DUGI, DB2 User Group Italia

Milano, 9 Aprile 2013 -- Roma 10 Aprile 2013

DB2 for z/OS

i Trigger INSTEAD OF ;

SELECT FROM INSERT, UPDATE, MERGE, DELETE con Colonne INCLUDE

massimo MACERA
IBM Italia, Software Group Education

Le informazioni contenute in questa presentazione non sono state sottoposte a nessuna revisione formale da parte di IBM e vengono distribuite così come sono, senza nessuna garanzia, né esplicita, né implicita. Chi utilizza queste informazioni lo fa sotto la propria responsabilità. Questo materiale è allineato ad una determinata versione del prodotto DB2 Mainframe e pertanto può essere soggetto a miglioramenti o a PTF (Programming Temporary Fixes) successivi alla data di rilascio di queste pagine.

Questa presentazione e tutte quelle degli anni precedenti, possono essere scaricate in formato pdf collegandosi al seguente sito internet:

http://cid-6d23af35b0e6758e.office.live.com/browse.aspx/.Public

le Viste Read-Only e i Trigger INSTEAD OF

Definizione : Vista Read Only

Definizione: una Vista è *read-only* se si verifica almeno una delle seguenti condizioni:

- La prima clausola FROM identifica **più di una** tabella o vista, o identifica una *table function*, una *nested table*, oppure una *common table*.
- La prima clausola SELECT specifica l'opzione DISTINCT
- La fullselect più esterna contiene la clausola GROUP BY.
- La fullselect più esterna contiene la clausola HAVING.
- La prima clausola SELECT contiene una funz. di aggregazione (SUM, AVG, MIN, ...)
- La vista contiene una subquery che accede la stessa tabella della fullselect esterna.
- La prima clausola FROM identifica una Vista read-only
- La prima clausola FROM identifica una Materialized Query Table *system-maintained*.
- La fullselect esterna NON è una subselect (contiene un operatore SET).

Modificare i Dati attraverso una Vista

Un utente che fa INSERT, UPDATE, DELETE, TRUNCATE sulla Vista, ripercuote questi cambiamenti sulla tabella di base da cui la Vista deriva.
Però queste istruzioni SQL che modificano i dati reali attraverso la vista sono possibili solamente se la Vista non è "***read-only***".

Una Vista *read-only* non può essere oggetto di una istruzione SQL di INSERT, UPDATE, DELETE, TRUNCATE. La subquery di un predicato sulla tabella di base, non può riferire una Vista che include GROUP BY o HAVING.

Per fare INSERT, UPDATE, DELETE, TRUNCATE su una Vista *read only* si possono usare i Triggers INSTEAD OF. Questi Triggers, fanno credere all'utente di modificare i dati della Vista (anche se questa è *read-only*), ma in realtà vanno a modificare i dati direttamente sulla tabella di base.

Trigger INSTEAD OF

Diversamente dagli altri trigger che sono definiti solo su tabelle, **i trigger INSTEAD OF sono definiti soltanto sulle Viste.**

i trigger INSTEAD OF sono trigger definiti solamente sulle Viste, che sono eseguiti al posto delle istruzioni INSERT, UPDATE, or DELETE lanciate sulle Viste *read-only*.

in pratica, un INSERT, UPDATE, DELETE lanciato su una Vista *read-only* scatena un Trigger fa Insert, Update, Delete sulle Tabelle Base da cui la vista deriva.

Esempio di Trigger INSTEAD OF

Creiamo una Tabella WEATHER che memorizza le diverse Città e le rispettive Temperature espresse in gradi Fahrenheit:

```
CREATE TABLE WEATHER
( CITY VARCHAR(25),
  TEMPF DECIMAL(5,2) ) ;
```

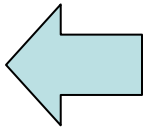
Poi creiamo una Vista read-only di nome CELSIUS_WEATHER per quegli utenti che preferiscono lavorare coi gradi Celsius anziché Fahrenheit:

```
CREATE VIEW CELCIUS_WEATHER (CITY, TEMPC)
    AS   SELECT CITY, (TEMPF – 32 ) * 5.00 /9.00
         FROM WEATHER ;
```

infine creiamo un Trigger INSTEAD OF sulla Vista CELSIUS_WEATHER che fa credere all'utente di inserire sulla vista i gradi Celsius, mentre invece il Trigger inserisce i corrispondenti gradi Fahrenheit sulla tabella WEATHER:

```
CREATE TRIGGER CW_INSERT
  INSTEAD OF INSERT ON CELCIUS_WEATHER
  REFERENCING NEW AS NEWCW  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    INSERT INTO WEATHER VALUES (NEWCW.CITY, 1.8*NEWCW.TEMPC+32)
  END ;
```

```
INSERT INTO CELSIUS_WEATHER
VALUES ('ROMA', 30) ;
```



Questa istruzione inserisce nella **tabella** WEATHER la riga **'ROMA' 86**

DROP TRIGGER e DROP VIEW

La DROP VIEW di una Vista che ha dei Trigger INSTEAD OF, elimina tutti gli INSTEAD OF trigger creati su quella Vista e i loro rispettivi Package.

DROP TRIGGER invalida i Packages (inclusi i trigger package) che dipendono dal INSTEAD OF Trigger che è stato eliminato.

Esempio:

- il corpo di un AFTER Trigger TR2 contiene una istruzione UPDATE VIEW V1
- la Vista read-only V1 ha un INSTEAD OF UPDATE Trigger TR1
- l'istruzione DROP TRIGGER TR1 fa invalidare il Package del TR2 perché TR2 dipende da TR1.

Restrizioni sui Trigger INSTEAD OF

- Su una VIEW può esserci un solo INSTEAD OF trigger per INSERT, un solo INSTEAD OF trigger per UPDATE, un solo INSTEAD OF trigger per DELETE.
- Un INSTEAD OF trigger NON può specificare la clausola WHEN.
- Un INSTEAD OF trigger NON deve specificare la clausola FOR EACH STATEMENT.
- Un INSTEAD OF trigger non può specificare UPDATE OF di una lista di colonne.
- Un INSTEAD OF trigger non può specificare una Vista definita con la clausola WITH CHECK OPTION.
- Un INSTEAD OF trigger non può specificare Viste che referenziano dati codificati con differenti Schemi di Codifica o valori CCSID.
- Un INSTEAD OF trigger non funziona con Cursor Update né Cursor Delete.

SELECT FROM INS, UPD, MRG, DEL con Colonne INCLUDE

Non l'Utente ma il DB2 scrive i dati

Talvolta non sono le applicazioni ad inserire direttamente i dati in tabella, ma è lo stesso DB2 che inserisce i dati, a fronte di:

-- valori di default,
-- identity columns,
-- colonne ROWID,
-- sequences,
-- before trigger

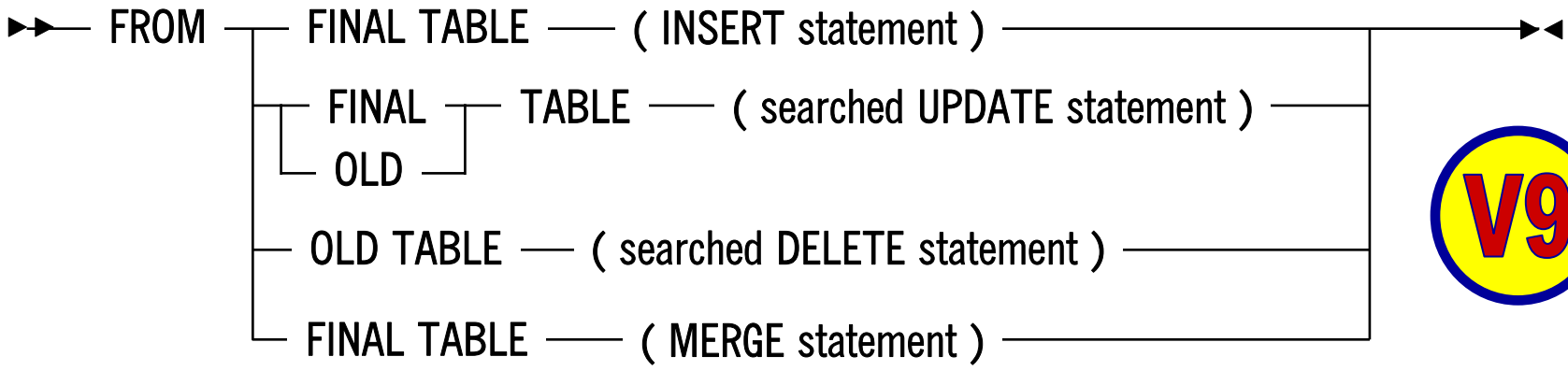
In questi casi, gli utenti hanno bisogno di determinare immediatamente i valori inseriti in tabella dal DB2. Per questo:

il DB2 V8 introdusse l'istruzione
- SELECT ... FROM INSERT.

il DB2 9 offre anche la possibilità di fare
- SELECT ... FROM MERGE,
- SELECT ... FROM UPDATE e
- SELECT ... FROM DELETE.

Nelle istruzioni SELECT ... FROM INSERT-UPDATE-DELETE-MERGE il passo che viene eseguito per primo è l'esecuzione delle istruzioni INSERT, UPDATE, DELETE, o MERGE. La parte di SELECT semplicemente ci mostra i valori che sono stati rispettivamente inseriti, aggiornati, cancellati o mergiati.

Sintassi di SELECT ... FROM ...



ESEMPI

```
SELECT * FROM OLD TABLE
 ( DELETE FROM employee
    WHERE emp_id = 12345 ) ;
```

```
SELECT numord INTO :hv-numord
FROM FINAL TABLE
 ( INSERT INTO ordini (numord, ...)
   VALUES (NEXT VALUE FOR seq_ordini, ...) ) ;
```

```
SELECT COL5 INTO :C5-HV  FROM FINAL TABLE
      (INSERT (COL1, COL2, COL5, COL7) INTO MY_TABLE
       VALUES ('MAX', 'MACERA', CURRENT DATE, 'IBM') ) ;
```


Esempi di SELECT … FROM INSERT -- 1/3

Colonna ROWID

La tabella EMP abbia 1000 righe. Noi vogliamo vedere i primi 5 valori della colonna EMP_ROWID [ROWID NOT NULL GENERATED ALWAYS] che sono stati generati e inseriti dal DB2 nella tabella EMP_PHOTO_RESUME:

```
DECLARE CS1 CURSOR FOR
  SELECT EMP_ROWID
  FROM FINAL TABLE ( INSERT INTO EMP_PHOTO_RESUME (EMPNO)
                       SELECT EMPNO FROM EMP              )
  FETCH FIRST 5 ROWS ONLY;
```

Vengono inserite 1000 righe in EMP_PHOTO_RESUME, ma solo le prime 5 sono restituite.

Esempi di SELECT ... FROM INSERT -- 2/3

Valori di DEFAULT

```
CREATE TABLE MIEI_PROGETTI
  ( COD_PROGETTO INTEGER NOT NULL,
    NOME_PROGETTO CHAR(20) NOT NULL WITH DEFAULT 'NOME PROJ NON DEFINITO',
    COD_REPARTO CHAR(8),
    COD_CAPO_PROG . . . . .  ) ;

SELECT NOME_PROGETTO INTO :nome_hv
FROM FINAL TABLE
   ( INSERT INTO MIEI_PROGETTI (COD_PROGETTO, COD_REPARTO, COD_CAPO_PROG)
      VALUES (:codprog-hv, :codreparto-hv, :codcapoprog-hv) ) ;
```

Esempi di SELECT … FROM INSERT -- 3/3

Identity Column

```
CREATE TABLE EMPLOYEE
 ( EMPNO INTEGER GENERATED ALWAYS AS IDENTITY,
  .....

DECLARE CS2 SCROLL CURSOR WITH ROWSET POSITIONING FOR
   SELECT EMPNO, NAME
   FROM FINAL TABLE
     ( INSERT INTO EMPLOYEE (NAME, TELEF) FOR 3 ROWS
       VALUES (:HVA1, :HVA2) ) ORDER BY INPUT SEQUENCE ;
```

SELECT ... FROM UPDATE

Mentre stiamo facendo UPDATE delle righe, è possibile fare SELECT dei valori che si stanno aggiornando, semplicemente specificando l'istruzione UPDATE nella clausola FROM della istruzione SELECT. Quando facciamo Update di una o più righe in tabella, possiamo reperire:

- il valore generato automaticamente dal DB2, ad es. per una colonna identity o ROWID;
- qualunque valore di default delle colonne;
- tutti i valori di una riga aggiornata, senza specificare i nomi individuali delle colonne.

In molti casi, è possibile usare l'istruzione SELECT FROM UPDATE, con la clausola FINAL TABLE. La FINAL TABLE contiene le righe appena aggiornate dall'Update che si sta eseguendo in quel momento sulla tabella (o vista).

```
SELECT sum(salary) INTO :hv-salary
FROM FINAL TABLE
  ( UPDATE emp
    SET salary = salary * 1.05
    WHERE job = 'DESIGNER' ) ;
```

```
DECLARE CS1 CURSOR FOR
SELECT lastname, bonus
FROM FINAL TABLE
(UPDATE emp
SET bonus = bonus * 1.3
WHERE job = 'DESIGNER');
FETCH CS1 INTO :lastname, :bonus;
```

Se per questa forma di Update si vogliono vedere i cambiamenti riga per riga, occorre dichiarare un Cursore per la SELECT

SELECT ... FROM DELETE

Mentre stiamo facendo DELETE di righe, è possibile fare SELECT delle righe che si stanno cancellando, semplicemente specificando l'istruzione DELETE nella clausola FROM della istruzione SELECT. Quando facciamo DELETE di una o più righe di tabella, possiamo reperire:

- qualunque valore di default delle colonne;
- tutti i valori della riga cancellata, senza specificare i nomi individuali delle colonne;
- valori calcolati basati sulle righe cancellate.

L'istruzione SELECT FROM DELETE deve utilizzare la clausola FROM OLD TABLE per reperire i valori cancellati. La OLD TABLE contiene le righe appena cancellate dal Delete che si sta eseguendo in quel momento sulla tabella (o vista).

```
SELECT sum(salary) INTO :salary
FROM OLD TABLE
(DELETE FROM emp
WHERE job = 'OPERATOR') ;
```

Nota che qui non è FINAL TABLE, ma OLD TABLE

```
DECLARE CS1 CURSOR FOR
SELECT YEAR(CURRENT DATE – HIREDATE)
FROM OLD TABLE
(DELETE FROM emp
 WHERE job = 'ANALYST');
FETCH CS1 INTO :years_of_service;
```

Se per questa forma di Delete si vogliono vedere le cancellazioni riga per riga, occorre dichiarare un Cursore per la SELECT

SELECT ... FROM MERGE

Mentre stiamo facendo MERGE è possibile fare SELECT di quelle stesse righe [aggiornate o cancellate con MERGE], semplicemente specificando l'istruzione MERGE nella clausola FROM dell'istruzione SELECT. Quando si fa MERGE di una o più righe in una tabella, è possibile reperire:

- i valori generati automaticamente dal DB2, ad esempio per le colonne ROWID o le colonne IDENTITY
- qualunque valore di default per le colonne
- tutti i valori di una riga risultato di un MERGE, senza specificare i nomi individuali delle colonne
- i valori calcolati basati sui cambiamenti delle righe risultato del MERGE

Con l'istruzione SELECT FROM MERGE si usa la clausola FINAL TABLE.
La FINAL TABLE contiene le righe della tabella (o vista) dopo che è avvenuto il MERGE.

Le Colonne INCLUDE

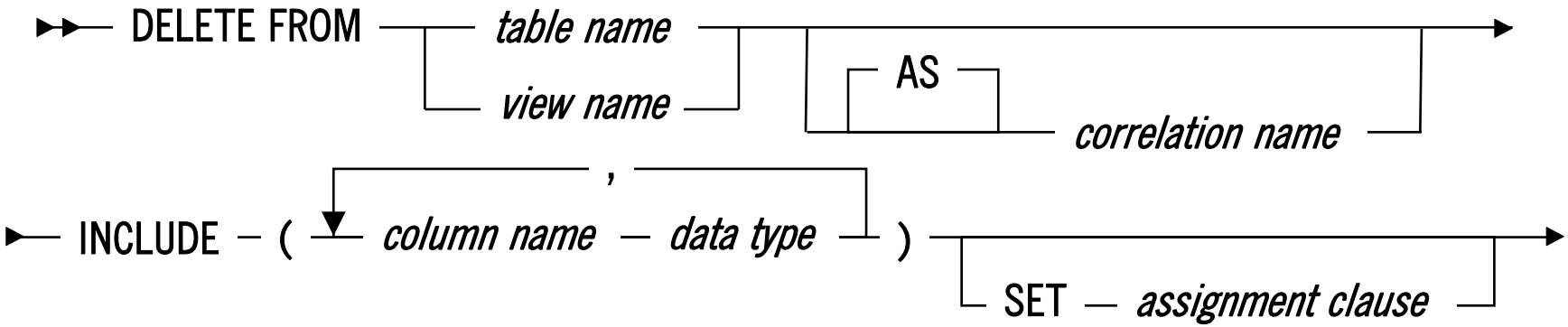
Colonne INCLUDE

- Le colonne INCLUDE sono delle colonne addizionali e virtuali che, pur non esistendo nella tabella oggetto di Ins/Upd/Del/Mrg, possono essere specificate nella tabella intermedia che contiene il risultato delle istruzioni INSERT / UPDATE / DELETE / MERGE.
- Le colonne INCLUDE non pregiudicano il risultato delle istruzioni SQL di modifica dei dati e non modificano la definizione delle tabelle base. Sono disponibili soltanto se le istruzioni INS / UPD / DEL / MRG sono annidate nella clausola FROM di una istruzione SELECT (o SELECT INTO).
- Una colonna INCLUDE può essere di qualunque *data type*, può assumere valori NULL, e deve avere un nome univoco rispetto a ogni altra colonna della tabella oggetto delle istruzioni SQL di modifica dei dati.
- È possibile riferire le colonne INCLUDE nella Select-list, nelle clausole ORDER BY, o WHERE della istruzione SELECT.
  Nel risultato della SELECT FROM INS/UPD/MRG/DEL, le Colonne INCLUDE compaiono all'estrema destra.

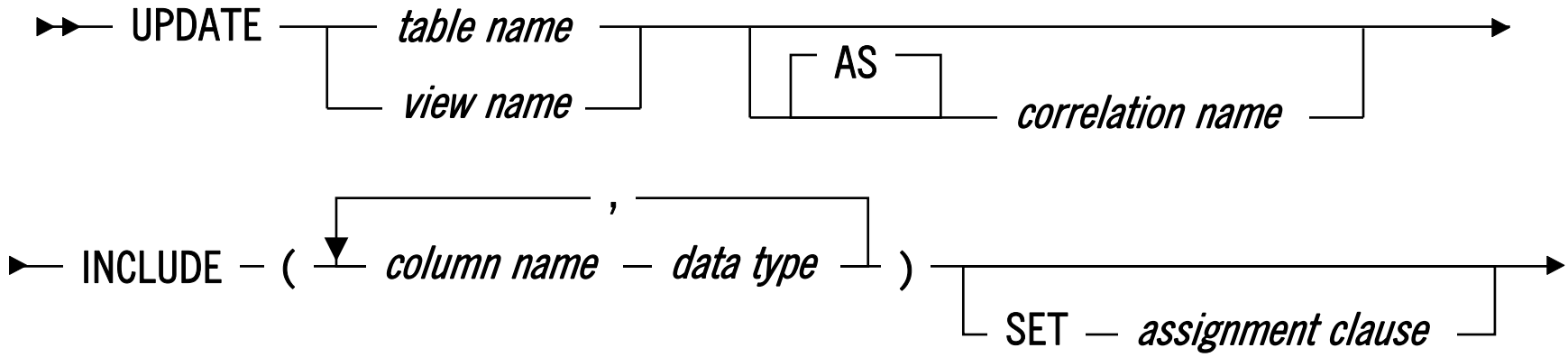
!

Sintassi per le Colonne INCLUDE

Sintassi per l'istruzione DELETE



Sintassi per l'istruzione UPDATE



Colonne INCLUDE nella SELECT… FROM INSERT

L'istruzione SELECT … FROM INSERT offre la possibilità di includere nella Select-list colonne addizionali e virtuali, i cui valori sono reperiti da una tabella (in questo esempio DEPT) specificata all'interno della INSERT. Si noti che nella tabella PROJ non ci sono colonne che danno informazioni sul Manager.


```
DECLARE CS1 CURSOR FOR
      SELECT manager_num, projname
      FROM FINAL TABLE
        (INSERT INTO proj (deptno) INCLUDE ( manager_num CHAR(6) )
          SELECT deptno, mgrno FROM dept);
```


Colonne INCLUDE nella SELECT… FROM INSERT

Per assegnare valori alle Colonne INCLUDE nelle operazioni di INSERT, si può usare la clausola VALUES. Un uso comune delle Colonne INCLUDE nelle operazioni di INSERT, consiste nel personalizzare l'ordinamento delle righe risultato. Ad esempio:

```
SELECT * FROM FINAL TABLE
      (INSERT INTO sales INCLUDE (sortkey integer) VALUES
         ( CURRENT DATE,  'Jimmy',    'Tel Aviv',  7,  3 ) ,
         ( CURRENT DATE,  'Massimo',  'Roma',      5,  1 ) ,
         ( CURRENT DATE,  'Luca',     'Carrara',   4,  2 ) )
ORDER BY sortkey
```

SALES_DATE	SALES_PERSON	REGION	SALES	SORTKEY
21/11/2012	MASSIMO	ROMA	5	1
21/11/2012	LUCA	CARRARA	4	2
21/11/2012	JIMMY	TEL AVIV	7	3

in una operazione di INSERT è anche possibile assegnare valori a una Colonna INCLUDE utilizzando una fullselect.

Colonne INCLUDE nella SELECT... FROM UPDATE

Per assegnare valori alle Colonne INCLUDE nelle operazioni di UPDATE o DELETE, si usa la clausola SET. Se nessun valore è assegnato alla Colonna INCLUDE nella clausola SET di una istruzione UPDATE o DELETE, viene restituito un valore NULL per quella colonna. Nelle istruzioni UPDATE si usano le Colonne INCLUDE anche per ricevere nel risultato, sia i valori della vecchia che della nuova colonna per una riga. Per esempio:

```
DECLARE CS1 CURSOR FOR
    SELECT lastname, Salary, Old_Salary
    FROM FINAL TABLE
      (UPDATE emp INCLUDE ( Old_Salary DECIMAL(9,2) )
       SET Salary = Salary * 1.20, Old_Salary = Salary
       WHERE job = 'INSTRUCTOR');
```

```
LASTNAME   SALARY    OLD_SALARY
---------+---------+-----------
MACERA     9600.00   8000.00
ROMNEY     8760.00   7300.00
CLINTON    9840.00   8200.00
```


Colonne INCLUDE nella SELECT… FROM DELETE

L'istruzione SELECT … FROM DELETE offre la possibilità di includere nella Select-list colonne addizionali e virtuali, i cui valori sono impostati con una clausola SET specificata all'interno della DELETE.


```
DECLARE CS1 CURSOR FOR
      SELECT lastname, salary, Years_Of_Service
      FROM OLD TABLE
        ( DELETE FROM emp INCLUDE ( Years_Of_Service INTEGER )
          SET Years_Of_Service = YEAR ( CURRENT DATE – Start_Date)
       WHERE job = 'INSTRUCTOR' );
```


Colonne INCLUDE nella SELECT... FROM MERGE

Source

S.id	S.amt
1	30
5	10
10	40
5	20
1	50

include column

Righe restituite

T.id	balance	status
1	1030	upd
5	10	ins
10	540	upd
5	30	upd
1	1080	upd

Account – target table

T.id	balance
1	1000
10	500
200	600
300	300
315	100
500	4000

```
SELECT id, balance, status  FROM FINAL TABLE
  (MERGE  INTO account AS T INCLUDE (status char(3) )
   USING (VALUES ((:hv_id, :hv_amt) FOR 5 ROWS)) AS S (id,amt)
   ON T.id = S.id
   WHEN MATCHED THEN
      UPDATE SET balance = T.balance + S.amt, status = 'upd'
   WHEN NOT MATCHED THEN
      INSERT (id, balance,status) VALUES (S.id, S.amt, 'ins' )
   NOT ATOMIC CONTINUE ON SQLEXCEPTION ) ;
```

Account - after

T.id	balance
1	1080
5	30
10	540
200	600
300	300
315	100
500	4000

RIFERIMENTI

Redbook -- DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More
revised edition, April 2005 -- SG24-6079-00

Redbook -- DB2 9 for z/OS Technical Overview
revised edition March 2010 -- SG24-7330-00

DB2 10 for z/OS, SQL Reference
edition, June 2012 -- SC19-2983-06