ISBN 2-9527275-0-3
ISBN 978-2-9527275-0-1

Corrector, a web interface for practice sessions

Fabien Coelho – fabien.coelho@ensmp.fr
CRI, ENSMP, 35, rue Saint-Honoré, 77305 Fontainebleau, France

Résumé

L'application Web Corrector valide automatiquement les réponses proposées par un étudiant en séance pratique. L'enseignant n'intervient que pour résoudre les problèmes de compréhension. Corrector est particulièrement adapté au langage de base de données SQL, dont les nombreuses requêtes équivalentes sont traitées astucieusement. D'autres modes de correction permettent de gérer la modification des données ou des structures, utilisent des signatures cryptographiques ou des expressions régulières.

Abstract

The Corrector web interface provides immediate feedback to students during practice sessions by validating answers. The teacher needs only focus on solving comprehension problems. Corrector is especially well fitted to the SQL database language, as it deals with questions which admit many equivalent queries. Other correction modes handle modifying data contents or structures, use cryptographic proof-of-success tokens or regular expressions.

Motivation

Teaching computer science becomes harder as students are less akin to focus on technical fields. Relational databases [2] and their SQL [5] data manipulation language is a technical subject which requires abstraction efforts. A typical class alternates theoretical blackboard presentations with practice sessions hands-on, where students develop queries to answer increasingly challenging questions. However, many queries are equivalent in the relational model, thus it is often unclear to a student whether an answer is right. Teacher validation during session is time consuming and reduces the available time to provide help.

Many graphical interfaces exist to help teach SQL. For instance, SQL Tutor [6] is a rule-based system which helps a student to find a fixed query from an exercise set. It provides a targeted help at the price of extensive tailoring, and cannot deal with equivalent queries. Other tools are dedicated to fill-in the gap exercises [8] or programming [7]. Our approach extends [4], but is not limited to SQL select statements.

Web interface

Our course uses PostgreSQL [1], a feature-full free standard object-relational database management system. An example data set about comics is provided for practice: although queries to find *the average salary in the finance department* and *the average number of pages of comics published by Delcourt* may have the same structure, students find the later more attractive. This preference changes when they begin to look for a job.

The Corrector web interface is built on top of PostgreSQL, using standard tools such as HTML, Apache, CGI, Perl. It is internationalized using *portable object* files, with message translations available in English and French. The security is handled based on permissions declared by the teacher. Careful configurations mitigate the risk of executing student code, by isolating and limiting the queries.

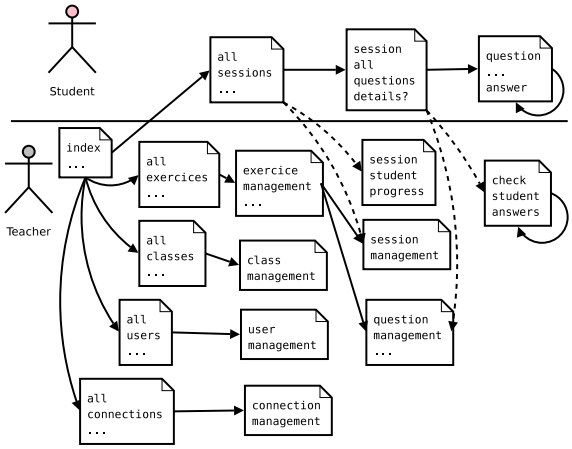

Fig. 1 Interface architecture

Figure 1 outlines the interface. Each page presents a dynamically generated contents based on the user identity and additional parameters. Students are restricted to the upper interface: it accesses active practice sessions which point to their questions. The rich teacher interface manages students, classes, exercises, questions, connections and sessions.

Correction techniques

Corrector validates proposed answer interactively, using from basic to advanced correction modes.

First, answers can be stored for later manual correction. The validation of a question is performed for all answers in a single step, and allows to provide a specific comments to each student.

Second, a set of correction modes compares the provided answer with a target string, including regular expression matches. These modes have been proven hard to be highly effective, especially when keywords are expected, as students are very imaginative.

Third, SQL queries are actually *executed* on the database: for selects, the result is compared to the result of a reference query, and the first difference is reported if any; for other statements, database transactions are used to rollback the student changes at the end of the validation, so that the reference database is not modified. The validation itself is performed with tailored queries to check whether the intended transformations was performed, before cancelling them.

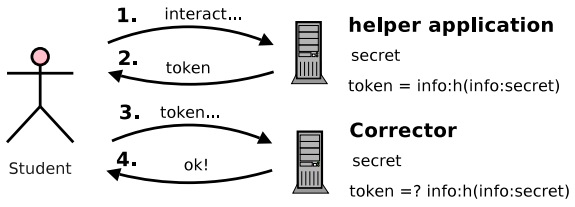

Fig. 2 Cryptographic validation

Fourth, a cryptographic token-based correction mode is provided, shown in Figure 2. The student interacts with a helper application, say a web server in a practice about the HTTP protocol. When the interaction succeeds, the helper application releases a proof of success token which involves a public information, such as the student login name obtained with the ident protocol, and the cryptographic hash of this public information and a question specific shared secret.

$$\text{token} = \text{info:}h(\text{info:secret})$$

When receiving the token, Corrector recomputes the hash from the provided information and the secret to validate it.

Discussion

Crafting questions suitable to the corrector interface is not straightforward. There is a communication problem, as there must be only one possible response from the student. Indeed, the automatic correction cannot deal with good alternate answers to ambiguous questions and attribute points to them. Thus all SQL query questions must tell precisely the attributes expected as well as the order in which the results must be presented, thus requiring full *order by* clauses in all answers.

Students often use the web interface as an interface to the database, not bothering to test queries first. Such strategy is effective for simple queries, but not for complex ones: the interface is not a substitute to the text and graphical database interfaces which provide help about syntax or semantical errors.

Most students never look at the expected answers after the practice, once the corrections are available. We're planning to push the relevant information, possibly through a simple mail pointing to a summary web page.

Some students develop creative strategies to get rid of the practice quickly while still getting the positive feedback from the interface. For instance, if a small number is expected, they would try all integers from 0 till they get the point, or they would submit an entire dictionary in the hope of hitting the wanted keyword. Browsing submitted answers is necessary to detect and deter with reduce marks such behavior.

Although the teacher interface allows to create new exercises, it is seldom used: a paper version of the practice is also useful, thus importation scripts take special LaTeX files with added comments to feed the system. However, the teacher interface is often used to fix questions on the fly during the session when students stumble upon problems or ambiguities.

Conclusion

More than 150 students have used Corrector on practices about databases, network protocols and cryptography. The teacher's time spent on preparing and performing lessons has **not** been reduce. However the student and teacher experiences are both improved. More time is spend on the preparation and the configuration, and less time on the correction.

I intend to distribute Corrector as a free software at some stage. However it requires a precise documentation, especially on the system installation issues which involve careful database and web server configuration, and which is yet to be written. Future works also involve new features, such as an examination mode which does not report the points to the student. Further descriptions can be found in a technical report [3].

References

[1] PostgreSQL. `www.postgresql.org`, 1996–2006.

[2] E. F. Codd. A relational model for large shared databanks. *Communications of the ACM*, 13(6):377–387, June 1970.

[3] F. Coelho. Corrector, a Web Interface for Practice Session with Immediate Feedback. Technical Report A/377/CRI, CRI, École des mines de Paris, Apr. 2006. `http://www.cri.ensmp.fr/classement/doc/A-377.pdf`.

[4] J. Coleman Prior. Online assessment of SQL query formulation skills. In *ACE '03: Proceedings of the fifth Australasian computing education conference on Computing education 2003*, pages 247–256, Darlinghurst, Australia, 2003. Australian Computer Society, Inc.

[5] ISO/IEC. Information technology - database languages - SQL, 2003. Standard 9075.

[6] A. Mitrovic. Learning SQL with a computerized tutor. In *SIGSCE'98, Atlanta, Georgia, USA*, pages 307–311, 1998.

[7] Y. Pisan, D. Richards, A. Sloane, H. Koncek, and S. Mitchell. Submit! a web-based system for automatic program critiquing. In *ACE '03: Proceedings of the fifth Australasian computing education conference on Computing education 2003*, pages – , Darlinghurst, Australia, 2003. Australian Computer Society, Inc.

[8] N. Truong, P. Roe, and P. Bancroft. Automated feedback for "fill in the gap" programming exercices. In *ACE'05: Proceedings of the seventh Australasian computing education conference on Computing education 2005*, pages 117–126, Darlinghurst, Australia, 2005. Australian Computer Society, Inc.

Typeset with LaTeX, document revision 1386