CS411 Project Topics

Stéphane Bressan, Pierre Senellart

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 tool should be implemented using the Bitnami WAP, LAP and MAP stacks with Apache, PHP and PostgreSQL. In addition, the Web interface can be implemented using HTML5, Javascript, SVG and other tools and languages working on most popular browsers.

The code should run on any of the three stacks. The functioning tool should be deployed on one of these stacks as a functioning publicly available service.

The service should comprise a sandbox for non-registered users to try the tool and its main functionalities as well as a complete version for registered users, where resource sharing and allocation is an issue.

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

1.3 General Deliverables

The deliverables comprise system documentation accessible from the administrative interface and an online user manual 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 A: Relational Data Generator: Integrity Constraints

Contact: Stéphane Bressan ⟨steph@nus.edu.sg⟩

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.

The teams working on Topic A and B should collaborate to allow the relational schema to be input or uploaded in the form of SQL data definition language statements with integrity constraints. They may consider a language for annotations for the quantities of data and their statistical distributions.

The tools of Topic A and B can be integrated into one single tool.

2.2 Topic B: Relational Data Generator: Realistic Data

Contact: Stéphane Bressan ⟨steph@nus.edu.sg⟩

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 from selected domains data 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.

The teams working on Topic A and B should collaborate to allow the relational schema to be input or uploaded in the form of SQL data definition language statements with integrity constraints in addition to any interactive user interface. They may consider a language for annotations.

The tools of Topic A and B can be integrated into one single tool.

2.3 Topic C: Entity-relationship Diagram Editor

Contact: Pierre Senellart ⟨pierre.senellart@nus.edu.sg⟩

The goal of this project is to implement a tool allowing users to interactively draw, save and edit entity-relationship diagrams following the notation of "Introduction to Database Systems" by Bressan and Catania. Extensions of this notation (e.g., aggregation) and other notation can be offered.

Users should be able to save the results for consumption by the tool of Topic D or for further editing.

The tools of Topic C and D can be integrated into one single tool.

2.4 Topic D: Entity-relationship to SQL DDL Generator

Contact: Stéphane Bressan ⟨steph@nus.edu.sg⟩

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 developing the tool of Topic C 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.

The tools of Topic C and D can be integrated into one single tool.

2.5 Topic E: Data generator for XML documents constrained by a DTD

Contact: Pierre Senellart ⟨pierre.senellart@nus.edu.sg⟩

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. You can choose the probability distribution implemented by this data generator arbitrarily, but if d is a document valid against the DTD, then the probability of generating a document *with same structure as* d (i.e., identical to d except for the value of text and attribute nodes) should be > 0. You can make simplifying assumptions on the language used to describe the DTD, but, as a bare minimum, the following features of DTDs should be kept: recursion, disjunction (the "|" operator), Kleene star (the "$*$" operator).

2.6 Topic F: Encoding XML into relations, and XPath into SQL

Contact: Pierre Senellart ⟨pierre.senellart@nus.edu.sg⟩

The goal of this project is to implement two functions:

- a function `XML2Relation` that transforms an arbitrary XML document d provided as input into a sequence of SQL `CREATE` and `INSERT` statements that creates a database D formed of one or multiple relations encoding the content of the XML document as a relation (you can use the encoding of your choice but no assumption can be made on the XML document as input);
- a function `XPath2SQL` that transforms a simple XPath expression q provided as input into a SQL `SELECT` statement Q that encodes this query (to simplify, you can assume that the XPath expression has a very simple form; at minimum, you need to support `/` and `//` navigation on the child and descendant axes).

The result of evaluating Q over D should be identical to the result of evaluating q over d.