Polytechnic University of Turin

Master of Science in Computer Engineering

Database Management Systems' second homework

Marco Micera

Academic Year 2017-2018

Contents

1 First exercise **2**
- 1.1 Exercise text: "Switching on/off the phone" . . . 2
- 1.2 Trigger design . . . 2
  - 1.2.1 Event . . . 2
  - 1.2.2 Execution semantics . . . 2
  - 1.2.3 Conditions . . . 2
  - 1.2.4 Actions . . . 2
- 1.3 `Active_Phone` trigger code . . . 2
- 1.4 `Off_Phone` trigger code . . . 3
- 1.5 Trigger testing . . . 4

2 Second exercise **6**
- 2.1 Exercise text: "Starting a phone call" . . . 6
- 2.2 Trigger design . . . 6
  - 2.2.1 Event . . . 6
  - 2.2.2 Execution semantics . . . 6
  - 2.2.3 Condition . . . 6
  - 2.2.4 Actions . . . 6
- 2.3 `Phone_Call_Start` trigger code . . . 6
- 2.4 Trigger testing . . . 8

3 Third exercise **10**
- 3.1 Exercise text: "Changing the maximum number of active calls" . 10
- 3.2 Trigger design . . . 10
  - 3.2.1 Event . . . 10
  - 3.2.2 Execution semantics . . . 10
  - 3.2.3 Condition . . . 10
  - 3.2.4 Actions . . . 10
- 3.3 `Max_Calls_Decrease` trigger code . . . 11
- 3.4 Trigger testing . . . 11

4 Fourth exercise **12**
- 4.1 Exercise text: "Service guarantee" . . . 12
- 4.2 Trigger design . . . 12
  - 4.2.1 Event . . . 12
  - 4.2.2 Execution semantics . . . 12
  - 4.2.3 Condition . . . 12
  - 4.2.4 Actions . . . 12
- 4.3 `Service_Guarantee` trigger code . . . 12
- 4.4 Trigger testing . . . 13

1 First exercise

1.1 Exercise text: "Switching on/off the phone"

(Insertion in the STATE_CHANGE table)
The change types are 'O' (on) and 'F' (off). When the phone is switched on, the corresponding information is stored in the TELEPHONE table. When it is switched off, the information should be removed. Furthermore, the cell to which the phone belongs should be identified and the current number of phones should be modified accordingly.

1.2 Trigger design

The trigger will be divided in two parts: one for the switching on action, and the other one for the switching off action, respectively called Active_Phone and Off_Phone as suggested by the exercise.

1.2.1 Event

- Insert on table STATE_CHANGE

1.2.2 Execution semantics

- Execution mode: AFTER the modification (for both triggers)
- Granularity: row level, each tuple is modified separately (for both triggers)

1.2.3 Conditions

- New tuple has attribute ChangeType = 'O' (for the Active_Phone trigger)
- New tuple has attribute ChangeType = 'F' (for the Off_Phone trigger)

1.2.4 Actions

1. Retrieve the possible CellId in which the telephone is in
2. Insert (delete) the corresponding row in the TELEPHONE table for the switched on (off) phone
3. If the phone is located within a cell, increase (or decrease, accordingly) the number of phones in it (CurrentPhone# attribute in the CELL table)

1.3 Active_Phone trigger code

```
CREATE OR REPLACE TRIGGER Active_Phone
    AFTER INSERT ON STATE_CHANGE FOR EACH ROW
    WHEN (NEW.ChangeType = 'O')
DECLARE
    -- The possible cell in which the phone is into
    BelongingCell NUMBER(38, 0);
BEGIN
```

```
    BEGIN
        -- Retrieving the possible cell in which the phone is into
        SELECT  CellId INTO BelongingCell
        FROM    CELL
        WHERE   :NEW.x < x1 AND :NEW.x >= x0
                AND :NEW.y < y1 AND :NEW.y >= y0;

        -- If the phone is outside any cell
        EXCEPTION WHEN NO_DATA_FOUND THEN
            -- No cell info will be then updated
            BelongingCell := NULL;
    END;

    -- The corresponding information is stored in the TELEPHONE table
    INSERT INTO TELEPHONE(PhoneNo, x, y, PhoneState)
        VALUES (:NEW.PhoneNo, :NEW.x, :NEW.y, 'On');

    -- If the phone is in a cell
    IF(BelongingCell IS NOT NULL) THEN
        -- the number of phones in the corresponding cell is increased
        UPDATE  CELL
        SET     CurrentPhone# = CurrentPhone# + 1
        WHERE   CellId = BelongingCell;
    END IF;
END;
```

1.4 Off_Phone trigger code

```
CREATE OR REPLACE TRIGGER Off_Phone
    AFTER INSERT ON STATE_CHANGE FOR EACH ROW
    WHEN (NEW.ChangeType = 'F')
DECLARE
    -- The possible cell in which the phone is into
    BelongingCell NUMBER(38, 0);
BEGIN
    BEGIN
        -- Retrieving the possible cell in which the phone is into
        SELECT  CellId INTO BelongingCell
        FROM    CELL
        WHERE   :NEW.x < x1 AND :NEW.x >= x0
                AND :NEW.y < y1 AND :NEW.y >= y0;

        -- If the phone is outside any cell
        EXCEPTION WHEN NO_DATA_FOUND THEN
            -- No cell info will be then updated
            BelongingCell := NULL;
    END;

    -- The corresponding phone record is removed
    DELETE FROM TELEPHONE WHERE PhoneNo = :NEW.PhoneNo;
```

```
    -- If the phone was in a cell
    IF(BelongingCell IS NOT NULL) THEN
        -- the number of phones in the corresponding cell is decreased
        UPDATE  CELL
        SET     CurrentPhone# = CurrentPhone# - 1
        WHERE   CellId = BelongingCell;
    END IF;
END;
```

1.5 Trigger testing

Initially, the given database has all empty tables except for the `CELL` table, which has the following entries:

	CELLID	X0	Y0	X1	Y1	CURRENTPHONE#	MAXCALLS
1	1	0	0	10	10	0	3
2	2	10	0	20	10	0	20
3	3	0	10	10	20	0	20
4	4	10	10	20	20	0	20

After the execution of:

```
INSERT INTO STATE_CHANGE(ChangeId, TimeStamp, PhoneNo, x, y, ChangeType)
        VALUES (1, sysdate, '333000010', 3, 3, 'O');
INSERT INTO STATE_CHANGE(ChangeId, TimeStamp, PhoneNo, x, y, ChangeType)
        VALUES (2, sysdate, '333000009', 15, 15, 'O');
```

Two rows will be inserted in the `STATE_CHANGE` table:

	CHANGEID	TIMESTAMP	PHONENO	X	Y	CHANGETYPE
1	1	18-NOV-17	333000010	3	3	O
2	2	18-NOV-17	333000009	15	15	O

As well as two new entries in the `TELEPHONE` table:

	PHONENO	X	Y	PHONESTATE
1	333000010	3	3	On
2	333000009	15	15	On

The number of current phones (`CurrentPhone#` in the `CELL` table) in each cell is modified as follows:

	CELLID	X0	Y0	X1	Y1	CURRENTPHONE#	MAXCALLS
1	1	0	0	10	10	1	3
2	2	10	0	20	10	0	20
3	3	0	10	10	20	0	20
4	4	10	10	20	20	1	20

After the execution of:

```
INSERT INTO STATE_CHANGE(ChangeId, TimeStamp, PhoneNo, x, y, ChangeType)
       VALUES (3, sysdate, '333000009', 15, 15, 'F');
```

Another row will be inserted in the `STATE_CHANGE` table:

	CHANGEID	TIMESTAMP	PHONENO	X	Y	CHANGETYPE
1	1	18-NOV-17	333000010	3	3	O
2	2	18-NOV-17	333000009	15	15	O
3	3	18-NOV-17	333000009	15	15	F

Causing the elimination of the corresponding entry in the `TELEPHONE` table:

	PHONENO	X	Y	PHONESTATE
1	333000010	3	3	On

The number of current phones in the corresponding `CELL` entry decreases by 1 unit:

	CELLID	X0	Y0	X1	Y1	CURRENTPHONE#	MAXCALLS
1	1	0	0	10	10	1	3
2	2	10	0	20	10	0	20
3	3	0	10	10	20	0	20
4	4	10	10	20	20	0	20

2 Second exercise

2.1 Exercise text: "Starting a phone call"

(Insertion in the `STATE_CHANGE` table)
The change type is `'C'`. If the cell in which the phone is located does not exceed the maximum number of calls it can manage (`MaxCalls` attribute), the phone state should become `'Active'`. If instead the cell exceeds the maximum call number, the phone call cannot be initiated. In this case, the information on the exception should be inserted in the `EXCEPTION_LOG` table. The `ExId` attribute is a counter, which is unique for a given cell.

2.2 Trigger design

2.2.1 Event

- Insert on table `STATE_CHANGE`

2.2.2 Execution semantics

- Execution mode: `AFTER` the modification
- Granularity: row level, each tuple is modified separately

2.2.3 Condition

- New tuple has attribute `ChangeType = 'C'`

2.2.4 Actions

1. Retrieve the possible `CellId` in which the telephone is in
2. If the telephone is in a cell:
   (a) Retrieve the cell's capacity in terms of the maximum supported number of active phone calls at the same time (`MaxCalls` attribute in the `CELL` table) and its vertexes
   (b) Retrieve the number of active phone calls in the cell
   (c) If the new call exceeds the cell's capacity, insert a row in the `EXCEPTION_LOG` table, after computing its `ExID`; otherwise, set the corresponding `PhoneState` attribute in the `TELEPHONE` table to `'Active'`

2.3 `Phone_Call_Start` trigger code

```
CREATE OR REPLACE TRIGGER Phone_Call_Start
    AFTER INSERT ON STATE_CHANGE FOR EACH ROW
    WHEN (NEW.ChangeType = 'C')
DECLARE
    -- The possible cell in which the phone is into
    BelongingCell NUMBER(38, 0);

    -- Cell's vertexes
```

```
    cellX0 DECIMAL(7, 2);
    cellY0 DECIMAL(7, 2);
    cellX1 DECIMAL(7, 2);
    cellY1 DECIMAL(7, 2);

    -- Maximum number of calls of the interested cell
    CellCallCapacity SMALLINT;

    -- Number of phones calling in the same cell
    ConcurrentCalls NUMBER;

    -- In case of exception, ID to be used
    NextExceptionId INTEGER;
BEGIN
    BEGIN
        -- Retrieving the possible cell in which the phone is into
        SELECT  CellId INTO BelongingCell
        FROM    CELL
        WHERE   :NEW.x < x1 AND :NEW.x >= x0
                AND :NEW.y < y1 AND :NEW.y >= y0;

        -- If the phone is outside any cell
        EXCEPTION WHEN NO_DATA_FOUND THEN
            -- No cell info will be then updated
            BelongingCell := NULL;
    END;

    -- Checking if the phone is in a cell
    IF(BelongingCell IS NOT NULL) THEN
        -- Retrieving:
        ---- The max number of calls in the cell
        ---- Cell's vertexes
        SELECT  MaxCalls, x0, y0, x1, y1
            INTO CellCallCapacity, cellX0, cellY0, cellX1, cellY1
        FROM    CELL
        WHERE   CellId = BelongingCell;

        -- Retrieving the number of phones calling in the cell
        SELECT  COUNT(*) INTO ConcurrentCalls
        FROM    TELEPHONE
        WHERE   PhoneState = 'Active'
                AND x < cellX1 AND x >= cellX0
                AND y < cellY1 AND y >= cellY0;

        -- The cell can manage this call
        IF(ConcurrentCalls + 1 <= CellCallCapacity) THEN
            -- Setting the phone's state to Active
            UPDATE TELEPHONE
                SET PhoneState = 'Active'
                WHERE PhoneNo = :NEW.PhoneNo;
```

```
        -- The call exceeds the cell's capacity
        ELSE
            -- Calculating the new exception ID
            SELECT  MAX(ExID) + 1 INTO NextExceptionId
            FROM    EXCEPTION_LOG
            WHERE   CellId = BelongingCell;

            -- If it's the first exception for this cell
            IF(NextExceptionId IS NULL) THEN
                -- No cell info will be then updated
                NextExceptionId := 1;
            END IF;

            -- Inserting an exception in the log table
            INSERT INTO EXCEPTION_LOG(ExId, CellId, ExceptionType)
                VALUES (NextExceptionId, BelongingCell, 'C');
        END IF;
    END IF;
END;
```

2.4 Trigger testing

After the execution of:

```
INSERT INTO STATE_CHANGE(ChangeId, TimeStamp, PhoneNo, x, y, ChangeType)
        VALUES (4, sysdate, '333000001', 3, 3, 'O');
INSERT INTO STATE_CHANGE(ChangeId, TimeStamp, PhoneNo, x, y, ChangeType)
        VALUES (6, sysdate, '333000004', 5, 5, 'O');
INSERT INTO STATE_CHANGE(ChangeId, TimeStamp, PhoneNo, x, y, ChangeType)
        VALUES (7, sysdate, '333000004', 5, 5, 'C');
INSERT INTO STATE_CHANGE(ChangeId, TimeStamp, PhoneNo, x, y, ChangeType)
        VALUES (8, sysdate, '333000001', 3, 3, 'C');
INSERT INTO STATE_CHANGE(ChangeId, TimeStamp, PhoneNo, x, y, ChangeType)
        VALUES (9, sysdate, '333000010', 3, 3, 'C');
```

there will be 3 active phones in cell 1:

	PHONENO	X	Y	PHONESTATE
1	333000010	3	3	Active
2	333000001	3	3	Active
3	333000004	5	5	Active

which has a maximum active call capacity of 3.

	CELLID	X0	Y0	X1	Y1	CURRENTPHONE#	MAXCALLS
1	1	0	0	10	10	3	3
2	2	10	0	20	10	0	20
3	3	0	10	10	20	0	20
4	4	10	10	20	20	0	20

Upon inserting these two rows in the `STATE_CHANGE` table:

```
INSERT INTO STATE_CHANGE(ChangeId, TimeStamp, PhoneNo, x, y, ChangeType)
        VALUES (10, sysdate, '333000020', 4, 4, 'O');
INSERT INTO STATE_CHANGE(ChangeId, TimeStamp, PhoneNo, x, y, ChangeType)
        VALUES (11, sysdate, '333000020', 4, 4, 'C');
```

a new phone is inserted in the `TELEPHONE` table:

	PHONENO	X	Y	PHONESTATE
1	333000010	3	3	Active
2	333000001	3	3	Active
3	333000004	5	5	Active
4	333000020	4	4	On

Notice that this phone is not in the `'Active'` state, since the first cell has already the maximum number of phone calls going on. Hence, a new row in `EXCEPTION_LOG` has been inserted, with type `'C'`, indicating a *concurrent phone **c**all overflow*.

	EXID	CELLID	EXCEPTIONTYPE
1	1	1	C

Of course, all the previous `STATE_CHANGE` rows have been inserted in the table, making the latter as follows:

	CHANGEID	TIMESTAMP	PHONENO	X	Y	CHANGETYPE
1	1	18-NOV-17	333000010	3	3	O
2	2	18-NOV-17	333000009	15	15	O
3	3	18-NOV-17	333000009	15	15	F
4	4	19-NOV-17	333000001	3	3	O
5	6	19-NOV-17	333000004	5	5	O
6	7	19-NOV-17	333000004	5	5	C
7	8	19-NOV-17	333000001	3	3	C
8	9	19-NOV-17	333000010	3	3	C
9	10	19-NOV-17	333000020	4	4	O
10	11	19-NOV-17	333000020	4	4	C

3 Third exercise

3.1 Exercise text: "Changing the maximum number of active calls"

(Update of `MaxCalls` in the `CELL` table)
The maximum number of active calls related to a single cell may be reduced by the cellular phone network for managing issues (decrease of the `MaxCalls` value in the `CELL` table). The update on the `MaxCalls` attribute for a single cell could cause an inconsistent situation in which the `MaxCalls` value in the `CELL` table becomes smaller than the number of currently Active phones (`PhoneState='Active'`) in the considered cell. If so, the corresponding `MaxCalls` attribute needs to be updated with the number of currently Active phones (`PhoneState='Active'`) in the considered cell.

3.2 Trigger design

3.2.1 Event

- Update of `MaxCalls` in the `CELL` table

3.2.2 Execution semantics

- Execution mode: `BEFORE` the modification, to modify the `MaxCalls` value before the event takes place
- Granularity: row level, each tuple is modified separately

3.2.3 Condition

- The `MaxCalls` value is decreased

3.2.4 Actions

1. Retrieve the number of active phone calls in the cell
2. If the new `MaxCalls` value is less than the number of active phone calls in the cell:
   (a) Set the new `MaxCalls` value equal to the number of active phone calls in the cell

3.3 Max_Calls_Decrease trigger code

```
CREATE OR REPLACE TRIGGER Max_Calls_Decrease
    BEFORE UPDATE OF MaxCalls ON CELL FOR EACH ROW
    WHEN (NEW.MaxCalls < OLD.MaxCalls)
DECLARE
    ActiveCallsInCell NUMBER;
BEGIN
    -- Retrieving the number of phones calling in the cell
    SELECT  COUNT(*) INTO ActiveCallsInCell
    FROM    TELEPHONE
    WHERE   PhoneState = 'Active'
            AND x < :NEW.x1 AND x >= :NEW.x0
            AND y < :NEW.y1 AND y >= :NEW.y0;

    -- If the update is inconsistent
    IF(ActiveCallsInCell > :NEW.MaxCalls) THEN
        :NEW.MaxCalls := ActiveCallsInCell;
    END IF;
END;
```

3.4 Trigger testing

The initial `CELL` table state is reported below:

	CELLID	X0	Y0	X1	Y1	CURRENTPHONE#	MAXCALLS
1	1	0	0	10	10	4	3
2	2	10	0	20	10	0	20
3	3	0	10	10	20	0	20
4	4	10	10	20	20	0	20

After the execution of:

```
UPDATE CELL SET MaxCalls = MaxCalls-2;
```

All rows will be update accordingly, except for cell 1, that already has 3 active phone calls in it and decreasing the `MaxCalls` attribute from 3 to 1 would bring the table in an inconsistent state: this is why the trigger takes action, and sets the latter to the number of active phone calls (3):

	CELLID	X0	Y0	X1	Y1	CURRENTPHONE#	MAXCALLS
1	1	0	0	10	10	4	3
2	2	10	0	20	10	0	18
3	3	0	10	10	20	0	18
4	4	10	10	20	20	0	18

4 Fourth exercise

4.1 Exercise text: "Service guarantee"

(Update of `MaxCalls` in the `CELL` table)
The cellular phone network administrator needs to guarantee a minimum level service. In particular, the maximum number of active calls, by considering all cells of the network, needs to be always greater than 30. Thus, updates on the `MaxCalls` attribute in the `CELL` table must always satisfy the constraint. When an update instruction on `MaxCalls` attribute in the `CELL` table does not satisfy this constraint, the trigger will raise an application error to disallow the instruction that activates the trigger.

4.2 Trigger design

4.2.1 Event

- Update of `MaxCalls` in the `CELL` table

4.2.2 Execution semantics

- Execution mode: `AFTER` the modification
- Granularity: to capture the effect on the entire modification

4.2.3 Condition

- No condition, the trigger fires every time

4.2.4 Actions

1. Compute the total amount of guaranteed phone calls in the network
2. Prevent any `MaxCalls` modification that violates the constraint

4.3 `Service_Guarantee` trigger code

```
CREATE OR REPLACE TRIGGER Service_Guarantee
        AFTER UPDATE OF MaxCalls ON CELL
DECLARE
    TotalMaxCalls INTEGER;
BEGIN
        SELECT SUM(MaxCalls) INTO TotalMaxCalls
        FROM   CELL;

        IF(TotalMaxCalls < 30) THEN
            raise_application_error(
                -20514,
            'The overall network should guarantee at least 30 calls'
                );
        END IF;
END;
```

4.4 Trigger testing

The initial `CELL` table state is reported below:

	CELLID	X0	Y0	X1	Y1	CURRENTPHONE#	MAXCALLS
1	1	0	0	10	10	4	3
2	2	10	0	20	10	0	18
3	3	0	10	10	20	0	18
4	4	10	10	20	20	0	18

After the execution of:

```
UPDATE CELL SET MaxCalls = MaxCalls-1;
```

the `Service_Guarantee` trigger won't prevent the operation as the total amount of guaranteed phone calls over the network remains greater than 30:

	CELLID	X0	Y0	X1	Y1	CURRENTPHONE#	MAXCALLS
1	1	0	0	10	10	4	3
2	2	10	0	20	10	0	17
3	3	0	10	10	20	0	17
4	4	10	10	20	20	0	17

Notice that the first cell's `MaxCalls` value didn't drop from 3 to 2, as the `Max_Calls_Decrease` trigger prevented this kind of operation.

This time, after the execution of:

```
UPDATE CELL SET MaxCalls = MaxCalls-10;
```

the `Service_Guarantee` trigger does prevent the operation as the total amount of guaranteed phone calls over the network would drop to 24. The trigger will then raise an application error:

```
Error starting at line : 21 in command -
UPDATE CELL SET MaxCalls = MaxCalls-10
Error report -
ORA-20514: The overall network should guarantee at least 30 calls
ORA-06512: at "SYSTEM.SERVICE_GUARANTEE", line 8
ORA-04088: error during execution of trigger 'SYSTEM.SERVICE_GUARANTEE'
```

and the `CELL` table does not get modified.