# Web Data Management - Final Project
# Due Date: 1.3.2013
# Submit in pairs. See submission instructions below.

January 20, 2013

## 1 Project Goal

In this project you will implement a question answering mechanism based on data from Wikipedia.

## 2 General Instructions

1. You can implement the project in C++, C#, Python or Java.
2. Keep your source code well-structured, clear, and documented.
3. There are many design choices to be made in such a project. Explain your choices in the code as well as in the separate documentation (see next).
4. You have to submit a separate documentation. This document should include:
   - Clear and exact instructions on how to use your application.
   - At least 5 use cases, i.e. particular queries that you have tested and led to results that you find interesting, highlighting different features of your implementation. Write down the queries, the results, and an explanation on the way these were computed. Also report run-time.
   - An estimation of the precision of your algorithm for a (small) test set of your choice. Provide the precision and your methodology of computing it. If the precision is low, you should explain why you think this is the case and describe your attempts to improve it.
   - Any choices that you have made in design, choices of heuristics, parameters (if relevant), etc.

- Any other information that you think is essential for the evaluation of your project.

**The documentation is an integral and important part of the project. The project grade will be greatly affected by the documentation quality**.

# 3 Submission Instructions

Burn you project, including source code, documentation, executables, data, and anything else that is needed to run it, on a CD. Write your names and ID numbers on the CD, and submit it to my mailbox: Building 37, mailbox 64.

Before you submit, copy the CD contents to a "clean" folder on another computer to check that it runs properly. Keep a copy of your project for your record.

# 4 Tasks

1. For this task you should implement the evaluation of safe queries on an in-memory probabilistic database. The schema of the database can be hard-coded, but data itself should be fed as external input to the program.

   The hard-coded schema should include the following relations, *and at least additional 4 relations of your choice* (choose these after exploring the Wikipedia dataset described in the next task).

   - BornIn(Person,Year)
   - Profession(Person,Prof)

   Design the schema, including keys for the different relations. You need to support some simple *positive relational algebra (combination of join, projection and selection* queries *for which a safe plan exists.* It is OK for the queries to be hard-coded, but your code should be designed in a way tat will allow to easily add support for queries (as this will be needed in questions that follow).

2. In this task you will use Information Extraction techniques to fill in your tables with data derived from Wikipedia. A parsed text derived from Wikipedia is available in the web-site, and you may use it.

   Implement multiple heuristics for Information Extraction for the specific relations you have designed (at least 3), and be explorative: try to get as much information as possible (including contradicting facts).

   Design a way to assign a probability (confidence score) for each heuristic (describe how your chosen way) and use these probabilities to decide the probability of derived tuples, stored in your database.

Report (as part of your documentation) the precision of the answers computed by your system on a test set of your choice.

3. Finally, implement a User Interface, allowing users to pose queries on your database. The supported queries should include at least the following, *and at least additional 5 types of queries of your choice*:

   - When were X born?
   - Which people who have profession X, were born in year Y?

   At least 2 of your additional queries should involve joins.

   The user should be able to pose this fixed set of queries (parameterized by choices) in a natural way, i.e. via your User Interface (that may be textual or graphical) Your system should compile the query into relational algebra query, feed it to the framework that you have implemented in Q1, and return a set of answers, each associated with probability (interpreted as the system confidence in the answer).