

# INFO-H-415 - Advanced Databases

## Session 1 Active Databases

Université libre de Bruxelles  
École polytechnique de Bruxelles

# Practicalities

12 exercise sessions

(on machines ; room indicated in Gehol

<http://gehol.ulb.ac.be>)

1 project

(starting at week 8)

# 12 exercise sessions

- Sessions 1 – 3 :        **Active** databases
- Sessions 4 – 6 :        **Temporal** databases
- Sessions 7 – 9 :        **Object** databases
- Sessions 10–12 :        **Spatial** databases

# Practicalities

## Course's Wiki

<http://cs.ulb.ac.be/public/teaching/infoh415>

## Teaching Assistant

**Stefan Eppe**

stefan.eppe@ulb.ac.be

Office : UB4.131

# Evaluation

- 25% for the **project**,
- 75% for the **written examination**

Do you have questions ?

Active Databases

# SQL Server Triggers

# Database triggers

A database trigger is **procedural code** that is automatically executed in response to certain **events** on a particular table or view in a database.

The trigger is mostly used for maintaining the **integrity** of the information on the database.

# SQL Server triggers

In SQL Server, triggers are executed directly after an **instruction** (i.e. not after each row or each transaction).

# SQL Server trigger types

- **AFTER triggers** are executed after the instruction takes place
- **INSTEAD OF triggers** do not execute the triggering instruction, but executes custom code in place of it

# SQL Server triggers

## Syntax

```
create trigger <name>  
on <table>  
{after|instead of} <list of events>  
as  
<transact-SQL-statements>
```

Possible events : insert, delete, update

# SQL Server triggers

Inside the `<transact-SQL-statements>`, special tables allow accessing the *newly created* and the *deleted* rows.

## Special tables

- **Inserted** : new or updated rows of the triggering transaction
- **Deleted** : deleted rows (or old state for updates) of the triggering transaction

Note that, since the trigger is executed at instruction level, these tables can contain many rows.

# Two possible actions

When a constraint violation is detected, two types of actions are possible :

## Abort

The transaction is cancelled with a `rollback` statement and an error is raised.

## Repair

An `update` statement modifies the database to make it consistent with the integrity constraints.

# Example of a trigger

Consider two relations :

- **Employee** (Name, Salary, Department)  
*with* Department *referencing* **Department**.DeptNo
- **Department** (DeptNo, Manager)  
*with* Manager *referencing* **Employee**.Name

We want to ensure that *the salary of an employee cannot be greater than that of his manager.*

*What are the events that could bring this rule to be violated?*

# Example of a trigger

- **Employee** (Name, Salary, Department)
- **Department** (DeptNo, Manager)

We want to ensure that *the salary of an employee cannot be greater than that of his manager.*

Constraint violating events :

- When adding an employee
- When modifying an employee's salary
- When modifying an employee's department
- When modifying department's manager

## Example of an aborting *after insert* trigger

```
create trigger salaryEmployee
on Employee
after insert
as
if exists(
    select *
    from Inserted I,
         Department D,
         Employee Mgr
    where I.DeptNo = D.DeptNo
         and D.Manager = Mgr.Name
         and Mgr.Salary < I.Salary )
begin
    raiserror 13000 'The salary of an employee
    cannot be greater than that of his manager'
    rollback
end
```

Active Databases

# Exercises

# Connecting to the database environment

- Log on to the computer with your *netid*
- Open *SQL Server Management Studio*
- Connect to the server “`CS-MSSQL`”  
(using Windows authentication)

# Loading the data set

Available on the labs web page :

<http://cs.ulb.ac.be/public/teaching/infoh415/tp>

## Set-up

- Create a “`infoh415-<your-netid>-PhD`” database (drop it if it already exists)
- Open and run `phd_createtable.sql`
- Open and run `phd_dbload.sql`  
**Caution** : Select the right database before running these scripts !

# Practical steps for the exercises

We suppose that the database is initially *consistent*.

## Steps

- ① Determine when a constraint can be violated.
- ② Then, decide on an action to be taken : *abort* or *repair*
- ③ Write the trigger
- ④ Test the trigger, by editing the data in a way that violates the constraint