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 7 hours frontal lecture (2 x 7 for part-time students) • 9 x 7 hours self-study for the practicum (10 x 7 for part-time students) • the rest is self-study for the theory
Examination:	Written examination and practicums (with oral check)
Course structure:	Lectures, self-study, and practicums
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 database in a modern, standard, SQL-based RDBMS `RDBMS` • **realise** integration between an OO application and a DBMS through an ORM mapping `ORM` • **analyse** and optimize performance of a queries `OPT` • **describe** the differences between relational and non-relational databases `NON-REL` • **describe** models of concurrency and transactions in a modern DBMS `TRANS-CONS`

Content:	• relevant concepts in relational databases and ORM's • performance issues with some query operators (`where`, `join`, etc.) • indexes and their role in performance • fundamental properties of DBMS's: atomicity, consistency, isolation, and durability (ACID) • concurrency and transactions • crash recovery • other tradeoffs than ACID and the CAP theorem: basic availability, soft state, and eventual consistency (BASE) • some BASE DBMS's: noSQL
Course owners:	Ahmad Omar, Francesco Di Giacomo
Date:	31 augustus 2015

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 Lecture 0 - Refresher

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

Topics

- Entities and relationships
- SQL operators

2.2 Lecture 1 - Object Relational Mapping

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

Topics

- Java database connectors JDBC
- Object relational mapping
- Java persistence API
- Sample implementation of ORM such as Hibernate or Eclipselink

2.3 Lecture 2 - Indexes

The second lecture covers performance issues in (relational) queries and the potential to use indexes to speed-up execution of such queries:

Topics

- Potentially slow query operators: `where`, `join`, etc.
- Sorting data by value: hash-tables and hashing algorithms
- Using a hash-index
- Sorting data by value comparison: balanced trees
- Using a tree-index

2.4 Lecture 3 - ACID

The third lecture covers the inflexible but precise framework of modern ACID (R)DBMS's:

Topics

- Transactions are integral: atomicity
- Database maintains valid state: consistency
- Transactions are serialized: isolation
- Database cannot break: durability

2.5 Lecture 4 - Concurrency

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

Topics

- Serialization
- Locks
- Deadlocks and their prevention

2.6 Lecture 5 - Crash recovery

The fifth lecture covers handling of disastrous events:

Topics

- Log
- Write-ahead log
- Checkpointing
- Recovery from a system crash

2.7 Lecture 6 - BASE

The sixth lecture covers an alternative tradeoff to ACID:

Topics

- Stay always connected: Basic availability
- Acceptance of partially wrong states: Soft state
- Slower propagation of updated information: Eventual consistency

2.8 Lecture 7 - Graph databases

The seventh 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 theoretical exam. The final grade is determined as follows:

`if theoryGrade` $\geq$ `75% & practicumCheckOK then return practicumGrade else return insufficient`
This means that the theoretical knowledge is a strict requirement in order to get the actual grade from the practicums, but it does not reflect the student's level of skill and as such does not further influence the grade.

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 practical examination is based on four assignments. The assignments are due by the end of the last week of the course (Week 7 - 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 final grade of the practicums must be $\geq 75\%$. The first two assignments must be done successfully. **provided that each practicum has a grade of at least 4**. Each practicum counts for 25% of the total grade.

Bijlage 1: Toetsmatrijs

Learning goals	Dublin descriptors
RDBMS	1, 2, 3, 4, 5
ORM	1, 2, 4, 5
OPT	1, 2, 4, 5
NON-REL	1, 5
TRANS-CONS	1, 4

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.

4.0.0.1 Assignment 1 The student is asked to implement a simple database for the management of the characters of a hypothetical Massive Multiplayer Online Role Playing Game (MMORPG) in Postgres given the specifications and the relational schema. Besides the student must implement an application that allows the user to subscribe to the game by specifying the user profile data, and execute the operations in the specifications on both his user account and the owned characters in the game.

4.0.0.2 Assignment 2 The student is asked to build indices over tables in the Postgres definition of the database for the given schema to optimize the performance of significant queries, and motivate his choices in a written report by analysing performance issues with those queries according to what explained in the theoretical classes.

4.0.0.3 Assignment 3 The student must implement a multi-thread version of assignment 1, where the program simulates multiple user starting different transactions to log in to their account and start a game session. The student should write a report about possible conflicts arising from different concurrent transactions trying to access server data.

4.0.0.4 Assignment 4 Given a database specification, the student is asked to realise a correspondent graph database model, and implement it in graph DBMS Neo4j. The user application should be able to interface to the database in Neo4j and implement the same functionalities as in Assignment 1.

Exam structure

What follows is the general structure of a DEV5 exam.

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

4.0.0.5 Question I: Impedence mismatch

Given the following code and data structure, both from the DB and the application, identify the impedence mismatch between equivalent representations

Answer:

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

4.0.0.6 Question II: Query optimization in SQL

Given the following queries identify at least N-columns to be indexed with suitable index

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

4.0.0.7 Question III: Transaction management and concurrency

Given the following N-queries, which are run in parallel, show plausible solutions

Answer: ...

Associated learning goals: `NON-REL`. **Points:** *25%*.

4.0.0.8 Question IV: NoSQL databases en Graph Theory

Transoform this relational data model into a graph model OR given this un-/directed graph identify the adjancy list or matrix of it

Concrete example of answer:

Grading: *All values are correct: full-points. At least half the values are correct: half points. Zero points otherwise.*