Nozione ed uso

- Operazioni eseguite automaticamente ogni volta che avviene un certo evento
- Uso:
  - Gestione di vincoli di integrità:
    - Per fallimento
    - Per modifica
  - Auditing:
    - Sicurezza
    - Statistiche
  - Valori derivati memorizzati
  - Facilitare lo sviluppo di applicazioni tramite RAD
  - Altro

Operazioni scatenanti

- `select ... from ... where`
- `insert into tabella [ ( [ colonna /,]* ) ] values ( [expr /, ]* )`
- `insert into tabella [ ( [ colonna /,]* ) ] subquery`
- `update table [nome] set [ column = expr / , ]* [where condizione]`
- `delete [ from ] tabella where condizione`

Sintassi

- Sintassi:

```
create [or replace] trigger nome
{before | after}
[  {delete | insert | update [of [column /,]* }
  /or]*
on tabella
[referencing { old as nome | new as nome }*]
[for each row]
[when (condition)]
blocco
```

- **`Referencing`** e **`when`** solo se **`for each row`**; per default, la vecchia riga è :old e la nuova è :new nel blocco, old e new nella condizione

Esempio

- Siano *prenotazioni* e *agenzie* due tabelle legate dall'attributo *agenzia* chiave esterna in *prenotazioni*.

- Creo una directory *ese5bdl* e vi copio i file in *~ghelli/bdl07/esercizi/ese5/*
- Mi collego al server Oracle che ho scelto e compilo nell'ordine:
  - create.sql
  - ese5.sql
  - trigger.sql

Esempio

- Modifico *menuAgenzie.html* sostituendo *YYY* con il mio nome utente di Oracle
- Copio *menuAgenzie.html* nella directory *~/public_html*
- Inserisco dati accedendo la pagina,

  *http://www.cli.di.unipi.it/*mioAccountWeb*/menuAgenzie.html*

trigger.sql (1/3)

- Il trigger alla prima prenotazione crea l'agenzia, per le successive ne aggiorna il totale di spese e di prenotazioni

trigger.sql (2/3)

```
create or replace trigger t1
before insert on prenotazioni
for each row
declare
  conta number;
begin
  select count(*) into conta
  from   agenzie
  where  nomeAgenzia = :new.agenzia;
```

trigger.sql (3/3)

```
if (conta = 0)
  then insert into agenzie
         values (:new.agenzia,1,:new.spesa);
  else update agenzie
       set    numPrenotazioni = numPrenotazioni + 1,
              spesaTot = spesaTot + :new.spesa
       where  nomeAgenzia = :new.agenzia;
```

trigger2.sql (1/3)

- Il trigger estende *trigger.sql* per controllare che ogni agenzia non abbia più di tre prenotazioni (limite massimo consentito), nel caso solleva un'eccezione

trigger2.sql (2/3)

```
create or replace package p_ese5 as
   …
   troppePrenotazioni exception;
end p_ese5;

create or replace trigger t1
before insert on prenotazioni
for each row
declare
 conta number;
 prenota number;
begin
 select count(*) into conta
 from   agenzie
 where  nomeAgenzia = :new.agenzia;
```

trigger2.sql (3/3)

```
else begin
       select numPrenotazioni into prenota
       from   agenzie
       where  nomeAgenzia = :new.agenzia;
       if (prenota = 3)
         then raise p_ese4.troppePrenotazioni;
       end if;

      update agenzie
       set    numPrenotazioni = numPrenotazioni + 1,
              spesaTot = spesaTot + :new.spesa
       where  nomeAgenzia = :new.agenzia;
     end;
```

Eccezione e trigger

```
create or replace package p_ese5 as
   procedure prenota(…);
   troppePrenotazioni exception;
end p_ese5;

create or replace package body p_ese5 as
   procedure prenota(…)
   begin
      …
      insert into prenotazioni
      values (…)
      …
      exception when p_ese5.troppePrenotazioni
                then …do something…
   end prenota;
end p_ese5;
```

Esercizi

- Realizzare una procedura per la cancellazione delle prenotazioni
- Realizzare un trigger che si occupi di eliminare le agenzie dal database quando non esistono più prenotazioni associate
- Realizzare una pagina di immissione dati più ricca, che consenta di selezionare il nome dell'agenzia tra quelle esistenti nel database

Limitazione

- Limitazione: non è possibile modificare la tabella che ha fatto partire il trigger o la chiave di una tabella coinvolta nel comando, se il trigger è **`for each row`** (**`insert into`** fa eccezione)

TRIGGER ILLECITO

```
create or replace trigger trigDelStat
after delete on prenota
for each row
declare
  conta number;
begin
  select count(*) into conta
  from   prenota
  where  login = :old.login;
  if conta = 0
  then delete from utenti where login = :old.login;
  else update utenti
       set    oreTotali = oreTotali - 1
       where  login = :old.login;
  end if;
end;
```

VERSIONE ACCETTATA

```
create or replace trigger trigDelRow
after delete on prenota
for each row
begin
  update utenti
  set    oreTotali = oreTotali - 1
  where  login = :old.login;
end;
```

VERSIONE ALTERNATIVA

```
create or replace trigger tridel
after delete on prenota
begin
  delete utenti u
  where  not exists
    (select *
     from   prenota
     where  login = u.login);
end;
```