INFO-H-415 - Advanced Databases

Session 1
Active Databases

Université libre de Bruxelles
École polytechnique de Bruxelles

Practicalities

Course's Wiki

`http://cs.ulb.ac.be/public/teaching/infoh415`

Teaching Assistant

Dejaegere Gilles
`Gilles.Dejaegere@ulb.ac.be`

12 exercise/QA sessions

- Sessions 1 – 3 : **Active** databases
- Sessions 4 – 6 : **Temporal** databases
- Sessions 7 – 9 : **Graph** databases (?)
- Sessions 10–12 : **Spatial** databases

- Some on site, some online (Q&A). Room indicated in `https://cloud.timeedit.net/be_ulb/web/`

Evaluation

- 25% for the **project**,
  - managed by Prof. Zimányi only !
- 75% for the **written examination**

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 transation).

Employee

SSN	Lab	Salary
6789	1	30 000
5555	2	40 000
4321	1	43 000
7777	4	25 000

```
UPDATE Employee
SET Salary = 0
WHERE Lab = 1;
```

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.

SQL Server triggers

Employee

SSN	Lab	Salary
6789	1	30 000
5555	2	40 000
4321	1	43 000
7777	4	25 000

```sql
UPDATE Employee
SET Salary = 0
WHERE Lab = 1;
```

Inserted

SSN	Lab	Salary
6789	1	0
4321	1	0

Deleted

SSN	Lab	Salary
6789	1	30 000
4321	1	43 000

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`** `(`<u>`Name`</u>`, Salary, Department)`
- **`Department`** `(`<u>`DeptNo`</u>`, 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 Emp-insertion-abort
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 ('Constraint Violation:
                The salary of an employee
                cannot be greater than
                that of his manager', 1, 1)
    rollback
end
```

Active Databases

Exercises

Training on your own machine :

- Download an IDE :
  - SQL Server Management Studio
  - Azure studio
- Download SQL Server Express

Connecting to the database environment from the computer rooms

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

Loading the data set

Available on the labs web page :

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

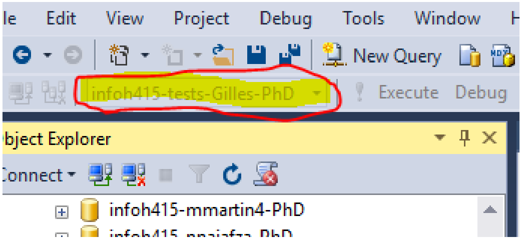
Set-up

- Create a "`infoh415-<`*your-netid*`>-PhD`" database
  (drop it if it already exists)
- Open and run `createDB.sql`
- Open and run `loadDB.sql`
  Caution : Select the right database before running these scripts !
  (see next slide)

Select the right database

Select the database **you created** either :

- using the client


le Edit View Project Debug Tools Window
New Query
infoh415-tests-Gilles-PhD
Execute Debug
bject Explorer
Connect
infoh415-mmartin4-PhD

- by starting your script by :

```
use database_name
```

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