Homework #3

Matteo Corain S256654

Data Science and Database Technology – A.Y. 2018-19

1 Trigger #1

1.1 Structure

Event	The triggering event is represented by an insertion on the table STATE_CHANGE (mutating table).
Condition	The triggering conditions cannot be expressed by means of simple predicates to be put in the WHEN clause.
Action	In case the change type is 'O', the trigger needs to: • Insert the information on the phone in the TELEPHONE table; • Retrieve the cell it will be connected to; • Increase the number of phones connected to the cell. In case the change type is 'F', the trigger needs to: • Delete the information on the phone from the TELEPHONE table; • Retrieve the cell it was connected to; • Decrease the number of phones connected to the cell.
Granularity	The trigger granularity is row-level: we need to access information on the particular tuple which has been inserted in the mutating table.
Execution mode	The trigger execution mode is after: there is no need to make modifications on the tuple inserted in the mutating table.

1.2 Code

```
CREATE OR REPLACE TRIGGER PhoneSwitching
AFTER INSERT ON STATE_CHANGE
FOR EACH ROW
DECLARE
    CurrentCell NUMBER;
BEGIN
    -- retrieve the corresponding cell
    SELECT CellId INTO CurrentCell
    FROM CELL
    WHERE X0 <= :NEW.X AND :NEW.X < X1
    AND Y0 <= :NEW.Y AND :NEW.Y < Y1;

    IF (:NEW.ChangeType = 'O') THEN
        -- insert the new phone into the TELEPHONE table
        INSERT INTO TELEPHONE(PhoneNo, X, Y, PhoneState)
        VALUES (:NEW.PhoneNo, :NEW.X, :NEW.Y, 'On');

        -- update the number of phones
        UPDATE CELL
        SET CurrentPhone# = CurrentPhone# + 1
        WHERE CellId = CurrentCell;
    ELSIF (:NEW.ChangeType = 'F') THEN
        -- remove the phone from the TELEPHONE table
```

```
            DELETE FROM TELEPHONE
            WHERE PhoneNo = :NEW.PhoneNo;

            -- update the number of phones
            UPDATE CELL
            SET CurrentPhone# = CurrentPhone# - 1
            WHERE CellId = CurrentCell;
        END IF;
    END;
```

1.3 Verification

1.3.1 Test #1

After the first insertion on STATE_CHANGE, the TELEPHONE and CELL tables show the following contents:

PhoneNo	X	Y	PhoneState
333000010	3	3	On

CellId	X0	Y0	X1	Y1	CurrentPhone#	MaxCalls
1	0	0	10	10	1	3
2	10	0	20	10	0	20
3	0	10	10	20	0	20
4	10	10	20	20	0	20

It is possible to notice that the phone 333000010 has been successfully switched on and connected to the cell #1, for which the current phone number was incremented by one.

1.3.2 Test #2

After the second insertion on STATE_CHANGE, the TELEPHONE and CELL tables show the following contents:

PhoneNo	X	Y	PhoneState
333000010	3	3	On
333000009	15	15	On

CellId	X0	Y0	X1	Y1	CurrentPhone#	MaxCalls
1	0	0	10	10	1	3
2	10	0	20	10	0	20
3	0	10	10	20	0	20
4	10	10	20	20	1	20

It is possible to notice that the phone 333000009 has been successfully switched on and connected to the cell #4, for which the current phone number was incremented by one.

1.3.3 Test #3

After the third insertion on STATE_CHANGE, the TELEPHONE and CELL tables show the following contents:

PhoneNo	X	Y	PhoneState
333000010	3	3	On

CellId	X0	Y0	X1	Y1	CurrentPhone#	MaxCalls
1	0	0	10	10	1	3
2	10	0	20	10	0	20
3	0	10	10	20	0	20
4	10	10	20	20	0	20

It is possible to notice that the phone 333000009 has been successfully switched off and disconnected from the cell #4, for which the current phone number was decremented by one.

2 Trigger #2

2.1 Structure

Event	The triggering event is represented by an insertion on the table `STATE_CHANGE` (mutating table).
Condition	The triggering condition is that the change type is equal to 'C'.
Action	The trigger needs to: • Retrieve the cell the phone is connected to; • Count the number of active (i.e. calling) phones in the corresponding cell; • If the call can be allocated, set the phone state to be 'Active'; • If the call cannot be allocated, insert a tuple in the `EXCEPTION_LOG` table, after computing the `ExId` to use.
Granularity	The trigger granularity is row-level: we need to access information on the particular tuple which has been inserted in the mutating table.
Execution mode	The trigger execution mode is after: there is no need to make modifications on the tuple inserted in the mutating table.

2.2 Code

```
CREATE OR REPLACE TRIGGER StartCall
AFTER INSERT ON STATE_CHANGE
FOR EACH ROW
WHEN (NEW.ChangeType = 'C')
DECLARE
    CurrentCell NUMBER;
    CellX0 NUMBER;
    CellX1 NUMBER;
    CellY0 NUMBER;
    CellY1 NUMBER;
    CellMaxCalls NUMBER;
    CallingCount NUMBER;
    LastException NUMBER;
BEGIN
    -- retrieve the corresponding cell
    SELECT CellId, X0, X1, Y0, Y1, MaxCalls
    INTO CurrentCell, CellX0, CellX1, CellY0, CellY1, CellMaxCalls
    FROM CELL
    WHERE X0 <= :NEW.X AND :NEW.X < X1
    AND Y0 <= :NEW.Y AND :NEW.Y < Y1;

    -- count phones calling in the cell
    SELECT COUNT(*) INTO CallingCount
    FROM TELEPHONE
    WHERE CellX0 <= :NEW.X AND :NEW.X < CellX1
    AND CellY0 <= :NEW.Y AND :NEW.Y < CellY1
    AND PhoneState = 'Active';

    -- check if call can be allocated
    IF (CallingCount < CellMaxCalls) THEN
```

```
            -- telephone becomes active
            UPDATE TELEPHONE
            SET PhoneState = 'Active'
            WHERE PhoneNo = :NEW.PhoneNo;
        ELSE
            -- retrieve the exception number to use
            SELECT MAX(ExId) INTO LastException
            FROM EXCEPTION_LOG
            WHERE CellId = CurrentCell;

            -- set to 0 if NULL
            IF (LastException IS NULL) THEN
                LastException := 0;
            END IF;

            -- insert new exception
            INSERT INTO EXCEPTION_LOG(ExId, CellId, ExceptionType)
            VALUES (LastException + 1, CurrentCell, 'C');
        END IF;
    END;
```

2.3 Verification

2.3.1 Test #1

After the first insertion on STATE_CHANGE, the TELEPHONE and CELL tables show the following contents:

PhoneNo	X	Y	PhoneState
333000010	3	3	On
333000001	3	3	On

CellId	X0	Y0	X1	Y1	CurrentPhone#	MaxCalls
1	0	0	10	10	2	3
2	10	0	20	10	0	20
3	0	10	10	20	0	20
4	10	10	20	20	0	20

The insertion activates trigger #1, which inserts the phone 333000001 and associates it to the cell #1.

2.3.2 Test #2

After the second insertion on STATE_CHANGE, the TELEPHONE and CELL tables show the following contents:

PhoneNo	X	Y	PhoneState
333000010	3	3	On
333000001	3	3	On
333000004	5	5	On

CellId	X0	Y0	X1	Y1	CurrentPhone#	MaxCalls
1	0	0	10	10	3	3
2	10	0	20	10	0	20
3	0	10	10	20	0	20
4	10	10	20	20	0	20

The insertion activates trigger #1, which inserts the phone 333000004 and associates it to the cell #1.

2.3.3 Test #3

After the third insertion on STATE_CHANGE, the TELEPHONE table shows the following contents:

PhoneNo	X	Y	PhoneState
333000010	3	3	On
333000001	3	3	On
333000004	5	5	Active

The phone 333000004 has been able to initiate the call, since the number of phones calling in its cell (1) is lower than the allowed maximum (3). Thus, no exceptions are reported.

2.3.4 Test #4

After the fourth insertion on STATE_CHANGE, the TELEPHONE table shows the following contents:

PhoneNo	X	Y	PhoneState
333000010	3	3	On
333000001	3	3	Active
333000004	5	5	Active

The phone 333000001 has been able to initiate the call, since the number of phones calling in its cell (2) is lower than the allowed maximum (3). Thus, no exceptions are reported.

2.3.5 Test #5

After the fifth insertion on STATE_CHANGE, the TELEPHONE table shows the following contents:

PhoneNo	X	Y	PhoneState
333000010	3	3	Active
333000001	3	3	Active
333000004	5	5	Active

The phone 333000004 has been able to initiate the call, since the number of phones calling in its cell (3) is exactly equal to the allowed maximum (3). Thus, no exceptions are reported.

2.3.6 Test #6

After the sixth insertion on STATE_CHANGE, the TELEPHONE and CELL tables show the following contents:

PhoneNo	X	Y	PhoneState
333000010	3	3	Active
333000001	3	3	Active
333000004	5	5	Active
333000020	4	4	On

CellId	X0	Y0	X1	Y1	CurrentPhone#	MaxCalls
1	0	0	10	10	4	3
2	10	0	20	10	0	20
3	0	10	10	20	0	20
4	10	10	20	20	0	20

The insertion activates trigger #1, which inserts the phone 333000020 and associates it to the cell #1.

2.3.7 Test #7

After the seventh insertion on STATE_CHANGE, the TELEPHONE table shows the following contents:

PhoneNo	X	Y	PhoneState
333000010	3	3	Active

333000001	3	3	Active
333000004	5	5	Active
333000020	4	4	On

In this case, the phone 333000020 has not been able to initiate the call, since the number of phones trying to call in its cell (4) is higher than the allowed maximum (3). Thus, its state does not change, and an exception is reported in the EXCEPTION_LOG table, whose content is shown below.

ExId	CellId	ExceptionType
1	1	C

3 Trigger #3

3.1 Structure

Event	The triggering event is represented by an update of the attribute MaxCalls on the table CELL (mutating table).
Condition	The triggering condition is that the number of maximum calls has been reduced from the previous one.
Action	The trigger needs to: • Count the number of active (i.e. calling) phones in the cell; • If it is higher than the new value of MaxCalls, update the value of the attribute to be equal to the count.
Granularity	The trigger granularity is row-level: we need to access information on the particular tuple which has been inserted in the mutating table.
Execution mode	The trigger execution mode is before: we may need to modify the tuple before it is inserted in the mutating table.

3.2 Code

```
CREATE OR REPLACE TRIGGER MaxCallChange1
BEFORE UPDATE OF MaxCalls ON CELL
FOR EACH ROW
WHEN (NEW.MaxCalls < OLD.MaxCalls)
DECLARE
    CallingCount NUMBER;
BEGIN
    -- count phones calling in the cell
    SELECT COUNT(*) INTO CallingCount
    FROM TELEPHONE
    WHERE :NEW.X0 <= X AND X < :NEW.X1
    AND :NEW.X0 <= Y AND Y < :NEW.Y1
    AND PhoneState = 'Active';

    IF (CallingCount > :NEW.MaxCalls) THEN
        -- set the new value of MaxCalls
        :NEW.MaxCalls := CallingCount;
    END IF;
END;
```

3.3 Verification

3.3.1 Test #1

After the first update on CELL, the CELL table shows the following contents:

CellId	X0	Y0	X1	Y1	CurrentPhone#	MaxCalls
1	0	0	10	10	4	2
2	10	0	20	10	0	18
3	0	10	10	20	0	18
4	10	10	20	20	0	18

It is possible to notice that the update is successful for cells #2, #3 and #4, but not for cell #1: in this case, in fact, there are two active phones in the cell, and reducing the `MaxCalls` by two (it was 3 before the update) is not feasible. The tuple is therefore updated with the number of active calls in the cell.

4 Trigger #4

4.1 Structure

Event	The triggering event is represented by an update of the attribute `MaxCalls` on the table `CELL` (mutating table).
Condition	The triggering conditions cannot be expressed by means of simple predicates to be put in the `WHEN` clause.
Action	The trigger needs to: • Compute the total number of calls supported by summing the values of the `MaxCall` attribute for all the cells; • If it is not higher than 30, raise an application error.
Granularity	The trigger granularity is statement-level: we need to access the entire mutating table to retrieve the requested measure.
Execution mode	The trigger execution mode is after: we need to access the mutating table after the update took place (assuming that the initial state of the table is correct).

4.2 Code

```
CREATE OR REPLACE TRIGGER MaxCallChange2
AFTER UPDATE OF MaxCalls ON CELL
DECLARE
    TotalMaxCalls NUMBER;
BEGIN
    -- count the total of the calls on the mutating table
    SELECT SUM(MaxCalls) INTO TotalMaxCalls
    FROM CELL;

    -- check the constraint and report errors
    IF (TotalMaxCalls <= 30) THEN
        RAISE_APPLICATION_ERROR(-20000,
        'The maximum number of calls needs to be always greater than 30');
    END IF;
END;
```

4.3 Verification

4.3.1 Test #1

After the first update on `CELL`, the `CELL` table shows the following contents:

CellId	X0	Y0	X1	Y1	CurrentPhone#	MaxCalls
1	0	0	10	10	4	2
2	10	0	20	10	0	17
3	0	10	10	20	0	17
4	10	10	20	20	0	17

In this case, the update does not violate the constraint for cells #2, #3 and #4 (while on cell #1 the update is prevented by the previous trigger) and, therefore, it is executed with no errors.

4.3.2 Test #2

After the second update on CELL, an error is risen by the DBMS:

```
ORA-20000: The maximum number of calls needs to be always greater than 30
```

The update, in fact, violates the constraint on the minimum number of cells and, therefore, it is rolled back by the system. Accordingly, the CELL table shows no changes.