Trigger Pada Oracle 10g

Mudafiq Riyan Pratama
mudafiq.riyan@yahoo.com
http://dhafiq-san.blogspot.com/

Pendahuluan

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. Implementasi trigger yang sering ditemui dalam dunia nyata adalah untuk mengeset dan mengubah nilai kolom dalam suatu tabel sehingga validasi nilai dari tabel tersebut akan terjaga. Adanya trigger dalam database akan meringankan kita dalam pembuatan aplikasi karena di dalam aplikasi yang kita buat, kita tidak perlu lagi untuk melakukan validasi data.

Membuat Trigger

Oracle 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.

Daftar event yang mungkin digunakan untuk mengaktifkan trigger:

Nama Event	Keterangan
BEFORE INSERT	Diaktifkan sekali sebelum statemen INSERT
AFTER INSERT	Diaktifkan sekali setelah statemen INSERT
BEFORE UPDATE	Diaktifkan sekali sebelum statemen UPDATE
AFTER UPDATE	Diaktifkan sekali setelah statemen UPDATE
BEFORE DELETE	Diaktifkan sekali sebelum statemen DELETE
AFTER DELETE	Diaktifkan sekali setelah statemen DELETE

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.

Format Penggunaan Trigger Pada Umumnya

```
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.

Trigger dapat dinyalakan sebelum (*before*) atau sesudah (*after*) eksekusi statement, dan dapat dinyalakan sekali tiap baris yang dipengaruhi atau sekali tiap statement. Kombinasi dari tiga faktor ini menetukan tipe trigger. Ada total 12 kemungkinan: 3 statement x 2 timing x 2 level.

Tabel berikut menunjukkan tipe dari trigger *Data Manipulating Language*

Kategori	Nilai	Keterangan
Statement	INSERT, DELETE, atau UPDATE	Menentukan jenis statement *Data Manipulation Language* yang menyebabkan trigger dinyalakan.
Timing	BEFORE atau AFTER	Menentukan apakah trigger menyala sebelum atau sesudah statement dieksekusi.
Level	Row atau Statement	Jika trigger adalah row-level, maka akan dinyalakan sekali untuk setiap baris yang dipengaruhi oleh statement trigger. Jika trigger adalah statement-level, maka akan dinyalakan sekali sebelum atau sesudah statement. Trigger row-level diidentifikasikan oleh klausa FOR EACH ROW dalam pendefinisian trigger.

Contoh penggunaan trigger

DOSEN

nip_dosen	nama_dosen	jumlah_wali

MAHASISWA

nrp_mhs	nama_mhs	id_wali

Gambar : contoh basisdata yang digunakan dalam pembuatan trigger

SQL pembuatan tabel dosen:

```
Create table dosen(
        nip_dosen char(10) primary key not null,
        nama_dosen char(20),
        jumlah_wali integer default 0
)
```

SQL pembuatan tabel mahasiswa:

```
Create table mahasiswa(
        nrp_mhs char(10) primary key not null,
        nama_mhs char(20),
        id_wali char(10) not null,
        foreign key (id_wali) references dosen(nip_dosen)
)
```

Pada tabel dosen JUMLAH_WALI diatur dengan nilai awal default null atau 0. NIP_DOSEN merupakan primary key pada tabel dosen dan menjadi references di tabel mahasiswa dengan foreign key ID_WALI dan NRP_MHS sebagai primary key pada tabel mahasiswa. Untuk menjaga agar nilai-nilai derived attribute tetap benar, dapat dilakukan dengan menggunakan trigger. Event-event yang dapat mempengaruhi perubahan nilai dari jumlah_wali sebagai berikut:

1. INSERT mahasiswa baru
2. UPDATE mahasiswa dari id_wali satu ke id_wali yang lain
3. DELETE mahasiswa yang ada

Ketiga event diatas akan digabungkan menjadi satu dalam sebuah trigger dengan SQL sebagai berikut:

```
CREATE OR REPLACE TRIGGER nambah
After
Begin
        If inserting then
                Update dosen
                Set jumlah_wali = jumlah_wali+1
                Where nip_dosen = :new.id_wali;
        End if;

        If deleting then
                Update dosen
                Set jumlah_wali = jumlah_wali-1
                Where nip_dosen = :old.id_wali;
        End if;

        If updating then
                Begin
                        Update dosen
                        Set jumlah_wali = jumlah_wali+1
                        Where nip_dosen = :new.id_wali;
                        Update dosen
                        Set jumlah_wali = jumlah_wali-1
                        Where nip_dosen = :old.id_walil;
                End;
        End if;
End;
```

Keterangan:

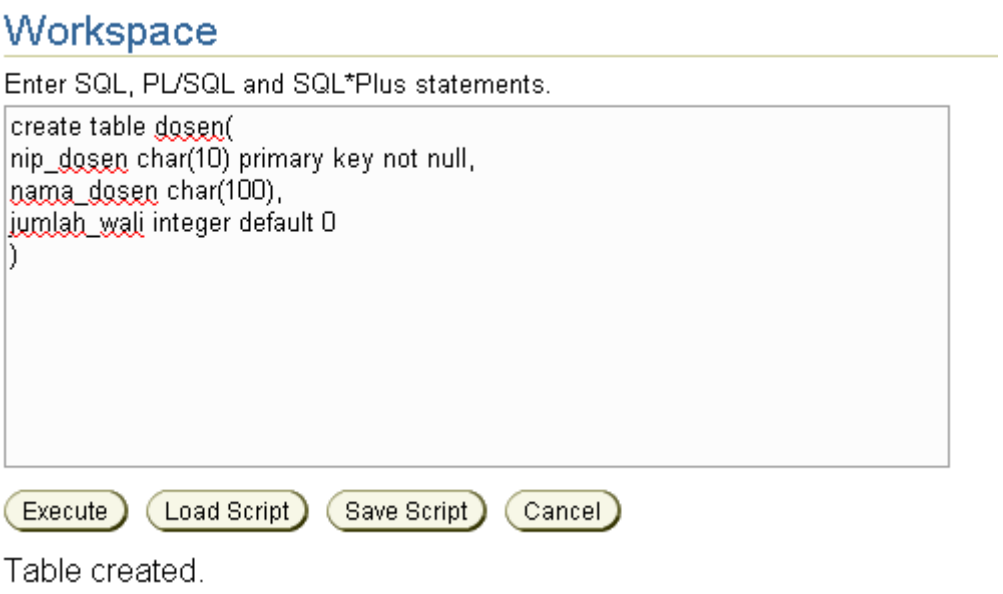
- Nama trigger yaitu **nambah**
- Event after menandakan bahwa trigger ini akan diaktifkan sekali setelah statement yang diinginkan dijalankan (INSERT, UPDATE, DELETE) pada tabel mahasiswa.
- Bila sebuah data di-insert-kan dalam tabel mahasiswa maka tabel dosen akan di update setelah perintahnya dijalankan dengan proses: *set jumlah_wali = jumlah_wali+1* dengan syarat *nip_dosen = :new.id_wali*.
- Bila data di delete, maka proses yang akan dijalankan oleh trigger setelah proses tersebut adalah: *set jumlah_wali = jumlah_wali-1* dengan syarat *nip_dosen = :old.id_wali*
- Bila data update, maka proses yang akan dilakukan oleh trigger yaitu menambahkan data kemudian menghapus data yang lama.

Penerapan Trigger

Berdasarkan contoh kasus di atas, mari kita memakai seperti kasus di atas, mempelajari mengenai kasus untuk dosen wali mahasiswa. Data yang ada pada dosen adalah jumlah mahasiwa yang menjadi wali dosen tersebut. Sedangkan pada mahasiswa, hanya menyantumkan NIP dari dosen walinya. Sehingga ketika ada mahasiswa yang ditambahkan, tentunya dosen wali yang bersangkutan akan menambahkan pula jumlah mahasiswa yang di walikan. Berikut tahapan-tahapan yang perlu dilakukan:

1. Untuk itu perlu membuat tabel dosen terlebih dahulu dengan query:

```
create table dosen(
nip_dosen char(10) primary key not null,
nama_dosen char(100),
jumlah_wali integer default 0
)
```



Tabel dosen memiliki kolom-kolom **nip_dosen, nama_dosen,** dan **jumlah_wali.** Yang mana kolom **jumlah_wali** ini sifatnya dinamis sesuai data mahasiswa yang di insert-kan. Sehingga ketika data mahasiswa di insert-kan, maka jumlah_wali pada dosen bersangkutan akan bertambah. Kolom jumlah_wali ini atributnya integer dan sifatnya default 0, yaitu ketika tidak ada data yang dimasukkan, maka data defaultnya akan bernilai 0.

2. Kemudian membuat tabel mahasiswa dengan query:

   create table mahasiswa(

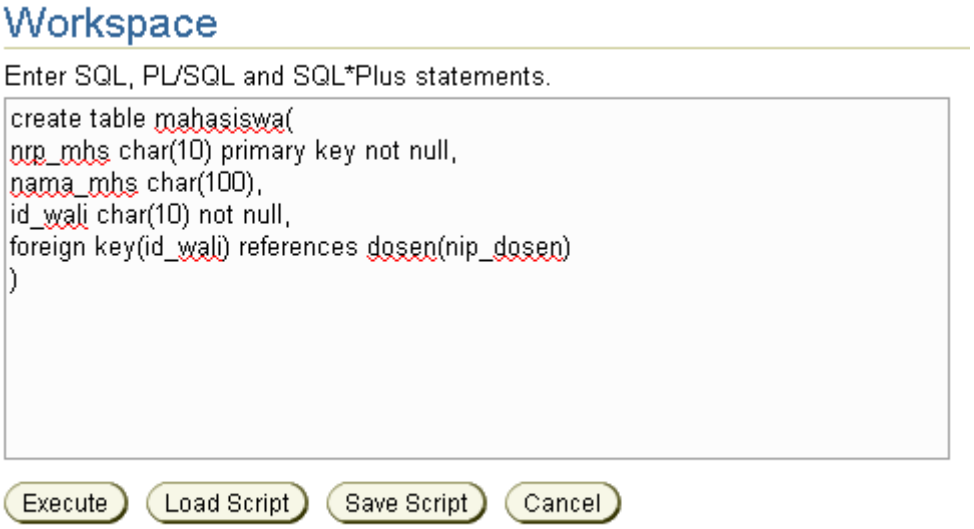
   nrp_mhs char(10) primary key not null,

   nama_mhs char(100),

   id_wali char(10) not null,

   foreign key(id_wali) references dosen(nip_dosen)

)



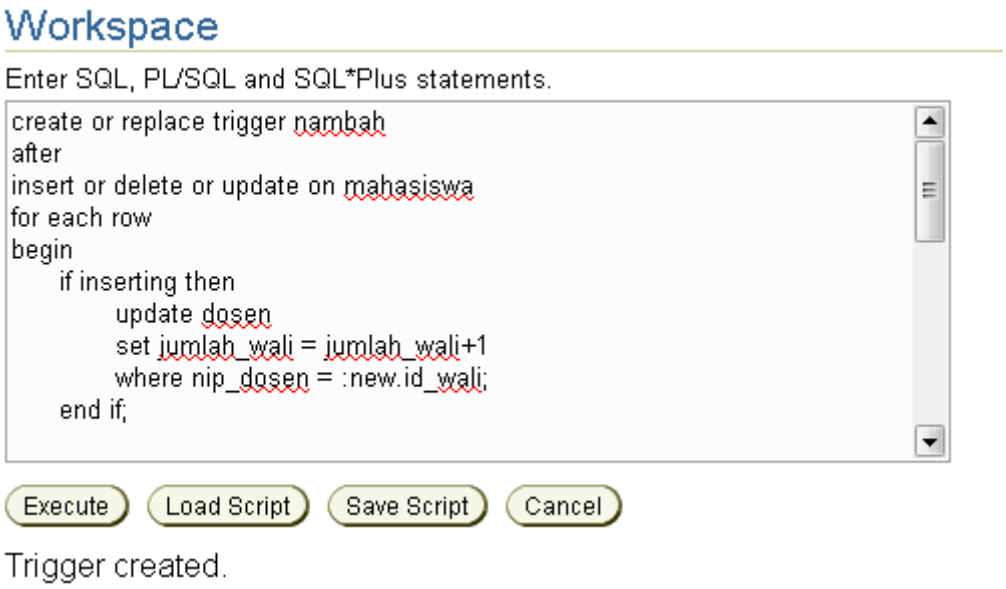
Tabel mahasiswa ini memiliki kolom-kolom: **nrp_mhs, nama_mhs, id_wali.** Yang mana id_wali dari tabel mahasiswa merupakan foreign key dari nip_dosen dari tabel dosen. Sehingga untuk memasukkan id_wali harus sesuai dengan data yang ada pada nip_dosen. Dan id_wali tidak boleh kosong, wajib diisi karena sifatnya not null.

3. Pada tahap ke-3 ini, kita akan membuat script untuk trigger. Berikut scriptnya:

```
create or replace trigger nambah
after
insert or delete or update on mahasiswa
for each row
begin
        if inserting then
                update dosen
                set jumlah_wali = jumlah_wali+1
                where nip_dosen = :new.id_wali;
        end if;

        if deleting then
                update dosen
                set jumlah_wali = jumlah_wali-1
                where nip_dosen = :old.id_wali;
        end if;

        if updating then
                begin
                update dosen
                set jumlah_wali = jumlah_wali+1
                where nip_dosen = :new.id_wali;
                update dosen
                set jumlah_wali = jumlah_wali-1
                where nip_dosen = :old.id_wali;
                end;
        end if;
end;
```



Script di atas merupakan pembuatan trigger untuk penanganan ketika data mahasiswa di insertkan, maka data pada jumlah_wali (tabel dosen) akan ikut terupdate juga.

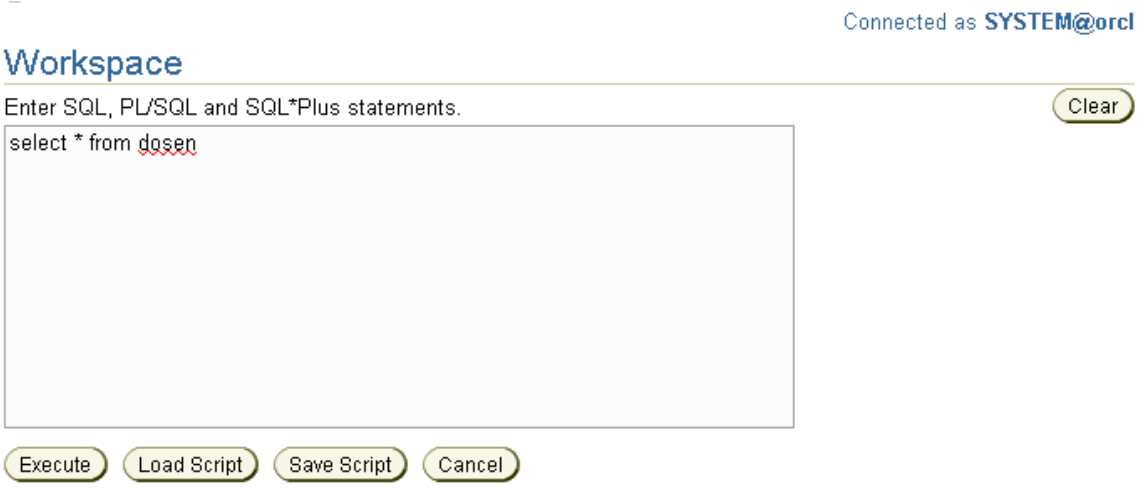
Maksud dari script trigger diatas adalah:

- Nama trigger yaitu **nambah**
- Event after menandakan bahwa trigger ini akan diaktifkan sekali setelah statement yang diinginkan dijalankan (INSERT, UPDATE, DELETE) pada tabel mahasiswa.
- Bila sebuah data di-insert-kan dalam tabel mahasiswa maka tabel dosen akan di update setelah perintahnya dijalankan dengan proses: *set jumlah_wali = jumlah_wali+1* dengan syarat *nip_dosen = :new.id_wali*.
- Bila data di delete, maka proses yang akan dijalankan oleh trigger setelah proses tersebut adalah: *set jumlah_wali = jumlah_wali-1* dengan syarat *nip_dosen = :old.id_wali*
- Bila data update, maka proses yang akan dilakukan oleh trigger yaitu menambahkan data kemudian menghapus data yang lama.

Pembuktian:

1. Kita memasukkan list data untuk dosen. Berikut query nya:
    - *insert into dosen (nip_dosen, nama_dosen) values ('001', 'Hariyadi,S.Kom')*
    - *insert into dosen (nip_dosen, nama_dosen) values ('002', 'Setio Basuki')*
    - *insert into dosen (nip_dosen, nama_dosen) values ('003', 'Yuda Munarko')*
    - *insert into dosen (nip_dosen, nama_dosen) values ('004', 'Gita Indah')*
    - *insert into dosen (nip_dosen, nama_dosen) values ('005', 'Mahar Faiq')*

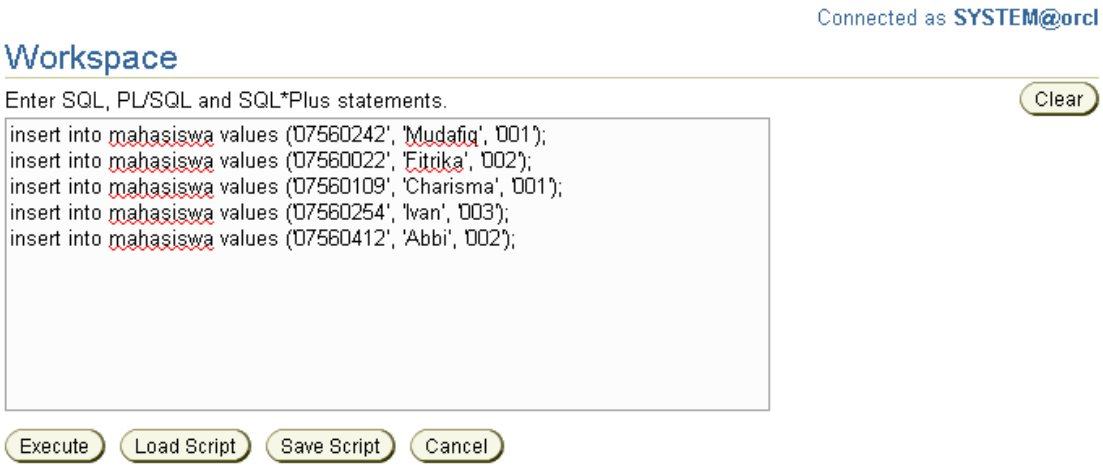
2. Setelah meng-inputkan data-data mengenai dosen, kita lihat isi dari tabel dosen mengunakan query: *select * from dosen*



NIP_DOSEN	NAMA_DOSEN	JUMLAH_WALI
001	Hariyadi,S.Kom	0
002	Setio Basuki	0
003	Yuda Munarko	0
004	Gita Indah	0
005	Mahar Faiq	0

Dari data-data yang sudah dimasukkan, lihatlah pada kolom **jumlah_wali**, isinya adalah 0 karena default untuk data kosong adalah 0. Karena data mahasiswa juga belum di insert-kan.

3. Mari insert-kan data beberapa mahasiswa menggunakan query-query berikut:
    - *insert into mahasiswa values ('07560242', 'Mudafiq', '001');*
    - *insert into mahasiswa values ('07560022', 'Fitrika', '002');*
    - *insert into mahasiswa values ('07560109', 'Charisma', '001');*
    - *insert into mahasiswa values ('07560254', 'Ivan', '003');*
    - *insert into mahasiswa values ('07560412', 'Abbi', '002');*



1 row created.

1 row created.

1 row created.

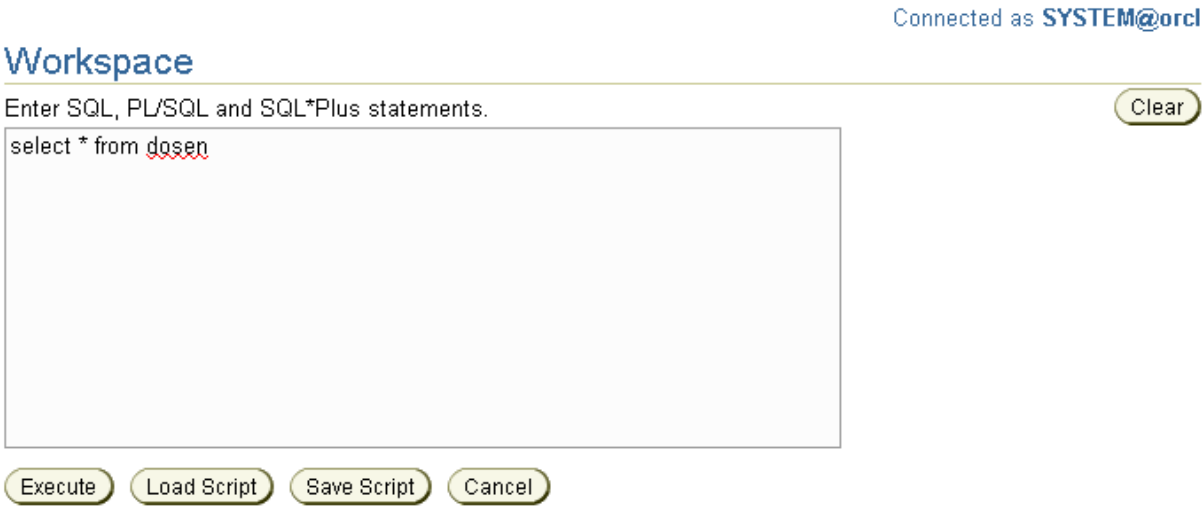
1 row created.

1 row created.

Tiap mahasiswa yang berbeda telah di-insert-kan data berdasarkan dosen walinya masing-masing. Pada kasus ini:

- Mudafiq, dosen walinya adalah nip_dosen=001
- Fitrika, dosen walinya adalah nip_dosen=002
- Charisma, dosen walinya adalah nip_dosen=001
- Ivan, dosen walinya adalah nip_dosen=003
- Abbi, dosen walinya adalah nip_dosen=002

4. Kemudian kita periksa kembali isi data dari tabel dosen menggunakan query: *select * from dosen*



NIP_DOSEN	NAMA_DOSEN	JUMLAH_WALI
001	Hariyadi,S.Kom	2
002	Setio Basuki	2
003	Yuda Munarko	1
004	Gita Indah	0
005	Mahar Faiq	0

Jelas bahwa data dari tabel dosen juga terupdate. Lihat pada jumlah_wali yang semulai nilainya 0 semua, sekarang dengan penambahan data-data dari tabel mahasiswa, pada tabel dosen juga terisi untuk jumlah_wali sesuai dengan nama mahasiswa dan dosen walinya masing-masing. Sehingga dapat diketahui bahwa dosen ini jumlah mahasiswa yang ditangani berapa. Dengan begitu semakin memudahkan programmer aplikasi untuk pembuatan sistem.

=0=0=0=0=0=0=0=0= **Selamat Mencoba** **=0=0=0=0=0=0=0=0=**

Biografi Penulis

Mudafiq Riyan Pratama. Lahir di Jember pada tanggal 9 Mei 1989. Kediaman di Jember. Memulai pendidikan TK dan SD di Jenggawah. Kemudian menempuh SMP di SMPN 6 Jember yang kemudian dilanjutkan ke SMAN 2 Jember. Dan saat ini sedang menempuh kuliah S1 jurusan Teknik Informatika di Universitas Muhammadiyah Malang angkatan 2007.

Didunia maya, penulis lebih sering memakai nama **Dhafiq Sagara**.

YM : mudafiq.riyan@yahoo.com
FB : ray_dafier@yahoo.co.id
Blog : http://dhafiq-san.blogspot.com