Università degli Studi di Roma 3
AA 2003/04
Tecnologia delle Basi di Dati

DB2 Trigger

realizzato da:
Gianforme Giorgio
Mancini Federico
Refice Tiziana

Outline

- Trigger
- Trigger in DB2
- HOW-TO create a Trigger
- Examples
- Views and INSTEAD OF Trigger

What's Trigger?

- A trigger defines a set of action and can be activated when a specified SQL operation -- a **DELETE**, **INSERT** or **UPDATE** -- occurs on a table.
- Triggers, unlike **referential** and **check constraints**, can even be used to update other tables.

Why Triggers ?!

- **Validate input data**

  Note that validation of non-transitional data is usually better handled by check and referential constraints. By contrast, triggers are appropriate for validation of transitional data, that is, validations which require comparisons between the value before and after an update operation.
- **Automatically generate values for newly inserted rows**
- **Read from other tables** for cross-referencing purposes
- **Write to other tables** for audit-trail purposes
- **Support alerts** (for example, through e-mail messages)

Benefits of Triggers

- **Faster application development**

  Because triggers are stored in the relational database, the actions performed by triggers do not have to be coded in each application.
- **Global enforcement of business rules**

  A trigger only has to be defined once, and then it can be used for any application that changes the table.
- **Easier maintenance**

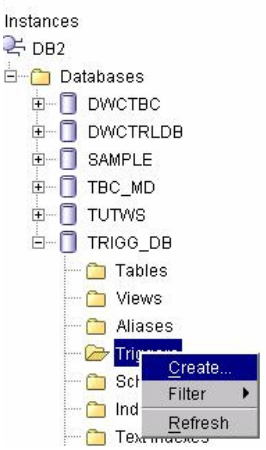
  If a business policy changes, only the corresponding trigger needs to change instead of each application program.

DB2 Triggers

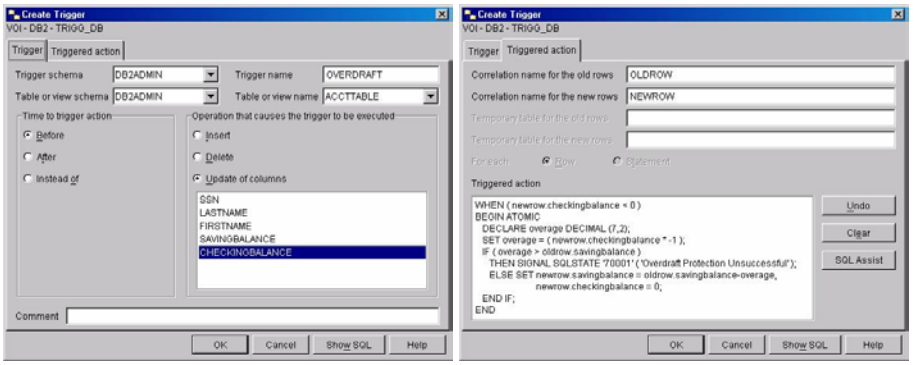
Steps to create a Trigger (1)

SSN	LASTNAME	FIRSTNAME	SAVINGBALANCE	CHECKINGBALANCE
111-11-1111	Rossi	Mario	1,500.00	1,000.00
222-22-2222	Bianchi	Giuseppe	2,000.00	4,000.00

DB2 Triggers
Steps to create a Trigger (2)

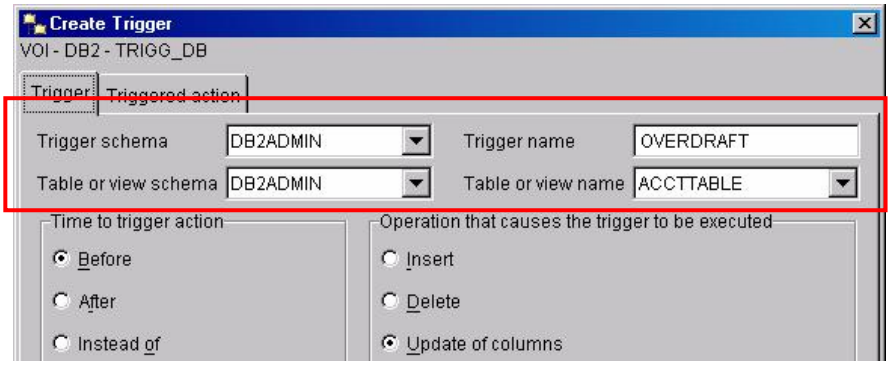


DB2 Triggers
Steps to create a Trigger (3)



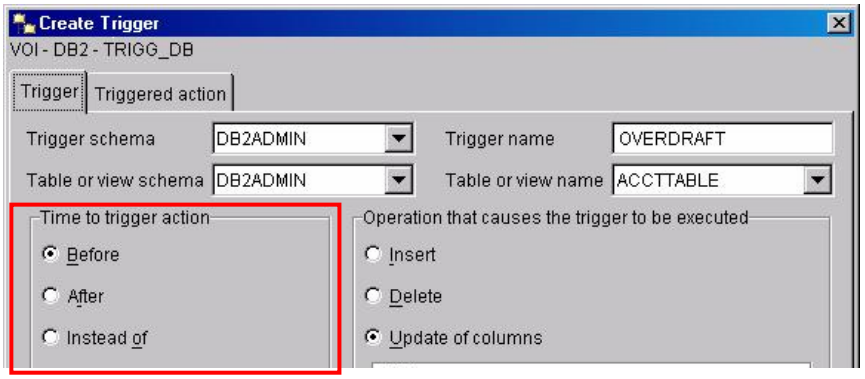
DB2 Triggers
Steps to create a Trigger (4)

- The **name of the trigger**
- The **name of the subject table**



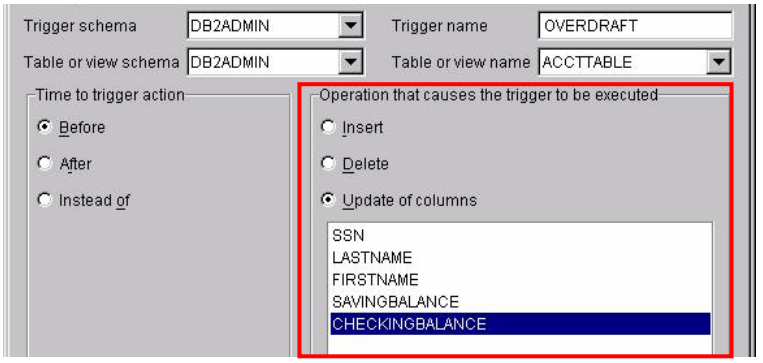
DB2 Triggers
Steps to create a Trigger (5)

- The **trigger activation time** (**NO CASCADE BEFORE** or **AFTER** the update operation executes)



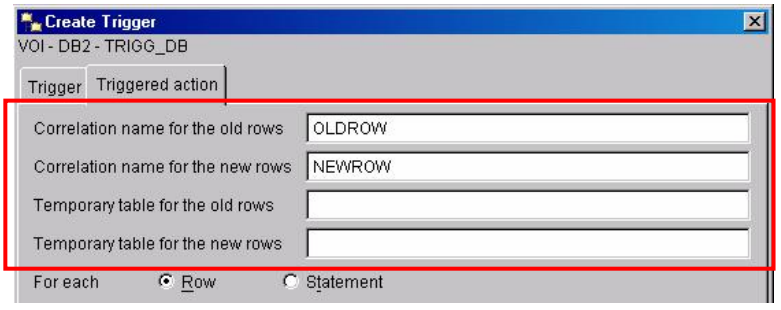
DB2 Triggers
Steps to create a Trigger (6)

- The **trigger event** (**INSERT**, **DELETE** or **UPDATE**)
  If the trigger event is UPDATE, then the trigger column list for the trigger event of the trigger



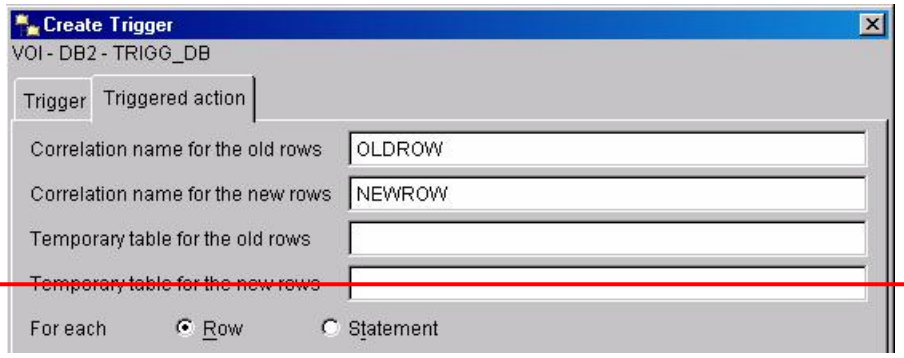
DB2 Triggers
Steps to create a Trigger (7)

- The **old values transition variable**, if any
- The **new values transition variable**, if any
- The **old values transition table**, if any
- The **new values transition table**, if any



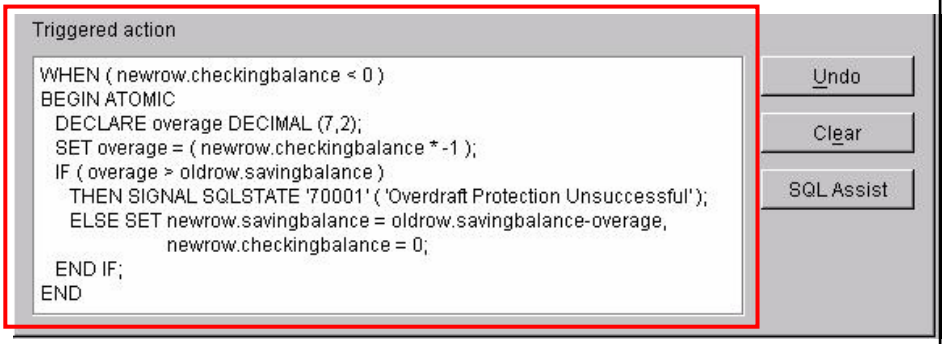
DB2 Triggers
Steps to create a Trigger (8)

- The **granularity** (**FOR EACH STATEMENT** or **FOR EACH ROW**)



DB2 Triggers
Steps to create a Trigger (9)

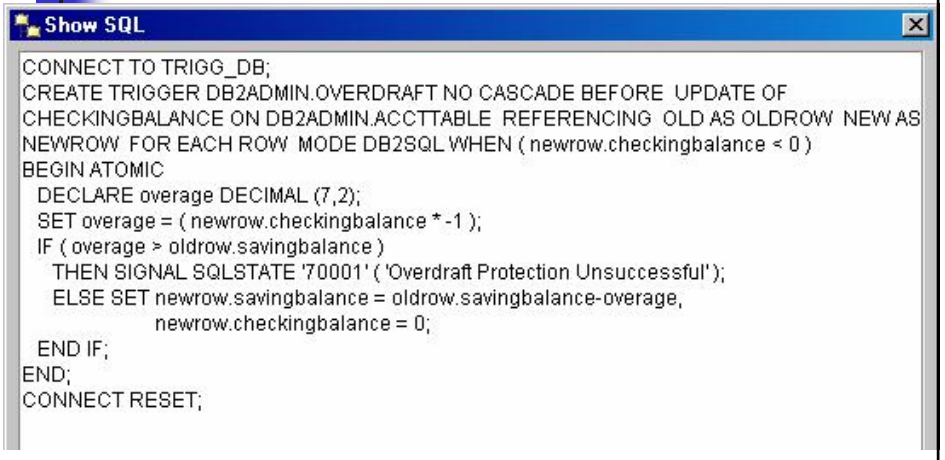
- The **triggered action** of the trigger (including a **triggered action condition** and **triggered SQL statement(s)**)



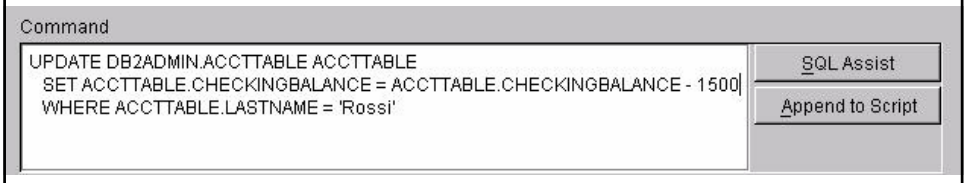
DB2 Triggers
Steps to create a Trigger (10)

- The **trigger creation timestamp**
- The **current function path** (for example, user-defined functions) (by specifying the SQL statement SET CURRENT FUNCTION PATH)

DB2 Triggers
Steps to create a Trigger (11)



DB2 Triggers
Steps to create a Trigger (12)



SSN	LASTNAME	FIRSTNAME	SAVINGBALANCE	CHECKINGBALANCE
111-11-1111	Rossi	Mario	1,000.00	0.00
222-22-2222	Bianchi	Giuseppe	2,000.00	4,000.00

DB2 Trigger
Prototype of a Trigger

CREATE TRIGGER [***nome trigger***]
[***activation time***][***trigger event***]
ON [***subject table***]
REFERENCING [***object***] **AS** [***name***]
[***granularity***] **MODE DB2SQL**
WHEN [***condition***]
BEGIN ATOMIC
[***triggered action***]
END

DB2 Trigger
Notes for Trigger Action

- I comandi SQL sono eseguiti come operazioni atomiche: se uno dei comandi fallisce allora viene effettuato il 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.)
- Per effettuare il rollback della transazione contenente l'operazione che ha attivato il trigger, si può utilizzare l'istruzione **SIGNAL**.

DB2 Trigger
Notes for granularity (1)

- If the set of affected rows is empty (that is, in the case of a searched UPDATE or DELETE in which the WHERE clause did not qualify any rows), a FOR EACH ROW trigger does not run. But a FOR EACH STATEMENT trigger still runs once.
- A granularity of FOR EACH STATEMENT is not supported for BEFORE triggers (BEFORE triggers must have a granularity of FOR EACH ROW).

DB2 Trigger
Notes for granularity (2)

TIPO DI TRIGGER	ROW TRIGGER	STATEMENT TRIGGER
BEFORE INSERT	NEW	*invalido*
BEFORE UPDATE	OLD,NEW	*invalido*
BEFORE DELETE	OLD	*invalido*
AFTER INSERT	NEW, NEW TABLE	NEW TABLE
AFTER UPDATE	OLD, OLD TABLE NEW, NEW TABLE	OLD_TABLE, NEW_TABLE
AFTER DELETE	OLD, OLD TABLE	OLD TABLE

DB2 Triggers
Notes for activation time (1)

- An AFTER trigger occur after the trigger event executes, and after the database manager checks all constraints that the trigger event may affect, including actions of referential constraints.
- A BEFORE trigger is activated before integrity constraints are checked and may be violated by the trigger event.
- DML operations are not allowed in BEFORE triggers. (Un BEFORE trigger può modificare solo le nuove tuple.)

DB2 Triggers
Notes for activation time (2)

- Di conseguenza, per modificare campi protetti da FOREIGN KEY

  - non si può usare un AFTER trigger perchè la presenza di una foreign key fa comunque fallire la modifica;

  - non si può usare un BEFORE trigger, in quanto in esso non sono consentite modifiche di altre tabelle

  Una possibile soluzione è quella di eliminare la foreign key e realizzare tutto tramite trigger.

- The order in which rows are processed by the trigger is arbitrary.

DB2 Triggers
Trigger Cascading (1)

- When you run a triggered SQL statement, it may cause the event of another trigger to occur, which in turn, causes the other trigger to be activated. Therefore, activating a trigger can cascade the activation of one or more other triggers.

- L'esecuzione di un trigger può provocare, direttamente o indirettamente, anche l'attivazione dello stesso trigger (**recursive trigger**)

DB2 Triggers
Trigger Cascading (2)

- Quando l'esecuzione di un' azione di un trigger T1 provoca l'attivazione di un altro trigger T2, si interrompe l'esecuzione di T1 per processare T2
- The run-time depth level of trigger cascading supported is 16. If a trigger at level 17 is activated, SQLCODE -724 (SQLSTATE 54038) will be returned and the triggering statement will be rolled back.
- **NO CASCADE** nella specifica BEFORE evidenzia che un BEFORE trigger non attiva mai un altro BEFORE trigger.

DB2 Triggers
Ordering of Multiple Triggers

- When triggers are defined using the CREATE TRIGGER statement, their creation time is registered in the database in form of a timestamp.
- The activation of the triggers, when there is more than one trigger that should be run at the same time, is conducted in ascending order of the timestamp value to ensure that new triggers can be used as incremental additions to the changes that affect the database.

DB2 Triggers
Examples

(Atzeni, "Basi di dati Architetture e linee di evoluzione", 2003, p263)

E' dato lo schema relazionale:
IMPIEGATO (Nome, Stipendio, DipNum)
DIPARTIMENTO (DipNum, NomeManager)

DIPNUM	NOMEMANAGER
1	Mario
2	Francesco
3	Carla

NOME	STIPENDIO	DIPNUM
Mario	80,000.00	1
Carla	90,000.00	3
Silvia	30,000.00	2
Marco	40,000.00	1
Francesco	70,000.00	2
Andrea	50,000.00	3

DB2 Triggers
Example 1 (1)

(Atzeni, "Basi di dati Architetture e linee di evoluzione", 2003, p263)

Definire un trigger che, quando un dipartimento è cancellato, mette a NULL il valore di DipNum degli impiegati appartenenti a quel dipartimento

```
CREATE TRIGGER DB2Admin.DelDip
AFTER DELETE ON DB2Admin.Dipartimento
REFERENCING  OLD AS OldRow  OLD_TABLE AS OldTable
FOR EACH ROW  MODE DB2SQL
WHEN ( EXISTS (SELECT *
                    FROM Impiegato
                    WHERE DipNum = OldRow.DipNum)  )
BEGIN ATOMIC
  UPDATE Impiegato
    SET DipNum = null
    WHERE DipNum = OldRow.DipNum;
END
```

DB2 Triggers
Example 1 (2)

(Atzeni, "Basi di dati Architetture e linee di evoluzione", 2003, p263)

```
DELETE FROM DB2Admin.Dipartimento Dipartimento
 WHERE Dipartimento.DipNum = 2
```

DIPNUM	NOMEMANAGER
1	Mario
2	Francesco
3	Carla

DIPNUM	NOMEMANAGER
1	Mario
3	Carla

NOME	STIPENDIO	DIPNUM
Mario	80,000.00	1
Carla	90,000.00	3
Silvia	30,000.00	2
Marco	40,000.00	1
Francesco	70,000.00	2
Andrea	50,000.00	3

NOME	STIPENDIO	DIPNUM
Mario	80,000.00	1
Carla	90,000.00	3
Silvia	30,000.00	
Marco	40,000.00	1
Francesco	70,000.00	
Andrea	50,000.00	3

DB2 Triggers
Example 2 (1)

(Atzeni, "Basi di dati Architetture e linee di evoluzione", 2003, p263)

Definire un trigger che cancella tutti gli impiegati appartenenti ad un dipartimento quando quest'ultimo è cancellato

```
CREATE TRIGGER DB2Admin.DelDip2
AFTER  DELETE  ON DB2Admin.Dipartimento
REFERENCING OLD AS OldRow OLD_TABLE AS OldTable
FOR EACH ROW  MODE DB2SQL
WHEN ( EXISTS (SELECT *
                    FROM DB2Admin.Impiegato
                    WHERE DipNum=OldRow.DipNum) )
BEGIN ATOMIC
  DELETE FROM DB2Admin.Impiegato
  WHERE DipNum = OldRow.DipNum;
END
```

DB2 Triggers
Example 2 (2)

(Atzeni, "Basi di dati Architetture e linee di evoluzione", 2003, p263)

```
DELETE FROM DB2Admin.Dipartimento Dipartimento
 WHERE Dipartimento.DipNum = 2
```

DIPNUM	NOMEMANAGER
1	Mario
2	Francesco
3	Carla

DIPNUM	NOMEMANAGER
1	Mario
3	Carla

NOME	STIPENDIO	DIPNUM
Mario	80,000.00	1
Carla	90,000.00	3
Silvia	30,000.00	2
Marco	40,000.00	1
Francesco	70,000.00	2
Andrea	50,000.00	3

NOME	STIPENDIO	DIPNUM
Mario	80,000.00	1
Carla	90,000.00	3
Marco	40,000.00	1
Andrea	50,000.00	3

DB2 Triggers
Example 3 (1)

(Atzeni, "Basi di dati Architetture e linee di evoluzione", 2003, p263)

Definire un trigger che, ogni volta vengono modificati gli stipendi, verifica che non vi siano dipartimenti in cui lo stipendio medio cresce più del 3%, e in tal caso annulla la modifica

DB2 Triggers
Example 3 (2)

(Atzeni, "Basi di dati Architetture e linee di evoluzione", 2003, p263)

```
CREATE TRIGGER DB2Admin.UpdateWage
BEFORE  UPDATE OF Stipendio ON DB2Admin.Impiegato
REFERENCING OLD AS OldRow NEW AS NewRow
OLD_TABLE AS OldTable NEW_TABLE AS NewTable
FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC
  DECLARE OldAvg DECIMAL (7,2); DECLARE NewAvg DECIMAL (7,2);
  DECLARE Avg DECIMAL (7,2); DECLARE Count DECIMAL (7,2);
  SET Count = SELECT COUNT(*) AS countrow FROM Impiegato
                    WHERE Impiegato.DipNum = NewRow.DipNum;
  SET Avg=(SELECT AVG(Stipendio) FROM Impiegato
              WHERE Impiegato.DipNum = NewRow.DipNum
                                  AND Impiegato.Nome <> NewRow.Nome);
  SET OldAvg = Avg + (OldRow.Stipendio - Avg) / Count;
  SET NewAvg = Avg + (NewRow.Stipendio - Avg) / Count;
  IF NewAvg>(OldAvg*1.03)
    THEN SIGNAL SQLSTATE '70001'
                        ('Lo stipendio medio del dipartimento è aumentato più del 3%');
  END IF;
END
```


DB2 Triggers
Example 3 (3)

(Atzeni, "Basi di dati Architetture e linee di evoluzione", 2003, p263)

```
UPDATE DB2Admin.Impiegato Impiegato
  SET Stipendio = Stipendio + 1000
  WHERE Impiegato.DipNum = 3 AND Impiegato.Nome = 'Andrea'
```

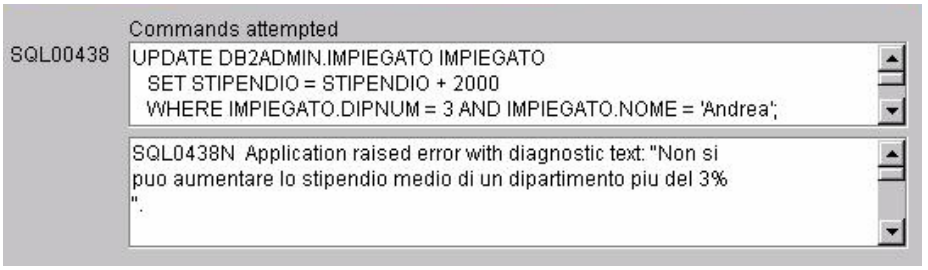
NOME	STIPENDIO	DIPNUM
Mario	80,000.00	1
Carla	90,000.00	3
Silvia	30,000.00	2
Marco	40,000.00	1
Francesco	70,000.00	2
Andrea	50,000.00	3

NOME	STIPENDIO	DIPNUM
Mario	80,000.00	1
Carla	90,000.00	3
Francesco	70,000.00	2
Marco	40,000.00	1
Silvia	30,000.00	2
Andrea	51,000.00	3

DB2 Triggers
Example 3 (3)

(Atzeni, "Basi di dati Architetture e linee di evoluzione", 2003, p263)

```
UPDATE DB2Admin.Impiegato Impiegato
  SET Stipendio = Stipendio + 2000
  WHERE Impiegato.DipNum = 3 AND Impiegato.Nome = 'Andrea'
```



DB2 Triggers
Example 4 (1)

(Atzeni, "Basi di dati Architetture e linee di evoluzione", 2003, p263)

Definire un trigger che, quando è cancellato un impiegato che svolge il ruolo di manager di un dipartimento, cancella quel dipartimento e tutti i suoi dipendenti

```
CREATE TRIGGER DB2Admin.DelMng
AFTER DELETE ON DB2Admin.Impiegato
REFERENCING OLD AS OldRow OLD_TABLE AS OldTable
FOR EACH ROW MODE DB2SQL
WHEN ( EXISTS (SELECT *
                    FROM DB2Admin.Dipartimento
                    WHERE NomeManager = OldRow.Nome ) )
BEGIN ATOMIC
  DELETE FROM DB2Admin.Dipartimento
  WHERE NomeManager = OldRow.Nome;
END
```

DB2 Triggers
Example 4 (2)

(Atzeni, "Basi di dati Architetture e linee di evoluzione", 2003, p263)

```
DELETE FROM DB2Admin.Impiegato Impiegato
  WHERE Impiegato.Nome = 'Mario'
```

DIPNUM	NOMEMANAGER
1	Mario
2	Francesco
3	Carla

DIPNUM	NOMEMANAGER
2	Francesco
3	Carla

NOME	STIPENDIO	DIPNUM
Mario	80,000.00	1
Carla	90,000.00	3
Silvia	30,000.00	2
Marco	40,000.00	1
Francesco	70,000.00	2
Andrea	50,000.00	3

NOME	STIPENDIO	DIPNUM
Carla	90,000.00	3
Silvia	30,000.00	2
Francesco	70,000.00	2
Andrea	50,000.00	3

DB2 Triggers
Example 5 (1)

(Atzeni, "Basi di dati Architetture e linee di evoluzione", 2003, p263)

Definire un trigger che, ogni qualvolta vengano modificati gli stipendi, verifica la loro media, e se supera 50.000 cancella tutti gli impiegati il cui stipendio è stato modificato e attualmente supera 80.000

DB2 Triggers
Example 5 (2)

(Atzeni, "Basi di dati Architetture e linee di evoluzione", 2003, p263)

```
CREATE TRIGGER DB2Admin.UpdateWage2
AFTER  UPDATE OF Stipendio ON DB2Admin.Impiegato
REFERENCING OLD_TABLE AS OldTable NEW_TABLE AS NewTable
FOR EACH STATEMENT  MODE DB2SQL
BEGIN ATOMIC
  DECLARE NewAvg DECIMAL(7,2);
  SET NewAvg=(SELECT AVG(STIPENDIO FROM Impiegato);
  IF NewAvg>50000
    THEN DELETE FROM Impiegato WHERE Impiegato.Nome IN
              (SELECT NewTable.Nome FROM NewTable
               WHERE NewTable.Stipendio>80000);
  END IF;
END
```


DB2 Triggers
Example 5 (3)

(Atzeni, "Basi di dati Architetture e linee di evoluzione", 2003, p263)

```
UPDATE DB2Admin.Impiegato Impiegato
  SET Stipendio = Stipendio + 11000
  WHERE Impiegato.Nome = 'Francesco'
```

DIPNUM	NOMEMANAGER
1	Mario
2	Francesco
3	Carla

DIPNUM	NOMEMANAGER
1	Mario
3	Carla

NOME	STIPENDIO	DIPNUM
Mario	80,000.00	1
Carla	90,000.00	3
Silvia	30,000.00	2
Marco	40,000.00	1
Francesco	70,000.00	2
Andrea	50,000.00	3

NOME	STIPENDIO	DIPNUM
Carla	90,000.00	3
Mario	80,000.00	1
Marco	40,000.00	1
Andrea	50,000.00	3

DB2 Triggers
Homework (1)

Si consideri una base di dati relativa ad un sistema di prenotazione (per viaggi aerei), con le relazioni:

- **PRENOTAZIONI (Numero, Volo, Data, PostiPrenotati)**
- **DISPONIBILITA (Volo, Data, PostiTotali, PostiDisponibili)**

in cui il valore dell'attributo PostiDisponibili, per ciascun volo (e giorno), pari alla differenza fra PostiTotali e la somma dei posti prenotati su tale volo. Sulla relazione Prenotazioni sono effettuati inserimenti ed eliminazioni, ma non modifiche.

Descrivere una o più regole attive che permettano di mantenere corretto il valore di PostiDisponibili a fronte di inserimenti ed eliminazioni nella relazione Prenotazioni.

DB2 Triggers
Homework (2)

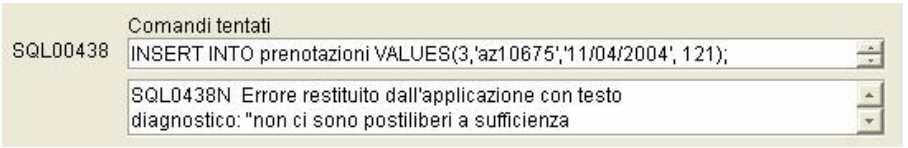
```
CREATE TRIGGER DB2Admin.PostiDisponibili
NO CASCADE BEFORE  INSERT ON DB2Admin.Prenotazioni
REFERENCING NEW AS NewRow
FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC
  IF NewRow.PostiPrenotati>(SELECT PostiDisponibili
                                FROM Disponibilita
                                WHERE Volo = NewRow.Volo)
    THEN SIGNAL SQLSTATE '70005' ('Non ci sono posti liberi a
                                    sufficienza');
  END IF;
END
```

DB2 Triggers
Homework (3)

```
CREATE TRIGGER DB2Admin.Posti
AFTER INSERT ON DB2Admin.Prenotazioni
REFERENCING NEW AS NewRow
FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC
  UPDATE Disponibilita
    SET PostiDisponibili = PostiDisponibili - NewRow.PostiPrenotati
    WHERE Disponibilita.Volo = NewRow.Volo;
END
```


DB2 Triggers
Homework (4)

```
INSERT INTO Prenotazioni VALUES(3,'az10675','11/04/2004', 121)
```



DB2 Triggers
Homework (5)

```
INSERT INTO Prenotazioni VALUES(3,'az10675','11/04/2004', 3)
```

VOLO	DATA	POSTITOTALI	POSTIDISPONIBILI
az10675	11-nov-04	120	120

VOLO	DATA	POSTITOTALI	POSTIDISPONIBILI
az10675	11-nov-04	120	117

DB2 Triggers
Homework (6)

```
CREATE TRIGGER DB2Admin.Cancella
AFTER DELETE ON DB2Admin.Prenotazioni
REFERENCING OLD AS OldRow
FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC
  UPDATE Disponibilita
  SET PostiDisponibili = PostiDisponibili + OldRow.PostiPrenotati
  WHERE Disponibilita.Volo = OldRow.Volo;
END
```

DB2 Triggers
Homework (7)

```
DELETE FROM Prenotazioni WHERE Numero = 3
```

VOLO	DATA	POSTITOTALI	POSTIDISPONIBILI
az10675	11-nov-04	120	117

VOLO	DATA	POSTITOTALI	POSTIDISPONIBILI
az10675	11-nov-04	120	120

Views

- Views are commonly used to separate the logical database schema from the physical schema.
- Unfortunately the desired transparency often falls short in the case of UPDATE, DELETE or INSERT operations, since all but the simplest views are not updatable.
- INSTEAD OF triggers makes all views updatable.

DB2 Views

We distinguish between three levels of updatability:

- **Deletable**: In order for DB2 to delete a row from a view it must be able to map the row specified in the view to one, and only one, row in a base table.
- **Updatable** : In order to update a column in a view, DB2 must not only be able to map the row specified in the view to one row in a base table, it also must be able to map the column to be updated to a column in a base table. Hence all views that are updatable must by definition also be deletable.
- **Insertable**: In order for a row to be inserted into a view, DB2 must be able to map the new row to a table and to map all the columns specified to columns in that table. Thus all views that are insertable are by definition updatable and hence deletable.

DB2 Views
Examples (1)

```
CREATE TABLE T1(c1 INT, c2 FLOAT)$
INSERT INTO T1 VALUES (5, 6.0), (6, 7.0), (5, 6.0)$

CREATE TABLE T2(c1 INT, c2 FLOAT)$
INSERT INTO T2 VALUES (5, 9.0), (5, 4.0), (7, 5.0)$
```

DB2 Views
Examples (2)

```
CREATE VIEW V2(c1, c2) AS SELECT c1, c2 * c2 FROM T1$
```

- DB2 knows which row in the view was produced by which row in the base table, so it's deletable.
- The V2.c1 column is updatable and therefore the view is updatable.
- However, V2.c2 is not updatable. The reason is that there is no way to decide the value of T1.c2 from any given V2.c2.
- Note that the view would be deletable even if no column had been updatable.

DB2 Views
Examples (3)

- From DB2 v8.1 a view is considered insertable even if only one of the columns in the view is updatable, as long as the other columns are not specified, they will be ignored.

```
INSERT INTO V2(c1) VALUES (7)$
```

- Each column value that is not provided is initialized to the implicit or explicit default value of the column. Since no explicit default was specified for T1.c2, the system will insert (7, NULL) into T1.

DB2 Views
Examples (4)

```
CREATE VIEW V3(c1, c2, c3) AS
SELECT T1.c1, T1.c2, T2.c2
FROM T1, T2
WHERE T1.c1 = T2.c1$
```

C1	C2	C3
5	6.0	4.0
5	6.0	4.0
5	6.0	9.0
5	6.0	9.0

- This view is not deletable. While each row in the view can be traced back to one row in each of the tables T1 and T2, deleting the first row (5, 6.0, 4.0) by deleting the respective rows in T1 and T2 would also, indirectly, delete the second (5, 6.0, 4.0) and one of the (5, 6.0, 9.0) rows. This behaviour is not intuitively clear.
- Since V3 is not deletable it is also not updatable.
- Similarly V3 is not insertable. Adding another row (5, 6.0, 9.0) could have varying outcomes, depending on the semantics chosen for the base table inserts.

DB2 Triggers
INSTEAD OF Triggers

- An INSTEAD OF trigger prevents DB2 from trying to interpret the view definition for the update operation, relying on the definer to come up with meaningful semantics.
- INSTEAD OF
  - it does not mean execute the trigger before attempting the triggered action;
  - it does not mean execute the trigger after attempting the triggered action;
  - it means, forget about the triggered action and execute this piece of code instead.
- INSTEAD OF triggers are always created for views, never for base tables.

DB2 Triggers - INSTEAD OF Triggers

Prototype

CREATE TRIGGER [***nome trigger***]
INSTEAD OF [***trigger event***] **ON** [***view name***]
REFERENCING [***object***] **AS** [***name***]
FOR EACH ROW MODE DB2SQL
BEGIN ATOMIC
[***triggered action***]
END

Differences with generic triggers:

- INSTEAD OF
- View name rather than subject table
- FOR EACH ROW granularity
- No Activation time and WHEN clause

DB2 Triggers - INSTEAD OF Triggers

Notes (1)

- INSTEAD OF triggers fire unconditionally:
  - have not a WHEN clause
  - have not column lists for UPDATE triggers.
- INSTEAD OF triggers have not the BEFORE/AFTER statement.
- INSTEAD OF triggers are always FOR EACH ROW triggers.
- INSTEAD OF triggers fire together with AFTER triggers.
- SQL Statements in the body of an INSTEAD OF trigger are complete semantic entities. This means, for example, an UPDATE statement inside of an INSTEAD OF trigger will cause the respective checks, RI and triggers to fire before the next statement and ultimately the next row in the transition table gets processed.

DB2 Triggers - INSTEAD OF Triggers

Notes (2)

- If you define an INSTEAD OF trigger you can do whatever you please. For this reason only users with CONTROL privileges on the view, the view definer, the SYSADM or the DBADM are allowed to create an INSTEAD OF trigger on the view.

DB2 Triggers - INSTEAD OF Triggers

Notes (3)

- **GET_DIAGNOSTICS ROW_COUNT** (a.k.a., **SQLCA.ERRD(3)**)
  Prior to DB2 V8, the number of rows modified through a view was always equal to the number of rows directly updated on a base table, where "directly" means excluding rows modified by triggers and delete cascade RI constraints. With INSTEAD OF triggers this is not true anymore, and the meaning of ROW_COUNT needs to be refined. ROW_COUNT in DB2 V8 specifies the number of rows that qualify for the update, delete or insert operation as given by the user. For an INSTEAD OF trigger on such an operation, this translates to the number of times the trigger is executed and to the cardinality of the transition tables.

DB2 Triggers - INSTEAD OF Triggers

Example (1)

```
CREATE TABLE PERSONS(ssn INT NOT NULL,
                     name VARCHAR(20) NOT NULL)$
CREATE TABLE EMPLOYEES(ssn INT NOT NULL,
                     company VARCHAR(20) NOT NULL,
                     salary DECIMAL(9,2))$
CREATE TABLE STUDENTS(ssn INT NOT NULL,
                     university VARCHAR(20) NOT NULL,
                     major VARCHAR(10))$
```

```
CREATE VIEW PERSONS_V(ssn, name, company, salary,
                      university, major) AS
SELECT P.ssn, name, company, salary, university, major
FROM PERSONS P
        LEFT OUTER JOIN EMPLOYEES E ON P.ssn = E.ssn
        LEFT OUTER JOIN STUDENTS S ON P.ssn = S.ssn$
```


DB2 Triggers - INSTEAD OF Triggers

Example (2) - INSERT

```
CREATE TRIGGER INSERT_PERSONS_V
  INSTEAD OF INSERT ON PERSONS_V
  REFERENCING NEW AS n FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    INSERT INTO PERSONS VALUES (n.ssn, n.name);
    IF n.university IS NOT NULL THEN
      INSERT INTO STUDENTS
        VALUES(n.ssn, n.university, n.major);
    END IF;
    IF n.company IS NOT NULL THEN
      INSERT INTO EMPLOYEES
        VALUES(n.ssn, n.company, n.salary);
    END IF;
  END$
```

DB2 Triggers - INSTEAD OF Triggers

Example (3) - DELETE

```
INSTEAD OF TRIGGER  FOR “DELETE” IN THE VIEW

CREATE TRIGGER DELETE_PERSONS_V
  INSTEAD OF DELETE ON PERSONS_V
  REFERENCING OLD AS o FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    DELETE FROM STUDENTS WHERE ssn = o.ssn;
    DELETE FROM EMPLOYEES WHERE ssn = o.ssn;
    DELETE FROM PERSONS WHERE ssn = o.ssn;
  END$
```


DB2 Triggers - INSTEAD OF Triggers

Example (4) – UPDATE (1)

```
CREATE TRIGGER UPDATE_PERSONS_V
  INSTEAD OF UPDATE ON PERSONS_V
  REFERENCING OLD AS o NEW AS n
  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    UPDATE PERSONS
      SET (ssn, name) = (n.ssn, n.name)
      WHERE ssn = o.ssn;
    IF n.university IS NOT NULL AND o.university IS NOT NULL
      THEN
        UPDATE STUDENTS
          SET (ssn, university, major)= (n.ssn, n.university, n.major)
          WHERE ssn = o.ssn;
```

DB2 Triggers - INSTEAD OF Triggers

Example (4) – UPDATE (2)

```
ELSE
  IF n.university IS NULL // cancellazione o è un impiegato
    THEN
      DELETE FROM STUDENTS WHERE ssn = o.ssn;
    ELSE // nuovo studente
      INSERT INTO STUDENTS
        VALUES(n.ssn, n.university, n.major);
  END IF;
  IF n.company IS NOT NULL AND o.company IS NOT NULL
    THEN
      UPDATE EMPLOYEES
        SET (ssn, company, salary) = (n.ssn, n.company, n.salary)
        WHERE ssn = o.ssn;
```


DB2 Triggers - INSTEAD OF Triggers

Example (4) – UPDATE (3)

```
ELSE
      IF n.company IS NULL//cancello impiegato o è studente
            THEN
              DELETE FROM EMPLOYEES WHERE ssn = o.ssn;
      ELSE // nuovo impiegato
              INSERT INTO EMPLOYEES
                VALUES(n.ssn, n.company, n.salary);
      END IF;
  END IF;
END$
```

DB2 Triggers - INSTEAD OF Triggers

Other applications

- **Avoiding recursive AFTER triggers**
  Using a view over the subject table and creating an INSTEAD OF trigger allows multiple updates to the subject table without causing a recursive firing of the trigger.

- **Implementing external tables**
  To extend the existing read access to a table UDF in order to also have insert, update and delete capabilities, it is feasible to define insert, update and delete-specific UDFs. Then you can create a view over the table UDF and define INSTEAD OF triggers to drive the other UDFs.

DB2 Triggers - INSTEAD OF Triggers

Potential improvements

- **View-level defaults**
  Since INSTEAD OF triggers completely detach the semantics of a view's query from its behaviour on insert, delete and update, it no longer seems sufficient to derive defaults only from the underlying tables. It seems prudent to consider views with explicit column defaults.
- **Statement-level INSTEAD OF triggers**
  While DB2 supports only row-level INSTEAD OF triggers today, there are various examples where a statement-level INSTEAD OF trigger might be beneficial.
- **Why only INSTEAD OF triggers on views?**
  Having introduced one kind of trigger to views, why stop there? Maybe BEFORE, AFTER triggers and even IDENTITY are other candidates.

Bibliography

- *Application Development Guide - Using Triggers in an Active DBMS* http://www-306.ibm.com/software/data/db2/udb/ad/v7/adg/db2a0/frm3toc.htm
- K.S.Bhogal, *DB2 Basics: Creating Your First Trigger in DB2 Universal Database*, http://www-106.ibm.com/developerworks/db2/library/techarticle/0308bhogal/0308bhogal.html
- S.Rielau, *INSTEAD OF Triggers - All Views are Updatable!*, http://www-106.ibm.com/developerworks/db2/library/techarticle/0210rielau/0210rielau.html
- D. Beneventano, *Trigger*, http://www.dbgroup.unimo.it/_domenico/BD20012002/TRIGGERDB22002.pdf
- E. Lefons, *Appunti del corso di Basi di Dati Avanzate*, http://www.di.uniba.it/~lefons/dispense/appunti%20DBA%20triennale%202003-04.pdf
- *DB2 Technical Support- DB2 Image, Audio, Video Extenders Administration and Programming: Object-oriented concepts*, http://www-306.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/document.d2w/report?fn=dmba5m25.htm