```
                                level    CHAR(20),
                                age      INTEGER,
                                PRIMARY KEY (snum),
                                CHECK ( (SELECT COUNT (S.snum)
                                         FROM Student S
                                         WHERE S.major = 'CS') > 0 ))
```

(o) Create an assertion as follows:

```
CREATE ASSERTION NotSameRoom
CHECK (   (SELECT   COUNT (*)
           FROM     Faculty F1, Faculty F2, Class C1, Class C2
           WHERE    F1.fid = C1.fid
           AND      F2.fid = C2.fid
           AND      C1.room = C2.room
           AND      F1.deptid ≠ F2.deptid) = 0)
```

Exercise 5.9 Discuss the strengths and weaknesses of the trigger mechanism. Contrast triggers with other integrity constraints supported by SQL.

Answer 5.9 A trigger is a procedure that is automatically invoked in response to a specified change to the database. The advantages of the trigger mechanism include the ability to perform an action based on the result of a query condition. The set of actions that can be taken is a superset of the actions that integrity constraints can take (i.e. report an error). Actions can include invoking new update, delete, or insert queries, perform data definition statements to create new tables or views, or alter security policies. Triggers can also be executed before or after a change is made to the database (that is, use old or new data).

There are also disadvantages to triggers. These include the added complexity when trying to match database modifications to trigger events. Also, integrity constraints are incorporated into database performance optimization; it is more difficult for a database to perform automatic optimization with triggers. If database consistency is the primary goal, then integrity constraints offer the same power as triggers. Integrity constraints are often easier to understand than triggers.

Exercise 5.10 Consider the following relational schema. An employee can work in more than one department; the *pct_time* field of the Works relation shows the percentage of time that a given employee works in a given department.

Emp(*eid:* `integer`, *ename:* `string`, *age:* `integer`, *salary:* `real`)
Works(*eid:* `integer`, *did:* `integer`, *pct_time:* `integer`)
Dept(*did:* `integer`, *budget:* `real`, *managerid:* `integer`)

Write SQL-92 integrity constraints (domain, key, foreign key, or CHECK constraints; or assertions) or SQL:1999 triggers to ensure each of the following requirements, considered independently.

1. Employees must make a minimum salary of $1000.
2. Every manager must be also be an employee.
3. The total percentage of all appointments for an employee must be under 100%.
4. A manager must always have a higher salary than any employee that he or she manages.
5. Whenever an employee is given a raise, the manager's salary must be increased to be at least as much.
6. Whenever an employee is given a raise, the manager's salary must be increased to be at least as much. Further, whenever an employee is given a raise, the department's budget must be increased to be greater than the sum of salaries of all employees in the department.

Answer 5.10 The answer to each question is given below.

1. This constraint can be added by modifying the Emp table:

```
CREATE TABLE Emp (  eid        INTEGER,
                    ename      CHAR(20),
                    age        INTEGER,
                    salary     REAL,
                    PRIMARY KEY (eid),
                    CHECK      ( salary > 1000))
```

2. Create an assertion as follows:

```
CREATE ASSERTION ManagerIsEmployee
CHECK ( ( SELECT    COUNT (*)
          FROM      Dept D
          WHERE     D.managerid NOT IN
                    (SELECT * FROM Emp))
          = 0)
```

3. This constraint can be added by modifying the Works table:

```
CREATE TABLE Works ( eid        INTEGER,
                     did        INTEGER,
                     pct_time   INTEGER,
                     PRIMARY KEY (eid, did),
                     CHECK (    (SELECT   COUNT (W.eid)
                                 FROM     Works W
                                 GROUP BY W.eid
                                 HAVING   Sum(pct_time) > 100) = 0))
```

4. Create an assertion as follows:

```
CREATE ASSERTION ManagerHigherSalary
CHECK (   SELECT    E.eid
          FROM      Emp E, Emp M, Works W, Dept D
          WHERE     E.eid = W.eid
          AND       W.did = D.did
          AND       D.managerid = M.eid
          AND       E.salary > M.salary)
```

5. This constraint can be satisfied by creating a trigger that increases a manager's salary to be equal to the employee who received the raise, if the manager's salary is less than the employee's new salary.

```
CREATE TRIGGER GiveRaise AFTER UPDATE ON Emp
         WHEN     old.salary < new.salary
         FOR EACH ROW
         BEGIN
                  UPDATE   Emp M
                  SET      M.Salary = new.salary
                  WHERE    M.salary < new.salary
                  AND      M.eid IN (SELECT   D.mangerid
                                     FROM     Emp E, Works W, Dept D
                                     WHERE    E.eid = new.eid
                                     AND      E.eid = W.eid
                                     AND      W.did = D.did);
         END
```

6. This constraint can be satisfied by extending the trigger in the previous question. We must add an `UPDATE` command to increase the budget by the amount of the raise if the budget is less than the sum of all employee salaries.

```
CREATE TRIGGER GiveRaise AFTER UPDATE ON Emp
```

```
WHEN    old.salary < new.salary
FOR EACH ROW
DECLARE
        _raise REAL;
BEGIN
        _raise := new.salary - old.salary;
        UPDATE  Emp M
        SET     M.Salary = new.salary
        WHERE   M.salary < new.salary
        AND     M.eid IN (SELECT  D.mangerid
                          FROM    Emp E, Works W, Dept D
                          WHERE   E.eid = new.eid
                          AND     E.eid = W.eid
                          AND     W.did = D.did);
        UPDATE  Dept D
        SET     D.budget = D.budget + _raise
        WHERE   D.did IN (SELECT  W.did
                          FROM    Emp E, Works W, Dept D
                          WHERE   E.eid = new.eid
                          AND     E.eid = W.eid
                          AND     D.did = W.did
                          AND     D.budget <
                          (SELECT  Sum(E2.salary)
                          FROM    Emp E2, Works W2
                          WHERE   E2.eid = W2.eid
                          AND     W2.dept = D.did));
        END
```