Bab 5

Database Trigger

POKOK BAHASAN:

- ✓ Pembuatan dan Penggunaan Trigger
- ✓ Statement trigger
- ✓ Row Trigger
- ✓ Menggunakan Old dan New Qualifiers
- ✓ Klausa WHEN pada trigger
- ✓ Perintah-perintah umum pada Trigger

TUJUAN BELAJAR:

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

- ✓ Memahami macam-macam tipe trigger
- ✓ Memahami trigger dan penggunaannya
- ✓ Dapat membuat database trigger
- ✓ Memahami database trigger yang dapat mengaktifkan sebuah aturan
- ✓ Menghapus database trigger

5.1. PENDAHULUAN

Trigger adalah blok PL/SQL atau prosedur yang berhubungan dengan table, view, skema atau database yang dijalankan secara implicit pada saat terjadi sebuah event.

Tipe dari trigger adalah :

- Application trigger : diaktifkan pada saat terjadi event yang berhubungan dengan sebuah aplikasi
- Database trigger : diaktifkan pada saat terjadi event yang berhubungan dengan data (seperti operasi DML) atau event yang berhubungan dengan sistem (semisal logon atau shutdown) yang terjadi pada sebuah skema atau database.

5.2. PENGGUNAAN TRIGGER

Trigger dibuat sesuai dengan keperluan. Ada kalanya trigger perlu dibuat, dan kadangkala tidak perlu dibuat.

Trigger perlu dibuat pada saat :

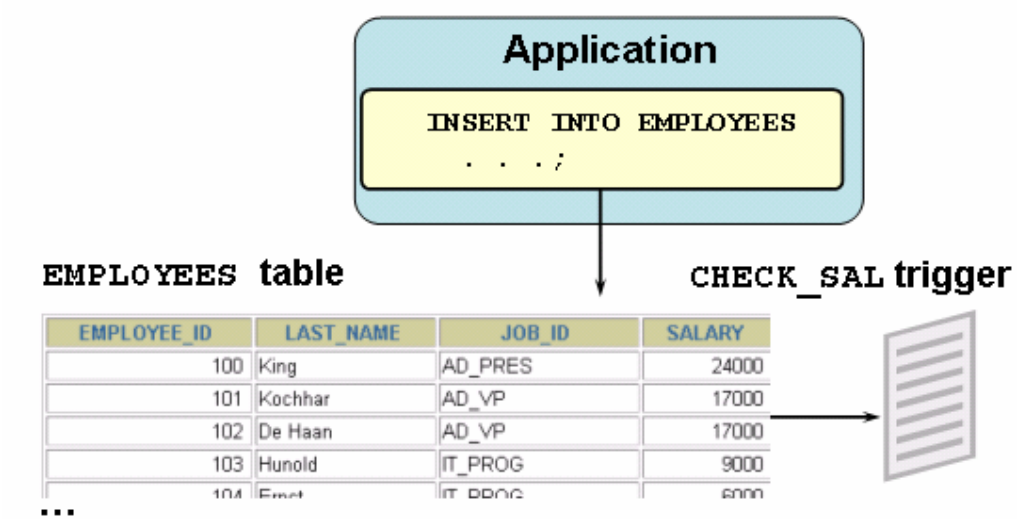
- membentuk sebuah aksi tertentu terhadap suatu event
- Memusatkan operasi global

Trigger tidak perlu dibuat, jika :

- Fungsionalitas yang diperlukan suatu ada pada Oracle server
- Duplikat atau sama dengan fungsi trigger yang lain.

Prosedur bisa dibuat dalam database, kemudian prosedur tersebut dipanggil pada trigger. Jika penggunaan trigger terlalu berlebihan, maka akan menyebabkan terjadi sifat ketidaktergantungan yang terlalu kompleks sehingga akan mempersulit pemeliharaan dari aplikasi yang besar.

Gambar berikut ini menunjukkan ilustrasi dari penggunaan trigger :



Gambar 5.1. Penggunaan Trigger

Pada gambar tersebut, database trigger CHECK_SAL memeriksa nilai gaji pada saat suatu aplikasi mencoba untuk memasukkan baris baru ke dalam table

EMPLOYEES. Nilai yang terletak pada jangkauan diluar kategori pekerjaan akan diabaikan.

Sintak penulisan dari database trigger, berisi komponen berikut :

1. Trigger timing :
    a. Untuk tabel : BEFORE, AFTER
    b. Untuk view : INSTEAD OF
2. Trigger event : INSERT, UPDATE atau DELETE
3. Nama tabel : yaitu nama tabel atau view yang berhubungan dengan trigger
4. Tipe trigger : Baris atau Pernyataan (statement)
5. klausa WHEN : untuk kondisi pembatasan
6. trigger body : bagian prosedur yang dituliskan pada trigger

5.3. KOMPONEN TRIGGER

Komponen dari sebuah trigger ada 6 (enam), yaitu : trigger timing, trigger event, nama tabel, tipe trigger, klausa WHEN, dan trigger body. Berikut ini penjelasan komponen dari trigger.

Trigger timing adalah waktu kapan trigger diaktifkan. Ada tiga macam trigger timing, yaitu :

- BEFORE : trigger dijalankan sebelum DML event pada tabel
- AFTER : trigger dijalankan setelah DML event pada tabel
- INSTEAD OF : trigger dijalankan pada sebuah view.

Trigger event ada 3 kemungkinan : INSERT, UPDATE atau DELETE.

Pada saat trigger event UPDATE, kita dapat memasukkan daftar kolom untuk mengidentifikasi kolom mana yang berubah untuk mengaktifkan sebuah trigger (contoh : UPDATE OF salary ...). Jika tidak ditentukan, maka perubahannya akan berlaku untuk semua kolom pada semua baris.

Tipe trigger ada 2 macam, yaitu :

- Statement : trigger dijalankan sekali saja pada saat terjadi sebuah event. Statement trigger juga dijalankan sekali, meskipun tidak ada satupun baris yang dipengaruhi oleh event yang terjadi.

- Row : trigger dijalankan pada setiap baris yang dipengaruhi oleh terjadinya sebuah event. Row trigger tidak dijalankan jika event dari trigger tidak berpengaruh pada satu baris pun.

Trigger body mendefinisikan tindakan yang perlu dikerjakan pada saat terjadinya event yang mengakibatkan sebuah trigger menjadi aktif.

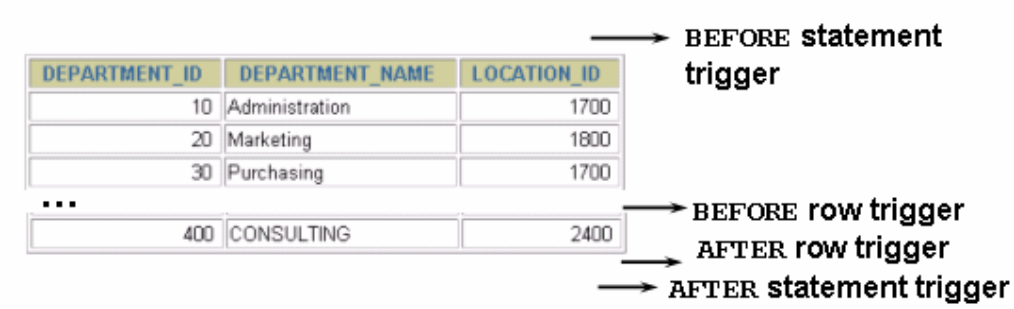
5.4. CONTOH PEMBUATAN TRIGGER

Contoh berikut ini akan mengaktifkan sebuah trigger pada saat sebuah baris tunggal dimanipulasi pada tabel :

Misal diberikan perintah DML untuk menyisipkan baris baru ke dalam tabel sebagai berikut :

INSERT INTO departments (department_id, department_name, location_id)
VALUES (400, 'CONSULTING', 2400);

Ilustrasi dari trigger timing untuk event tersebut adalah sebagai berikut :

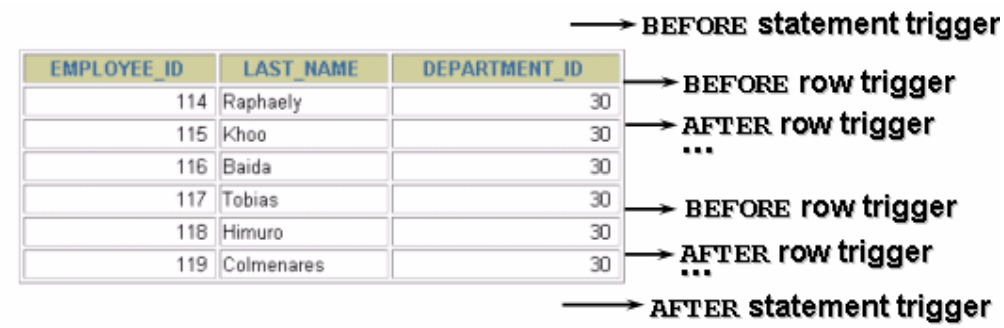


Gambar 5.2. Ilustrasi *timing* pada Trigger

Jika DML statement berlaku untuk lebih dari satu baris yang ada pada tabel (multiple row), semisal :

UPDATE employees
SET salary = salary * 1.1
WHERE department_id = 30;

Maka ilustrasi dari trigger timing untuk event tersebut adalah sebagai berikut :



EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
114	Raphaely	30
115	Khoo	30
116	Baida	30
117	Tobias	30
118	Himuro	30
119	Colmenares	30

5.5. DML STATEMENT TRIGGER

Berikut ini sintak atau cara penulisan untuk pembuatan DML Statement trigger :

CREATE [OR REPLACE] TRIGGER *trigger_name*
timing
***event1* [OR *event2* OR *event3*]**
ON *table_name*
trigger_body

Berikut contoh pembuatan DML Statement trigger :

```
CREATE OR REPLACE TRIGGER secure_emp
 BEFORE INSERT ON employees
 BEGIN
 IF (TO_CHAR(SYSDATE,'DY') IN ('SAT','SUN')) OR
   (TO_CHAR(SYSDATE,'HH24:MI') NOT BETWEEN '08:00' AND '18:00')
  THEN RAISE_APPLICATION_ERROR (-20500,'Penyisipan data pada table
EMPLOYEES hanya diperbolehkan selama jam kerja');
 END IF;
END;
/
```

Contoh trigger diatas akan membatasi penyisipan baris baru ke dalam table EMPOYEES diperbolehkan hanya pada jam kerja mulai hari Senin sampai Jum'at. Jika

user menyisipkan baris baru diluar ketentuan tersebut, missal pada hari SAbtu maka akan tampil pesan kesalahan.

Perintah berikut ini akan menguji trigger SECURE_EMP dengan memberikan perintah SQL berikut ini pada jam diluar jam kerja, sebagai berikut :

```
INSERT INTO employees (employee_id, last_name,first_name, email, hire_date,
job_id, salary, department_id)
VALUES (300, 'Smith', 'Rob', 'RSMITH', SYSDATE,'IT_PROG', 4500, 60);
```

Perintah tersebut akan memberikan pesan kesalahan :

```
INSERT INTO employees (employee_id, last_name, first_name, email,
       *
ERROR at line 1:
ORA-20500: You may insert into EMPLOYEES table only during business hours.
ORA-06512: at "PLSQL.SECURE_EMP", line 4
ORA-04088: error during execution of trigger 'PLSQL.SECURE_EMP'
```

5.6. MENGKOMBINASIKAN EVENT PADA TRIGGER

Beberapa event pada trigger bisa dikombinasikan dalam sebuah trigger dengan menggunakan predikat kondisional INSERTING, UPDATING dan DELETING. Berikut ini akan dibuat trigger yang menggunakan predikat kondisional INSERTING, UPDATING dan DELETING untuk membatasi manipulasi data pada tabel EMPLOYEES hanya diperbolehkan pada setiap jam kerja mulai hari Senin sampai Jum'at.

```
BEFORE INSERT OR UPDATE OR DELETE ON employees
BEGIN
 IF (TO_CHAR (SYSDATE,'DY') IN ('SAT','SUN')) OR
   (TO_CHAR (SYSDATE, 'HH24') NOT BETWEEN '08' AND '18')
 THEN
  IF     DELETING THEN
    RAISE_APPLICATION_ERROR (-20502,'You may delete  from
EMPLOYEES table only during business hours.');
  ELSIF  INSERTING THEN
    RAISE_APPLICATION_ERROR (-20500,'You may insert into
           EMPLOYEES table only during business hours.');
  ELSIF   UPDATING ('SALARY') THEN
```

```
    RAISE_APPLICATION_ERROR (-20503,'You may update
                  SALARY only during business hours.');
  ELSE
    RAISE_APPLICATION_ERROR (-20504,'You may update
        EMPLOYEES table only during normal hours.');
  END IF;
 END IF;
END;
```

5.7. ROW TRIGGER

Berikut ini sintak atau cara penulisan untuk membuat Row Trigger :

CREATE [OR REPLACE] TRIGGER ***trigger_name***

timing

event1 **[OR** ***event2*** **OR** ***event3*****]**

ON ***table_name***

[REFERENCING OLD AS ***old*** **| NEW AS** ***new*****]**

FOR EACH ROW

[WHEN (*condition***)]**

trigger_body

Contoh berikut ini akan dibuat row trigger dengan timing BEFORE untuk membatasi operasi DML pada table EMPLOYEES hanya diperbolehkan untuk pegawai yang memiliki kode pekerjaan ‘AD_PRES’ dan ‘AD_VP’ serta memiliki gaji kurang dari 15000.

```
CREATE OR REPLACE TRIGGER restrict_salary
 BEFORE  INSERT OR UPDATE OF salary ON employees
  FOR EACH ROW
  BEGIN
   IF  NOT (:NEW.job_id IN ('AD_PRES', 'AD_VP'))
      AND :NEW.salary > 15000
   THEN
    RAISE_APPLICATION_ERROR (-20202,'Employee
                                        cannot earn this amount');
   END IF;
END;
/
```

Jika kita mencoba memberikan perintah SQL sebagai berikut, maka akan ditampilkan pesan kesalahan :

```
UPDATE employees
SET salary = 15500
WHERE last_name = 'Russell';
```

5.8. MENGGUNAKAN OLD DAN NEW QUALIFIERS

Pada Row Trigger, nilai dari kolom sebelum dan sesudah perubahan data dapat dirujuk dengan menggunakan OLD dan NEW *qualifier*. OLD dan NEW hanya digunakan pada Row Trigger. OLD dan NEW menggunakan prefiks (:) untuk pernyataan dalam perintah SQL. Jika qualifier ini terlibat dalam pembatasan kondisi pada klausa WHEN, maka tidak digunakan prefiks (:).

Row triggers akan menurunkan unjuk kerja jika banyak dilakukan update pada table yang cukup besar.

Contoh Trigger berikut ini menggunakan OLD dan NEW qualifier pada Row Trigger :

```
CREATE OR REPLACE TRIGGER audit_emp_values
 AFTER DELETE OR INSERT OR UPDATE ON employees
 FOR EACH ROW
BEGIN
  INSERT INTO audit_emp_table (user_name, timestamp,
    id, old_last_name, new_last_name, old_title,
    new_title, old_salary, new_salary)
  VALUES (USER, SYSDATE, :OLD.employee_id,
     :OLD.last_name, :NEW.last_name, :OLD.job_id,
     :NEW.job_id, :OLD.salary, :NEW.salary );
END;
/
```

Untuk memeriksa hasil dari pembuatan trigger diatas, diberikan perintah SQL sebagai berikut :

```
INSERT INTO employees
          (employee_id, last_name, job_id, salary, ...)
 VALUES (999, 'Temp emp', 'SA_REP', 1000, ...);

UPDATE employees
 SET salary = 2000, last_name = 'Smith'
 WHERE employee_id = 999;
```

```
1 row created.
1 row updated.
```

Hasil dari perintah SQL tersebut adalah akan disimpan record perubahan pada table AUDIT_EMP_TABLE sebagai hasil dari operasi Trigger :

SELECT user_name, timestamp, ... FROM audit_emp_table

USER_NAME	TIMESTAMP	ID	OLD_LAST_N	NEW_LAST_N	OLD_TITLE	NEW_TITLE	OLD_SALARY	NEW_SALARY
PLSQL	28-SEP-01			Temp emp		SA_REP		1000
PLSQL	28-SEP-01	999	Temp emp	Smith	SA_REP	SA_REP	1000	2000

5.9. PENGGUNAAN KLAUSA WHEN PADA TRIGGER

Untuk membatasi operasi trigger hanya pada baris yang memenuhi kondisi tertentu, maka digunakan klausa WHEN. Berikut ini akan dibuat trigger pada tabel EMPLOYEES yang menghitung komisi yang diterima oleh seorang pegawai pada saat sebuah baris ditambahkan ke dalam tabel EMPLOYEES, atau pada saat dilakukan modifikasi pada gaji pegawai.

```
CREATE OR REPLACE TRIGGER derive_commission_pct
  BEFORE INSERT OR UPDATE OF salary ON employees
  FOR EACH ROW
  WHEN (NEW.job_id = 'SA_REP')
BEGIN
  IF  INSERTING
    THEN :NEW.commission_pct := 0;
  ELSIF :OLD.commission_pct IS NULL
    THEN :NEW.commission_pct := 0;
  ELSE
   :NEW.commission_pct := :OLD.commission_pct + 0.05;
  END IF;
END;
/
```

Pada klausa WHEN, penggunaan OLD dan NEW qualifier tidak dengan prefiks (:). Untuk menggunakan NEW qualifier, gunakan BEFORE Row Trigger, jika timing BEFORE pada trigger diatas diganti dengan AFTER, maka akan didapat pesan kesalahan :

```
CREATE OR REPLACE TRIGGER derive_commission_pct*
ERROR at line 1:
ORA-04084: cannot change NEW values for this trigger type
```

5.10. PERINTAH UMUM

Berikut ini perintah-perintah umum yang digunakan pada trigger.

Untuk mengaktifkan atau menonaktifkan database trigger, digunakan perintah :

ALTER TRIGGER *trigger_name DISABLE* | *ENABLE*

Untuk mengaktifkan atau menonaktifkan semua trigger yang berlaku untuk sebuah tabel, digunakan perintah :

ALTER TABLE *table_name DISABLE* | *ENABLE* ALL

Untuk melakukan kompilasi ulang sebuah trigger, digunakan perintah :

ALTER TRIGGER *trigger_name* COMPILE

Untuk menghapus trigger dari database, digunakan perintah :

DROP TRIGGER *trigger_name*

Catatan : Semua trigger yang berlaku pada sebuah tabel akan dihapus pada saat tabel tersebut dihapus dari database.

RINGKASAN:

- Trigger adalah blok PL/SQL atau prosedur yang berhubungan dengan table, view, skema atau database yang dijalankan secara implicit pada saat terjadi event.
- Tipe dari trigger adalah : Application trigger (diaktifkan pada saat terjadi event yang berhubungan dengan sebuah aplikasi) dan database trigger (diaktifkan pada saat terjadi event yang berhubungan dengan data)
- Trigger dibuat pada saat yang tepat jika diperlukan yaitu untuk membentuk sebuah aksi tertentu terhadap suatu event dan memusatkan operasi global
- Penggunaan trigger yang terlalu berlebihan akan menyebabkan terjadi sifat ketidaktergantungan yang terlalu kompleks sehingga akan mempersulit pemeliharaan dari aplikasi yang besar.
- Trigger berisi komponen-komponen : trigger timing, trigger event, nama tabel, tipe trigger, klausa WHEN dan trigger body.
- Beberapa event pada trigger bisa dikombinasikan dalam sebuah trigger dengan menggunakan predikat kondisional INSERTING, UPDATING dan DELETING
- Pada Row Trigger, nilai dari kolom sebelum dan sesudah perubahan data dapat dirujuk dengan menggunakan OLD dan NEW *qualifier*.

LATIHAN SOAL :

1. Perubahan pada data hanya diperbolehkan selama jam kerja dari jam 8:45 pagi sampai 17.30 , dari Senin hingga Jum'at. Buat stored procedure dengan nama SECURE_DML untuk mencegah DML statement dijalankan diluar dari jam kerja, dengan menampilkan pesan "Perubahan pada data hanya diperbolehkan hanya pada jam kerja"
2. Buat statement trigger pada tabel JOBS untuk memanggil prosedur diatas.
3. Implementasikan trigger berikut pada table JOBS sehubungan dengan kenaikan gaji pegawai. Buat stored procedure dengan nama UPD_EMP_SAL untuk mengupdate jumlah gaji. Prosedur ini menerima dua parameter : job id dari gaji yang akan diubah dan nilai minimum salary yang baru. Prosedur ini dijalankan dari trigger yang dibuat pada table JOBS.
4. Lanjutan dari soal nomer 3, buat row trigger dengan nama UPDATE_EMP_SALARY pada table JOBS yang memanggil prosedur UPD_EMP_SAL, pada saat minimum gaji pada table JOBS diubah untuk suatu job ID tertentu.