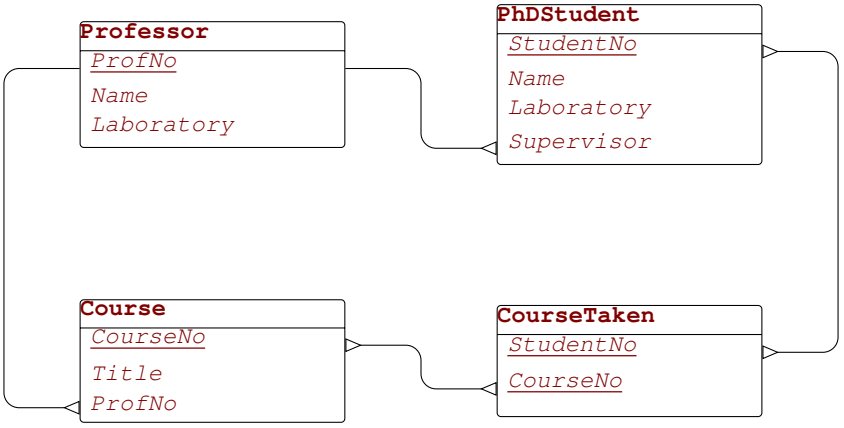
INFO-H-415 - Advanced Databases

# Session 1 - Active Databases (1/3)

Consider the following database schema:




Define in SQL Server a set of triggers that ensure the following constraints:

**Exercise 1.** A PhD student must work in the same laboratory as his/her supervisor.

**Exercise 2.** A PhD student must take at least one course.

**Exercise 3.** A PhD student must take all courses taught by his/her supervisor.

# Details of the database for the exercises

## Table creation script

```
create table Professor (
  ProfNo char(5) not null,
  Name varchar(25) not null,
  Laboratory varchar(25) not null,
  constraint PK_Professor primary key (ProfNo)
)
create table PhDStudent (
  StudentNo char(9) not null,
  Name varchar(25) not null,
  Laboratory varchar(25),
  Supervisor char(5),
  constraint PK_PhDStudent primary key (StudentNo),
  constraint FK_PhDStudent_Professor
     foreign key (Supervisor) references Professor(ProfNo)
)
create table Course (
  CourseNo char(7) not null,
  Title varchar(30) not null,
  ProfNo char(5),
  constraint PK_Course primary key (CourseNo),
  constraint FK_Course_Professor
     foreign key (ProfNo) references Professor(ProfNo)
)
create table CourseTaken (
  StudentNo char(9) not null,
  CourseNo char(7) not null,
  constraint PK_CourseTaken primary key (StudentNo,CourseNo),
  constraint FK_CourseTaken_Student
     foreign key (StudentNo) references PhDStudent(StudentNo),
  constraint FK_CourseTaken_Course
     foreign key (CourseNo) references Course(CourseNo)
)
```

## Initial data in the tables

Professor

| ProfNo | Name | Laboratory |
|---|---|---|
| 12345 | John B. Smith | Databases |
| 33344 | Franklin T. Wong | Databases |
| 99988 | Alicia J. Zelaya | Networks |
| 98765 | Jennifer S. Wallace | Web Technologies |
| 66688 | Ramesh K. Narayan | Web Technologies |

PhDStudent

| StudentNo | Name | Laboratory | Supervisor |
|---|---|---|---|
| 453453453 | Joyce A. English | Databases | 12345 |
| 987987987 | Ahmad V. Jabbar | Databases | 33344 |
| 888665555 | James A. Borg | Web Technologies | 66688 |

Course

| CourseNo | Title | ProfNo |
|---|---|---|
| INFO364 | Introduction to Databases | 12345 |
| INFO365 | Advanced Databases | 33344 |
| INFO378 | XML | 66688 |
| INFO379 | Web Services | 66688 |

CourseTaken

| StudentNo | CourseNo |
|---|---|
| 453453453 | INFO364 |
| 453453453 | INFO365 |
| 987987987 | INFO364 |
| 987987987 | INFO365 |
| 888665555 | INFO378 |

INFO-H-415 - Advanced Databases

# Solutions for Session 1 - Active Databases (1/3)

## ➤ Solution to Exercise 1

### Constraint

*"A PhD student must work in the same laboratory as his/her supervisor."*

### **Events** that may violate the constraint

a) Insert into **PhDStudent**

b) Update of *Laboratory* or *Supervisor* in **PhDStudent**

c) Update of *Laboratory* in **Professor**

d) Delete from **Professor**

### Actions

For each event, we have to choose whether to *abort* the transaction (by rolling back and generating a message) or to *repair* it (applying modifications that ensure that the constraint is satisfied). Note that for the latter, there are often several ways to ensure constraint satisfaction; choosing the way of applying changes that enforce the database's consistency therefore depends on the actual implementation context. In the following, we propose one of the often many possible ways of repairing the violated constraints.
For the present exercises, when applicable, a trigger creation statement for both scenarios – *aborting* or *repairing* – is proposed.

⚠ **CAUTION** Beware, however, that, for each constraint, only one single rule should be active at any given time. Thus, ensure that the first trigger does not exist or has been deactivated or deleted before testing the second one, and vice-versa.

#### Events a) and b)

#### 1. *Aborting* the transaction

```
create trigger StudSameLabAsSuperv_PhDStud_InsUpd_Abort
on PhDStudent
after insert, update
as
if exists (
    select * from Inserted I, Professor P
     where P.ProfNo = I.Supervisor
       and P.Laboratory <> I.Laboratory )
begin
    raiserror ('Constraint Violation:
             A PhD student must work in the same
     laboratory as his/her supervisor', 1, 1)
    rollback
end
```

**2. *Repairing* the transaction**

```
create trigger StudSameLabAsSuperv_PhDStud_InsUpd_Repair
on PhDStudent
after insert, update
as
begin
    update PhDStudent
       set Laboratory = (
                select P.Laboratory
                  from Professor P
                 where P.ProfNo = Supervisor )
     where StudentNo in (
                select I.StudentNo
                  from Inserted I )
end
```

In this case, repairing by making the necessary updates in **PhDStudent** will probably be the better choice, as the value that *Laboratory* should take is univocally determined by the consistency rule.

### Event c)

**1. *Aborting* the transaction**

```
create trigger StudSameLabAsSuperv_Prof_Upd_Abort
on Professor
after update
as
if exists (
    select * from Inserted I, PhDStudent S
     where I.ProfNo = S.Supervisor
       and I.Laboratory <> S.Laboratory )
begin
    raiserror ('Constraint Violation:
             A PhD student must work in the same
     laboratory as his/her supervisor', 1, 1)
    rollback
end
```

**2. *Repairing* the violated constraints**

```
create trigger StudSameLabAsSuperv_Prof_Upd_Repair
on Professor
after update
as
begin
    update PhDStudent
       set Laboratory = (
                select I.Laboratory
                  from Inserted I
                 where Supervisor = I.ProfNo )
     where Supervisor in (
                select I2.ProfNo
                  from Inserted I2 )
end
```

Again, *repairing* the violated constraints should be preferred in this case too. The assumption is that, when a professor leaves her lab for another one, her PhD students are following.

### Event d)

In this case, there are several possibilities.

- The professor is deleted and the attributes *Laboratory* and *Supervisor* of the PhD students who worked for the deleted professor are set to `null`.
- The transaction is rolled back, preventing a professor to be deleted when there are PhD students associated to her. This is taken care of by the referential integrity.
- The professor is deleted and all PhD students associated with him are also deleted. This is taken care of by the referential integrity with the option `on update cascade`.

We here provide the trigger corresponding to the first of these cases.

```
alter table PhDStudent
drop constraint FK_PhDStudent_Professor
```

```
create trigger StudSameLabAsSuperv_Prof_Del_Repair
on Professor
after delete
as
begin
    update PhDStudent
       set Laboratory = null,
           Supervisor = null
     where Supervisor in (
                select ProfNo
                  from Deleted )
end
```

⚠ **CAUTION** As SQL Server does not implement the option `on delete set null` for the referential integrity, it is necessary to drop the foreign key constraint in the table **PhDStudent**.

# ➤ Solution to Exercise 2

## Constraint

*"A PhD student must take at least one course."*

Contrarily to Exercise 1, this constraint is a "negative" one, in the sense that it prevents that there does not exist any PhD student that does not take any course, but it provides no information about a way of automatically determining the one course that should be taken "by default", should no other be provided. Consequently, it will not be possible without any additional assumption, to *repair* violated constraints. All violating events will thus simply result in aborting the transaction.

## **Events** that may violate the constraint

a) Insert into **PhDStudent**

b) Update of *StudentNo* in **CourseTaken**

c) Delete from **CourseTaken**

d) Delete from **Course**

### Actions

#### Event a)

```
create trigger PhDStudMinOneCourse_PhDStud_Ins_Abort
on PhDStudent
after insert
as
if exists (
   select * from Inserted I
    where not exists (
              select *
                from CourseTaken
               where StudentNo = I.StudentNo ) )
begin
   raiserror ('Constraint Violation:
   A PhD student must take at least one course', 1, 1)
   rollback
end
```

⚠ **CAUTION** This does not work in SQL Server, since a trigger is executed immediately after the triggering instruction. Thus, embedding several inserts (into **PhDStudent** and **CourseTaken**) into one transaction would not help.
Practically, thus, and without any further assumption, it will *not* be possible to ensure that this constraint verified, as the aborting trigger would prevent any insertion into the table **PhDStudent**.

#### Events b) and c)

```
create trigger PhDStudMinOneCourse_PhDStud_Ins_Abort
on CourseTaken
after update, delete
as
if exists (
    select * from Deleted D
     where D.StudentNo not in (
               select StudentNo
                 from CourseTaken ) )
begin
    raiserror ('Constraint Violation:
    A PhD student must take at least one course', 1, 1)
    rollback
end
```

#### Event d)

Removing an entry from **Course** could indirectly affect the number of courses taken by one or several PhD students. This case, however, should be handled with the `on update cascade` option of the referential integrity constraint on the *CourseNo* field of **CourseTaken**.

## ➤ Solution to Exercise 3

### Constraint

*"A PhD student must take all courses taught by his/her supervisor."*

### **Events** that may violate the constraint

a) Insert into **PhDStudent**

b) Update of *Supervisor* in **PhDStudent**

c) Insert into **Course**

d) Update of *ProfNo* in **Course**

e) Update of *StudentNo* or *CourseNo* in **CourseTaken**

f) Delete from **CourseTaken**

### Actions

**Events a) and b)**

**1. *Aborting* the transaction**

```
create trigger StudAllCoursesOfSuperv_Stud_InsUpd_Abort
on PhDStudent
after insert, update
as
if exists (
   select * from Inserted I
    where exists (
              select *
                from Course C
               where C.ProfNo = I.Supervisor
                 and C.CourseNo not in (
                                 select T.CourseNo
                                   from CourseTaken T
                                  where T.StudentNo = I.StudentNo  ) ) )
begin
   raiserror ('Constraint Violation:
       A PhD student must take all the courses
       given by his supervisor', 1, 1)
   rollback
end
```

**2. *Repairing* the transaction**

```
create trigger StudAllCoursesOfSuperv_Stud_InsUpd_Repair
on PhDStudent
after insert, update
as
begin
    insert into CourseTaken (StudentNo, CourseNo)
    select I.StudentNo, C.CourseNo
      from Inserted I,
           Professor P,
           Course C
     where I.Supervisor = P.ProfNo
       and C.ProfNo = P.ProfNo
       and C.CourseNo not in (
                 select T.CourseNo
                   from CoursTaken T
                  where T.StudentNo = I.StudentNo )
end
```

The rules implemented by this trigger can be challenged, for instance, with the following change of supervisor for the student named Joyce. The lab she belongs to will automatically be changed to “Web Technologies” by the trigger.

```
begin transaction
    update PhDStudent
       set Supervisor = 66688
     where StudentNo = 453453453
commit transaction
```

**Events c) and d)**

Aborting the transaction, particularly in SQL Server (where triggers are executed immediately after the triggering instruction), would not work (well).
The repairing rule being implicitly defined, or at least suggested, by the constraint (namely to automatically enrol the student in the added course), it will be the method of choice for this case.
Note that the `update` case, where a Professor would abandon a course, has not to be handled here explicitely, since letting a Student take courses from Professors that are not his supervisor is not forbidden.

```
create trigger StudAllCoursesOfSuperv_Course_InsUpd_Repair
on Course
after insert, update
as
begin
    insert into CourseTaken (StudentNo, CourseNo)
    select S.StudentNo, I.CourseNo
      from Inserted I,
           Professor P,
           PhDStudent S
     where C.ProfNo = P.ProfNo
       and S.Supervisor = P.ProfNo
       and I.CourseNo not in (
                select T.CourseNo
                  from CourseTaken T
                 where T.StudentNo = C.StudentNo )
end
```

**Events e) and f)**

```
create trigger StudAllCoursesOfSuperv_CourseTaken_UpdDel_Abort
on CourseTaken
after update, delete
as
if exists (
    select *
      from Deleted D,
           Course C,
           PhDStudent S
     where D.CourseNo = C.CourseNo
       and C.ProfNo = S.Supervisor
       and D.StudentNo = S.StudentNo )
begin
   raiserror ('Constraint Violation:
       A PhD student must take all the courses
       given by his supervisor', 1, 1)
end
```