INF344, Télécom ParisTech

MayBMS

Pierre Senellart (pierre.senellart@telecom-paristech.fr)
(based on original material by A. Amarilli)

20 June 2016

The goal of this lab session is to play with relational formalisms to manipulate probabilistic data. This is done with MayBMS[1], a probabilistic database implementation built on top of PostgreSQL[2].

We consider a *truth finding* application, where we have extracted facts from several sources (i.e., websites). A first table, Trust, indicates which sources are trustworthy. Another table, Claims, indicates which sources support which fact. Both tables are uncertain: we do not know which sources are correct, and we are not sure of whether a source supports a statement (i.e., errors may have been made when extracting facts from sources).

We represent the uncertainty on these two relations using the TID model. We consider the following instance:

Trust

source	proba
Legifrance	0.95
Wikipedia	0.8
Doctissimo	0.4
Gorafi	0.2
Onion	0.1

Claims

source	claim	proba
Legifrance	42	0.7
Wikipedia	42	0.9
Gorafi	42	0.6
Wikipedia	51	0.7
Doctissimo	51	0.4
Wikipedia	66	0.3
Gorafi	66	0.9

1 Submission and Evaluation

Submit your solution by Sunday, July 3, 11:59pm on the submission server for full credit. Submissions received after this date and before Monday, July 4, 8:30am will incur a penalty of 4 points out of 20. Submissions received after this second deadline will not be considered. As usual, for this lab you only need to submit the `MayBMS.java` file, not the whole Eclipse project.

Your submission will first be run against the public tests; as they are identical to the ones provided in the PublicTests class, there is no need to use the submission server unless your submission passes these tests locally. A score out of 20 points will be given as an indication (this score is not part of the grade). When confident about your submission, you can release it to test it against the secret (release) tests. The score obtained on the release tests will be used as a grade for this lab session. You are allowed at most 3 release submissions per hour. Evaluation of your submissions will typically take several minutes.

[1] http://maybms.sourceforge.net/
[2] http://www.postgresql.org/

2 Environment

You will connect to an instance of the MayBMS probabilistic relational database management system on the `tiresias.enst.fr` machine (only accessible from within the school network). Your username and password for access to this database engine are your Unix login on Télécom machines (your password is also your login, it is not your Unix password). You have access to one database on the `tiresias` machine, whose name is also your Unix login.

You can connect to the MayBMS instance:

- from the command line, using the `psql` program, which is used as follows:

  ```
  psql -h host -U login database
  ```

  where `host` is the server name, `login` the user name, `database` the database name; the `psql` tool will ask for your password;

- from a Java program, using the standard JDBC interface to PostgreSQL, see `https://jdbc.postgresql.org/documentation/83/connect.html`. You can also check `http://docs.oracle.com/javase/tutorial/jdbc/basics/processingsqlstatements.html` for an introduction to JDBC.

3 Assignment

1. Write the `getLogin()` method, that should just return your Unix login.
2. Write the `connectToMayBMS()` method, that should establish a JDBC connection with the MayBMS instance and store that connection in the `db` class attribute of the `MayBMS` class. The argument of the function is the name of the database to connect to.
3. Write the `sendQuery()` method, that takes as argument a SQL query, send it to the MayBMS instance, and return the corresponding `ResultSet` object.
4. In this question. We disregard the proba column of all tables, and consider that all tuples have probability 1. We wish to determine the answer to the following query: "Which sources in our dataset are trustworthy and make at least one claim?" Implement the `getSources()` function that returns a `Set<String>` of all such trustworthy sources making at least one claim.
5. We now want to take into account the column proba as being the probability of each tuple in the TID model. By hand, in order to understand how such a probability can be computed, compute the answer to the query of the previous question with the correct probabilities according to the TID model. The result should be a probabilistic annotation of the result of the previous question. Do this in two steps: first, compute for each source the probability that it makes a claim, and then combine this with the Trust table to obtain the final answer. There is no code to write in this question and nothing to hand out, but it is critical that you perform this computation by hand to be able to follow the next questions. You can use a calculator or the computer for numerical computations.
6. Now, we will use standard PostgreSQL to perform the same computation as in the previous question, *without relying on the features of MayBMS*.

Again, you can proceed in two steps (first computing for each source the probability that it makes a claim, and then combining this with the Trust table). This will yield a SQL query with nesting, the nested query representing the first step, the enclosing query the second one.

For the first step, using the SQL `GROUP BY` operator, you will write a query that computes the projection of the Claims table to **source**, with the correct probabilities. *Hint:* As there is no `PRODUCT` aggregate function in SQL, use the `SUM` aggregate and the `exp` and `ln` functions to write the query. Check that the probabilities that you obtain correspond to the intermediate result computed by hand in the previous question (before using table Trust).

For the second step, write a query that joins the result of the first step (as a nested query) with the table Trust and computes the final query result with the correct probabilities in column proba. Check that you obtain the same result as in the end of the previous question.

Finally, implement the `getTrustworthySources()` function that evaluates this query with nesting on the database, and put the result in a Java `Map`, mapping each source to its probability.

7. We now move to the use of MayBMS operators for probability computation. We want to execute the same queries as before, but this time using the specific features of the probabilistic DBMS that we are using, MayBMS[3]. We will use the Trust and Claims table created in the previous section. First, create tables TrustP and ClaimsP that are the probabilistic counterparts of the deterministic tables Trust and Claims, using the MayBMS `PICK TUPLES` command (check the MayBMS tutorial to see how to use this command). Because of a specificity of the JDBC interface, these tables need to be created with the `psql` command line tool, not within the Java program.

8. We will now use MayBMS to evaluate the query studied in the previous questions, on the TrustP and ClaimsP tables. We will be able to leave probability evaluation to MayBMS.

   Implement the `getTrustworthySourcesMayBMS()` that should behave exactly as the `getTrustworthySources()` except that it should use MayBMS `conf()` aggregate function for probability computation.

9. Let us now showcase the power of MayBMS with a more complex query: "Which claims are stated by at least two different sources which are both trustworthy?". Write this query in the relational algebra, translate it to an SQL query, and execute it on the TrustP and ClaimsP tables. Use this to compute the probability, for each claim, that it was stated by two different trustworthy sources. Implement accordingly the `getClaimsTwoDifferentTrustworthySourcesMayBMS()` function that returns, as a Java `Map`, for each claim, the probability that this claim is stated by two different trustworthy sources.

10. We now consider the following additional table in SQL, which indicates uncertainty about whether a claim is relevant to the user:

Relevant

claim	proba
42	0.2
51	0.4
66	0.8

[3] A manual and quick tutorial can be found at `http://maybms.sourceforge.net/manual/index.html`

This table is already present in the DBMS, under the name Relevant. Create the RelevantP probabilistic table as you did for the two other tables.

11. We wish to determine whether there is a trustworthy source that makes a relevant claim. We want to determine the probability of this query using MayBMS. Using the two probabilistic tables TrustP, ClaimsP, and RelevantP, write the `isTrustworthySourceMakesRelevantClaimMayBMS()` function that should return the probability that the query is true.

12. The query of the previous question is a typical example of a *hard* query. We are able to compute its probability because we are using a very small dataset, but this would not be possible with a slightly larger dataset, since only an exponential-time algorithm exists for this task. Using the `aconf()` aggregate function (that takes as argument a pair (ϵ, δ) and returns a relative ϵ-approximation with probability $1 - \delta$), approximate the probability of the same query in the `isTrustworthySourceMakesRelevantClaim2MayBMS()` function. We want an approximation within a factor of 0.01, and with probability 0.99.