HOGESCHOOL ROTTERDAM / CMI

Advanced Databases (Development 5)

INFDEV23-5

Number of study points:	4 ects
Course owners:	Francesco Di Giacomo

Modulebeschrijving

Module name: (Development 5)	Advanced Databases
Module code:	INFDEV23-5
Study points and hours of effort for full-time students:	This module gives 4 ects, in correspondance with 84 hours: • 16 hours of frontal lectures. • the rest is self-study for the theory and practicum.
Examination:	Written examination
Course structure:	Lectures, self-study, exercises
Prerequisite knowledge:	Object-oriented programming, database design and implementation, declarative programming in SQL.
Learning tools:	• Book: Database management systems (3rd edition); authors Ramakrishnan and Gehrke • Presentations (in pdf): found on N@tschool and on the GitHub repository `https://github.com/hogeschool/Development-5(part-time)` • Assignments, to be done at home (pdf): found on N@tschool and on the GitHub repository `https://github.com/hogeschool/Development-5(part-time)`
Connected to competences:	• Analysis, design, and realisation of software at level 2
Learning objectives:	At the end of the course, the student can: • **realise** a normalized relational database and implement an application to execute operations on it `RDBMS,NORM` • **describe** the differences between relational and non-relational databases `NONREL` • **use** the map-reduce paradigm to run queries. `NONREL` • **describe** models of concurrency and transactions in a modern DBMS `TRANS`

Content:	• relevant concepts in relational databases normalization • fundamental properties of DBMS's: atomicity, consistency, isolation, and durability (ACID) • concurrency and transactions • Map-reduce paradigm. • ACID properties. • Graph databases.
Course owners:	Francesco Di Giacomo
Date:	3 september 2018

1 General description

Databases are ubiquitous within the field of ICT. Many business needs are centered around the gathering, elaboration, etc. of large amounts of data. This data is crucially connected to real-world operations and thus its constant availability and timely elaboration is of unmissable importance.

This course covers advanced aspects of data processing and elaboration within the different sets of tradeoffs of the precise but limiting ACID framework and the imprecise but forgiving BASE framework. Furthermore, it explores alternatives to the relational paradigm by introducing the principles of *map-reduce* and *graph* paradigms.

1.1 Relationship with other teaching units

This course builds upon the basic programming and database courses. The required knowledge covers topics from object-oriented programming (including lambda abstractions), database design and implementation (ERD and relational modelling), and declarative programming in SQL.

2 Course program

The course is structured into six lecture units. The lecture units are not necessarily in a one-to-one correspondence with the course weeks.

2.1 Unit 1 - Review

The course starts with a quick review on SQL and RDBMS's:

Topics

- Entities and relationships
- SQL operators

2.2 Unit 2 - Normalization

Theoretical concepts behind normalization of relational models. Normalization algorithms:

Topics

- Redundancy problem.
- Definition of functional dependencies.
- Normal forms definition.

2.3 Unit 3 - Normalization algorithms

In this unit we cover the main normalization techniques.

Topics

- Normalization in 1NF, 2NF, 3NF, BCNF.
- Exercises on normalization.

2.4 Unit 4 - Concurrency

The fourth lecture covers handling of potentially conflicting concurrent query execution in an ACID DBMS:

Topics

- ACID property.
- Serialization
- Locks
- Deadlocks and their prevention

2.5 Unit 5 - NOSQL: Map-Reduce paradigm

The fifth lecture covers map-reduce paradigm:

Topics

- Map function.
- Reduce function.
- Map-Reduce is SELECT-FROM-WHERE
- Idea behind NoSQL
- LINQ and map-reduce in LINQ.

2.6 Unit 6 - Graph databases

The sixth lecture covers a specific example of no-SQL databases, specifically graph databases:

Topics

- Directed vs undirected graphs
- Adjacency list vs matrix
- Algorithms on graphs
- Case study: Neo4J

3 Assessment

The course is tested with an exam made of two parts: a theory part and a practical part. The final grade is determined as follows:

```
0.6 * practical + 0.4 * theory
```

To pass the exam the final score must be $\geq$ than 5.5. The theory part will contain questions about database normalization and transaction concurrency. The practical part will contain programming exercises about map-reduce and graph databases. The students must bring a working laptop with them with the necessary tools to run code in C# and with Neo4j installed.

Bijlage 1: Toetsmatrijs

Learning goals	Dublin descriptors
RDBMS	1, 2, 3, 4, 5
NORM	1, 2, 5
TRANS	1, 4
NONREL	1, 2, 5

Dublin-descriptors:

1. Knowledge and understanding
2. Applying knowledge and understanding
3. Making judgements
4. Communication
5. Learning skills

Exam structure

Theory

What follows is the general structure of a DEV5 exam. You can find a concrete exam sample in the course page on natschool.

Associated learning goals: `NORM`, `RDBMS`.

3.0.0.1 Question I: Normalization

Given a relational schema and defined functional dependencies, provide a normalized version in BCNF

Associated learning goals: `TRANS`, `RDBMS`.

3.0.0.2 Question II: Transaction management and concurrency

Given the following N-queries, which are run in parallel, show a strict 2PL scheduling. Detect deadlocks in a given transaction executions and identify how to break them.

Practice

Associated learning goals: `NONREL`.

3.0.0.3 Question I: Map-reduce

Implement the given queries with the map-reduce paradigm.

Associated learning goals: `NONREL`.

3.0.0.4 Question II: NoSQL databases en Graph Theory

Transform a given entity-relationship diagram into a graph model and implement the given queries in Cypher