Practice #2 solution

The aim of the practice is to design of some *triggers* which allow to maintain the consistency between tables of different databases after the update of them.

Exercise #1

The following relations are given:

IMP (EMPNO, DEPTNO, ENAME, JOB, SAL) DIP (DEPTNO, DNAME, LOC, MINSAL, MAXSAL)

Database before the execution of the trigger

IMP table

EMPNO	DEPTNO	ENAME	JOB	SAL
7000	10	SMITH	CLERK	850
7010	10	SCOTT	ANALYST	1600
7020	10	BLAKE	SALESMAN	1400
7030	10	SMITH	MANAGER	2500
7040	20	SMITH	CLERK	800
7050	20	SCOTT	ANALYST	1600
7060	30	ADAMS	CLERK	900
7070	30	JAMES	CLERK	1000
7080	40	ALLEN	CLERK	850

DIP table

DEPTNO	DNAME	LOC	MINSAL	MAXSAL
10	ACCOUNTING	NEW YORK	100	2500
20	RESEARCH	DALLAS	150	3000
30	SALES	CHICAGO	120	2500
40	OPERATIONS	BOSTON	200	2100

Trigger

CREATE OR REPLACE TRIGGER UP_SAL AFTER UPDATE OF DNAME ON DIP FOR EACH ROW WHEN(OLD.DNAME='ACCOUNTING' AND NEW.DNAME='SALES') BEGIN --- update the salary of the employees of that department UPDATE IMP SET SAL=SAL+100 WHERE DEPTNO=:OLD.DEPTNO; END;

Update statement

UPDATE DIP SET DNAME = 'SALES' WHERE DNAME='ACCOUNTING';

Database after the execution of the trigger

IMP table

EMPNO	DEPTNO	ENAME	JOB	SAL
7000	10	SMITH	CLERK	950
7010	10	SCOTT	ANALYST	(1700
7020	10	BLAKE	SALESMAN	1500
7030	10	SMITH	MANAGER	2600
7040	20	SMITH	CLERK	800
7050	20	SCOTT	ANALYST	1600
7060	30	ADAMS	CLERK	900
7070	30	JAMES	CLERK	1000
7080	40	ALLEN	CLERK	850

DIP table

DEPTNO	DNAME	LOC	MINSAL	MAXSAL
10	SALES	NEW YORK	100	2500
20	RESEARCH	DALLAS	150	3000
30	SALES	CHICAGO	120	2500
40	OPERATIONS	BOSTON	200	2100

Comments

The trigger fires after an update of the DNAME attribute of the DIP table (mode *after*). The execution granularity is *row level* because it must access the old and the new value of DNAME to check if the update is the one requested ('ACCOUNTING'->'SALES').

Exercise #2

The following relations are given:

CARDS (CARDNO, NAME, STATUS) FLIGHTS (FLIGHTID, DEPARTURETIME, DEPARTURECITY, ARRIVALCITY, MILES) TICKETS (TICKETID, FLIGHTID, FLIGHTDATE, NAME, CARDNO) CREDITS (TICKETID, CARDNO, MILES) NOTIFY (CARDNO, NOTIFYNO, NOTIFYDATE, OLDSTATUS, NEWSTATUS, TOTALMILES)

We must design a trigger which manages the database integrity after a new ticket issue (insert of a record in the TICKETS table), updating concordantly the tables CREDITS, CARDS (if the issue implies a status change) and NOTIFY (to notify the status change).

Database before the execution of the trigger

CARDNO	NAME	STATUS
10	JOHN	SILVER
20	KEN	GOLD
30	LUCY	PREMIUM
40	BRUCE	SILVER
50	BILL	SILVER

FLIGHTS table

CARDS table

FLIGHTID	DEPARTURETIME	DEPARTURECITY	ARRIVALCITY	MILES
RN12K	00:45	ROME	NEW YORK	12000
BT18K	12:34	BERLIN	токуо	18000
PS22K	23:59	PARIS	SIDNEY	22000
LT08K	23:59	LONDON	TORONTO	8000
TS01K	11:00	TURIN	S.MARIA DI LEUCA	1000

TICKETS table

TICKETID	FLIGHTID	FLIGHTDATE	NAME	CARDNO
T01	LT08K	01-APR-07	JOHN	10

CREDITS table

TICKETID	CARDNO	MILES
T01	10	8000

Table NOTIFY is initially empty

Trigger

CREATE OR REPLACE TRIGGER NEW TICKET AFTER INSERT ON TICKETS FOR EACH ROW WHEN (NEW.CARDNO IS NOT NULL) DECLARE DISTANZA NUMBER; DISTTOTALE NUMBER; VECCHIOSTATO CHAR(10); NUOVOSTATO CHAR(10); IDNOTIFY NUMBER; BEGIN --- STEP 1 --- find the miles of the flight of the ticket SELECT MILES INTO DISTANZA FROM FLIGHTS WHERE FLIGHTID=:NEW.FLIGHTID; --- update the total miles INSERT INTO CREDITS (TICKETID, CARDNO, MILES) VALUES (:NEW.TICKETID, :NEW.CARDNO, DISTANZA); --- STEP 2 SELECT SUM(MILES) INTO DISTTOTALE FROM CREDITS WHERE CARDNO=:NEW.CARDNO; --- take the status of the customer to check if it --- is changed => see STEP 3 SELECT STATUS INTO VECCHIOSTATO FROM CARDS WHERE CARDNO=:NEW.CARDNO; IF (DISTTOTALE>30000 AND DISTTOTALE<50000 AND VECCHIOSTATO<>'GOLD') THEN ---change status from SILVER to GOLD UPDATE CARDS SET STATUS='GOLD' WHERE CARDNO=:NEW.CARDNO; ELSE IF (DISTTOTALE>50000 AND VECCHIOSTATO<>'PREMIUM') THEN ---change status from GOLD to PREMIUM UPDATE CARDS SET STATUS='PREMIUM' WHERE CARDNO=:NEW.CARDNO; END IF; END IF; ---STEP 3 --- take the status of the customer to check if --- it is changed SELECT STATUS INTO NUOVOSTATO FROM CARDS WHERE CARDNO=:NEW.CARDNO;

IF (VECCHIOSTATO<>NUOVOSTATO) THEN --- status is changed => insert in NOTIFY --- read last NOTIFYNO of the CARDNO of the customer --- to insert a new NOTIFYNO for the notification SELECT MAX (NOTIFYNO) INTO IDNOTIFY FROM NOTIFY WHERE CARDNO=:NEW.CARDNO; IF (IDNOTIFY IS NOT NULL) THEN --- there is at least another notification for the customer INSERT INTO NOTIFY (CARDNO, NOTIFYNO, NOTIFYDATE, OLDSTATUS, NEWSTATUS, TOTALMILES) VALUES (:NEW.CARDNO, IDNOTIFY+1, :NEW.FLIGHTDATE, VECCHIOSTATO, NUOVOSTATO, DISTTOTALE); ELSE --- first notification for the customer INSERT INTO NOTIFY (CARDNO, NOTIFYNO, NOTIFYDATE, OLDSTATUS, NEWSTATUS, TOTALMILES) VALUES (:NEW.CARDNO, 1, :NEW.FLIGHTDATE, VECCHIOSTATO, NUOVOSTATO, DISTTOTALE); END IF; END IF; END;

Verify the results achieved

STEP 1

INSERT INTO TICKETS (TICKETID, FLIGHTID, FLIGHTDATE, NAME, CARDNO)
VALUES ('T02', 'RN12K', '01-MAR-07', 'PIPPO', NULL);

INSERT INTO TICKETS (TICKETID, FLIGHTID, FLIGHTDATE, NAME, CARDNO)
VALUES ('T03', 'RN12k', '02-APR-07', 'BILL', 50);

TICKETS table

TICKETID	FLIGHTID	FLIGHTDATE	NAME	CARDNO
T01	LT08K	01-APR-07	JOHN	10
T02	RN12K	01-MAR-07	PIPPO	121
T03	RN12K	02-APR-07	BILL	50

CREDITS table

TICKETID	CARDNO	MILES
T01	10	8000
T03	50	12000

STEP 2

INSERT INTO TICKETS (TICKETID, FLIGHTID, FLIGHTDATE, NAME, CARDNO)
VALUES ('T04', 'RN12K', '03-MAG-07', 'BILL', 50);

INSERT INTO TICKETS (TICKETID, FLIGHTID, FLIGHTDATE, NAME, CARDNO)
VALUES ('T05', 'RN12K', '03-MAG-07', 'BILL', 50);

TICKETS table

TICKETID	FLIGHTID	FLIGHTDATE	NAME	CARDNO
T04	RN12K	03-MAG-07	BILL	50
T01	LT08K	01-APR-07	JOHN	10
T02	RN12K	01-MAR-07	PIPPO	-
T03	RN12K	02-APR-07	BILL	50
T05	RN12K	03-MAG-07	BILL	50

CREDITS table

TICKETID	CARDNO	MILES
T04	50	12000
T01	10	8000
T03	50	12000
T05	50	12000

CARDS table

CARDNO	NAME	STATUS
10	JOHN	SILVER
20	KEN	GOLD
30	LUCY	PREMIUM
40	BRUCE	SILVER
50	BILL	GOLD

STEP 3

INSERT INTO TICKETS (TICKETID, FLIGHTID, FLIGHTDATE, NAME, CARDNO)
VALUES ('T06', 'RN12K', '03-MAG-07', 'BILL', 50);

INSERT INTO TICKETS (TICKETID, FLIGHTID, FLIGHTDATE, NAME, CARDNO)
VALUES ('T07', 'RN12K', '03-MAG-07', 'BILL', 50);

TICKETS table

TICKETID	FLIGHTID	FLIGHTDATE	NAME	CARDNO
T02	RN12K	01-MAR-07	PIPPO	
T04	RN12K	03-MAG-07	BILL	50
T06	RN12K	03-MAG-07	BILL	50
T01	LT08K	01-APR-07	JOHN	10
тоз	RN12K	02-APR-07	BILL	50
T05	RN12K	03-MAG-07	BILL	50
T07	RN12K	03-MAG-07	BILL	50

CREDITS table

TICKETID	CARDNO	MILES
T04	50	12000
T06	50	12000
T01	10	8000
T03	50	12000
T05	50	12000
T07	50	12000

CARDS table

CARDNO	NAME	STATUS
10	JOHN	SILVER
20	KEN	GOLD
30	LUCY	PREMIUM
40	BRUCE	SILVER
50	BILL	PREMIUM

NOTIFY table

CARDNO	NOTIFYNO	NOTIFYDATE	OLDSTATUS	NEWSTATUS	TOTALMILES
50	1	03-MAG-07	GOLD	PREMIUM	60000

Comments

The trigger fires after the issue of a new ticket (insert in TICKETS table). As the previous one, the execution granularity is row level because we need to know if the issued ticket is associated with a card number (CARDNO IS NOT NULL), condition which fires the trigger (WHEN clause), because in the other case we don't need the trigger to fire. Trigger is written in 3 different steps so we can verify step by step the correctness. In particular at step 2 (update of STATUS in CARDS) we check the updates even if the old status (stored in the variable VECCHIOSTATO) because in some cases the update of that attribute on each execution of the trigger could be inefficient in term of execution cost (in the case the flight make the change of the total count of miles but not implies the exceed of the threshold to change STATUS).

Exercise #3

The following relations are given:

IMP (EMPNO, ENAME, JOB, SAL) SUMMARY (JOB, NUM)

Database before the execution of the trigger

IMP table

EMPNO	ENAME	JOB	SAL
1	VERDI	SEGRETARIA	800
2	ROSSI	BANCHIERE	900
3	BIANCHI	BANCHIERE	1100

SUMMARY table

JOB	NUM
SEGRETARIA	1
BANCHIERE	2

Trigger to manage the insert in IMP

CREATE OR REPLACE TRIGGER INS IMP AFTER INSERT ON IMP FOR EACH ROW DECLARE N NUMBER; M NUMBER; BEGIN --- check if there are employees that --- do the same job SELECT COUNT(*) INTO N FROM SUMMARY WHERE JOB=:NEW.JOB; IF(N=0) THEN --- it's the first employee that --- do that job INSERT INTO SUMMARY (JOB, NUM) VALUES(:NEW.JOB, 1); ELSE --- there is at least one employee --- that already does that job

SELECT NUM INTO M FROM SUMMARY WHERE JOB=:NEW.JOB;

UPDATE SUMMARY SET NUM=M+1 WHERE JOB=:NEW.JOB; END IF; END;

Update statement:

INSERT INTO IMP(EMPNO, ENAME, JOB, SAL) VALUES(4, 'NERI', 'CORRIERE', 750);

Database after the execution of the trigger

IMP table

EMPNO	ENAME	JOB	SAL
1	VERDI	SEGRETARIA	800
2	ROSSI	BANCHIERE	900
3	BIANCHI	BANCHIERE	1100
4	NERI	CORRIERE	750

SUMMARY table

JOB	NUM
SEGRETARIA	1
BANCHIERE	2
CORRIERE	1

Trigger to manage the update of JOB in IMP

CREATE OR REPLACE TRIGGER UPD_IMP AFTER UPDATE OF JOB ON IMP FOR EACH ROW DECLARE N NUMBER; M NUMBER; X NUMBER; BEGIN --- check if there are employees that --- do the same job SELECT COUNT(*) INTO N FROM SUMMARY WHERE JOB=:NEW.JOB; --- increment the new JOB IF(N=0) THEN --- it's the first employee that --- do that job INSERT INTO SUMMARY (JOB, NUM) VALUES(:NEW.JOB, 1); ELSE --- there is at least one employee --- that already does that job SELECT NUM INTO M FROM SUMMARY WHERE JOB=:NEW.JOB; UPDATE SUMMARY SET NUM=M+1 WHERE JOB=:NEW.JOB; END IF; --- decrement the old JOB SELECT NUM INTO X FROM SUMMARY WHERE JOB=:OLD.JOB; IF(X=1) THEN --- there was only one person that did that job => delete the record from SUMMARY DELETE FROM SUMMARY WHERE JOB=:OLD.JOB; ELSE --- there was other people that did that job => decrement NUM of the record in SUMMARY UPDATE SUMMARY SET NUM=X-1 WHERE JOB=:OLD.JOB; END IF; END;

Update statement

UPDATE IMP SET JOB='CORRIERE' WHERE EMPNO=2;

Database after the execution of the trigger

IMP table

EMPNO	ENAME	JOB	SAL
1	VERDI	SEGRETARIA	800
2	ROSSI	CORRIERE	900
3	BIANCHI	BANCHIERE	1100

SUMMARY table

JOB	NUM
CORRIERE	1
SEGRETARIA	1
BANCHIERE	1

Comments

Both triggers are row level because we need to access the JOB field of the inserted or updated tuple of the table IMP; for this reason we check if after the modification of IMP it exists a tuple (and if so how many of them) with the same JOB value using the SUMMARY table despite the fact that the check would be easier by counting the tuples in IMP; however row level triggers cannot access the mutating table.