IMPLEMENTASI AUDIT DBMS ORACLE 10g PADA DBMS MYSQL DAN DBMS POSTGRESQL

TUGAS AKHIR

Oleh :

Arman Juniardi **3310801096**

Iwan Kurniawan **3310801097**

Disusun untuk memenuhi syarat kelulusan Program Diploma III

PROGRAM STUDI TEKNIK INFORMATIKA

POLITEKNIK NEGERI BATAM

BATAM

2011

LEMBAR PENGESAHAN

Batam, 16 Agustus 2011

Pembimbing,

Mira Chandra Kirana, ST

NIK. 109064

LEMBAR PERNYATAAN

Dengan ini, saya:

NIM : 3310801096

Nama : Arman Juniardi

adalah mahasiswa Teknik Informatika Politeknik Negeri Batam yang menyatakan bahwa tugas akhir dengan judul:

IMPLEMENTASI AUDIT DBMS ORACLE 10g PADA DBMS MYSQL DAN DBMS POSTGRESQL

disusun dengan:

1. tidak melakukan plagiat terhadap naskah karya orang lain
2. tidak melakukan pemalsuan data
3. tidak menggunakan karya orang lain tanpa menyebut sumber asli atau tanpa ijin pemilik

Jika kemudian terbukti terjadi pelanggaran terhadap pernyataan di atas, maka saya bersedia menerima sanksi apapun termasuk pencabutan gelar akademik.

Lembar pernyataan ini juga memberikan hak kepada Politeknik Negeri Batam untuk mempergunakan, mendistribusikan ataupun memproduksi ulang seluruh hasil Tugas Akhir ini.

Batam, 16 Agustus 2011

Arman Juniardi
3310801096

LEMBAR PERNYATAAN

Dengan ini, saya:

NIM : 3310801097

Nama : Iwan Kurniawan

adalah mahasiswa Teknik Informatika Politeknik Negeri Batam yang menyatakan bahwa tugas akhir dengan judul:

IMPLEMENTASI AUDIT DBMS ORACLE 10g PADA DBMS MYSQL DAN DBMS POSTGRESQL

disusun dengan:

1. tidak melakukan plagiat terhadap naskah karya orang lain
2. tidak melakukan pemalsuan data
3. tidak menggunakan karya orang lain tanpa menyebut sumber asli atau tanpa ijin pemilik

Jika kemudian terbukti terjadi pelanggaran terhadap pernyataan di atas, maka saya bersedia menerima sanksi apapun termasuk pencabutan gelar akademik.

Lembar pernyataan ini juga memberikan hak kepada Politeknik Negeri Batam untuk mempergunakan, mendistribusikan ataupun memproduksi ulang seluruh hasil Tugas Akhir ini.

Batam, 16 Agustus 2011

Iwan Kurniawan
3310801097

KATA PENGANTAR

Puji syukur kepada Allah SWT atas rahmat dan karunia-Nya sehingga Matakuliah Tugas Akhir ini dapat diselesaikan dengan judul "IMPLEMENTASI AUDIT DBMS ORACLE 10g PADA DBMS MYSQL DAN DBMS POSTGRESQL".

Pada kesempatan ini penulis ingin mengucapkan terima kasih kepada:

1. Kepada orang tua yang telah memberikan semangat dan dukungan.
2. Bapak Uuf Brajawidagda selaku Ketua Prodi Teknik Informatika sekaligus Dosen Pengampu Mata Kuliah Tugas Akhir.
3. Ibu Mira Chandra Kirana selaku pembimbing yang selalu siap membantu saat diperlukan dan memberikan bimbingan serta masukan yang sangat berguna.
4. Para dosen serta rekan-rekan mahasiswa dan seluruh pihak yang ikut serta membantu sehingga Tugas Akhir ini dapat diselesaikan.

Penulis menyadari bahwa selama melaksanakan Tugas Akhir banyak hal-hal yang dapat dipelajari dan semuanya tidak lepas dari kesalahan maupun kekurangan. Akhir kata mohon maaf atas kesalahan dan kekurangan serta keterbatasan pada Tugas Akhir ini. Kritik dan saran yang membangun sangatlah diharapkan. Semoga Tugas Akhir ini bermanfaat dan bisa dikembangkan pada masa yang akan datang.

Batam, 16 Agustus 2011

Penulis

ABSTRAK

IMPLEMENTASI AUDIT DBMS ORACLE 10g PADA DBMS MYSQL DAN DBMS POSTGRESQL

Oracle 10*g* merupakan *database management system* (DBMS) yang memiliki keamanan *database auditing* yang disebut dengan fitur audit yaitu *audit trail*, *fine grained audit* dan *value base auditing.* Selain Oracle 10g, MySQL dan PostgreSQL juga dapat digunakan untuk melakukan *database auditing*. Implementasi audit Oracle 10g pada MySQL dan PostgreSQL dilakukan untuk mengetahui jenis audit Oracle 10g apa saja yang dapat diimplementasikan pada MySQL dan PostgreSQL.

Untuk mencapai hal tersebut, hal pertama yang dilakukan adalah melakukan implementasi fitur audit Oracle 10g. Hasil implementasi tersebut dijadikan acuan dalam implementasi pada MySQL dan PostgreSQL. Pada MySQL, implementasi dilakukan pada *file log* yaitu menggunakan *general log* dengan hasil penyimpanan *log* ke *database*. Untuk PostgreSQL implementasi dilakukan pada fitur *log* PostgreSQL dengan hasil penyimpanan *log* ke *database*. Hal kedua yang dilakukan yaitu membuat tabel untuk menampung hasil audit, membuat *trigger* dan *function.*

Dari hasil implementasi yang telah dilakukan, audit oracle 10g yang dapat diimplementasikan pada MySQL dan PostgreSQL adalah *Audit trail* dengan menggunakan *general log* dan *log postgresql*. *Value base auditing* dengan menggunakan *trigger* dan *function*. Untuk *fine grained auditing* sudah dapat dilakukan dengan tanpa kondisi sedangkan untuk menggunakan kondisi hal ini belum dapat dilakukan karena perbedaan tipe data pada MySQL dan PostgreSQL.

Kata Kunci: *database management system, database, database auditing, audit trail, fine grained audit, value base auditing, log postgresql, log file, general log, trigger, function,* Oracle 10*g*, MySQL dan PostgreSQL.

ABSTRACT

IMPLEMENTATION OF AUDIT DBMS ORACLE 10g ON DBMS MYSQL AND DBMS POSTGRESQL

Oracle 10g is a database management system (DBMS) which has a security auditing database called the audit features namely audit trail, fine-grained auditing and value base auditing. In addition to Oracle 10g, MySQL and PostgreSQL can also be used to perform database auditing. Oracle 10g audit implementation toward the MySQL and PostgreSQL conducted to determine the type of Oracle 10g audit anything that can be implemented in MySQL and PostgreSQL.

To achieve this, the first thing done is implement auditing features of Oracle 10g. The results are used as a reference implementation in the implementation of the MySQL and PostgreSQL. In MySQL, the implementation is done on the log file using the general log to log to a database storing the result. For PostgreSQL implementation done on the features of PostgreSQL logs with the results of log storage to database. The second thing to do is create a table to accommodate the results of the audit, create triggers and functions.

From the results of the implementation has been done, the audit oracle 10g that can be implemented in MySQL and PostgreSQL are the audit trail by using the general log and log postgresql. Value base auditing using triggers and functions. For fine grained auditing can be performed with no conditions while using this condition can not be done because different types of data in MySQL and PostgreSQL.

Keywords: database management system, database, database auditing, audit trail, fine-grained auditing, value base auditing, postgresql logs, log files, general log, trigger, function, Oracle 10g, MySQL and PostgreSQL.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Bab I Pendahuluan

I.1 Latar Belakang

Keamanan database merupakan suatu proteksi terhadap pemakaian data oleh pemakai yang tidak punya kewenangan ataupun yang mempunyai kewenangan. Kebutuhan akan keamanan basis data *(database)* timbul dari kebutuhan untuk melindungi data[1]. Salah satunya adalah dari pengguna (*user)* yang memiliki wewenang memanipulasi data. Penyalahgunaan database oleh *user* ada yang di sengaja dan tidak disengaja. Yang tidak disengaja seperti kesalahan dalam memasukkan data pada database sedangkan yang disengaja seperti merubah data untuk kepentingan pribadi. Sebagai contoh kasus penyalahgunaan database yang disengaja, seorang pegawai bank swasta yang melakukan penggelapan uang. Pelaku mengalihkan dana milik nasabah ke rekening pelaku tersebut[2].

Untuk mengawasi user yang memiliki wewenang memanipulasi data dibutuhkanlah *database management system* (DBMS) yang memiliki keamanan *database auditing*. *Database auditing* merupakan bagian dari keamanan DBMS yang bertujuan untuk memantau aktifitas yang dilakukan oleh pengguna *database*. Oracle 10g merupakan DBMS yang memiliki keamanan *database auditing* yang disebut dengan fitur audit. Fitur audit ini digunakan untuk mencatat informasi, yaitu informasi aktifitas yang mencurigakan pada suatu *database* yang dilakukan oleh *user* dengan menggunakan fasilitas *audit trail*, *fine grained audit* dan *value base auditing*. Fasilitas *audit trail* ini digunakan untuk mengetahui *user* mana saja dan apa saja yang dilakukan oleh *user* tehadap objek-objek database. Dan *fine grained audit* sendiri digunakan untuk dapat mengamati dan melacak kegiatan

[1] http://catur.dosen.akprind.ac.id/2011/02/07/perlunya-keamanan-database/
[2] http://www.detiknews.com/read/2011/03/25/145918/1601341/10/tilep-duit-nasabah-rp-17-m-karyawan-bank-swasta-ditahan-polisi[2] http://www.detiknews.com/read/2011

user yang memanipulasi isi dari tabel *database*. Sedangankan *value base auditing* mencatat perubahan yang terjadi pada suatu objek database berdasarkan isi dari database.

Selain Oracle 10g, terdapat beberapa *software* DBMS yang ada yaitu SQL Server, Microsoft Access, Microsoft Visual FoxPro, IBM DB2, MySQL, PostgreSQL, FirebirdSQL dan Paradox. Dari *software-software* DBMS yang ada tersebut, MySQL dan PostgreSQL merupakan *software* yang bersifat *open source* dan *multiplatform* yaitu dapat berjalan diberbagai *operating system* seperti *windows* dan *linux*. Selain itu MySQL dan PostgreSQL mempunyai fitur yang dapat mencatat aktifitas yang dilakukan pengguna *database* yang disebut *log*. *Log* ini digunakan untuk mencatat informasi yaitu akitifitas dan even seperti menghapus, mengedit dan menambah data di *database* dengan menggunakan fasilitas yang dimiliki. Dalam MySQL fasilitas yang digunakan adalah *general log* yang berfungsi untuk mencatat apa saja yang dilakukan *user* terhadap objek-objek database. Sedangkan pada PostgreSQL, fasilitas yang digunakan adalah *log* PostgreSQL yang berfungsi untuk mencatat segala kegiatan pengguna *database*.

Dari beberapa penjelasan tersebut Oracle 10g memiliki fitur audit yang lebih lengkap dalam melakukan *database auditing*. Oleh karena itu, dilakukanlah implementasi audit Oracle 10g pada MySQL dan PostgreSQL. Dengan tujuan untuk mengetahui jenis audit Oracle 10g apa saja yang dapat diimplementasikan pada MySQL dan PostgreSQL.

I.2 Rumusan Masalah

Rumusan masalah dari tugas akhir ini yaitu bagaimana cara menerapkan audit DBMS Oracle 10g pada DBMS MySQL dan DBMS PostgreSQL dan jenis audit Oracle 10g apa saja yang dapat diimplementasikan pada DBMS MySQL dan DBMS PostgreSQL.

I.3 Batasan Masalah

Batasan masalah dari tugas akhir ini yaitu:

- Hanya mengimplementasikan fitur *audit trail*, *value base auditing* dan *fine grained auditing* pada DBMS MySQL dan DBMS PostgreSQL dengan DBMS Oracle 10g sebagai acuan.
- Versi DBMS MySQL yang digunakan adalah MySQL 5.5.8 dan versi DBMS PostgreSQL yang digunakan adalah PostgreSQL 9.0.

I.4 Tujuan

Tujuan dari tugas akhir ini adalah:

- Mengetahui cara menerapkan audit Oracle 10g pada DBMS MySQL dan DBMS PostgreSQL.
- Mengetahui jenis audit Oracle 10g apa saja yang dapat diimplementasikan pada DBMS MySQL dan DBMS PostgreSQL.

I.5 Sistematika Penulisan

Sistematika penulisan laporan Tugas Akhir ini adalah sebagai berikut:

BAB I Pendahuluan

Berisikan latar belakang, rumusan masalah, batasan masalah, tujuan dan sistematika penulisan

BAB II Landasan Teori

Berisikan penjelasan tentang Database, DBMS, DBMS Oracel 10g, Fitur DBMS Oracle 10g, Audit DBMS Oracle 10g, *Trigger* DBMS Oracle 10g, DBMS MySQL, *Log file* MySQL, *Trigger* DBMS MySQL, DBMS PostgreSQL, Fitur DBMS PostgreSQL, *Log* PostgreSQL dan *Trigger* DBMS PostgreSQL.

BAB III Analisis

Berisikan penjelasan tentang audit DBMS Oracle 10g, Audit pada DBMS MySQL, Audit pada DBMS PostgreSQL dan Langkah penyelesaian Masalah.

BAB IV Perancangan

Berisikan rancangan *database*, rancangan tabel untuk menampung hasil audit, rancangan trigger audit pada DBMS Oracle 10g, rancangan trigger audit pada DBMS MySQL dan rancangan trigger audit pada DBMS PostgreSQL.

BAB V Implementasi dan Pengujian

Berisikan implementasi rancangan database untuk menampung hasil audit, implementasi trigger audit pada DBMS Oracle 10g, implementasi trigger audit pada DBMS MySQL dan implemetasi trigger audit pada DBMS PostgreSQL. Dan Pengujian berisikan skenario pengujian, hasil pengujian terhadap trigger audit pada DBMS Oracle 10g, hasil pengujian terhadap trigger audit pada DBMS MySQL dan hasil pengujian terhadap trigger audit pada DBMS PostgreSQL.

BAB VI Kesimpulan Dan Saran

Berisi kesimpulan dari hasil penelitian dan saran untuk pengembangan penelitian.

Bab II Landasan Teori

II.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 **Henry F.Korth** ada tiga model data basis data yaitu:

- Model data Hirarkis

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

- 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.

- 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:

  - Data dapat diakses secara cepat.
  - Struktur basis data mudah diubah.

- Data disajikan secara logis sehingga pengguna tidak perlu mengetahui bagaimana data disimpan.
- Pengguna mudah membuat query yang kompleks untuk mengambil data. Pengguna mudah menerapkan integritas data.
- Pengguna mudah membuat dan memodifikasi program aplikasi.
- Bahasa standar (SQL) sudah dibuat.

Kekurangan model basis data relasional adalah:

- Kelompok informasi atau tabel yang berbeda harus dihubungkan untuk mengambil data.
- Pengguna harus memahami hubungan antartabel.
- Pengguna harus belajar SQL[3].

II.1.1 Kemanan *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.

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[4].

---

[3] http://blog.re.or.id/model-basis-data.htm
[4] http://akbar.staff.gunadarma.ac.id

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 [5].

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

- *Integritas*, system akan mempunyai integritas bila ia berjalan menurut spesifikasinya. Perancang system berusaha untuk mengembangkan system 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 system 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 system. Teknik yang efektif untuk mendapatkan daya kontrol system ini adalah dengan membagi system menjadi subsistem yang menangani transaksi secara terpisah.

II.1.2 *Structure Query Language* (SQL)

Pernyataan SQL dapat dikelompokkan menjadi 5 kelompok DDL, DML, DCL dan pengendali transaksi. Berikut penjulasan masing kelompok tersebut.

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

   DDL merupakan kelompok perintah yang berfungsi untuk mendefinisikan atribut-atribut *database*, table, atribut (kolom), batasan-batasan terhadap

[5]http://student.eepisits.edu/~pungky/Modul/Administrasi%20Basis%20Data%20%20Rengga/Day-09/11%20-%20Keamanan%20Database%20Oracle.pdf

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**

   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[6].

[6] http://lecturer.eepis-its.edu/~tessy/tutorial/sqlold/Introduction%20to%20Oracle.pdf

II.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 igunakan 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 :

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[7].

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 II.1.

[7] http://id.wikipedia.org/wiki/Sistem_manajemen_basis_data

Tabel II.1 Tabel perbandingan fitur DBMS

BDMS / Description	Postgresql	Ms Access	FoxPro	Paradox	Mysql	SQL Server	Oracle
Tipe	Client Server	File based server	File based server	File based server	Client Server	Client Server	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 Rollback	Yes	No	No	No	Yes, with InnoDB	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.

II.3 DBMS Oracle 10g

Oracle pertama kali di buat pada 1977 oleh Larry Ellison, Bob Miner, dan Ed Oates sebagai software Developmen Laboratories menjadi no-komersial relation database. Kemudian berdiri perusahaan yang bernama oracle, yang menjadikan produk komersial relation database management system pertama di pasaran. Sejak itu, oracle menempati posisi puncak dalam pasar relation database management system, bahkan dengan seiringnya waktu oracle membuat relation database server. Sampai saat ini, oracle telah berkembang pesat leading technology melalui pengenalan client/server database pertama (1986). 64 bit database (1995) dan linux database (1999). DBMS Oracle selalu pengalami pengembangan hingga DBMS Oracle 10g.

Oracle 10g dengan teknologi grid, lebih tepat didekrisipkan sebagai dynamic cluster, yang mana applikasi server dapat ditambahkan menjadi kluster jika diperlukan dan resource kluster dapat di susun ulang sesuai kebutuhan bisnis. Sebagai contoh, katakanlah kita memiliki enam server yang menjalankan OLTP (*Online Transaction Processing*) dan dua server sebagai data warehousing, dan dan kita menemukan bahwa OLTP server dalam keadaan rusak atau terganggu, dengan mengkombinasikan keenam server dalam sebuah grid, maka dua data warehouse server dapat menjadi bagian dari OLTP sementara waktu, dan sebagai *resource* yang sama.

II.3.1 Fitur DBMS Oracle 10g

DBMS Oracle 10g menyediakan banyak fitur yang belum tentu bisa dimiliki oleh relation database pada umumnya. Fitur-fitur yang disediakan oleh DBMS Oracle 10g secara umum mencakup delapan kategori yaitu :

1. *High Avaibility* yaitu memastikan akses dan ketersediaan data pada kondisi kritis.

2. *Scalability*

   Mampu dikembangkan untuk berbagai jenis skala bisnis yang dapat berkembang menjadi sangat besar.
3. *Security* yaitu digunakan untuk mendukung keamanan data yang terjamin.
4. *Application development* yaitu memiliki kemampuan untuk membangun berbagai applikasi.
5. *Manageability* yaitu memiliki kemampuan untuk beradaptasi, memonitor, mendiagnosa dan memperbaiki dirinya sendiri sehingga mudah dimanage.
6. *Datawarehousing* yaitu Oracle 10g menyediakan dasar teknologi untuk membangun businnes intelligance yang lengkap dan solusi data warehousing.
7. *Integration* yaitu Oracle 10g memperbolehkan customer untuk men-sinkronisasi dan asikronisasi datanya termasuk replication, distributed SQL dan banyak lagi.
8. *Content Management* yaitu Oracle 10g mendukung content management web bagi penggunanya.

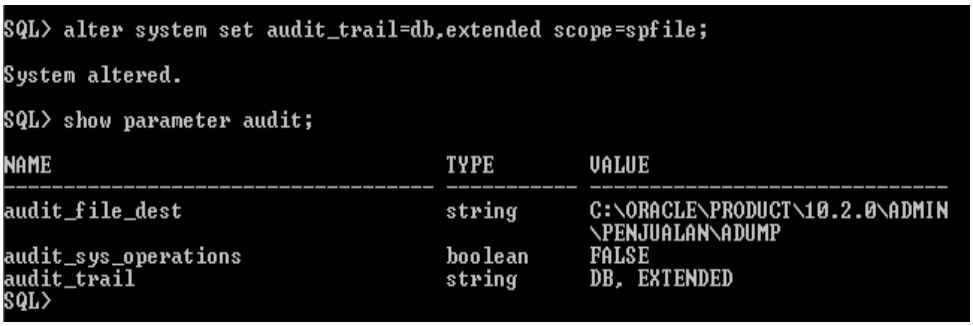
II.3.2 Audit DBMS Oracle 10g

Audit adalah suatu kegiatan yang digunakan untuk melakukan penyelidikan teknis dari suatu penggunaan data dan informasi yang terdapat dalam *database*. Aksi-aksi yang paling umum yang biasanya teraudit adalah memulai sebuah sesi, mematikan sesi dan melakukan koneksi kedatabase menggunakan hak-hak administrative. Jenis-jenis audit yang lain dapat mencatat perubahan yang dibuat untuk tabel-tabel atau tampilan-tampilan tertentu. Jejak audit disimpan ditempat yang aman dan dapat digunakan dalam berbagai cara. Dalam DBMS Oracle 10g terdapat beberapa fasilitas audit yang dapat digunakan yaitu:

1. Audit Trail

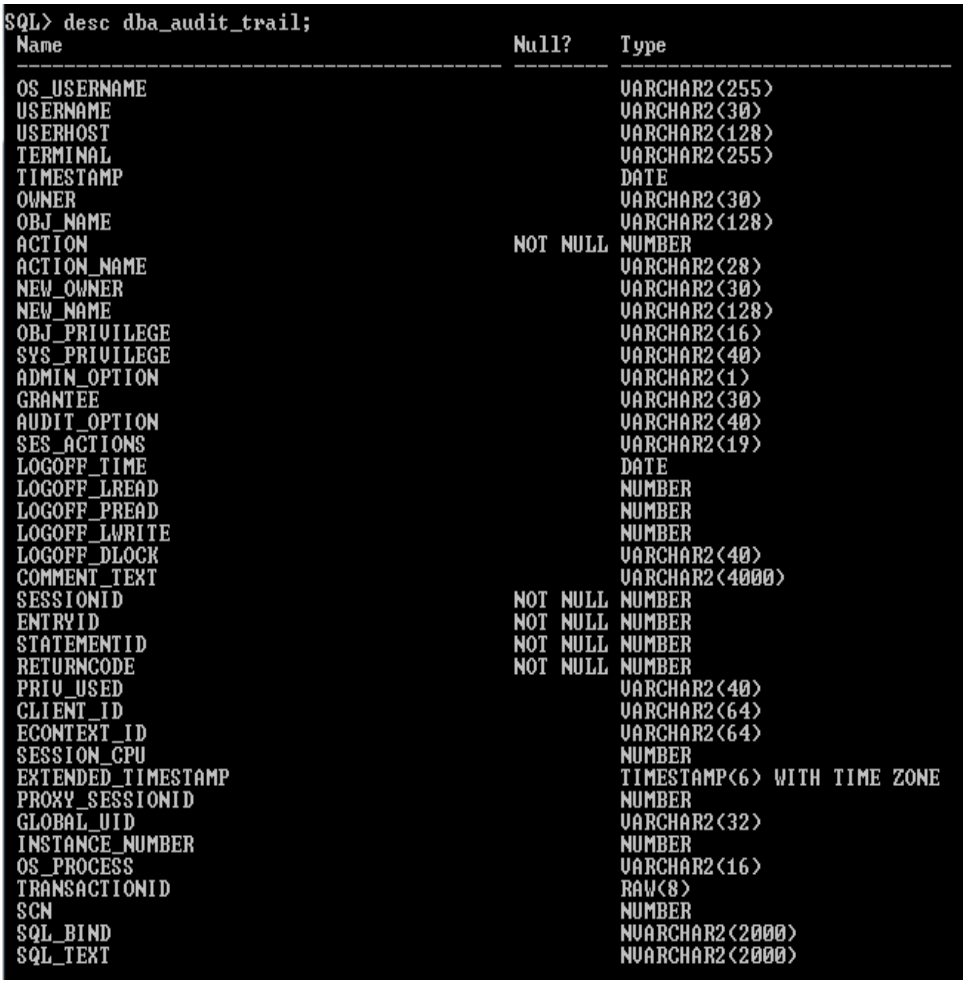
   Mencatat kegiatan yang dilakukan oleh pengguna database, seperti proses login dan logout, mencatat perintah-perintah yang dilakukan pengguna

tehadap object database dan mencatat perintah apa saja yang terjadi pada object tertentu pada database. Log Audit trail yang bagus, bila diurutkan berdasarkan waktu bisa membentuk suatu kronologis manipulasi data. Ada tiga pilihan *value* pada *audit trail* yaitu *DB, OS* dan *NONE*. Berikut ini cara atau contoh aktifasi audit trail dengan nilai *value=DB*. Yang dapat dilihat pada gambar II.1.



Gambar II.1 aktifasi audit trail

Dari gambar tersebut terlihat bahwa audit_trail diset db, extended itu berarti hasil audit trail akan disimpan pada tabel yaitu pada tabel *aud$*. Dan *audit_sys_operations* diset false, berarti untuk user administrator tidak akan diaudit. Informasi yang dapat ditampilkan dari *audit_trail* ini dapat dilihat dari *direktory view* yang telah disediakan secara default oleh DBMS Oracle yaitu pada *view dba_audit_trail*. Berikut informasi yang dapat ditampilkan dari *view dba_audit_trail* yang dapat dilihat pada gambar II.2.



```
SQL> desc dba_audit_trail;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 OS_USERNAME                                        VARCHAR2(255)
 USERNAME                                           VARCHAR2(30)
 USERHOST                                           VARCHAR2(128)
 TERMINAL                                           VARCHAR2(255)
 TIMESTAMP                                          DATE
 OWNER                                              VARCHAR2(30)
 OBJ_NAME                                           VARCHAR2(128)
 ACTION                                    NOT NULL NUMBER
 ACTION_NAME                                        VARCHAR2(28)
 NEW_OWNER                                          VARCHAR2(30)
 NEW_NAME                                           VARCHAR2(128)
 OBJ_PRIVILEGE                                      VARCHAR2(16)
 SYS_PRIVILEGE                                      VARCHAR2(40)
 ADMIN_OPTION                                       VARCHAR2(1)
 GRANTEE                                            VARCHAR2(30)
 AUDIT_OPTION                                       VARCHAR2(40)
 SES_ACTIONS                                        VARCHAR2(19)
 LOGOFF_TIME                                        DATE
 LOGOFF_LREAD                                       NUMBER
 LOGOFF_PREAD                                       NUMBER
 LOGOFF_LWRITE                                      NUMBER
 LOGOFF_DLOCK                                       VARCHAR2(40)
 COMMENT_TEXT                                       VARCHAR2(4000)
 SESSIONID                                 NOT NULL NUMBER
 ENTRYID                                   NOT NULL NUMBER
 STATEMENTID                               NOT NULL NUMBER
 RETURNCODE                                NOT NULL NUMBER
 PRIV_USED                                          VARCHAR2(40)
 CLIENT_ID                                          VARCHAR2(64)
 ECONTEXT_ID                                        VARCHAR2(64)
 SESSION_CPU                                        NUMBER
 EXTENDED_TIMESTAMP                                 TIMESTAMP(6) WITH TIME ZONE
 PROXY_SESSIONID                                    NUMBER
 GLOBAL_UID                                         VARCHAR2(32)
 INSTANCE_NUMBER                                    NUMBER
 OS_PROCESS                                         VARCHAR2(16)
 TRANSACTIONID                                      RAW(8)
 SCN                                                NUMBER
 SQL_BIND                                           NVARCHAR2(2000)
 SQL_TEXT                                           NVARCHAR2(2000)
```

Gambar II.2 Field *view dba_audit_trail*

2. *Value Base Auditing*

   Jika audit trail mencatat segala perintah yang dilakukan user terhadap object database. Dan jenis object ini mencatat perubahan yang terjadi pada suatu objek database berdasarkan isi dari database tersebut.

   Terdapat 3 jenis audit yaitu:

   - Insert data yaitu mencatat data-data yang telah di insert oleh user ke tabel yang telah disediakan.
   - Update data yaitu mencatat data-data yang telah di delete oleh user ke tabel yang telah disediakan.
   - Delete data yaitu mencatat data-data yang telah di delete oleh user ke tabel yang telah disediakan.

3. *Fine Grained Auditing*

Fitur ini digunakan untuk mengawasi data yang diakses berdasarkan isi yang disimpan dalam tabel *fga_log$*. Dalam tabel *fga_log$* tersebut telah disediakan *direktory view* yang dapat digunakan untuk melihat isi tabel *fga_log$*. Direktory tersebut adalah *view dba_fga_audit_trail*. FGA sendiri dapat dihubungkan ke tabel atau *view* dan menghubungkannya dengan paket *DBMS_FGA*. Terdapat 4 paket atau dalam dbms fga yang dapat dilihat pada tabel II.2

Tabel II.2 Paket DBMS FGA

Nama Paket	**Keterangan**
ADD_POLICY	Membuat aturan audit dengan menggunakan keterangan sebagai kondisi audit
DROP_POLICY	Menghapus audit
ENABLE_ POLICY	Mengaktifkan policy
DISABLE_POLICY	Mematikan policy

Dari keempat paket tersebut. Paket ADD_POLICY memiliki beberapa parameter yang akan digunakan dalam menbuat aturan audit. Parameter paket tersebut dapat dilihat pada table II.3

Tabel II.3 Parameter paket ADD_POLICY

Parameter	**Keterangan**
Object_schema	Skema dari obyek yang akan diaudit. Nilai default: NULL.
Object_name	Nama dari object yang akan di audit
Policy_name	Nama dari aturan
Audit_condition	Sebuah kondisi di baris yang menunjukkan kondisi pemantauan. NULL diperbolehkan dan bertindak sebagai TRUE. Nilai default: NULL
Audit_column	Daftar kolom yang akan diaudit
Enable	Untuk mengaktifkan aturan, default : TRUE
Audit_trail	Untuk mengisi TEKS SQL dan LSQLBIND di fga_log $. Nilai default: DB_EXTENDED

FGA dapat mencatat dan menunjukkan operasi yang dilakukan dan operasi mencoba yang dilakukan oleh user sukses atau tidak. Hal ini bisa menjadi overhead untuk administrator. Berikut ini informasi yang dapat didapat dari direktory view dba_fga_audit_trail yang dapat dilihat pada gambar II.3

```
SQL> desc dba_fga_audit_trail;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SESSION_ID                                NOT NULL NUMBER
 TIMESTAMP                                          DATE
 DB_USER                                            VARCHAR2(30)
 OS_USER                                            VARCHAR2(255)
 USERHOST                                           VARCHAR2(128)
 CLIENT_ID                                          VARCHAR2(64)
 ECONTEXT_ID                                        VARCHAR2(64)
 EXT_NAME                                           VARCHAR2(4000)
 OBJECT_SCHEMA                                      VARCHAR2(30)
 OBJECT_NAME                                        VARCHAR2(128)
 POLICY_NAME                                        VARCHAR2(30)
 SCN                                                NUMBER
 SQL_TEXT                                           NVARCHAR2(2000)
 SQL_BIND                                           NVARCHAR2(2000)
 COMMENT$TEXT                                       VARCHAR2(4000)
 STATEMENT_TYPE                                     VARCHAR2(7)
 EXTENDED_TIMESTAMP                                 TIMESTAMP(6) WITH TIME ZONE
 PROXY_SESSIONID                                    NUMBER
 GLOBAL_UID                                         VARCHAR2(32)
 INSTANCE_NUMBER                                    NUMBER
 OS_PROCESS                                         VARCHAR2(16)
 TRANSACTIONID                                      RAW(8)
 STATEMENTID                                        NUMBER
 ENTRYID                                            NUMBER
```

Gambar II.3 Field view dba_fga_audit_trail

II.3.3 *Trigger* DBMS Oracle 10g

Trigger adalah blok PL/SQL yang disimpan dalam database dan akan diaktivasi ketika kita melakukan statement-statement SQL (DELETE, UPDATE, dan INSERT) pada sebuah tabel. Aktivasi trigger didasarkan pada event yang terjadi di dalam tabel tersebut sehingga trigger dapat membantu dalam menjaga integritas dan konsistensi data. Oracle 10g telah menyediakan statement CREATE TRIGGER untuk membuat sebuah trigger yang selanjutnya akan diaktivasi berdasarkan event tertentu. Secara umum, event trigger terbagi menjadi dua, yaitu BEFORE (sebelum) dan AFTER (setelah). Event tersebut menandakan kapan trigger akan diaktivasi, apakah sebelum ataukah sesudah proses yang dilakukan di dalam tabel bersangkutan.

Model-model yang digunakan untuk menspesifikasi aturan basis data aktif adalah model ECA *(Event-Condition-Action)***.** Aturan dalam model ECA memiliki tiga komponen yaitu:

1. *Event*

   Event yang memicu suatu *rule* tersebut biasanya berupa operasi perubahan basis data yang secara eksplisit ditambahkan ke basis data. Namun pada umumnya, bisa berupa *event temporal* atau jenis event eksternal yang lain.

2. *Condition*

   Condition menentukan apakah suatu *rule* dijalankan atau tidak. Ketika suatu trigger dijalankan, maka bagian condition (bersifat optional) akan dievaluasi jika didefinisikan oleh yang membuat trigger. Jika evaluasi bagian condition bernilai TRUE maka aksi suatu *rule* dijalankan.

3. *Action*

   Action biasanya berupa statement sql.

Sintaks umum penciptaan trigger pada DBMS Oracle 10g adalah:

CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER} triggering_event
[referencing_clause]
[WHEN trigger_condition]
[FOR EACH ROW]
Trigger_body;

Dimana *trigger_name* adalah nama trigger, *triggering_event* menspesifikasikan event yang *firing* (menyalakan) trigger, dan *trigger_body* adalah kode utama untuk trigger. *Referencing_clause* digunakan untuk menunjuk pada data dalam baris yang saat ini sedang dimodifikasi dengan nama yang berbeda. Jika ada *trigger_condition* dalam klausa WHEN, pertama akan dievaluasi dan kemudian *trigger_body* dieksekusi hanya jika hasil evaluasi bernilai TRUE.[8]

[8] http://mudafiq.student.umm.ac.id/files/2010/05/mudafiq-trigger.pdf

II.4 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.

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 5 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.[9]

II.4.1 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 sudah tersedia di berbagai platform, termasuk Windows. Di seri 3.23 MySQL

[9] http://id.wikipedia.org/wiki/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*.[10]

II.4.2 *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.

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**.

[10] http://ramdhana.wordpress.com/2009/03/12/sejarah-mysql/

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].[11]

```
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
```

[11] http://dev.mysql.com/doc/mysql-security-excerpt/5.0/en/index.html

- 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 II.4

```
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 II.4 general_log

II.4.3 *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:

- Melakukan *update* data otomatis jika terjadi perubahan. Contohnya adalah dalam sistem penjualan, jika dientri barang baru maka stock akan bertambah secara otomatis.

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

Berikut ini contoh trigger yang akan mencatat aktivitas ke tabel **log** setiap terjadi proses insert ke tabel pelanggan.[12]

```
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 ;
```

II.5 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.[13]

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

[12] http://achmatim.net/2010/02/24/mengenal-trigger-di-mysql/
[13] http://id.wikipedia.org/wiki/PostgreSQL

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 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.

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

- 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.

- 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.

- 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.

- 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.

- 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.

II.5.1 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 di mana 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.

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 Bruce Momjian, 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.

II.5.2 Fitur DBMS PostgrSQL

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

- 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.

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

- 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.

- 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.

- Menyediakan tipe tambahan seperti konstruksi geometris (titik, garis dan semacamnya), pengalamatan jaringan TCP/IP, ID ethernet card, ISBN/ISSN dan lain-lain.
- Tipe baru bisa didefinisikan dengan fungsi dan operator yang diperlukan
- PostgreSQL mendukung penyimpanan binary large object, termasuk gambar, suara atau video. Objek ini bisa diambil sebagian atau seluruhnya oleh aplikasi klien.
- Dukungan pada international character set, multibyte character encodings dan unicode.
- Perhatian pada sorting, case-sensitivity, dan formatting.[14]

II.5.3 *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*

[14] http://krismar05.wordpress.com/postgresql-2/

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 II.4

Tabel II.4 Parameter Log line prefix

Parameter	Keterangan
%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*

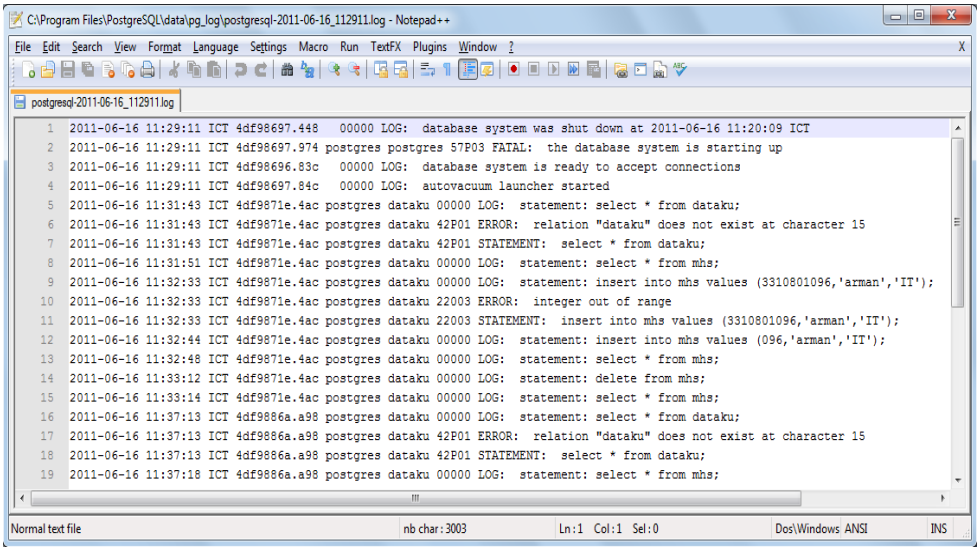
   Digunakan untuk mensetting statement apa saja yang akan di audit terdapat 3 pilihan value yang dapat dilihat pada tabel II.5.

Tabel II.5 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 II.5.[15]

[15] http://www.postgresql.org/docs/9.0/static/runtime-config-logging.html



Gambar II.5 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/postgresq
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 II.6 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.[16]

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

II.5.4 *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*.

[16] http://conocimientolibre.wordpress.com/2008/04/21/ten-new-features-that-make-postgresql-83-a-must-have/

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 tabel 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).[17]

[17] http://rojulman.web.id/index.php?pg=15&dt=1&dts=4

Bab III Analisis

III.1 Audit DBMS Oracle 10g

Berdasar audit yang bisa dilakukan di DBMS Oracle 10g yang telah dijelaskan pada landasan teori maka audit yang dapat diimplemetasikan adalah sebagai berikut:

III.1.1 *Audit Trail*

Berikut ini audit yang dapat dilakukan dengan menggunakan fasilitas audit trail:

1. Audit Session
   - Audit pengguna yang melakukan login berhasil ke database.

     Audit ini dapat dilakukan dengan DBMS Oracle 10g dengan menggunakan fitur audit session yang hasil nya dapat dilihat pada *view directory dba_audit_session*. Untuk menggunakan audit ini diperlukan pengaturan audit sebagai berikut.

     SQL> **audit session by access whenever successfull;**

     Contoh hasil audit pengguna yang melakukan login berhasil ke database dapat dilihat pada gambar III.1.

```
SQL> select username,extended_timestamp,action_name,returncode
  2  from dba_audit_session
  3  where action_name='LOGON' and returncode=0;

USERNAME   EXTENDED_TIMESTAMP                  ACTION_NAM RETURNCODE
---------- --------------------------------- ---------- ----------
ARMAN      07-08-2011 23.28.55,056000 +07:00 LOGON               0
IWAN       07-08-2011 23.29.06,272000 +07:00 LOGON               0
ADMIN_PEL  07-08-2011 23.29.28,221000 +07:00 LOGON               0
ADMIN_BAR  07-08-2011 23.30.03,555000 +07:00 LOGON               0
```

Gambar III.1 audit login berhasil

   - Audit pengguna yang melakukan login gagal ke database.

     Audit ini dapat dilakukan dengan DBMS Oracle 10g dengan menggunakan fitur audit session yang hasil nya dapat dilihat pada *view directory dba_audit_session*. Untuk menggunakan audit ini diperlukan pengaturan audit sebagai berikut.

SQL> **audit session by access whenever not successfull;**

Contoh hasil audit pengguna yang melakukan login gagal ke database dapat dilihat pada gambar III.2.

```
SQL> select username,extended_timestamp,action_name,returncode
  2  from dba_audit_session
  3  where action_name='LOGON' and returncode !=0;

USERNAME   EXTENDED_TIMESTAMP                  ACTION_NAM RETURNCODE
---------- ---------------------------------- ---------- ----------
ADMIN_PEL  07-08-2011 23.18.24,753000 +07:00 LOGON            1017
IWAN       07-08-2011 23.18.36,437000 +07:00 LOGON            1045
ADMIN_BAR  07-08-2011 23.20.42,860000 +07:00 LOGON            1017
ARMAN      07-08-2011 23.20.48,772000 +07:00 LOGON            1017
```

Gambar III.2 audit login gagal

Catatan : Returncode merupakan kode yang ada di DBMS Oracle 10g yang berarti jika Nilai Returncode=0 maka user tesebut berhasil login ke database. Dan jika Returncode !=0 maka user tesebut gagal masuk/ditolak masuk ke database.

- Audit pengguna yang melakukan logout dari database.

  Audit ini dapat dilakukan dengan DBMS Oracle 10g dengan menggunakan fitur audit session yang hasil nya dapat dilihat pada *view directory dba_audit_session*. Untuk menggunakan audit ini diperlukan pengaturan audit sebagai berikut.

  SQL> **audit session by access;**

  Contoh hasil audit pengguna yang melakukan logout dari database dapat dilihat pada gambar III.3.

```
SQL> select username,extended_timestamp,action_name,returncode
  2  from dba_audit_session
  3  where action_name='LOGOFF';

USERNAME   EXTENDED_TIMESTAMP                  ACTION_NAM RETURNCODE
---------- ---------------------------------- ---------- ----------
ADMIN_PEL  07-08-2011 23.10.57,157000 +07:00 LOGOFF             0
ADMIN_BAR  07-08-2011 23.15.45,336000 +07:00 LOGOFF             0
IWAN       07-08-2011 23.19.04,689000 +07:00 LOGOFF             0
ARMAN      07-08-2011 23.20.53,343000 +07:00 LOGOFF             0
ARMAN      07-08-2011 23.28.55,056000 +07:00 LOGOFF             0
IWAN       07-08-2011 23.29.06,272000 +07:00 LOGOFF             0
ADMIN_PEL  07-08-2011 23.29.28,221000 +07:00 LOGOFF             0
ADMIN_BAR  07-08-2011 23.30.03,555000 +07:00 LOGOFF             0

8 rows selected.
```

Gambar III.3 audit pengguna melakukan logout

- Audit pengguna yang melakukan login dan logout pada database baik yang berhasil maupun yang gagal.

  Audit ini dapat dilakukan dengan DBMS Oracle 10g dengan menggunakan fitur audit session yang hasil nya dapat dilihat pada *view directory dba_audit_session*. Untuk menggunakan audit ini diperlukan pengaturan audit sebagai berikut.

  SQL> **audit session by access whenever successfull;**

  SQL> **audit session by access whenever not successfull;**

  Contoh hasil audit pengguna yang melakukan login dan logout dari database baik yang berhasil maupun yang gagal dapat dilihat pada gambar III.4.

```
SQL> select username,extended_timestamp,action_name,returncode
  2  from dba_audit_session;

USERNAME   EXTENDED_TIMESTAMP                  ACTION_NAM RETURNCODE
---------- ---------------------------------- ---------- ----------
ADMIN_BAR  07-08-2011 23.30.03,555000 +07:00 LOGOFF              0
ADMIN_PEL  07-08-2011 23.29.28,221000 +07:00 LOGOFF              0
ARMAN      07-08-2011 23.20.53,343000 +07:00 LOGOFF              0
IWAN       07-08-2011 23.19.04,689000 +07:00 LOGOFF              0
ADMIN_BAR  07-08-2011 23.15.45,336000 +07:00 LOGOFF              0
ADMIN_PEL  07-08-2011 23.10.57,157000 +07:00 LOGOFF              0
IWAN       07-08-2011 23.29.06,272000 +07:00 LOGON               0
ARMAN      07-08-2011 23.28.55,056000 +07:00 LOGON               0
ARMAN      07-08-2011 23.20.48,772000 +07:00 LOGON            1017
ADMIN_BAR  07-08-2011 23.20.42,860000 +07:00 LOGON            1017
IWAN       07-08-2011 23.18.36,437000 +07:00 LOGON            1045

USERNAME   EXTENDED_TIMESTAMP                  ACTION_NAM RETURNCODE
---------- ---------------------------------- ---------- ----------
ADMIN_PEL  07-08-2011 23.18.24,753000 +07:00 LOGON            1017

12 rows selected.
```

Gambar III.4 audit pengguna yang login dan logout

- Audit pengguna tertentu yang melakukan login dan logout dari database.

  Audit ini dapat dilakukan dengan DBMS Oracle 10g dengan menggunakan fitur audit session yang hasil nya dapat dilihat pada *view directory dba_audit_session*. Untuk menggunakan audit ini diperlukan pengaturan audit sebagai berikut.

  SQL> **audit session by arman by access whenever successfull;**

  SQL> **audit session by arman by access whenever not successfull;**

Contoh hasil audit pengguna tertentu yang melakukan login dan logout dari database dapat dilihat pada gambar III.5.

```
SQL> select username,extended_timestamp,action_name,returncode
  2  from dba_audit_session
  3  where username='ARMAN';

USERNAME   EXTENDED_TIMESTAMP                ACTION_NAM RETURNCODE
---------- ------------------------------- ---------- ----------
ARMAN      07-08-2011 23.20.48,772000 +07:00 LOGON            1017
ARMAN      07-08-2011 23.20.53,343000 +07:00 LOGOFF              0
ARMAN      07-08-2011 23.28.55,056000 +07:00 LOGOFF              0
```

Gambar III.5 audit pengguna tertentu yang login dan logout

- Audit pengguna login pada waktu tertentu.

  Audit ini dapat dilakukan dengan DBMS Oracle 10g dengan menggunakan fitur audit session yang hasil nya dapat dilihat pada *view directory dba_audit_session*. Untuk menggunakan audit ini diperlukan pengaturan audit sebagai berikut.

  SQL> **audit session by access whenever successfull;**

  SQL> **audit session by access whenever not successfull;**

  Contoh hasil audit pengguna tertentu yang melakukan login dapat dilihat pada gambar III.6.

```
SQL> select username,extended_timestamp,action_name,returncode
  2  from dba_audit_session
  3  where extended_timestamp='07-08-2011 23.30.03,555000 +07:00';

USERNAME   EXTENDED_TIMESTAMP                ACTION_NAM RETURNCODE
---------- ------------------------------- ---------- ----------
ADMIN_BAR  07-08-2011 23.30.03,555000 +07:00 LOGOFF              0
```

Gambar III.6 audit pengguna tertentu yang login

2. Audit statement pengguna dan object database
   - Audit statement pengguna yang melakukan perubahan pada object database (statement DDL).

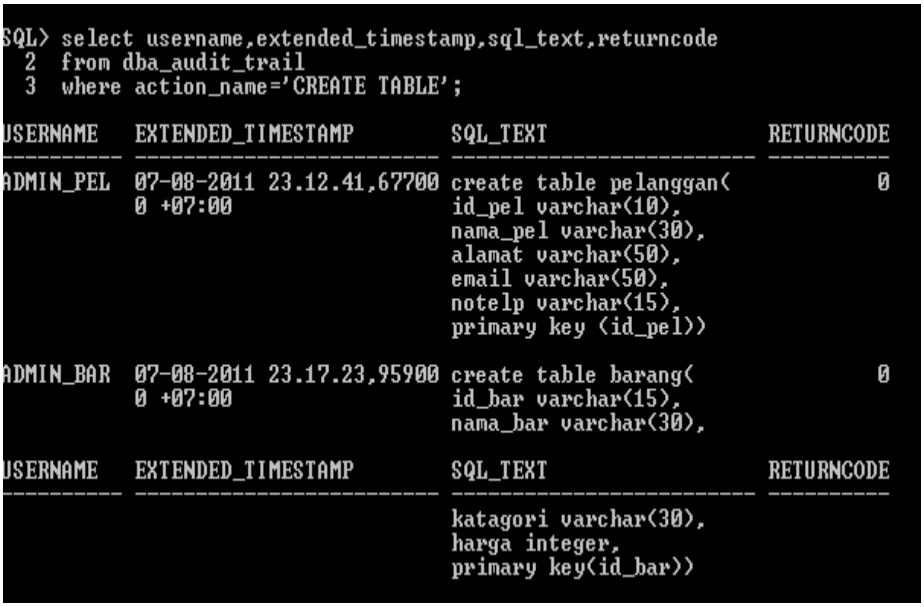
     Audit ini dapat dilakukan dengan DBMS Oracle 10g dengan menggunakan fitur audit trail yang hasil nya dapat dilihat pada *view directory dba_audit_trail*. Audit bertujuan untuk mecatat perintah-perintah yang dapat mempengaruhi atau menambah object dari sebuah database misalkan penambahan sebuah tabel atau menambah *field* (kolom) dari suatu tabel yang

ada, menghapus suatu tabel, membuat suatu trigger dan lain sebagainya. Untuk menggunakan audit ini diperlukan pengaturan audit sebagai berikut.

SQL> **audit all privileges by access whenever successfull;**

SQL> **audit all privileges by access whenever not successfull;**

Contoh hasil audit pengguna yang melakukan perubahan pada object database dapat dilihat pada gambar III.7.



Gambar III.7 audit pengguna melakukan perubahan object

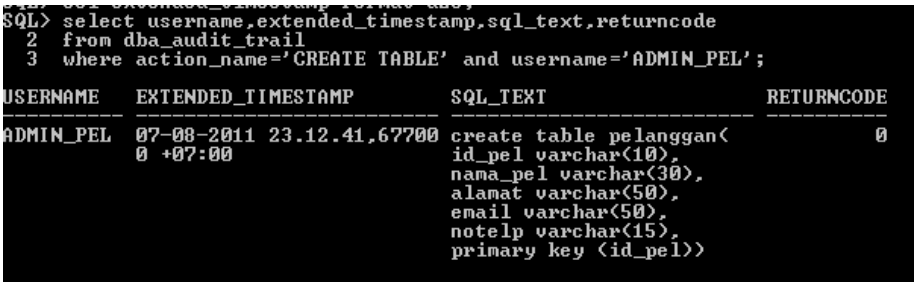
- Audit statement pengguna tertentu yang melakukan perubahan pada object database (statement DDL).

  Audit ini dapat dilakukan dengan DBMS Oracle 10g dengan menggunakan fitur audit trail yang hasil nya dapat dilihat pada *view directory dba_audit_trail*. Audit bertujuan untuk mecatat perintah-perintah yang dapat mempengaruhi atau menambah object dari sebuah database misalkan penambahan sebuah tabel atau menambah *field* (kolom) dari suatu tabel yang ada, menghapus suatu tabel, membuat suatu trigger dan lain sebagainya oleh pengguna tertentu. Untuk menggunakan audit ini diperlukan pengaturan audit sebagai berikut.

  SQL> **audit all privileges by admin_pel by access whenever successfull;**

SQL> **audit all privileges by admin_pel by access whenever not successfull;**

Contoh hasil audit pengguna tertentu yang melakukan perubahan pada object database dapat dilihat pada gambar III.8.



```
SQL> select username,extended_timestamp,sql_text,returncode
  2  from dba_audit_trail
  3  where action_name='CREATE TABLE' and username='ADMIN_PEL';

USERNAME   EXTENDED_TIMESTAMP          SQL_TEXT                    RETURNCODE
---------- -------------------------- -------------------------- ----------
ADMIN_PEL  07-08-2011 23.12.41,67700  create table pelanggan(             0
           0 +07:00                   id_pel varchar(10),
                                      nama_pel varchar(30),
                                      alamat varchar(50),
                                      email varchar(50),
                                      notelp varchar(15),
                                      primary key (id_pel))
```

Gambar III.8 audit pengguna tertentu yang melakukan perubahan object

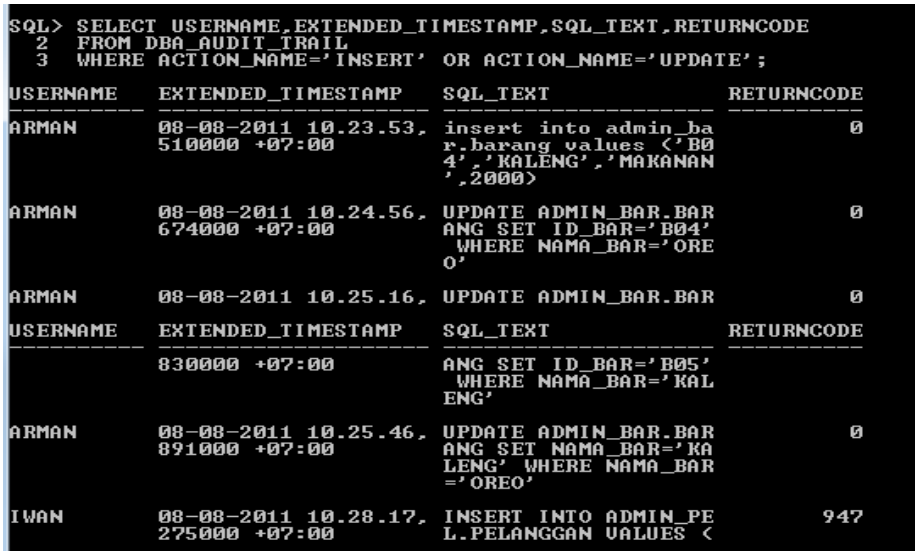
- Audit statement pengguna yang melakukan perubahan isi dari suatu object database (statement DML).

  Audit ini dapat dilakukan dengan DBMS Oracle 10g dengan menggunakan fitur audit trail yang hasil nya dapat dilihat pada *view directory dba_audit_trail*. Audit bertujuan untuk mecatat perintah-perintah yang memanipulasi isi dari object dari sebuah database misalkan penambahan isi suatu tabel, perubahan isi dari suatu tabel, penghapusan isi dari suatu tabel dan ada nya usaha untuk melihat isi dari suatu tabel. Untuk menggunakan audit ini diperlukan pengaturan audit sebagai berikut.

  SQL> **audit insert table,update table,delete table,select table by access whenever successfull;**

  SQL> **audit insert table,update table,delete table,select table by access whenever not successfull;**

  Contoh hasil audit pengguna yang melakukan perubahan isi dari suatu object database dapat dilihat pada gambar III.9.



```
SQL> SELECT USERNAME,EXTENDED_TIMESTAMP,SQL_TEXT,RETURNCODE
  2  FROM DBA_AUDIT_TRAIL
  3  WHERE ACTION_NAME='INSERT' OR ACTION_NAME='UPDATE';

USERNAME   EXTENDED_TIMESTAMP   SQL_TEXT              RETURNCODE
---------- ------------------- --------------------- ----------
ARMAN      08-08-2011 10.23.53, insert into admin_ba          0
           510000 +07:00        r.barang values ('B0
                                4','KALENG','MAKANAN
                                ',2000)

ARMAN      08-08-2011 10.24.56, UPDATE ADMIN_BAR.BAR          0
           674000 +07:00        ANG SET ID_BAR='B04'
                                 WHERE NAMA_BAR='ORE
                                O'

ARMAN      08-08-2011 10.25.16, UPDATE ADMIN_BAR.BAR          0

USERNAME   EXTENDED_TIMESTAMP   SQL_TEXT              RETURNCODE
---------- ------------------- --------------------- ----------
           830000 +07:00        ANG SET ID_BAR='B05'
                                 WHERE NAMA_BAR='KAL
                                ENG'

ARMAN      08-08-2011 10.25.46, UPDATE ADMIN_BAR.BAR          0
           891000 +07:00        ANG SET NAMA_BAR='KA
                                LENG' WHERE NAMA_BAR
                                ='OREO'

IWAN       08-08-2011 10.28.17, INSERT INTO ADMIN_PE        947
           275000 +07:00        L.PELANGGAN VALUES (
```

Gambar III.9 audit pengguna yang melakukan perubahan isi object

- Audit statement pengguna tertentu yang melakukan perubahan isi dari suatu object database (statement DML).

  Audit ini dapat dilakukan dengan DBMS Oracle 10g dengan menggunakan fitur audit trail yang hasil nya dapat dilihat pada *view directory dba_audit_trail*. Audit bertujuan untuk mecatat perintah-perintah yang memanipulasi isi dari object dari sebuah database misalkan penambahan isi suatu tabel, perubahan isi dari suatu tabel, penghapusan isi dari suatu tabel dan adanya usaha untuk melihat isi dari suatu tabel oleh pengguna tertentu.

  Untuk menggunakan audit ini diperlukan pengaturan audit sebagai berikut.

  SQL> **audit insert table,update table,delete table,select table by iwan by access whenever successfull;**

  SQL> **audit insert table,update table,delete table,select table by iwan by access whenever not successfull;**

  Contoh hasil audit pengguna tertentu yang melakukan perubahan isi dari suatu object database dapat dilihat pada gambar III.10.

```
SQL> SELECT USERNAME,EXTENDED_TIMESTAMP,SQL_TEXT,RETURNCODE
  2  FROM DBA_AUDIT_TRAIL
  3  WHERE USERNAME='IWAN' AND ACTION_NAME='INSERT';

USERNAME   EXTENDED_TIMESTAMP   SQL_TEXT             RETURNCODE
---------- -------------------- -------------------- ----------
IWAN       08-08-2011 10.28.17, INSERT INTO ADMIN_PE        947
           275000 +07:00        L.PELANGGAN VALUES (
                                'P04','BOY','BINTAN'
                                )

IWAN       08-08-2011 10.28.36, INSERT INTO ADMIN_PE          0
           666000 +07:00        L.PELANGGAN VALUES (
                                'P04','BOY','BINTAN'
                                ,'GMAIL','2312')
```

Gambar III.10 audit pengguna tertentu melakukan perubahan isi object

- Audit Pengguna yang melakukan perintah pengendalian pengaksesan data (statement DCL).

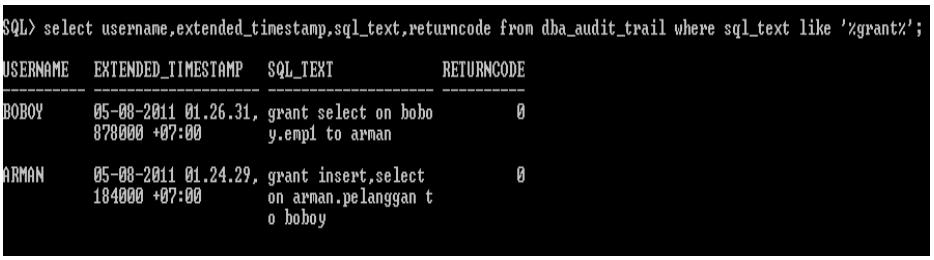
  Audit ini dapat dilakukan dengan DBMS Oracle 10g dengan menggunakan fitur audit trail yang hasil nya dapat dilihat pada *view directory dba_audit_trail*. Audit bertujuan untuk mecatat perintah-perintah pengendalian pengaksesan data oleh suatu pengguna terhadap pengguna yang lain seperti pemberian hak akses untuk memanipulasi data pada tabel yang

dibuat suatu user ke user yang lain atau sebaliknya dan mencabut hak akses yang telah diberikan. Untuk menggunakan audit ini diperlukan pengaturan audit sebagai berikut.

SQL> **audit grant any object by access whenever successfull;**

SQL> **audit grant any object by access whenever not successfull;**

Contoh hasil audit pengguna yang melakukan perintah pengendalian pengaksesan data dapat dilihat pada gambar III.11.



```
SQL> select username,extended_timestamp,sql_text,returncode from dba_audit_trail where sql_text like '%grant%';

USERNAME   EXTENDED_TIMESTAMP   SQL_TEXT            RETURNCODE
---------- ------------------- ------------------- ----------
BOBOY      05-08-2011 01.26.31, grant select on bobo         0
           878000 +07:00        y.emp1 to arman

ARMAN      05-08-2011 01.24.29, grant insert,select          0
           184000 +07:00        on arman.pelanggan t
                                o boboy
```

Gambar III.11 audit pengguna melakukan pengendalian pengaksesan data

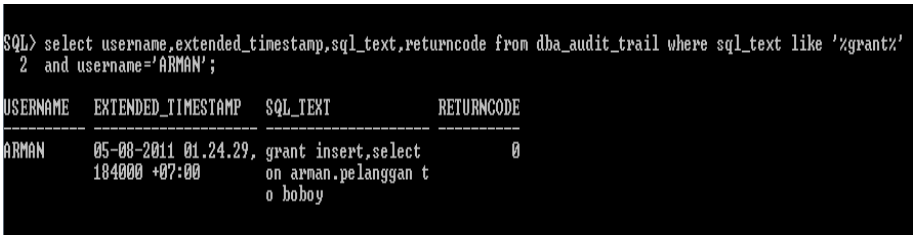
- Audit Pengguna tertentu yang melakukan perintah pengendalian pengaksesan data (statement DCL).

  Audit ini dapat dilakukan dengan DBMS Oracle 10g dengan menggunakan fitur audit trail yang hasil nya dapat dilihat pada *view directory dba_audit_trail*. Audit bertujuan untuk mecatat perintah-perintah pengendalian pengaksesan data oleh suatu pengguna terhadap pengguna yang lain seperti pemberian hak akses untuk memanipulasi data pada tabel yang dibuat suatu user ke user yang lain atau sebaliknya dan mencabut hak akses yang telah diberikan oleh pengguna tertentu. Untuk menggunakan audit ini diperlukan pengaturan audit sebagai berikut.

  SQL> **audit grant any object by arman by access whenever successfull;**

  SQL> **audit grant any object by arman by access whenever not successfull;**

  Contoh hasil audit pengguna tertentu yang melakukan perintah pengendalian pengaksesan data dapat dilihat pada gambar III.12.



Gambar III.12 audit pengguna tertentu melakukan perintah pengendalian pengaksesan data

3. Audit Object Database.

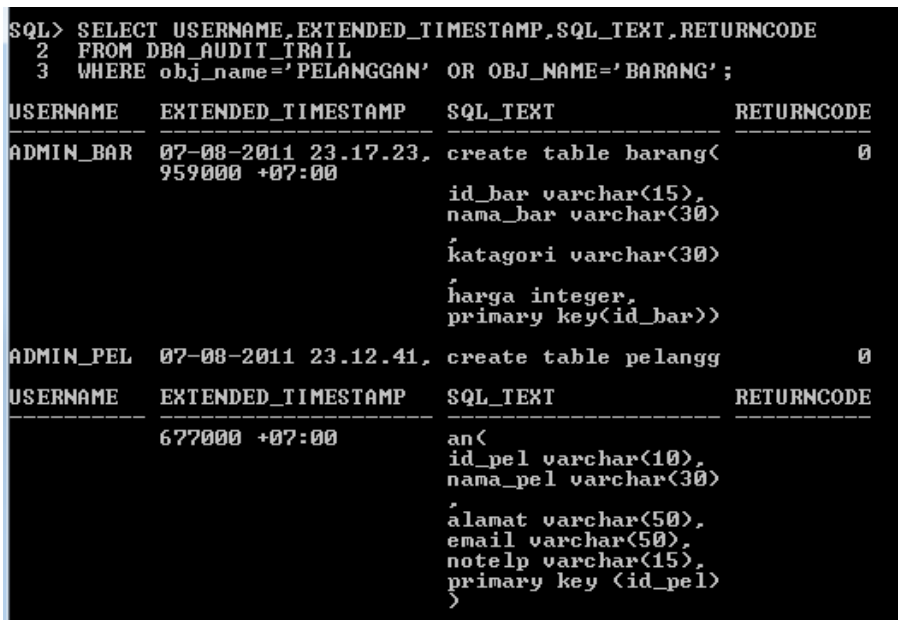
- Audit statement (perintah) yang terjadi pada object tertentu (DDL Statement).

  Audit ini bertujuan untuk mencatat statement atau query apa yang digunakan oleh pengguna database terhadap object database. Apakah terdapat object baru, apakah ada suatu object yang mengalami perubahan atau sebaliknya mengalami. Untuk menggunakan audit ini diperlukan pengaturan audit sebagai berikut.

  SQL> **audit all privilege by access whenever successfull;**

  SQL> **audit all privilege by access whenever not successfull;**

  Contoh hasil audit statement yang terjadi pada object tertentu dapat dilihat pada gambar III.13.



Gambar III.13 audit statement pada object tertentu

- Audit statement (perintah) yang terjadi pada isi dari object database (DML Statement).

  Audit ini bertujuan untuk mencatat statement atau query apa yang digunakan oleh pengguna database terhadap isi dari object database. Apakah menggunakan perintah penambahan, pembaharuan, penghapusan atau peritah penampilan data-data object tertentu. Untuk menggunakan audit ini diperlukan pengaturan audit sebagai berikut.

  SQL> **audit all privilege on pelanggan by access whenever successfull;**

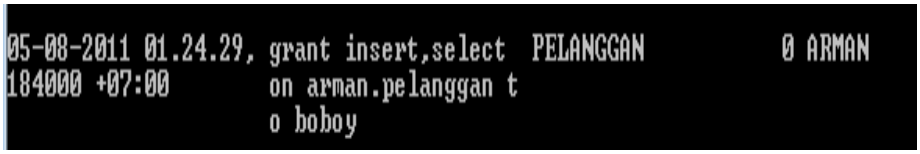
  SQL> **audit all privilege on barang by access whenever not successfull;**

Contoh hasil audit terjadi pada isi dari object database dapat dilihat pada gambar III.14.

Gambar III.14 audit isi object database

- Audit statement (perintah) perubahan pada pengendalian pengaksesan data pada object database (DCL Statement).

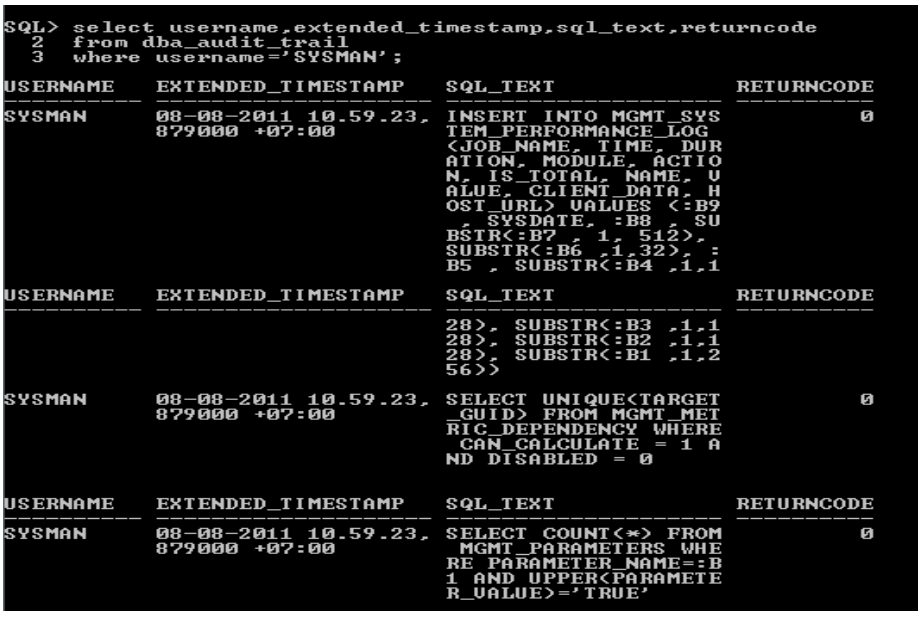
  Audit ini bertujuan untuk mencatat statement atau query apa yang digunakan oleh pengguna database terhadap isi dari object database untuk hak akses ke data tersebut. Contoh audit statement dapat dilihat pada gambar III.15.



Gambar III.15 audit statement

- Audit administrator

  Audit ini digunakan untuk mengaudit user administrator. Karena selain user bisa, user admin juga perlu diaudit. Tidaka menutup kemungkinan bahwa password dari user adminstrator tersebut diketahui oleh pihak yang tidak berwewenang. Hasil dari implementasi yang telah dilakukan dapat dilihat pada gambar III.16.



Gambar III.16 audit administrator

III.1.2 *Value Based Auditing*

Audit ini berfungsi untuk menampung informasi apa yang dimasukkan, dirubah dan di hapus oleh user. Untuk mengetahui history perubahan data dapat dilakukan dengan cara menyimpan data lama sebelum diupdate. Jenis audit yang seperti ini disebut sebagai *Value-Based Auditing (VBA)*. Untuk membuat jenis audit ini kita bisa menggunakan trigger. Sebelum membuat trigger, yang akan dilakukan adalah membuat tabel yang telah dirancang pada bagian rancangan tabel untuk menampung hasil trigger tersebut.

Berikut ini audit yang dapat dilakukan dengan menggunakan trigger audit:

- Audit Insert data

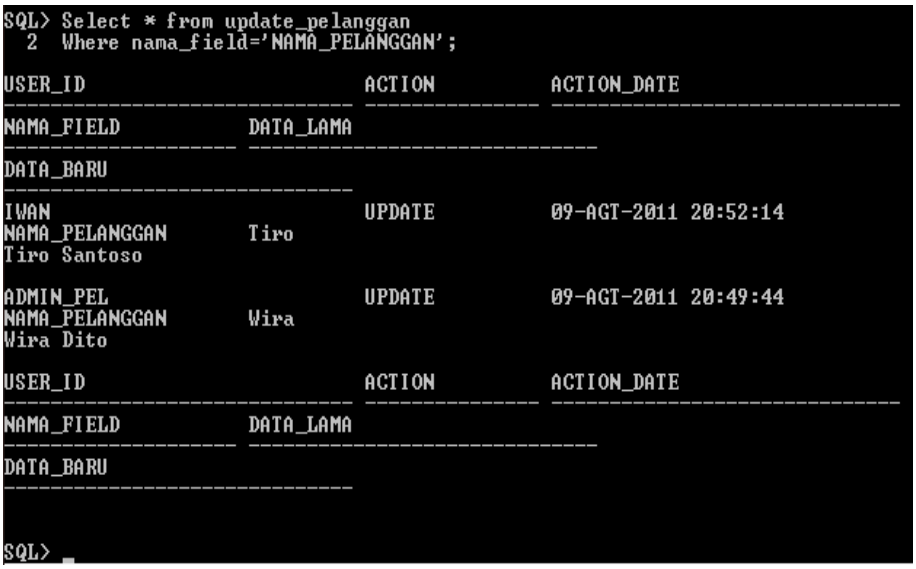
  Audit ini berfungsi untuk mencatat data-data masukan dari user dengan suatu tabel yang telah dibuat berdasarkan isi yang dimasukkan oleh user tesebut. Contoh hasil trigger audit insert data dapat dilihat pada gambar III.17.

```
SQL> select * from ins_del_pelanggan
  2   where action='INSERT';

USER_ID                        ACTION          ACTION_DATE
------------------------------ --------------- ------------------------------
ID_PELANGGAN NAMA_PELANGGAN                   EMAIL
------------ -------------------------------- ------------------------------
ALAMAT                         NOTELP
------------------------------ ---------------
IWAN                           INSERT          09-AGT-2011 20:42:52
           4 Mardianto                         Mardianto@gmail.com
Sei Panas                      85668457800

ADMIN_PEL                      INSERT          09-AGT-2011 20:39:25
           1 Wira                              Wira@yahoo.com
Duta Mas                       8566516377

USER_ID                        ACTION          ACTION_DATE
------------------------------ --------------- ------------------------------
ID_PELANGGAN NAMA_PELANGGAN                   EMAIL
------------ -------------------------------- ------------------------------
ALAMAT                         NOTELP
------------------------------ ---------------

ADMIN_PEL                      INSERT          09-AGT-2011 20:39:49
           2 Takim                             Takim@yahoo.com
Patam                          8568610105

IWAN                           INSERT          09-AGT-2011 20:42:25
           3 Tiro                              Wira@yahoo.com

USER_ID                        ACTION          ACTION_DATE
------------------------------ --------------- ------------------------------
ID_PELANGGAN NAMA_PELANGGAN                   EMAIL
------------ -------------------------------- ------------------------------
ALAMAT                         NOTELP
------------------------------ ---------------
Tanjung Uban                   85766459595

SQL>
```

Gambar III.17 audit insert data

- Audit Update data

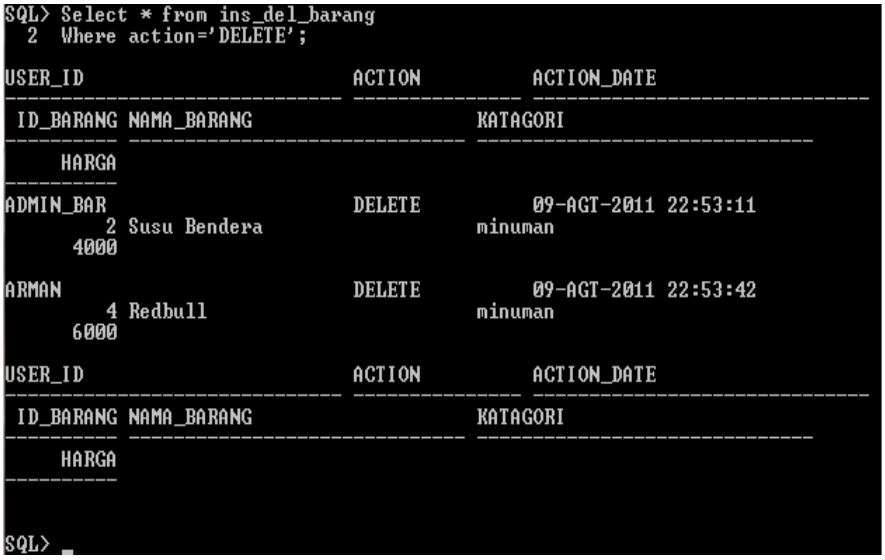
  Audit ini berfungsi untuk mencatat data-data perubahan data dari user pada suatu tabel yang telah dibuat berdasarkan isi yang dirubah oleh user tesebut. Contoh hasil trigger audit update data dapat dilihat pada gambar III.18.



Gambar III.18 audit update data

- Audit Delete data

  Audit ini berfungsi untuk mencatat data-data yang telah dihapus oleh user pada suatu tabel yang telah dibuat berdasarkan isi yang dihapus oleh user tesebut. Contoh hasil trigger audit delete data dapat dilihat pada gambar III.19.



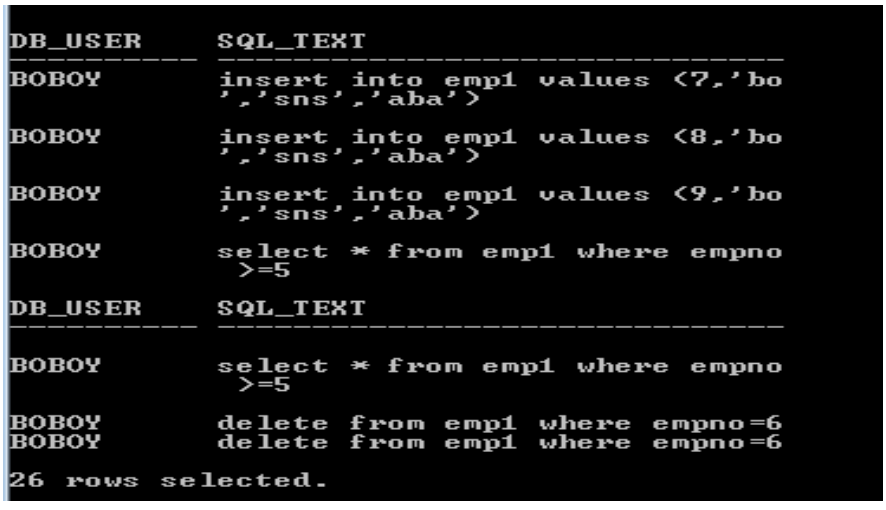
Gambar III.19 audit delete data

III.1.3 *Fine Grained Auditing*

Audit berdasarkan policy (peraturan) yang telah ditentukan *fine grained audit*. Audit ini digunakan untuk mengaudit suatu object yang lebih spesifik.

- Audit pernyataan DML kolom tertentu dengan kondisi NULL.

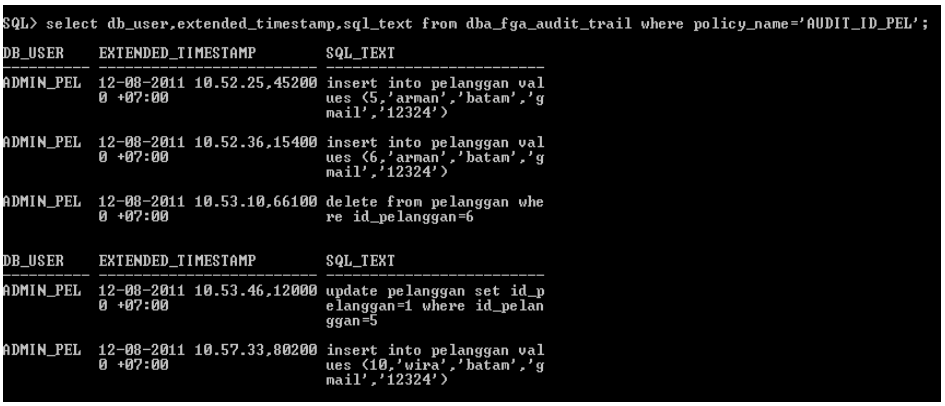
  Audit ini digunakan untuk mengaudit pernyataan tertentu pada suatu field pada suatu tabel dengan kondisi NULL. Jadi semua penyataan yang mengakses suatu field tersebut akan dicatat. Contoh hasil audit pernyataan DML kolom tertentu dapat dilihat pada gambar III.20.



Gambar III.20 audit DML kolom tertentu dengan kondisi NULL

- Audit pernyataan DML kolom tertentu dengan kondisi yang ditentukan.

  Audit ini digunakan untuk mengaudit pernyataan tertentu pada suatu field pada suatu tabel dengan kondisi yang ditentukan. Jadi hanya penyataan yang sesuai dengan ketentuan yang akan diaudit. Contoh audit dapat dilihat dilihat pada gambar III.21.



Gambar III.21 audit DML kolom tertentu dengan kondisi tertentu

III.2 Audit pada DBMS MySQL

Untuk melakukan audit pada DBMS MySQL dapat menggunakan versi MySQL 5.0 keatas karna versi MySQL ini sudah mempunyai fitur yang lengkap seperti *log file*, *trigger* dan *procedure* untuk melakukan audit. Seperti yang dijelaskan pada landasan teori yaitu DBMS MySQL tidak memilki fitur khusus yang digunakan untuk melakukan audit database. Akan tetapi MySQL memilki fitur yang bernama *file log* yang bisa digunakan untuk *database auditing*. Tidak semua dari *file log* tersebut dapat digunakan untuk *database auditing*. *File log* yang dapat digunakan adalah *general log* yang berisi segala *sintak* dan *session* yang dilakukan oleh pengguna database. Penyimpanan file log ada 2 yaitu *file log* disimpan di sebuah file text dan *file log* disimpan pada database.

Untuk penyimpanan *file log* di sebuah file lebih sederhana dan jumlah space yang digunakan akan lebih sedikit dari pada menggunakan tabel. Akan tetapi ketika file

tersebut disimpan kedalam sebuah file maka pekerjaan audit akan memerlukan waktu yang lama karna proses pencarian data dilakukan secara manual. Dibandingkan penyimpanan hasil audit kesebuah database akan mempercepat pencarian data dengan menggunakan *query*. Oleh karena itu pada tugas akhir ini akan dilakukan penyimpanan hasil audit ke database. Dengan tujuan agar mempermudah dalam pencarian data dan kebutuhan akan audit itu sendiri.

Sesuai dengan audit yang dapat dilakukan di DBMS Oracel diatas, berikut ini adalah audit Oracle 10g yang dapat di implemtasikan pada DBMS MySQL.

1. Audit Session
   - Audit pengguna yang melakukan login berhasil ke database.

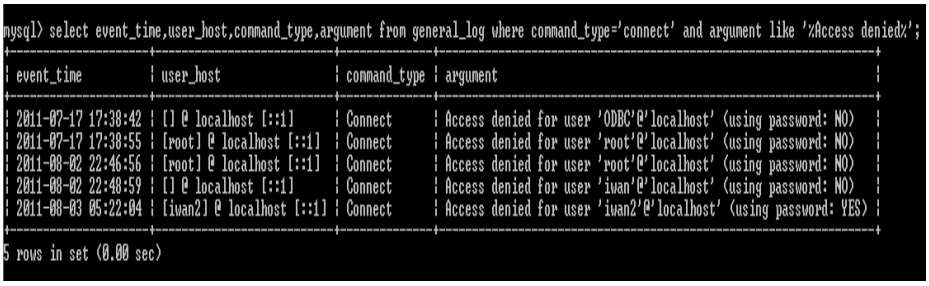
     Audit ini dapat dilakukan dengan DBMS MySQL dengan menggunakan log general yang hasil nya dapat dilihat pada database mysql yaitu pada tabel *general_log*. Contoh hasil audit pengguna yang melakukan login berhasil ke database dapat dilihat pada gambar III.22.

```
mysql> select event_time,user_host,command_type,argument from general_log where command_type='connect' and argument like '%on%';
+---------------------+---------------------------+--------------+-----------------------+
| event_time          | user_host                 | command_type | argument              |
+---------------------+---------------------------+--------------+-----------------------+
| 2011-07-17 17:07:19 | [root] @ localhost [::1]  | Connect      | root@localhost on     |
| 2011-07-17 17:21:29 | [root] @ localhost [::1]  | Connect      | root@localhost on     |
| 2011-07-17 17:38:42 | [ODBC] @ localhost [::1]  | Connect      | ODBC@localhost as  on |
| 2011-07-17 17:38:55 | [root] @ localhost [::1]  | Connect      | root@localhost on     |
| 2011-07-17 17:39:01 | [root] @ localhost [::1]  | Connect      | root@localhost on     |
| 2011-07-25 09:31:51 | [root] @ localhost [::1]  | Connect      | root@localhost on     |
| 2011-08-02 22:46:56 | [root] @ localhost [::1]  | Connect      | root@localhost on     |
| 2011-08-02 22:47:02 | [root] @ localhost [::1]  | Connect      | root@localhost on     |
| 2011-08-02 22:48:59 | [iwan] @ localhost [::1]  | Connect      | iwan@localhost as  on |
| 2011-08-02 22:49:06 | [iwan2] @ localhost [::1] | Connect      | iwan2@localhost on    |
| 2011-08-02 22:50:20 | [root] @ localhost [::1]  | Connect      | root@localhost on     |
| 2011-08-03 05:13:30 | [root] @ localhost [::1]  | Connect      | root@localhost on     |
| 2011-08-03 05:22:04 | [iwan2] @ localhost [::1] | Connect      | iwan2@localhost on    |
| 2011-08-03 05:22:08 | [iwan2] @ localhost [::1] | Connect      | iwan2@localhost on    |
| 2011-08-04 22:20:21 | [root] @ localhost [::1]  | Connect      | root@localhost on     |
+---------------------+---------------------------+--------------+-----------------------+
15 rows in set (0.00 sec)
```

Gambar III.22 audit login berhasil

   - Audit pengguna yang melakukan login gagal ke database.

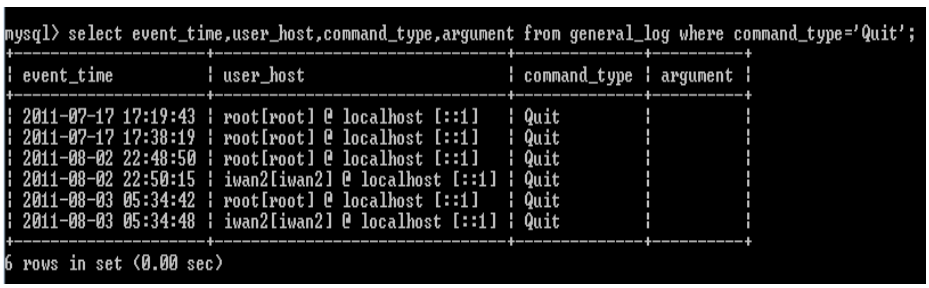
     Audit ini dapat dilakukan dengan DBMS MySQL dengan menggunakan log general yang hasil nya dapat dilihat pada database mysql yaitu pada tabel *general_log*. Contoh hasil audit pengguna yang melakukan login gagal ke database dapat dilihat pada gambar III.23.



Gambar III.23 audit login gagal

- Audit pengguna yang melakukan logout dari database.

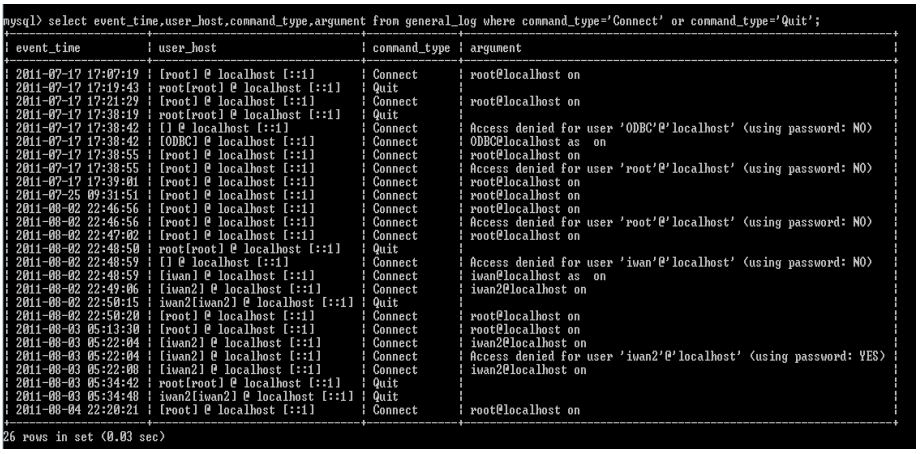
  Audit ini dapat dilakukan dengan DBMS MySQL dengan menggunakan log general yang hasil nya dapat dilihat pada database mysql yaitu pada tabel *general_log*. Contoh hasil audit pengguna yang melakukan logout dari database dapat dilihat pada gambar III.24.



Gambar III.24 audit logout

- Audit pengguna yang melakukan login dan logout pada database baik yang berhasil maupun yang gagal.

  Audit ini dapat dilakukan dengan DBMS MySQL dengan menggunakan log general yang hasil nya dapat dilihat pada database mysql yaitu pada tabel *general_log*. Contoh hasil audit pengguna yang melakukan login dan logout pada database baik yang berhasil maupun yang gagal dapat dilihat pada gambar III.25.

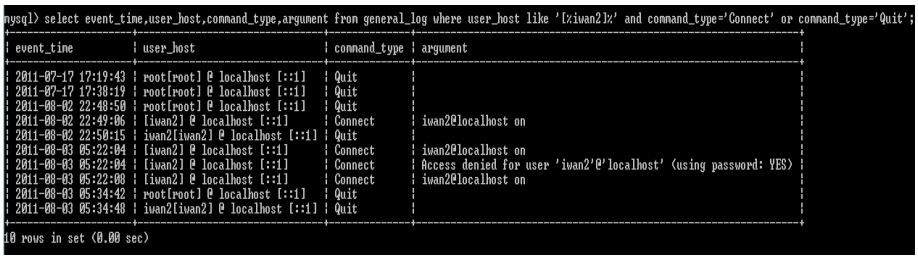


```
mysql> select event_time,user_host,command_type,argument from general_log where command_type='Connect' or command_type='Quit';
+---------------------+-------------------------------+--------------+-----------------------------------------------------------------+
| event_time          | user_host                     | command_type | argument                                                        |
+---------------------+-------------------------------+--------------+-----------------------------------------------------------------+
| 2011-07-17 17:07:19 | [root] @ localhost [::1]      | Connect      | root@localhost on                                               |
| 2011-07-17 17:19:43 | root[root] @ localhost [::1]  | Quit         |                                                                 |
| 2011-07-17 17:21:29 | [root] @ localhost [::1]      | Connect      | root@localhost on                                               |
| 2011-07-17 17:38:19 | root[root] @ localhost [::1]  | Quit         |                                                                 |
| 2011-07-17 17:38:42 | [] @ localhost [::1]          | Connect      | Access denied for user 'ODBC'@'localhost' (using password: NO)  |
| 2011-07-17 17:38:42 | [ODBC] @ localhost [::1]      | Connect      | ODBC@localhost as  on                                           |
| 2011-07-17 17:38:55 | [root] @ localhost [::1]      | Connect      | root@localhost on                                               |
| 2011-07-17 17:38:55 | [root] @ localhost [::1]      | Connect      | Access denied for user 'root'@'localhost' (using password: NO)  |
| 2011-07-17 17:39:01 | [root] @ localhost [::1]      | Connect      | root@localhost on                                               |
| 2011-07-25 09:31:51 | [root] @ localhost [::1]      | Connect      | root@localhost on                                               |
| 2011-08-02 22:46:56 | [root] @ localhost [::1]      | Connect      | root@localhost on                                               |
| 2011-08-02 22:46:56 | [root] @ localhost [::1]      | Connect      | Access denied for user 'root'@'localhost' (using password: NO)  |
| 2011-08-02 22:47:02 | [root] @ localhost [::1]      | Connect      | root@localhost on                                               |
| 2011-08-02 22:48:50 | root[root] @ localhost [::1]  | Quit         |                                                                 |
| 2011-08-02 22:48:59 | [] @ localhost [::1]          | Connect      | Access denied for user 'iwan'@'localhost' (using password: NO)  |
| 2011-08-02 22:48:59 | [iwan] @ localhost [::1]      | Connect      | iwan@localhost as  on                                           |
| 2011-08-02 22:49:06 | [iwan2] @ localhost [::1]     | Connect      | iwan2@localhost on                                              |
| 2011-08-02 22:50:15 | iwan2[iwan2] @ localhost [::1] | Quit        |                                                                 |
| 2011-08-02 22:50:20 | [root] @ localhost [::1]      | Connect      | root@localhost on                                               |
| 2011-08-03 05:13:30 | [root] @ localhost [::1]      | Connect      | root@localhost on                                               |
| 2011-08-03 05:22:04 | [iwan2] @ localhost [::1]     | Connect      | iwan2@localhost on                                              |
| 2011-08-03 05:22:04 | [iwan2] @ localhost [::1]     | Connect      | Access denied for user 'iwan2'@'localhost' (using password: YES) |
| 2011-08-03 05:22:08 | [iwan2] @ localhost [::1]     | Connect      | iwan2@localhost on                                              |
| 2011-08-03 05:34:42 | root[root] @ localhost [::1]  | Quit         |                                                                 |
| 2011-08-03 05:34:48 | iwan2[iwan2] @ localhost [::1] | Quit        |                                                                 |
| 2011-08-04 22:20:21 | [root] @ localhost [::1]      | Connect      | root@localhost on                                               |
+---------------------+-------------------------------+--------------+-----------------------------------------------------------------+
26 rows in set (0.03 sec)
```

Gambar III.25 audit login dan logout berhasil maupun gagal

- Audit pengguna tertentu yang melakukan login dan logout dari database.

  Audit ini dapat dilakukan dengan DBMS MySQL dengan menggunakan log general yang hasil nya dapat dilihat pada database mysql yaitu pada tabel *general_log*. Contoh hasil audit pengguna tertentu yang melakukan login dan logout dari database dapat dilihat pada gambar III.26.



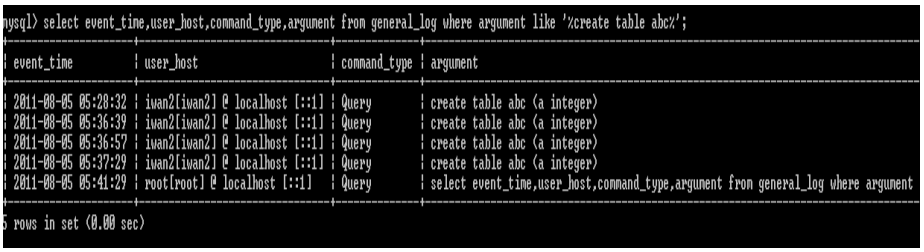
```
mysql> select event_time,user_host,command_type,argument from general_log where user_host like '[%iwan2]%' and command_type='Connect' or command_type='Quit';
+---------------------+-------------------------------+--------------+-----------------------------------------------------------------+
| event_time          | user_host                     | command_type | argument                                                        |
+---------------------+-------------------------------+--------------+-----------------------------------------------------------------+
| 2011-07-17 17:19:43 | root[root] @ localhost [::1]  | Quit         |                                                                 |
| 2011-07-17 17:38:19 | root[root] @ localhost [::1]  | Quit         |                                                                 |
| 2011-08-02 22:48:50 | root[root] @ localhost [::1]  | Quit         |                                                                 |
| 2011-08-02 22:49:06 | [iwan2] @ localhost [::1]     | Connect      | iwan2@localhost on                                              |
| 2011-08-02 22:50:15 | iwan2[iwan2] @ localhost [::1] | Quit        |                                                                 |
| 2011-08-03 05:22:04 | [iwan2] @ localhost [::1]     | Connect      | iwan2@localhost on                                              |
| 2011-08-03 05:22:04 | [iwan2] @ localhost [::1]     | Connect      | Access denied for user 'iwan2'@'localhost' (using password: YES) |
| 2011-08-03 05:22:08 | [iwan2] @ localhost [::1]     | Connect      | iwan2@localhost on                                              |
| 2011-08-03 05:34:42 | root[root] @ localhost [::1]  | Quit         |                                                                 |
| 2011-08-03 05:34:48 | iwan2[iwan2] @ localhost [::1] | Quit        |                                                                 |
+---------------------+-------------------------------+--------------+-----------------------------------------------------------------+
10 rows in set (0.00 sec)
```

Gambar III.26 audit pengguna tertentu login dan logout

2. Audit pengguna database
   - Audit pengguna yang melakukan perubahan pada object database (statement DDL).

     Audit ini dapat dilakukan dengan DBMS MySQL dengan menggunakan *log general* yang hasil nya dapat dilihat pada database mysql yaitu pada tabel *general_log*. Audit ini bertujuan untuk mecatat perintah-perintah yang dapat mempengaruhi atau menambah object dari sebuah database misalkan penambahan sebuah tabel atau menambah *field* (kolom) dari suatu tabel yang ada, menghapus suatu tabel, membuat suatu trigger dan lain sebagainya.

Contoh hasil audit pengguna yang melakukan perubahan pada object database dapat dilihat pada gambar III.27.

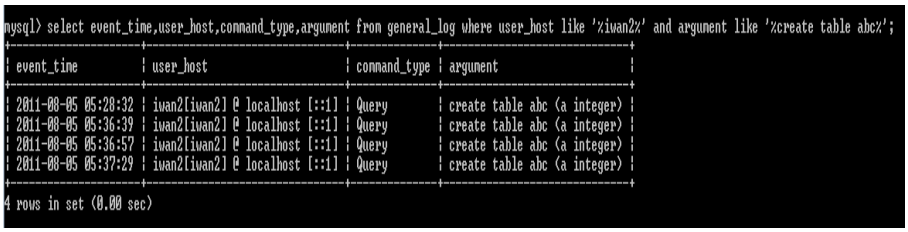


```
mysql> select event_time,user_host,command_type,argument from general_log where argument like '%create table abc%';
+---------------------+-------------------------------+--------------+-----------------------------------------------------------------------------------------------------------------------+
| event_time          | user_host                     | command_type | argument                                                                                                              |
+---------------------+-------------------------------+--------------+-----------------------------------------------------------------------------------------------------------------------+
| 2011-08-05 05:28:32 | iwan2[iwan2] @ localhost [::1] | Query        | create table abc (a integer)                                                                                          |
| 2011-08-05 05:36:39 | iwan2[iwan2] @ localhost [::1] | Query        | create table abc (a integer)                                                                                          |
| 2011-08-05 05:36:57 | iwan2[iwan2] @ localhost [::1] | Query        | create table abc (a integer)                                                                                          |
| 2011-08-05 05:37:29 | iwan2[iwan2] @ localhost [::1] | Query        | create table abc (a integer)                                                                                          |
| 2011-08-05 05:41:29 | root[root] @ localhost [::1]   | Query        | select event_time,user_host,command_type,argument from general_log where argument like '%create table abc%'           |
+---------------------+-------------------------------+--------------+-----------------------------------------------------------------------------------------------------------------------+
5 rows in set (0.00 sec)
```

Gambar III.27 audit pengguna melakukan perubahan object tertentu

- Audit pengguna tertentu yang melakukan perubahan pada object database (statement DDL).

  Audit ini dapat dilakukan dengan DBMS MySQL dengan menggunakan *log general* yang hasil nya dapat dilihat pada database mysql yaitu pada tabel *general_log*. Audit ini bertujuan untuk mecatat perintah-perintah yang dapat mempengaruhi atau menambah object dari sebuah database misalkan penambahan sebuah tabel atau menambah *field* (kolom) dari suatu tabel yang ada, menghapus suatu tabel, membuat suatu trigger dan lain sebagainya oleh pengguna tertentu. Contoh hasil audit pengguna tertentu yang melakukan perubahan pada object database dapat dilihat pada gambar III.28.



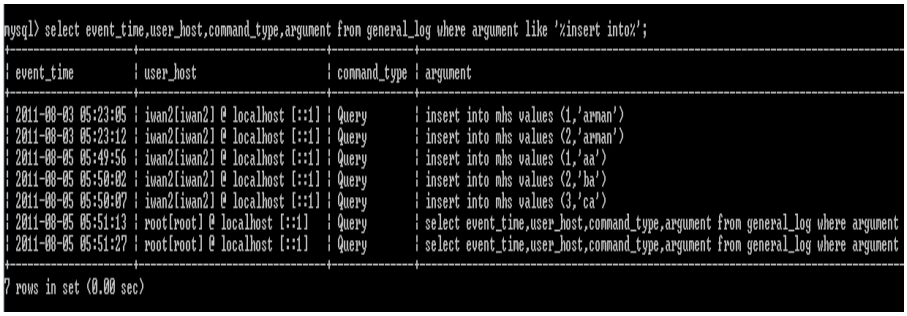
```
mysql> select event_time,user_host,command_type,argument from general_log where user_host like '%iwan2%' and argument like '%create table abc%';
+---------------------+-------------------------------+--------------+-----------------------------+
| event_time          | user_host                     | command_type | argument                    |
+---------------------+-------------------------------+--------------+-----------------------------+
| 2011-08-05 05:28:32 | iwan2[iwan2] @ localhost [::1] | Query        | create table abc (a integer) |
| 2011-08-05 05:36:39 | iwan2[iwan2] @ localhost [::1] | Query        | create table abc (a integer) |
| 2011-08-05 05:36:57 | iwan2[iwan2] @ localhost [::1] | Query        | create table abc (a integer) |
| 2011-08-05 05:37:29 | iwan2[iwan2] @ localhost [::1] | Query        | create table abc (a integer) |
+---------------------+-------------------------------+--------------+-----------------------------+
4 rows in set (0.00 sec)
```

Gambar III.28 audit pengguna tertentu melakukan perubahan object tertentu

- Audit pengguna yang melakukan perubahan isi dari suatu object database (statement DML).

  Audit ini dapat dilakukan dengan DBMS MySQL dengan menggunakan *log general* yang hasil nya dapat dilihat pada database mysql yaitu pada tabel *general_log*. Audit bertujuan untuk mecatat perintah-perintah yang memanipulasi isi dari object dari sebuah database misalkan penambahan isi

suatu tabel, perubahan isi dari suatu tabel, penghapusan isi dari suatu tabel dan adanya usaha untuk melihat isi dari suatu tabel. Contoh hasil audit pengguna yang melakukan perubahan isi dari suatu object database dapat dilihat pada gambar III.29.

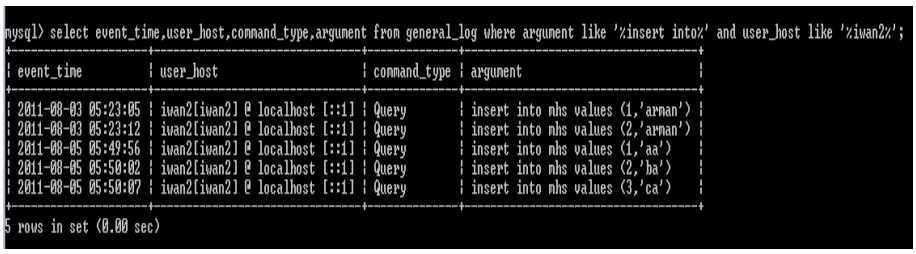


```
mysql> select event_time,user_host,command_type,argument from general_log where argument like '%insert into%';
+---------------------+-----------------------------+--------------+-----------------------------------------------------------------------------------------------------------------------+
| event_time          | user_host                   | command_type | argument                                                                                                              |
+---------------------+-----------------------------+--------------+-----------------------------------------------------------------------------------------------------------------------+
| 2011-08-03 05:23:05 | iwan2[iwan2] @ localhost [::1] | Query        | insert into mhs values (1,'arnan')                                                                                    |
| 2011-08-03 05:23:12 | iwan2[iwan2] @ localhost [::1] | Query        | insert into mhs values (2,'arnan')                                                                                    |
| 2011-08-05 05:49:56 | iwan2[iwan2] @ localhost [::1] | Query        | insert into mhs values (1,'aa')                                                                                       |
| 2011-08-05 05:50:02 | iwan2[iwan2] @ localhost [::1] | Query        | insert into mhs values (2,'ba')                                                                                       |
| 2011-08-05 05:50:07 | iwan2[iwan2] @ localhost [::1] | Query        | insert into mhs values (3,'ca')                                                                                       |
| 2011-08-05 05:51:13 | root[root] @ localhost [::1]   | Query        | select event_time,user_host,command_type,argument from general_log where argument like '%insert into%'               |
| 2011-08-05 05:51:27 | root[root] @ localhost [::1]   | Query        | select event_time,user_host,command_type,argument from general_log where argument like '%insert into%' and user_host |
+---------------------+-----------------------------+--------------+-----------------------------------------------------------------------------------------------------------------------+
7 rows in set (0.00 sec)
```

Gambar III.29 Audit pengguna yang melakukan perubahan isi dari suatu object database

- Audit pengguna tertentu yang melakukan perubahan isi dari suatu object database (statement DML).

  Audit ini dapat dilakukan dengan DBMS MySQL dengan menggunakan *log general* yang hasil nya dapat dilihat pada database mysql yaitu pada tabel *general_log*. Audit bertujuan untuk mecatat perintah-perintah yang memanipulasi isi dari object dari sebuah database misalkan penambahan isi suatu tabel, perubahan isi dari suatu tabel, penghapusan isi dari suatu tabel dan adanya usaha untuk melihat isi dari suatu tabel oleh pengguna tertentu. Contoh hasil audit pengguna tertentu yang melakukan perubahan isi dari suatu object database dapat dilihat pada gambar III.30.

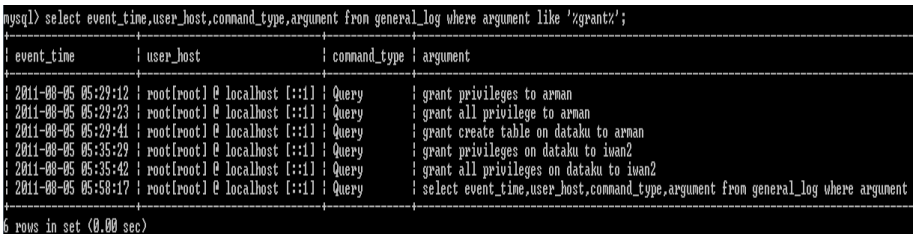


```
mysql> select event_time,user_host,command_type,argument from general_log where argument like '%insert into%' and user_host like '%iwan2%';
+---------------------+-----------------------------+--------------+-----------------------------------+
| event_time          | user_host                   | command_type | argument                          |
+---------------------+-----------------------------+--------------+-----------------------------------+
| 2011-08-03 05:23:05 | iwan2[iwan2] @ localhost [::1] | Query        | insert into mhs values (1,'arnan') |
| 2011-08-03 05:23:12 | iwan2[iwan2] @ localhost [::1] | Query        | insert into mhs values (2,'arnan') |
| 2011-08-05 05:49:56 | iwan2[iwan2] @ localhost [::1] | Query        | insert into mhs values (1,'aa')    |
| 2011-08-05 05:50:02 | iwan2[iwan2] @ localhost [::1] | Query        | insert into mhs values (2,'ba')    |
| 2011-08-05 05:50:07 | iwan2[iwan2] @ localhost [::1] | Query        | insert into mhs values (3,'ca')    |
+---------------------+-----------------------------+--------------+-----------------------------------+
5 rows in set (0.00 sec)
```

Gambar III.30 Audit pengguna tertentu yang melakukan perubahan isi dari suatu object database

- Audit Pengguna yang melakukan perintah pengendalian pengaksesan data (statement DCL).

  Audit ini dapat dilakukan dengan DBMS MySQL dengan menggunakan *log general* yang hasil nya dapat dilihat pada database mysql yaitu pada tabel *general_log*. Audit bertujuan untuk mecatat perintah-perintah pengendalian pengaksesan data oleh suatu pengguna terhadap pengguna yang lain seperti pemberian hak akses untuk memanipulasi data pada tabel yang dibuat suatu user ke user yang lain atau sebaliknya dan mencabut hak akses yang telah diberikan. Contoh hasil audit pengguna yang melakukan perintah pengendalian pengaksesan data dapat dilihat pada gambar III.31.

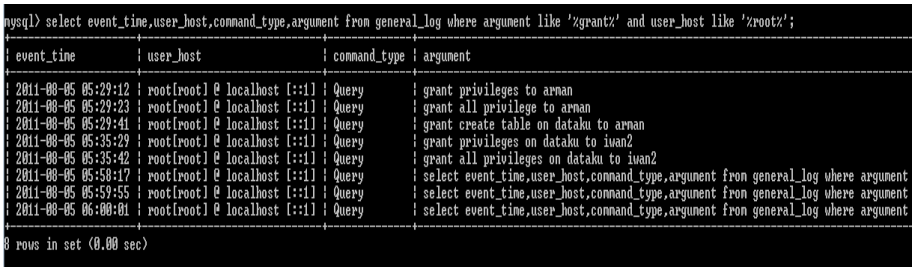


```
mysql> select event_time,user_host,command_type,argument fron general_log where argument like '%grant%';
+---------------------+-------------------------+--------------+-----------------------------------------------------------------------------------------------------------+
| event_time          | user_host               | command_type | argument                                                                                                  |
+---------------------+-------------------------+--------------+-----------------------------------------------------------------------------------------------------------+
| 2011-08-05 05:29:12 | root[root] @ localhost [::1] | Query        | grant privileges to arman                                                                                 |
| 2011-08-05 05:29:23 | root[root] @ localhost [::1] | Query        | grant all privilege to arman                                                                              |
| 2011-08-05 05:29:41 | root[root] @ localhost [::1] | Query        | grant create table on dataku to arman                                                                     |
| 2011-08-05 05:35:29 | root[root] @ localhost [::1] | Query        | grant privileges on dataku to iwan2                                                                       |
| 2011-08-05 05:35:42 | root[root] @ localhost [::1] | Query        | grant all privileges on dataku to iwan2                                                                   |
| 2011-08-05 05:58:17 | root[root] @ localhost [::1] | Query        | select event_time,user_host,command_type,argument fron general_log where argument like '%grant%';        |
+---------------------+-------------------------+--------------+-----------------------------------------------------------------------------------------------------------+
6 rows in set (0.00 sec)
```

Gambar III.31 Audit Pengguna yang melakukan perintah pengendalian pengaksesan data

- Audit Pengguna tertentu yang melakukan perintah pengendalian pengaksesan data (statement DCL).

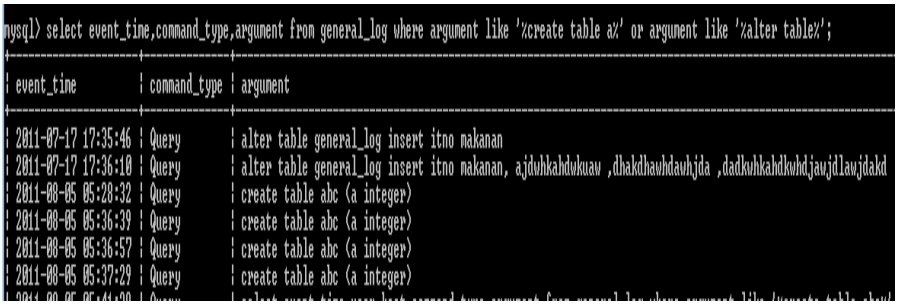
  Audit ini dapat dilakukan dengan DBMS MySQL dengan menggunakan *log general* yang hasil nya dapat dilihat pada database mysql yaitu pada tabel *general_log*. Audit bertujuan untuk mecatat perintah-perintah pengendalian pengaksesan data oleh suatu pengguna terhadap pengguna yang lain seperti pemberian hak akses untuk memanipulasi data pada tabel yang dibuat suatu user ke user yang lain atau sebaliknya dan mencabut hak akses yang telah diberikan oleh pengguna tertentu. Contoh hasil audit pengguna tertentu yang melakukan perintah pengendalian pengaksesan data dapat dilihat pada gambar III.32.



Gambar III.32 Audit Pengguna tertentu yang melakukan perintah pengendalian pengaksesan data

3. Audit Object Database.
   - Audit statement (perintah) yang terjadi pada object tertentu (DDL Statement).

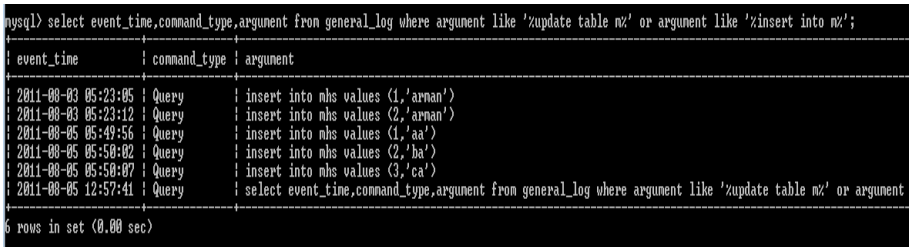
     Audit ini bertujuan untuk mencatat statement atau query apa yang digunakan oleh pengguna database terhadap object database. Apakah terdapat object baru, apakah ada suatu object yang mengalami perubahan atau sebaliknya mengalami. Contoh hasil audit statement yang terjadi pada object tertentu dapat dilihat pada gambar III.33.



Gambar III.33 Audit statement pada object tertentu

   - Audit statement (perintah) yang terjadi pada isi dari object database (DML Statement).

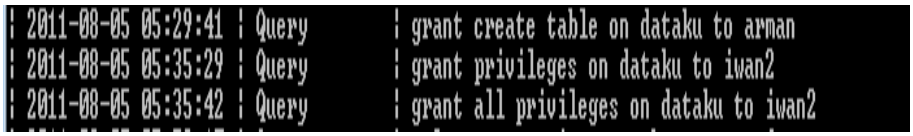
     Audit ini bertujuan untuk mencatat statement atau query apa yang digunakan oleh pengguna database terhadap isi dari object database. Apakah menggunakan perintah penambahan, pembaharuan, penghapusan atau peritah penampilan data-data object database. Contoh hasil audit terjadi pada isi dari object database dapat dilihat pada gambar III.34.



Gambar III.34 Audit isi dari object

- Audit statement (perintah) perubahan pada pengendalian pengaksesan data pada object database (DCL Statement).

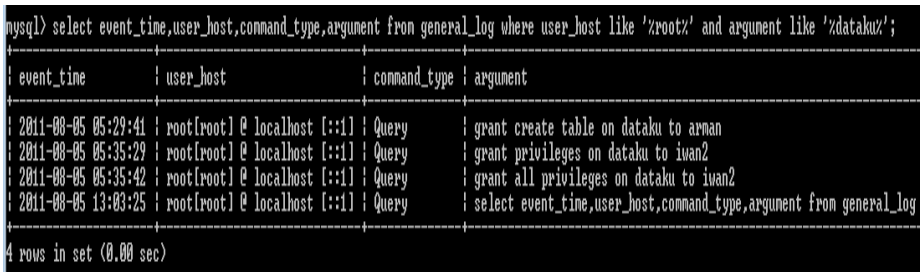
  Audit ini bertujuan untuk mencatat statement atau query apa yang digunakan oleh pengguna database terhadap isi dari object database untuk hak akses ke data tersebut. Hasil dari audit ini dapat dilihat pada gambar III.35



Gambar III.35 Audit perubahan pada pengendalian pengaksesan data pada object database

- Audit administrator

  Audit ini digunakan untuk mengaudit user administrator. Karena selain user biasa user admin juga perlu diaudit. Tidaka menutup kemungkinan bahwa password dari user adminstrator tersebut diketahui oleh pihak yang tidak berwewenang. Contoh dari audit user administrator dapat dilihat pada gambar III. 36.



Gambar III.36 Audit administrator

- Audit pernyataan DML kolom tertentu dengan kondisi NULL.

  Audit ini digunakan untuk mengaudit pernyataan tertentu pada suatu field pada suatu tabel dengan kondisi NULL. Jadi semua penyataan yang mengakses suatu field tersebut akan dicatat. Hasil audit ini dapat dilihat pada gambar III.37.

```
mysql> select event_tine,user_host,argument from general_log
    -> where user_host like 'admin_pel%'
    -> and (argument like 'insert into pelanggan %' or argument like 'update pelanggan %'
    -> or argument like 'delete%pelanggan'or argument like 'select%pelanggan');
+---------------------+---------------------------------+-----------------------------------------------------------------------------------+
| event_tine          | user_host                       | argument                                                                          |
+---------------------+---------------------------------+-----------------------------------------------------------------------------------+
| 2011-08-11 12:36:14 | admin_pel[admin_pel] @ localhost [::1] | select * from pelanggan                                                    |
| 2011-08-11 12:37:03 | admin_pel[admin_pel] @ localhost [::1] | insert into pelanggan values ('P01','arman','sei panas','aaa','aaa')       |
| 2011-08-11 12:37:40 | admin_pel[admin_pel] @ localhost [::1] | insert into pelanggan values ('P01','arman','sei panas','aaa','aaa')       |
| 2011-08-11 15:32:58 | admin_pel[admin_pel] @ localhost [::1] | insert into pelanggan values ('P01','iwan','sei panas','gmail','1234')     |
| 2011-08-11 15:33:03 | admin_pel[admin_pel] @ localhost [::1] | insert into pelanggan values ('P02','iwan','sei panas','gmail','1234')     |
| 2011-08-11 15:33:07 | admin_pel[admin_pel] @ localhost [::1] | insert into pelanggan values ('P03','iwan','sei panas','gmail','1234')     |
| 2011-08-11 15:33:53 | admin_pel[admin_pel] @ localhost [::1] | update pelanggan set id_pelanggan='P05' where id_pelanggan='P01'            |
| 2011-08-11 15:35:24 | admin_pel[admin_pel] @ localhost [::1] | insert into pelanggan values ('P03','iwan','sei panas','gmail','1234')     |
| 2011-08-11 15:35:30 | admin_pel[admin_pel] @ localhost [::1] | insert into pelanggan values ('P00','iwan','sei panas','gmail','1234')     |
| 2011-08-11 15:35:37 | admin_pel[admin_pel] @ localhost [::1] | select * from pelanggan                                                    |
| 2011-08-11 15:37:26 | admin_pel[admin_pel] @ localhost [::1] | insert into pelanggan values ('P10','arman','panas','yahoo','123')         |
+---------------------+---------------------------------+-----------------------------------------------------------------------------------+
11 rows in set (0.00 sec)
```

Gambar III.37 Audit pernyataan DML kolom tertentu dengan kondisi NULL

III.3 Audit pada DBMS PostgreSQL

Untuk melakukan audit pada DBMS PostgreSQL dapat menggunakan versi PostgreSQL 7.0 keatas karna versi PostgreSQL ini sudah mempunyai fitur yang lengkap seperti *log*, *trigger* dan *procedure* untuk melakukan audit. Seperti hal nya dengan DBMS MySQL, DBMS PostgreSQL juga tidak memilki fitur audit khusus dan memilki fitur log yang dapat digunakan untuk audit database. Secara default log postgresql disimpan pada sebuah file. Dan pada penelitian ini akan mengimpor hasil log tersebut kesebuah file dengan cara mengcopy file tersebut kedalam suatu tabel di database. Dengan tujuan agar mempermudah dalam pencarian data dan kebutuhan akan audit itu sendiri.

Sesuai dengan audit yang dapat dilakukan di DBMS Oracel diatas, berikut ini audit Oracle 10g yang dapat diimplementasikan pada DBMS PostgreSQL.

1. Audit Session
   - Audit pengguna yang melakukan login berhasil ke database.

Audit ini dapat dilakukan dengan DBMS PostgreSQL dengan menggunakan *log line prefix* yang hasil nya dapat dilihat di dalam database postgres didalam tabel postgres_log. Hasil audit pengguna yang melakukan login berhasil ke database dapat dilihat pada gambar III.38.

Gambar III.38 Audit pengguna yang melakukan login berhasil

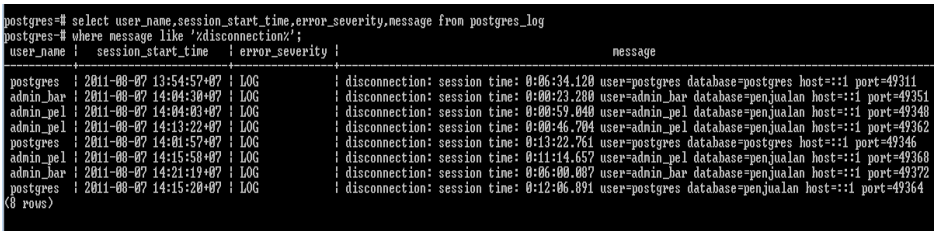
- Audit pengguna yang melakukan login gagal ke database.

  Audit ini dapat dilakukan dengan DBMS PostgreSQL dengan menggunakan *log postgresql* yang hasil nya dapat dilihat di dalam database postgres didalam tabel postgres_log. Contoh hasil audit pengguna yang melakukan login berhasil ke database dapat dilihat pada gambar III.39.

Gambar III.39 Audit pengguna yang melakukan login gagal

- Audit pengguna yang melakukan logout dari database.

  Audit ini dapat dilakukan dengan DBMS PostgreSQL dengan menggunakan *log postgresql* yang hasil nya dapat dilihat di dalam database postgres didalam tabel postgres_log. Contoh hasil audit pengguna yang melakukan logout ke database dapat dilihat pada gambar III.40.

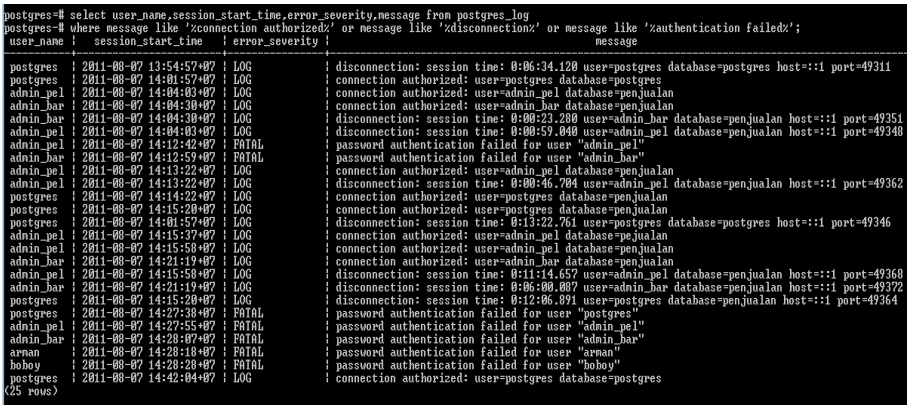


```
postgres=# select user_name,session_start_time,error_severity,message from postgres_log
postgres-# where message like '%disconnection%';
 user_name |   session_start_time   | error_severity |                                        message
-----------+------------------------+----------------+-----------------------------------------------------------------------------------------
 postgres  | 2011-08-07 13:54:57+07 | LOG            | disconnection: session time: 0:06:34.120 user=postgres database=postgres host=::1 port=49311
 admin_bar | 2011-08-07 14:04:30+07 | LOG            | disconnection: session time: 0:00:23.280 user=admin_bar database=penjualan host=::1 port=49351
 admin_pel | 2011-08-07 14:04:03+07 | LOG            | disconnection: session time: 0:00:59.040 user=admin_pel database=penjualan host=::1 port=49340
 admin_pel | 2011-08-07 14:13:22+07 | LOG            | disconnection: session time: 0:00:46.704 user=admin_pel database=penjualan host=::1 port=49362
 postgres  | 2011-08-07 14:01:57+07 | LOG            | disconnection: session time: 0:13:22.761 user=postgres database=postgres host=::1 port=49346
 admin_pel | 2011-08-07 14:15:58+07 | LOG            | disconnection: session time: 0:11:14.657 user=admin_pel database=penjualan host=::1 port=49368
 admin_bar | 2011-08-07 14:21:19+07 | LOG            | disconnection: session time: 0:06:00.087 user=admin_bar database=penjualan host=::1 port=49372
 postgres  | 2011-08-07 14:15:20+07 | LOG            | disconnection: session time: 0:12:06.891 user=postgres database=penjualan host=::1 port=49364
(8 rows)
```

Gambar III.40 Audit pengguna yang melakukan logout

- Audit pengguna yang melakukan login dan logout pada database baik yang berhasil maupun yang gagal.

  Audit ini dapat dilakukan dengan DBMS PostgreSQL dengan menggunakan *log postgresql* yang hasil nya dapat dilihat di dalam database postgres didalam tabel postgres_log. Contoh hasil audit pengguna yang melakukan login dan log out pada database baik yang berhasil maupun yang gagal dapat dilihat pada gambar III.41.

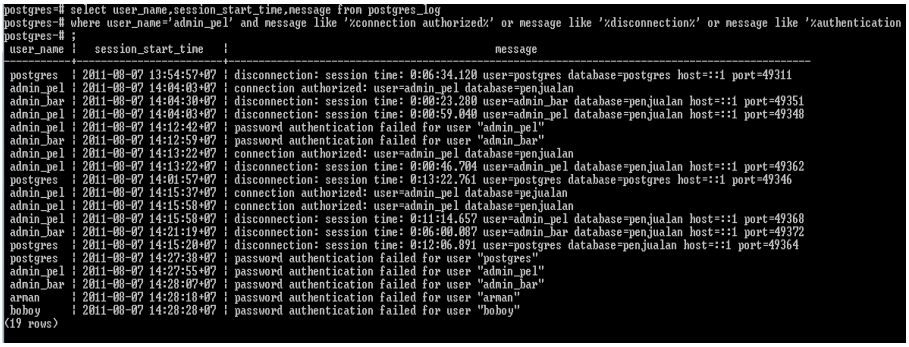


```
postgres=# select user_name,session_start_time,error_severity,message from postgres_log
postgres-# where message like '%connection authorized%' or message like '%disconnection%' or message like '%authentication failed%';
 user_name |   session_start_time   | error_severity |                                        message
-----------+------------------------+----------------+-----------------------------------------------------------------------------------------
 postgres  | 2011-08-07 13:54:57+07 | LOG            | disconnection: session time: 0:06:34.120 user=postgres database=postgres host=::1 port=49311
 postgres  | 2011-08-07 14:01:57+07 | LOG            | connection authorized: user=postgres database=postgres
 admin_pel | 2011-08-07 14:04:03+07 | LOG            | connection authorized: user=admin_pel database=penjualan
 admin_bar | 2011-08-07 14:04:30+07 | LOG            | connection authorized: user=admin_bar database=penjualan
 admin_bar | 2011-08-07 14:04:30+07 | LOG            | disconnection: session time: 0:00:23.280 user=admin_bar database=penjualan host=::1 port=49351
 admin_pel | 2011-08-07 14:04:03+07 | LOG            | disconnection: session time: 0:00:59.040 user=admin_pel database=penjualan host=::1 port=49340
 admin_pel | 2011-08-07 14:12:42+07 | FATAL          | password authentication failed for user "admin_pel"
 admin_bar | 2011-08-07 14:12:59+07 | FATAL          | password authentication failed for user "admin_bar"
 admin_pel | 2011-08-07 14:13:22+07 | LOG            | connection authorized: user=admin_pel database=penjualan
 admin_pel | 2011-08-07 14:13:22+07 | LOG            | disconnection: session time: 0:00:46.704 user=admin_pel database=penjualan host=::1 port=49362
 postgres  | 2011-08-07 14:14:22+07 | LOG            | connection authorized: user=postgres database=penjualan
 postgres  | 2011-08-07 14:15:20+07 | LOG            | connection authorized: user=postgres database=penjualan
 postgres  | 2011-08-07 14:01:57+07 | LOG            | disconnection: session time: 0:13:22.761 user=postgres database=postgres host=::1 port=49346
 admin_pel | 2011-08-07 14:15:37+07 | LOG            | connection authorized: user=admin_pel database=pejualan
 admin_pel | 2011-08-07 14:15:58+07 | LOG            | connection authorized: user=admin_pel database=penjualan
 admin_bar | 2011-08-07 14:21:19+07 | LOG            | connection authorized: user=admin_bar database=penjualan
 admin_pel | 2011-08-07 14:15:58+07 | LOG            | disconnection: session time: 0:11:14.657 user=admin_pel database=penjualan host=::1 port=49368
 admin_bar | 2011-08-07 14:21:19+07 | LOG            | disconnection: session time: 0:06:00.087 user=admin_bar database=penjualan host=::1 port=49372
 postgres  | 2011-08-07 14:15:20+07 | LOG            | disconnection: session time: 0:12:06.891 user=postgres database=penjualan host=::1 port=49364
 postgres  | 2011-08-07 14:27:38+07 | FATAL          | password authentication failed for user "postgres"
 admin_pel | 2011-08-07 14:27:55+07 | FATAL          | password authentication failed for user "admin_pel"
 admin_bar | 2011-08-07 14:28:07+07 | FATAL          | password authentication failed for user "admin_bar"
 arman     | 2011-08-07 14:28:18+07 | FATAL          | password authentication failed for user "arman"
 boboy     | 2011-08-07 14:28:28+07 | FATAL          | password authentication failed for user "boboy"
 postgres  | 2011-08-07 14:42:04+07 | LOG            | connection authorized: user=postgres database=postgres
(25 rows)
```

Gambar III.41 Audit pengguna yang melakukan login dan logout

- Audit pengguna tertentu yang melakukan login dan logout dari database.

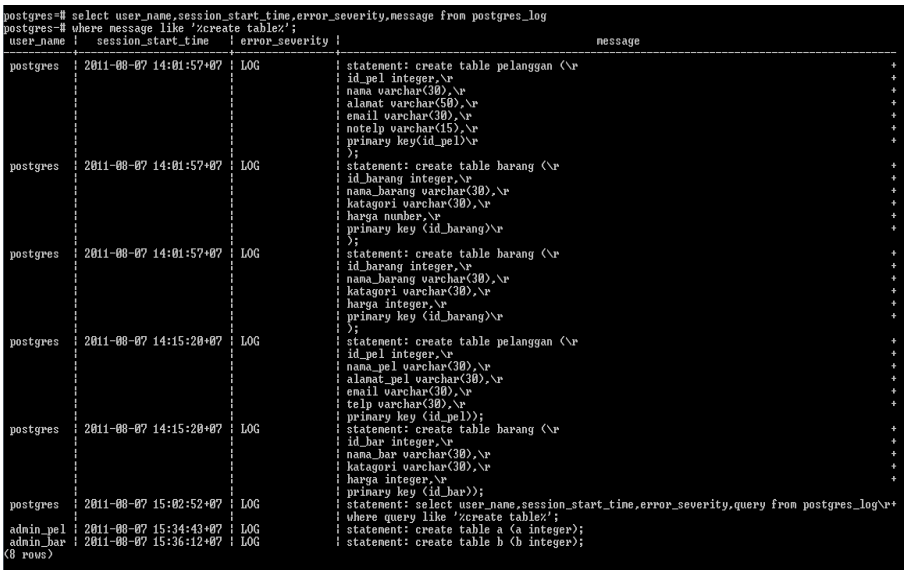
  Audit ini dapat dilakukan dengan DBMS PostgreSQL dengan menggunakan *log postgresql* yang hasil nya dapat dilihat di dalam database postgres didalam tabel postgres_log. Contoh hasil audit pengguna tertentu yang melakukan login dan logout ke database dapat dilihat pada gambar III.42.



Gambar III.42 Audit pengguna tertentu yang melakukan login dan logout

2. Audit pengguna database
   - Audit pengguna yang melakukan perubahan pada object database (statement DDL).

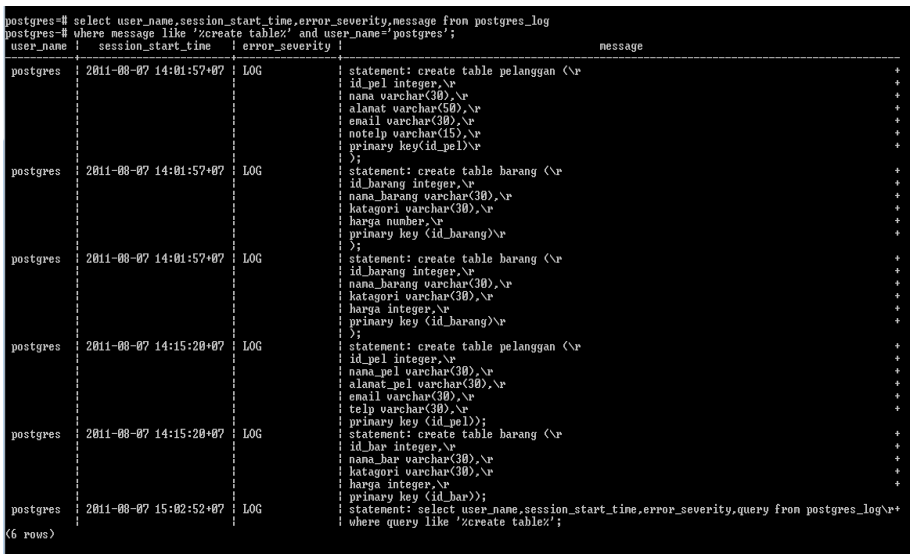
     Audit ini dapat dilakukan dengan DBMS PostgreSQL dengan menggunakan *log postgresql* yang hasil nya dapat dilihat di dalam database postgres didalam tabel postgres_log. Audit ini bertujuan untuk mecatat perintah-perintah yang dapat mempengaruhi atau menambah object dari sebuah database misalkan penambahan sebuah tabel atau menambah *field* (kolom) dari suatu tabel yang ada, menghapus suatu tabel, membuat suatu trigger dan lain sebagainya. Contoh hasil audit pengguna yang melakukan perubahan pada object database dapat dilihat pada gambar III.43.



Gambar III.43 Audit pengguna yang melakukan perubahan pada object database

- Audit pengguna tertentu yang melakukan perubahan pada object database (statement DDL).

  Audit ini dapat dilakukan dengan DBMS PostgreSQL dengan menggunakan *log postgresql* yang hasil nya dapat dilihat di dalam database postgres didalam tabel postgres_log. Audit ini bertujuan untuk mecatat perintah-perintah yang dapat mempengaruhi atau menambah object dari sebuah database misalkan penambahan sebuah tabel atau menambah *field* (kolom) dari suatu tabel yang ada, menghapus suatu tabel, membuat suatu trigger dan lain sebagainya oleh pengguna tertentu. Contoh hasil audit pengguna tertentu yang melakukan perubahan pada object database dapat dilihat pada gambar III.44.

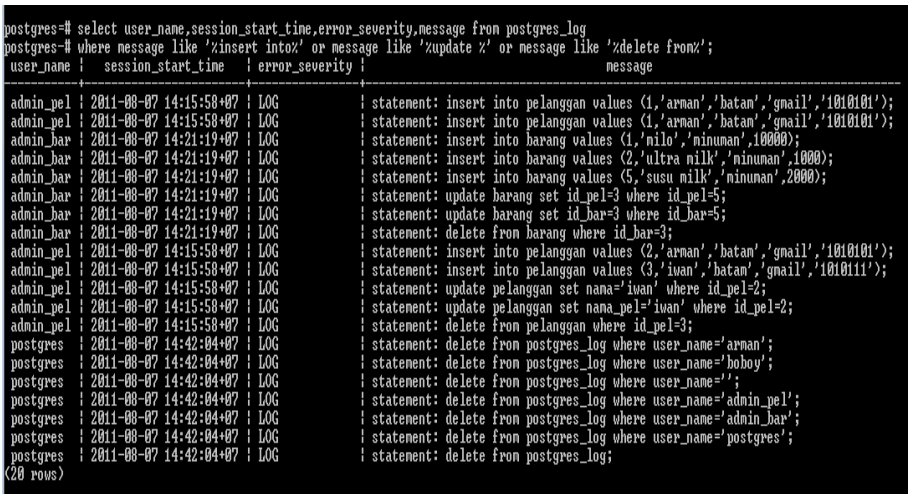


Gambar III.44 Audit pengguna tertentu yang melakukan perubahan pada object database

- Audit pengguna yang melakukan perubahan isi dari suatu object database (statement DML).

  Audit ini dapat dilakukan dengan DBMS PostgreSQL dengan menggunakan *log postgresql* yang hasil nya dapat dilihat di dalam database postgres didalam tabel postgres_log. Audit bertujuan untuk mecatat perintah-perintah

yang memanipulasi isi dari object dari sebuah database misalkan penambahan isi suatu tabel, perubahan isi dari suatu tabel, penghapusan isi dari suatu tabel dan adanya usaha untuk melihat isi dari suatu tabel. Contoh hasil audit pengguna yang melakukan perubahan isi dari suatu object database dapat dilihat pada gambar III.45.



```
postgres=# select user_name,session_start_time,error_severity,message from postgres_log
postgres-# where message like '%insert into%' or message like '%update %' or message like '%delete from%';
 user_name |   session_start_time   | error_severity |                                message
-----------+------------------------+----------------+-----------------------------------------------------------------------
 admin_pel | 2011-08-07 14:15:58+07 | LOG            | statement: insert into pelanggan values (1,'arman','batam','gmail','1010101');
 admin_pel | 2011-08-07 14:15:58+07 | LOG            | statement: insert into pelanggan values (1,'arman','batam','gmail','1010101');
 admin_bar | 2011-08-07 14:21:19+07 | LOG            | statement: insert into barang values (1,'milo','minuman',10000);
 admin_bar | 2011-08-07 14:21:19+07 | LOG            | statement: insert into barang values (2,'ultra milk','minuman',1000);
 admin_bar | 2011-08-07 14:21:19+07 | LOG            | statement: insert into barang values (5,'susu milk','minuman',2000);
 admin_bar | 2011-08-07 14:21:19+07 | LOG            | statement: update barang set id_pel=3 where id_pel=5;
 admin_bar | 2011-08-07 14:21:19+07 | LOG            | statement: update barang set id_bar=3 where id_bar=5;
 admin_bar | 2011-08-07 14:21:19+07 | LOG            | statement: delete fron barang where id_bar=3;
 admin_pel | 2011-08-07 14:15:58+07 | LOG            | statement: insert into pelanggan values (2,'arman','batam','gmail','1010101');
 admin_pel | 2011-08-07 14:15:58+07 | LOG            | statement: insert into pelanggan values (3,'iwan','batam','gmail','1010111');
 admin_pel | 2011-08-07 14:15:58+07 | LOG            | statement: update pelanggan set nama='iwan' where id_pel=2;
 admin_pel | 2011-08-07 14:15:58+07 | LOG            | statement: update pelanggan set nama_pel='iwan' where id_pel=2;
 admin_pel | 2011-08-07 14:15:58+07 | LOG            | statement: delete fron pelanggan where id_pel=3;
 postgres  | 2011-08-07 14:42:04+07 | LOG            | statement: delete from postgres_log where user_name='arman';
 postgres  | 2011-08-07 14:42:04+07 | LOG            | statement: delete from postgres_log where user_name='boboy';
 postgres  | 2011-08-07 14:42:04+07 | LOG            | statement: delete from postgres_log where user_name='';
 postgres  | 2011-08-07 14:42:04+07 | LOG            | statement: delete from postgres_log where user_name='admin_pel';
 postgres  | 2011-08-07 14:42:04+07 | LOG            | statement: delete from postgres_log where user_name='admin_bar';
 postgres  | 2011-08-07 14:42:04+07 | LOG            | statement: delete from postgres_log where user_name='postgres';
 postgres  | 2011-08-07 14:42:04+07 | LOG            | statement: delete from postgres_log;
(20 rows)
```

Gambar III.45 Audit pengguna yang melakukan perubahan isi dari suatu object database

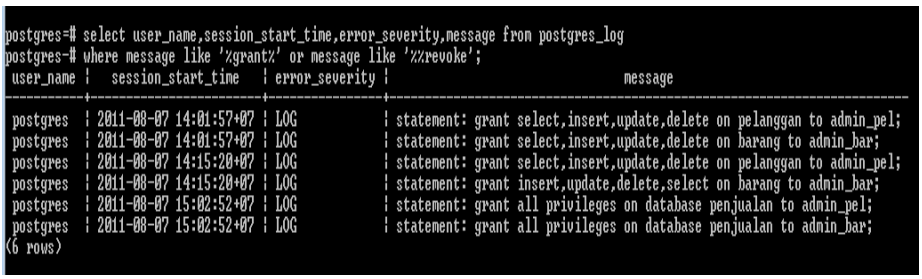
- Audit pengguna tertentu yang melakukan perubahan isi dari suatu object database (statement DML).

  Audit ini dapat dilakukan dengan DBMS PostgreSQL dengan menggunakan *log postgresql* yang hasil nya dapat dilihat di dalam database postgres didalam tabel postgres_log. Audit bertujuan untuk mecatat perintah-perintah yang memanipulasi isi dari object dari sebuah database misalkan penambahan isi suatu tabel, perubahan isi dari suatu tabel, penghapusan isi dari suatu tabel dan adanya usaha untuk melihat isi dari suatu tabel oleh pengguna tertentu. Contoh hasil audit pengguna tertentu yang melakukan perubahan isi dari suatu object database dapat dilihat pada gambar III.46.

Gambar III.46 Audit pengguna tertentu yang melakukan perubahan isi dari suatu object database

- Audit Pengguna yang melakukan perintah pengendalian pengaksesan data (statement DCL).

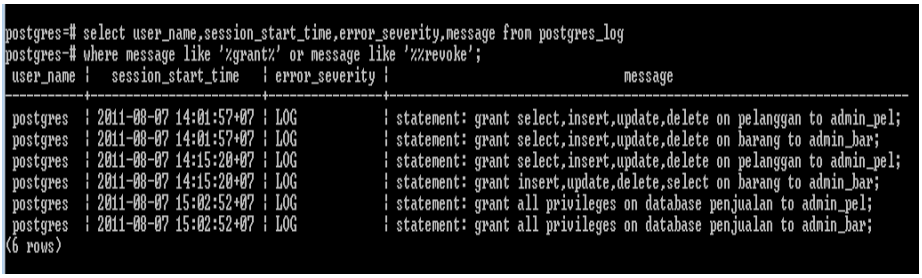
  Audit ini dapat dilakukan dengan DBMS PostgreSQL dengan menggunakan *log postgresql* yang hasil nya dapat dilihat di dalam database postgres didalam tabel postgres_log. Audit bertujuan untuk mecatat perintah-perintah pengendalian pengaksesan data oleh suatu pengguna terhadap pengguna yang lain seperti pemberian hak akses untuk memanipulasi data pada tabel yang dibuat suatu user ke user yang lain atau sebaliknya dan mencabut hak akses yang telah diberikan. Contoh hasil audit pengguna yang melakukan perintah pengendalian pengaksesan data dapat dilihat pada gambar III.47.



Gambar III.47 Audit Pengguna yang melakukan perintah pengendalian pengaksesan data

- Audit Pengguna tertentu yang melakukan perintah pengendalian pengaksesan data (statement DCL).

Audit ini dapat dilakukan dengan DBMS PostgreSQL dengan menggunakan *log postgresql* yang hasil nya dapat dilihat di dalam database postgres didalam tabel postgres_log. Audit bertujuan untuk mecatat perintah-perintah pengendalian pengaksesan data oleh suatu pengguna terhadap pengguna yang lain seperti pemberian hak akses untuk memanipulasi data pada tabel yang dibuat suatu user ke user yang lain atau sebaliknya dan mencabut hak akses yang telah diberikan oleh pengguna tertentu. Contoh hasil audit pengguna tertentu yang melakukan perintah pengendalian pengaksesan data dapat dilihat pada gambar III.48.



Gambar III.48 Audit Pengguna tertentu yang melakukan perintah pengendalian pengaksesan data

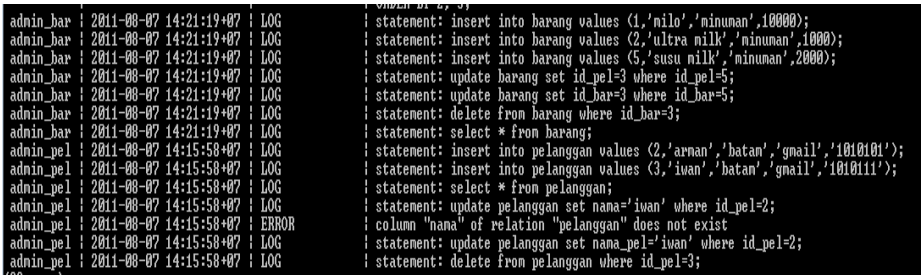
3. Audit Object Database.
   - Audit statement (perintah) yang terjadi pada object tertentu (DDL Statement).

     Audit ini dapat dilakukan dengan DBMS PostgreSQL dengan menggunakan *log postgresql* yang hasil nya dapat dilihat di dalam database postgres didalam tabel postgres_log. Audit ini bertujuan untuk mencatat statement atau query apa yang digunakan oleh pengguna database terhadap object database. Apakah terdapat object baru, apakah ada suatu object yang mengalami perubahan atau sebaliknya mengalami. Contoh hasil audit statement yang terjadi pada object tertentu dapat dilihat pada gambar III.49.

Gambar III.49 Audit statement yang terjadi pada object tertentu

- Audit statement (perintah) yang terjadi pada isi dari object database (DML Statement).

  Audit ini dapat dilakukan dengan DBMS PostgreSQL dengan menggunakan *log postgresql* yang hasil nya dapat dilihat di dalam database postgres didalam tabel postgres_log. Audit ini bertujuan untuk mencatat statement atau query apa yang digunakan oleh pengguna database terhadap isi dari object database. Apakah menggunakan perintah penambahan, pembaharuan, penghapusan atau peritah penampilan data-data object database. Contoh hasil audit terjadi pada isi dari object database dapat dilihat pada gambar III.50.



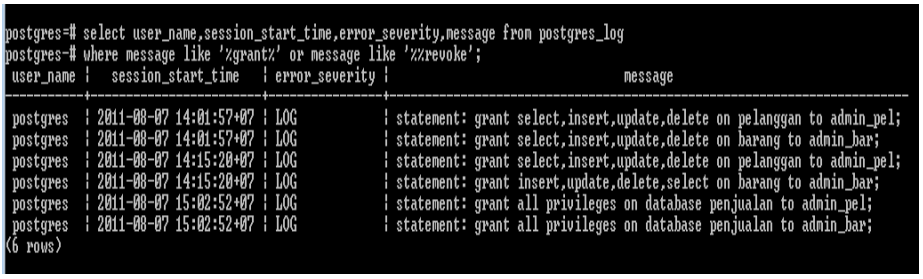
Gambar III.50 Audit statement yang terjadi pada isi object

- Audit statement (perintah) perubahan pada pengendalian pengaksesan data pada object database (DCL Statement).

  Audit statement (perintah) yang terjadi pada isi dari object database (DML Statement).

  Audit ini dapat dilakukan dengan DBMS PostgreSQL dengan menggunakan *log postgresql* yang hasil nya dapat dilihat di dalam database postgres didalam tabel postgres_log. Audit ini bertujuan untuk mencatat statement

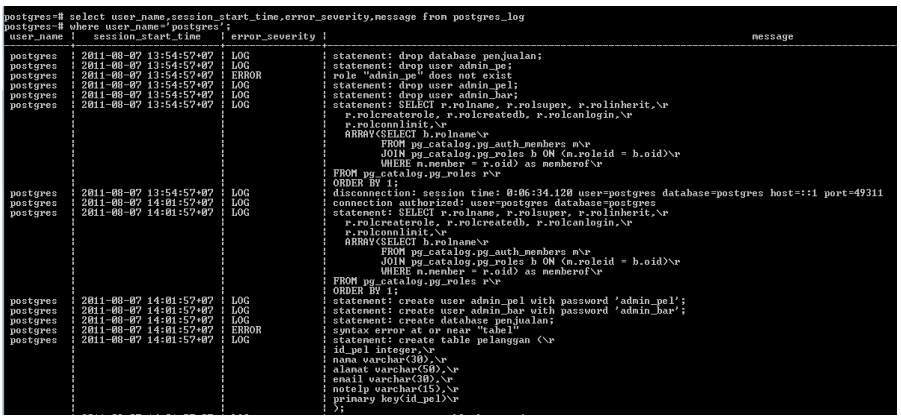
atau query apa yang digunakan oleh pengguna database terhadap isi dari object database untuk hak akses ke data tersebut. Contoh hasil audit ini dapat dilihat pada gambar III.51.



Gambar III.51 Audit statement perubahan pada pengendalian pengaksesan data pada object database

- Audit administrator

  Audit ini digunakan untuk mengaudit user administrator. Karena selain user biasa user admin juga perlu diaudit. Tidaka menutup kemungkinan bahwa password dari user adminstrator tersebut diketahui oleh pihak yang tidak berwewenang. Contoh dari audit user administrator dapat dilihat pada gambar III.52.



Gambar III.52 Audit administrator

- Audit pernyataan DML kolom tertentu dengan kondisi NULL.

Audit ini digunakan untuk mengaudit pernyataan tertentu pada suatu field pada suatu tabel dengan kondisi NULL. Jadi semua penyataan yang mengakses suatu field tersebut akan dicatat. Contoh audit ini dapat dilihat pada gambar III.53.

Gambar III.53 Audit pernyataan DML kolom tertentu dengan kondisi NULL

III.4 Langkah Penyelesaian Masalah

Dari hasil implementasi yang dilakukan pada ketiga DBMS tersebut maka diperolehlah pebandingan yang dapat dilihat pada tabel III.1.

Tabel III.1 Perbandingan audit

DBMS / Jenis Audit	Kategori	Oracle 10g	MySQL 5.5.8	PostgreSQL 9.0
Audit Trail	Pengguna login berhasil	V	V	V
	Pengguna login gagal	V	V	V
	Pengguna logout	V	V	V
	Pengguna proses login dan logout	V	V	V
	Pengguna login dan logout	V	V	V
	Pengguna yang melakukan perintah DDL	V	V	V
	Pengguna tertentu yang	V	V	V

	melakukan perintah DDL			
	Pengguna yang melakukan perintah DML	V	V	V
	Pengguna tertentu yang melakukan perintah DML	V	V	V
	Pengguna yang melakukan perintah DCL	V	V	V
	Pengguna tertentu yang melakukan perintah DCL	V	V	V
	Statement DDL pada object tertentu	V	V	V
	Statement DML pada object tertentu	V	V	V
	Statement DCL pada object tertentu	V	V	V
	Audit user adminstrator	V	V	V
Value Base Auditing	Audit insert data	V	?	?
	Audit delete data	V	?	?
	Audit update data	V	?	?
Fine grained Auditing	Mengaudit kolom tertentu dengan menggunakan kondisi tertentu.	V	X	X
	Audit pernyataan DML kolom tertentu dengan kondisi NULL.	V	V	V

Catatan: Semua jenis audit dan katagori audit diatas merupakan audit yang penyimpanan hasil audit disimpan di tabel database dan untuk MySQL menggunakan *general log* dan PostgreSQL menggunakan *log* PostgreSQL.

Dari hasil resume perbandingan tersebut dapat disimpulkan bahwa ada beberapa audit di DBMS Oracle 10g yang belum bisa dilakukan oleh DBMS MySQL dan Postgresql yaitu:

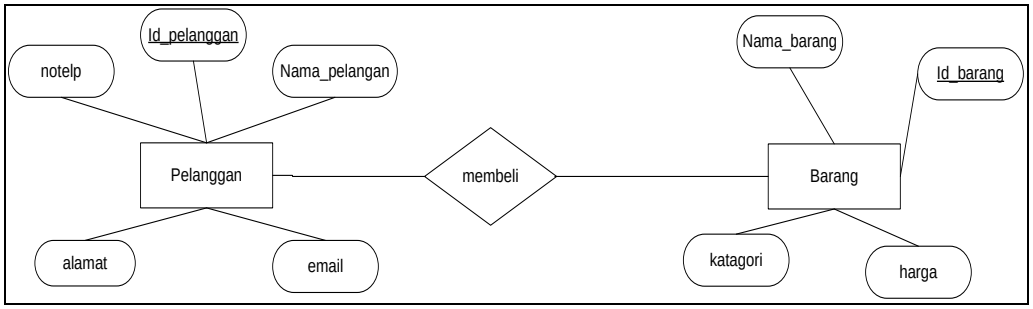
1. *Value base auditing*.
2. *Fine Grained Auditing* dengan menggunakan kondisi.

Untuk point 1 diatas dapat dilakukan dengan alternatif yaitu menggunakan fitur yang dimiliki oleh DBMS MySQL dan DBMS PostgreSQL yaitu dengan trigger. Sedangakan untuk point 2 yaitu audit *fga* dengan menggunakan kondisi tertentu hal tersebut belum dapat dilakukan karna adanya perbedaan tipe data antara DBMS Oracle 10g dengan DBMS MySQL dan PostgreSQL.

Bab IV Perancangan

IV.1 Rancangan *Database*

Database yang digunakan dalam pengujian adalah database penjualan yang dirancang dan dibuat oleh penulis. Dalam database ini terdapat 2 tabel yang saling berelasi yaitu pelanggan dan barang. Database penjualan ini bertujuan untuk mencatat data pelanggan yang membeli barang. Database digambarkan dalam diagram ER pada gambar IV.1.



Gambar IV.1 ERD Database Penjualan

User database ini diasumsikan ada 8 orang yang dapat dilihat pada tabel IV.1

Tabel IV.1 Tabel user

Nama User	**Hak Akses**
Admin_pel	Hak akses dapat membuat table, dapat melakukan insert, update, delete dan select pada tabel yang dibuat admin_pel itu sendiri yaitu tabel pelanggan
Iwan	Hak akses hanya pada tabel pelanggan yaitu (insert, update, delete dan select)
Prima	Hak akses hanya pada tabel pelanggan yaitu (insert, update, delete dan select)
Fitra	Hak akses hanya pada tabel pelanggan yaitu (insert, update, delete dan select)
Admin_bar	Hak akses dapat membuat tabel, dapat melakukan insert, update, delete dan select pada tabel yang dibuat admin_pel itu

	sendiri yaitu tabel barang
Arman	Hak akses hanya pada tabel barang yaitu (insert, update, delete dan select)
Andra	Hak akses hanya pada tabel barang yaitu (insert, update, delete dan select)
Wira	Hak akses hanya pada tabel barang yaitu (insert, update, delete dan select)

IV.2 Rancangan Tabel Untuk Menampung Hasil Audit

Rancangan tabel audit ini akan dilakukan dan diimplementasikan pada setiap DBMS berikut ini adalah rancangan tabel yang akan diimplementasikan pada setiap DBMS.

IV.2.1 Rancangan Tabel Audit Untuk Tabel Pelanggan

Rancangan yang akan dibuat pada tabel pelanggan untuk menampung hasil audit adalah sebagai berikut.

1. Tabel ins_del_pelanggan berfungsi untuk menampung data-data yang di masukkan dan di delete user pada tabel pelanggan. Deskripsi tabel ins_del_pelanggan dapat dilihat pada tabel IV.2.

Tabel IV.2 Tabel ins_del_pelanggan

Nama_field	Type	Constraint
User_id	Varchar2(30)	Null
Action	Varchar2(15)	Null
Action_date	Varchar2(30)	Null
Id_pelanggan	Integer	Null
Nama_pelanggan	Varchar2(30)	Null
Email	Varchar2(30)	Null
Alamat	Varchar2(30)	Null
Notelp	Varchar2(15)	Null

2. Tabel update_pelanggan berfungsi untuk menampung data lama sebelum di update dan menampung data baru setelah di update berdasarkan field yang di rubah oleh user pada tabel pelanggan. Deskripsi tabel update_pelanggan dapat dilihat pada tabel IV.3

Tabel IV.3 Tabel Update_pelanggan

Field	Type	Constraint
User_id	Varchar2(30)	Null
Action	Varchar2(15)	Null
Action_date	Varchar2(30)	Null
Nama_field	Varchar2(20)	Null
Data_lama	Varchar2(30)	Null
Data_baru	Varchar2(30)	Null

IV.2.2 Rancangan Tabel Audit Untuk Tabel Barang

Rancangan yang akan dibuat pada tabel barang untuk menampung hasil audit adalah sebagai berikut.

1. Tabel ins_del_barang berfungsi untuk menampung data-data yang di masukkan dan di delete user pada tabel barang. Deskripsi tabel ins_del_barang dapat dilihat pada tabel IV.4

Tabel IV.4 Tabel ins_del_barang

Field	Type	Constraint
User_id	Varchar2(30)	Null
Action	Varchar2(15)	Null
Action_date	Varchar2(30)	Null
Id_barang	Integer	Null
Nama_barang	Varchar2(30)	Null
Katagori	Varchar2(30)	Null
Harga	Integer	Null

2. Tabel update_barang berfungsi untuk menampung data lama sebelum di update dan menampung data baru setelah di update berdasarkan field yang di rubah oleh user pada tabel barang. Deskripsi tabel update_barang dapat dilihat pada tabel IV.5

Tabel IV.5 Tabel Update_barang

Field	Type	Constraint
User_id	Varchar2(30)	Null
Action	Varchar2(15)	Null
Action_date	Varchar2(30)	Null
Nama_field	Varchar2(20)	Null
Data_lama	Varchar2(30)	Null
Data_baru	Varchar2(30)	Null

IV.3 Rancangan *Trigger* Audit Pada DBMS Oracle 10g

Rancangan trigger audit DBMS Oracle yang akan dilakukan adalah dengan membuat suatu tabel yang berfungsi untuk menampung informasi apa yang dimasukkan, dirubah dan di hapus oleh user. Untuk mengetahui history perubahan data dapat dilakukan dengan cara menyimpan data lama sebelum diupdate. Jenis audit yang seperti ini disebut sebagai Value-Based Auditing (VBA). Untuk membuat jenis audit ini kita bisa menggunakan TRIGGER. Sebelum membuat trigger, yang akan dilakukan adalah membuat tabel yang telah dirancang pada bagian rancangan tabel untuk menampung hasil trigger tersebut. Berikut deskripsi trigger yang akan dibuat.

IV.3.1 Rancangan *Trigger* Audit Pada Tabel Pelanggan

1. Trigger trg_insert_pel

   Trigger ini berfungsi untuk mencatat data-data yang ditambah oleh user pada tabel pelanggan ke tabel ins_del_pelanggan. Aturan trigger dapat dilihat pada tabel IV.5.

Tabel IV.5 Tabel trg_insert_pel

Bagian	Keterangan	Nilai
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	INSERT
trigger body (action)	apa *action* dari trigger	Memasukkan data ke tabel ins_del_pelanggan ketika user melakukan insert data. Data yang akan masuk ke tabel ins_del_pelanggan adalah user yang melakukan insert data, string ‘INSERT’, waktu user melakukan insert, isi dari tabel data pelanggan yang di masukkan user yaitu id_pelanggan, nama_pelanggan, email, alamat dan notelp.

2. Trigger trg_upd_id_pelanggan

   Trigger ini berfungsi untuk mencatat perubahan data, yaitu menyimpan data lama khususnya field id_pelanggan yang ada di tabel pelanggan

sebelum tabel pelanggan mengalami perubahan. Aturan trigger dapat dilihat pada tabel IV.6

Tabel IV.6 Tabel trg_upd_id_pel

Bagian	**Keterangan**	**Nilai**
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	UPDATE
trigger body (action)	apa *action* dari trigger	Memasukkan data ke tabel update_pelanggan ketika user melakukan update data. Data yang akan masuk ke tabel update_pelanggan adalah user yang melakukan update data, string 'UPDATE', waktu user melakukan update, isi dari tabel data pelanggan yang di rubah user yaitu id_pelanggan lama dan id_pelanggan baru.

3. Trigger trg_upd_nama

Trigger ini berfungsi untuk mencatat perubahan data, yaitu menyimpan data lama khususnya field nama_pelanggan yang ada di tabel pelanggan sebelum tabel pelanggan mengalami perubahan. Aturan trigger dapat dilihat pada tabel IV.7

Tabel IV.7 Tabel trg_upd_nama

Bagian	**Keterangan**	**Nilai**
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	UPDATE
trigger body	apa *action* dari trigger	Memasukkan data ke tabel update_pelanggan ketika user melakukan update data. Data yang akan masuk ke tabel update_pelanggan adalah user yang melakukan update data, string 'UPDATE', waktu user melakukan update, isi dari tabel data pelanggan yang di rubah user yaitu nama pelanggan lama dan nama pelanggan

		baru.

4. Trigger trg_upd_email

   Trigger ini berfungsi untuk mencatat perubahan data, yaitu menyimpan data lama khususnya field email yang ada di tabel pelanggan sebelum tabel pelanggan mengalami perubahan. Aturan trigger dapat dilihat pada tabel IV.8

Tabel IV.8 Tabel trg_upd_email

Bagian	**Keterangan**	**Nilai**
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	UPDATE
trigger type	berapa kali body trigger dieksekusi	ROW
trigger body	apa *action* dari trigger	Memasukkan data ke tabel update_pelanggan ketika user melakukan update data. Data yang akan masuk ke tabel update_pelanggan adalah user yang melakukan update data, string 'UPDATE', waktu user melakukan update, isi dari tabel data pelanggan yang di rubah user yaitu email lama dan email baru.

5. Trigger trg_upd_alamat

   Trigger ini berfungsi untuk mencatat perubahan data, yaitu menyimpan data lama khususnya field alamat yang ada di tabel pelanggan sebelum tabel pelanggan mengalami perubahan. Aturan trigger dapat dilihat pada tabel IV.9

Tabel IV.9 Tabel trg_upd_alamat

Bagian	**Keterangan**	**Nilai**
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan	UPDATE

	trigger terpacu	
trigger body	apa *action* dari trigger	Memasukkan data ke tabel update_pelanggan ketika user melakukan update data. Data yang akan masuk ke tabel update_pelanggan adalah user yang melakukan update data, string 'UPDATE', waktu user melakukan update, isi dari tabel data pelanggan yang di rubah user yaitu alamat lama dan alamat baru.

6. Trigger trg_upd_notelp

   Trigger ini berfungsi untuk mencatat perubahan data, yaitu menyimpan data lama khususnya field notelp yang ada di tabel pelanggan sebelum tabel pelanggan mengalami perubahan. Aturan trigger dapat dilihat pada tabel IV.9

Tabel IV.9 Tabel trg_upd_notelp

Bagian	**Keterangan**	**Nilai**
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	UPDATE
trigger body	apa *action* dari trigger	Memasukkan data ke tabel update_pelanggan ketika user melakukan update data. Data yang akan masuk ke tabel update_pelanggan adalah user yang melakukan update data, string 'UPDATE', waktu user melakukan update, isi dari tabel data pelanggan yang di rubah user yaitu notelp lama dan notelp baru.

7. Trigger trg_delete_pelanggan

   Trigger ini berfungsi untuk mencatat data-data yang dihapus oleh user pada tabel pelanggan ke tabel ins_del_pelanggan. Aturan trigger dapat dilihat pada tabel IV.10

Tabel IV.10 Tabel trg_delete_pelanggan

Bagian	Keterangan	Nilai
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	DELETE
trigger body	apa *action* dari trigger	Memasukkan data ke tabel ins_del_pelanggan ketika user melakukan delete data. Data yang akan masuk ke tabel ins_del_pelanggan adalah user yang melakukan delete data, string 'DELETE', waktu user melakukan delete, isi dari tabel data pelanggan yang di hapus user yaitu id_pelanggan, nama_pelanggan, email, alamat dan notelp.

IV.3.2 Rancangan *Trigger* Audit Pada Tabel Barang

1. Trigger trg_insert_bar

   Trigger ini berfungsi untuk mencatat data-data yang ditambah oleh user pada tabel barang ke tabel ins_del_bar. Aturan trigger dapat dilihat pada tabel IV.11

Tabel IV.11 Tabel trg_insert_bar

Bagian	Keterangan	Nilai
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	INSERT
trigger body	apa *action* dari trigger	Memasukkan data ke tabel ins_del_barang ketika user melakukan insert data. Data yang akan masuk ke tabel ins_del_barang adalah user yang melakukan insert data, string 'INSERT', waktu user melakukan insert, isi dari tabel barang yang di masukkan user yaitu id_barang, nama_barang, katagori dan harga.

2. Trigger trg_upd_id_barang

   Trigger ini berfungsi untuk mencatat perubahan data, yaitu menyimpan data lama khususnya field id_barang yang ada di tabel barang sebelum

tabel barang mengalami perubahan. Aturan trigger dapat dilihat pada tabel IV.12

Tabel IV.12 Tabel trg_upd_id_barang

Bagian	Keterangan	Nilai
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	UPDATE
trigger body	apa *action* dari trigger	Memasukkan data ke tabel update_barang ketika user melakukan update data. Data yang akan masuk ke tabel update_barang adalah user yang melakukan update data, string 'UPDATE', waktu user melakukan update, isi dari tabel barang yang di rubah user yaitu id_barang lama dan id_barang baru.

3. Trigger trg_upd_nama_barang

   Trigger ini berfungsi untuk mencatat perubahan data, yaitu menyimpan data lama khususnya field nama_barang yang ada di tabel barang sebelum tabel barang mengalami perubahan. Aturan trigger dapat dilihat pada tabel IV.13.

Tabel IV.13 Tabel trg_upd_nama_barang

Bagian	Keterangan	Nilai
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	UPDATE
trigger body	apa *action* dari trigger	Memasukkan data ke tabel update_barang ketika user melakukan update data. Data yang akan masuk ke tabel update_barang adalah user yang melakukan update data, string 'UPDATE', waktu user melakukan update, isi dari tabel barang yang di rubah user yaitu id_barang lama dan id_barang baru.

4. Trigger trg_upd_katagori

   Trigger ini berfungsi untuk mencatat perubahan data, yaitu menyimpan data lama khususnya field katagori yang ada di tabel barang sebelum tabel barang mengalami perubahan. Aturan trigger dapat dilihat pada tabel IV.14.

Tabel IV.14 Tabel trg_upd_katagori

Bagian	Keterangan	Nilai
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	UPDATE
trigger body	apa *action* dari trigger	Memasukkan data ke tabel update_barang ketika user melakukan update data. Data yang akan masuk ke tabel update_barang adalah user yang melakukan update data, string 'UPDATE', waktu user melakukan update, isi dari tabel barang yang di rubah user yaitu katagori lama dan katagori baru.

5. Trigger trg_upd_harga

   Trigger ini berfungsi untuk mencatat perubahan data, yaitu menyimpan data lama khususnya field harga yang ada di tabel barang sebelum tabel barang mengalami perubahan. Aturan trigger dapat dilihat pada tabel IV.15.

Tabel IV.15 Tabel trg_upd_harga

Bagian	Keterangan	Nilai
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	UPDATE
trigger body	apa *action* dari trigger	Memasukkan data ke tabel update_barang ketika user melakukan update data. Data yang akan masuk ke tabel update_barang adalah user yang

		melakukan update data, string ‘UPDATE’, waktu user melakukan update, isi dari tabel barang yang di rubah user yaitu harga lama dan harga baru

6. Trigger trg_delete_barang

   Trigger ini berfungsi untuk mencatat data-data yang dihapus oleh user pada tabel barang ke tabel ins_del_barang. Aturan trigger dapat dilihat pada tabel IV.16.

Tabel IV.16 Tabel trg_delete_barang

Bagian	**Keterangan**	**Nilai**
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	DELETE
trigger body	apa *action* dari trigger	Memasukkan data ke tabel ins_del_barang ketika user melakukan delete data. Data yang akan masuk ke tabel ins_del_barang adalah user yang melakukan delete data, string ‘DELETE’, waktu user melakukan delete, isi dari tabel data barang yang di hapus user yaitu id_barang, nama_barang, katagori dan harga.

IV.4 Rancangan *Trigger* Audit Pada DBMS MySQL

Rancangan *trigger* audit pada DBMS MySQL yang akan dilakukan adalah dengan membuat suatu tabel yang berfungsi untuk menampung informasi apa yang dimasukkan, dirubah dan di hapus oleh user. Untuk mengetahui history perubahan data dapat dilakukan dengan cara menyimpan data lama sebelum diupdate. Berikut deskripsi trigger yang akan dibuat.

IV.4.1 Rancangan *Trigger* Audit Pada Tabel Pelanggan

1. Trigger *trg_insert_pel*

   Trigger ini berfungsi untuk mencatat data-data yang ditambah oleh user pada tabel pelanggan ke tabel ins_del_pelanggan. Aturan trigger dapat dilihat pada tabel IV.17.

Tabel IV.17 Tabel trg_insert_pel

Bagian	Keterangan	Nilai
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	INSERT
trigger body	apa *action* dari trigger	Memasukkan data ke tabel ins_del_pelanggan ketika user melakukan insert data. Data yang akan masuk ke tabel ins_del_pelanggan adalah user yang melakukan insert data, string 'INSERT', waktu user melakukan insert, isi dari tabel data pelanggan yang di masukkan user yaitu id_pelanggan, nama_pelanggan, email, alamat dan notelp.

2. Trigger trg_upd_pelanggan

   Trigger ini berfungsi untuk mencatat perubahan data, yaitu menyimpan data-data lama pada semua field yang ada di tabel pelanggan sebelum tabel pelanggan mengalami perubahan. Aturan trigger dapat dilihat pada tabel IV.18.

Tabel IV.18 Tabel trg_upd_pelanggan

Bagian	Keterangan	Nilai
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	UPDATE
trigger body	apa *action* dari trigger	Jika id_pelanggan lama Tidak sama dengan id_pelanggan baru maka Data akan dimasukkan data ke tabel update_pelanggan ketika user melakukan update data. Data yang akan masuk ke tabel update_pelanggan adalah user yang melakukan update data, string 'UPDATE', waktu user

		melakukan update, string ‘ID_PELANGGAN’, isi dari tabel data pelanggan yang di rubah user yaitu id_pelanggan lama dan id_pelanggan baru. Jika nama_pelanggan lama Tidak sama dengan nama_pelanggan baru maka Data akan dimasukkan ke tabel update_barang ketika user melakukan update data. Data yang akan masuk ke tabel update_barang adalah user yang melakukan update data, string ‘UPDATE’, waktu user melakukan update, string ‘NAMA_PELANGGAN’, isi dari tabel barang yang di rubah user yaitu id_barang lama dan id_barang baru. Jika email lama tidak sama dengan email baru maka Data akan dimasukkan ke tabel update_pelanggan ketika user melakukan update data. Data yang akan masuk ke tabel update_pelanggan adalah user yang melakukan update data, string ‘UPDATE’, waktu user melakukan update, string ‘EMAIL’, isi dari tabel data pelanggan yang di rubah user yaitu email lama dan email baru. Jika alamat lama tidak sama dengan alamat baru maka Data akan dimasukkan ke tabel update_pelanggan ketika user melakukan update data. Data yang akan masuk ke tabel update_pelanggan adalah user yang melakukan update data, string ‘UPDATE’, waktu user melakukan update, string ‘ALAMAT’, isi dari tabel data pelanggan yang di rubah user yaitu alamat lama dan alamat baru. Jika notelp lama tidak sama dengan notelp baru maka Memasukkan data ke tabel update_pelanggan ketika user melakukan update data. Data yang akan masuk ke tabel update_pelanggan adalah user yang melakukan update data, string ‘UPDATE’, waktu user melakukan update, string ‘NOTELP’, isi dari tabel data pelanggan yang di rubah user yaitu notelp lama dan notelp baru.

3. Trigger trg_delete_pelanggan

   Trigger ini berfungsi untuk mencatat data-data yang dihapus oleh user pada tabel pelanggan ke tabel ins_del_pelanggan. Aturan trigger dapat dilihat pada tabel IV.19.

Tabel IV.19 Tabel trg_delete_pelanggan

Bagian	**Keterangan**	**Nilai**
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data	DELETE

	pada tabel yang menyebabkan trigger terpacu	
trigger body	apa *action* dari trigger	Memasukkan data ke tabel ins_del_pelanggan ketika user melakukan delete data. Data yang akan masuk ke tabel ins_del_pelanggan adalah user yang melakukan delete data, string 'DELETE', waktu user melakukan delete, isi dari tabel data pelanggan yang di hapus user yaitu id_pelanggan, nama_pelanggan, email, alamat dan notelp.

IV.4.2 Rancangan *Trigger* Audit Pada Tabel Barang

1. Trigger trg_insert_bar

   Trigger ini berfungsi untuk mencatat data-data yang ditambah oleh user pada tabel barang ke tabel ins_del_bar. Aturan trigger dapat dilihat pada tabel IV.20.

Tabel IV.20 Tabel trg_insert_bar

Bagian	**Keterangan**	**Nilai**
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	INSERT
trigger body	apa *action* dari trigger	Memasukkan data ke tabel ins_del_barang ketika user melakukan insert data. Data yang akan masuk ke tabel ins_del_barang adalah user yang melakukan insert data, string 'INSERT', waktu user melakukan insert, isi dari tabel barang yang di masukkan user yaitu id_barang, nama_barang, katagori dan harga.

2. Trigger trg_upd_barang

   Trigger ini berfungsi untuk mencatat perubahan data, yaitu menyimpan data-data lama pada semua field yang ada di tabel barang sebelum tabel barang mengalami perubahan. Aturan trigger dapat dilihat pada tabel IV.21.

Tabel IV.21 Tabel trg_upd_barang

Bagian	Keterangan	Nilai
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	UPDATE
trigger body	apa *action* dari trigger	Jika id_barang lama tidak sama dengan id_barang baru maka Data akan dimasukkan ke tabel update_barang ketika user melakukan update data. Data yang akan masuk ke tabel update_barang adalah user yang melakukan update data, string 'UPDATE', waktu user melakukan update, string 'ID_BARANG', isi dari tabel barang yang di rubah user yaitu id_barang lama dan id_barang baru. Jika nama_barang lama tidak sama dengan nama_barang baru maka Data akan dimasukkan ke tabel update_barang ketika user melakukan update data. Data yang akan masuk ke tabel update_barang adalah user yang melakukan update data, string 'UPDATE', waktu user melakukan update, string 'NAMA_BARANG', isi dari tabel barang yang di rubah user yaitu nama_barang lama dan nama_barang baru. Jika katagori lama tidak sama dengan katagori baru maka Data akan dimasukkan ke tabel update_barang ketika user melakukan update data. Data yang akan masuk ke tabel update_barang adalah user yang melakukan update data, string 'UPDATE', waktu user melakukan update, string 'KATAGORI', isi dari tabel barang yang di rubah user yaitu katagori lama dan katagori baru. Jika harga lama tidak sama dengan harga baru maka Data akan dimasukkan ke tabel update_barang ketika user melakukan update data. Data yang akan masuk ke tabel update_barang adalah user yang melakukan update data, string 'UPDATE', waktu user melakukan update, string 'HARGA', isi dari tabel barang yang di rubah user yaitu harga lama dan harga baru.

3. Trigger trg_delete_barang

   Trigger ini berfungsi untuk mencatat data-data yang dihapus oleh user pada tabel barang ke tabel ins_del_barang. Aturan trigger dapat dilihat pada tabel IV.22.

Tabel IV.22 Tabel trg_delete_barang

Bagian	**Keterangan**	**Nilai**
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	DELETE
trigger body	apa *action* dari trigger	Memasukkan data ke tabel ins_del_barang ketika user melakukan delete data. Data yang akan masuk ke tabel ins_del_barang adalah user yang melakukan delete data, string 'DELETE', waktu user melakukan delete, isi dari tabel data barang yang di hapus user yaitu id_barang, nama_barang, katagori dan harga.

IV.5 Rancangan *Trigger* Audit Pada DBMS PostgreSQL

Rancangan trigger audit pada DBMS PostgreSQL yang akan dilakukan adalah dengan dengan membuat suatu tabel yang berfungsi untuk menampung informasi apa yang dimasukkan, dirubah dan di hapus oleh user. Untuk mengetahui history perubahan data dapat dilakukan dengan cara menyimpan data lama sebelum diupdate. Dalam Perancangan trigger audit pada postgreSQL ini menggunakan suatu fungsi untuk mengembalikan nilai yang telah kita tentukan. Berikut deskripsi fungsi dan trigger yang akan dibuat.

IV.5.1 Rancangan *Function* Untuk *Trigger*

Perancangan trigger audit pada postgreSQL menggunakan suatu fungsi untuk mengembalikan nilai yang telah kita tentukan. Masing-masing trigger memakai satu fungsi yang berisi statemen untuk melakukan pencatatan ke tabel yang telah di tentukan. Berikut fungsi yang akan di buat:

IV.5.1.1 Fungsi pelanggan_ins_trigger

Fungsi ini akan di gunakan pada saat menggunakan trigger trg_insert_pel. Berikut pembuatan fungsi pelanggan_ins_trigger()

```
CREATE OR REPLACE FUNCTION pelanggan_ins_trigger()
RETURNS TRIGGER AS $$
BEGIN
INSERT INTO ins_del_pelanggan
 (user_id, action, action_date , id_pelanggan, nama_pelanggan,
email, alamat, notelp)
VALUES     (current_user,     'INSERT'     ,     current_timestamp,
NEW.Id_pelanggan,   NEW.nama_pelanggan,   NEW.email,   NEW.alamat,
NEW.notelp);
RETURN NULL;
END;
$$
LANGUAGE plpgsql;
```

IV.5.1.2 Fungsi pelanggan_upd_trigger

Fungsi ini akan di gunakan pada saat menggunakan trigger trg_upd_pelanggan. Berikut pembuatan fungsi pelanggan_upd_trigger()

```
CREATE OR REPLACE FUNCTION pelanggan_upd_trigger()
RETURNS TRIGGER AS $$
BEGIN

IF (OLD.id_pelanggan<>NEW.id_pelanggan) THEN
INSERT INTO UPDATE_PELANGGAN
(User_id, action, action_date, nama_field, data_lama, data_baru)
VALUES(current_user, 'UPDATE' , current_timestamp, 'ID_PELANGGAN',
OLD.id_pelanggan, NEW.id_pelanggan);
END IF;

IF (OLD.nama_pelanggan<>NEW.nama_pelanggan) THEN
INSERT INTO UPDATE_PELANGGAN
(User_id, action, action_date, nama_field, data_lama, data_baru)
VALUES(current_user, 'UPDATE', current_timestamp,'NAMA_PELANGGAN',
OLD.nama, NEW.nama);
END IF;

IF (OLD.email<>NEW.email) THEN
INSERT INTO UPDATE_PELANGGAN
(User_id, action, action_date, nama_field, data_lama, data_baru)
VALUES (current_user, 'UPDATE' , current_timestamp, 'EMAIL',
OLD.email, NEW.email);
END IF;

IF (OLD.alamat<>NEW.alamat) THEN
INSERT INTO UPDATE_PELANGGAN
(User_id, action, action_date, nama_field, data_lama, data_baru)
```

```
VALUES (current_user, 'UPDATE' , current_timestamp, 'ALAMAT',
OLD.alamat, NEW.alamat);
END IF;

IF (OLD.notelp<>NEW. notelp) THEN
INSERT INTO UPDATE_PELANGGAN
(User_id, action, action_date, nama_field, data_lama, data_baru)
VALUES (current_user, 'UPDATE' , current_timestamp, ' NOTELP',
OLD. notelp, NEW. notelp);
END IF;

RETURN NULL;
END;
$$
LANGUAGE plpgsql;
```

IV.5.1.3 Fungsi pelanggan_del_trigger

Fungsi ini akan di gunakan pada saat menggunakan trigger trg_delete_pelanggan.

Berikut pembuatan fungsi pelanggan_del_trigger()

```
CREATE OR REPLACE FUNCTION pelanggan_del_trigger()
RETURNS TRIGGER AS $$
BEGIN
INSERT INTO ins_del_pelanggan
 (user_id, action, action_date , id_pelanggan, nama_pelanggan,
email, alamat, notelp)
VALUES     (current_user,     'DELETE'     ,     current_timestamp,
OLD.Id_pelanggan,   OLD.nama_pelanggan,   OLD.email,   OLD.alamat,
OLD.notelp);
RETURN NULL;
END;
$$
LANGUAGE plpgsql;
```

IV.5.1.4 Fungsi barang_ins_trigger

Fungsi ini akan di gunakan pada saat menggunakan trigger trg_insert_barang.

Berikut pembuatan fungsi barang_ins_trigger()

```
CREATE OR REPLACE FUNCTION barang_ins_trigger()
RETURNS TRIGGER AS $$
BEGIN
INSERT INTO ins_del_barang
 (user_id, action, action_date , id_barang, nama_barang, katagori,
harga)
VALUES (current_user, 'INSERT' , current_timestamp, NEW.Id_barang,
NEW.nama_barang, NEW.katagori, NEW.harga);
RETURN NULL;
END;
$$
LANGUAGE plpgsql;
```

IV.5.1.5 Fungsi barang_upd_trigger

Fungsi ini akan di gunakan pada saat menggunakan trigger trg_upd_barang. Berikut pembuatan fungsi barang_upd_trigger()

```
CREATE OR REPLACE FUNCTION barang_upd_trigger()
RETURNS TRIGGER AS $$
BEGIN

IF (OLD.id_barang <>NEW.id_barang) THEN
INSERT INTO UPDATE_BARANG
(User_id, action, action_date, nama_field, data_lama, data_baru)
VALUES (current_user, 'UPDATE' , current_timestamp, 'ID_BARANG',
OLD.id_barang, NEW.id_barang);
END IF;

IF (OLD.nama_barang <>NEW.nama_barang) THEN
INSERT INTO UPDATE_BARANG
(User_id, action, action_date, nama_field, data_lama, data_baru)
VALUES (current_user,'UPDATE' , current_timestamp,'NAMA_BARANG',
OLD.nama_barang, NEW.nama_barang);
END IF;
IF (OLD.katagori<>NEW.katagori) THEN
INSERT INTO UPDATE_BARANG
(User_id, action, action_date, nama_field, data_lama, data_baru)
VALUES (current_user, 'UPDATE' , current_timestamp, 'KATAGORI',
OLD.katagori, NEW.katagori);
END IF;

IF (OLD.harga<>NEW.harga) THEN
INSERT INTO UPDATE_BARANG
(User_id, action, action_date, nama_field, data_lama, data_baru)
VALUES (current_user, 'UPDATE', current_timestamp, 'HARGA',
OLD.Harga, NEW.Harga);
END IF;
RETURN NULL;
END;
$$
LANGUAGE plpgsql;
```

IV.5.1.6 Fungsi barang_del_trigger

Fungsi ini akan di gunakan pada saat menggunakan trigger trg_delete_barang. Berikut pembuatan fungsi barang_del_trigger()

```
CREATE OR REPLACE FUNCTION barang_del_trigger()
RETURNS TRIGGER AS $$
BEGIN
INSERT INTO ins_del_barang
 (user_id, action, action_date , id_barang, nama_barang, katagori,
harga)
VALUES (current_user, 'DELETE' , current_timestamp, OLD.Id_barang,
OLD.nama_barang, OLD.katagori, OLD.harga);
RETURN NULL;
END;
$$
```

```
LANGUAGE plpgsql;
```

IV.5.2 Rancangan *Trigger* Audit Pada Tabel Pelanggan

1. Trigger trg_insert_pelanggan

   Trigger ini berfungsi untuk mencatat data-data yang ditambah oleh user pada tabel pelanggan ke tabel ins_del_pelanggan. Aturan trigger dapat dilihat pada tabel IV.23.

Tabel IV.23 Tabel trg_insert_pelanggan

Bagian	Keterangan	Nilai
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	INSERT
trigger body	apa *action* dari trigger	EXECUTE PROCEDURE pelanggan_ins_trigger();

2. Trigger trg_upd_pelanggan

   Trigger ini berfungsi untuk mencatat perubahan data, yaitu menyimpan data-data lama pada semua field yang ada di tabel pelanggan sebelum tabel pelanggan mengalami perubahan. Aturan trigger dapat dilihat pada tabel IV.24.

Tabel IV.24 Tabel trg_upd_pelanggan

Bagian	Keterangan	Nilai
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	UPDATE
trigger body	apa *action* dari trigger	EXECUTE PROCEDURE pelanggan_upd_trigger();

3. Trigger trg_delete_pelanggan

   Trigger ini berfungsi untuk mencatat data-data yang dihapus oleh user pada tabel pelanggan ke tabel ins_del_pelanggan. Aturan trigger dapat dilihat pada tabel IV.25.

Tabel IV.25 Tabel trg_delete_pelanggan

Bagian	Keterangan	Nilai
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	DELETE
trigger body	apa *action* dari trigger	EXECUTE PROCEDURE pelanggan_del_trigger();

IV.5.3 Rancangan *Trigger* Audit Pada Tabel Barang

1. Trigger trg_insert_barang

   Trigger ini berfungsi untuk mencatat data-data yang ditambah oleh user pada tabel barang ke tabel ins_del_barang. Aturan trigger dapat dilihat pada tabel IV.26.

Tabel IV.26 Tabel trg_insert_barang

Bagian	Keterangan	Nilai
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	INSERT
trigger body	apa *action* dari trigger	EXECUTE PROCEDURE barang_ins_trigger();

2. Trigger trg_upd_barang

   Trigger ini berfungsi untuk mencatat perubahan data, yaitu menyimpan data-data lama pada semua field yang ada di tabel barang sebelum tabel barang mengalami perubahan. Aturan trigger dapat dilihat pada tabel IV.27

Tabel IV.27 Tabel trg_upd_barang

Bagian	Keterangan	Nilai
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang	UPDATE

	menyebabkan trigger terpacu	
trigger body	apa *action* dari trigger	EXECUTE PROCEDURE barang_upd_trigger();

3. Trigger trg_delete_barang

   Trigger ini berfungsi untuk mencatat data-data yang dihapus oleh user pada tabel barang ke tabel ins_del_barang. Aturan trigger dapat dilihat pada tabel IV.28.

Tabel IV.28 Tabel trg_delete_barang

Bagian	**Keterangan**	**Nilai**
trigger timing	ketika trigger berelasi ke event	AFTER
trigger event	manipulasi data pada tabel yang menyebabkan trigger terpacu	DELETE
trigger body	apa *action* dari trigger	EXECUTE PROCEDURE barang_del_trigger();

Bab V Implementasi dan Pengujian

V.1 Implementasi

V.1.1 Implementasi Rancangan *Database* Dan Tabel Audit Pelanggan Dan Tabel Audit Barang

Berikut ini implementasi rancangan database dan implentasi tabel audit pelanggan dan tabel audit barang.

Tabel V.1 Implementasi Tabel

Nama Trigger	Fungsi	Scripts
pelanggan	Tempat menyimpan data-data pelanggan.	`CREATE TABLE pelanggan ( id_pelanggan INTEGER NOT NULL, nama_pelanggan VARCHAR2 (20), email VARCHAR2 (20), alamat VARCHAR2 (20), notelp VARCHAR2 (15), CONSTRAINT pk_pelanggan PRIMARY KEY (id_pelanggan) );`
barang	Tempat menyimpan data-data barang .	`CREATE TABLE barang ( id_barang INTEGER NOT NULL, nama_barang VARCHAR2 (20), katagori VARCHAR2 (20), harga integer, CONSTRAINT pk_barang primary key (id_barang) );`
ins_del_pelanggan	Menyimpan data-data hasil insert dan delete pada tabel pelanggan.	`CREATE TABLE INS_DEL_PELANGGAN( user_id varchar2 (30), action varchar2 (15), action_date varchar2 (30), id_pelanggan integer, nama_pelanggan varchar2 (30), email varchar2 (30), alamat varchar2 (30), notelp varchar2 (15));`

Nama Trigger	Fungsi	Scripts
update_pelanggan	Menyimpan data-data hasil update pada tabel pelanggan.	CREATE TABLE UPDATE_PELANGGAN(user_id VARCHAR2 (30), action VARCHAR2 (15), action_date VARCHAR2 (30), nama_field VARCHAR2 (20), data_lama VARCHAR2 (30), data_baru VARCHAR2 (30));
ins_del_barang	Menyimpan data-data hasil insert dan delete pada tabel barang.	CREATE TABLE INS_DEL_BARANG (user_id varchar2 (30), action varchar2 (15), action_date varchar2 (30), id_barang integer, nama_barang varchar2 (30), katagori varchar2 (30), harga integer);
update_barang	Menyimpan data-data hasil update pada tabel barang.	CREATE TABLE UPDATE_BARANG (user_id VARCHAR2 (30), action VARCHAR2 (15), action_date VARCHAR2 (30), nama_field VARCHAR2 (20), data_lama VARCHAR2 (30), data_baru VARCHAR2 (30));

V.1.2 Implementasi *Trigger* Audit Pada DBMS Oracle 10g

Berikut ini implementasi rancangan trigger pada tabel pelanggan dan trigger pada tabel barang.

Tabel V.2 Implementasi Tabel

Nama Trigger	Fungsi	Scripts
trg_insert_pel	Memasukkan data ke tabel ins_del_pelanggan ketika user melakukan insert data pada tabel pelanggan.	CREATE OR REPLACE TRIGGER trg_insert_pel AFTER INSERT ON ADMIN_PEL.PELANGGAN FOR EACH ROW BEGIN INSERT INTO ins_del_pelanggan (user_id,action,action_date, id_pelanggan,nama_pelanggan,email,alamat,notelp) VALUES (USER, 'INSERT', TO_CHAR (SYSDATE,'DD-MON-YYYY HH24:MI:SS'), :NEW.id_pelanggan, :NEW.nama_pelanggan,:NEW.email,:NEW.alamat, :NEW.notelp); END;

		/
trg_upd_id_pelanggan	Memasukkan data ke tabel update_pelanggan ketika user melakukan update data pada tabel pelanggan.	CREATE OR REPLACE TRIGGER trg_upd_id_pelanggan AFTER UPDATE OF ID_PELANGGAN ON ADMIN_PEL.PELANGGAN FOR EACH ROW BEGIN INSERT INTO UPDATE_PELANGGAN VALUES (USER,'UPDATE',TO_CHAR (SYSDATE,'DD-MON-YYYY HH24:MI:SS'),'ID_PELANGGAN', :OLD.ID_PELANGGAN, :NEW.ID_PELANGGAN); END;/
trg_upd_nama	Memasukkan data ke tabel update_pelanggan ketika user melakukan update data yaitu pada field nama pada tabel pelanggan.	CREATE OR REPLACE TRIGGER trg_upd_nama AFTER UPDATE OF NAMA_PELANGGAN ON ADMIN_PEL.PELANGGAN FOR EACH ROW BEGIN INSERT INTO UPDATE_PELANGGAN VALUES (USER,'UPDATE',TO_CHAR (SYSDATE,'DD-MON-YYYY HH24:MI:SS'),'NAMA_PELANGGAN', :OLD.NAMA_PELANGGAN, :NEW.NAMA_PELANGGAN); END; /
trg_upd_email	Memasukkan data ke tabel update_pelanggan ketika user melakukan update data yaitu pada field email pada tabel pelanggan.	CREATE OR REPLACE TRIGGER trg_upd_email AFTER UPDATE OF EMAIL ON ADMIN_PEL.PELANGGAN FOR EACH ROW BEGIN INSERT INTO UPDATE_PELANGGAN

		VALUES (USER,'UPDATE',TO_CHAR (SYSDATE,'DD-MON-YYYY HH24:MI:SS'),'EMAIL',:OLD .EMAIL,:NEW.EMAIL); END; /
trg_upd_alamat	Memasukkan data ke tabel update_pelanggan ketika user melakukan update data yaitu pada field alamat pada tabel pelanggan.	CREATE OR REPLACE TRIGGER trg_upd_alamat AFTER UPDATE OF ALAMAT ON ADMIN_PEL.PELANGGAN FOR EACH ROW BEGIN INSERT INTO UPDATE_PELANGGAN VALUES (USER,'UPDATE',TO_CHAR (SYSDATE,'DD-MON-YYYY HH24:MI:SS'),'ALAMAT', :OLD.ALAMAT, :NEW.ALAMAT); END; /
trg_upd_notelp	Memasukkan data ke tabel update_pelanggan ketika user melakukan update data yaitu pada field notelp pada tabel pelanggan.	CREATE OR REPLACE TRIGGER trg_upd_notelp AFTER UPDATE OF NOTELP ON ADMIN_PEL.PELANGGAN FOR EACH ROW BEGIN INSERT INTO UPDATE_PELANGGAN VALUES (USER,'UPDATE',TO_CHAR (SYSDATE,'DD-MON-YYYY HH24:MI:SS'),'NOTELP', :OLD.NOTELP, :NEW.NOTELP); END; /
trg_delete_pelanggan	Memasukkan data ke tabel ins_del_pelanggan ketika user melakukan delete data	CREATE OR REPLACE TRIGGER trg_delete_pelanggan AFTER DELETE ON ADMIN_PEL.PELANGGAN FOR EACH ROW

	pada tabel pelanggan.	BEGIN INSERT INTO ins_del_pelanggan (user_id,action,action_date, id_pelanggan,nama_pelanggan,email,alamat,notelp) VALUES (USER, 'DELETE', TO_CHAR (SYSDATE,'DD-MON-YYYY HH24:MI:SS'), :OLD.id_pelanggan, :OLD.nama_pelanggan, :OLD.email, :OLD.alamat, :OLD.notelp); END; /
trg_insert_bar	Memasukkan data ke tabel ins_del_barang ketika user melakukan insert data pada tabel barang.	CREATE OR REPLACE TRIGGER trg_insert_bar AFTER INSERT ON ADMIN_BAR.BARANG FOR EACH ROW BEGIN INSERT INTO ins_del_barang (user_id,action, action_date, id_barang,nama_barang, katagori ,harga) VALUES (USER, 'INSERT', TO_CHAR (SYSDATE,'DD-MON-YYYY HH24:MI:SS'), :NEW.id_barang, :NEW.nama_barang, :NEW.katagori, :NEW.harga); END; /
trg_upd_id_barang	Memasukkan data ke tabel update_barang ketika user melakukan update data yaitu pada field id_barang pada tabel barang.	CREATE OR REPLACE TRIGGER trg_upd_id_barang AFTER UPDATE OF ID_BARANG ON ADMIN_BAR.BARANG FOR EACH ROW BEGIN INSERT INTO UPDATE_BARANG VALUES (USER,'UPDATE',TO_CHAR (SYSDATE,'DD-MON-YYYY HH24:MI:SS'),'ID_BARANG', :OLD.ID_BARANG, :NEW.ID_BARANG); END; /

trg_upd_nama_barang	Memasukkan data ke tabel update_barang ketika user melakukan update data yaitu pada field nama_barang pada tabel barang.	CREATE OR REPLACE TRIGGER trg_upd_nama_barang AFTER UPDATE OF NAMA_BARANG ON ADMIN_BAR.BARANG FOR EACH ROW BEGIN INSERT INTO UPDATE_BARANG VALUES (USER,'UPDATE',TO_CHAR (SYSDATE,'DD-MON-YYYY HH24:MI:SS'),'NAMA_BARANG', :OLD.NAMA_BARANG, :NEW.NAMA_BARANG); END; /
trg_upd_katagori	Memasukkan data ke tabel update_barang ketika user melakukan update data yaitu pada field katagori pada tabel barang.	CREATE OR REPLACE TRIGGER trg_upd_katagori AFTER UPDATE OF KATAGORI ON ADMIN_BAR.BARANG FOR EACH ROW BEGIN INSERT INTO UPDATE_BARANG VALUES (USER,'UPDATE',TO_CHAR (SYSDATE,'DD-MON-YYYY HH24:MI:SS'),'KATAGORI', :OLD.KATAGORI, :NEW.KATAGORI); END; /
trg_upd_harga	Memasukkan data ke tabel update_barang ketika user melakukan update data yaitu pada field harga pada tabel barang.	CREATE OR REPLACE TRIGGER trg_upd_harga AFTER UPDATE OF HARGA ON ADMIN_BAR.BARANG FOR EACH ROW BEGIN INSERT INTO UPDATE_BARANG VALUES (USER,'UPDATE',TO_CHAR

		(SYSDATE,'DD-MON-YYYY HH24:MI:SS'),'HARGA', :OLD.HARGA, :NEW.HARGA); END; /
trg_delete_barang	Memasukkan data ke tabel ins_del_pelanggan ketika user melakukan delete data pada tabel barang.	CREATE OR REPLACE TRIGGER trg_delete_barang AFTER DELETE ON ADMIN_BAR.BARANG FOR EACH ROW BEGIN INSERT INTO ins_del_barang (user_id,action, action_date, id_barang,nama_barang, katagori ,harga) VALUES (USER, 'DELETE', TO_CHAR (SYSDATE,'DD-MON-YYYY HH24:MI:SS'), :OLD.id_barang, :OLD.nama_barang, :OLD.katagori, :OLD.harga); END; /

V.1.3 Implementasi *Trigger* Audit pada DBMS MySQL

Berikut ini implementasi rancangan trigger pada tabel pelanggan dan trigger pada tabel barang

Tabel V.3 Implementasi Tabel

Nama Trigger	Fungsi	Scripts
trg_insert_pel	Memasukkan data ke tabel ins_del_pelanggan ketika user melakukan insert data pada tabel pelanggan.	DELIMITER// CREATE TRIGGER trg_insert_pel AFTER INSERT ON pelanggan FOR EACH ROW BEGIN INSERT INTO ins_del_pelanggan (user_id, action, action_date, id_pelanggan, nama_pelanggan, email, alamat, notelp) VALUES (user(),CONCAT('INSERT'),now(),NEW.Id_pelanggan,NEW.nama_pelanggan, NEW.email, NEW.alamat, NEW.notelp); END; //

trg_upd_pelanggan	Memasukkan data user yang melakukan update ke tabel update_pelanggan	DELIMITER//
		CREATE TRIGGER trg_upd_pelanggan
		AFTER UPDATE ON PELANGGAN
		FOR EACH ROW
		BEGIN
		IF OLD.id_pelanggan<>NEW.id_pelanggan
		THEN INSERT INTO UPDATE_PELANGGAN (User_id, action, action_date, nama_field, data_lama, data_baru)
		VALUES
		(user(), CONCAT ('UPDATE'), now(),CONCAT ('ID_PELANGGAN'), CONCAT(OLD.id_pelanggan), CONCAT (NEW.id_pelanggan));
		END IF;
		
		IF OLD.nama_pelanggan<>NEW.nama_pelanggan
		THEN INSERT INTO UPDATE_PELANGGAN (User_id, action, action_date, nama_field, data_lama, data_baru)
		VALUES
		(user(), CONCAT ('UPDATE'), now(),CONCAT ('NAMA_PELANGGAN'),CONCAT(OLD.nama_pelanggan), CONCAT (NEW.nama_pelanggan));
		END IF;
		
		IF OLD.email<>NEW.email THEN INSERT INTO UPDATE_PELANGGAN (User_id, action, action_date, nama_field, data_lama, data_baru)
		VALUES (user(), CONCAT ('UPDATE'), now(),CONCAT ('EMAIL'), CONCAT(OLD.email), CONCAT (NEW.email));
		END IF;
		
		IF OLD.alamat<>NEW.alamat THEN INSERT INTO UPDATE_PELANGGAN (User_id, action, action_date, nama_field, data_lama, data_baru)
		VALUES
		(user(), CONCAT ('UPDATE'), now(),CONCAT ('ALAMAT'), CONCAT(OLD.alamat), CONCAT (NEW.alamat));
		END IF;

		IF OLD.notelp<>NEW. notelp THEN
		INSERT INTO UPDATE_PELANGGAN
		(User_id, action, action_date, nama_field, data_lama, data_baru)
		VALUES
		(user(), CONCAT ('UPDATE'), now(),CONCAT (' NOTELP'), CONCAT(OLD. notelp), CONCAT (NEW. notelp));
		END IF;
		END;
		//

trg_delete_pelanggan	Memasukkan data ke tabel ins_del_pelanggan ketika user melakukan delete data pada tabel pelanggan.	DELIMITER// CREATE TRIGGER trg_delete_pelanggan AFTER DELETE ON pelanggan FOR EACH ROW BEGIN INSERT INTO ins_del_pelanggan (user_id, action, action_date, id_pelanggan, nama, email, alamat, notelp) VALUES (user(),CONCAT('DELETE'),now(),OLD.Id_pelanggan,OLD.nama_pelanggan, OLD.email, OLD.alamat, OLD.notelp); END; //
trg_insert_bar	Memasukkan data ke tabel ins_del_barang ketika user melakukan insert data pada tabel barang.	DELIMITER // CREATE TRIGGER trg_insert_bar AFTER INSERT ON barang FOR EACH ROW BEGIN INSERT INTO ins_del_barang (user_id, action, action_date, id_barang, nama_barang, katagori, harga) VALUES (user(),CONCAT('INSERT'),now(),NEW.Id_barang,NEW.nama_barang, NEW.katagori, NEW.harga); END; //
trg_upd_barang	Memasukkan data user yang melakukan update ke tabel update_barang.	DELIMITER// CREATE TRIGGER trg_upd_barang AFTER UPDATE ON BARANG FOR EACH ROW BEGIN IF OLD.id_barang<>NEW.id_barang THEN INSERT INTO UPDATE_BARANG (User_id, action, action_date, nama_field, data_lama, data_baru) VALUES

		(user(),CONCAT('UPDATE'),now()
		,CONCAT
		('ID_BARANG'),CONCAT(OLD.id_ba
		rang), CONCAT
		(NEW.id_barang));
		END IF;
		
		IF
		OLD.nama_barang<>NEW.nama_bara
		ng
		THEN INSERT INTO UPDATE_BARANG
		(User_id, action,
		action_date, nama_field,
		data_lama, data_baru)
		VALUES
		(user(),CONCAT('UPDATE'),now()
		,CONCAT ('NAMA_BARANG'),
		CONCAT(OLD.nama_barang),CONCAT
		(NEW.nama_barang));
		END IF;
		
		IF OLD.katagori<>NEW.katagori
		THEN
		INSERT INTO UPDATE_BARANG
		(User_id,action,action_date,na
		ma_field, data_lama,
		data_baru)
		VALUES
		(user(), CONCAT ('UPDATE'),
		now(),CONCAT ('KATAGORI'),
		CONCAT(OLD.katagori), CONCAT
		(NEW.katagori));
		END IF;
		
		IF OLD.harga<>NEW.harga THEN
		INSERT INTO UPDATE_BARANG
		(User_id,action,action_date,na
		ma_field, data_lama,
		data_baru)
		VALUES
		(user(), CONCAT ('UPDATE'),
		now(),CONCAT ('HARGA'),
		CONCAT(OLD.harga), CONCAT
		(NEW.harga));
		END IF;
		END;
		//

trg_delete_barang	Memasukkan data ke tabel ins_del_barang ketika user melakukan delete data pada tabel barang.	DELIMITER// CREATE TRIGGER trg_delete_barang AFTER DELETE ON barang FOR EACH ROW BEGIN INSERT INTO ins_del_barang (user_id,action,action_date,id _barang, nama_barang, katagori, harga) VALUES (user(),CONCAT('DELETE'),now()

		,OLD.Id_barang,OLD.nama_barang , OLD.katagori, OLD.harga); END; //

V.1.4 Implementasi *Trigger* Audit pada DBMS PostgreSQL

Berikut ini implementasi rancangan trigger pada tabel pelanggan dan trigger pada tabel barang

Tabel V.4 Implementasi Tabel

Nama Trigger	Fungsi	Scripts
trg_insert_pelanggan	Mengexecusi fungsi pelanggan_ins_trigger()	CREATE TRIGGER trg_insert_pelanggan AFTER INSERT ON pelanggan FOR EACH ROW EXECUTE PROCEDURE pelanggan_ins_trigger();
trg_upd_pelanggan	Mengexecusi fungsi pelanggan_del_trigger()	CREATE TRIGGER trg_upd_pelanggan AFTER UPDATE ON pelanggan FOR EACH ROW EXECUTE PROCEDURE pelanggan_upd_trigger();
trg_delete_pelanggan	Mengexecusi fungsi pelanggan_upd_trigger()	CREATE TRIGGER trg_delete_pelanggan AFTER DELETE ON pelanggan FOR EACH ROW EXECUTE PROCEDURE barang_del_trigger();
trg_insert_barang	Mengexecusi fungsi pelanggan_ins_trigger()	CREATE TRIGGER trg_insert_barang AFTER INSERT ON barang FOR EACH ROW EXECUTE PROCEDURE barang_ins_trigger();
trg_upd_barang	Mengexecusi fungsi pelanggan_del_trigger()	CREATE TRIGGER trg_upd_barang AFTER UPDATE ON barang FOR EACH ROW EXECUTE PROCEDURE barang_upd_trigger();
trg_delete_barang	Mengexecusi fungsi	CREATE TRIGGER trg_delete_barang AFTER DELETE ON barang

	pelanggan_upd_trigger()	FOR EACH ROW EXECUTE PROCEDURE barang_del_trigger();

V.2 Pengujian

V.2.1 Skenario Pengujian

Skenario pengujian dilakukan pada masing-masing fitur DBMS. Database yang telah dirancang dan dibuat tersebut akan diimplementasikan masing-masing DBMS.

Untuk data yang dijadikan pengujian akan diterapkan dan diimplementasikan pada database yang telah dibuat dan data tersebut merupakan data yang sama untuk masing-masing DBMS. Berikut ini deskripsi data pengujian masing-masing tabel.

V.2.1.1 Skenario Pengujian Pada Tabel Pelanggan

Data yang akan dimasukkan oleh user ke tabel pelanggan dapat di lihat pada tabel V.5

Tabel V.5 Masukkan Data Tabel Pelanggan

User	Melakukan	Id_pelanggan	Nama_pelanggan	Email	Alamat	Notelp
Admin_pel	Insert	1	Takim	Takim@yahoo.com	Patam	085668610105
Iwan	Insert	2	Mardianto	Mardianto@gmail.com	Sei Panas	085668457800
Prima	Insert	3	Rikson	Rikson@yahoo.com	Baloi	085665661987
Fitra	Insert	4	Seli	Seli@gmail.com	Tiban	081365661702

Data yang akan diubah oleh user pada tabel pelanggan dapat di lihat pada tabel V.6

Tabel V.6 Perubahan Data Tabel Pelanggan

User	Melakukan	Field	Data lama	Data baru
Admin_pel	Update	Nama_pelanggan	Takim	Wira dito

Iwan	Update	Nama_pelanggan	Mardianto	Tiro Santoso
Prima	Update	Email	Rikson@yahoo.com	Hendra@gmail.com
Fitra	Update	Alamat	Tiban	Batu aji

Data yang akan dihapus oleh user pada tabel pelanggan dapat di lihat pada tabel V.7

Tabel V.7 Penghapusan Data Pelanggan

User	**Melakukan**	**Id_pelanggan**	**Nama_ Pelanggan**	**Email**	**Alamat**	**Notelp**
Admin_pel	Delete	1	Wira dito	Takim@yahoo.com	Patam	085668610105
Iwan	Delete	2	Tiro Santoso	Mardianto@gmail.com	Sei Panas	085668457800
Prima	Delete	3	Rikson	Hendra@gmail.com	Baloi	085665661987
Fitra	Delete	4	Seli	Seli@gmail.com	Batu aji	081365661702

V.2.1.2 Skenario Pengujian Pada Tabel Barang

Data yang akan dimasukkan oleh user ke tabel barang dapat di lihat pada tabel V.8

Tabel V.8 Masukkan Data Barang

User	**Melakukan**	**Id_barang**	**Nama_barang**	**Katagori**	**Harga**
Admin_bar	Insert	1	Susu bendera	Makanan	4000
Arman	Insert	2	Redbull	Minuman	6000
Andra	Insert	3	Milo	Minuman	21000
Wira	Insert	4	Molto	Sabun	10000

Data yang akan diubah oleh user ke tabel barang dapat di lihat pada tabel V.9

Tabel V.9 Perubahan Data Barang

User	Melakukan	Field	Data lama	Data baru
Admin_bar	Update	Katagori	Makanan	Minuman
Arman	Update	Harga	6000	10000
Andra	Update	Nama_barang	Milo	Coklat
Wira	Update	Id_barang	4	5

Data yang akan dihapus oleh user pada tabel barang dapat di lihat pada tabel V.10

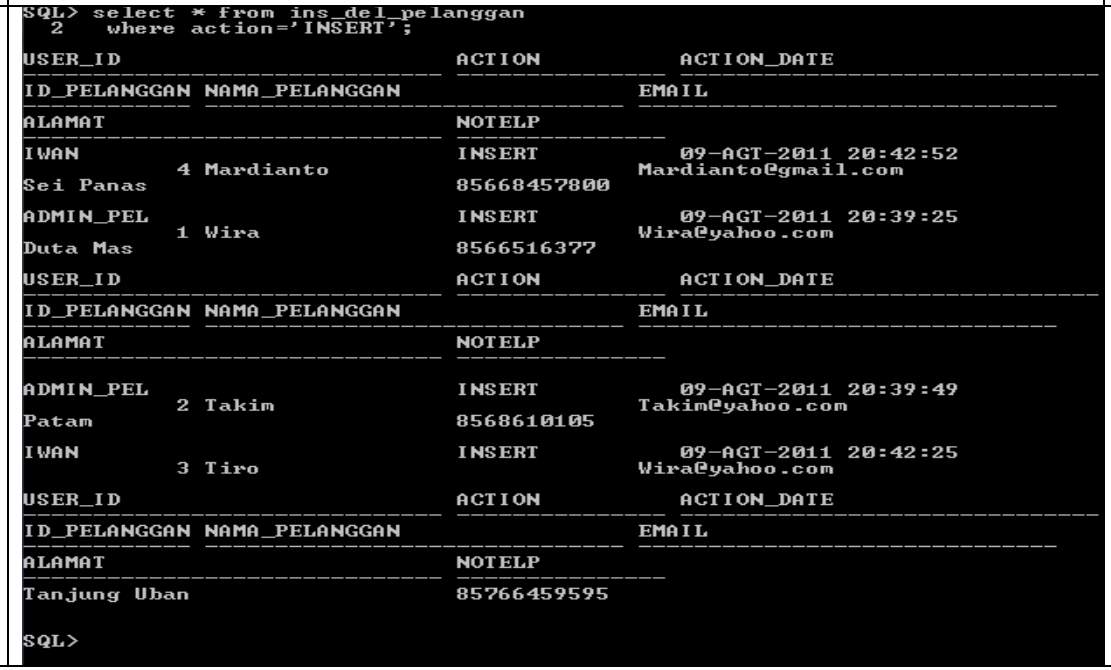
Tabel V.10 Penghapusan Data Barang

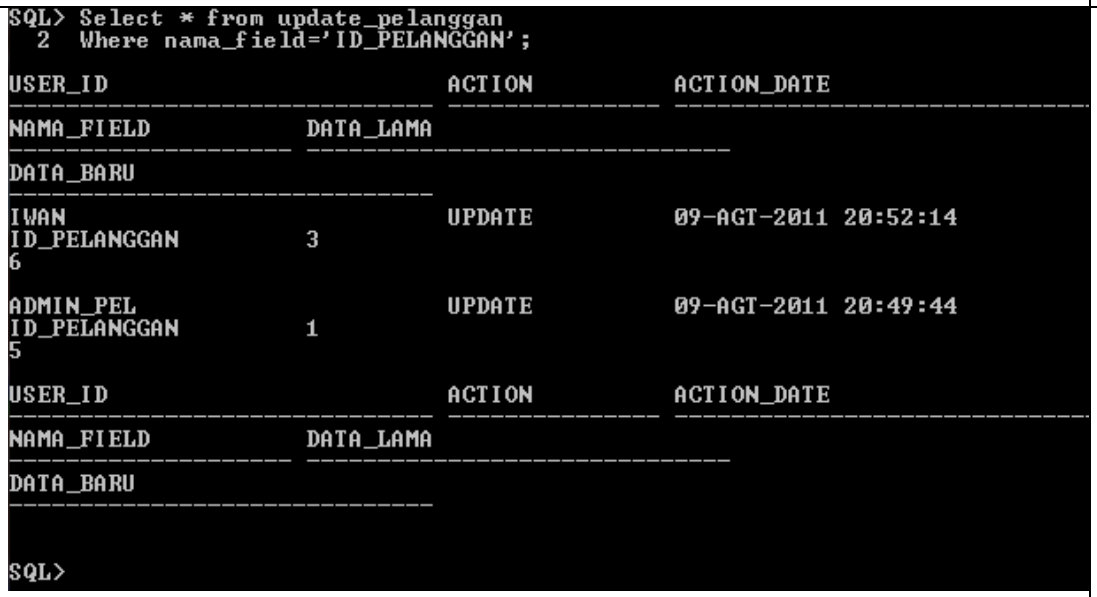
User	Melakukan	Id_barang	Nama_barang	Katagori	Harga
Admin_bar	Delete	1	Susu bendera	Minuman	4000
Arman	Delete	2	Redbull	Minuman	10000
Andra	Delete	3	coklat	Minuman	21000
Wira	Delete	5	Molto	Sabun	10000

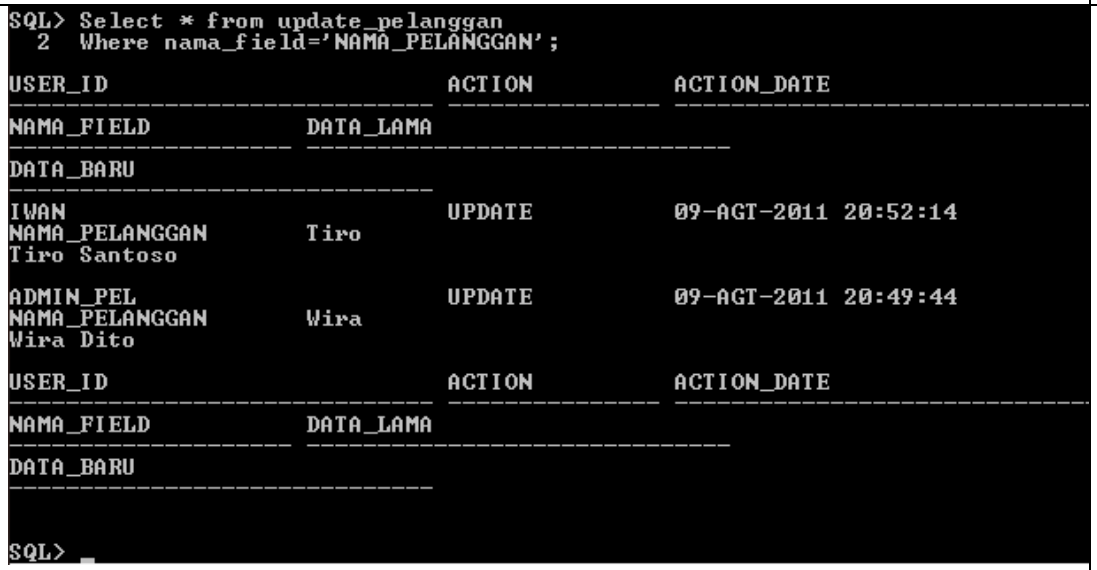
V.2.2 Hasil Pengujian *Trigger* Audit Pada DBMS Oracle 10g

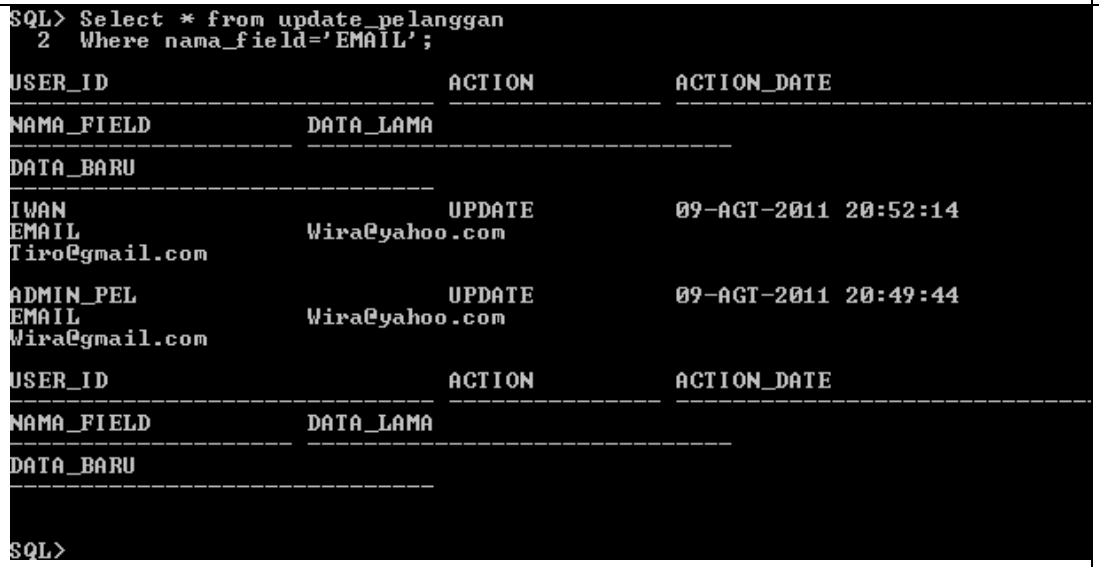
Untuk mendapatkan hasil yang maksimal maka diperlukan suatu pengujian di masing-masing *Trigger* yang berfungsi untuk menampilkan audit yang diterapkan dalam DBMS Oracle 10g, berikut tabel pengujian yang dijelaskan pada tabel V.5

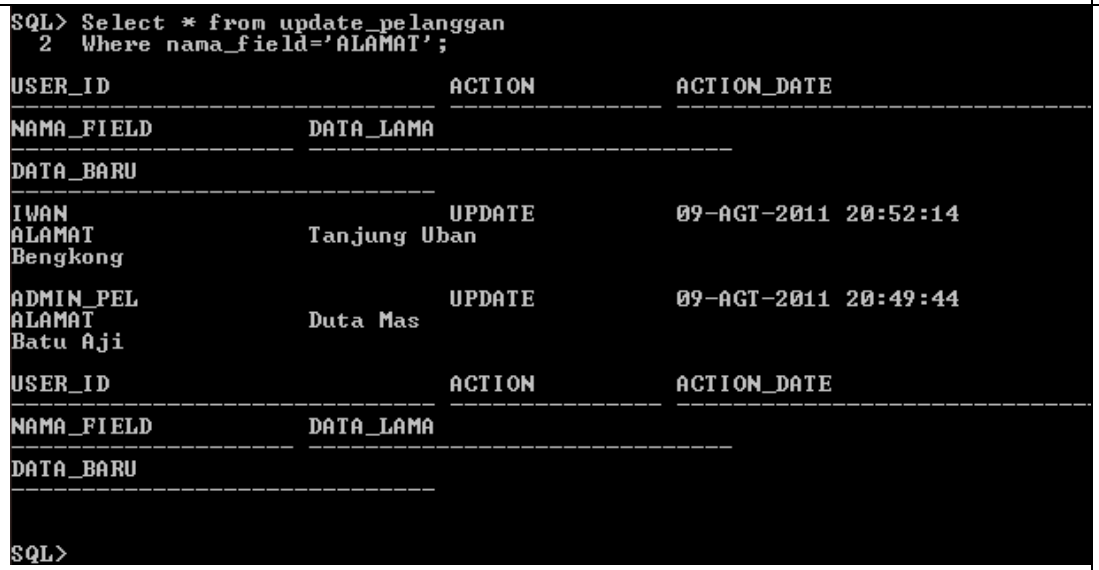
Tabel V.11 Pengujian trigger DBMS Oracle 10g

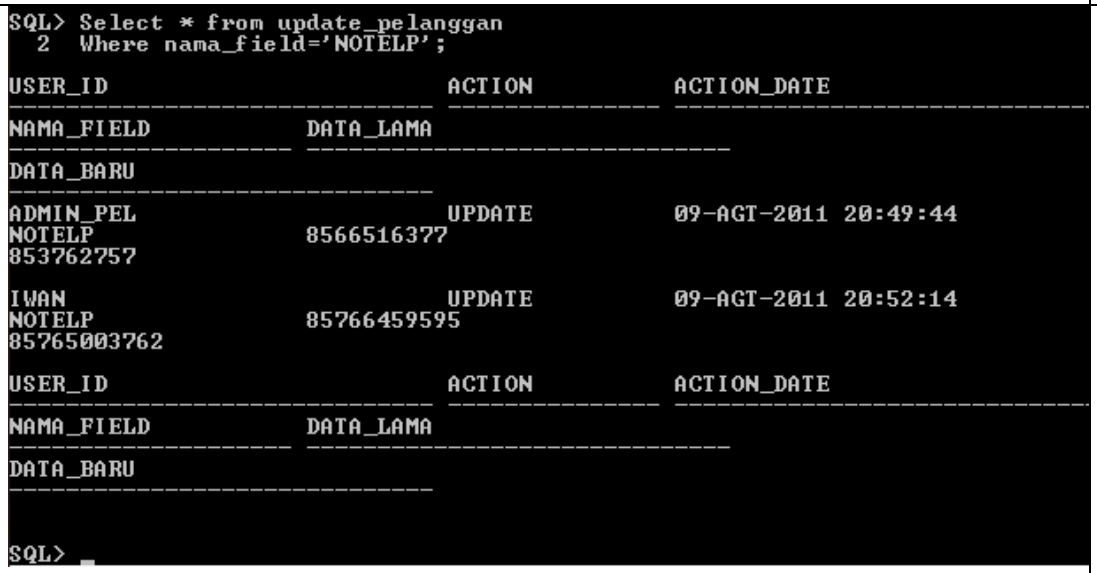
Objek	Nama objek	Hasil uji	Keterangan
Trigger	trg_insert_pel		Sukses

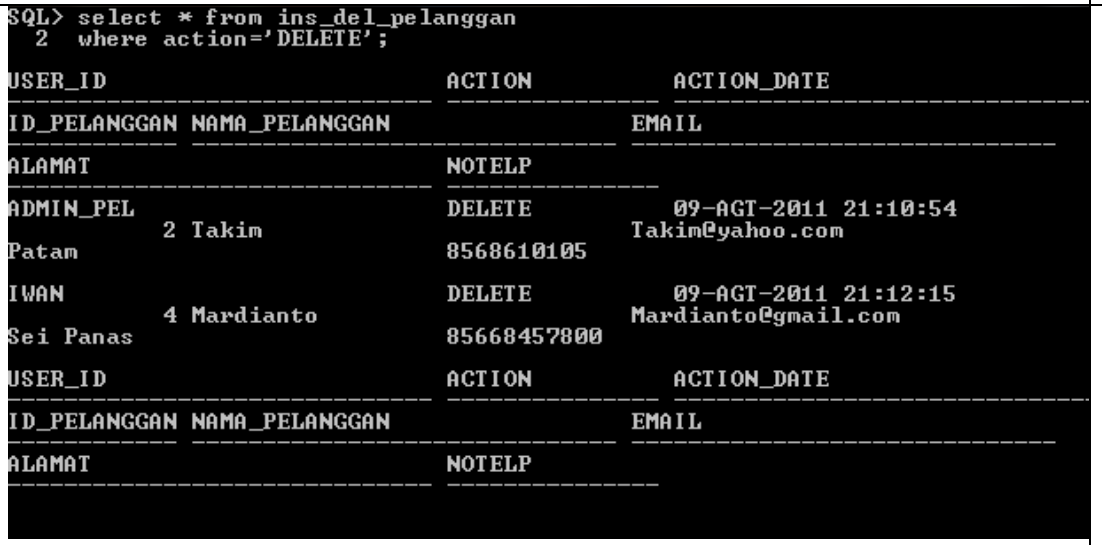
Objek	Nama objek	Hasil uji	Keterangan
	trg_upd_id_pelanggan		Sukses

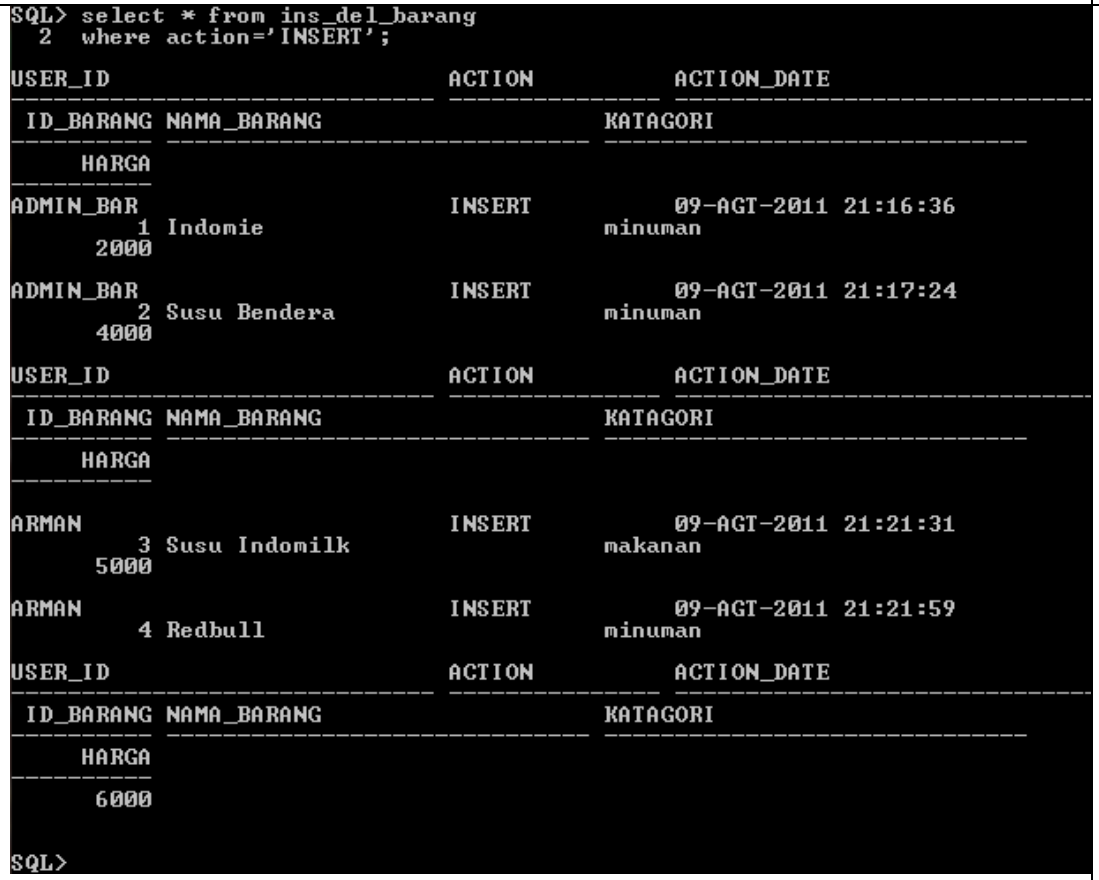
Objek	Nama objek	Hasil uji	Keterangan
	trg_upd_nama		Sukses

Objek	Nama objek	Hasil uji	Keterangan
	trg_upd_email		Sukses

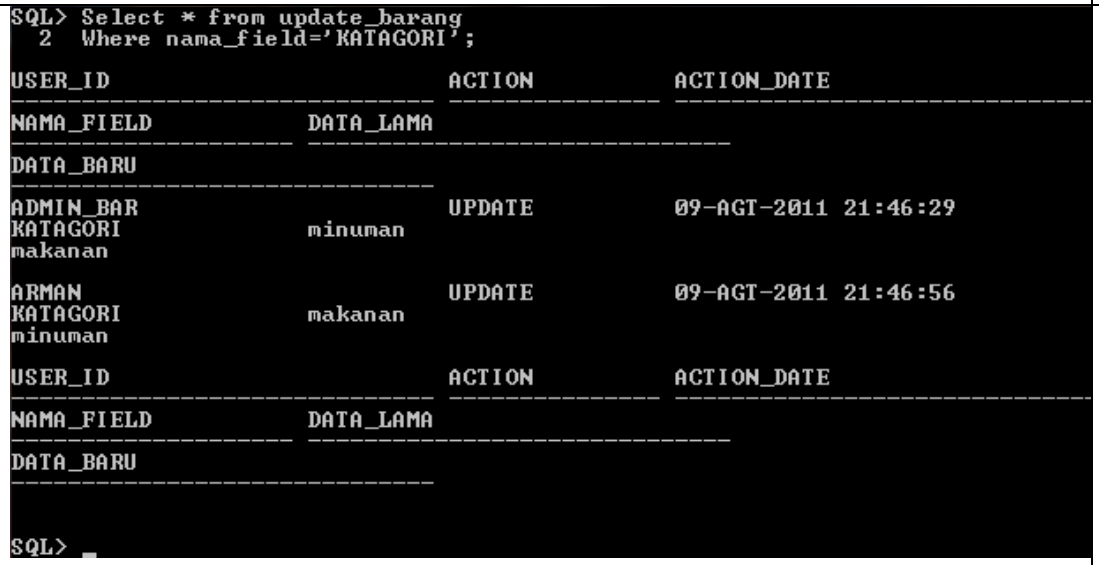
Objek	Nama objek	Hasil uji	Keterangan
	trg_upd_alamat		Sukses

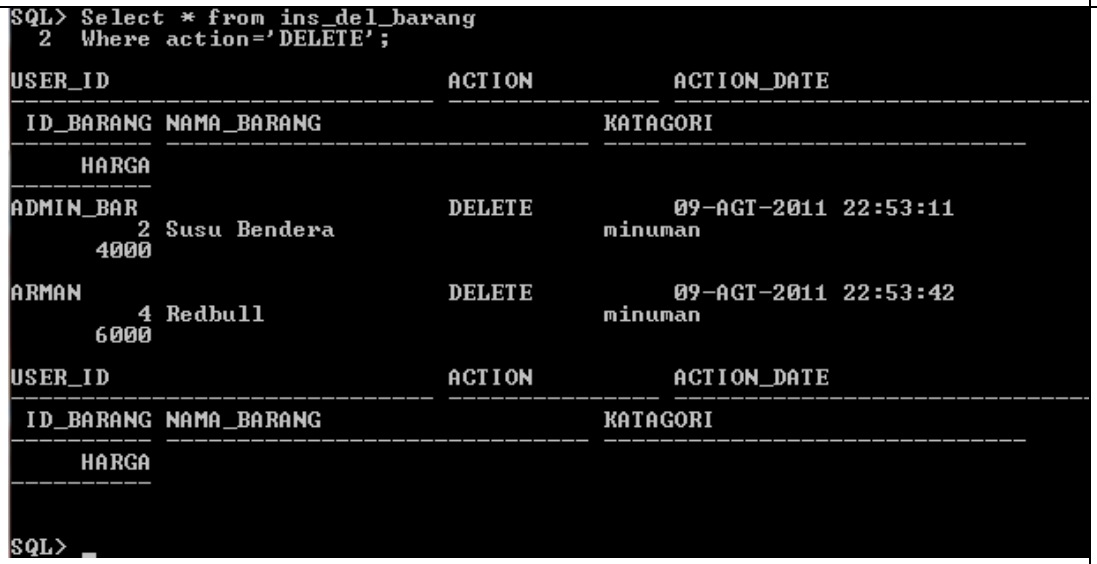
Objek	Nama objek	Hasil uji	Keterangan
	trg_upd_notelp		Sukses

Objek	Nama objek	Hasil uji	Keterangan
	trg_insert_bar		Sukses

Objek	Nama objek	Hasil uji	Keterangan
	trg_upd_id_barang		Sukses

Objek	Nama objek	Hasil uji	Keterangan
	trg_upd_nama_barang		Sukses

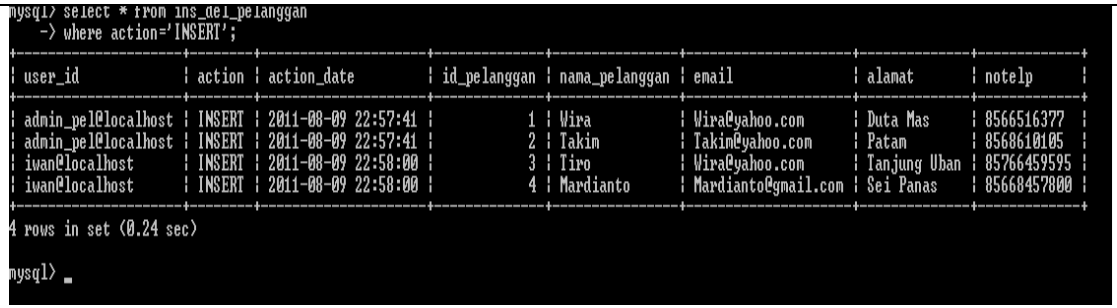
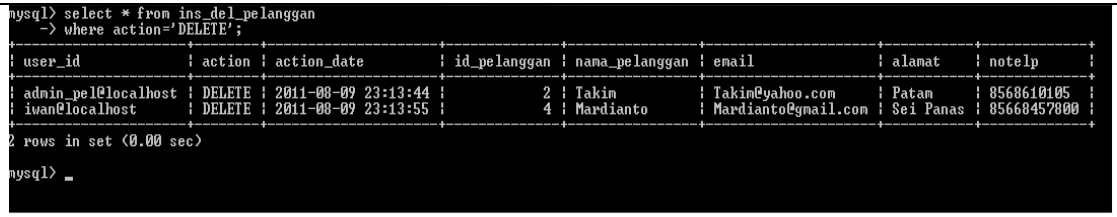
Objek	Nama objek	Hasil uji	Keterangan
	trg_upd_katagori		Sukses

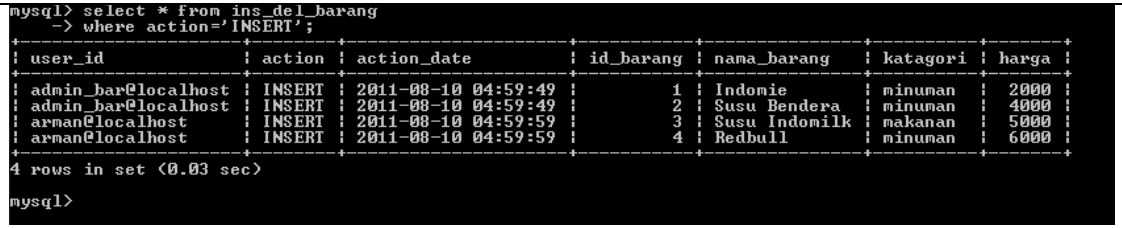
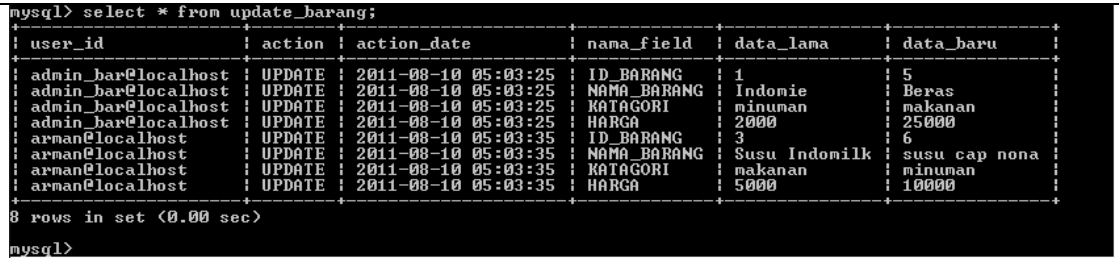
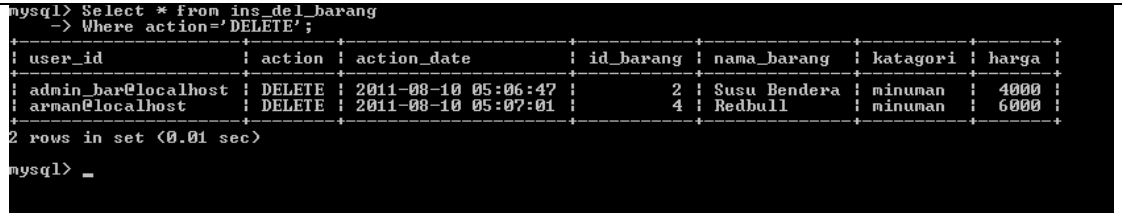
Objek	Nama objek	Hasil uji	Keterangan
	trg_delete_barang		Sukses

V.2.3 Hasil Pengujian *Trigger* Audit pada DBMS MySQL

Untuk mendapatkan hasil yang maksimal maka diperlukan suatu pengujian di masing-masing Trigger yang berfungsi untuk menampilkan audit yang diterapkan dalam DBMS MySQL, berikut tabel pengujian yang dijelaskan pada tabel V.12

Tabel V.12 Pengujian trigger DBMS MySQL

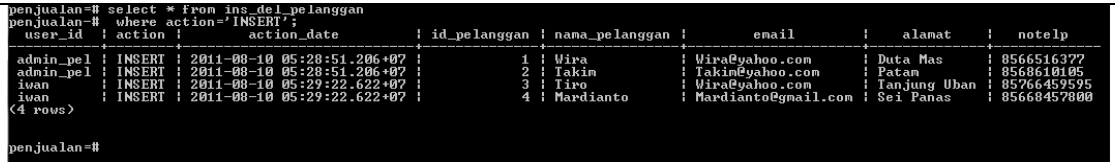
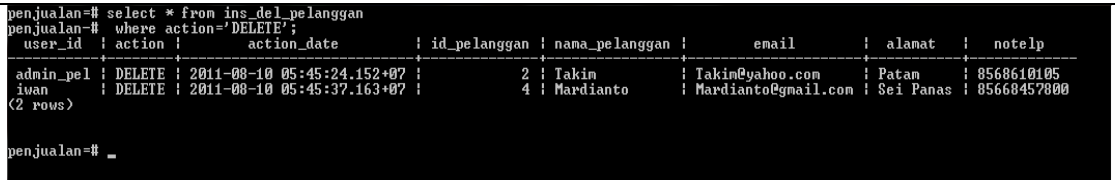
Objek	Nama objek	Hasil uji	Keterangan
Trigger	trg_insert_pel		Sukses
	trg_upd_pelanggan		Sukses
	trg_delete_pelanggan		Sukses

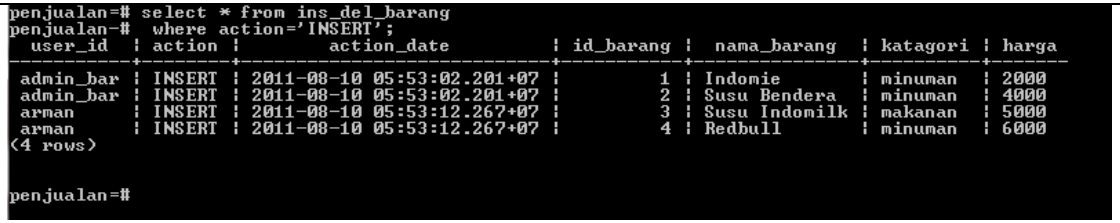
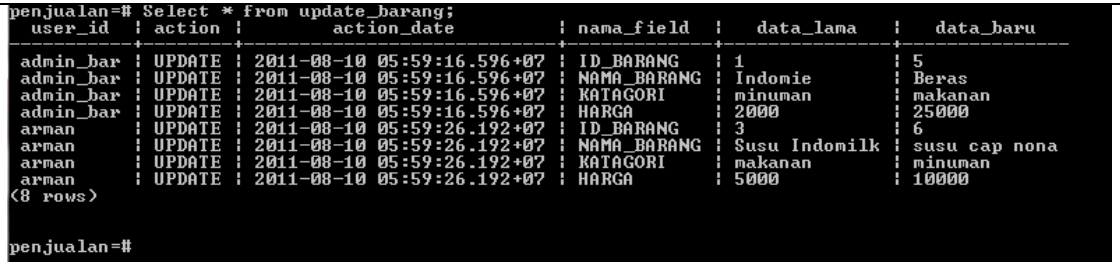
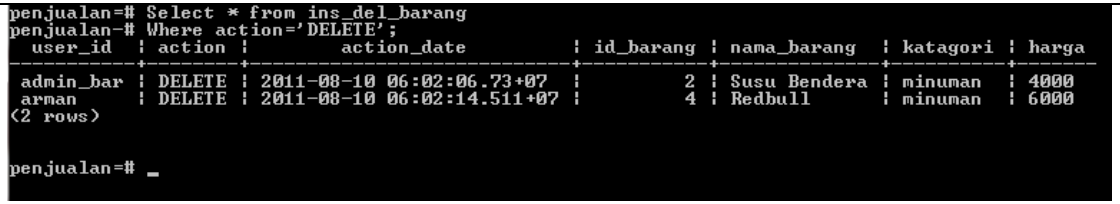
Objek	Nama objek	Hasil uji	Keterangan
	trg_insert_bar		Sukses
	trg_upd_barang		Sukses
	trg_delete_barang		Sukses

V.2.4 Hasil Pengujian *Trigger* Audit Pada DBMS PosgreSQL

Untuk mendapatkan hasil yang maksimal maka diperlukan suatu pengujian di masing-masing Trigger yang berfungsi untuk menampilkan audit yang diterapkan dalam DBMS PostgreSQL, berikut tabel pengujian yang dijelaskan pada tabel V.13

Tabel V.13 Pengujian trigger DBMS PostgreSQL

Objek	Nama objek	Hasil uji	Keterangan
Trigger	trg_insert_pelanggan		Sukses
	trg_upd_pelanggan		Sukses
	trg_delete_pelanggan		Sukses

Objek	Nama objek	Hasil uji	Keterangan
Trigger	trg_insert_barang		Sukses
	trg_upd_barang		Sukses
	trg_delete_barang		Sukses

Bab VI Kesimpulan dan Saran

VI.1 Kesimpulan

Kesimpulan dari Tugas Akhir ini adalah:

1. Audit DBMS MySQL tersebut diterapkan dengan menggunakan:
   - *General log*, dengan mengimpor data dari file text *general log* ke tabel dan untuk mepermudah untuk mencari informasi yang dibutuhkan dengan berbantukan *Query*.
   - *Trigger* digunakan untuk melakukan *value base auditing.*
2. Audit DBMS PostgreSQL tersebut diterapkan dengan menggunakan:
   - *Log postgresql*, hasil dari log tersebut di copy ke sebuah tabel untuk mempermudah dalam menerapkan *audit trail* pada DBMS PostgreSQL.
   - *Trigger* dan *function* digunakan untuk menerapkan *value base auditing.*
3. Untuk *audit trail* jenis audit yang dapat di implementasikan pada DBMS MYSQL dan PostgreSQL adalah pengguna login berhasil, pengguna login gagal, pengguna logout, pengguna proses login dan logout, pengguna yang melakukan perintah DDL, pengguna tertentu yang melakukan perintah DDL, pengguna yang melakukan perintah DML, pengguna tertentu yang melakukan perintah DML, pengguna yang melakukan perintah DCL, pengguna tertentu yang melakukan perintah DCL, statement DDL pada object tertentu, statement DML pada object tertentu, statement DCL pada object tertentu, audit user administrator.

4. Untuk *value base auditing* jenis audit yang dapat di implementasikan pada DBMS MySQL dan PostgreSQL adalah statement *audit insert, update* dan *delete* data berdasarkan perubahan isi dari *database*.

5. Untuk audit *fga* sudah dapat diimplementasi pada DBMS MySQL dan DBMS PostgreSQL tanpa menggunakan kondisi sedangkan untuk menggunakan kondisi hal ini belum dapat dilakukan karena perbedaan tipe data antara DBMS Oracle 10g dengan DBMS MySQL dan DBMS PostgreSQL.

VI.2 Saran

Diharapkan dalam pengembangan tugas akhir ini dapat mengimplementasikan fitur *fine grained auditing* dengan menggunakan kondisi pada DBMS MySQL dan DBMS PostgreSQL dengan menggunakan perpaduan antara *trigger* dan *procedure.*

DAFTAR PUSTAKA

1. Ramon A. Mata-Toledo dan Pauline K. Chusman. 2004. *Dasar-dasar Database Relasional*. Jakarta: Erlangga.
2. Oracle. *Oracle 10g Administration I Study Guide*. Oracle
3. Sigit Suyanntoro. 2006. *Pengolahan Database dengan MySQL*. Yogyakarta: Andi.
4. MySQL. 2007. *MySQL Guide to MySQL Documentation*. MySQL
5. http://catur.dosen.akprind.ac.id/2011/02/07/perlunya-keamanan-database/ diakses pada tanggal 19 Maret 2011.
6. http://www.detiknews.com/read/2011/03/25/145918/1601341/10/tilep-duit-nasabah-rp-17-m-karyawan-bank-swasta-ditahan-polisi diakses pada tanggal 19 Maret 2011.
7. http://catur.dosen.akprind.ac.id/2011/02/07/perlunya-keamanan-database/ diakses pada tanggal 20 maret 2011.
8. http://www.detiknews.com/read/2011/03/25/145918/1601341/10/tilep-duit-nasabah-rp-17-m- diakses pada tanggal 23 Maret 2011.
9. http://blog.re.or.id/model-basis-data.htm diakses pada tanggal 25 Maret 2011.
10. http://akbar.staff.gunadarma.ac.id diakses pada tanggal 28 Maret 2011.
11. http://student.eepisits.edu/~pungky/Modul/Administrasi%20Basis%20Data%20%20Rengga/Day-09/11%20%20Keamanan%20Database%20Oracle.pdf diakses pada tanggal 30 Maret 2011.
12. http://lecturer.eepis-its.edu/~tessy/tutorial/sqlold/Introduction%20to%20Oracle.pdf diakses pada tanggal 12 April 2011.
13. http://id.wikipedia.org/wiki/Sistem_manajemen_basis_data diakses pada tanggal 13 April 2011.
14. http://id.wikipedia.org/wiki/Orcle diakses pada tanggal 15 April 2011.

15. http://mudafiq.student.umm.ac.id/files/2010/05/mudafiq-trigger.pdf diakses pada tanggal 20 April 2011.
16. http://id.wikipedia.org/wiki/MySQL diakses pada tanggal 28 April 2011.
17. http://dev.mysql.com/doc/mysql-security-excerpt/5.0/en/index.html diakses pada tanggal 12 Mei 2011.
18. http://achmatim.net/2010/02/24/mengenal-trigger-di-mysql/ diakses pada tanggal 15 Mei 2011.
19. http://id.wikipedia.org/wiki/PostgreSQL diakses pada tanggal 20 Mei 2011.
20. http://www.postgresql.org/docs/9.0/static/runtime-config-logging.html diakses pada tanggal 30 Mei 2011.
21. http://rojulman.web.id/index.php?pg=15&dt=1&dts=4 diakses pada tanggal 30 Juni 2011.