Universitat Politècnica de Catalunya

Master Thesis

Datamining on an online judge

Author:
Maxime Marlier

Supervisor:
Jordi Petit

A thesis submitted in fulfillment of the requirements for the degree of

in the

Universitat Politècnica de Catalunya
Master in Innovation and Research in Informatics

Last build : 2016-04-29, 19:11
Revision : f83ac1c
Revisiondate : 2016-04-29

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Abstract

Facultat d'Informatica de Barcelona
Master in Innovation and Research in Informatics

Datamining on an online judge

by Maxime Marlier

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Contents

List of Figures

List of Tables

Chapter 1

Introduction, motivation and goals

Chapter 2

State of the art

2.1 Database description

The figure 2.1 (page 4), is a part of the *Entiry Relationship Diagram* representing the interessing part of the database.

Here is the list of the table in the database:

2.1.1 users

users	
♦**user_id**	**text**
•creation_date	date
•administrator	int
•instructor	int
•demo	int
◦unregistered	int

Description:
The first table contains the users. For this analysis, the user table has been anonymized. We only refer to a user ID, and his contributions in the data base. Personal data from users will not be used for analysis. Only the creation date is kept for a time based analysis.

However, it's needed to exclude some *non-representative* users :

- Some users used for developement ([list])
- Users with a id patern different that *Uxxxxx* (Users used for competition for exemple)
- Demonstrations users ($demo == 1$).
- Instructors, administrators and unregistred users (cf flags atributes in the database).

Numbers:
In term of numbers, the database contains:

- 10565 users in total.
- 55 unregistred users.
- 50 instructors.
- 7 administrators.
- 1 demo user.

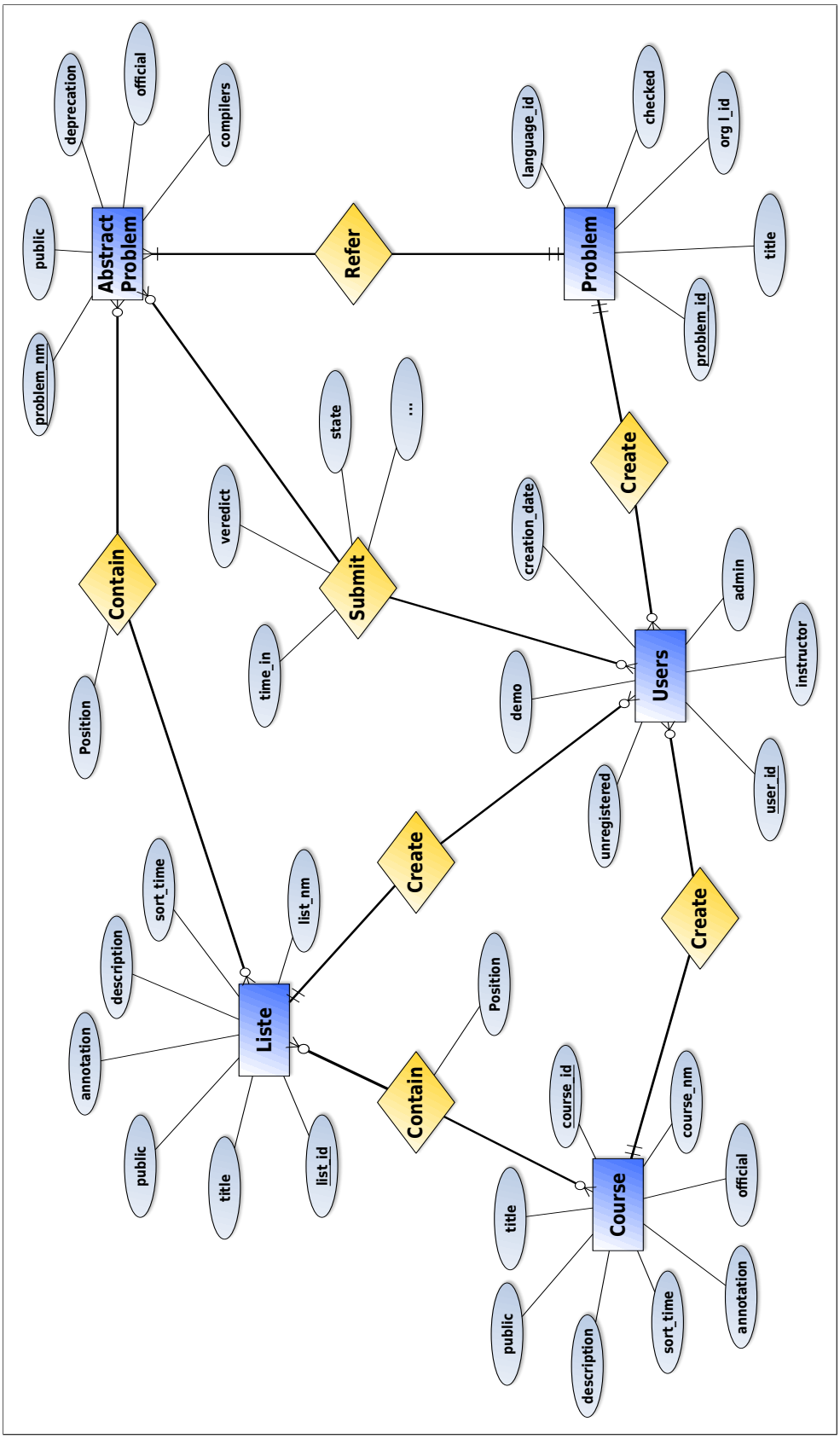

FIGURE 2.1: Entiry Relationship Diagram

2.1.2 problems and abstractproblems

problems	
♦problem_id	text
•problem_nm	text
•language_id	text
•title	text
•original_language_id	text
•checked	int

Description:
A single problem could be proposed in various languages but the language variation doesn't affect the technical details of a same problem. That means that the way how a submission would be processed is never linked to the language[1].

That explains those two tables discribing the problems. The first one colled *abstractproblems* contains the technical informations for submission management. The second one, *problems*, is the description of a problem, according to a specific language (*language_id*) and refering to a *abstractproblems*. -nm There is different types of problems, distinguishable by their *problem_nm* patern:

- *Pxxxxx*
  Those type of problems will be our baseline for the analysis. In fact, they are the *offical* problems initialy present in the database, created by the designers. We can consider them as *right* and *relevant* in term of submission and verdict[2].
- *Xxxxxx*
  The letter *X* means *externe*. Those problems have been created by users (*instructors*) and havn't been validatd by anyone. Moreover, only a portion of users can acces to it (Those who suscribed to the courses related to the same instructor)
- *Gxxxxx*
  The letter *G* means *game*. Those problems are used on a very specific scenario. There is only a very few of them and they will be ignored in our analysis.
- *deprecated*
  Obviously, this type of problem is not relevant for the analysis.

Numbers:

- 1909 abstractproblems in total.
- 1325 Pxxxxx like abstractproblems.
- 575 Xxxxxx like abstractproblems.
- 9 Gxxxxx like abstractproblems.
- 85 deprecated abstractproblems (including 21 Pxxxx type).

abstractproblems	
♦problem_nm	text
•user_id	text
•public	int
•official	int
◦compilers	text
◦deprecation	text
•checked	int

Languages distribution:

[1] There are actually few problems which differ between languages for inputs or outputs regarding to the language but those are negligible

[2] The concept of verdict will be explain in the following section *submissions table*

TABLE 2.1: Frequency distribustion of verdict accros every relevant submissions (Related to figure 2.2 on page 7)

Acronym	Verdict	%
AC	Accepted	43.62
WA	Wrong Answer	30.06
EE	Execution Error	11.41
CE	Compilation Error	10.70
PE	Presentation Error	3.62
SC	Scored	0.30
IC	Invalid Character	0.29
SE	Setter Error	0.01
FE	Fatal Errors	0.00
NC	Noncompliant Solution	0.00
Pending	Pending Submission	0.00
IE	Internal Error	0.00

2.1.3 submissions

submissions	
♦submission_uid	text
•user_id	text
•problem_id	text
•submission_id	text
•compiler_id	text
•state	text
•time_in	timestamp(0)
◦time_out	timestamp(0)
◦veredict	text
◦veredict_info	text
◦internal_error	text
◦legacy	int
◦veredict_publics	text
◦ok_publics_but_wrong	int
◦score	text

Description:

Every instance in this table represents the submission of a solution for a specific problem (*problem_id*) by a specific user (*user_id*) at a given time/moment (*time_in (time_out)*). From that submission (after a internal process) will stand out a *verdict* meaningful of the submission correctness.

This table is one of the most important for our analysis. Indeed, this on contains the usage history of the Jutge.

Numbers:

- 1605270 submissions in total.
- 43.62% of accepted submissions (cf. *Verdict distribution*)
-

Verdict distribution : As shown on the following table and on the figure 2.2, 95% of the submissions are distributed among those 4 verdict :

- Accepted (43.62%).
- Wrong Answer (30.06%).
- Execution Error (11.41%).
- Compilation Error (10.70%).

So for our first analysis, we will group the non-accepted instances and consider a boolean structure with only accepted or rejected submissions.

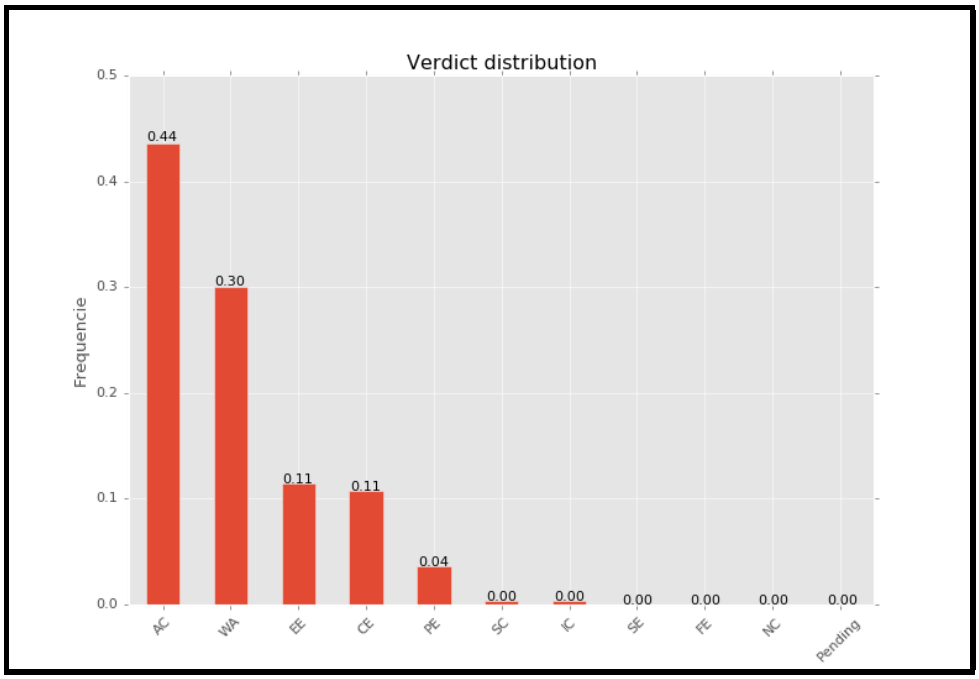

FIGURE 2.2: Frequency distribustion of verdict accros every relevant submissions

2.1.4 Courses organisation

As shown on the figure 2.3, there is several levels under the idea of courses. To summarize, a course invloves a list of users subsribed to it. Than a course is devided into sections, and those sections are basically lists of problems.

All of thisi is implemented with 5 tables in the database (All described hereinafter).

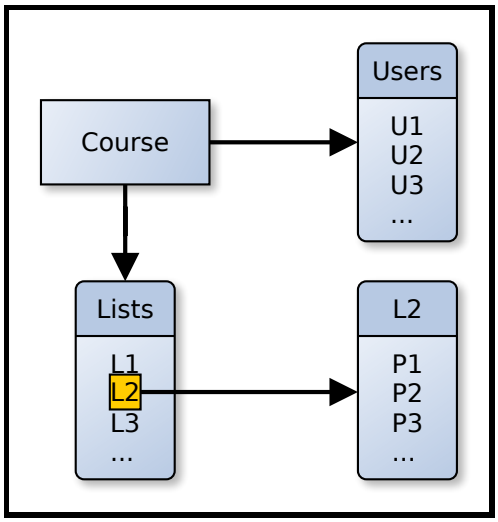

FIGURE 2.3: Courses organisation - Subscribed users, Lists of problems, problems

2.1.5 courses

Description
General description of a course, linked to creator user and containing , inter alia, its title, description and annotation.

courses	
♦course_id	text
•user_id	text
•course_nm	text
◦title	text
◦description	text
◦annotation	text
•public	int
•official	int
◦sort time	timestamp

Numbers

- 112 courses in total.
- 18 of them are public.

2.1.6 coursesusers

Description
Join table for the many-to-many relation between users and courses, matching a *users_id* with a *course_id*. It further includes an attribute for users known as tutor for the concerned course.

coursesusers	
♦course_id	text
◦user_id	text
•tutor	int
◦tag	text

2.1.7 lists

Description
General description of a list of problems, linked to creator user and containing , inter alia, its title, description and annotation. Conceptually, a list is very closed to a course. The conceptual difference appears in a many-to-many relation between those tables. The idea is to include lists inside courses.

lists	
♦list_id	text
•user_id	text
•lit_nm	text
•title	text
◦description	text
◦annotation	text
•public	int
•official	int
◦short time	timestamp

Numbers

- 607 lists in total.
- 157 of them are public.

2.1.8 courseslists

Description
Join table for the many-to-many relation between courses and lists of problems, matching a *list_id* with a *course_id*. It further includes an attribute indicating the position of that list inside the concerned course.

courseslists	
♦course_id	text
♦list_id	text
•position	int

2.1.9 listitems

Description
Join table for the many-to-many relation between lists and problems, matching a *list_id* with a *problem_nm*. It further includes an attribute indicating the position of the problem in that list.

listitems	
♦list_id	text
♦position	int
◦problem_nm	text
◦description	text

2.1.10 problemstags

problemstags	
♦problem_nm	text
•tag	text

Description
This is a arbitrary classification of the problems, defined by problems creators or tutors. This information wont be used for the analysis but can be intresting in a further validation process.

2.1.11 compilers

compilers	
♦compiler_id	text
•name	text
•language	text
•extension	text
◦description	text
◦version	text
◦flags1	text
◦flags2	text
◦type	text
◦warning	text
◦status	text
◦notes	text
◦	

Description
This table countains the techincal information of compilers inplemented into the Jutge. When you want to submit a solution to a problem, you can chose between differents laguages, or event different version of compilers.

As shown on the compiler distribution, this information is not relevant for our analysis because the mostly used language inside the application is *C++* (90%)! The information of lagage used for a submission will be ignored in the begining.

Chapter 3

Methodology

Chapter 4

Development of the proposal/technical/work

Chapter 5

Evaluation of the proposal/technical/work

Chapter 6

Conclusions

Appendix A

Appendix

Bibliography

[1] Leslie Lamport, *LaTeX: a document preparation system*, Addison Wesley, Massachusetts, 2nd edition, 1994.