HOGESCHOOL ROTTERDAM / CMI

Development 1

INFDEV03-5 (INFDEV22-5)

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

Modulebeschrijving

Module name:	Development 1
Module code:	INFDEV03-5 (INFDEV22-5)
Study points and hours of effort for full-time students:	This module gives 4 ects, in correspondance with 112 hours: • 3 x 6 hours frontal lecture. • the rest is self-study for the theory and practicum.
Examination:	Written examination and practicum (with oral check)
Course structure:	Lectures, self-study, and practicum
Prerequisite knowledge:	None.
Learning tools:	• Book: Database management systems (3rd edition); authors Ramakrishnan and Gehrke • Book: NO SQL Distilled; authors Sadalage and Fowler • Presentations (in pdf): found on N@tschool and on the GitHub repository `https://github.com/hogeschool/INFDEV03-5` • Assignments, to be done at home (pdf): found on N@tschool and on the GitHub repository `https://github.com/hogeschool/INFDEV03-5`
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` • **realise** a document-based database and 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 and document databases. • BASE vs ACID. • Graph databases.
Course owners:	Ahmad Omar, Francesco Di Giacomo
Date:	22 augustus 2016

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.

1.1 Relationship with other teaching units

This course builds upon the basic databases course.

Knowledge acquired through the databases course is also useful for some of the projects. A word of warning though: projects and development courses are largely independent, so some things that a student learns during the development courses are not used in the projects, some things that a student learns during the development courses are indeed used in the projects, but some things done in the projects are learned within the context of the project and not within the development courses.

2 Course program

The course is structured into seven lecture units. The seven lecture units take place during the seven weeks of the course, but are not necessarily in a one-to-one correspondance 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 - NOSLQ: Map-Reduce paradigm

The fifth lecture covers map-reduce paradigm:

Topics

- Map function.
- Reduce function.
- Map-Reduce is SELECT-FROM-WHERE
- Idea behind NoSQL
- Document database.

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 two exams: a series of practical assignments, a brief oral check of the practical assignments, and a written exam. The final grade is determined as follows:

0.6 * practicum + 0.4 * written exam
To pass the exam you must have a positive (i.e. ≥ 5.5) grade in both parts.

Motivation for grade A professional software developer is required to be able to program code which is, at the very least, *correct*.
In order to produce correct code, we expect students to show: *i*) a foundation of knowledge about how a programming language actually works in connection with a simplified concrete model of a computer; *ii*) fluency when actually writing the code.
The quality of the programmer is ultimately determined by his actual code-writing skills, therefore the final grade comes only from the practicums. The quick oral check ensures that each student is able to show that his work is his own and that he has adequate understanding of its mechanisms. The theoretical exam tests that the required foundation of knowledge is also present to avoid away of programming that is exclusively based on intuition, but which is also grounded in concrete and precise knowledge about what each instruction does.

3.1 Theoretical examination

The general shape of a theoretical exam for the course is made up of a series of highly structured open questions. In each exam the content of the questions will change, but the structure of the questions will remain the same. For the structure (and an example) of the theoretical exam, see the appendix.

3.2 Practical examination

The first two assignments are due by the end of week 5 of the course (Week 5 - Friday - 17.00). The other two are due by the end of week 8 of the course (Week 8 - Friday - 17.00). The assignments are handed in via GitHub, by sending an email to the lecturer with:

- Subject: **student number - course code - practicums**
- Content: **a link to the GitHub repository with your sources**

The repository:

- Has as name **student number - course code - practicums**
- Has one directory per assignment
- Has one readme file per assignment which sums up the results

The score for each assignment is given in the following way:

- **Assignment 1:** 3 pts.
- **Assignment 2:** 2 pts.
- **Assignment 3:** 4 pts.

The minimum grade is 1, and those points are added starting from the minimum score. It is mandatory to pass the exam to submit the first two assignments, and to score at least 6 points. The minimum requirement for an assignment to be evaluated is that it compiles and runs without crashing.

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

4 Practicum assignments

In this section the practicum assignments and their grading criteria are listed.

Assignment 1 The student is asked to implement a simple database for a given real-world scenario. This database is managed by a management application written in a chosen OO-language (Java or C#). The student is also ask to provide analysis on possible functional dependencies and provide a normalized version of the database.

Assignment 2 Implement the scenario of the first assignment in a document-based database. The implementation must come with a management application for the database using the map-reduce paradigm to execute given queries and to populate the database.

Assignment 3 Given the scenario described in the previous assignments, the student is asked to realise a correspondent graph database model, and implement it in graph DBMS Neo4j. The database must be populated by using the Cypher language coming with Neo4j, as well as the queries.

Exam structure

What follows is the general structure of a DEV5 exam.

Associated learning goals: `RDBMS`. **Points:** *25%.*

4.0.0.1 Question I: SQL queries

Given a database definition writes the SQL code for the requested queries

Associated learning goals: `NORM`, `RDBMS`. **Points:** *25%.*

4.0.0.2 Question II: Normalization

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

Associated learning goals: `TRANS`, `RDBMS`. **Points:** *25%.*

4.0.0.3 Question III: Transaction management and concurrency

Given the following N-queries, which are run in parallel, show plausible solutions
Associated learning goals: `NONREL`. **Points:** *25%.*

4.0.0.4 Question IV: Map-reduce

Implement the given queries with the map-reduce paradigm.

Associated learning goals: `NONREL`. **Points:** *25%.*

4.0.0.5 Question V: NoSQL databases en Graph Theory

Transoform a given relational data model into a graph model and implement the given queries in Cypher