

Oracle

Creazione di un trigger in Oracle:

```
CREATE TRIGGER Nome  
{BEFORE | AFTER} Eventi  
ON Relazione  
[REFERENCING Riferimenti]  
[FOR EACH ROW]  
[WHEN Condizione]  
Blocco PL/SQL;
```

- eseguiti prima o dopo l'occorrere dell'evento che li attiva
BEFORE o AFTER
- eventi: insert, delete, update [of *Lista Attributi*]
- FOR EACH ROW: trigger eseguito per ogni tupla coinvolta nell'esecuzione

Combinando le opzioni BEFORE e AFTER con l'opzione FOR EACH ROW risultano quattro possibilità:

| | | opzione FOR EACH ROW |
|----------------|--|---|
| opzione BEFORE | <i>trigger before statement:</i> il trigger è attivato un'unica volta prima dell'esecuzione del comando che lo attiva | <i>trigger before row:</i> il trigger è attivato prima di modificare ogni tupla coinvolta dall'esecuzione del comando che attiva il trigger |
| opzione AFTER | <i>trigger after statement:</i> il trigger è attivato un'unica volta dopo l'esecuzione del comando che lo attiva | <i>trigger after row:</i> il trigger è attivato dopo aver modificato ogni tupla coinvolta dall'esecuzione del comando che attiva il trigger |

- **WHEN:** condizione del trigger (può essere specificata solo quando il trigger è eseguito per ogni tupla), solo un predicato semplice che coinvolge solo gli attributi della tupla modificata
- **REFERENCING:** dichiarazione di nomi per indicare i valori correnti e/o precedenti delle tuple che sono state modificate
 - *OLD AS Nome Tupla Old*
 - *NEW AS Nome Tupla New*

Default: `old` e `new`

- **PL/SQL:** azione del trigger. PL/SQL è un linguaggio di programmazione a tutti gli effetti, sia interrogazioni che aggiornamenti
- **altri comandi:** `ALTER TRIGGER` con opzioni `ENABLE` e `DISABLE`, `DROP TRIGGER`

Oracle:ESEMPIO

```
CREATE TRIGGER controlla_stipendio
BEFORE insert, update of Stipendio, Mansione
ON Impiegati
FOR EACH ROW
WHEN (new.Mansione <> 'PRESIDENT')
DECLARE /* inizio del blocco PL/SQL */
minstip number;
maxstip number;
BEGIN
SELECT minstip, maxstip
FROM Stipendi
WHERE Mansione =:new.Mansione;
IF (:new.Stipendio < minstip OR :new.Stipendio
> maxstip)
THEN raise_application_error (-20601, 'stipendio'
|| :new.Stipendio || 'fuori dal range per la mansione'
|| :new.Mansione || 'per l'impiegato' || :new.Nome);
END IF;
END; /* fine del blocco PL/SQL */
```

DB2

Creazione di un trigger in DB2:

```
CREATE TRIGGER Nome
{ NO CASCADE BEFORE | AFTER }
{ insert | delete | update [of Lista Attributi]}
ON Relazione
[REFERENCING { OLD AS Variabile |
              NEW AS Variabile |
              OLD_TABLE AS Variabile |
              NEW_TABLE AS Variabile }
[{ FOR EACH STATEMENT | FOR EACH ROW }
MODE DB2SQL]
[WHEN Condizione]
BEGIN ATOMIC Sequenza Comandi SQL
END!
```

- eventi: solo operazioni di aggiornamento della base di dati insert, delete, update [of *Lista Attributi*]

- prima o dopo l'esecuzione dell'operazione che li attiva (*before trigger* contro *after trigger*)
- riferiti alla singola tupla (*row trigger*) o all'intero comando (*statement trigger*). Differenza tra Oracle e DB2: in DB2 un *before trigger* deve essere necessariamente riferito alla singola tupla
- **WHEN**: condizione del trigger. Interrogazione complessa contenente anche sottointerrogazioni e join

Variabili di transizione (clausola REFERENCING): per riferirsi allo stato corrente e allo stato precedente. Quattro tipi:

- *old row*: valore della tupla modificata prima dell'esecuzione dell'operazione che ha attivato il trigger
- *new row*: valore della tupla modificata dopo l'esecuzione dell'operazione che ha attivato il trigger
- *old table*: transition table contenente tutte le tuple come prima dell'esecuzione dell'operazione che ha attivato il trigger

- *new table*: transition table contenente tutte le tuple come modificate dopo l'esecuzione dell'operazione che ha attivato il trigger

Utilizzate sia nella condizione che nell'azione.
Restrizione: la clausola `REFERENCING` della definizione di un trigger deve contenere al più una variabile di ogni tipo

DB2:ESEMPIO

Variabili di transizione:

```
REFERENCING NEW AS newrow
  /* dichiara una variabile new row */
REFERENCING OLD AS oldrow
  /* dichiara una variabile old row */
REFERENCING OLD_TABLE AS oldtable
  /* dichiara una variabile old table */

WHEN (newrow.Stipendio<oldrow.Stipendio)
WHEN (newrow.Stipendio>
      (SELECT max(Stipendio)
       FROM Impiegati
       WHERE Mansione=newrow.Mansione))
WHEN ((SELECT count(*) FROM oldtable)>100)
```

- azione: lista dei comandi SQL. Eseguiti come operazioni atomiche: se uno dei comandi fallisce \Rightarrow rollback dell'operazione che ha attivato il trigger e di tutti i comandi nell'azione. Non viene effettuato il rollback della transazione contenente l'operazione che ha attivato il trigger
- NO CASCADE: usata nella specifica di trigger di tipo *before*, evidenziare che un *before* trigger non attiva mai un altro *before* trigger
- MODE DB2SQL: usata nella clausola che specifica se un trigger debba essere eseguito per ogni tupla o per ogni comando. Garantisce che le applicazioni esistenti non saranno influenzate da possibili estensioni future
- comandi per cancellare e modificare trigger

DB2:ESEMPIO

```
CREATE TRIGGER assumi_imp
AFTER insert ON Dipendenti
REFERENCING NEW AS newrow
FOR EACH ROW MODE DB2SQL
UPDATE Dipendenti
    SET Ndip = Ndip + 1
    WHERE Imp# = newrow.Dirigente!
```

```
CREATE TRIGGER cancella_imp
AFTER delete ON Dipendenti
REFERENCING OLD AS oldrow
FOR EACH ROW MODE DB2SQL
UPDATE Dipendenti
    SET Ndip = Ndip -1
    WHERE Imp# = oldrow.Dirigente!
```

DB2:ESEMPIO (CONT.)

```
CREATE TRIGGER trasferisci_imp
AFTER update of Dirigente ON Dipendenti
REFERENCING OLD AS oldrow NEW AS newrow
FOR EACH ROW MODE DB2SQL
```

```
begin atomic
```

```
    UPDATE Dipendenti
```

```
        SET Ndip = Ndip - 1
```

```
        WHERE Imp# = oldrow.Dirigente;
```

```
    UPDATE Dipendenti
```

```
        SET Ndip = Ndip + 1
```

```
        WHERE Imp# = newrow.Dirigente;
```

```
end!
```

```
CREATE TRIGGER propaga_imp
```

```
AFTER update of Ndip ON Dipendenti
```

```
REFERENCING OLD AS oldrow NEW AS newrow
```

```
FOR EACH ROW MODE DB2SQL
```

```
UPDATE Dipendenti
```

```
    SET Ndip = Ndip + newrow.Ndip - oldrow.Ndip
```

```
    WHERE Imp# = newrow.Dirigente!
```