Constraints, Triggers and Active Databases

Chapter 9

Integrity Constraints (Reminder)

- Key (*relation*): a list of attributes
  - Primary key (Sql: PRIMARY KEY)
  - Secondary key (UNIQUE)
- Foreign key (*inter-relation*): from R1 to R2
  - Referenced attributes (in R2) must be UNIQUE or PRIMARY KEY
  - Values of the foreign key in R1 must appear in R2
  - Sql: FOREIGN KEY (<atts>) REFERENCES R2(<atts>)

Enforcing Foreign Key Constraints

FOREIGN KEY (<atts>) REFERENCES R2(<atts>)

- **Reject** violation (default mode):
  - Insert in R1 some tuple whose value (<atts>) is not in R2: *rejected*
  - Update in R1 some tuple whose value (<atts>) is not in R2: *rejected*
  - Delete in R2 some tuple whose value (<atts>) is in R1: *rejected*
  - Update in R2 some tuple whose value (<atts>) changed and the old value appear in some tuples R1: *rejected*
- **Cascade**: Change in R2 will be updated to R1
  - Delete in R2 some tuple whose value (<atts>) is in R1: tuples with corresponding value are *deleted in R1*
  - Update in R2 some tuple whose value (<atts>) changed and the old value appear in some tuples R1: *changes made in R1*
- **Set-Null**: Change in R2 will be updated to R1
  - Delete in R2 some tuple whose value (<atts>) is in R1: value of (<atts>) are set to NULL
  - Update in R2 some tuple whose value (<atts>) changed and the old value appear in some tuples R1: value of (<atts>) are set to NULL

IC – Attribute-Based Constraints

- NULL valued prohibited (Sql: NOT NULL)
- CHECK
  - attribute-based: CHECK (*cond*) – where *cond* can be anything that can occur in a SQL query (select-from-where-*cond*), use with care!
  - tuple-based: same as attribute-based

4

IC – Schema-Level Based

- Assertions: A boolean-valued SQL expression that must be true at all time
- Sql: CREATE ASSERTION <name>

  CHECK <condition>
  - <name>: name of the assertion
  - <condition>: condition that can occur in a SQL query (select-from-where-*cond*)
    - must be true at the time the assertion is created
    - must remain true ALL the time

5

IC – Schema-Level Based – Trigger

- Element of the database schema
- General form:

  ON <*event*> IF <*condition*> THEN <*action*>
  - *Event*- request to execute database operation
  - *Condition* - predicate evaluated on database state
  - *Action* – execution of procedure that might involve database updates
- Example:

  ON updating maximum course enrollment
  IF number registered > new max enrollment limit
  THEN deregister students using LIFO policy

6

Trigger Details

- **Activation -** Occurrence of the *event*
- **Consideration -** The point, after activation, when *condition* is evaluated
  - Immediate or deferred (when the transaction requests to commit)
  - *Condition* might refer to both the state before and the state after *event* occurs

7

Trigger Details

- **Execution -** point at which *action* occurs
  - With deferred consideration, execution is also deferred
  - With immediate consideration, execution can occur immediately after consideration or it can be deferred
    - If execution is immediate, execution can occur before, after, or instead of triggering event.
    - Before triggers adapt naturally to maintaining integrity constraints: violation results in rejection of event.

8

Trigger Details

- **Granularity -**
  - Row-level granularity: change of a single row is an event (a single UPDATE statement might result in multiple events)
  - Statement-level granularity: events are statements (a single UPDATE statement that changes multiple rows is a single event).

9

Trigger Details

- **Multiple Triggers**
  - How should multiple triggers activated by a single event be handled?
    - Evaluate one condition at a time and if true immediately execute action or
    - Evaluate all conditions, then execute actions
  - The execution of an action can affect the truth of a subsequently evaluated condition so the choice is significant.

10

Triggers in SQL/3

- **Events**: INSERT, DELETE, or UPDATE statements or changes to individual rows caused by these statements
- **Condition**: Anything allowed in a WHERE clause
- **Action**: An individual SQL statement or a program written in the language of Procedural Stored Modules (PSM) (which can contain embedded SQL statements)

11

Triggers in SQL:1999

- **Consideration**: Immediate
  - Condition can refer to both the state of the affected row or table before *and* after the event occurs
- **Execution**: Immediate - can be before or after the execution of triggering event .
  - Action of before trigger cannot modify the database
- **Granularity**: Both row-level and statement-level granularity

12

Before Trigger Example
(row granularity)

```
CREATE TRIGGER  Max_EnrollCheck
  BEFORE INSERT ON Transcript
      REFERENCING NEW AS N   --row to be added
  FOR EACH ROW
  WHEN
  ((SELECT  COUNT (T.StudId) FROM Transcript T
     WHERE  T.CrsCode = N.CrsCode
              AND T.Semester = N.Semester)
   >=
   (SELECT C.MaxEnroll FROM Course C
     WHERE C.CrsCode = N.CrsCode ))
  ABORT TRANSACTION
```

13

After Trigger Example
(row granularity)

```
CREATE TRIGGER LimitSalaryRaise
  AFTER UPDATE OF Salary ON Employee
  REFERENCING OLD AS O
                      NEW AS N
  FOR EACH ROW
  WHEN (N.Salary - O.Salary > 0.05 * O.Salary)
     UPDATE Employee          -- action
     SET Salary = 1.05 *  O.Salary
     WHERE Id = O.Id
```

Note: The action itself is a triggering event (but in this case a chain reaction is not possible)

14

After Trigger Example
(statement granularity)

```
CREATE TRIGGER RecordNewAverage
  AFTER UPDATE OF Salary ON Employee
  FOR EACH STATEMENT
    INSERT INTO Log
    VALUES  (CURRENT_DATE,
                  SELECT AVG (Salary)
                  FROM  Employee)
```

15