BAB II
LANDASAN TEORI

2.1. Database

Database merupakan sekumpulan data yang berhubungan secara logika dan memiliki beberapa arti yang saling berpautan. Beberapa komponen yang dimiliki oleh sistem basis data yaitu *hardware*, *software*, *database*, DBMS dan *user*. Istilah lain dalam basis data yaitu permodelan data yang merupakan kumpulan perangkat konseptual untuk menggambarkan data, hubungan antar data dan batasan data. Menurut (F.Korth, 2001)ada tiga model data basis data yaitu:

1. Model data *Hirarkis*

   Model data hirarkis menjelaskan hubungan logika antara data dalam bentuk hubungan bertingkat.

2. Model data Jaringan

   Hampir sama seperti model data *hirarkis*, tetapi dalam model data jaringan suatu node dibawahnya biasa mempunyai hubungan dengan lebih banyak dari node diatasnya.

3. Model *Relational*

   Basis data akan dinyatakan dalam bentuk tabel-tabel dua dimensi. Setiap tabel terdiri atas lajur mendatar yang disebut dengan baris *(row/record)* dan lajur vertikal yang disebut kolom *(column/field)*. Baris-baris ini akan tersusun membentuk satu tabel, yang biasanya tersimpan dalam satu file. Tabel-tabel ini secara keseluruhan merupakan penyajian dari atribut data yang saling berhubungan.

   Keuntungan model basis data *relasional* adalah

   a. Data dapat diakses secara cepat.
   b. Struktur basis data mudah diubah.
   c. Data disajikan secara logis sehingga pengguna tidak perlu mengetahui bagaimana data disimpan.
   d. Pengguna mudah membuat *query* yang kompleks untuk mengambil data. Pengguna mudah menerapkan integritas data.

e. Pengguna mudah membuat dan memodifikasi program aplikasi.
f. Bahasa standar (SQL) sudah dibuat.

Kekurangan model basis data *relasional* adalah:

a. Kelompok informasi atau tabel yang berbeda harus dihubungkan untuk mengambil data.
b. Pengguna harus memahami hubungan antartabel.
c. Pengguna harus belajar SQL.

2.1.1. Keamanan *Database*

Keamanan *database* merupakan proteksi terhadap pengrusakan data dan pemakaian data oleh pemakai yang tidak punya kewenangan ataupun yang mempunyai kewenangan. Keamanan *database* dapat dilihat sebagai berikut. (Suyanto, 2012)

1. Manusia

   Wewenang pemakai harus dilakukan dengan berhati-hati untuk mengurangi kemungkinan adanya manipulasi oleh pemakai yang berwenang.

2. Sistem Operasi (SO)

   Kelemahan pada SO ini memungkinkan pengaksesan data oleh pihak tak berwenang, karena hampir seluruh jaringan sistem *database* menggunakan akses jarak jauh.

3. Sistem *Database*

Pengaturan hak pemakai yang baik. Sebuah *database* harus memilki Batasan - batasan terhadap user. Misalnya suatu user hanya dapat mengakses tabel A dan tidak bisa mengakses tabel B. (Rijayanti, 2013)

Sistem database yang aman dapat memastikan kerahasian data yang tedapat didalamnya. Berikut ini adalah aspek dari keamanan database

1. Membatasi akses data ke data dan *service*.
2. Melakukan autentifikasi pada *user*.
3. *Memonitoring* aktifitas-aktifitas yang mencurigakan yang dilakukan oleh user.

Sebuah sistem harus mempunyai tiga *property* (sifat), yaitu:

- *Integritas*, system akan mempunyai integritas bila ia berjalan menurut spesifikasinya. Perancang sistem berusaha untuk mengembangkan sistem yang mempunyai *integritas* fungsional, yaitu kemampuan untuk melanjutkan operasi, apabila salah satu atau lebih dari komponennya tidak berjalan.
- *Audibilitas*, ia akan bersifat *audible* jika ia memiliki *visibilitas* dan *accountability* (daya perhitungan). Bila sistem memiliki *audibilitas* maka mudah bagi seseorang untuk memeriksa, memverifikasi atau menunjukkan penampilannya.
- Daya kontrol, daya kontrol memungkinan manajer untuk menangani pengerahan atau penghambatan pengaruh terhadap sistem. Teknik yang efektif untuk mendapatkan daya kontrol sistem ini adalah dengan membagi sistem menjadi subsistem yang menangani transaksi secara terpisah.

2.1.2. Structure Query Language (SQL)

Pernyataan SQL dapat dikelompokkan menjadi 4 kelompok DDL, DML, DCL dan pengendali transaksi. Berikut penjulasan masing kelompok tersebut. (Warman & Ramdaniansyah, 2018)

1. ***Data Definition Language*** **(DDL)**

   DDL merupakan kelompok perintah yang berfungsi untuk mendefinisikan atribut-atribut *database*, table, atribut (kolom), batasan-batasan terhadap suatu atribut serta hubungan antar table. Yang termasuk kelompok DDL ini adalah:

   - *CREATE* untuk menciptakan table ataupun *indeks*.
   - *ALTER* untuk mengubah struktur table.
   - *DROP* untuk menghapus table ataupun *indeks.*

2. ***Data Manipulation Language*** **(DML)**

   Adalah kelompok perintah yang berfungsi untuk memanipulasi data, misalnya untuk pengambilan, penyisipan pengubahan dan penghapusan data. Yang termasuk DML adalah:

   a. *SELECT* memilih data.
   b. *INSERT* menambah data.
   c. *DELETE* menghapus data.
   d. *UPDATE* mengubah data.

3. ***Data Control Language*** **(DCL)**

   Berisi perintah-perintah untuk mngendalikan pengaksesan data. Yang termasuk DCL adalah:

   a. *GRANT* memberikan kendali pada pengaksesan data.
   b. *REVOKE* mencabut kemampuan pengaksesan data.
   c. *LOCK TABLE* mengunci table.

4. Pengendali Transaksi (DTL)

Adalah perintah-perintah yang berfungsi untuk mengendalikan pengeksekusian transaksi. Yang termasuk kelompok ini adalah:

a. *COMMIT* menyetujui rangkaian perintah yang berhubungan erat yang telah berhasil dilakukan.

b. *ROLLBACK* membatalkan transaksi yang dilakukan karena adanya kesalahan atau kegagalan pada salah satu rangkaian perintah.

2.2. Database Management System (DBMS)

Database management system (DBMS) merupakan suatu sistem *software* yang memungkinkan *user* untuk mengindentifikasikan, membuat dan memelihara *database* maupun menyediakan akses yang terkontrol terhadap data. Sebelum adanya DBMS data pada umumnya disimpan dalam bentuk flat file, yaitu file teks yang ada pada sistem operasi. Sampai sekarangpun masih ada aplikasi yang menyimpan data dalam bentuk flat file secara langsung. Menyimpan data dalam bentuk flat file mempunyai kelebihan dan kekurangan. Penyimpanan dalam bentuk ini akan mempunyai manfaat yang optimal jika ukuran filenya relatif kecil, seperti file *password*. File *password* pada umumnya hanya digunakan untuk menyimpan nama yang jumlahnya tidak lebih dari 1000 orang. Selain dalam bentuk flat file, penyimpanan data juga dapat dilakukan dengan menggunakan program bantu seperti *spreadsheet*. Penggunaan spreadsheet ini memperbaiki beberapa kelemahan dari flat file, seperti bertambahnya kecepatan dalam pengolahan data. Namun demikian metode ini masih memiliki banyak kelemahan, diantaranya adalah masalah manajemen dan keamanan data yang masih kurang. Sehingga muncul lah sebuah sistem penyimpanan data yaitu DBMS.

Penyimpanan data dalam bentuk DBMS mempunyai banyak manfaat dan kelebihan dibandingkan dengan penyimpanan dalam bentuk flat file atau *spreadsheet*, diantaranya: (Yuliansyah, 2014)

1. *Performance* yang didapat dengan penyimpanan dalam bentuk DBMS cukup besar, sangat jauh berbeda dengan performance data yang disimpan dalam bentuk flat file. Disamping memiliki unjuk kerja yang lebih baik, juga akan didapatkan efisiensi penggunaan media penyimpanan dan memori.
2. *Integritas* data lebih terjamin dengan penggunaan DBMS. Masalah redudansi sering terjadi dalam DBMS. Redudansi adalah kejadian berulangnya data atau kumpulan data yang sama dalam sebuah database yang mengakibatkan pemborosan media penyimpanan.
3. *Independensi*. Perubahan struktur database dimungkinkan terjadi tanpa harus mengubah aplikasi yang mengaksesnya sehingga pembuatan antarmuka ke dalam data akan lebih mudah dengan penggunaan DBMS.
4. *Sentralisasi*. Data yang terpusat akan mempermudah pengelolaan database. Kemudahan di dalam melakukan bagi pakai dengan DBMS dan juga kekonsistenan data yang diakses secara bersama-sama akan lebiih terjamin dari pada data disimpan dalam bentuk file atau worksheet yang tersebar.
5. *Sekuritas*. DBMS memiliki sistem keamanan yang lebih fleksibel daripada pengamanan pada file sistem operasi. Keamanan dalam DBMS akan memberikan keluwesan dalam pemberian hak akses kepada pengguna.

Banyak program basis data yang sudah sering di gunakan, misalnya: FoxPro, Access, Paradox dan lain sebagainya. Itu merupakan contoh dari DBMS yang digunakan pada PC dalam skala yang relatif kecil. Dalam skala yang lebih besar, dikenal beberapa DBMS yang sering digunkan, antara lain Oracle, MySQL, PostgreSQL, SQLServer dan lain-lain. Berikut ini adalah tabel perbandingan fitur yang dimiliki oleh beberapa DBMS yang dapat dilihat pada tabel 2.1.

Tabel 2.1 Tabel perbandingan fitur DBMS

DBMS / Description	Postgresql	Ms Access	FoxPro	Paradox	Mysql	SQL Server	Oracle
Tipe	Client	File based server	File based server	File based server	Client	Client	Client Server
ACID Complaint	Yes	No	No	No	No	Yes	Yes
Support Transsction	Yes	No	No	No	Yes	Yes	Yes
Support Store procedure	Yes	Using Query	No	No	Yes	Yes	Yes
Support Trigger	Yes	No	No	No	Yes	Yes	Yes
Multi user	Unlimited	Limited	Limited to 50	Limited to 255	Unlimited	Unlimited	Unlimited
Host Platform	Linux, Unix	Windows	Windows	Linux, Unix	Linux, Unix	Windows	Linux, Unix

	dan Windows			dan Windows	dan Windows		dan Windows
Support Procedure	Yes	No	No	No	Yes	Yes	Yes
Transaction With	Yes	No	No	No	Yes, with	Yes	Yes
Max Tabel size	64 TB	?	?	?	4 GB	1 Exabyte	No obvious limit
Max Singe Field	1 GB	?	?	?	?	2 GB	2GB
Max number of row	Unlimited	?	?	?	No obvious limit	Limited by storage	No obvious limit
Max number ofcolums	1600	?	?	?	?	1024	1000

Dari beberapa perbandingan DBMS tersebut terlihat bahwa DBMS Oracle, Mysql dan Postgresql lebih banyak memilki fitur. Selain itu ketiga DBMS ini juga dapat berjalan diberbagai sistem operasi.

2.3. DBMS MySQL

DBMS MySQL adalah sebuah implementasi dari *sistem manajemen basisdata relasional (RDBMS)* yang didistribusikan secara gratis dibawah *lisensi* GPL (*General Public License*). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basisdata yang telah ada sebelumnya; SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian basisdata, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis. Kehandalan suatu sistem basisdata (DBMS) dapat diketahui dari cara kerja pengoptimasi-nya dalam melakukan proses perintah-perintah SQL yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. Sebagai peladen basis data, MySQL mendukung operasi basisdata transaksional maupun operasi basisdata non-transaksional. Pada modus operasi non-transaksional, MySQL dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak peladen basisdata kompetitor lainnya. Namun demikian pada modus non-transaksional tidak ada jaminan atas reliabilitas terhadap data yang tersimpan, karenanya modus non-transaksional hanya cocok untuk jenis aplikasi yang tidak membutuhkan reliabilitas data seperti aplikasi blogging berbasis web (*wordpress*), CMS, dan sejenisnya. Untuk kebutuhan sistem yang ditujukan untuk bisnis sangat disarankan untuk menggunakan modus basisdata transaksional, hanya saja sebagai konsekuensinya unjuk kerja MySQL pada modus transaksional tidak secepat unjuk kerja pada modus non-transaksional. (Paul , 2013)

MySQL memiliki beberapa keistimewaan, antara lain:

1. *Portabilitas*. MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lagi.
2. Perangkat lunak sumber terbuka. MySQL didistribusikan sebagai perangkat lunak sumber terbuka, dibawah lisensi GPL sehingga dapat digunakan secara gratis.

3. Multi-user. MySQL dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.

4.*'Performance tuning'*, MySQL memiliki kecepatan yang menakjubkan dalam menangani query sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.

5. Ragam tipe data. MySQL memiliki ragam tipe data yang sangat kaya, seperti *signed / unsigned integer, float, double, char, text, date, timestamp*, dan lain-lain.

6. Perintah dan Fungsi. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam perintah (*query*).

7. Keamanan. MySQL memiliki beberapa lapisan keamanan seperti level subnetmask, nama *host*, dan izin akses *user* dengan sistem perizinan yang mendetail serta sandi terenkripsi.

8. Skalabilitas dan Pembatasan. MySQL mampu menangani basis data dalam skala besar, dengan jumlah rekaman (*records*) lebih dari 50 juta dan 60 ribu tabel serta milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.

9. Konektivitas. MySQL dapat melakukan koneksi dengan klien menggunakan protokol TCP/IP, *Unix soket* (UNIX), atau *Named Pipes* (NT).

10. Lokalisasi. MySQL dapat mendeteksi pesan kesalahan pada klien dengan menggunakan lebih dari dua puluh bahasa. Meski pun demikian, bahasa Indonesia belum termasuk di dalamnya.

11. Antar Muka. MySQL memiliki antar muka (*interface*) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Application Programming Interface)*.

12. Klien dan Peralatan. MySQL dilengkapi dengan berbagai peralatan (*tool*)yang dapat digunakan untuk administrasi basis data, dan pada setiap peralatan yang ada disertakan petunjuk online.

13. Struktur tabel. MySQL memiliki struktur tabel yang lebih *fleksibel* dalam menangani ALTER TABLE, dibandingkan basis data lainnya semacam PostgreSQL ataupun Oracle.

2.3.1. Versi MySQL

MySQL versi 1.0 dirilis Mei 1996 secara terbatas kepada empat orang. Baru di bulan Oktober versi 3.11.0 dilepas ke publik. Namun mula-mula kode ini tidak diberikan di bawah lisensi General Public License. Barulah pada Juni 2000 MySQL AB mengumumkan bahwa sejak versi 3.23.19, MySQL adalah software bebas berlisensi GPL. Versi publik pertama, yang hanya berjalan di Linux dan Solaris serta sebagian besar masih belum terdokumentasi itu, dengan berangsur-angsur diperbaiki dan ditambah fitur demi fiturnya, tapi tetap dengan fokus utama pengembangan pada kelangsingan dan kecepatan. Artinya, fitur yang menyebabkan MySQL menjadi lambat tidaklah ditambahkan, atau ditunda dulu, atau ditambahkan tapi menjadi fitur yang opsional. Barulah di versi-versi akhir 3.22, yaitu sepanjang tahun 1998–1999 MySQL menjadi semakin popular dan dilirik orang. Stabilitasnya sudah baik. Kecepatannya meningkat dan sudahtersedia di berbagai platform, termasuk Windows. Di seri 3.23 MySQL menambahkan tiga jenis tabel baru yaitu MyISAM, BerkeleyDB dan InnoDB ketiga tabel ini

menambahkan fitur-fitur baru dalam DBMS MySQL. Selanjutnya di seri yang baru berjalan hingga 4.0 tahap alfa ini, pengembang MySQL berjanji akan menjadikan MySQL satu derajat lebih tinggi lagi. Fitur-fitur yang sejak dulu diminta akan dikabulkan, seperti subselek (di 4.1), union (4.0), foreign key constraint (4.0 atau 4.1—meski InnoDB sudah menyediakan ini di 3.23.x), stored procedure (4.1), view (4.2), cursor (4.1 atau 4.2), trigger (4.1). MySQL AB tetap berdedikasi mengembangkan dan memperbaiki MySQL, serta mempertahankan MySQL sebagai database open source terpopuler. Hingga saat ini MySQL terus mengalami perkembangan hingga versi 5.x.x dan versi MySQL yang tebaru adalah 5.5.10, beberapa fitur yang tersedia pada versi 5.x.x keatas ini seperti *log file* dan *Trigger.* (Silalahi & Wahyudi, 2018)

2.3.2 Log File MySQL

Log file adalah file yang berisi catatan aktivitas pengguna. Kegunaan *log file* sangat banyak, misalnya untuk program *webserver*, *file log* dapat menunjukkan *hit* yang diterima oleh suatu situs, menunjukkan halaman mana saja dalam situs yang dicoba diakses, *browser* apa saja yang digunakan oleh pengunjung situs, dan lain- lain. (Rokhmah, Prayudi, & Sugiantoro, 2017)

Log file pada DBMS MySQL terdiri atas beberapa jenis, sesuai dengan fungsinya masing-masing yaitu:

1. *General Log*

   Merupakan catatan umum apa yg dilakukan oleh MySQL. *Server* MySQL menulis informasi ketika *user* (klien) terkoneksi ataupun tidak terkoneksi dan setiap informasi tersebut diterima dari klien. *General log* ini sangat berguna ketika kita mencurigai adanya kesalahan di klien dan ingin mengetahui apa yang klien kirim ke *server*. Untuk mengaktifkan fitur *log general* ini dilakukan dengan melakukan konfigurasi pada file **my.ini.**

Konfigurasi tersebut dilakukan dengan menambahkan *sintaks* seperti berikut dibawah tulisan [mysqld].

```
general_log = 1

general_log_file     =
"C:/xampp/mysql/data/log/mysql
general.log"
```

2. *Error Log*

Berisi semua informasi pesan kesalahan yang terjadi dalam *server* MySQL. Informasi pada *error log* akan bertambah jika *server* MySQL berhenti tiba-tiba dan harus direstart. Jika MySQL menemukan suatu tabel yang membutuhkan pengecekan atau perbaikan, maka informasi tersebut akan dicatat pada *error log*. Selain itu *error log* ini juga berfungsi untuk mencatat proses *shutdown* dan *startup database*. Untuk mengaktifkan error log ini dilakukan dengan melakukan konfigurasi pada file **my.ini**. Konfigurasi tersebut dilakukan dengan menambahkan sintaks seperti berikut dibawah tulisan [mysqld].

```
log_error = "C:/xampp/mysql/data/log/mysql- error.log"
```

3. *Binary Log*

Log ini berfungsi untuk mencatat semua perintah yang menyebabkan perubahan data dalam basis data seperti perintah *INSERT, DELETE, UPDATE* dan sebagainya. Isi *Binary Log* dapat dilihat menggunakan perintah *mysqlbinlog*. *Binary log* juga berguna untuk proses *replikasi* basis data. Dalam proses ini *binary log* digunakan oleh *server master* untuk menyimpan tiap perintah yang akan dikirim ke *server slave*. *Server master* kemudian mengirim *event log* ke dalam *binary log* yang ada di *server slave*, sehingga dengan demikian *server slave* juga melakukan apa yang dilakukan oleh *server master*.

4. *Slow Query Log*

Slow Query Log pada *MySQL* berisi semua *query SQL* yang dijalankan (waktu eksekusi) melebihi selang waktu tertentu. Selang waktu yang dimaksud ditentukan pada variabel sistem *server MySQL (mysqld)*. Saat menjalankan *instance MySQL (mysqld.exe)*, terdapat pilihan *long_query_time*, yang digunakan untuk pengecekan apakah akan menulis ke log atau tidak. Isi dari pilihan tersebut merupakan nilai dalam detik, jika suatu query yang dikirim dari *client* dijalankan selama selang waktu lebih dari yang ditentukan dari pilihan *long_query_time*, maka *query* yang bersangkutan akan ditulis ke *file log*. Untuk mengaktifkan *slow query log* ini dilakukan dengan melakukan konfigurasi pada file **my.ini**. Konfigurasi tersebut dilakukan dengan menambahkan sintaks seperti berikut dibawah tulisan [mysqld].

```
slow_query_log = 1

slow_query_log_file =

"C:/xampp/mysql/data/log/mysql-slow.log"
```

Untuk membaca *File Log* dapat menggunakan aplikasi teks editor seperti *Wordpad*, *notepad++*, dan lainya. Pemantauan *file log* dapat menjadi sebuah pekerjaan yang sangat melelahkan karena jika jumlah data yang telah di catat banyak maka proses pencarian data sangat sulit sehingga diperlukan suatu cara untuk mengimpor file *log* tersebut kedalam sebuah tabel. Dalam DBMS *Mysql* sendiri telah menyediakan tempat penyimpanan secara default yaitu pada database *mysql* dengan nama tabel *general_log*. Tabel *general_log* ini hanya dapat menampung isis dari *general_log* dan bukan untuk *log file* yang lain. Berikut ini cara yang digunakan untuk mengimpor atau mengubah tempat penyimpanan *log general* dari file text ke database.

Terlebih dahulu konfigurasi di file my.ini dirubah menjadi konfigurasi sebagai berikut.

```
slow_query_log    = 1

log_output        = TABLE
```

Selanjutnya melalui command promt masuk ke MySQL sebagai user adminstrator (root) lalu ketikkan peritah berikut.

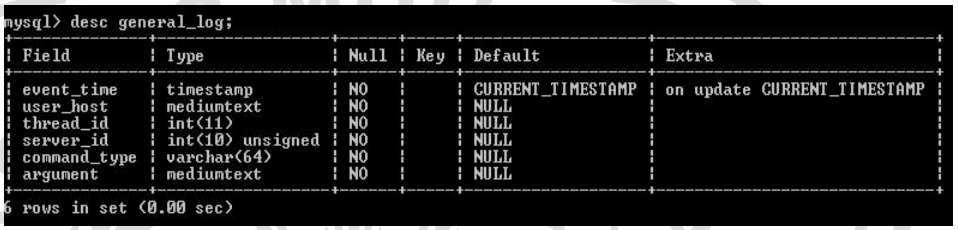
```
mysql> ALTER TABLE
mysql.general_log ENGINE =
MyISAM; Query OK, 2 rows
affected (0.19 sec)

Records: 2 Duplicates: 0 Warnings: 0
```

```
mysql> SET GLOBAL
general_log = 'ON';
Query OK, 0 rows
affected (0.03 sec)

mysql> show create table mysql.general_log;
```

Dari hasil impor data tersebut berikut ini adalah data atau informasi yan dapat diperoleh dari general_log yang dapat dilihat pada gambar 2.1



```
mysql> desc general_log;
+--------------+------------------+------+-----+-------------------+-----------------------------+
| Field        | Type             | Null | Key | Default           | Extra                       |
+--------------+------------------+------+-----+-------------------+-----------------------------+
| event_time   | timestamp        | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| user_host    | mediumtext       | NO   |     | NULL              |                             |
| thread_id    | int(11)          | NO   |     | NULL              |                             |
| server_id    | int(10) unsigned | NO   |     | NULL              |                             |
| command_type | varchar(64)      | NO   |     | NULL              |                             |
| argument     | mediumtext       | NO   |     | NULL              |                             |
+--------------+------------------+------+-----+-------------------+-----------------------------+
6 rows in set (0.00 sec)
```

Gambar 2.1 general_log

2.3.2. *Trigger* DBMS MySQL

Pada DBMS MySQL, trigger digunakan untuk memanggil satu atau beberapa perintah SQL secara otomatis sebelum atau sesudah terjadi proses *INSERT, UPDATE* atau *DELETE* dari suatu tabel. Sebagai contoh misalnya kita ingin menyimpan id pelanggan secara otomatis ke tabel '*log*' sebelum menghapus data di tabel pelanggan.

Di DBMS MySQL, Triggers mulai dikenal di versi MySQL 5.0, dan di versi saat ini (5.1.4) fungsionalitasnya sudah bertambah. Pada versi selanjutnya pihak pengembang MySQL berjanji akan lebih menguatkan (menambah) fitur trigger ini.

Trigger pada DBMS MySQL sering digunakan, antara lain untuk:

a. Melakukan *update* data otomatis jika terjadi perubahan. Contohnya adalah dalam sistem penjualan, jika dientri barang baru maka stock akan bertambah secara otomatis.
b. *Trigger* dapat digunakan untuk mengimplementasikan suatu sistem *log*. Setiap terjadi perubahan, secara otomatis akan menyimpan ke tabel *log*.
c. *Trigger* dapat digunakan untuk melakukan validasi dan verifikasi data sebelum data tersebut disimpan.

Sintak umum trigger pada DBMS MySQL.

```
CREATE TRIGGER name
[BEFORE|AFTER] [INSERT|UPDATE|DELETE]
ON tablename
FOR EACH ROW statement
```

Keterangan dari bentuk umum perintah membuat trigger:

- **name**, Nama trigger mengikuti peraturan penamaan variabel / identifier dalam MySQL
- **[BEFORE | AFTER]** digunakan untuk menentukan kapan proses secara otomatis akan dieksekusi, sebelum atau sesudah proses.
- **[INSERT | UPDATE | DELETE]** digunakan untuk menentukan event (proses) yang dijadikan trigger (pemicu) untuk menjalankan perintah- perintah di dalam triggers.

- **tablename**, merupakan nama tabel dimana trigger berada.
- **statement**, merupakan sekumpulan perintah atau query yang akan secara otomatis dijalankan jika event / proses yang didefinisikan sebelumnya aktif.

Statement atau perintah dalam trigger dapat berupa satu perintah saja, dan dapat juga beberapa perintah sekaligus. Jika terdapat beberapa perintah dalam *trigger*, maka gunakan perintah **BEGIN** dan **END** untuk mengawali dan mengakhiri perintah. Di dalam statement trigger, kita dapat mengakses record tabel sebelum atau sesudah proses dengan menggunakan **NEW** dan **OLD. NEW** digunakan untuk mengambil record yang akan diproses (insert atau update), sedangkan **OLD** digunakan untuk mengakses record yang sudah diproses (update atau delete)

2.3.3. *Trigger* DBMS MySQL

Pada DBMS MySQL, trigger digunakan untuk memanggil satu atau beberapa perintah SQL secara otomatis sebelum atau sesudah terjadi proses *INSERT, UPDATE* atau *DELETE* dari suatu tabel. Sebagai contoh misalnya kita ingin menyimpan id pelanggan secara otomatis ke tabel '*log*' sebelum menghapus data di tabel pelanggan.

Di DBMS MySQL, Triggers mulai dikenal di versi MySQL 5.0, dan di versi saat ini (5.1.4) fungsionalitasnya sudah bertambah. Pada versi selanjutnya pihak pengembang MySQL berjanji akan lebih menguatkan (menambah) fitur trigger ini.

Trigger pada DBMS MySQL sering digunakan, antara lain untuk:

a. Melakukan *update* data otomatis jika terjadi perubahan. Contohnya adalah dalam sistem penjualan, jika dientri barang baru maka stock akan bertambah secara otomatis.
b. *Trigger* dapat digunakan untuk melakukan validasi dan verifikasi data sebelum data tersebut disimpan.

c. *Trigger* dapat digunakan untuk mengimplementasikan suatu sistem log. Setiap terjadi perubahan, secara otomatis akan menyimpan ke tabel log.

Sintak umum trigger pada DBMS MySQL.

```
CREATE TRIGGER name
  [BEFORE|AFTER] [INSERT|UPDATE|DELETE]
  ON tablename
  FOR EACH ROW statement
```

Keterangan dari bentuk umum perintah membuat trigger:

a. ***name***, Nama trigger mengikuti peraturan penamaan variabel / identifier dalam MySQL
b. **[*BEFORE* | *AFTER*]** digunakan untuk menentukan kapan proses secara otomatis akan dieksekusi, sebelum atau sesudah proses.
c. **[*INSERT* | *UPDATE* | *DELETE*]** digunakan untuk menentukan event (proses) yang dijadikan trigger (pemicu) untuk menjalankan perintah- perintah di dalam triggers.
d. ***tablename***, merupakan nama tabel dimana trigger berada.
e. ***statement***, merupakan sekumpulan perintah atau query yang akan secara otomatis dijalankan jika event / proses yang didefinisikan sebelumnya aktif.

Statement atau perintah dalam *trigger* dapat berupa satu perintah saja, dan dapat juga beberapa perintah sekaligus. Jika terdapat beberapa perintah dalam *trigger*, maka gunakan perintah **BEGIN** dan **END** untuk mengawali dan mengakhiri perintah. Di dalam statement *trigger*, kita dapat mengakses record tabel sebelum atau sesudah proses dengan menggunakan **NEW** dan **OLD. NEW** digunakan

untuk mengambil record yang akan diproses (insert atau update), sedangkan **OLD** digunakan untuk mengakses record yang sudah diproses (update atau delete).

Berikut ini contoh trigger yang akan mencatat aktivitas ke tabel **log** setiap terjadi proses insert ke tabel pelanggan.

```
DELIMITER $

CREATE TRIGGER penjualan.before_insert
BEFORE INSERT ON

penjualan.pelanggan

FOR EACH ROW BEGIN

INSERT INTO `log` (description, `datetime`,
user_id)

VALUES (CONCAT('Insert data ke tabel
pelanggan id_plg = ', NEW.id_pelanggan),
now(), user());

END;$$

DELIMITER ;
```

24. DBMS PostgreSQL

PostgreSQL adalah sebuah sistem basis data yang disebarluaskan secara bebas menurut Perjanjian lisensi BSD. Piranti lunak ini merupakan salah satu basis data yang paling banyak digunakan saat ini, selain MySQL dan Oracle. PostgreSQL menyediakan fitur yang berguna untuk replikasi basis data. PostgreSQL telah mempelopori konsep-konsep objek relasional yang sekarang banyak digunakan oleh banyak database komersial. RDBMS tradisional memberikan dukungan model data yang terdiri atas sejumlah nama relasi yang

memuat atribut dari tipe tertentu. Pada sistem komersial saat ini, tipe yang mungkin termasuk floating point number, integer, character string, money dan date. Model ini kurang mencukupi untuk aplikasi pemrosesan data di masa depan. Model relasional berhasil menggantikan model sebelumnya karena kesederhanaannya. Kesederhanaan ini membuat implementasi dari aplikasi tertentu menjadi sulit. PostgreSQL menawarkan peningkatan kemampuan dengan menggabungkan mencukupi untuk aplikasi pemrosesan data di masa depan. Model relasional berhasil menggantikan model sebelumnya karena kesederhanaannya. Kesederhanaan ini membuat implementasi dari aplikasi tertentu menjadi sulit. PostgreSQL menawarkan peningkatan kemampuan dengan menggabungkan konsep tertentu sehingga user bisa dengan mudah memperluas sistemnya, seperti penyediaan fitur inheritance, tipe data dan fungsi. Selain itu postgreSQL juga memiliki fitur lain yang berfungsi untuk menambahkan kekuatan dan fleksibilitas, misalnya: constraint, trigger, rule dan transaction *integrity*. (Ardiansyah & Rainis, 2013)

Ada beberapa jalan untuk mengukur kualitas perangkat lunak, yaitu fitur, kinerja, reliabilitas, dukungan dan harga. Penjelasannya adalah sebagai berikut:

a. Fitur : PostgreSQL memiliki kebanyakan fitur yang ada pada produk DBMS komersial, seperti transaction, subselect, trigger, view, foreign key, referential integrity dan sophisticated locking. Juga terdapat fasilitas seperti user-defined type, inheritance, rule dan multi-version concurrency control untuk mengurangi lock contention.
b. Kinerja : setiap koneksi user ditangani dengan membuat proses unik. Proses back-end akan melakukan share buffer data dan mengunci informasi. Dengan menggunakan multiple CPU, maka multiple backend bisa berjalan dengan mudah pada CPU yang

berbeda. Jika dibandingkan dengan MySQL, PostgreSQL lebih lambat pada operasi insert dan update karena memiliki overhead transaksi. Saat ini fleksibilitas dan fitur sedang dikembangkan, sekaligus meningkatkan kinerja melalui profiling dan analisis *source code*.

c. Reliabilitas: suatu DBMS harus mempunyai reliabilitas. Kode yang direalease harus stabil dan memiliki jumlah bug minimal. Setiap release hendaknya mengalami paling tidak satu bulan beta testing. Dari hasil testing tersebut akan nampak *release* mana yang siap digunakan.

d. Dukungan : mailing list memungkinkan sejumlah besar *developer* dan *user* menangani masalah yang timbul. Akses langsung ke *developer,* komunitas *user*, manual dan kode sumber membuat dukungan PostgreSQL cukup superior dibanding dengan DBMS lain. Selain itu, juga tersedia pula dukungan komersial untuk yang memerlukannya.

e. Harga : PostgreSQL gratis untuk semua pemakaian, baik komersial maupun tidak. Kita bisa pula menambahkan kode tanpa adanya batasan, kecuali pada bagian yang dinyatakan dalam BSD.

2.4.1. Versi PostgreSQL

Pertengahan 1996, postgreSQL (baca: *post-grés-kju-él*), dengan label versi dimulai dari angka 6.0 (versi terakhir dari Postgres/Berkeley adalah 4.2, dan Postgres95 dianggap versi 5.x). Di sinilah, dan juga berlanjut di keluarga 7.0–7.1, banyak terjadi peningkatan dalam hal skalabilitas, fitur, dan kecepatan.

Meskipun demikian, perbaikan berlangsung tidak secara tiba-tiba, melainkan berangsur-angsur. Para pengembangnya perlu terlebih dulu masih perlu membenahi kode-kode lama dan kode yang belum sepenuhnya dimengerti. Hingga versi 6.4 (1998) misalnya di mana banyak ditambahkan fitur baru seperti dukungan karakter

internasional, bahasa stored procedure baru, view, dan beberapa sintaks SQL tambahan banyak terjadi masalah stabilitas. Beberapa pemakai melaporkan menjalankan proses server PostgreSQL yang lalu secara misterius tiba-tiba mati tanpa laporan apa-apa di log alias crash. Sebagian yang lain melaporkan diskonek secara acak. Dan sebagian lagi mengeluhkan kurang memuaskannya kinerja PostgreSQL. Bahkan ada pemakai yang membelot ke MySQL. Periode ini merupakan saat-saat yang cukup mengkhawatirkan bagi popularitas PostgreSQL contohnya, lihat *www.phpbuilder.com/columns/tim20000705.php3* dimana Tim Perdue menceritakan bahwa di tahun 1999, ia terpaksa beralih ke MySQL dalam membangun *SourceForge*. Kinerja PostgreSQL terlalu berbeda dengan MySQL sehingga mau tak mau pengguna setia PostgreSQL ini harus berganti *database*. (Yuliardi, 2000)

Versi 6.5 menurut pengembang PostgreSQL merupakan babak baru pemahaman mereka terhadap keseluruhan *source code* PostgreSQL. Versi ini juga merupakan versi perbaikan bug yang penting; ada banyak bug seperti berbagai kasus crash, kebocoran memori, dan kejanggalan/kekurangan pada sintaks SQL-nya diperbaiki. Selain itu, di versi 6.5 ditambahkan MVCC oleh Vadim, yang berpotensi meningkatkan kinerja PostgreSQL secara signifikan. MVCC, *atau Multi Version Concurrency Control*, serupa dengan InnoDB pada MySQL dalam hal memberikan kemampuan PostgreSQL memperlihatkan lebih dari satu versi tampilan data bagi klien. Perubahan data yang dibuat oleh klien yang sedang melakukan transaksi tidak akan terlihat dulu oleh klien lain sebelum transaksi dicommit. Ini menghindari locking yang tidak perlu.

Versi 6.5.x (1999, seri terakhir dari 6.x) cukup berhasil dan memuaskan bagi para pemakainya. Namun masih ada beberapa kekurangan PostgreSQL yang dirasakan mengganjal bagi banyak orang. Kekurangan-kekurangan ini lambat laun diperbaiki di seri

7.x, dan menurut (Momjian, 2001), di seri 7.3 ia berharap PostgreSQL akan sepenuhnya layak dan sebanding dengan database komersial dalam hal fitur penting. Satu keterbatasan yang paling menyebalkan yaitu ukuran data maksimum sebuah field hanya 8–32KB. Ini menyebabkan orang sulit menyimpan teks panjang atau gambar di dalam database. Keterbatasan ini akhirnya dihapuskan di 7.1. Penambahan penting lainnya antara lain foreign key constraint (ditambahkan di 7.0), write-ahead logging untuk peningkatan keamanan dan kinerja (7.1), serta *OUTER JOIN*. Masih ada lagi fitur seperti replikasi yang rencananya akan ditambahkan setelah 7.2.

Pengguna setia PostgreSQL boleh berbangga dengan seri 7.x. Di seri ini PostgreSQL mulai menantang dan bahkan mengungguli MySQL dalam hal kecepatan, terutama di *query-query* kompleks dan pada kondisi load tinggi. Dalam artikelnya Tim Perdue melaporkan hasil benchmark MySQL 3.23 dan PostgreSQL 7.0 dan kesimpulannya adalah: PostgreSQL memang telah menjadi semakin baik. Dan kecepatannya cukup mengagumkan dan stabil. Versi terbaru PostgreSQL sendiri untuk saat ini adalah PostgreSQL 9.1 yang masih beta dan pada versi 7.0 keatas PostgreSQL sudah mendukung *log dan trigger*.

2.4.2. Fitur DBMS PostgreSQL

Terdapat beberapa fitur dari postgreSQL yang berkaitan dengan query, misalnya:

a. Foreign key: dukungan standar *CREATE TABLE ... FOREIGN KEY* untuk *referential integrity*. Dukungan aksi berbeda untuk melakukan update dan menghapus data termasuk cascading, restricting dan restoring ke nilai *default* atau NULL. Fitur ini sangat penting untuk integritas data di banyak aplikasi. Foreign key bisa dideteksi dengan alat bantu pemodelan basis data

seperti Erwin untuk memudahkan perancangan dan dokumentasi basis data.

b. Join: mengimplementasikan tipe join SQL99: *inner join, left, right, full outer join, natural join*.

c. *View*: digunakan untuk menyimpan statement SQL select. *View* bisa dipergunakan untuk melakukan enkapsulasi *query* yang kompleks pada level *server* dan implementasi granularitas akses *privilege*. Keberadaan view akan menyederhanakan perancangan dan pemeliharaan basis data.

d. *Trigger*: *trigger* adalah prosedur yang dipanggil oleh *database* pada saat tertentu, umumnya jika terjadi insert atau update. Penerapan pada logging setiap waktu suatu record disisipkan atau dihapus atau untuk *update* suatu *field* setiap kali *field* lainnya berubah. Dukungan *trigger* bisa dilakukan sebelum atau sesudah suatu perintah dijalankan. Fungsi trigger bisa ditulis dalam bahasa C atau bahasa *prosedural*.

e. Menyediakan tipe tambahan seperti konstruksi geometris (titik, garis dan semacamnya), pengalamatan jaringan *TCP/IP, ID ethernet card, ISBN/ISSN* dan lain-lain.

f. Tipe baru bisa didefinisikan dengan fungsi dan operator yang diperlukan

g. PostgreSQL mendukung penyimpanan binary *large object*, termasuk gambar, suara atau video. Objek ini bisa diambil sebagian atau seluruhnya oleh aplikasi klien.

h. Dukungan pada international *character set*, *multibyte character encodings dan unicode.*

i. Perhatian pada *sorting, case-sensitivity, dan formatting*. (Belluano, 2017)

2.4.3. Log DBMS PostgreSQL

Untuk melakukan logging atau mengaktifkan fitur *log postgresql* dapat dilakukan pada konfigurasi di file *postgresql.conf*. Dalam file *postgresql.conf* ini terdapat parameter yang harus diaktifkan untuk melakukan *log* atau audit. Parameter tersebut adalah:

1. *Log Destination*

   Log detination merupakan parameter yang digunakan untuk menentukan penyimpanan yang akan di audit. Terdapat 4 pilihan untuk parameter ini yaitu *stderr, csvlog, syslog* dan *event log*. Default pilihannya adalah stderr. Pilihan csvlog dapat digunakan untuk mengcopy data log ke sebuah tabel.

2. *Logging Collector*

   Digunakan untuk mengaktifkan catatan dari parameter log destination. Jika memilih salah satu parameter *dari log destination*, *log collector* ini harus diset ON.

3. *Log Directory*

   Digunakan untuk menset direktori tempat penyimpanan hasil audit. Default pilihannya adalah *pg_log*.

4. *Log Filename*

   Digunakan untuk menentukan nama dari file *log* yang dihasilkan. Default nama file log adalah postgresql-%Y-%m-%d_%H%M%S.log.

5. *Log Line Prefix*

   Digunakan untuk menentukan hal-hal apa saja yang akan ditampilkan dalam file log. Untuk menentukan hal tersebut parameter ini harus diset dengan parameter yang telah tersedia.

Berikut ini adalah parameter yang tersedia dapat dilihat pada tabel 2.2

Tabel 2.2 Parameter Log line prefix

Parameter	Keteranga
%a	Menampilkan nama aplikasi
%u	Menampilkan nama user
%d	Menampilkan nama database
%r	Menampilkan remote host dan port
%h	Menampilkan remote host
%p	Menampilkan Proses ID
%t	Menampilkan timestamp tanpa miliseconds
%m	Menampilkan timestamp dan miliseconds
%i	Menampilkan command tag
%e	Menampilkan SQL state
%c	Menampilkan Session ID
%l	Menampilkan Session line number
%s	Menampilkan Session start timestamp
%v	Menampilkan Virtual transaction ID
%x	Menampilkan Transaksion ID
%q	Mengakhiri audit jika tidak ada session lagi

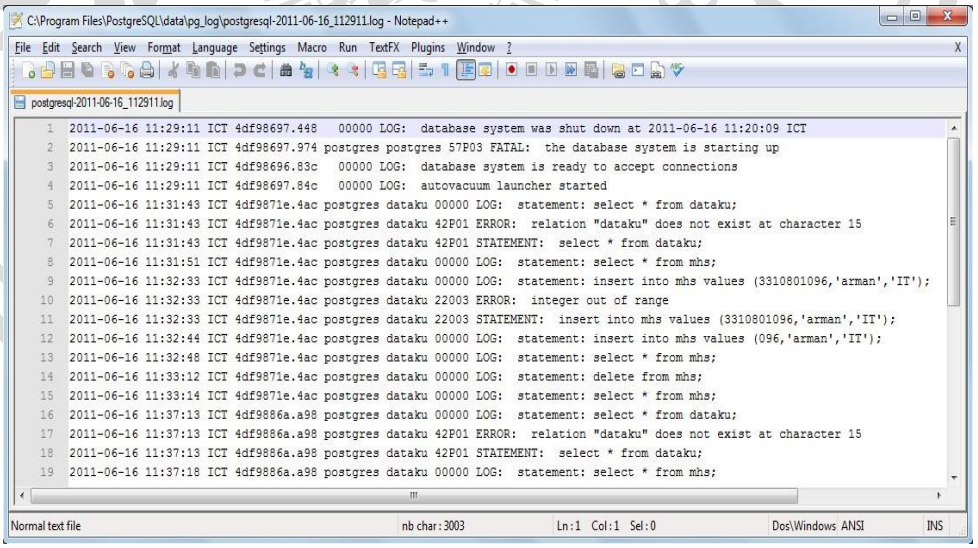
6. *Log Statement*

Digunakan untuk mensetting statement apa saja yang akan di audit terdapat 3 pilihan value yang dapat dilihat pada tabel 2.3.

Tabel 2.3 Parameter Log line prefix

Parameter	Keterangan
None	Tidak menammpilkan apapun
DDL	Menampilkan perintah DDL (insert,update,select dan delete)
DM	Menampilkan perintah DML (create table, alter table dan grant)
ALL	Menampilkan semua (DDL dan DML)

Berikut ini contoh hasil dari log postgresql yang dapat dilihat pada gambar 2.2



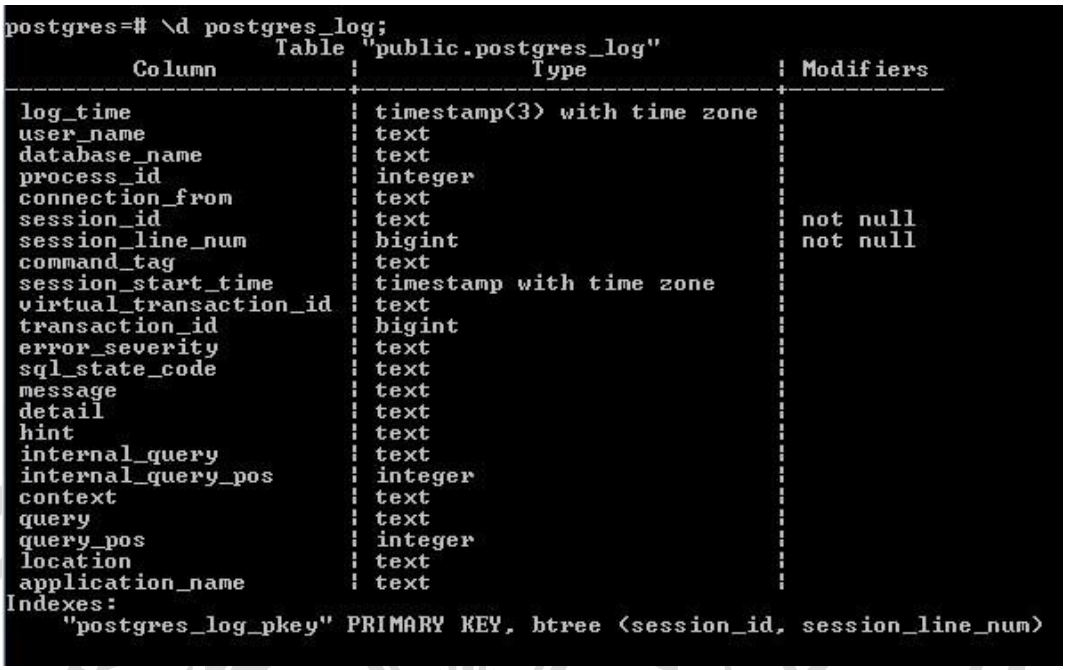
Gambar 2.2 Contoh Log PostgreSQL

Untuk membaca Log PostgreSQL dapat menggunakan aplikasi teks editor seperti *Wordpad*, *notepad++*, dan lainya. Pemantauan *log* dapat menjadi sebuah pekerjaan yang sangat melelahkan karena jika jumlah data yang telah di catat banyak maka proses pencarian data sangat sulit sehingga diperlukan suatu cara untuk mengimpor file log tersebut kedalam sebuah table. PostgreSQL sendiri mendukung untuk melakukan *loging* ke suatu tabel akan tetapi terdapat langkah-langkah yang harus dilakukan jika ingin melakukan logging kesebuah tabel di DMBS postgresql berikut ini adalah tahap-tahap yang harus dilakukan.

Parameter dalam file postgresql.conf diset sebagai berikut:

1. logging_collector = on
2. logging_destination = csvlog
3. log_min_error_statement = on
4. log_error_verbosity = verbose
5. log_directory = /var/log/postgresql
6. log_filename = postgresql-%H
7. log_rotation_age = 1
8. log_truncate_on_rotation = on
9. log_rotation_size = 0 # disabled

Buat tabel untuk menampung data-data dari file log tersebut. Dokumentasi dari PostgreSQL menawarkan contoh tabel sebagai berikut untuk membuat suatu file log CSV. Berikut deskripsi tabel yang dimaksud:



```
postgres=# \d postgres_log;
                  Table "public.postgres_log"
         Column          |            Type             | Modifiers
-------------------------+-----------------------------+-----------
 log_time                | timestamp(3) with time zone |
 user_name               | text                        |
 database_name           | text                        |
 process_id              | integer                     |
 connection_from         | text                        |
 session_id              | text                        | not null
 session_line_num        | bigint                      | not null
 command_tag             | text                        |
 session_start_time      | timestamp with time zone    |
 virtual_transaction_id  | text                        |
 transaction_id          | bigint                      |
 error_severity          | text                        |
 sql_state_code          | text                        |
 message                 | text                        |
 detail                  | text                        |
 hint                    | text                        |
 internal_query          | text                        |
 internal_query_pos      | integer                     |
 context                 | text                        |
 query                   | text                        |
 query_pos               | integer                     |
 location                | text                        |
 application_name        | text                        |
Indexes:
    "postgres_log_pkey" PRIMARY KEY, btree (session_id, session_line_num)
```

Gambar 2.3 Deskripsi tabel postgres_log

Setelah tabel tersebut dibuat berikut ini adalah perintah yang digunakan untuk mengcopy file dari file CSV ke dalam tabel postgres_log.

COPY postgres_log FROM '/full/path/to/logfile.csv' WITH csv;

2.4.4 Trigger DBMS PostgreSQL

Trigger DBMS PostgreSQL adalah fungsi yang akan di*eksekusi* sebelum atau sesudah proses *INSERT, UPDATE* atau *DELETE* pada suatu tabel, baik untuk setiap perubahan record pada tabel maupun tiap kali perintah SQL dijalankan. Fungsi yang akan dijalankan oleh Trigger harus didefinisikan dahulu sebelum Trigger diciptakan. Fungsi yang didefinisikan harus tanpa argumen dan mempunyai nilai balik trigger.

Contoh penggunaan trigger di PostgreSQL.

```
-- nama fungsi fgUpdateStok

CREATE OR REPLACE FUNCTION fgUpdateStok() RETURNS
TRIGGER as '

DECLARE

current_stok int4;

BEGIN

select into current_stok stok from produk where id=

NEW.id_produk;

current_stok = current_stok + NEW.jumlah;

update produk set stok = current_stok where id=
NEW.id_produk;

return NEW;

END

' LANGUAGE 'plpgsql';
```

Penjelasan:

```
CREATE OR REPLACE FUNCTION fgUpdateStok() RETURNS
TRIGGER as '
```

Fungsi bernama fgUpdateStok dan mempunyai nilai balik trigger

```
current_stok int4;
```

deklarasikan variabel *current_stok* bertipe *integer* untuk menyimpan nilai stok

```
select into current_stok stok from
produk where id = NEW.id_produk;
```

memberi nilai variabel current_stok diambil dari hasil query ke tabel Produk untuk setiap penambahan produk yang di input melalui tabel Penerimaan, keyword *NEW* digunakan untuk menandakan record yang terbaru di input pada tabel Penerimaan.

```
current_stok = current_stok + NEW.jumlah;
```

menambahkan variabel current_stok dengan record baru *(Field jumlah)* dari table Penerimaan

```
update produk set stok = current_stok where id =
NEW.id_produk;
```

Update nilai Field stok pada tabel Produk dengan nilai yang baru (Setelah ditambahkan dengan jumlah penerimaan produk).

2.5 Penelitian Sebelumnya

Sebagai bahan pertimbangan dalam penelitian tentang metode Audit Trail sebagai salah satu keamanan pada databse, maka penulis akan mencantumkan beberapa hasil penelitian terdahulu oleh beberapa peneliti sebagai berikut:

1. Penelitian yang pertama yaitu oleh Abisyena & Githa, Agustus 2017 dengan judul "Implementasi Database Auditing dengan Memanfaatkan Sinkronisasi DBMS". Penelitian ini menerapakan database auditing yang dapat menjadi komponen penting dalam keamanan basisdata dan kepatuhan. Database Adminstrator perlu lebih waspada dalam teknik yang digunakan untuk melindungi data perusahaan, serta mamantau dan memasitkan bahwa perlindungan yang memadai terhadap data tersedia. Pada tingkat tinggi, database auditing merupakan fasilitas untuk melacak otoritas dan pengguna sumber dayaa database. Ketika fungsi audit diaktifkan, setiap operasi database yang diaudit menghasilkan jejak audit dari perubahan informasi yang dilakukan. Jejak audit dari operasi database yang dihasilkan, memungkinkan DBA (Database Administrator) memelihara audit trail dari waktu ke waktu, untuk melakukan analisis tentang pola akses dan modifikasi terhadap data pada DBMS (Database Management System).

2. Penelitian keua yaitu (Ruslam, Yuwono, Aviandi, & Indrawati, 2013) dengan judul "AUDIT KEAMANAN INFORMASI: STUDI KASUS PT XYZ" . Data yang digunakan dalam penelitian ini adalah percetakan billing statement atau rekening koran dan penyediaan jasa kurir. Dalam menjalankan bisnisnya, PT XYZ bekerja sama dengan beberapa bank di Indonesia (lokal) dan bank asing, salah satunya adalah Bank ABC. Bank ABC, sebagai salah satu pelanggan PT XYZ merupakan bank yang sudah melakukan sertifikasi ISO 27001. Bank ABC melakukan audit keamanan informasi pada PT XYZ selama kurang lebih 2 (dua) bulan, yaitu dari bulan Februari 2012 sampai dengan bulan Maret 2012. Di mana Bank ABC melakukan audit keamanan informasi di PT XYZ, yang meliputi audit terhadap informasi yang diterima, informasi yang diproses, serta informasi yang dicetak dan didistribusikan kepada pelanggan sesuai alamat pelanggan. Permasalahan yang dihadapi dalam penelitian ini yaitu divisi TI PT. XYZ mengalami kesulitan untuk mengetahui batasan-batasan atau ruang lingkup audit yang digunakan oleh bank ABC, sehingga divisi TI hanya terbatas pada audit kepatuhan keamanan informasi terhadap visi dan misi best services perusahaan. Jika di tengah jalan didapatkan kebijakan dan prosedur yang lemah maka akan direkomendasikan kebijakan dan prosedur baru. Penelitian ini dilakukan untuk menyelesaikan permasalahan keamanan informasi melalui audit keamanan informasi dengan memberikan rekomendasi perbaikan kebijakan dan prosedur keamanan informasi sehingga lebih komprehensif sesuai praktik terbaik dan standar internasional keamanan informasi, ISO/IEC 27001: 2005, serta terpenuhinya kriteria keamanan informasi di lingkungan PT XYZ dan pada akhirnya dapat mendukung PT XYZ dalam memberikan best services

pengelolaan dokumen kepada pelanggan sesuai dengan visi dan misi PT XYZ. Sedangkan ruang lingkup dan batasan dalam penelitian ini yaitu hanya terbatas pada audit kepatuhan keamanan informasi untuk memberikan rekomendasi kebijakan dan prosedur guna meningkatkan keamanan informasi di lingkungan PT XYZ sesuai dengan hasil audit yang dilakukan oleh peneliti. Penelitian ini tidak mencakup implementasi, monitoring, dan evaluasi dari rekomendasi perbaikan kebijakan dan prosedur.


UNIVERSITAS MUHAMMADIYAH
GRESIK