Volume 4 Nomor 1 Januari 2019

INFORMASI INTERAKTIF

JURNAL INFORMATIKA DAN TEKNOLOGI INFORMASI

PROGRAM STUDI TEKNIK INFORMATIKA – FAKULTAS TEKNIK -UNIVERSITAS JANABADRA

INFORMASI INTERAKTIF	Vol. 4	No. 1	Hal. 1 - 62	Yogyakarta Januari 2019	ISSN 2527-5240

INFORMASI INTERAKTIF

ISSN 2527-5240

DEWAN EDITORIAL

Alamat Redaksi : Program Studi Teknik Informatika Fakultas Teknik
Universitas Janabadra
Jl. Tentara Rakyat Mataram No. 55-57
Yogyakarta 55231
Telp./Fax : (0274) 543676
E-mail: informasi.interaktif@janabadra.ac.id
Website : http://e-journal.janabadra.ac.id/

Frekuensi Terbit : 3 kali setahun

JURNAL INFORMASI INTERAKTIF merupakan media komunikasi hasil penelitian, studi kasus, dan ulasan ilmiah bagi ilmuwan dan praktisi dibidang Teknik Iinformatika. Diterbitkan oleh Program Studi Teknik Informatika Fakultas Teknik Universitas Janabadra di Yogyakarta, tiga kali setahun pada bulan Januari, Mei dan September.

INFORMASI INTERAKTIF

ISSN 2527-5240

DAFTAR ISI

ISSN 2527-5240

PENGANTAR REDAKSI

Puji syukur kami panjatkan kehadirat Allah Tuhan Yang Maha Kuasa atas terbitnya JURNAL INFORMASI INTERAKTIF Volume 4, Nomor 1, Edisi Janauari 2019. Pada edisi kali ini memuat 9 (sembilan) tulisan hasil penelitian dalam bidang teknik informatika.

Harapan kami semoga naskah yang tersaji dalam JURNAL INFORMASI INTERAKTIF edisi Januari tahun 2019 dapat menambah pengetahuan dan wawasan di bidangnya masing-masing dan bagi penulis, jurnal ini diharapkan menjadi salah satu wadah untuk berbagi hasil-hasil penelitian yang telah dilakukan kepada seluruh akademisi maupun masyarakat pada umumnya.

Redaksi

IMPLEMENTASI *DATABASE SECURITY* MENGGUNAKAN KONSEP *ROLE-BASED ACCESS CONTROL* (RBAC) DALAM RANCANGAN *DATABASE* SISTEM INFORMASI MANAJEMEN SEKOLAH DENGAN POSTGRESQL

Achmad Yusron Arif[1], Ema Utami[2], Suwanto Raharjo[3]

[12] Magister Teknik Informatika, Univeristas AMIKOM Yogyakarta
[3] Teknik Informatika, FTI, IST AKPRIND Yogyakarta

Email : [1]achmad.arif@students.amikom.ac.id, [2]ema.u@amikom.ac.id, [3]wa2n@nrar.net

ABSTRACT

School management information system is an important application used to manage data and information of a school in order to provide the best service to the community. To create an effective and efficient information system, an effective database design is needed. This needs to be done because if the database is not designed properly, in the future when a system starts to develop, the integrity data stored in the database will decrease. One aspect that needs to be considered is database security. Database security is very important, because the data stored in the database is important information for schools, such as student, financial and employee data. One of the method that can be used to improve database security is Role-Based Access Control (RBAC), which is limiting access to users or roles according to their respective roles. In this study the RBAC method was applied in school management information systems using the PostgreSQL database.

Keywords: *Role Based Access Control, RBAC, Database Security, PostgreSQL, SIM Sekolah*

1. PENDAHULUAN

Perkembangan teknologi saat ini sangat cepat, sehingga mampu berkontribusi dalam segala bidang, mulai dari bidang komunikasi, informasi, pendidikan, militer, dll. Salah satu perkembangan dalam bidang teknologi informasi adalah *database*. Menurut Hartono (2016) *database* adalah sistem untuk menyimpan dan mengolah data. Konsep ini sama seperti media penyimpanan data konvensional yang masih menggunakan kertas kemudian disimpan dalam folder-folder yang berbeda sesuai dengan kategorinya. Kemudian, ada sebuah mekanisme dimana dalam suatu organisasi hanya orang-orang tertentu saja yang bisa mengakses file kertas dalam folder tersebut, misal hanya yang memiliki jabatan sekelas *manager* atau diatasnya. Konsep ini disebut dengan hak akses [1].

Hak akses ini adalah salah satu dalam konsep keamanan *database*. Keamanan menjadi masalah penting yang perlu diperhatikan oleh *database developer,* karena jika keamanan tidak dipertimbangkan akan memungkinkan *hacker* untuk merusak dan menurunkan integritas data yang disimpan dalam *database*. Menurut Ardiansyah, dkk (2016) *hacker* adalah orang yang dengan sengaja mempelajari dan memodifikasi suatu program dengan tujuan untuk mendapatkan keuntungan atau hanya sekadar tertantang [2]. Sebagai contoh, jika suatu *database* perusahaan berhasil dibobol maka data perusahaan yang bersifat rahasia akan dengan mudah dimanipulasi dan dicuri sehingga akan mengakibatkan kerugian yang tidak sedikti.

Dalam perkembangan keamanan *database* dikenal konsep *Role-Based Access Control* (RBAC). Menurut Dongdong, dkk (2017) RBAC adalah metode untuk pembagian akases dalam suatu *database*. Kata *role* disini adalah kumpulan hak akses yang bisa diberikan kepada suatu *user*. *User* dapat memiliki relasi *many to many* dengan *role*, dan relasi *role* dengan *permission* juga *many to many*. Dalam konsep RBAC ini memungkinkan dibuat sebuah *role* dengan *permission* tertinggi, sehingga *user* yang masuk dalam *role* tersebut bisa membuat *role* dan *user* lain dengan tingkat *permission* sama dengan atau dibawahnya [3]. Contoh kasus masalah ini adalah dalam sistem informasi manajemen sekolah, dalam sistem informasi tersebut ada beberapa pihak yang terlibat misal siswa dan karyawan bagian keuangan. Dalam *database* biasa hanya terdapat satu *user* saja dengan *default* nama *user* adalah *root*. Apabila *database* berhasil dibobol oleh *hacker* dengan

mengakses *user root* maka *hacke* bisa dengan mudah merusak data yang ada pada sistem manajemen sekolah. Dengan menggunakan metode RBAC maka dibuat *role* dalam *database* yaitu *role* siswa dan *role* karyawan. Dengan asumsi bahwa jumlah *user* dalam *database* sama dengan jumlah siswa dan karyawan. Jika *role* ini diaktifkan maka siswa hanya bisa mengedit tabel yang diberi akses, begitu juga dengan karyawan. Jadi, dengan metode ini jika terjadi pembobolan pada *user* siswa maka akan lebih aman, karena *hacker* hanya bisa mengedit tabel dengan *role* siswa bukan karyawan.

Atas latar belakang tersebut maka dalam penelitian ini akan dijelaskan mengenai konsep dan penerapan RBAC dalam sistem informasi manajemen sekolah dengan menggunakan *database* PostgreSQL.

2. TINJAUAN PUSTAKA

Menurut Wilkinson (1992) data dapat didefinisikan sebagai fakta, angka dan simbol yang mentah untuk dijadikan sebagai suatu *input* dalam sebuah sistem informasi. Sedangkan informasi ialah data-data yang telah diolah dan telah diketahui nilai kebenarannya. Dari dua definisi tersebut maka diketahui definisi dari *database* yaitu kumpulan *record* atau data yang saling berhubungan satu sama lain. Sedangkan *database relational* ialah kumpulan tabel yang saling berhubungan yang menggambarkan dirinya sendiri [4]. Untuk saat ini ada banyak *database* yang bisa digunakan baik bersifat *open source* maupun berbayar. Contoh *database open source* yang populer adalah MySQL dan PostgreSQL. Dalam penelitian ini menggunakan *database* PostgreSQL karena dalam MySQL belum mendukung metode keamanan RBAC.

PostgreSQL termasuk *Object Relational Database Management System* (ORDBMS) yang pertama kali dikembangkan oleh Berkely Computer Science Department di tahun 1977-1985. Kemudian, seiring berjalannya waktu PostgreSQL mengalami perkembangan yang signifikan, dimulai pada tahun 1996 dengan diterapkannya standard diatas ANSI-SQL92 dan mampu memenuhi syarat kebutuhan sebuah *database opensource*, maka PostgreSQL mampu bersaing dengan *database* besar lain seperti Oracle. Kemudian sampai saat ini PostgreSQL memiliki fitur seperti *type, class, rule, RBAC, row-level security, function* [5].

Salah satu yang membuat PostgreSQL menjadi *database* yang populer adalah kehandalannya dalam mengolah data dengan skala besar tetapi memberikan keamanan yang lebih baik jika dibandingkan dengan MySQL. Seperti yang telah dijelaskan pada pendahuluan, keamanan sangat penting untuk diperhatikan untuk mencegah *hacker* merusak data yang ada pada *database*. Menurut Connoly, dk (2005) *user* adalah salah satu celah keamanan yang perlu diperhatikan, karea itu harus diperhatikan basis kontrolnya seperti : (1) Hak istimewa untuk mengakses obyek dalam *database*. (2) *Access control* yaitu teknik keamanan untuk mengatur siapa dan jadi apa suatu *user* dalam *database*. (3) *Views*, metode untuk membatasi pengguna melihat data. (4) *Backup and Recovery* yaitu menyimpan dan mengembalikan data untuk mengantisipasi terjadinya kerusakan dalam *database* [6].

Pada intinya RBAC adalah konsep untuk membagi *role* atau hak akses kepada setiap *user* dalam suatu *database*. Tujuan dari pembagian *role* ini adalah untuk meminimalisir jika salah satu *user* dibobol sehingga data-data lain yang tidak ada pada *role*nya tetap aman. Menurut Dongdong, dkk (2017) konsep RBAC ini dapat diterapkan sesuai dengan kebutuhan organisasi [1]. Contoh dalam sistem informasi manajemen sekolah ada pihak karyawan dan siswa. Maka *role* dalam *database* bisa dibuat dua yaitu karyawan dan siswa. Tetapi, jumlah *user* dalam *role* karyawan dan siswa tersebut sama dengan jumlah karyawan dan siswa yang ada dalam sekolah tersebut. Hubungan relasi dalam metode ini adalah *user* dengan *role*, kemudian *role* dengan hak akses ke dalam *table* dalam *database*. Berdasarkan dokumentasi PostgreSQL Group (2018) *role database* memiliki beberapa macam atribut, yaitu : *login privilage, superuser status, database creation, role creation, initialing replication,* dan *password* [7].

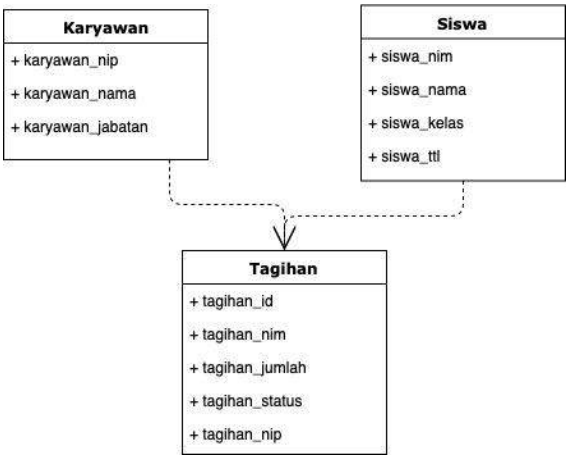
3. METODE PENELITIAN

Penelitian ini menggunakan metode eskperimen, pengertian eksperimen menurut Arboleda (1981) adalah metode penelitian dengan cara melakukan manipulasi satu atau lebih variabel dengan sengaja untuk dicari pengaruhnya pada satu atau variabel lain yang diteliti. Variabel bebas adalah variabel yang dimanipulasi, sedangkan variabel terikat adalah variabel yang dilihat pengaruhnya [8]. Dalam

pengumpulan data dilakukan tiga tahap yaitu: (1) Studi pustaka, dalam tahap ini peneliti melakukan kajian terhadap buku atau jurnal yang terkait. (2) Pembuatan data, yaitu dengan cara membuat data yang bersifat *dummy* tetapi berdasarkan keadaan lapangan yang sebenarnya. (3) Eksperimen, yaitu melakukan uji coba terhadap data yang dibuat dengan cara membuat *database* lalu dilakukan penerapan metode *Role-Based Access Control*.

4. HASIL DAN PEMBAHASAN

Penerapan metode RBAC dalam *database* disesuaikan dengan kebutuhan dari organisasi yang membutuhkannya. Dalam penelitian ini menggunakan contoh dalam administrasi sekolah, lebih spesifiknya lagi penelitian ini fokus pada administrasi keuangan sekolah. Secara garis besar, pihak yang terlibat keuangan dalam suatu sekolah adalah bagian keuangan dan siswa yang melakukan pembayaran. Untuk itu maka dibuatlah *database* dengan tiga tabel yaitu tabel siswa, tabel karyawan keuangan dan tabel tagihan.



Gambar 1. *Class Diagram*

Diasumsikan bahwa jumlah karyawan dan siswa sama dengan jumlah *user* dalam *database* dengan *role* siswa dan karyawan. Langkah pertama adalah dengan membuat *database* `admin_kampus` menggunakan perintah `create database admin_sekolah;` . Karena menggunakan *database* PostgreSQL maka *default user* yang digunakan adalah `achmadyusronarif`. *Default user* bukan postgres karena dalam komputer miliki peneliti *default user* sudah diganti menggunakan nama peneliti. Setelah dibuat *database* `admin_sekolah` maka selanjutnya dibuatlah tabel sesuai dengan gambar 1. Setelah tabel dibuat maka struktur *database* akan menjadi seperti berikut :

```
                List of relations
Schema |          Name          |   Type   |      Owner
--------+------------------------+----------+-----------------
public | karyawan               | table    | achmadyusronarif
public | siswa                  | table    | achmadyusronarif
public | tagihan                | table    | achmadyusronarif
public | tagihan_tagihan_id_seq | sequence | achmadyusronarif
(4 rows)
```

Gambar 2. Tabel dalam *database* `admin_sekolah`

Ada tabel dengan tipe data *sequence* dikarenakan dalam tabel `tagihan_id` menggunaka auto increament. Masukkan data ke dalam tabel sesuai dengan data pada tabel 1, tabel 2, dan tabel 3.

Tabel 1. Karyawan

NIP	Nama	Jabatan
256789	Sri Tanjung	Bendahara 1
256790	Achmad Maulana	Bendahara 2

Tabel 2. Siswa

NIM	Nama	Kelas	TTL
17511043	Luthfan Ali	X	20 Januari 1995
17511044	Achmad Arif	X	1 Juli 1995
17511045	Surya Negara	XI	20 Juli 1994
17511046	Janaka Budi	XI	30 April 1994
17511047	Setya Mandela	XI	30 Mei 1994

Tabel 3. Tagihan

ID	NIM	Jumlah	Status	NIP
1	17511043	Rp 300.000	BELUM	
2	17511044	Rp 300.000	LUNAS	256789
3	17511045	Rp 250.000	BELUM	
4	17511046	Rp 250.000	LUNAS	256789
5	17511047	Rp 250.000	BELUM	

Dalam tabel 3 terdapat baris yang kolom NIP kosong, hal ini dikarenakan status pembayaran yang belum lunas. Jadi, dalam kolom NIP akan diisi jika telah dilakukan pembayaran dan diisi dengan NIP karyawan yang menggantu status tagihan dari belum

menjadi lunas. Dalam *database* ini dibuat 2 *role* yaitu *role* karyawan dengan nama role `admin_sekolah_role_karyawan` dan *role* siswa dengan nama `admin_sekolah_role_siswa`. Dengan menggunakan perintah *query* yaitu `create role nama_role` ***atribut***; . Nama_role diisi dengan dua role yang akan dibuat, atribut diisi dengan atribut *role* yang digunakan. Atribut superuser untuk *role* karyawan dan atribut login untuk siswa. Masukkan data pada tabel 1, tabel 2 dan tabel 3 ke dalam *database* dengan menggunakan perintah `insert into karyawan(karyawan_nip,karyawan_nama,karyawan_jabatan) values(256790,'Achmad Maulana','Bendahara 2');` . Perintah tersebut diulang hingga semua data telah masuk ke dalam *database*. Dibuat *user role* dari setiap pengguna sistem informasi yaitu karyawan dan siswa dengan jumlah pengguna 7 orang dalam 2 *role* menggunakan *query* `create role` **`nama_user`** `login;` , gambar 3 adalah daftar *user* setelah dibuat.

List of roles

Role name	Attributes	Member of
achmadyusronarif	Superuser, Create role, Create DB	{}
admin_sekolah_17511043		{admin_sekolah_siswa}
admin_sekolah_17511044		{admin_sekolah_siswa}
admin_sekolah_17511045		{admin_sekolah_siswa}
admin_sekolah_17511046		{admin_sekolah_siswa}
admin_sekolah_17511047		{admin_sekolah_siswa}
admin_sekolah_karyawan	Superuser	{}
admin_sekolah_karyawan_achmad		{admin_sekolah_karyawan}
admin_sekolah_karyawan_sri		{admin_sekolah_karyawan}
admin_sekolah_siswa	Superuser	{}

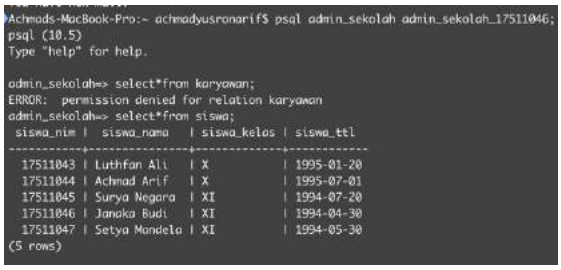
Gambar 3. Daftar *user database*

Untuk daftar *privilege* dan nilai *value* perhatikan tabel 4 berikut :

Tabel 4. Simbol *privilege* PostgreSQL

Simbol	Arti
r	SELECT ("read")
w	UPDATE ("write")
a	INSERT ("append")
d	DELETE

Role siswa tidak diberi hak untuk mengakses data tabel karyawan, tetapi karyawan diberi hak untuk mengakses tabel siswa dan tagihan. Siswa hanya diberi hak untuk membaca (*select / read*) pada tabel siswa dan tagihan saja. Untuk memberikan akses tabel kepada suatu *role* menggunakan perintah `grant` **`select`** `on siswa to admin_sekolah_siswa`; *select* bisa diganti dengan *update*, *insert* atau *delete* atau juga bisa dengan kombinasi semuanya menggunakan tanda koma (,) sebagai pemisah *privilege*. Gambar 4 adalah ujicoba dengan mengakses menggunakan *user* `admin_sekolah_17511046`.



```
Achmads-MacBook-Pro:~ achmadyusronarif$ psql admin_sekolah admin_sekolah_17511046;
psql (10.5)
Type "help" for help.

admin_sekolah=> select*from karyawan;
ERROR:  permission denied for relation karyawan
admin_sekolah=> select*from siswa;
 siswa_nim |  siswa_nama   | siswa_kelas | siswa_ttl
-----------+---------------+-------------+------------
  17511043 | Luthfan Ali   | X           | 1995-01-20
  17511044 | Achmad Arif   | X           | 1995-07-01
  17511045 | Surya Negara  | XI          | 1994-07-20
  17511046 | Janaka Budi   | XI          | 1994-04-30
  17511047 | Setya Mandela | XI          | 1994-05-30
(5 rows)
```

Gambar 4. Daftar *user database*

Dari gambar 2 terlihat bahwa *user* `admin_sekolah_17511046` tidak bisa mengakses tabel `karyawan` karena tidak diberi akses, tetapi tetap bisa mengakses tabel `siswa`. Untuk *role* karyawan diberi akses untuk mengakses semua tabel yang ada, menggunakan perintah grant all on all tables in schema public to `admin_sekolah_karyawan;` , gunakan perintah `\dp` untuk mengetahui *access privileges* seperti pada gambar 5 berikut :

Access privileges

Name	Type	Access privileges
karyawan	table	achmadyusronarif=arwdDxt/achmadyusronarif admin_sekolah_karyawan=arwdDxt/achmadyusronarif
siswa	table	achmadyusronarif=arwdDxt/achmadyusronarif admin_sekolah_siswa=r/achmadyusronarif admin_sekolah_karyawan=arwdDxt/achmadyusronarif
tagihan	table	achmadyusronarif=arwdDxt/achmadyusronarif admin_sekolah_karyawan=arwdDxt/achmadyusronarif admin_sekolah_siswa=r/achmadyusronarif
tagihan_tagihan_id_seq	sequence	

Gambar 5. Daftar *access privileges*

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan analisis yang telah dilakukan maka dapat disimpulkan sebagai berikut:

1. Jumlah dan *access privileges* dari sebuah *role* dapat disesuaikan dengan kebutuhan suatu organisasi.
2. Dengan menggunakan metode *role-based access control* dapat mengurangi kemungkinan rusaknya integritas data jika *database* dibobol oleh *hacker*, hal ini dikarenakan jika *hacker* hanya bisa membobol satu *role* atau *user* maka hanya tabel yang mendapatkan akses ke *role* tersebut yang bisa diakses oleh *hacker*.

3. Menjaga integritas data, yaitu pada penelitian ini *role* siswa hanya diberi akses untuk membaca data tabel saja tidak untuk memanipulasi data, sehingga mengurangi kemungkinan untuk siswa memanipulasi data baik dengan atau tidak disengaja.

5.2 Saran

1. Perlu penelitian lebih lanjut untuk mengetes performa *database* setelah diberi metode keamanan *role-based access control*.
2. Dalam penelitian lebih lanjut bisa diimplementasikan ke *database* lain seperti Oracle, dan membandingkannya dengan PostgreSQL untuk mengetahui mana yang lebih baik dari kedua *database* tersebut.
3. Sistem keamanan ini bisa dikembangkan dengan kombinasi bahasa pemrograman seperti PHP, Java, dll supaya bisa digunakan dalam sebuah sistem informasi yang bisa digunakan secara langsung oleh *end user*.

DAFTAR PUSTAKA

[1] Hartono, N. (2015). Keuntungan Penggunaan External Function Pada Database Postgresql, CSRID Journal, Vol.7

[2] Ardiansyah, SS., Raharjo, S., Triyono J. (2016). Analisis Keamanan Serangan Sql Injection Berdasarkan Metode Koneksi Database. Jurnal SCRIPT Vol. 4

[3] Dongdong, Liu., Shilliang, XU., Yan ,Zhang., Fuxiao, TAN., Lei, NIU., Jia, ZHAO., (2017). Role - based Access Control in Educational Administration System, MATEC Web of Conferences 139, 00120.

[4] Susilo, Gatot. (2016). Keamanan Basis Data Pada Sistem Informasi Di Era Global, Jurnal TRANSFORMASI, Vol. 12, No. 2.

[5] Yuliardi., 2000, Panduan Administrasi Database PostgreSQL, http://www.kki.go.id/assets/data/menu/PostgreSQL.pdf

[6] Connoly, T., Begg, C., 2005, DATABASE SYSTEMS A Practical Approach to Design, Implementation, and Management, Fourth Edition, Addison Wiley, United States of America

[7] Group, T.P.G.D., 2018, PostgreSQL 10.5 Documentation, https://www.postgresql.org/files/documentation/pdf/10/postgresql-10-A4.pdf

[8] Setyanto, A.E. (2015) Memperkenalkan Kembali Metode Eksperimen dalam Kajian Komunikasi, Jurnal Ilmu Komunikasi, Volume 3, Nomor 1