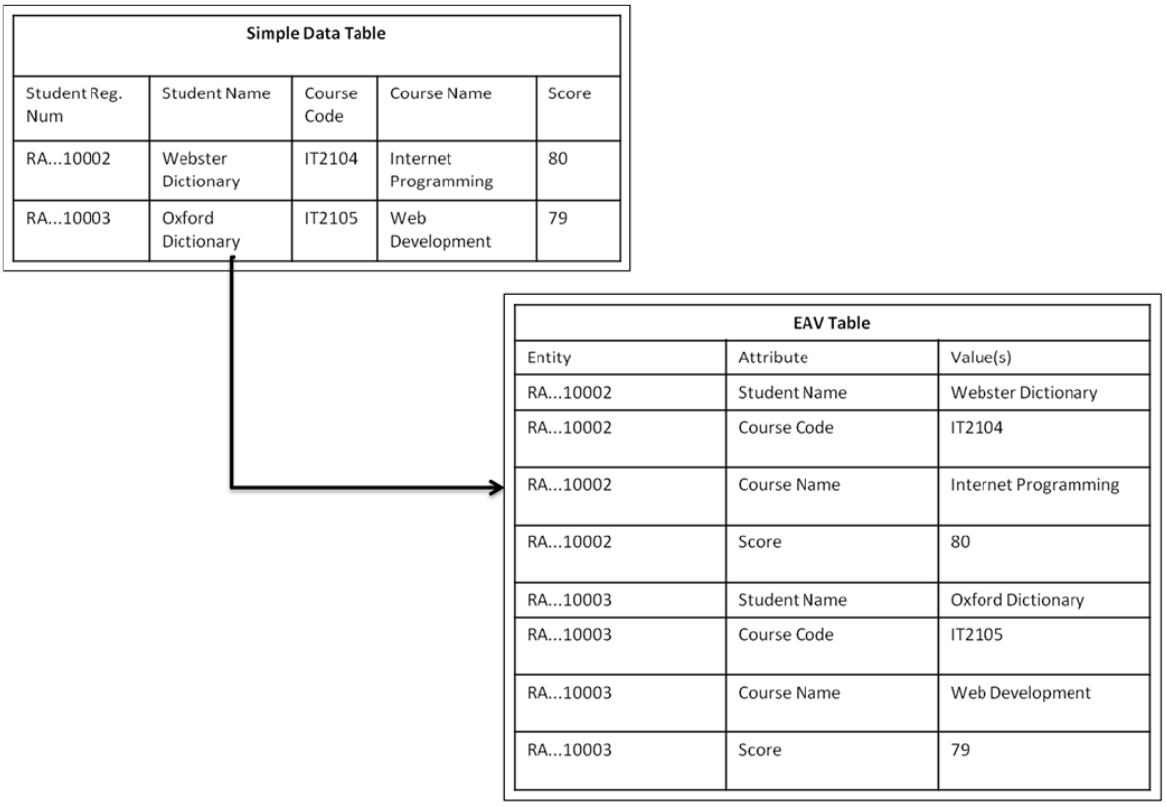
Tabel 1 Contoh *EAV Model*

Entity	*Attribute*	*Value*
18217099	*StudentName*	Andi
18217099	*Year*	2017
18217100	*StudentName*	Vina
18217100	*Year*	2017

C. *Generic Database*

Generic database adalah sebuah *database* yang menerapkan *EAV model* sebagai struktur data yang disimpan. *Generic database* menggunakan struktur data yang lebih umum jika dibandingkan dengan *database* konvensional yang biasa digunakan dalam *database* aplikasi. *Generic database* mendefinisikan sebuah relasi dengan standar yang umum [2]. Model yang digunakan pada *generic database* didasarkan pada *EAV model.*

Generic database, menawarkan fleksibilitas dalam penyimpanan data. Hal ini disebabkan oleh struktur data yang digunakan oleh *generic database*. Dengan fleksibilitas yang dimiliki, modifikasi data pada *generic database* dapat dilakukan lebih mudah. Modifikasi atau penambahan tipe data dapat dilakukan tanpa mengubah *physical schema* dari *database*.



Simple Data Table

Student Reg. Num	Student Name	Course Code	Course Name	Score
RA...10002	Webster Dictionary	IT2104	Internet Programming	80
RA...10003	Oxford Dictionary	IT2105	Web Development	79

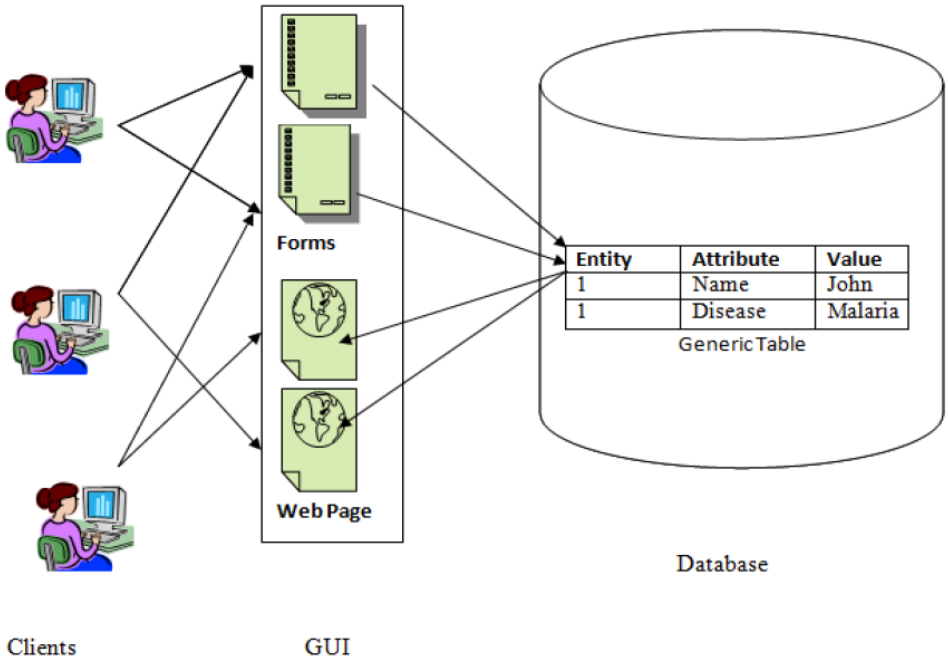
EAV Table

Entity	Attribute	Value(s)
RA...10002	Student Name	Webster Dictionary
RA...10002	Course Code	IT2104
RA...10002	Course Name	Internet Programming
RA...10002	Score	80
RA...10003	Student Name	Oxford Dictionary
RA...10003	Course Code	IT2105
RA...10003	Course Name	Web Development
RA...10003	Score	79

Gambar 1 Konversi *relational database* menjadi *generic database* [2].

Gambar 1 menjelaskan bagaimana konversi dari *relational database* menjadi *generic database*. Pada *relational database*, satu tabel menjelaskan sebuah entitas dan atribut dari entitas tersebut merupakan kolom dari tabel tersebut. *Value* dari data atau *tuple* pada *relational database* merupakan baris dari tabel. Pada *generic database*, tabel relasional diubah menjadi sebuah tabel yang lebih *general* atau umum. Tabel hanya memiliki kolom *entity*, *attribute*, dan *value*. Hal ini sesuai dengan *EAV model* yang sudah dibahas sebelumnya.

Proses akses data pada *generic database* juga lebih mudah jika dibandingkan dengan *relational database*.



Gambar 2 Akses pada *generic database* [1]

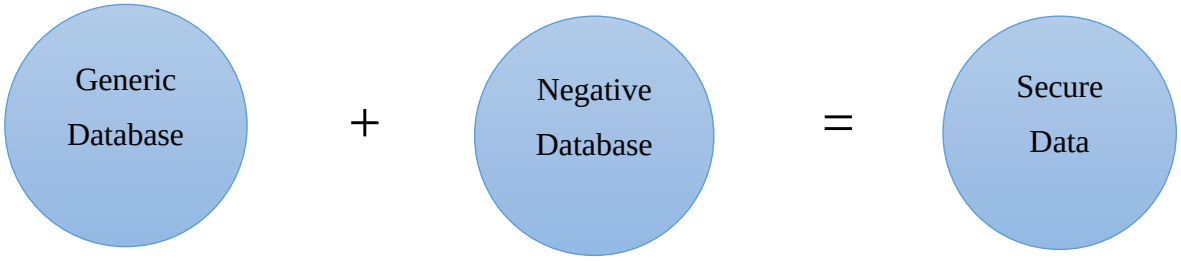
Pada gambar 2 diperlihatkan bahwa dengan akses pada *generic database* dapat dilakukan melalui berbagai macam *forms*. Semua *forms* hanya mengakses satu *database* yang sama sehingga penambahan data dapat dilakukan dengan lebih mudah [1].

D. *Negative Database*

Keamanan dari *database* merupakan sesuatu yang sangat penting. Data yang disimpan pada *database* ada yang mengandung informasi sensitif dari pemilik data. Hal ini akan berbahaya jika terjadi pencurian data oleh pihak yang tidak bertanggung jawab.

Dalam menjaga keamanan dari *database*, salah satu teknik yang dapat digunakan adalah *negative database*. Pada *database* normal (*positive*), data yang disimpan merupakan data asli yang sudah diencrypt dengan berbagai macam metode seperti RSA. Berbeda dengan *negative database*, data yang disimpan merupakan data dalam jumlah besar yang menyimpan data asli dan juga data palsu secara bersamaan [4]. Hal ini menyebabkan pengguna yang tidak memiliki hak akan memperoleh data yang palsu ketika mencoba melakukan pencurian data.

Negative database memberikan keamanan dengan menambahkan sebuah *layer* yang akan menyaring data sensitif yang masuk.



Gambar 3 *Negative database* dan *generic database* untuk keamanan data [2]

Negative database ditambahkan pada sebuah *generic database* untuk meningkatkan keamanan dari database tersebut. Dengan penambahan *negative database*, data dapat diakses oleh pengguna yang memiliki ha katas data tersebut.

II. PERUMUSAN MASALAH

Makalah ini akan membahas tentang penerapan dari *negative database* untuk meningkatkan keamanan dari *generic database*. Ada beberapa pertanyaan yang perlu dijawab untuk menyelesaikan permasalahan, yaitu :

- Bagaimana cara kerja *negative database* dalam memberikan keamanan kepada sebuah *generic database*?
- Apa hasil yang diberikan dengan penerapan *negative database* untuk *generic database*?

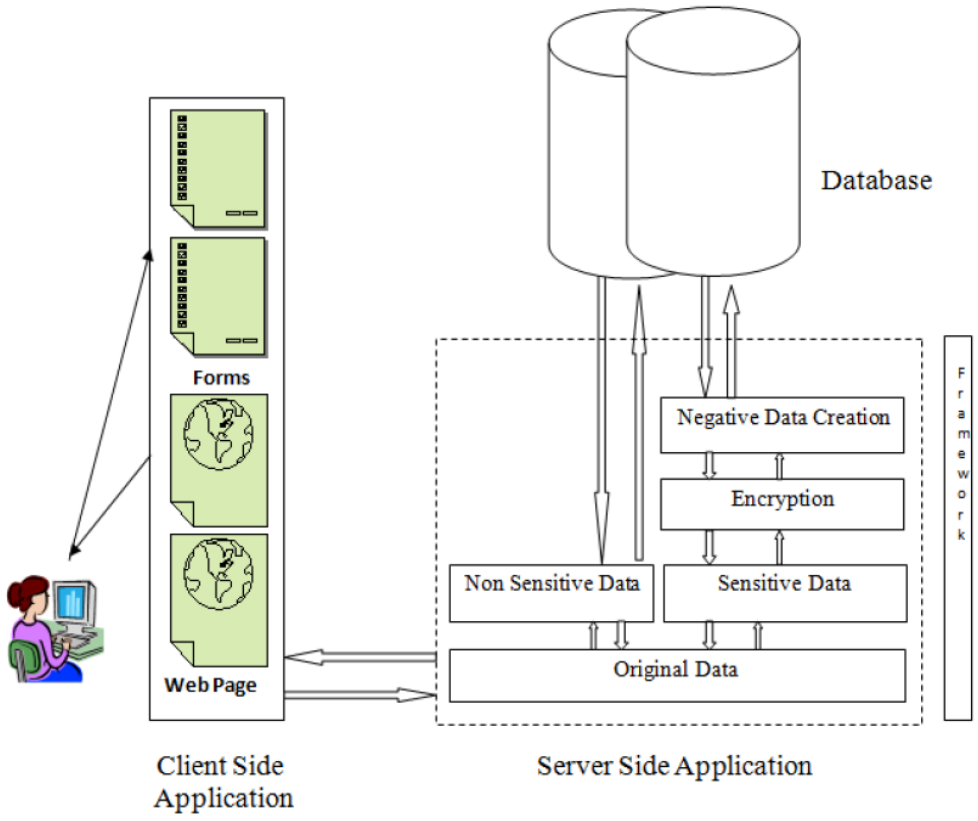
Dengan menjawab pertanyaan di atas, pemahaman mengenai penerapan *negative database* dapat diperoleh dengan baik.

III. IMPLEMENTASI

Dalam penerapan *negative database*, terdapat beberapa aspek yang perlu dipahami. Aspek-aspek yang perlu dipahami adalah Arsitektur dari *negative database*, pembuatan *negative database*, dan keluaran yang diberikan oleh *negative database*.

A. Arsitektur *Negative Database*

Dalam menerapkan *negative database*, akan terbentuk sebuah arsitektur sistem yang berbeda dari sebelumnya. *Negative database* menambahkan sebuah *layer* keamanan pada *server* untuk menyaring data yang sensitif sebelum dimasukan ke dalam *database*.



Gambar 4 Arsitektur *negative database* [1]

Dalam penerapan *negative database*, data akan dimasukan dalam berbagai bentuk *form* oleh *client* melalui aplikasi. Data tersebut akan dikirimkan ke *server* untuk diproses. Pada *server*, data akan disaring berdasarkan sensitif atau tidaknya data tersebut. Data yang tidak

bersifat sensitif akan langsung dikirimkan ke *database* untuk disimpan. Data yang sensitif akan melalui beberapa proses modifikasi untuk meningkatkan keamanan data.

Dalam proses modifikasi, data akan dienkripsi terlebih dahulu. Setelah dienkirpsi, data tersebut akan diproses pada *negative database*. Proses yang dilakukan adalah menggabungkan data dengan data palsu untuk meningkatkan keamanan data. Setelah melalui *negative database*, data yang sudah dimodifikasi dikirimkan ke *database* untuk disimpan.

Proses pengambilan data akan melewati tahap yang sama untuk masing-masing data. Data yang tidak sensitif akan langsung diambil dari database dan dikirimkan kepada *client.* Data yang sensitif akan diproses terlebih dahulu menjadi data asli sebelum dikirimkan kepada *client*. Data sensitif hanya akan diberikan kepada *client* yang memiliki hak akses terhadap data tersebut.

B. Implementasi *Negative Database*

Dalam melakukan modifikasi data menggunakan *negative database*, data dari *generic database* akan dipecah menjadi beberapa bagian. Ambil nilai n = 3 untuk melakukan pemecahanan data. Tabel dari *generic database* akan berubah menjadi *entity, attribute, value 1, value 2, value 3.*

Tabel 2 Tabel pada *generic database* setelah pemecahan data

Entity	*Attribute*	*Value 1*	*Value 2*	*Value 3*

Proses modifikasi data pada *negative database* akan melalui beberapa proses sebagai berikut.

1. Enkripsi data

Pada *server*, data yang dimasukan akan dienkripsi terlebih dahulu sebelum diproses pada *negative database*. Berikut adalah algoritma untuk melakukan enkirpsi data.

1. Let input=: sensitive data
2. Let a[ ]=: Toascii(input); /* The input is converted to its corresponding integer value by using it's ASCII value for every character of a given text. */
3. Encrypt []= RSA_Public_Key(a); /*Get the text to be encrypted and pass it through RSA_Public_Key().*/
4. Let cipher_text= double_hex(Encrypt); // converting encrypted text to base 32 to reduce its length.

Gambar 5 Algoritma enkripsi data pada *negative database* [1]

Dalam proses enkripsi, data yang sensitif akan dikonversi menjadi nilai ascii nya terlebih dahulu. Setelah konversi, data akan dienkripsi dengan menggunakan algoritma RSA. Kemudian hasil dari enkripsi dengan RSA dikonversi menjadi base 32 untuk mengurangi panjang teks.

2. Penggabungan data asli dan palsu

Setelah proses enkripsi, data yang asli akan digabungkan dengan data palsu pada *negative database*. Kemanan data dengan pemalsuan data akan lebih baik dari penggunaan enkripsi RSA saja. Jika *public key* hilang, data yang disimpan akan berada dalam risiko. Dalam proses pemalsuan data, algoritma yang digunakan adalah sebagai berikut.

1. Get the cipher_text input from the previous section and pass it through break_data().
2. Break_data()
    a. Assume a number "n" i.e. the number of value fields in the modified EAV model.
    b. Convert the cipher_text to the length that is multiple of "n" by appending zeros in the start.
    c. Split the cipher_text in "n" groups, each group as cipher_val[][] having n rows and (length/n) columns.
3. Generate_negative_data()
    a. For each row of cipher_val[][] from (i:=0) to (i<n) is to be stored in a different value field of a modified EAV generic model.
        i. Take a variable string Str="";
        ii. Take an integer q=0;
        iii. For each (q=0) to (q<(length/n))
            1. gr:= generated_chars(k); //Generate "k" random characters.
            2. Str=str+gr+cipher_val[i][q]; //concatenate
        iv. gr:= generated_chars(k);
        v. str=str+gr;
    b. We get "n" encrypted strings with some negative data, where the useful data is only at the position at multiple of "k+1" and the remaining data is negative.
4. We store these n strings as negative data to the negative database in different "n" fields.

Gambar 6 Algoritma untuk modifikasi data dengan penyisipan data palsu [1]

Pertama, data yang sudah dienkripsi diambil untuk dipecah menjadi n bagian. Pada kasus ini data akan dipecah menjadi n = 3 bagian. Jika panjang dari data tidak dapat dibagi dengan n, tambahkan 0 pada awal data hingga data dapat dibagi dengan n. Setelah dipecah, data palsu akan mulai disisipkan pada data asli.

Dalam menyisipkan data palsu, tetapkan sebuah nilai k yang merupakan jumlah karakter yang akan disisipkan pada data asli. Karakter ini akan menjadi data palsu yang memanipulasi nilai dari data asli.

Untuk setiap data baris data, definisikan variabel str = “” (*string* kosong). Kemudian *generate* k buah karakter sembarang dan gabungkan str dengan k karakter tersebut. Lalu, gabungkan str dengan karakter pertama dari data pada baris tersebut. Setelah itu, *generate* k karakter sembarangan lagi dan gabungkan dengan str sebelumnya. Lalu gabungkan Kembali str terakhir dengan karakter kedua pada data dengan str. Setelah itu gabungkan lagi k karakter sembarang dan lakukan berulang hingga karakter data pada baris tersebut habis. Lakukan proses ini untuk setiap baris pada data. Setelah karakter pada data habis, *generate* k karakter dan gabungkan str dengan k karakter tersebut untuk mengakhiri modifikasi data.

Setelah modifikasi data dilakukan, diperoleh n buah *string* yang merupakan penggabungan data asli dan palsu. Hasil *string* ini akan dikirimkan untuk disimpan pada *database*.

IV. ILUSTRASI DAN HASIL

Untuk memahami lebih dalam cara kerja dari *negative database,* dibuat sebuah ilustrasi dari penggunaan *negative database* dalam menyimpan suatu data. Contoh data yang akan disimpan terdiri dari 4 atribut yaitu *name*, *age*, *blood sugar (FV),* dan *blood sugar (PP)*.

Tabel 3 Data untuk ilustrasi

Entity	*Attribute*	*Value*
1	*Name*	Andi
1	*Age*	35
1	*Blood Sugar (FV)*	109
1	*Blood Sugar (PV)*	130

Pada tabel di atas, data yang dianggap data yang sensitif merupakan data *Blood Sugar (FV)* dan *Blood Sugar (PP). Name* dan *age* dianggap tidak sensitif sehingga akan langsung disimpan pada *database*.

Hal pertama yang dilakukan adalah melakukan enkripsi pada data sensitif. Data yang akan dienkripsi adalah nilai dari *Blood Sugar (FV)* dan *Blood Sugar (PP)*.

Hasil enkripsi dari dua data yang sensitif adalah 6S8D75 untuk 109 dan 065ERD untuk 130. Hasil enkripsi ini kemudian dibagi menjadi n baris. Pada kasus ini ambil n = 3. Hasil pemecahan dari kedua nilai adalah sebagai berikut

109	=	6S
		8D
		75
130	=	06
		5E
		RD

Hasil pemecahan ini kemudian akan dimodifikasi dengan menyisipkan data palsu pada masing-masing baris. Lakukan modifikasi data untuk nilai 109 terlebih dahulu.

Hal pertama yang dilakukan adalah *generate* k buah karakter sembarang dan masukan pada sebuah variabel str. Pada kasus ini ambil k = 4. Setelah itu gabungkan karakter 6 dengan str. *Generate* lagi 4 buah karakter sembarang dan gabungkan dengan str. Gabungkan nilai S dengan str. Data pada baris pertama dari 109 sudah habis. Oleh karena itu, *generate* 4 karakter dan gabungkan dengan str sebagai penutup. Lakukan hal ini untuk 2 baris lainnya dan 3 baris dari data 130.

Setelah modifikasi data dilakukan, diperoleh hasil sebagai berikut.

109		
6S	=	ASD7**6**ADFA**S**96ER
8D	=	6DF7**8**A6SD**D**77AS
75	=	LAKS**7**67AS**5**DA98
130		
06	=	ASD6**0**4A65**6**SD48
5E	=	WSD9**5**F879**E**ASA5
RD	=	D7GS**R**D6VS**D**6DF4

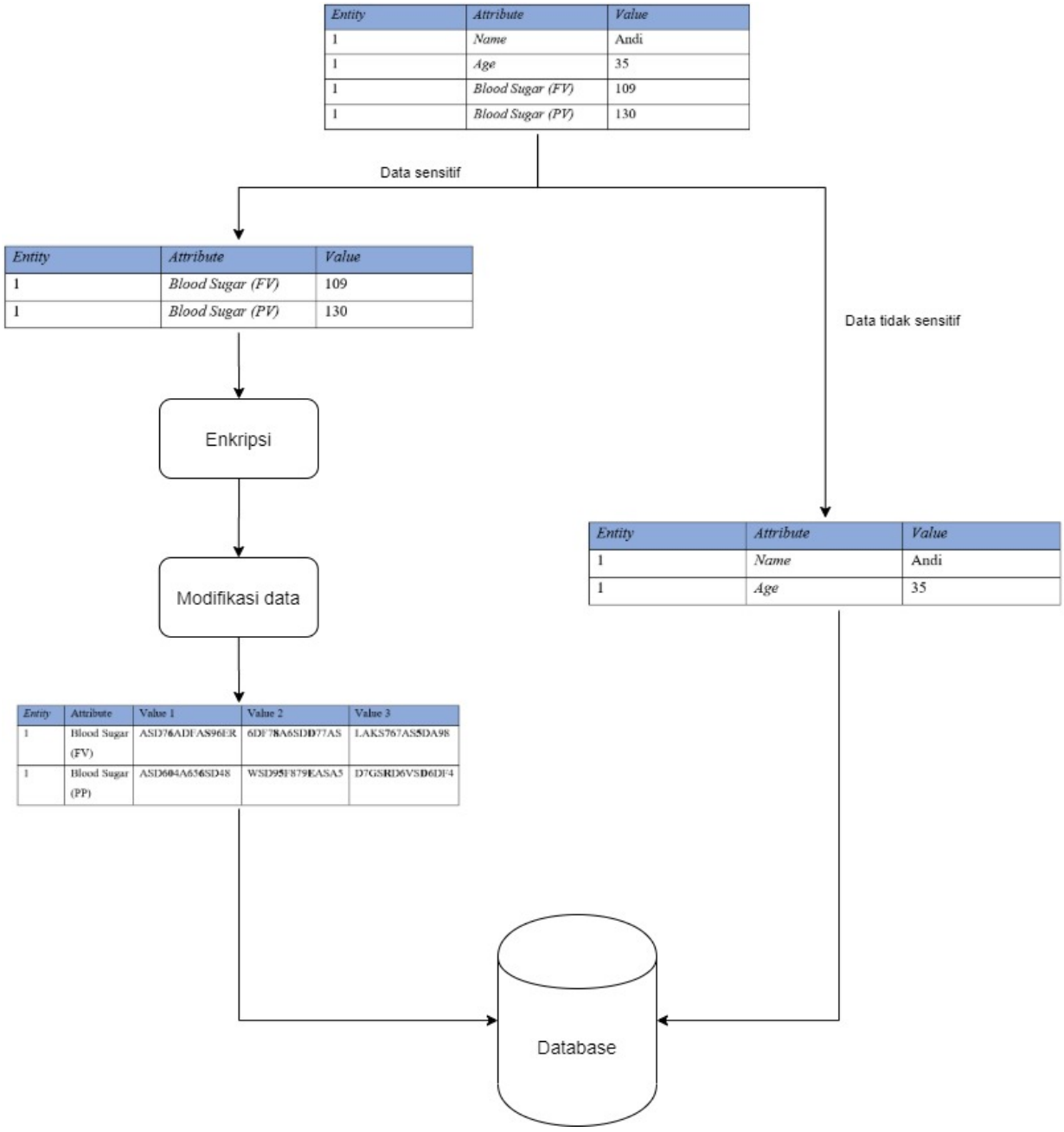
Data di atas kemudian dimasukan ke dalam sebuah tabel EAV dengan 5 kolom yang terdiri dari *entity, attribute, value 1, value 2,* dan *value 3.*

Tabel 4 Tabel EAV untuk data sensitif

Entity	Attribute	Value 1	Value 2	Value 3
1	Blood Sugar (FV)	ASD7**6**ADFA**S**96ER	6DF7**8**A6SD**D**77AS	LAKS**7**67AS**5**DA98
1	Blood Sugar (PP)	ASD6**0**4A65**6**SD48	WSD9**5**F879**E**ASA5	D7GS**R**D6VS**D**6DF4

Tabel di atas merupakan data palsu yang akan disimpan pada *database* untuk mencegah pencurian data. Dapat dilihat dari data yang disimpan, hanya data yang berada pada posisi kelipatan (k+1) yang merupakan data asli. Selain itu, data yang disimpan adalah data palsu yang mengecoh pengguna yang tidak berhak atas data tersebut.

Dari ilustrasi proses penyimpanan data di atas, dapat disimpulkan secara umum proses penyimpanan data dengan menggunakan *negative database* dapat digambarkan dengan gambar berikut ini.



Gambar 7 Ilustrasi penyimpanan data dengan *negative database* [1]

V. LAIN-LAIN TENTANG *NEGATIVE DATABASE*

A. Kelebihan *Negative Database*

Dalam penerapannya, *negative database* memiliki beberapa kelebihan. Berikut adalah kelebihan dari penggunaan *negative database*.

- Memberikan keamanan tambahan kepada *database* dari pengguna yang tidak memiliki hak atas data.
- Memberikan data palsu kepada pencuri data

- Memberi keamanan transfer data.
- Memberi keamanan lebih jika dibandingkan dengan enkripsi data saja

B. Kekurangan *Negative Database*

Walaupun *negative database* memberikan keamanan yang lebih baik, tetap ada beberapa kelemahan yang dimiliki. Berikut adalah kelemahan dari *negative database.*

- Algoritma dengan menyisipkan data palsu dengan panjang yang sama akan gagal jika panjang data palsu bisa ditebak oleh pencuri data
- Algoritma yang menyisipkan data dengan ukuran yang berbeda akan meningkatkan kompleksitas algoritma

C. *Negative Database* di Masa Depan

Konsep dari *negative database* dalam hal keamanan sudah cukup baik. Namun, masih banyak pengembangan yang harus dilakukan. Perlu dicari algoritma yang lebih baik untuk memberikan keamanan dari *database*. Algoritma yang lebih sulit dipecahkan dalam hal melakukan konversi dari data palsu ke data asli. Selain itu, penerapan *negative database* pada *database* juga belum dilakukan. Perlu adanya penelitian lebih lanjut agar *negative database* dapat diterapkan untuk meningkatkan keamanan dari data.

VI. KESIMPULAN

Dari penjelasan di atas, dapat ditarik beberapa kesimpulan mengenai penerapan *negative database* untuk *generic database*. *Negative database* merupakan sebuah alternatif teknik yang dapat digunakan untuk meningkatkan keamanan dari *database*. *Negative database* akan menyediakan sebuah *layer* keamanan tambahan. Pada *layer* tersebut, dilakukan modifikasi pada data asli dengan menyisipkan data palsu. Data yang dihasilkan dari *layer* ini kemudian disimpan pada *database*. Hal ini meningkatkan keamanan dengan mencegah pencuri data untuk mengambil data asli dari *database*. Dalam proses implementasi, algoritma yang paling mudah adalah dengan menyisipkan sejumlah karakter sembarang di antara data yang sudah dienkripsi. Penyimpanan data juga dipecah menjadi beberapa *value* untuk menambah tingkat keamanan dari data. Dengan penggunaan *negative database*, keamanan dari data pada *database* dapat ditingkatkan.

REFERENSI

[1] G. Dubey, V. Khurana and S. Sachdeva, "Implementing security technique on generic database," 2015 Eighth International Conference on Contemporary Computing (IC3), Noida, 2015, pp. 370-376.

[2] C. Egbunike and S. Rajendran, "The implementation of negative *database* as a security technique on a generic database system," 2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT), Kollam, 2017, pp. 1-8.

[3] F. Esponda, "Everything that is not important: Negative databases [Research Frontier]," in IEEE Computational Intelligence Magazine, vol. 3, no. 2, pp. 60-63, May 2008.

[4] A. Patel, N. Sharma and M. Eirinaki, "Negative Database for Data Security," 2009 International Conference on Computing, Engineering and Information, Fullerton, CA, 2009, pp. 67-70.

[5] T. Jastrow and T. Preuss, "The Entity-Attribute-Value Data Model in a Multi-tenant Shared Data Environment," 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), Krakow, 2015, pp. 494-497.