# INFO-H-415 - Advanced Databases

## Sessions 2 & 3
## Active Databases

Université libre de Bruxelles
École polytechnique de Bruxelles

Active Databases

# SQL Server Triggers

## SQL Server triggers

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

### 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.

Active Databases

# SQL Server Constraints

## Types of constraints

`CHECK`

`FOREIGN KEY`

`UNIQUE`

## `CHECK` constraints

`CHECK` is used to set a constraint on a single row.

### Example

*"The salary of an employee must be grater than 1000€."*

**`Employee`**`( Name, Salary, Dept )`

```
alter table Employee
add constraint CK_EmployeeSalary1000
check( Salary >= 1000 )
```

## `CHECK` constraints

### Where can a constraint be used ?

**`Employee`**`( ID, Name, Salary, Dept )`

*"The salary of an employee with the smallest ID must be the highest."*

*"The salary of an employee with ID smaller than 3000 must be smaller than 1000€."*

| | | | |
|---|---|---|---|
| 700 | John | 1050 | 4 |

## `FOREIGN KEY` constraints

Adds a foreign key.

### Example

**`Employee`**`( Name, Salary, Dept )`

`Dept` *references* `Department.DeptNo`

**`Department`**`( DeptNo, Manager )`

```
alter table Employee
add constraint FK_Employee_Dep
foreign key( Dept )
    references Department( DeptNo )
```

## UNIQUE constraints

Used to set a uniqueness constraint on a (set of) attributes, for instance to be allowed to define a foreign key on non-primary keys.

### Syntax

```
alter table <t_name>
add constraint <c_name>
unique( <field_list> )
```

## `UNIQUE` constraints

Suppose “DeptNo” is neither a key neither unique.

**`Department`**`( DeptNo, DeptName )`

| | |
|---|---|
| 1 | Physics |
| 1 | Computer Science |

**`Employee`**`( Name, Salary, Dept )`

`Dept` *references* `Department.DeptNo`

| | | |
|---|---|---|
| John | 1050 | 1 |

## Date-related functions

**`getdate`**`()`

– Returns the current date.

**`dateadd`**`( interval, n, date )`

– `interval` : year, month, day, ...

– Returns the date (date + (n*interval))

**`datediff`**`( interval, start, end )`

– Returns the number of intervals between start and end

Active Databases

# Exercises

## Connecting to the database environment

- Start Microsoft Windows
- Open a session with your *netid*
- Launch *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>*`-`**`Active`**" database
  (drop it if it already exists)
- Open and run **`activeSqlserver`**`_createtable.sql`
- Open and run **`activeSqlserver`**`_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. Decide on an action to be taken : *abort* or *repair*
3. Decide which approach to use (trigger, `CHECK`, `FOREIGN KEY`, `UNIQUE`)
4. Write the trigger or constraint
5. Test the trigger/constraint, by editing the data in a way that violates the constraint