Chapter 1

Syntax and Semantics of Active Databases

1.1 Exercises

1.1. Given the relational database schema

Employee(Name, Salary, Department)
Department(Dept-No, Manager)

define the following active rules in Starburst, Oracle, and DB2:

a. A rule that, whenever a department is deleted from the database, sets to null the value of the Department attribute for those tuples in relation Employee having the number of the deleted department.
b. A rule that, whenever a department is deleted from the database, deletes all employees in the deleted department.
c. A rule that, whenever the salary of an employee exceeds the salary of its manager, sets the salary of the employee to the salary of the manager.
d. A rule that, whenever salaries are updated, if the total of the updated salaries exceeds their total before the updates, then gives all the employees of the 'Research' department a 5% salary cut.

Complete this exercise by writing the same triggers in Chimera on the following object-oriented schema (which is equivalent to the previous relational schema):

```
create object class Employee
attributes Name: string,
           Salary: integer,
           Department: Dept
end;

create object class Dept
attributes Manager: Employee
```

end;

Answer:

- In Starbust:

a. **Example 1.1**

```
CREATE RULE SetNull ON Department
WHEN DELETED
THEN  UPDATE Employee
      SET Department = Null
      WHERE Department IN (SELECT DeptNo
                           FROM DELETED)
```

b. **Example 1.2**

```
CREATE RULE CascDel ON Department
WHEN DELETED
THEN  DELETE FROM Employee
      WHERE Department IN (SELECT DeptNo
                           FROM DELETED)
```

c. **Example 1.3**

```
CREATE RULE CheckSal ON Employee
WHEN INSERTED, UPDATED Salary
THEN  UPDATE Employee X
      SET Salary = (SELECT Salary
                   FROM Employee, Department
                   WHERE X.Department = DeptNo AND
                         Name = Manager)
            WHERE Salary ¿ (SELECT Salary
                   FROM Employee, Department
                   WHERE X.Department = DeptNo AND
                         Name = Manager)
```

d. **Example 1.4**

```
CREATE RULE CutSal ON Employee
WHEN UPDATED Salary
IF    (SELECT SUM(Salary) FROM NewUpdated) ¿
      (SELECT SUM(Salary) FROM OldUpdated)
THEN  UPDATE Employee
      SET Salary = 0.95 * Salary
      WHERE Department = 'Research'
```

- In Oracle:

a. **Example 1.5**

```
CREATE TRIGGER SetNull
AFTER DELETE ON Department
REFERENCING OLD AS DelDept
        UPDATE Employee
```

```
        SET Department = Null
        WHERE Department IN (SELECT DeptNo
                             FROM DelDept);
```

b. **Example 1.6**

```
CREATE TRIGGER CascDel
AFTER DELETE ON Department
REFERENCING OLD AS DelDept
        DELETE FROM Employee
        WHERE Department IN (SELECT DeptNo
                             FROM DelDept);
```

c. **Example 1.7**

```
CREATE TRIGGER CheckSal
AFTER INSERT, UPDATE OF Salary ON Employee
FOR EACH ROW
        IF NEW.Salary ¿ (SELECT Salary FROM Employee, Department
                         WHERE NEW.Department = DeptNo AND
                               Name = Manager)
        THEN
          UPDATE NEW
          SET Salary = (SELECT Salary FROM Employee, Department
                        WHERE NEW.Department = DeptNo AND
                              Name = Manager)
        END;
```

d. **Example 1.8**

```
CREATE TRIGGER CutSal
AFTER UPDATE OF Salary ON Employee
        IF (SELECT SUM(Salary) FROM NEW) ¿
          (SELECT SUM(Salary) FROM OLD)
        THEN
          UPDATE Employee
          SET Salary = 0.95 * Salary
          WHERE Department = 'Research'
        END;
```

- In DB2:

a. **Example 1.9**

```
CREATE TRIGGER SetNull
AFTER DELETE ON Department
REFERENCING OLD_TABLE AS DelDept
FOR EACH STATEMENT
        UPDATE Employee
        SET Department = Null
        WHERE Department IN (SELECT DeptNo
                             FROM DelDept);
```

b. **Example 1.10**

```
CREATE TRIGGER CascDel
```

```
AFTER DELETE ON Department
REFERENCING OLD_TABLE AS DelDept
FOR EACH STATEMENT
      DELETE FROM Employee
      WHERE Department IN (SELECT DeptNo
                           FROM DelDept);
```

c. **Example 1.11**

```
CREATE TRIGGER CheckSal1
AFTER INSERT ON Employee
REFERENCING NEW AS NEmp
FOR EACH ROW
WHEN NEmp.Salary ¿ (SELECT Salary
     FROM Employee, Department
     WHERE NEmp.Department = DeptNo AND
           Name = Manager)
              UPDATE NEmp
              SET Salary = (SELECT Salary
     FROM Employee, Department
     WHERE NEmp.Department = DeptNo AND
           Name = Manager);
```

Example 1.12

```
CREATE TRIGGER CheckSal2
AFTER UPDATE OF Salary ON Employee
REFERENCING NEW AS NEmp
FOR EACH ROW
WHEN NEmp.Salary ¿ (SELECT Salary
     FROM Employee, Department
     WHERE NEmp.Department = DeptNo AND
           Name = Manager)
              UPDATE NEmp
              SET Salary = (SELECT Salary
     FROM Employee, Department
     WHERE NEmp.Department = DeptNo AND
           Name = Manager);
```

d. **Example 1.13**

```
CREATE TRIGGER CutSal
AFTER UPDATE OF Salary ON Employee
REFERENCING NEW_TABLE AS NEmp, OLD_TABLE AS OEmp
FOR EACH STATEMENT
WHEN (SELECT SUM(Salary) FROM NEmp) ¿
      (SELECT SUM(Salary) FROM OEmp)
      UPDATE Employee
      SET Salary = 0.95 * Salary
      WHERE Department = 'Research'
```

- In Chimera:

a. **Example 1.14**

```
define trigger SetNull
   for Dept
   events      delete
   condition   occurred(delete,X), Employee(X), E.Department = X
   action      modify(Employee.Department,E,null)
end;
```

b. **Example 1.15**

```
define trigger CascDel
   for Dept
   events      delete
   condition   occurred(delete,X), Employee(X), E.Department = X
   action      delete(Employee,E)
end;
```

c. **Example 1.16**

```
define trigger CheckSal
   for Employee
   events      create, modify(Salary)
   condition   Self.Salary ¿ Self.Department.Manager.Salary
   action      modify(Employee.Salary,Self,Self.Department.Manager.Salary)
end;
```

d. **Example 1.17**

```
define trigger CutSal
   for Employee
   events      modify(Salary)
   condition   sum(X.Salary where Employee(X)) ¿
                sum(old(X.Salary) where Employee(X)),
                Employee(E), E.department = 'Research'
   action      modify(Employee.Salary,E,E.Salary*0.95)
end;
```

1.2. Referring to the relational schema above, define in Starburst or Chimera a deferred trigger R_1 that, whenever an employee who is a manager is deleted, also deletes all employees in the department managed by the deleted employee, along with the department itself.

Then define another deferred trigger R_2 that, whenever salaries are updated, checks the average of the updated salaries; if it exceeds $50,000$, then it deletes all employees whose salary was updated and now exceeds $80,000$.

Consider next a database state containing six employees: Jane, Mary, Bill, Jim, Sam, and Sue, with the following management structure:

- Jane manages Mary and Jim

- Mary manages Bill
- Jim manages Sam and Sue

Now suppose that a user transaction deletes employee Jane and updates salaries in a way such that the average updated salaries exceeds 50,000 and Mary's updated salary exceeds 80,000. Describe the trigger processing started at the end of this transaction.

Answer:

Example 1.18

```
CREATE RULE R1 ON Employee
WHEN DELETED
THEN  DELETE FROM Employee
        WHERE Department IN (SELECT DeptNo
                              FROM Department
                              WHERE Manager IN (SELECT Name
          DELETE FROM Department                FROM DELETED))
        WHERE Manager IN (SELECT Name
                           FROM DELETED)
```

Example 1.19

```
CREATE RULE R2 ON Employee
WHEN UPDATED Salary
IF (SELECT AVG(Salary) FROM NEWUPDATED) ¿ 50,000
THEN  DELETE FROM Employee
        WHERE Name IN (SELECT Name
                        FROM NEWUPDATED)
              AND Salary ¿ 80,000
```

Both rules R_1 and R_2 are triggered by the transition. Rule R_1 is triggered with respect to the set $\{Jane\}$ of deleted employees. Suppose that rule R_2 has priority over rule R_1:

- R_2 is executed and Mary is deleted (R_2 is not triggered again);
- R_1 is triggered with respect to the set $\{Jane, Mary\}$ of deleted employees, thus Bill and Jim are deleted;
- R_1 is triggered with respect to the set $\{Bill, Jim\}$ of deleted employees, thus Sam and Sue are deleted;
- R_1 is triggered with respect to the set $\{Sam, Sue\}$ of deleted employees, no more employee are deleted and reactive processing stops.

1.3. Given the Chimera class Employee, with an attribute Salary and a class RichEmployee with the same schema, define in Chimera a set of triggers ensuring that in any database state the set of instances of the class RichEmployee coincides with the set of instances of the class Employee whose value for attribute Salary is greater than 50,000.

Answer:

Example 1.20

```
define trigger MakeRich for Employee
   events create, modify(Salary)
   condition Employee(E), occurred(create, modify(Salary), E),
        E.Salary ¿ 50,000, not E in RichEmployee
   actions specialize(Employee, RichEmployee, E)
```

Example 1.21

```
define trigger MakePoor for Employee
   events create, modify(Salary)
   condition RichEmployee(E), occurred(create, modify(salary), E),
        E.Salary ¡= 50,000
   actions generalize(RichEmployee, Employee, E)
```

Chapter 2

Applications of Active Databases

2.1 Exercises

2.1. Given the relational database schema

PhDStudent(Email, Name, Area, Supervisor)
Prof(Email, Name, Area)
Course(Title, Prof)
CoursesTaken(PhDSt, Course)

Derive the triggers for maintaining the following integrity constraints:

a. Each PhD student must work in the same area as their supervisor.

b. Each PhD student must take at least one course.

c. Each PhD student must take the courses taught by their supervisor.

Answer:

a. **Example 2.1**

```
define rule SameArea
   when inserted(PhDStudent), updated(PhDStudent.Supervisor)
   if     EXISTS S: (SELECT * FROM PhDStudent
                     WHERE Area ¡¿ (SELECT Area FROM Prof
                                    WHERE Email = S.Supervisor))
   then  UPDATE PhDStudent
         SET Area = (SELECT Area FROM Prof
                      WHERE Email = S.Supervisor)
         WHERE Email = S.Email
```

b. **Example 2.2**

```
define rule ACourse
   when inserted(PhDStudent), deleted(CoursesTaken)
   updated(CoursesTaken.PhDSt)
   if     EXISTS S: (SELECT * FROM PhDStudent
```

```
                    WHERE NOT EXISTS (SELECT *
                                      FROM CoursesTaken
                                      WHERE PhDSt = S.Email))
        then  rollback
```

c. **Example 2.3**

```
    define rule CoursesSup
      when inserted(PhDStudent), inserted(Course), deleted(CoursesTaken)
      updated(PhDStudent.Supervisor), updated(CoursesTaken.PhDSt)
      updated(CoursesTaken.Course), updated(CoursesTaken.PhDSt)
      if    EXISTS S: (SELECT * FROM PhDStudent
                       WHERE (SELECT Title FROM Course
                              WHERE Prof = S.Supervisor)
                              NOT IN
                              (SELECT Course FROM CoursesTaken
                              WHERE PhDSt = S.Email)
      then  INSERT INTO CoursesTaken
            SELECT Email, Title
            FROM PhStudent, Course
            WHERE Prof = Supervisor
```

2.2. Given the relational database schema

Employee(<u>Name</u>, DeptCode)
Department(<u>DeptCode</u>, DeptName, City, Budget)

and the view MilanEmp, defined as

```
SELECT Name
FROM Employee, Department
WHERE Employee.DeptCode = Department.DeptCode
    AND Department.City = 'Milano'
```

a. Define the triggers for incrementally maintaining the view
b. Define the triggers for handling updates (insertions and deletions) through the view

Answer:

a. **Example 2.4**

```
    define rule InsInMEmp
      when inserted(Employee), updated(Employee.DeptCode)
            updated(Department.City)
      if    EXISTS E: (SELECT * FROM Employee
                       WHERE (SELECT City FROM Department
                              WHERE DeptCode = E.DeptCode)
```

```
                    = 'Milano')
    then  INSERT INTO MilanEmp
          SELECT Name FROM Employee
          WHERE Name = E.Name
```

Example 2.5

```
define rule DelFromMEmp
  when deleted(Employee)
  if    EXISTS E: (SELECT * FROM DELETED
                   WHERE (SELECT City FROM Department
                          WHERE DeptCode = E.DeptCode)
                   = 'Milano')
  then  DELETE FROM MilanEmp
        WHERE Name = E.Name
```

Example 2.6

```
define rule DelFromMEmp2
  when updated(Employee.Dept), updated(Department.City)
  if    EXISTS E: (SELECT * FROM MilanEmp
                   WHERE (SELECT City FROM Department, Employee
                          WHERE DeptCode = E.DeptCode AND Name = E.Name)
                   ¡¿ 'Milano')
  then  DELETE FROM MilanEmp
        WHERE Name = E.Name
```

b. **Example 2.7**

```
define rule InsThroughMEmp
  when inserted(MilanEmp)
  if    EXISTS E: (SELECT * FROM INSERTED
  then  INSERT INTO Employee
        VALUES(E.Name,Null)
```

Example 2.8

```
define rule DelThroughMEmp
  when deleted(MilanEmp)
  if    EXISTS E: (SELECT * FROM DELETED
  then  DELETE FROM Employee
        WHERE Name = E.Name
```

2.3. Consider the set of rules specified in Section **??**. Draw a simple energy management network consisting of a few nodes and connections, then populate the corresponding classes in Chimera. Think of simple update operations and generate the corresponding execution trace, listing triggers that are subsequently considered and executed; show the final quiescent state obtained at the end of rule processing. Think of one update operation that causes the execution of the abort trigger R8.

Answer:

****