IMPLEMENTASI DATABASE AKTIF PADA SISTEM INFORMASI TUGAS AKHIR FAKULTAS MIPA

SKRIPSI

JAMES SONGLI PATANDIANAN

H13115305

PROGRAM STUDI SISTEM INFORMASI

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2022

IMPLEMENTASI DATABASE AKTIF PADA SISTEM INFORMASI TUGAS AKHIR FAKULTAS MIPA

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana sains pada program studi Sistem Informasi Departemen Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas hasanuddin

JAMES SONGLI PATANDIANAN

H13115305

PROGRAM STUDI SISTEM INFORMASI

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2022

IMPLEMENTASI DATABASE AKTIF PADA SISTEM INFORMASI TUGAS AKHIR FAKULTAS MIPA

Disusun dan diajukan oleh

JAMES SONGLI PATANDIANAN
H13115305

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian studi Program Sarjana Program Studi Sistem Informasi Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin Pada tanggal 9 Mei 2023 dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui

Pembimbing Utama

Dr. Hendra, S.Si, M.Kom.
NIP. 197601022002121001

Pembimbing Pendamping

Ir. Eliyah Acantha Manapa Sampetoding, S.Kom., M.Kom.
NIP. 327322191191006

Ketua Program Studi

Dr. Hendra, S.Si, M.Kom.
NIP. 197601022002121001

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini:

Nama : James Songli Patandianan

NIM : H13115305

Program Studi : Sistem Informasi

Jenjang : S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul :

Implementasi Database Aktif Pada Sistem Informasi Tugas Akhir Fakultas Mipa

Adalah karya tulisan saya sendiri, bukan merupakan pengambil alihan tulisan orang lain dan bahwa skripsi/tesis/disertasi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan isi skripsi/tesis/disertasi ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Makassar, 9 Mei 2023

James Songli Patandianan

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan kehadirat Allah Yang Maha Kuasa karena berkat rahmat serta kehendak-Nya sehingga penulis dapat menyelesaikan penyusunan skripsi yang berjudul "IMPLEMENTASI DATABASE AKTIF PADA SISTEM INFORMASI TUGAS AKHIR FAKULTAS MIPA" ini dapat diselesaikan guna memenuhi salah satu persyaratan dalam menyelesaikan pendidikan pada program studi Sistem Informasi Departemen Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin Makassar.

Selama proses pengerjaan skripsi ini,banyak dinamika yang dihadapi penulis. kendala-kendala dan permasalahan yang penulis hadapi, baik mengenai literatur serta dalam menganalisis data, tentunya tidak dapat diatasi tanpa bantuan dari berbagai pihak.

Disamping itu, penulis juga ingin mengucapkan terima kasih dan penghargaan yang setinggi-tingginya kepada semua pihak yang telah membantu dalam proses penulisan skripsi ini diantaranya adalah:

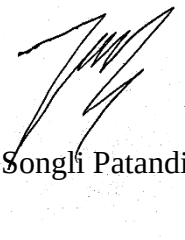
1. Rektor Universitas Hasanuddin, Bapak Prof. Dr. Ir. Jamaluddin Jompa, M.Sc. beserta jajarannya.
2. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam, Dr. Eng. Amiruddin beserta jajarannya.
3. Ketua Departemen Matematika FMIPA, Prof. Dr. Nurdin, S.Si., M.Si.
4. Bapak Dr. Hendra, S.Si, M.Kom. sebagai pembimbing utama yang telah banyak memberikan arahan, ide, motivavsi kepada penulis dalam banyak hal.
5. Ir. Eliyah Acantha Manapa Sampetoding, S.Kom., M.Kom. sebagai pembimbing pertama yang telah memberikan masukan dan arahan kepada penulis.
6. Bapak/Ibu Dosen Departemen Matematika Unhas yang telah memberikan ilmu kepada penulis selama menjadi mahasiswa di Departemen Matematika,

dan seluruh Staff Departemen Matematika dan Prodi Sistem Informasi Unhas yang telah membantu penulis dalam urusan berkas administrasi.

7. Teruntuk kedua orang tua saya terima kasih telah berjuang dan terus mendoakan, dan dukungan penulis hingga dapat menyelesaikan skripsi ini dengan baik.
8. Teruntuk saudara dan keluarga yang selalu mendukung, memberi semangat, dukungan moral, nasehat, doa, dan bisa memberikan kekuatan bagi penulis sehingga berada dititik ini.
9. Teman Kontrakan Sukabumi44 yang telah berjuang bersama dalam suka dan duka selama ini.
10. Teman – teman Program Studi Ilmu Komputer 2015 yang telah berjuang bersama dalam suka dan duka selama ini.

Penulis berharap semoga Tuhan Yang Maha Esa mengkaruniakan rahmat dan hidayah-Nya kepada mereka semua. Semoga skripsi ini dapat bermanfaat bagi kita semua, Aamin.

Makassar, 9 Mei 2023



James Songli Patandianan

DAFTAR ISI

ABSTRAK

Seiring dengan perkembangan zaman, teknologi pun berkembang sangat pesat yang juga menuntut manusia untuk mengikuti perkembangan tersebut. Maka hal ini pun menyebabkan munculnya berbagai jenis aplikasi yang dapat mempermudah pekerjaan maupun menyelesaikan masalah. Saat ini terdapat beberapa pengolah database yang populer dalam pengembangan aplikasi seperti MySQL. Oleh karena itu, penelitian ini mengimplementasikan database aktif pada sistem informasi tugas akhir Fakultas MIPA. Penelitian ini dilakukan dengan cara menganalisis database aktif agar dapat diimplementasikan untuk memproses tugas-tugas di belakang layar dan tidak lagi diproses melalui script aplikasi pada Sistem Informasi Tugas Akhir (SITA) di lingkungan Fakultas MIPA Universitas Hasanuddin. Dalam database terdapat sebuah Trigger sebagai sebuah prosedur yang terdapat pada SQL Server yang secara otomatis aktif jika data di dalam tabel berubah karena adanya perintah SQL (INSERT, UPDATE atau DELETE). Berdasarkan dari hasil analisis yang telah dilakukan implementasi database aktif pada sistem informasi tugas akhir fakultas MIPA, trigger pada database aktif berhasil diimplementasikan pada data dummy yang dibuat menyerupai database Sistem Informasi Tugas Akhir MIPA diantaranya, berupa trigger untuk profil mahasiswa, informasi seminar, mengusulkan seminar proposal, ketika user melakukan pengarsipan data, pengubahan data, penginputan data, dan penghapusan data. Sehingga diharapkan penelitian ini dapat bermanfaat untuk penambahan beberapa fitur untuk kedepannya pada aplikasi SI-TA MIPA.

Keywords : Database, MySQL, Trigger

ABSTRACT

Along with the times, technology is developing very rapidly which also requires humans to keep up with these developments. So this also causes the emergence of various types of applications that can simplify work and solve problems. Currently there are several popular database processors in application development such as MySQL. Therefore, this study implements an active database on the final assignment information system for the Faculty of Mathematics and Natural Sciences. This research was conducted by analyzing active databases so that they can be implemented to process tasks behind the scenes and are no longer processed through application scripts in the Final Assignment Information System (SITA) within the Hasanuddin University Faculty of Mathematics and Natural Sciences. In the database there is a Trigger as a procedure contained in SQL Server which is automatically activated if the data in the table changes due to an SQL command (INSERT, UPDATE or DELETE). Based on the results of the analysis that has been carried out implementing an active database on the MIPA faculty final project information system, triggers on active databases have been successfully implemented on dummy data that are made to resemble the MIPA Final Project Information System database including, in the form of triggers for student profiles, seminar information, proposing seminar proposals , when the user performs data archiving, changing data, inputting data, and deleting data. So it is hoped that this research can be useful for adding several features for the future in the SI-TA MIPA application.

Keywords : Database, MySQL, Trigger

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan perkembangan zaman, teknologi pun berkembang sangat pesat yang juga menuntut manusia untuk mengikuti perkembangan tersebut. Maka hal ini pun menyebabkan munculnya berbagai jenis aplikasi yang dapat memepermudah pekerjaan maupun menyelesaikan masalah.

Mysql adalah sebuah *server database open source* yang terkenal yang digunakan berbagai aplikasi terutama untuk *server* atau membuat WEB. Mysql berfungsi sebagai SQL (*Structured Query Language*) yang dimiliki sendiri dan sudah diperluas oleh Mysql umumnya digunakan bersamaan dengan PHP untuk membuat aplikasi *server* yang dinamis dan *powerfull* (Istiono, Hijrah, & Sutarya, 2016).

Database merupakan sekumpulan data yang terhubung secara *logic* dan dapat digunakan bersama-sama dalam suatu jaringan komputer. Dalam *database* terdapat sebuah *Trigger*, *Trigger* merupakan sebuah prosedur yang terdapat pada SQL *Server* yang secara otomatis aktif jika data di dalam tabel berubah karena adanya perintah SQL (*INSERT*, *UPDATE* atau *DELETE*).

Sistem Informasi Tugas Akhir merupakan sebuah aplikasi yang berfungsi dalam pengusulan judul, mengusulkan pembimbing, dan wadah mahasiswa untuk melakukan bimbingan tugas akhir. Pada *database* Sistem Informasi Tugas Akhir (SITA) Fakultas MIPA Universitas Hasanuddin pengolahan *database* masih menggunakan *script* aplikasi dan masih belum optimal.

Berdasarkan uraian di atas maka solusi dari permasalahan ini adalah dengan melakukan analisis pada *database* aktif dengan memperhatikan *Trigger* yang dapat di implementasikan pada *database* Sistem Informasi Tugas Akhir (SITA) agar *database* tidak lagi diproses melalui *script* aplikasi. Sehingga dari permasalahan diatas, dibuatlah skripsi dengan judul **"IMPLEMENTASI DATABASE AKTIF PADA SISTEM INFORMASI TUGAS AKHIR FAKULTAS MIPA".**

1.2 Rumusan Masalah

Berdasarkan uraian di atas maka yang menjadi rumusan masalah dalam penelitian ini yaitu bagaimana penerapan *Trigger* database aktif pada Sistem Informasi Tugas Akhir (SITA) di lingkungan Fakultas MIPA Universitas Hasanuddin.

1.3 Tujuan

Tujuan dari penelitian ini yaitu menganalisis *database* aktif agar dapat diimplementasikan untuk memproses tugas-tugas di belakang layar dan tidak lagi diproses melalui *script* aplikasi pada Sistem Informasi Tugas Akhir (SITA) di lingkungan Fakultas MIPA Universitas Hasanuddin.

1.4 Manfaat

Dengan adanya implementasi *database* aktif pada data Sistem Informasi Tugas Akhir (SITA), database tidak lagi diproses melalui *script* aplikasi.

1.5 Batasan Masalah

Batasan masalah yang dilakukan pada pengerjaan tugas akhir ini adalah:

1. Hanya dilakukan pada ruang lingkup Fakultas MIPA Universitas Hasanuddin.
2. Data yang digunakaan dalam analisis database *Trigger* adalah data *dummy.*
3. Aplikasi yang digunakan dalam pengolahan data adalah mysql.

BAB II
TINJAUAN PUSTAKA

2.1 Landasan Teori

2.1.1 Sistem

Sistem Menurut Fat menyimpulkan bahwa: Sistem adalah suatu himpunan suatu benda nyata atau abstrak (*a set of thing*) yang terdiri dari bagian - bagian atau komponen - komponen yang saling berkaitan, berhubunagan, berketergantungan, saling mendukung, yang secara keseluruhan bersatu dalam kesatuan (*Unity*) untuk mencapai tujuan tertentu secara efesien dan efektif (Septiani, Afni, & Andharsaputri, 2019).

2.1.2 Informasi

Menurut Kusrini dalam mendefinisikan bahwa “Informasi merupakan hasil olahan data, dimana data tersebut sudah diproses dan diinterpretasikan menjadi sesuatu yang bermakna untuk pengambilan keputusan. Informasi juga diartikan sebagai himpunan dari data yang relevan dengan satu atau beberapa orang dalam suatu waktu.” Informasi adalah data yang telah diorganisasi, dan telah memiliki kegunaan dan manfaat. Agar bermanfaat, informasi harus memiliki kualitas atau karakteristik sebagai berikut, relevan, dapat dipercaya, lengkap, tepat waktu, mudah dipahami, dapat diuji kebenarannya (Septiani, Afni, & Andharsaputri, 2019).

2.1.3 Sistem Informasi

Sistem informasi merupakan penggabungan dari sistem dan informasi, dengan demikian bisa didefinisikan bahwa sistem informasi adalah kumpulan dari subsub sistem yang saling terintegrasi dan berkolaborasi untuk menyelesaikan masalah tertentu dengan cara mengolah data dengan alat yang namanya komputer sehingga memiliki nilai tambah dan bermanfaat bagi pengguna. Sistem informasi merupakan penerapan sistem didalam organisasi untuk mendukung informasi yang dibutuhkan oleh semua tingkat manajemen (Septiani, Afni, & Andharsaputri, 2019).

Sistem Informasi (SI) adalah kombinasi dari teknologi informasi dan aktivitas orang yang menggunakan teknologi itu untuk mendukung operasi dan manajemen. Dalam arti yang sangat luas, istilah sistem informasi yang sering digunakan merujuk kepada interaksi antara orang, proses algoritmik, data, dan teknologi (Nugroho, 2016).

2.1.4 Sistem Informasi Tugas Akhir (SITA) Mahasiswa

Sistem Informasi Tugas Akhir merupakan sebuah aplikasi yang berfungsi untuk mempermudah dalam pengusulan judul, mengusulkan pembimbing, dan wadah mahasiswa untuk melakukan bimbingan tugas akhir.

Sistem Informasi (SI) Tugas Akhir yang ditujukan untuk mempersingkat proses administrasi TA seperti proses pendaftaran judul TA, pendaftaran seminar TA 1, pendaftaran sidang TA 2, pendaftaran yudisium, dan bimbingan TA, serta proses pemantauan mahasiswa yang melaksanakan TA oleh kaprodi, dosen, ataupun staf program studi (Mustianti, Ketut Widiartha, & Albar, 2020).

sistem informasi tugas akhir dan skripsi dapat membantu dan meningkatkan kinerja pengguna di bagian divisi administrasi akademik dan kemahasiswaan dan panitia dalam memproses data. Hal ini juga mendukung penelitian yang peneliti lakukan karena dengan adanya sistem informasi tugas akhir atau skripsi sangat membantu meningkatkan kinerja panitia skripsi dan mahasiswa juga sekretariat dalam melaksanakan seluruh prosedur yang ada pada pelaksanaan tugas akhir atau skripsi (Suwita, 2020).

2.1.5 MySQL

Mysql adalah sebuah *server database open source* yang terkenal yang digunakan berbagai aplikasi terutama untuk *server* atau membuat WEB. Mysql berfungsi sebagia SQL (*Structured Query Language*) yang dimiliki sendiri dan sudah diperluas oleh Mysql umumnya digunakan bersamaan dengan PHP untuk membuat aplikasi *server* yang dinamis dan *powerfull*. Tidak sama dengan proyek-proyek seperti *Apache*, dimana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL

AB, dimana memegang hak cipta hampir atas semua kode sumbernya, Kedua orang Swedia dan satu orang Findlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael"Monty" Widenius.

Mysql adalah sebuah implementasi dari sistem manajemen basisdata relasional (RDMS) 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 basis data yang telah ada sebelumnya; SQL(Structured Query Language). SQL adalah sebuah konsep pengoperasian basis data, 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 basis data transaksional maupun operasi basis data non transaksional. Pada modus operasi non-transaksional, MySQL dapat dikatakan unggul dalam hal untuk 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 ditunjukkan untuk bisnis sangat disarankan untuk menggunakan modus basisdata transaksional, hanya saja sebagai konsekuensinya untuk kerja MySQL pada modus transaksaional tidak secepat untuk kerja pada modus non-transaksional (Istiono, Hijrah, & Sutarya, 2016).

Menurut Rulianto Kurniawan (2010 :16) MySQL merupakan suatu jenis *database server* yang sangat terkenal. MySQL termasuk jenis RDBMS (*Relational Database Manajement System*). MySQL mendukung bahasa pemrograman PHP,

bahasa permintaan yang terstruktur, karena pada penggunaannya SQL memiliki berberapa aturan yang telah distandarkan oleh asosiasi yang bernama ANSI. MySQL merupakan RDBMS (*Relational Database Management System*) *server*. RDBMS adalah program yang memungkinkan pengguna *database* untuk membuat, mengelola, dan menggunakan data pada suatu *model relational*. Dengan demikian, tabel-tabel yang ada pada database memiliki relasi antara satu tabel dengan tabel lainnya.

Beberapa keunggulan dari MySQL yaitu :

- Cepat, handal dan mudah dalam penggunaannya. MySQL lebih epat tiga sampai empat kali dari pada database *server* komersial yang beredar saat ini, mudah diatur dan tidak memerlukan seseorang yang ahli untuk mengatur administrasi pemasangan MySQL.
- Didukung oleh berbagai bahasa *Database Server* MySQL dapat memberikan pesan Error dalam berbagai bahasa seperti Belanda, Portugis, Spanyol, Inggris, Perancis, Jerman, dan Italia.
- Mampu membuat tabel berukuran sangat besar. Ukuran maksimal dari setiap tabel yang dapat dibuat dengan MySQL adalah 4 GB sampai dengan ukuran file yang dapat ditangani oleh sistem operasi yang dipakai.
- Lebih murah MySQL bersifat *open source* dan didistribusikan dengan gratis tanpa biaya untuk UNIX platform, OS/2 dan *Windows Platform*. Melekatnya integrasi PHP dengan MySQL. Keterikatan antara PHP dengan MySQL yang sama-sama *Software Open-Source* sangat kuat, sehingga koneksi yang terjadi lebih cepat jika dibandingkan dengan menggunakan *database server* lainnya. Modul MySQL di PHP telah dibuat *Built-in* sehingga tidak memerlukan konfigurasi tambahan pada File konfigurasi Php ini (Hermiati, Asnawati, & Kanedi, 2021).

2.1.6 Basis Data

Basis data adalah kumpulan data yang terhubung secara *logical*, terdeskripsi dari data yang dirancang untuk memenuhi kebutuhan informasi dari sebuah organisasi. Basis data merupakan sebuah tempat pengumpulan data yang sangat

besar yang digunakan secara bersama-sama oleh berbagai departemen (Whitten, et al., 2004).

Sistem basis data adalah sistem menyimpan informasi dan memungkinkan penggunanya mengambil dan mengubah informasi itu saat diperlukan. Komponen-komponen penting. sistem basis data adalah: data, perangkat keras (*hardware*), perangkat lunak (*software*), dan pengguna (*user*) yang dikategorikan sebagai *Administrator, Programmer* dan *End-user*. *Database Management System* (DBMS) dirancang khusus sebagai aplikasi yang berinteraksi dengan pengguna, dengan aplikasi lain dan dengan basis data (*database*) itu sendiri untuk menangkap dan menganalisa data. DBMS adalah perangkat lunak khusus yang digunakan untuk membuat, mengontrol dan mengelola basis data. DBMS antara lain adalah: MySQL™, PostgreSQL™, SQLite™, Microsoft SQL Server™, Microsoft Access™, Oracle™, Sybase™, dBASE™, FoxPro™, dan IBM DB2™ (Hamidy, 2017).

Kegunaan Basisdata:

Suatu *database* dibentuk untuk mengatasi masalah yang sering dihadapi di dalam pengolahan data seperti :

- Redudansi dan Inkonsistensi Data Penyimpanan data yang sama pada beberapa tempat atau media penyimpanan yang mengakibatkan terjadinya pemborosan media penyimpanan. Penyimpanan data yang sama dan berulang-ulang di beberapa file dapat mengakibatkan inkonsistensi (tindak konsisten).
- Keamanan Data Dengan *database* managemen, sistem kemananan data bisa dicapai. Misalnya: data mengenai gaji pengawai hanya boleh dibuka oleh bagian keuangan dan personalia, bagian lain tindak diperbolehkan menggunakannya dengan membuat suatu *password* dan wewenang atau *userautorization* dan bersih.
- Kesuliltan Mengakses Data *Database* dapat mengakses kesulitan dalam mengakses data karena mampu mengambil data secara langsung dengan program aplikasi yang mudah digunakan.

- Isolasi Data untuk Standarisasi Jika data tersebar dalam bentuk format yang tidak sama, maka ini menyulitkan dalam menulis program aplikasi untuk mengambil dan menyimpan data. Maka suatu database haruslah dibuat suatu format, sehingga mudah dibuat program aplikasinya (Sovia & Febio, 2011).

2.1.7 Database Aktif

Database aktif adalah sistem database yang diperluas dengan aturan aktif, yaitu prosedur tersimpan yang secara otomatis dipanggil oleh DBMS ketika peristiwa tertentu terjadi.

Bentuk aturan aktif yang paling terkenal disebut *Event-Condition-Action* (ECA); Aturan ECA memiliki pola tiga kali lipat: bagian *event* menentukan event yang harus terjadi agar *rule* terpanggil; bagian kondisi menyatakan prakondisi untuk mengeksekusi rule; bagian action sebenarnya berisi kode untuk menjalankan (Comai, Fraternali, Psaila, & Tanca, 1996).

Sistem basis data aktif didasarkan pada pendekatan yang berbeda untuk aturan dalam DBMS daripada deduktif sistem basis data. Mereka mewakili perilaku aktif yang awalnya diterapkan di AI dan pakar sistem.

Aturan dalam database aktif biasanya terdiri dari tiga bagian: peristiwa, kondisi dan tindakan. Ini sangat cocok dengan perilaku reaktif. Ketika beberapa peristiwa terjadi, kondisi dievaluasi dan jika benar, tindakan dilakukan. Aturan seperti itu dikenal sebagai event-condition-action atau aturan ECA. Dengan demikian, semantik aturan aktif bersifat prosedural [Ceri dan Ramakrishnan, 1996].

Aturan aktif tidak harus memuat ketiga bagian tersebut. Bagian acara atau kondisi bisa dihilangkan. Kemudian kita berbicara tentang aturan kondisi-tindakan (sering disebut aturan produksi) atau aturan acara-aksi. Setiap jenis aturan aktif memiliki jenis aturan penggunaan dan produksi yang spesifik sangat mirip dengan aturan deduktif (Koz´ak, 2011).

a. MySQL TRIGGER

Trigger di MySQL adalah sekumpulan pernyataan SQL yang berada di katalog sistem. Ini adalah tipe khusus dari prosedur tersimpan yang dipanggil secara otomatis sebagai respon terhadap suatu *event*. Setiap *Trigger* dikaitkan dengan tabel, yang diaktifkan pada pernyataan DML apa pun seperti *INSERT*, *UPDATE*, atau *DELETE*.

Trigger disebut prosedur khusus karena tidak dapat dipanggil secara langsung seperti prosedur tersimpan. Perbedaan utama antara *Trigger* dan prosedur adalah bahwa *Trigger* dipanggil secara otomatis saat peristiwa modifikasi data dibuat pada table. Sebaliknya, prosedur tersimpan harus dipanggil secara eksplisit.

b. Manfaat Trigger database
    - *Trigger* membantu penulis menegakkan aturan bisnis.
    - *Trigger* membantu penulis memvalidasi data bahkan sebelum dimasukkan atau diperbarui.
    - *Trigger* membantu penulis menyimpan catatan seperti memelihara jejak audit dalam tabel.
    - *Trigger* SQL menyediakan cara alternatif untuk memeriksa integritas data.
    - *Trigger* memberikan cara alternatif untuk menjalankan tugas terjadwal.
    - *Trigger* meningkatkan kinerja kueri SQL karena tidak perlu dikompilasi setiap kali kueri dijalankan.
    - *Trigger* mengurangi kode sisi klien yang menghemat waktu dan tenaga.
    - *Trigger* membantu penulis menskalakan aplikasi penulis di berbagai platform.
    - *Trigger* mudah di *maintenance*.
c. Keterbatasan Penggunaan Trigger di MySQL
    - *Trigger* MySQL tidak mengizinkan penggunaan semua validasi; trigger hanya menyediakan validasi tambahan. Misalnya, penulis dapat

menggunakan batasan *NOT NULL, UNIQUE, CHECK*, dan *FOREIGN KEY* untuk validasi sederhana.

- *Trigger* dipanggil dan dieksekusi tanpa terlihat dari aplikasi klien. Oleh karena itu, tidak mudah memecahkan masalah yang terjadi di lapisan basis data.
- *Trigger* dapat meningkatkan *overhead server database*.

d. Jenis Trigger di MySQL

Penulis dapat mendefinisikan maksimal enam jenis tindakan atau peristiwa dalam bentuk *Trigger*:

- *Before Insert*: diaktifkan sebelum penyisipan data ke dalam tabel.
- *AFTER Insert*: diaktifkan setelah penyisipan data ke dalam tabel.
- *Before Update*: diaktifkan sebelum pembaruan data dalam tabel.
- *AFTER Update*: diaktifkan setelah pembaruan data dalam tabel.
- *Before Delete*: diaktifkan sebelum data dihapus dari tabel.
- *AFTER Delete*: diaktifkan setelah penghapusan data dari tabel.

Berikut penjelasannya :

1. MySQL BEFORE INSERT TRIGGER

Before Insert Trigger di MySQL dipanggil secara otomatis setiap kali operasi penyisipan dijalankan. Pada artikel ini, penulis akan belajar cara membuat *Trigger* sebelum *insert* dengan sintaks dan contohnya.

Syntax :

Berikut ini adalah sintaks untuk membuat *Trigger BEFORE INSERT* di MySQL:

```
CREATE TRIGGER Trigger_name
BEFORE INSERT
ON table_name FOR EACH ROW
Trigger_body ;
```

Parameter sintaks *Trigger* dapat dijelaskan seperti di bawah ini:

- Pertama, penulis akan menentukan nama *trigger* yang ingin penulis buat yang harus unik dalam skema.
- Kedua, penulis akan menentukan waktu tindakan *trigger*, yang seharusnya *BEFORE INSERT*. *trigger* ini akan dipanggil sebelum setiap modifikasi baris terjadi pada tabel.
- Ketiga, penulis akan menentukan nama tabel yang terkait dengan *trigger* harus ditulis setelah kata kunci *ON*. Jika penulis tidak menentukan nama tabel, *trigger* tidak dapat dibuat.
- Terakhir, penulis akan menentukan pernyataan untuk eksekusi saat *trigger* dipanggil.

Jika penulis ingin mengeksekusi banyak pernyataan, penulis akan menggunakan blok *BEGIN END* yang berisi kumpulan kueri untuk menentukan logika *trigger*. Lihat sintaks di bawah ini:

```
DELIMITER $$
CREATE TRIGGER Trigger_name BEFORE
INSERT
ON table_name FOR EACH ROW
BEGIN
  variable declarations
  Trigger code
END$$
DELIMITER ;
```

Batasan :

- Penulis dapat mengakses dan mengubah nilai *NEW* hanya dalam *trigger BEFORE INSERT*.

- Penulis tidak dapat mengakses *OLD* Jika penulis mencoba mengakses nilai *OLD*, penulis akan mendapatkan kesalahan karena nilai *OLD* tidak ada.
- Penulis tidak dapat membuat *trigger BEFORE INSERT* pada *VIEW*.

2. MySQL AFTER INSERT TRIGGER

After Insert Trigger di MySQL dipanggil secara otomatis setiap kali peristiwa penyisipan terjadi di tabel. Pada artikel ini, penulis akan mempelajari cara membuat *Trigger AFTER insert* dengan sintaks dan contohnya.

Syntax :

Berikut ini adalah sintaks untuk membuat *trigger AFTER INSERT* di MySQL:

```
CREATE TRIGGER Trigger_name
AFTER INSERT
ON table_name FOR EACH ROW
Trigger_body ;
```

Parameter sintaks *Trigger AFTER INSERT* dapat dijelaskan seperti di bawah ini:

- Pertama, penulis akan menentukan nama *trigger* yang ingin penulis buat yang harus unik dalam skema.
- Kedua, penulis akan menentukan *trigger action time*. Yang mana harus ditulis AFTER INSERT untuk memanggil trigger.
- Ketiga, penulis akan menentukan nama tabel yang terkait dengan *trigger*. Tabel harus ditulis setelah kata kunci *ON*. Jika penulis tidak menentukan nama tabel, *trigger* tidak dapat dibuat.
- Terakhir, penulis akan menentukan *trigger body* yang berisi satu atau lebih pernyataan untuk dieksekusi saat *trigger* diaktifkan.

Jika penulis ingin mengeksekusi banyak pernyataan, penulis akan menggunakan blok BEGIN END yang berisi kumpulan kueri SQL untuk menentukan logika *Trigger*. Lihat sintaks di bawah ini :

```
DELIMITER $$
CREATE TRIGGER Trigger_name AFTER INSERT
ON table_name FOR EACH ROW
BEGIN
  variable declarations
  Trigger code
END$$
DELIMITER ;
```

Batasan :

- Penulis dapat mengakses nilai *NEW* tetapi tidak dapat mengubahnya dalam *Trigger AFTER INSERT*.
- Penulis tidak dapat mengakses *OLD* Jika penulis mencoba mengakses nilai *OLD*, penulis akan mendapatkan kesalahan karena tidak ada *OLD* pada *Trigger INSERT*.
- Penulis tidak dapat membuat *trigger AFTER INSERT* pada *VIEW*.

3. MySQL BEFORE UPDATE Trigger

BEFORE UPDATE Trigger di MySQL dipanggil secara otomatis setiap kali operasi pembaruan dijalankan pada tabel yang terkait dengan *Trigger*. Pada artikel ini, penulis akan mempelajari cara membuat *before update trigger* dengan sintaks dan contohnya.

Syntax :

Berikut ini adalah sintaks untuk membuat *Trigger BEFORE UPDATE* di MySQL:

```
CREATE TRIGGER Trigger_name
BEFORE UPDATE
ON table_name FOR EACH ROW
Trigger_body ;
```

Parameter sintaks *Trigger BEFORE UPDATE* dijelaskan sebagai berikut :

- Pertama, penulis akan menentukan nama *trigger* yang ingin penulis buat yang harus unik dalam skema.
- Kedua, penulis akan menentukan *trigger action time*, yang seharusnya *BEFORE UPDATE*. *Trigger* ini akan dipanggil sebelum setiap baris diubah pada tabel.
- Ketiga, penulis akan menentukan nama tabel yang terkait dengan *trigger* yang harus ditulis setelah kata kunci *ON*. Jika penulis tidak menentukan nama tabel, *Trigger* tidak dapat dibuat.
- Terakhir, penulis akan menentukan *trigger body* yang berisi pernyataan untuk dieksekusi saat *Trigger* diaktifkan.

Jika penulis ingin mengeksekusi banyak pernyataan, penulis akan menggunakan blok *BEGIN END* yang berisi kumpulan kueri untuk menentukan logika *Trigger*. Lihat sintaks di bawah ini :

```
DELIMITER $$
CREATE TRIGGER Trigger_name BEFORE UPDATE
ON table_name FOR EACH ROW
BEGIN
  variable declarations
  Trigger code
END$$
DELIMITER ;
```

Batasan :

- Penulis tidak dapat memperbarui nilai *OLD* dalam *Trigger BEFORE UPDATE*.
- Penulis dapat mengubah nilai *NEW*.
- Penulis tidak dapat membuat *Trigger BEFORE UPDATE* pada *VIEW*.

4. MySQL AFTER UPDATE TRIGGER

Trigger AFTER UPDATE di MySQL dipanggil secara otomatis setiap kali peristiwa *UPDATE* diaktifkan di tabel yang terkait dengan *Trigger*. Pada artikel ini, penulis akan mempelajari cara membuat *trigger AFTER UPDATE* dengan sintaks dan contohnya.

Syntax :

Berikut ini adalah sintaks untuk membuat *trigger AFTER UPDATE* di MySQL:

```
CREATE TRIGGER Trigger_name
AFTER UPDATE
ON table_name FOR EACH ROW
Trigger_body ;
```

Penulis dapat menjelaskan parameter sintaks *trigger AFTER UPDATE* seperti di bawah ini:

- Pertama, penulis akan menentukan nama *trigger* yang ingin penulis buat yang harus unik dalam skema.
- Kedua, penulis akan menentukan *trigger action time*, yang seharusnya *AFTER UPDATE*. *Trigger* ini akan dipanggil setelah terjadi perubahan di setiap baris pada tabel.
- Ketiga, penulis akan menentukan nama tabel yang terkait dengan *Trigger* yang harus ditulis setelah *ON* Jika penulis tidak menentukan nama tabel, *Trigger* tidak dapat dibuat.
- Terakhir, penulis akan menentukan *trigger body* yang berisi pernyataan untuk dieksekusi saat *trigger* diaktifkan.

Jika penulis ingin mengeksekusi lebih dari satu pernyataan, penulis akan menggunakan blok *BEGIN END* yang berisi kumpulan kueri SQL untuk menentukan logika *Trigger*. Lihat sintaks di bawah ini:

```
DELIMITER $$
CREATE TRIGGER Trigger_name AFTER UPDATE
ON table_name FOR EACH ROW
BEGIN
  variable declarations
  Trigger code
END$$
DELIMITER ;
```

Batasan :

- Penulis dapat mengakses baris *OLD* tetapi tidak dapat memperbaruinya.
- Penulis dapat mengakses baris *NEW* tetapi tidak dapat memperbaruinya.
- Penulis tidak dapat membuat *Trigger AFTER UPDATE* pada *VIEW*.

5. MySQL BEFORE DELETE Trigger

BEFORE DELETE Trigger di MySQL dipanggil secara otomatis setiap kali operasi penghapusan dijalankan di tabel. Pada artikel ini, penulis akan belajar cara membuat BEFORE DELETE *trigger* dengan sintaks dan contohnya.

Syntax :

Berikut ini adalah sintaks untuk membuat *trigger BEFORE DELETE* di MySQL:

```
CREATE TRIGGER Trigger_name
BEFORE DELETE
ON table_name FOR EACH ROW
Trigger_body ;
```

Parameter sintaks *trigger BEFORE DELETE* dapat dijelaskan seperti di bawah ini:

- Pertama, penulis akan menentukan nama *Trigger* yang ingin penulis buat yang harus unik dalam skema.
- Kedua, penulis akan menentukan waktu *trigger action time*, yang seharusnya *BEFORE DELETE*. *Trigger* ini akan dipanggil sebelum setiap baris perubahan terjadi pada tabel.
- Ketiga, penulis akan menentukan nama tabel yang terkait dengan *Trigger*. Itu harus ditulis setelah kata kunci *ON*. Jika penulis tidak menentukan nama tabel, *trigger* tidak dapat dibuat.
- Terakhir, penulis akan menentukan pernyataan untuk eksekusi saat *trigger* diaktifkan.

Jika penulis ingin mengeksekusi banyak pernyataan, penulis akan menggunakan blok *BEGIN END* yang berisi kumpulan kueri untuk menentukan logika *trigger*. Lihat sintaks di bawah ini :

```
DELIMITER $$
CREATE TRIGGER Trigger_name BEFORE DELETE
ON table_name FOR EACH ROW
BEGIN
  variable declarations
  Trigger code
END$$
DELIMITER ;
```

Batasan :

- Penulis dapat mengakses baris *OLD* tetapi tidak dapat memperbaruinya dalam *Trigger BEFORE DELETE*.
- Penulis tidak dapat mengakses baris *NEW*. Itu karena tidak ada baris baru.
- Penulis tidak dapat membuat *trigger* BEFORE *DELETE* pada VIEW.

6. MySQL AFTER DELETE Trigger

Trigger AFTER DELETE di MySQL dipanggil secara otomatis setiap kali peristiwa penghapusan dieksekusi di tabel. Pada artikel ini, penulis akan mempelajari cara membuat *trigger AFTER DELETE* dengan sintaks dan contohnya.

Syntax :

Berikut ini adalah sintaks untuk membuat *trigger AFTER DELETE* di MySQL:

```
CREATE TRIGGER Trigger_name
AFTER DELETE
ON table_name FOR EACH ROW
Trigger_body ;
```

Parameter sintaks *trigger AFTER DELETE* dapat dijelaskan seperti di bawah ini:

- Pertama, penulis akan menentukan nama *Trigger* yang ingin penulis buat yang harus unik dalam skema.
- Kedua, penulis akan menentukan *trigger action time*, yang seharusnya *AFTER DELETE*. *Trigger* ini akan dipanggil setelah setiap baris perubahan terjadi pada tabel.
- Ketiga, penulis akan menentukan nama tabel yang terkait dengan *Trigger*. Itu harus ditulis setelah kata kunci *ON*. Jika penulis tidak menentukan nama tabel, *Trigger* tidak akan ada.
- Terakhir, penulis akan menentukan badan *trigger body* yang berisi pernyataan untuk dieksekusi saat *trigger* dipanggil.

Jika penulis ingin mengeksekusi banyak pernyataan, penulis akan menggunakan blok *BEGIN END* yang berisi kumpulan kueri SQL untuk menentukan logika *trigger*. Lihat sintaks di bawah ini :

```
DELIMITER $$
CREATE TRIGGER Trigger_name AFTER DELETE
ON table_name FOR EACH ROW
BEGIN
  variable declarations
  Trigger code
END$$
DELIMITER ;
```

Batasan :

- Penulis dapat mengakses baris *OLD* tetapi tidak dapat memperbaruinya di *trigger AFTER DELETE*.
- Penulis tidak dapat mengakses baris *NEW*, karena tidak ada baris *NEW*.
- Penulis tidak dapat membuat *trigger AFTER DELETE* pada *VIEW*.

2.1.8 Konvensi Penamaan

Konvensi penamaan adalah seperangkat aturan yang penulis ikuti untuk memberikan nama unik yang sesuai. Ini menghemat waktu penulis untuk menjaga

agar pekerjaan tetap teratur dan mudah dipahami. Oleh karena itu, penulis harus menggunakan nama unik untuk setiap *trigger* yang terkait dengan tabel. Namun, sebaiknya tetapkan nama *trigger* yang sama untuk tabel yang berbeda.

Konvensi penamaan berikut harus digunakan untuk memberi nama *trigger* di MySQL :

```
(BEFOR | AFTER) table_name (INSERT | UPDATE | DELETE)
```

Dengan demikian,

***Trigger* Activation Time:** BEFORE | AFTER

***Trigger* Event:** INSERT | UPDATE | DELETE

2.1.9 Cara membuat Trigger di MySQL

Penulis dapat menggunakan pernyataan *CREATE TRIGGER* untuk membuat *trigger* baru di MySQL. Di bawah ini adalah sintaks untuk membuat *trigger* di MySQL :

```
CREATE TRIGGER Trigger_name
(AFTER | BEFORE) (INSERT | UPDATE | DELETE)
ON table_name FOR EACH ROW
BEGIN
 --variable declarations
--Trigger code
END;
```

a. MySQL Create Trigger

Pada artikel ini, penulis akan belajar cara membuat *trigger* pertama di MySQL. Penulis dapat membuat *trigger* baru di MySQL dengan menggunakan pernyataan *CREATE TRIGGER*. Ini untuk memastikan bahwa penulis memiliki hak *trigger*

saat menggunakan perintah *CREATE TRIGGER*. Berikut ini adalah sintaks dasar untuk membuat *trigger* :

```
CREATE TRIGGER Trigger_name  Trigger_time Trigger_event
ON table_name FOR EACH ROW
BEGIN
--variable declarations
--Trigger code
END;
```

Penjelasan Parameter :

Pembuatan *syntax trigger* berisi parameter berikut :

***Trigger*_name:** Ini adalah nama *trigger* yang penulis ingin buat yang harus ditulis setelah pernyataan *CREATE TRIGGER*. Ini untuk memastikan bahwa nama *Trigger* harus unik di dalam skema.

***Trigger*_time:** Ini adalah *trigger action time*, yang seharusnya *BEFORE* atau *AFTER*. Ini adalah parameter yang diperlukan saat menentukan *trigger*. Ini menunjukkan bahwa *trigger* akan dipanggil sebelum atau setelah setiap modifikasi baris terjadi pada tabel.

***Trigger*_event:** Ini adalah jenis nama operasi yang mengaktifkan *trigger*. Itu bisa berupa operasi *INSERT*, *UPDATE*, atau *DELETE*. *Trigger* hanya dapat memanggil satu peristiwa pada satu waktu. Jika penulis ingin mendefinisikan *trigger* yang dipanggil oleh banyak peristiwa, diperlukan untuk menentukan beberapa *trigger*, dan satu *trigger* untuk setiap peristiwa.

Table_name: Ini adalah nama tabel yang terkait dengan *trigger* yang harus ditulis setelah kata kunci *ON*. Jika penulis tidak menentukan nama tabel, *trigger* tidak dapat dibuat.

***BEGIN END Block*:** Akhirnya, penulis akan menentukan pernyataan untuk eksekusi saat *trigger* diaktifkan. Jika penulis ingin mengeksekusi banyak pernyataan, penulis akan menggunakan blok *BEGIN END* yang berisi kumpulan kueri untuk menentukan logika *trigger*.

Body Trigger dapat mengakses nilai kolom, yang dipengaruhi oleh pernyataan DML. Pengubah *NEW* dan *OLD* digunakan untuk membedakan nilai kolom *BEFORE* dan AFTER eksekusi pernyataan DML. Penulis dapat menggunakan nama kolom dengan pengubah *NEW* dan *OLD* sebagai *OLD*.col_name dan *NEW*.col_name. *OLD*.column_name menunjukkan kolom dari baris yang ada sebelum pembaruan atau penghapusan terjadi. *NEW*.col_name menunjukkan kolom dari baris baru yang akan disisipkan atau baris yang sudah ada setelah diperbarui.

Misalnya, penulis ingin memperbarui nama kolom message_info menggunakan *trigger*. Di badan *Trigger*, penulis dapat mengakses nilai kolom sebelum pembaruan sebagai *OLD*.message_info dan nilai baru *NEW*.message_info.

Penulis dapat memahami ketersediaan pengubah OLD dan NEW dengan tabel di lihat pada tabel 2.1 :

Tabel 2.1 Pengubahan data

Trigger Event	***OLD***	***NEW***
INSERT	*No*	*Yes*
UPDATE	*Yes*	*Yes*
DELETE	*Yes*	*No*

b. MySQL Show/List Triggers

Show atau list *trigger* sangat dibutuhkan ketika penulis memiliki banyak database yang berisi berbagai tabel. Terkadang penulis memiliki nama *trigger* yang sama di banyak basis data; kueri ini memainkan peran penting dalam kasus ini. Penulis bisa mendapatkan informasi *trigger* di *server database* menggunakan pernyataan di bawah ini. Pernyataan ini mengembalikan semua *Trigger* di semua basis data:

```
mysql>SHOW TRIGGERS;
```

Langkah-langkah berikut diperlukan untuk mendapatkan daftar semua *Trigger*:

Langkah 1: Buka command prompt MySQL dan masuk ke server database menggunakan kata sandi yang telah penulis buat saat instalasi MySQL. Setelah koneksi berhasil, penulis dapat menjalankan semua pernyataan SQL.

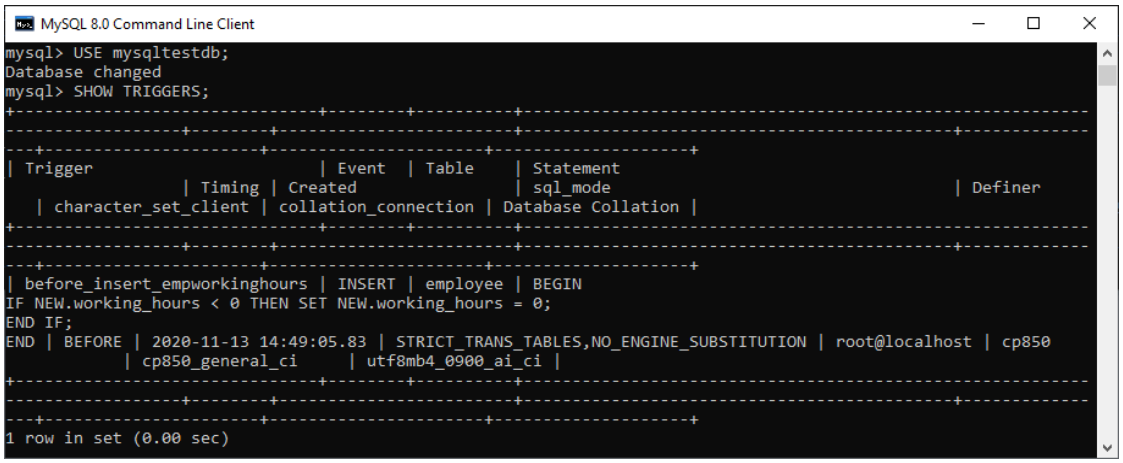
Langkah 2: Selanjutnya, pilih *database* tertentu dengan menggunakan perintah di bawah ini:

```
mysql> USE database_name;
```

Langkah 3: Terakhir, jalankan perintah *SHOW TRIGGER*S. Mari penulis memahaminya dengan contoh yang diberikan di bawah ini. Misalkan penulis memiliki nama *database* "mysqltestdb" yang berisi banyak tabel. Kemudian jalankan pernyataan di bawah ini untuk mencantumkan *trigger*:

```
mysql> USE mysqltestdb;
mysql>SHOW TRIGGERS;
```

Output berikut menjelaskannya dengan lebih jelas dapat dilihat pada gambar 2.1 :



Gambar 2.1 contoh show trigger

Jika penulis ingin menampilkan atau mencantumkan informasi *trigger* dalam *database* tertentu dari *database* saat ini tanpa berpindah, MySQL memungkinkan penulis untuk menggunakan klausa *FROM* atau *IN*, diikuti dengan nama *database*. Pernyataan berikut menjelaskannya dengan lebih jelas:

```
mysql> SHOW TABLES IN database_name;
```

Pernyataan di atas juga dapat ditulis sebagai:

```
mysql> SHOW TABLES FROM database_name;
```

Ketika penulis menjalankan pernyataan di atas, penulis akan mendapatkan hasil yang sama.

c. MySQL DROP Trigger

Penulis dapat *drop/delete/remove Trigger* di MySQL menggunakan pernyataan *DROP TRIGGER*. Penulis harus sangat berhati-hati saat menghapus *Trigger* dari tabel. Karena setelah penulis menghapus *Trigger*-nya, itu tidak dapat dipulihkan. Jika *trigger* tidak ditemukan, pernyataan *DROP TRIGGER* melontarkan kesalahan.

MySQL memungkinkan penulis untuk *drop/delete/remove Trigger* salah satumya dengan cara MySQL *Command Line Client*

Penulis dapat menghapus *Trigger* yang ada dari *database* dengan menggunakan pernyataan *DROP TRIGGER* dengan sintaks di bawah ini:

```
DROP TRIGGER [IF EXISTS] [schema_name.]Trigger_name;
```

Penjelasan Parameter :

Parameter yang digunakan dalam sintaks DROP *trigger* dapat dilihat pada tabel 2.2:

Tabel 2.2 Parameter

Parameter	**Schema_name**
*Trigger*_name	Itu adalah nama *trigger* yang ingin penulis hapus dari server database. Ini adalah parameter yang diperlukan.
Schema_name	Ini adalah nama database tempat *trigger* berada. Jika kita melewatkan parameter ini, pernyataan akan menghapus *trigger* dari database saat ini.
IF_EXISTS	Ini adalah parameter opsional yang secara kondisional menghapus *trigger* hanya jika ada di server database.

Jika penulis menghapus *trigger* yang tidak ada, penulis akan mendapatkan kesalahan. Namun, jika penulis telah menentukan klausa *IF EXISTS*, MySQL memberikan NOTE alih-alih kesalahan.

Perlu dicatat bahwa penulis harus memiliki hak istimewa *TRIGGER* sebelum mengeksekusi pernyataan *DROP TRIGGER* untuk tabel yang terkait dengan *trigger*. Selain itu, menghapus tabel akan secara otomatis menghapus semua *trigger* yang terkait dengan tabel tersebut.

Contoh MySQL *DROP Trigger*;

Mari penulis lihat bagaimana penulis bisa menghapus *Trigger* yang terkait dengan tabel melalui sebuah contoh. Jadi pertama, penulis akan menampilkan semua *Trigger* yang tersedia di database yang dipilih menggunakan pernyataan di bawah ini:

```
mysql> SHOW TRIGGERS IN employeedb;
```

Setelah menjalankan pernyataan tersebut, penulis dapat melihat bahwa ada dua *Trigger* bernama before_update_salaries dan sales_info_before_update,dapat dilihat pada gambar 2.2:

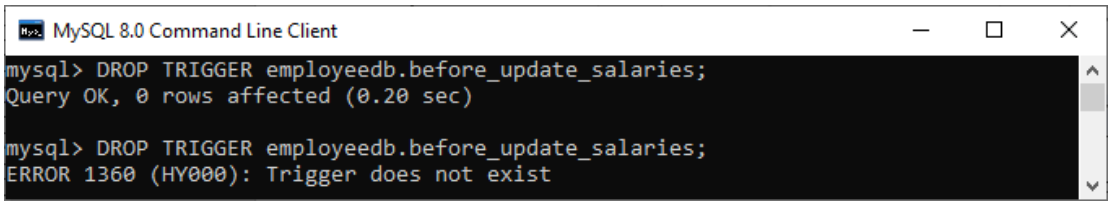
Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer
before_update_salaries	UPDATE	salary_account	BEGIN IF new.amount < old.a...	BEFORE	2020-11-2...	STRICT_TRA...	root@localhost
sales_info_BEFORE_UPDATE	UPDATE	sales_info	BEGIN DECLARE error_msg VARC...	BEFORE	2020-11-2...	STRICT_TRA...	root@localhost

Gambar 2.2 contoh drop trigger pertama

Jika penulis ingin menghapus *trigger* before_update_salaries, jalankan pernyataan di bawah ini:

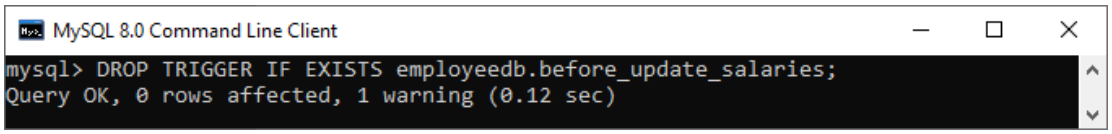
```
mysql> DROP TRIGGER employeedb.before_update_salaries
```

Ini akan berhasil menghapus *trigger* dari *database*. Jika penulis menjalankan pernyataan di atas lagi, maka akan mengembalikan pesan kesalahan. Lihat hasilnya pada gambar 2.3.



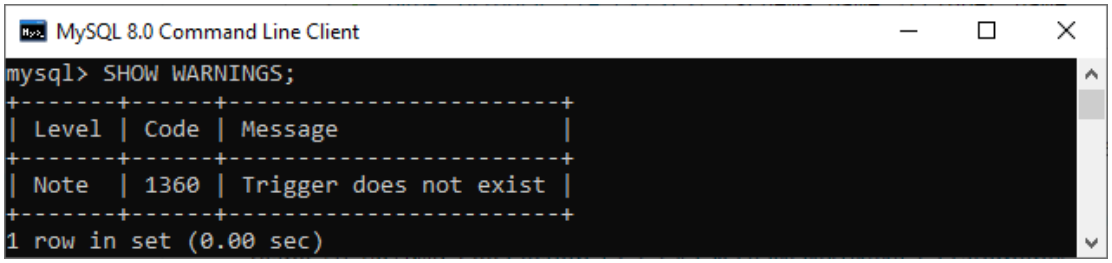
Gambar 2.3 contoh drop trigger kedua

dapat dilihat pada gambar 2.4.



Gambar 2.4 contoh drop trigger ketiga

Penulis dapat mengeksekusi pernyataan SHOW WARNING yang menghasilkan NOTE untuk *trigger* yang tidak ada saat menggunakan IF EXISTS. Dapat dilihat pada gambar 2.5 (javatpoint.com, 2022).



Gambar 2.5 contoh drop trigger ketiga