ORACLE ACADEMY

TRIGGER
(PL/SQL PART 3)

ORACLE DATABASE 10g

Teknik Informatika UNIKOM (2010)
Disusun Oleh : Andri Heryandi, M.T. (andri@heryandi.net)

Definisi Trigger

- Trigger adalah PL/SQL yang diasosiasikan dengan sebuah tabel dan akan aktif (terpicu/trigger) ketika sebuah event terjadi pada tabel tersebut.
- Trigger hanya terjadi ketika ada eksekusi INSERT, DELETE dan UPDATE pada table yang bersangkutan.
- Waktu eksekusi trigger yang mungkin terjadi terdiri dari 2 yaitu BEFORE dan AFTER dari statement SQL-nya.

Contoh Penggunaan Trigger

ORACLE 3

- Memberikan suatu nilai tertentu sebelum suatu data diinsert atau diupdate
- Menyimpan history data suatu tabel. Biasanya digunakan untuk pelacakan perubahan data.
- Melakukan operasi yang tidak disediakan oleh Oracle (misalnya mensimulasikan Foreign Key ON UPDATE CASCADE).
- Melakukan Referential Integrity.

ORACLE ACADEMY Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)

1. Trigger dapat digunakan untuk mengubah data sebelum proses INSERT dilakukan atau untuk memberikan nilai default. Misalnya mengubah data yang diluar nilai yang diperbolehkan. Misalnya, jika ada pengisian nilai di atas 100, maka akan dijadikan 100.
2. Anda dapat menyimpan data suatu record ke tabel lain (misalnya history) sebelum data tersebut diupdate atau didelete. Sehingga semua perubahan data dapat terlacak dari sejak data itu dibuat.
3. Oracle tidak mendukung Foreign Key dengan mode ON UPDATE CASCADE. Dengan trigger anda bisa membuat semacam mekanisme untuk melakukan hal tersebut.
4. Anda bisa membuat referensial integrity seperti foreign key atau constraint dengan trigger.

Sintax Trigger


```
CREATE [OR REPLACE] TRIGGER nama_trigger
 timing  event1 [OR event2 OR event3]
ON nama_objek
[[REFERENCING OLD AS old | NEW AS new]
 FOR EACH ROW
 [WHEN (condition)]]
isi_rigger
```


Komponen sintak pembuatan Trigger adalah :

- **timing** menandakan waktu kapan trigger akan terpicu. Bagian ini dapat diberi **BEFORE** atau **AFTER**
- **event** diisi engan kejadian DML apa yang akan memicu trigger. Bagian ini dapat diisi dengan **SELECT**, **UPDATE**, **DELETE**.
- **Nama_objek** digunakan untuk menentukan objek database yang akan mempunyai trigger. Misalnya nama tabel.
- Untuk trigger baris, anda dapat menentukan:
  - Sebuah klausa **`REFERENCING`** digunakan untuk memberikan nama lain terhadap data baru atau lama (secara default adalah OLD dan NEW)
  - **`FOR EACH ROW`** menentukan bahwa trigger merupakan trigger baris
  - **`WHEN`** digunakan untuk menerapkan kondisi bagian mana yang akan dieksekusi
- **Isi_trigger** adalah aksi yang akan dilakukan oleh trigger. Biasanya dapat berbentuk sebuah anonymous PL/SQL atau pemanggilan procedure dengan CALL.

Mengakses Data Baru dan Lama

ORACLE

- Di dalam trigger, anda dapat mengakses data lama dan data baru. Data lama dapat direference dengan record OLD dan data baru dapat direference dengan record NEW.

OPERASI	NEW (READ/WRITE)	OLD (READ)
INSERT	√	
UPDATE	√	√
DELETE		√

- Untuk mengacu ke sebuah field dapat ditulis dengan NEW.namafield atau OLD.namafield.

Perbedaan Trigger dgn Procedure

Triggers	Procedures
Dibuat dengan CREATE TRIGGER	Dibuat denganCREATE PROCEDURE
Data dictionary yang berisi source code berada di USER_TRIGGERS	Data dictionary yang berisi source code berada di USER_SOURCE.
Dipanggil secara implicit dengan pemanggilan DML	Dipanggil secara explisit dengan menuliskan EXEC
COMMIT, SAVEPOINT, dan ROLLBACK tidak diperbolehkan.	COMMIT, SAVEPOINT, dan ROLLBACK diperbolehkan.

Contoh Kasus Penggunaan Trigger

- Ada sebuah bank mempunyai 2 tabel yaitu tabel Tabungan dan Transaksi
  - Tabel Tabungan berisi data tabungan.

    Dengan Kolom : NoRek, Nama, Kota dan Saldo
  - Tabel Transaksi berisi data transaksi. Transaksi yang tersedia adalah Setoran Tunai (C:Cash), dan Transfer (T:Transfer)

    Dengan Kolom : NoTransaksi, Waktu, NoRek1, Besar, NoRek2

Contoh Kasus Penggunaan Trigger


```
CREATE TABLE TblTabungan(
   NoRek INTEGER PRIMARY KEY,
   Nama VARCHAR(50),
   Kota VARCHAR(30),
   Saldo NUMBER DEFAULT 0,
);
CREATE TABLE TblTransaksi(
   NoTran INTEGER PRIMARY KEY,
   Waktu DATE,
   Jenis CHAR(1),  -- C:Cash, T:Transfer
   NoRek1 INTEGER,
   Besar NUMBER,
   NoRek2 INTEGER  -- Diisi Untuk NoRek Tujuan
);
```

Contoh Kasus Penggunaan Trigger

- Telah tersedia 2 sequence yaitu
  - SeqNoRek : sequence untuk membuat nomor rekening pada tabel tabungan
  - SeqNoTransaksi : sequence untuk membuat nomor transaksi pada tabel transaksi.

```
CREATE SEQUENCE SeqNoRek;
CREATE SEQUENCE SeqNoTransaksi;
```


Contoh Kasus Penggunaan Trigger

- Ada sebuah procedure untuk penambahan tabungan baru. Procedure akan melakukan pemeriksaan data dan pemberian nomor rekening secara autoincrement sebelum tabungan baru disimpan. Pengecekan yang dibuat adalah :
  - Pemeriksaan Nama nasabah
  - Pemeriksaan Kota nasabah

Contoh Kasus Penggunaan Trigger


```
CREATE OR REPLACE
PROCEDURE TambahTabungan(pNama TblTabungan.Nama%TYPE,
                         pKota TblTabungan.Kota%TYPE,
                         pSaldo TblTabungan.Saldo%TYPE)
IS
BEGIN
   IF (pNama IS NULL) OR (TRIM(pNAMA)='') THEN
       RAISE_APPLICATION_ERROR(-20000,'Nama Nasabah Tidak
Boleh Kosong');
   END IF;
   IF (pKota IS NULL) OR (TRIM(pKota)='') THEN
       RAISE_APPLICATION_ERROR(-20001,'Kota Nasabah Tidak
Boleh Kosong');
   END IF;
   INSERT INTO TblTabungan
      VALUES(SeqNoRek.NEXTVAL,pNama,pKota,pSaldo);
   COMMIT;
END;
```


Contoh Kasus Penggunaan Trigger

- Untuk contoh data, ada 4 rekening yang dibuat.

```
EXEC TambahTabungan('AA','Aceh',100000);
EXEC TambahTabungan('Bebe','Bandung',200000);
EXEC TambahTabungan('Cece','Cimahi',300000);
EXEC TambahTabungan('Dede','Dumai',400000);
```

- Data setelah eksekusi penambahan contoh data

```
SELECT * FROM TblTabungan;
```

NOREK	NAMA	KOTA	SALDO
1	AA	Aceh	100000
2	Bebe	Bandung	200000
3	Cece	Cimahi	300000
4	Dede	Dumai	400000

Contoh Trigger 1

- Buat suatu trigger untuk mendefaultkan nilai saldo ketika ada penambahan data tabungan baru. Jika saldo yang diinputkan kurang dari nol (0) maka harus dianggap/diganti dengan nol.

Contoh Trigger 1

ORACLE 14

- Sebelum ada trigger :

```
EXEC TambahTabungan('Erni','Ende',-10000);
```

PL/SQL procedure successfully completed.

```
SELECT * FROM TblTabungan;
```

NOREK	NAMA	KOTA	SALDO
1	AA	Aceh	100000
2	Bebe	Bandung	200000
3	Cece	Cimahi	300000
4	Dede	Dumai	400000
5	Erni	Ende	-10000

ORACLE ACADEMY Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)

Dalam contoh ini ada pemanggilan pembuatan tabungan baru dengan nilai saldo yang negatif. Karena tidak ada pengecekan saldo dalam procedure TambahTabungan, maka saldo yang negatif dianggap data yang valid sehingga data baru tersebut tersimpan di tabel.

Ada 2 cara yang bisa dilakukan adalah

1. Menambah validasi saldo pada procedure TambahTabungan
2. Membuat trigger yang akan memperbaiki nilai tabungan jika salah (negatif) menjadi nilai yang valid misalnya nilai 0 (nol).

Contoh Trigger 1

- Pembuatan trigger :

```
CREATE OR REPLACE TRIGGER trig_fix_saldo
BEFORE INSERT ON TblTabungan
FOR EACH ROW
BEGIN
   IF :NEW.Saldo<0 THEN
       :NEW.Saldo:=0;
   END IF;
END;
```

Trigger created.

Keterangan trigger :

1. Nama trigger adalah trig_fix_saldo
2. Trigger akan dipicu sebelum (**BEFORE**) terjadi penyisipan data baru (**INSERT**) pada tabel **TblTabungan**
3. Untuk setiap baris baru
4. Jika saldo yang diinsertkan (:NEW.Saldo) kurang dari 0 (nol), maka isilah saldo yang akan diinsertkan dengan nol (:NEW.Saldo:=0).

Contoh Trigger 1

- Sebelum ada trigger :

```
EXEC TambahTabungan('Ferdi','Fakfak',-10000);
```

PL/SQL procedure successfully completed.

```
SELECT * FROM TblTabungan;
```

NOREK	NAMA	KOTA	SALDO
1	AA	Aceh	100000
2	Bebe	Bandung	200000
3	Cece	Cimahi	300000
4	Dede	Dumai	400000
5	Erni	Ende	-10000
6	Ferdi	Fakfak	0

Contoh Trigger 2

- Buatlah suatu history kota nasabah. Jika ada perubahan kota nasabah, maka nama kota yang diubah (lama) harus disimpan disebuah tabel khusus.
- Hal ini dibuat untuk membuat suatu penelurusan lokasi nasabah.

Keterangan:

Untuk contoh kali ini data yang akan disimpan hanya kota saja. Anda dapat menyimpan history semua kolom dari tabel nasabah

Contoh Trigger 2

- Pembuatan Tabel TblHistoryKota

```
CREATE TABLE TblHistoryKota(
   Waktu TIMESTAMP,
   NoRek INTEGER,
   Kota VARCHAR(30)
);
```


Contoh Trigger 2

- Pembuatan Trigger

```
CREATE OR REPLACE TRIGGER trig_history_kota
AFTER UPDATE ON TblTabungan
FOR EACH ROW
BEGIN
   IF UPPER(:NEW.Kota)<>UPPER(:OLD.Kota) THEN
       INSERT INTO TblHistoryKota
          VALUES(SYSDATE,:OLD.NoRek,:OLD.Kota);
   END IF;
END;
```

Trigger created.

Keterangan :
Pencatatan history kota hanya akan dilakukan jika kota yang baru tidak sama dengan kota lamanya (**`UPPER(:NEW.Kota)<>UPPER(:OLD.Kota)`**)

Contoh Trigger 2

- Pengujian Trigger

```
SELECT * FROM TblTabungan WHERE NoRek=1;
UPDATE TblTabungan SET Kota='Abepura' WHERE NoRek=1;
COMMIT;
```

NOREK	NAMA	KOTA	SALDO
1	AA	Aceh	100000

1 row updated.

```
SELECT * FROM TblTabungan WHERE NoRek=1;
```

NOREK	NAMA	KOTA	SALDO
1	AA	Abepura	100000

Contoh Trigger 2

- Pemeriksaan tabel History_Kota

```
SELECT * FROM TblHistoryKota;
```

WAKTU	NOREK	KOTA
07-JUN-10 01.56.02.000000 AM	1	Aceh

Contoh Trigger 2

ORACLE 22

- Melihat perubahan data kota

```
SELECT SYSDATE Waktu, Kota
   FROM TblTabungan WHERE NoRek=1
UNION
SELECT Waktu, Kota
   FROM TblHistoryKota WHERE NoRek=1
ORDER BY Waktu DESC
```

WAKTU	KOTA
07-JUN-10 02.06.56.000000 AM	Abepura
07-JUN-10 01.56.02.000000 AM	Aceh

Data paling atas adalah kota paling baru (kota sekarang).

ORACLE ACADEMY Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)

Keterangan :

1. SYSDATE pada SELECT pertama untuk menunjukkan data sekarang
2. UNION digunakan untuk menggabungkan kota sekarang (select pertama) dengan kota dulu (select kedua).

Ada baiknya kalau SELECT ini dibuat sebagai sebuah view.

Contoh Trigger 2

- Membuat View untuk melihat history kota

```
CREATE OR REPLACE VIEW v_HistoryKota
AS
SELECT NoRek,SYSDATE Waktu, Kota FROM TblTabungan
UNION
SELECT NoRek,Waktu, Kota FROM TblHistoryKota
ORDER BY Waktu DESC
```

```
SELECT * FROM v_HistoryKota WHERE NoRek=1
```

NOREK	WAKTU	KOTA
1	07-JUN-10 02.11.37.000000 AM	Abepura
1	07-JUN-10 01.56.02.000000 AM	Aceh

Keterangan :

1. SYSDATE pada SELECT pertama untuk menunjukkan data sekarang
2. UNION digunakan untuk menggabungkan kota sekarang (select pertama) dengan kota dulu (select kedua).

Ada baiknya kalau SELECT ini dibuat sebagai sebuah view.

Contoh Trigger 3

- Buatlah suatu procedure untuk melakukan transaksi tunai. Data yang menjadi parameter hanya NoRekening dan Besar setoran.
- Buatlah suatu procedure untuk melakukan transaksi transfer. Data yang menjadi parameter adalah NoRekening pengirim, NoRekening Penerima dan besar transfer.
- Jika suatu transaksi terjadi, maka harus mentrigger pengupdatean data TblTabungan dengan mengupdate saldo milik masing-masing nasabah.

Contoh Trigger 3 (cash)

- Membuat prosedure Transaksi_Cash

```
CREATE OR REPLACE
PROCEDURE Transaksi_Cash(pNoRek TblTransaksi.NoRek1%TYPE,
                         pBesar TblTransaksi.Besar%TYPE)
IS
  cnt_norek INTEGER;
BEGIN
  IF (pNoRek IS NULL) THEN
1     RAISE_APPLICATION_ERROR(-20000,'No Rekening tidak boleh kosong');
  END IF;
  IF (pBesar IS NULL) OR (pBesar<=0) THEN
2     RAISE_APPLICATION_ERROR(-20001,'Besar transaksi harus lebih dari 0');
  END IF;
  SELECT count(*) INTO cnt_norek FROM TblTabungan WHERE NoRek=pNoRek;
3 IF cnt_norek<>1 THEN
     RAISE_APPLICATION_ERROR(-20002,'No Rekening tidak ditemukan');
  END IF;
  INSERT INTO TblTransaksi
4    VALUES(SeqNoTransaksi.NEXTVAL,SYSDATE,'C',pNoRek,pBesar,NULL);
  COMMIT;
END;
```


Keterangan :

1. Pemeriksaan NoRekening tidak boleh kosong
2. Pemeriksaan Besar setoran tidak boleh kosong dan harus lebih besar dari 0
3. Memeriksa apakan rekening tujuan ada.
4. Lakukan proses penyimpanan transaksi. Proses ini harus dapat mentrigger update saldo di tabel TblTabungan. Setelah semua proses selesai, maka lakukan COMMIT.

Contoh Trigger 3 (cash)

- Membuat Trigger Transaksi_Cash

```
CREATE OR REPLACE TRIGGER trig_Transaksi_Cash
AFTER INSERT ON TblTransaksi
FOR EACH ROW
BEGIN
1  IF UPPER(:NEW.Jenis)='C' THEN
     2  UPDATE TblTabungan SET saldo=saldo+:NEW.Besar
              WHERE NoRek=:NEW.NoRek1;
   END IF;
END;
```


Keterangan :

1. Update hanya dijalankan ketika jenis transaksi (:NEW.Jenis) sama dengan ‘C’ (Cash).
2. Update terhadap tabel tabungan pada nomor rekening sesuai dengan nomor rekening yang diinputkan di tabel transaksi sebesar besar transaksi

Contoh Trigger 3 (cash)

ORACLE 27

- Pengujian transaksi Cash (No Rekening ada)

```
SELECT * FROM TblTabungan WHERE NoRek=1;
EXEC Transaksi_Cash(1,25000);
```

NOREK	NAMA	KOTA	SALDO
1	AA	Abepura	100000

1

PL/SQL procedure successfully completed.

```
SELECT * FROM TblTransaksi
```

NOTRAN	WAKTU	JEN	NOREK1	BESAR	NOREK2
1	07-JUN-10 02.41.46.000000 AM	C	1	25000	

2

```
SELECT * FROM TblTabungan WHERE NoRek=1;
```

NOREK	NAMA	KOTA	SALDO
1	AA	Abepura	125000

3

ORACLE ACADEMY Oracle-academy@if-unikom oleh : Andri Heryandi, M.T. (2010)

Keterangan :

1. Data sebelum dilakukan transaksi
2. Data transaksi
3. Data setelah dilakukan transfer

Contoh Trigger 3 (cash)

- Pengujian transaksi Cash (No Rekening Tidak Ada)

```
EXEC Transaksi_Cash(50,25000);
```

```
BEGIN Transaksi_Cash(50,25000); END;

*

ERROR at line 1:
ORA-20002: No Rekening tidak ditemukan
ORA-06512: at "DB97025.TRANSAKSI_CASH", line 14
ORA-06512: at line 1
```

Contoh Trigger 3 (transfer)

- Membuat prosedure Transaksi_Transfer

```
CREATE OR REPLACE
PROCEDURE Transaksi_Transfer(pNoRek1 TblTransaksi.NoRek1%TYPE,
                             pNoRek2 TblTransaksi.NoRek2%TYPE,
                             pBesar TblTransaksi.Besar%TYPE)
IS
  cnt_norek INTEGER;                                          (1)
  saldorek1 TblTabungan.Saldo%TYPE;
BEGIN
   IF (pNoRek1 IS NULL) THEN                                  (2)
       RAISE_APPLICATION_ERROR(-20000,'No Rekening pengirim tidak boleh
kosong');
   END IF;
   IF (pNoRek2 IS NULL) THEN                                  (3)
       RAISE_APPLICATION_ERROR(-20001,'No Rekening penerima tidak boleh
kosong');
   END IF;
   IF (pBesar IS NULL) OR (pBesar<=0) THEN                    (4)
       RAISE_APPLICATION_ERROR(-20002,'Besar transaksi harus lebih dari 0');
   END IF;
```


Keterangan :

1. Pembuatan variable local
2. Pemeriksaan NoRekening pengirim tidak boleh kosong
3. Pemeriksaan NoRekening penerima tidak boleh kosong
4. Pemeriksaan Besar transfer tidak boleh kosong dan harus lebih besar dari 0

Contoh Trigger 3 (transfer)

- Membuat prosedure Transaksi_Transfer

```
1  SELECT count(*) INTO cnt_norek FROM TblTabungan WHERE NoRek=pNoRek1;
   IF cnt_norek<>1 THEN
       RAISE_APPLICATION_ERROR(-20003,'No Rekening tidak ditemukan');
   ELSE
       SELECT Saldo INTO saldorek1 FROM TblTabungan WHERE NoRek=pNoRek1;
2      IF saldorek1<pBesar THEN
          RAISE_APPLICATION_ERROR(-20004,'Saldo Pengirim tidak cukup.
Tersedia Rp. '||saldorek1);
       END IF;
   END IF;
3  SELECT count(*) INTO cnt_norek FROM TblTabungan WHERE NoRek=pNoRek2;
   IF cnt_norek<>1 THEN
       RAISE_APPLICATION_ERROR(-20005,'No Rekening penerima tidak
ditemukan');
   END IF;
4  INSERT INTO TblTransaksi
      VALUES(SeqNoTransaksi.NEXTVAL,SYSDATE,'T',pNoRek1,pBesar,pNoRek2);
   COMMIT;
END;
```


Keterangan :

1. Pemeriksaan apakah No Rekening pengirim ada? Jika tidak ada (hasil count(*) <> 1) maka tampilkan pesan salah.
2. Jika norekening pengirim ada, maka periksa saldo milik rekening pengirim. Jika tidak mencukupi (Saldorek1<pBesar) maka tampilkan pesan kesalahan
3. Pemeriksaan apakah No Rekening penerima ada?. Jika tidak ada tampilkan pesan kesalahannya.
4. Penyimpanan data transaksi. Penyimpanan ini akan melakukan trigger pengupdatean saldo penerima dan pengirim. Jika semua proses selesai, maka proses diakhiri dengan COMMIT.

Contoh Trigger 3 (transfer)

- Membuat Trigger Transaksi_Transfer

```
CREATE OR REPLACE TRIGGER trig_Transaksi_Transfer
AFTER INSERT ON TblTransaksi
FOR EACH ROW
BEGIN
1  IF UPPER(:NEW.Jenis)='T' THEN
       UPDATE TblTabungan SET saldo=saldo-:NEW.Besar
2            WHERE NoRek=:NEW.NoRek1;
       UPDATE TblTabungan SET saldo=saldo+:NEW.Besar
             WHERE NoRek=:NEW.NoRek2;
   END IF;
END;
```


Keterangan :

1. Update hanya dijalankan ketika jenis transaksi (:NEW.Jenis) sama dengan ‘T’ (Transfer).
2. Update saldo di rekening pengirim dan penerima sesuai besar transfer.

Contoh Trigger 3 (transfer)

- Pengujian transaksi Transfer (No Rekening ada)

```
SELECT * FROM TblTabungan WHERE NoRek=1 or NoRek=2;
EXEC Transaksi_Transfer(1,2,25000);
```

NOREK	NAMA	KOTA	SALDO
1	AA	Abepura	125000
2	Bebe	Bandung	200000

1

PL/SQL procedure successfully completed.

```
SELECT * FROM TblTransaksi
```

NOTRAN	WAKTU	JEN	NOREK1	BESAR	NOREK2
1	07-JUN-10 02.41.46.000000 AM	C	1	25000	
2	07-JUN-10 03.13.56.000000 AM	T	1	25000	2

2

Keterangan :
1. Data sebelum dilakukan transaksi
2. Data transaksi

Contoh Trigger 3 (transfer)

- Periksa status saldo setelah transaksi transfer

```
SELECT * FROM TblTabungan WHERE NoRek=1 or NoRek=2;
```

NOREK	NAMA	KOTA	SALDO
1	AA	Abepura	100000
2	Bebe	Bandung	225000

Keterangan :

Contoh Trigger 3 (transfer)

- Pengujian transaksi Transfer (No Rekening pengirim tidak ada)

```
EXEC Transaksi_Transfer(100,2,25000);
```

```
BEGIN Transaksi_Transfer(100,2,25000); END;

*

ERROR at line 1:
ORA-20003: No Rekening tidak ditemukan
ORA-06512: at "DB97025.TRANSAKSI_TRANSFER", line 19
ORA-06512: at line 1
```


Contoh Trigger 3 (transfer)

- Pengujian transaksi Transfer (Saldo Tidak Cukup)

```
EXEC Transaksi_Transfer(1,2,99999999);
```

```
BEGIN Transaksi_Transfer(1,2,99999999); END;

*

ERROR at line 1:
ORA-20004: Saldo Pengirim tidak cukup. Tersedia Rp. 100000
ORA-06512: at "DB97025.TRANSAKSI_TRANSFER", line 23
ORA-06512: at line 1
```


Contoh Trigger 3 (transfer)

- Pengujian transaksi Transfer (No Rekening penerima tidak ada)

```
EXEC Transaksi_Transfer(1,200,25000);
```

```
BEGIN Transaksi_Transfer(1,200,25000); END;

*

ERROR at line 1:
ORA-20005: No Rekening penerima tidak ditemukan
ORA-06512: at "DB97025.TRANSAKSI_TRANSFER", line 28
ORA-06512: at line 1
```


Contoh Trigger 4

- Buatlah suatu trigger yang mensimulasikan referential integrity pada Foreign Key untuk ON UPDATE CASCADE
- Jika terjadi update pada NoRek di TblTabungan, maka semua data history_kota akan ikut berubah pula.
- Ingat: Oracle tidak menganjurkan adanya ON UPDATE CASCADE, tetapi jika benar-benar diperlukan, dapat dibuat trigger yang bekerja seperti ON UPDATE CASCADE.

Contoh Trigger 4

- Membuat Trigger Update NoRek

```
CREATE OR REPLACE TRIGGER trig_update_norek
AFTER UPDATE ON TblTabungan
FOR EACH ROW
BEGIN
   IF :NEW.NoRek<>:OLD.NoRek THEN
       UPDATE TblHistoryKota SET NoRek=:New.NoRek
           WHERE NoRek=:OLD.NoRek;
       -- JIKA MASIH ADA RELASI KE TABEL LAIN
       -- UPDATE KE TABEL LAIN BISA DISIMPAN DI SINI
   END IF;
END;
```


Keterangan :

1. Update hanya dijalankan ketika NoRek baru tidak sama dengan NoRek lama.
2. Update data di tabel HistoryKota untuk menyesuaikan dengan NoRek baru.

Contoh Trigger 4

- Pengujian trigger Update Norek

```
SELECT * FROM TblTabungan WHERE NoRek=1
```

NOREK	NAMA	KOTA	SALDO
1	AA	Abepura	100000

```
SELECT * FROM TblHistoryKota WHERE NoRek=1
```

WAKTU	NOREK	KOTA
07-JUN-10 01.56.02.000000 AM	1	Aceh

Contoh Trigger 4

- Update Nomor Rekening

```
UPDATE TblTabungan SET NoRek=999 WHERE NoRek=1
```

1 row updated.

Contoh Trigger 4

- Pengujian trigger Update Norek

```
SELECT * FROM TblTabungan
```

NOREK	NAMA	KOTA	SALDO
999	AA	Abepura	100000
2	Bebe	Bandung	225000
3	Cece	Cimahi	300000
4	Dede	Dumai	400000
5	Erni	Ende	-10000
6	Ferdi	Fakfak	0

```
SELECT * FROM TblHistoryKota
```

WAKTU	NOREK	KOTA
07-JUN-10 01.56.02.000000 AM	999	Aceh

Perhatikan bahwa NoRek di TblHistoryKota terupdate ketika NoRek di TblTabungan diupdate.