# CS4221 Project Topics

Stéphane Bressan

## 1 General Objectives, Requirements, Deliverables, and Evaluation

### 1.1 Objectives

The objective of this project is to design and implement a practical tool for the design and tuning of database applications using, but not exclusively, notions and techniques studied in this module.

### 1.2 Requirements

The tools should be implemented using the Bitnami Django stack with Apache, Python and PostgreSQL stored procedures (if necessary, can be written in PL/pgSQL or Python). In addition, the Web interface can be implemented using, as necessary and exclusively, HTML5, Javascript, SVG.

If an external library is necessary (anything that does not come out-of-the-box once the stack is freshly installed), it must be free, open source, not require any external account creation for use; consult us before adding one. The tools should be deployable uniformely as a publicly available service.

The tools should offer a user-friendly administrative interface for the management of users and resources.

### 1.3 General Deliverables

The deliverables comprise the source code, system documentation in HTML and an online user manual in HTML integrated with the general user interface.

The deliverables comprise a 30 minutes presentation cum demonstration (including questions and answers).

### 1.4 Evaluation

The evaluation is on 20 marks.

A functioning tool with basic features and documentation gives 10 marks. Major bugs, limitations, missing documentation incur penalties. 0 to 4 additional marks are assigned to the advanced features. 0 to 3 additional marks are assigned to the interface and usability. 0 to 3 additional marks are assigned to the quality of presentation and demonstration.

## 2 Topics

### 2.1 Topic 1: Relational Data Generator: Integrity Constraints

The goal of this project is to implement a data generator that takes as input a relational schema consisting of several table definitions with constraints and provides as output a set of data for this schema and its constraints. The data should agree with the SQL constraints indicated: `PRIMARY KEY`, `NOT NULL`, `UNIQUE` but also `FOREIGN KEY`, and possibly some simple `CHECK` constraints. Users should be able to control the statistical distribution of the selectivity of columns and join conditions between primary and foreign keys, as well as other distributions, if possible. The output format can be one or more of CSV, SQL `INSERT` statement (for some database management systems dialect), Excel, or other relevant formats.

### 2.2 Topic 2: Relational Data Generator: Realistic Data

The goal of this project is to implement a data generator that takes as input a relation schema and provides as output a set of realistic data from selected domains for this relation. Users should be able to control the statistical distribution of the selectivity of columns if possible. The tool allows users to generate tables with realistic data such as first and last names of person by nationality or ethnic background, addresses, telephone numbers, and credit card numbers. The system should be easily extensible if new domains and data for these domains become available. Data should be correlated if required by the user. For instance, German names may be required to have German address and telephone numbers. Users should be able to control the statistical distribution of the data. For instance, a user can require 10% European names, 10% Japanese names, 70% Chinese names, and 10% other names.

### 2.3 Topic 3: Entity-relationship Diagram Editor

The goal of this project is to implement a tool allowing users to interactively draw, save and edit entity-relationship diagrams. The tool should offer the option of several entity-relationship diagram (including the notation of "Introduction to Database Systems" by Bressan and Catania, as well as a crow foot notation). For each notation there should be a reference. The tool should include notational devices for aggregation and, possibly, hierarchies.

The project requires the design an XML notation for storing and exchanges entity-relationship diagrams. The notation needs to be designed with the teams assigned Topics 4, 5 and 6.

### 2.4 Topic 4: Entity-relationship to SQL DDL Generator

The goal of this project is to implement a tool that allows users to upload entity-relationship models in a format designed and agreed with the team working on Topic 3 and to generate, interactively or with annotations, if necessary, the corresponding SQL Data Definition Language statements with domains and constraints. The output is a working SQL code for PostgreSQL. In addition, different SQL dialects for different database management systems can be managed. Interactions or annotations are needed for the choice of the primary key, additional constraints and domains, for instance.

### 2.5 Topic 5: Entity-relationship to XML Schema Generator

The goal of this project is to implement a tool that allows users to upload entity-relationship models in a format designed and agreed upon with the team working on Topic 3 and to generate, interactively or with annotations, if necessary, the corresponding XML Schema with domains and constraints. Interactions or annotations are needed for the choice of the primary key, additional constraints, domains and nesting order, for instance. The tool should also be able to generate simple DTDs.

### 2.6 Topic 6: Entity-relationship to JSON Schema

The goal of this project is to implement a tool that allows users to upload entity-relationship models in a format designed and agreed upon with the team working on Topic 3 and to generate, interactively or with annotations, if necessary, the corresponding JSON Schema with domains and constraints. Interactions or annotations are needed for the choice of the primary key, additional constraints, domains and nesting order, for instance.

### 2.7 Topic 7: Data generator for XML documents constrained by a DTD

The goal of this project is to implement a data generator that takes as input a DTD describing a schema of possible XML documents, and that provides as output a (well-formed) XML document at random satisfying this DTD. Users should be able to control the statistical distribution interactively or by annotating the DTD. You may make reasonable simplifying assumptions on the language used to describe the DTD. The following features of DTDs should be kept: recursion, disjunction (the "|" operator), Kleene star (the "$*$" operator).

### 2.8 Topic 8: Data generator for JSON documents constrained by a JSON Schema

The goal of this project is to implement a data generator that takes as input a JSON Schema describing a schema of possible XML documents, and that provides as output a (well-formed) XML document at random satisfying this JSON Schema. Users should be able to control the statistical distribution interactively or by annotating the JSON Schema. You may make reasonable simplifying assumptions on the language used to describe the JSON Schema.

### 2.9 Topic 9: XML to JSON to XML to Relational

The goal of this project is to implement XML-to-JSON and JSON-to-XML, Relational-to-JSON and JSON-to-Relational, XML-to-Relational and Relational -to-XMLconverters. You may decide to explicitly ignore some advanced XML and JSON language constructs and syntax.

### 2.10 Topic 10: Relational to Neo4J to Relational

The goal of this project is to implement converters mapping a relational database instance to a Neo4J graph (Neo4J DML instructions to create the graph - and possibly some schema instruction such as constraints and indexes) and vice versa (SQL DDL and DML instructions to create the database and insert the data). The relationships in the Neo4J graph correspond to the referential integrity in the relational database (FOREIGN KEY constraints). You may decide to explicitly ignore some advanced SQL and Neo4J constructs and syntax.

### 2.11 Topic 11: SQL Lab

The goal of this project is to implement an automatic SQL lab system. Instructors set tests to be taken online. A test consists of a database and a sequence of questions, a visible database schema and a visible database instance. Instructors (or the system) create several hidden instances of the same schema. Instructors set some questions. A question is a query in English. The answer to the question is the corresponding query in SQL. Users take a test. When a user submits her answer to a question, the system checks the answer against the visible and hidden databases. The system validates the query if the answer by the user is identical to the answer by the instructor on all databases. The system may also measure and report the comparative performance of the answers (planning time, estimated and measured running time).