International Journal of Engineering Education Vol. 32, No. 3(A), pp. 1289–1312, 2016
Printed in Great Britain

0949-149X/91 $3.00+0.00

A Software Tool for E-Assessment of Relational Database Skills*

ALBERTO ABELLÓ[1], XAVIER BURGUÉS[1], M. JOSÉ CASANY[1], CARME MARTÍN[1], CARME QUER[1], M. ELENA RODRÍGUEZ[2], OSCAR ROMERO[1] and TONI URPÍ[1]
[1] Universitat Politècnica de Catalunya, Jordi Girona 1–3, Campus Nord, Barcelona, Spain.
E-mail: {aabello, diafebus, mjcasany, martin, cquer, oromero, urpi}@essi.upc.edu
[2] Universitat Oberta de Catalunya, Rambla del Poblenou 156, Barcelona, Spain, Email: mrodriguezgo@uoc.edu

The objective of this paper is to present a software tool for the e-assessment of relational database skills. The tool is referred to as LearnSQL (Learning Environment for Automatic Rating of Notions of SQL). LearnSQL is able to provide automatic feedback, and grade the responses of relational database exercises. It can assess the acquisition of knowledge and practical skills in relational database that are not assessed by other systems. The paper also reports on the impact of using the tool over the past 8 years by 2500 students.

Keywords: computing engineering education; interactive learning environments; improving classroom teaching; blended learning; authoring tools and methods

1. Introduction

The increase of several forms of e-learning in the last few years (e.g., Massive Open Online Courses, Virtual/Online Universities, etc.) has fostered the emergence of e-assessment systems to support the automatic evaluation of students' responses to proposed exercises.

E-assessment can be defined as the process where information and communication technologies are used for the management of the end-to-end assessment process [1, 2]. In other words, e-assessment deals with methodologies, processes, and web-based software tools that allow systematic inferences and judgments to be made about the learners' skills, knowledge, and capabilities [3]. The web-based software tools used in e-assessment can provide different capabilities. These tools are referred to in the literature as e-assessment systems when they allow the delivery of assessment activities, the recording of responses, timely feedback, automatic grading, and weighted-average grade calculation, thus enabling formative assessment during the teaching and learning process [4–7].

E-assessment systems were primarily used for testing the acquisition of declarative knowledge, for example, by means of automatically corrected quizzes, compounds of simple types of questions such as multiple-choice question, etc. [4, 8]. However, cognitive skills where students have to apply their analytical, creative, and constructive skills cannot be assessed via multiple-choice tests or equivalent forms of basic assessment items [9, 10]. In order to test higher order cognitive capabilities, sophisticated assessment activities must be designed and an interactive dynamic environment must be introduced. This implies that e-assessment systems that support skill assessment are technologically more complex because of the computational difficulties of representing and simulating higher order cognitive exercises and their automatic marking. Furthermore, they are specifically developed for specific course content or fields within a discipline.

In traditional face-to-face courses of European universities, one factor that promoted the emergence of e-assessment systems was the adaptation in 2010 of the academic programs to the European Higher Education Area (EHEA), which encouraged some level of blended learning. A wide range of e-assessment systems and tools that support blended learning emerged in most fields of Computer Engineering. Some examples of these systems and the fields in which they are used are Jutge.org [11], Mooshak [12] and the system presented in [13] for programming languages; VerilUOC [14], LabView [15], and e-EDU [16] for digital circuit design; and the TEA system [17] which includes an intelligent tutor system for teaching logic [18]. In addition, [19] and [20] provided surveys of automated assessment tools for programming courses.

As previously stated, each field has its own difficulties. This implies that the developed e-assessment systems and tools are specific to a particular field. In the case of the database field, several difficulties arise. First, the variety and diversity of exercises requires dealing with a broad set of evaluation methods to guarantee the correctness of students' responses. Second, the complex technology that the Database Management Systems (DBMS) require must be taken into account in the correction

* Accepted 20 February 2016.

of exercises. Because of these difficulties, most of the existing database e-assessment systems only address the learning of how to write SQL Queries, and to the authors' knowledge, no system addresses the complete set of database topics that are covered in common university database courses [21].

The objective of this paper is to provide a detailed overview of LearnSQL (Learning Environment for Automatic Rating of Notions of SQL), which is a software tool that allows the automatic and efficient e-learning and e-assessment of relational database skills. LearnSQL supports the learning and assessment of the following relational databases topics: SQL DML, SQL DDL, Logical Design, Normalization, Relational Algebra, SQL/PSM, Triggers, Multi-dimensional Operations, Programming with SQL, Optimization and Cost Estimation. Since 2008, LearnSQL has been used for 16 semesters with an average of 160 students per semester, although the system was not completely deployed in all database courses until the autumn semester of 2009. Prior to the use of a full-fledged version of LearnSQL, some pilots were conducted and some preliminary results were discussed in [22, 23]. A demonstration of LearnSQL was performed in [24]. Finally, a proposal to increase the collaborative learning capabilities of LearnSQL was presented in [25].

In order to achieve the previously stated objective, this work presents, on the one hand, the LearnSQL system architecture, interface, and functionality, comparing it with other existing systems. The comparison focuses on the topics that, to the authors' knowledge, are not addressed by other similar systems. On the other hand, the paper presents a quantitative and qualitative evaluation of LearnSQL that has been elaborated over the eight academic years of use in all the database courses of the Facultat d'Informàtica de Barcelona (FIB) of the Universitat Politècnica de Catalunya—Barcelona Tech (UPC).

As an e-learning tool, LearnSQL encourages learning because it provides students with exercises to practice and learn the topics that are studied in the course. Teachers prepare tests composed of several exercises that are to be solved during face-to-face laboratory classes and other exercises that have to be solved at home. In laboratory classes, the students get used to the learning environment. They solve exercises thanks to the timely and immediate feedback provided by LearnSQL and, if necessary, with the teachers' advice. In both laboratory classes and at home, the students can make as many submissions as necessary to solve the exercises in order to learn from the LearnSQL feedback. Although students can send their responses to LearnSQL directly by typing the response on a form, they are encouraged to practice and solve the exercises using a DBMS before they send their responses to LearnSQL.

From the assessment perspective, LearnSQL allows the e-assessment of the students' learned skills because it can automatically correct and grade the students' responses. The teachers prepare exam tests using LearnSQL. During the exam, the students solve the exercises using the DBMS. When the students come up with a possible response, they send it to LearnSQL to be corrected. However, the teachers limit the number of submissions during the exams, and a penalty is applied in the case of retries. Thus, the resulting grade is an indication of the students' learned skills.

The rest of the paper is organized as follows. Section 2 presents the LearnSQL system, its architecture, its software quality aspects (which make it a suitable e-assessment system), and the functionalities of the system that support the learning of relational database skills. Section 3 presents the quantitative and qualitative results of the use of LearnSQL. Section 4 presents the lessons learned by the teachers after eight years of using the system. Finally, Section 5 outlines the conclusions.

2. The LearnSQL system

This section analyzes the LearnSQL system from both a technological and educational perspective. Subsection 2.1 describes the system architecture of LearnSQL and discusses its non-functional quality characteristics. Subsection 2.2 describes the functionalities of LearnSQL as an e-assessment system [26, 27] and compares it with other systems that pursue similar functionalities. Finally, Subsection 2.3 describes the interaction of the system with teachers that define exercises and with students that solve them.

2.1 System architecture

The architecture of LearnSQL, composed of four subsystems (see Fig. 1), was planned in conformance with the IMS QTI specification [28].

- The Authoring Tool (AT) is used by teachers of database courses to create and manage exercises to be solved by students. These exercises are stored in the Items Bank database. The AT is currently implemented as a desktop application that connects to the remote Items Bank database.
- The Remote Test Module (RTM) is used by students to solve exercises. This module also provides functionalities to teachers, such as grouping exercises in tests, monitoring students' progress, publishing grades, and giving feedback to the students, etc. The RTM is implemented as

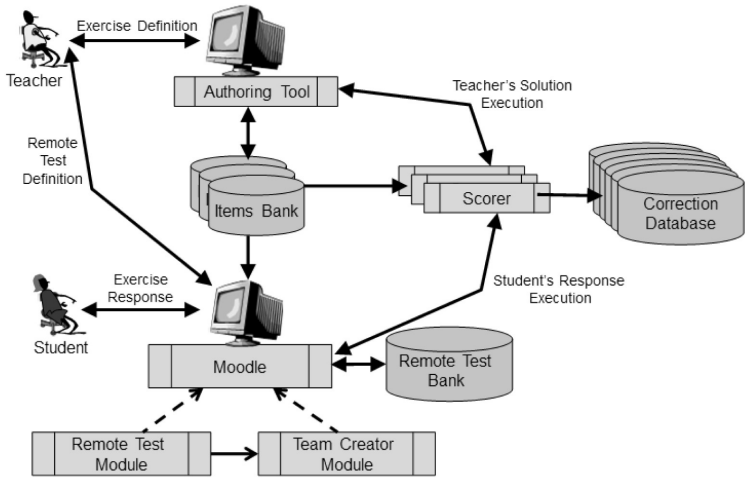


Fig. 1. LearnSQL system architecture.

an activity module that extends Moodle 1.9. Moodle provides the LearnSQL system with the Learning Management System (LMS) infrastructure to manage courses, groups, students, teachers, etc.

- The Team Creator Module (TCM) is used by teachers and students to create teams of students. LearnSQL provides the possibility of solving exercises collaboratively [29] and to interact with the system as a team. When a test is created the teacher may decide whether it must be solved individually or by teams. If the exercises are to be solved by teams, the responses and grades of the exercises are associated to all the team members. Even though teams may be composed of two or more students, the database teachers find that teams of two students are the most appropriate. As in the case of RTM, the TCM is implemented as another module that extends Moodle.
- The Scorer (SCR) is a web service that corrects exercise responses. This scorer is required by the AT when defining of new exercises. It is also required by the RTM during the correction stage of the students' responses. The SCR configuration allows setting the specific DBMS that is used in the course (the DBMS used to date are PostgreSQL, Oracle, and SQL Server), in which the exercise responses will be corrected. The SCR is currently implemented as a SOAP web service.

The four LearnSQL subsystems are loosely coupled. This aspect is especially important for the SCR web service, which is the most critical subsystem of LearnSQL. Thanks to the system architecture selected, this subsystem does not require any maintenance unless the correction of new types of exercises is added to the system.

The ISO 25010 standard [30] proposes the following non-functional characteristics to be taken into account in the evaluation of the quality of a software system: Usability, Reliability, Performance efficiency, Security, Compatibility, Maintainability, and Portability. All of these characteristics were taken into account in the implementation of LearnSQL.

Usability. Usability is successfully achieved taking into account that students learn how to use the LearnSQL system in just one hour of laboratory class. The positive opinions of the students regarding satisfaction are shown in the results of the surveys conducted by the authors at the end of each course (see Subsection 3.2).

Reliability. The system is available 24×7, and the server administrators have provided mechanisms to re-establish the service in case of interruption or failure. The authors' experience shows that the service rarely has to be re-established due to a connection break with the databases (once or twice a year at most).

Performance efficiency. The efficiency of LearnSQL is particularly critical during the correction of exercises and influenced the implementation of the SRC (e.g., the SRC uses connection pools to the Item Banks and to the Correction Databases). In Subsection 3.1, the authors show this efficiency by presenting the average time needed by the SCR to correct exercises.

Security. There are two different threats that must be taken into account: the security of the different subsystems, and the security of their communications with each other. With regard to the subsystems, the Moodle login system prevents unauthorized access to the RTM. Specifically, for the SCR, the web service requests require authentication and authorization, and the requests must come from specific IP addresses. The security for access to the AT, the Items Bank database and the Remote Test Bank database is provided by the users

and the privileges defined in the DBMS where these databases are stored (currently, PostgreSQL). With regard to the communications, they are guaranteed thanks to the use of Hypertext Transfer Protocol Secure (HTTPS) and Secure Sockets Layer (SSL).

Compatibility. LearnSQL runs in a shared environment in the FIB laboratory. No problems of coexistence with other systems have been identified in all the years that LearnSQL has been used.

Maintainability. In order to achieve maintainability, teachers with knowledge of the architecture and the design of the system have been involved in the LearnSQL implementation. The result is a well-structured, modifiable, and testable code that can be easily extended to correct new types of exercises without requiring the update of any part of the existing code. Extending LearnSQL to correct exercises of a new topic simply requires implementing a new corrector module that extends the SCR through polymorphic behavior.

Portability. Portability depends on the degree to which LearnSQL can be adapted for different or evolving hardware, software or other operational or usage environments. First, all of the software products used in the LearnSQL implementation platform (Java, php, PostgreSQL, etc.) are portable to different hardware environments. Second, even though all of the software products on the implementation platform are freely available, new versions are constantly being released. The weakest point of LearnSQL is the adaptation of the RTM and the TCM to the continuous new releases of Moodle (some of which introduce important changes in the LMS code structure). Third, the potential scalability, thanks to the possibility of replicating the SCR and the Items Bank (that it accesses), gives LearnSQL adaptability to environment changes that require more capacity in term of the number of students to be supported.

2.2 Functional aspects

This section describes the functionalities of LearnSQL as an e-assessment system [4, 31]. It also compares these functionalities with the ones offered by other similar systems identified using the snowball searching method [32]. The key paper to start our snowballing search was the one that presents the ADVICE system [21]. Since not all of the existing systems have related publications that describe them, relational database e-assessment open systems were also considered. Finally, the authors obtained a list of 24 possible systems [33]. From this list, 6 were chosen (see Table 1) to be compared with LearnSQL. The selection was made based on the following criteria: the general functionalities provided as learning environments (e.g., supported learning strategies, learners' progress monitoring), the relational database topics that the system helps to practice and assess, the existence of scientific publications that describe the system and the current availability of the system.

2.2.1 General functionalities

The general functionalities provided by an e-assessment system can be grouped into nine categories (Table 1).

Learning Environment. This is an environment that allows the following to be performed: manage enrolled students, control environment accesses, provide course material and activities to students, manage a gradebook, etc. The LearnSQL RTM is implemented as a new activity of Moodle. Thus, Moodle provides LearnSQL with its learning environment, and the RTM extends this environment to define tests that are composed of exercises that were previously created by means of the AT. The systems similar to LearnSQL that were chosen do not extend an existing widely known LMS; instead, they provide their own learning environment.

Collaborative Learning. Collaboration is possible when students can work on the same exercises as a team. LearnSQL uses the TCM to create teams of students. Thanks to this module, tests can be assigned to be solved by individual students or by teams. When tests are assigned to teams, the teammates do not need to be in the same physical location to solve the exercises; they can interact through LearnSQL in order to collaborate in the resolution of the exercises. Of the related systems chosen, only the ACME system allows student teams to collaborate with each other to solve exercises as LearnSQL does.

Progress Monitoring. Monitoring exists when the teacher is able to follow the advances in the learning progress of the students. In LearnSQL, teachers have access to the exercise responses submitted by the students and teams, the feedback given by LearnSQL to the students, and the grade assigned to each response submitted. Most related systems provide similar monitoring progress functionality, except SQL-LTM, which does not mention this issue.

Adaptive Learning. This learning is provided when a system allows the assignment of exercises that are adapted to the students' learning progress. LearnSQL does not currently provide adaptive learning, nor do most of the related systems. Only Tutor and Pahl & Kenny systems do provide this functionality.

Correction Approach. In a system where exercises are automatically corrected, it is important to know the correction approach that is used by the system in order to determine the reliability of the correction result. For relational database assessment tools,

there exists different approaches for automatically correcting exercises (see Appendix A). LearnSQL is multiple test cases based (see Table 1). The teacher defines several test cases for each exercise, and just one of them is provided to the students (named public test case). The full set of tests cases is checked through the AT. The advantage of this approach is that since each test case checks a certain aspect of the student's response (e.g., joins are missing), the feedback message associated to failing that test is specific for the error of the student's response and may help and motivate the student to improve his/her response. The drawback of the correction approaches based on tests is the difficulty in the evaluation of quality in students' responses (e.g., how to identify responses that include redundancies). In order to give feedback about the quality of responses the teachers make a manual post-analysis of the students' responses. This manual analysis does not consume a lot of time, considering the number of students per course and that the post-analysis is performed once the system correction is finished. This analysis needs also to be carried out by the teacher to be able to provide accurate feedback [6] to the students during face-to-face classes. In general, the different types of feedback contribute to the engagement of the students and to their motivation to continue the learning process [4, 34, 35].

Correction System. In systems where exercises are automatically corrected (and specifically in systems that correct programs or codes exercises), it is important to know the specific platform on which these codes are going to be corrected. In relational database learning tools, this platform is a DBMS. Nowadays, there are many DBMS (both commercial and open source) and their databases languages and features may vary from one another (e.g., specific variants of the SQL standard or specific aspects in the physical design). LearnSQL allows the correction of exercises in different DBMS (stated in the SCR configuration). This is also the case for the Stanford Online BD-Course (currently correcting exercises on SQLite) and ADVICE. Other systems, such as TUTOR and SQL-LTM, only analyze responses regarding defined constraints or rules that only conform to the standard SQL and not to specific DBMS.

Grading Approach. E-assessment systems that have the possibility of grading the responses can use different grading approaches. In LearnSQL, grades are numeric from 0 to 10 (0 being the lowest grade and 10 the highest). The grading polices supported by LearnSQL are the following:

Table 1. General functionalities

	LearnSQL	**ADVICE [21]**	**ACME [36]**	**TUTOR [37]**	**Pahl&Kenny [38]**	**SQL-LTM [39]**	**DB-Course [40]**
Learning Environment	Moodle	ADVICE	ACME	TUTOR	Proprietary Environment	AEQ	Coursera Stanford Online
Collaborative Learning	Team Support	NO	YES	NO	NO	NO	NO
Progress Monitoring	YES	YES	YES	YES	YES	–	YES
Adaptive Learning	NO	NO	NO	YES	YES	NO	NO
Correction Approach[1]	Multiple test cases – Quality validation	Single test case	Single test case	Constraints	Syntactical analysis after response normalization	Syntactical – Transformations	Single test case
Correction System	Configurable	Configurable	Not Reported	NO	Oracle	NO[2]	SQLite
Grading Approach	Binary/Real/ Public Test Case	Binary	Binary	–	Real	–	Binary
Feedback	Clues for failed test cases	Differences in execution results	Differences in execution results	Clues based on constraints violated	Error Types identified	Error type and subtype identified	Differences in execution results
Retries	YES	YES	YES	YES	YES	YES	YES

[1] Existent correction approaches are described in Appendix A.
[2] It allows trials on several DBMS; however, corrections are not made for a specific DBMS.

Binary, the grade is 10 (all test cases passed successfully) or 0 (not all test cases passed successfully); Real, the grade corresponds to the sum of the weights of test cases passed successfully; or Public Test Case, the grade is the Real grade, only if the public test case is passed successfully. The related systems that grade the correction result offer binary grading; only the Pahl & Kenny system uses a real grading policy based on the syntactical rules successfully passed.

Generated Feedback. As stated above, the feedback to the students' responses is very important in facilitating learning and in motivating and engaging students. There are studies based on the different message categories that e-assessment systems can use to provide feedback and the effect of these messages on the learning interests of the students [41]. LearnSQL provides clues for each test case that has not been passed successfully (see Subsection 2.3 and Appendix B where examples of these clues are shown), offering possible reasons for why the test case failed. Systems that use a single test case based correction approach may not distinguish the reasons for the test to fail, since it has to be designed to test any possible error of student's response. These systems usually give feedback based on showing the difference between the result of the execution of student's response versus the one obtained executing the teacher's solution. Systems that use other correction approaches give information about the type of error found. In these systems, the most important drawback is that error messages are not specific to one exercise but are general error messages.

Possibility of Retries. After receiving feedback, e-assessment systems allow students to make several submissions (retries) to resolve the exercises. If a system is used for grading, an interesting system feature for the teacher is to be able to define penalties when the student retries sending a new response. In LearnSQL, teachers can define how many times a student or team can retry a specific exercise, and they can also define a penalty to be applied based on the number of retries when the exercise is used in an exam. Moreover, LearnSQL supports different retry grading policies: the grade of the exercise can be the higher grade obtained in the different submissions, or the grade obtained in the last submission. All of the related systems studied offer the possibility of retries.

2.2.2 E-assessment functionalities of relational database skills

The skills that a relational databases learning system may help to learn and assess appear in rows of Table 2. It is important to note that the authors consider that a system is able to correct exercise responses when it does an analysis or execution of the response. Therefore, if one of the related systems compared with LearnSQL only provides students with quizzes to assess knowledge

Table 2. Relational database skills e-assessment functionalities

Skills	LearnSQL	ADVICE	ACME	TUTOR	Pahl & Kenny	SQL-LTM	DB-Course
SQL Queries	YES	YES	YES	YES	YES	YES	YES
SQL Ins/Del/Upd	YES	YES	NO	NO	NO	NO	YES
DDL	YES	Limited	NO	NO	NO	NO	NO
Conceptual Modeling	NO	YES	YES	KERMIT [42]	NO	NO	NO
Logical Design	YES	YES	YES	NO	NO	NO	NO
Normalization	YES	YES	YES	NORMIT [43]	NO	NO	NO
Relational Algebra	YES	YES	YES	NO	NO	NO	YES
SQL/PSM	YES	NO	NO	NO	NO	NO	NO
Triggers	YES	NO	NO	NO	NO	NO	YES
Multi-dimensional Operations	YES	NO	NO	NO	NO	NO	NO
Programming with SQL	YES	NO	NO	NO	NO	NO	NO
Optimization using Mat. Views	YES	NO	NO	NO	NO	NO	NO
Optimization using Indexes	YES	NO	NO	NO	NO	NO	NO
Optimization of Workloads	YES	NO	NO	NO	NO	NO	NO
Cost Estimation	YES	NO	NO	NO	NO	NO	NO

of a specific topic/skill, a "NO" is stated in the corresponding cell of the table.

SQL Queries. Students should be able to write query sentences to retrieve certain data from a database. The exercises that support learning this skill ask students to write SQL sentences that obtain specific data from a database (i.e., SELECT sentence). This skill is included in practically all of the relational database assessment systems, and specifically in all of the ones that were selected for the comparison.

SQL Ins/Del/Upd. Students should be able to write modification sentences to change the data stored in a database. The exercises that support learning this skill ask students to write SQL sentences that add, remove and/or update specific data from a database (i.e., INSERT, DELETE and/or UPDATE sentences). Taking into account that these sentences may violate integrity constraints in the database, other exercises ask students to write SQL sentences that violate certain integrity constraints. LearnSQL is able to correct both types of exercises. Aside from LearnSQL, only ADVICE and the DB-Course in Coursera and Standford Online can correct these types of exercises.

DDL. Students should be able to create the structure and other components of a database. The exercises that support learning this skill asks students to write SQL sentences that create or modify a database schema (e.g., CREATE and ALTER sentences of structures such as tables, views or indexes). Aside from LearnSQL, only the ADVICE system can correct exercises to practice this skill. However, ADVICE is limited to the assessment of exercises to create tables and does not support exercises to create other database structures.

Conceptual Modelling. The design of a database begins creating the conceptual model of the objects and associations in the real world that have to be represented in the database. Students should learn how to create these conceptual models for a specific database. The exercises that support learning this skill ask students to make a model in a conceptual modelling language (e.g., UML or ER). LearnSQL is not able to correct this type of exercises. This is the only limitation of LearnSQL when compared to related systems. The reason for this is that conceptual modelling is not included in the databases courses at the authors' university because it is not a task that is specific to databases development (it is taught in general software engineering courses).

Logical Design. The logical design of a database is obtained by translating the conceptual model to obtain the database schema. Although tools exist that can do this translation automatically, it is advisable for students to practice how to do it themselves. The exercises that support learning this skill ask students to give the structure of a database from a conceptual model in UML or ER notation. LearnSQL, ADVICE and ACME can correct this type of exercises but require the response of the exercise to be written in different languages (i.e., as SQL CREATE TABLE sentences in case of LearnSQL and ADVICE and relational schema notation in case of ACME).

Normalization. The structure or logical design of a relational database must follow certain well-established normalization rules. Students should learn and practice these rules, identify when the rules are not satisfied, and know how to transform a database schema to satisfy them. The exercises that support learning this skill ask students to give the new structure of a database that satisfies the normalization rules starting from a database structure that does not satisfy them. The systems that correct logical design exercises also support the correction of normalization exercises. Additionally, the system NORMIT that is an extension of TUTOR also supports it.

Relational Algebra. Relational databases are based on relational algebra since the idea behind the model is that tables in a database are sets of rows. Students should learn how to write query sentences in relational algebra, not only to better comprehend the relational database model, but also because DBMS use relational algebra to optimize the SQL queries. This would help students understand how a DBMS would solve a certain query. The exercises that support learning this skill ask students to write sequences of algebraic operations that obtain specific data from a database. This skill is included in four systems, including LearnSQL. Since relational algebra does not have a standard user notation, all of the systems have different notations to write the algebra operations. In LearnSQL the correction of this type of exercise requires of previously running a compiler that translates relational algebra expressions to SQL.

SQL/PSM. In relational databases, it is possible to define functions (called Stored Procedures) that are managed and executed by the DBMS. One advantage of these functions is that they can encapsulate certain management of the database data that must be used from different parts of a software system or different systems. Stored procedures have several advantages such as: (1) they make it possible for developers to ignore the internal logic of the function; (2) the DBMS can optimize the functions; and (3) the interactions between the function and the database have a smaller cost than if these interactions were done from an external function outside the database. Students should practice programming SQL/PSM functions. The exercises that

support learning this skill describe what the functions must do. The students' responses must be written in a language that is supported by the DBMS where the exercise is corrected (e.g., in PostgreSQL the responses are required in PL/pgSQL language). LearnSQL is the only system that includes exercises on this topic.

Triggers. In relational databases, it is possible to define triggers, which are Event-Condition-Action rules that can be activated by the execution of an INSERT/DELETE/UPDATE sentence on a certain table of a database. Triggers give the opportunity to define implicit actions to be done when an event occurs. They are typically used to track the occurrence of certain events in the database, to change derived data when basic data is modified, or to define integrity constraints that have to be checked every time an event occurs. In these cases, the definition of a trigger avoids replicating this behavior every time the events occur, and developers can also ignore the existence of the trigger definition and the collateral effects of executing a certain modification sentence. Students should practice programming triggers. The exercises that support learning this skill ask students to write one or more triggers from a description of the behavior that they must implement. This skill is only included in LearnSQL and in the system used in the Stanford Online DB-Course.

Multidimensional Operations. A full introduction to databases should not only cover transactional systems but also decisional systems. In this context, a conceptual model can be presented as a multidimensional view of the real world, and a multidimensional algebra can be used as the means to query the model and obtain the required aggregated information LearnSQL is the only system that supports this skill. It includes two types of exercises that support learning this skill: 1) exercises in which students have to build an SQL query corresponding to a sequence of multidimensional algebraic operations (i.e., assuming the ROLAP approach and star-join representation); 2) exercises in which the response must be a query that address the relational database directly and uses SQL-99 constructs (e.g., ROLLUP/ CUBE/ GROUPING) to obtain aggregated information from a set of tables.

Programming with SQL. Relational databases are commonly accessed from programs by means of certain programming API (Application Programming Interface), such as JDBC (Java Database Connectivity) or ODBC (Open Database Connectivity). Students should practice using these API. The exercises that support learning this skill ask students to write a program that accesses the database to implement a specific functionality. This type of exercise is only included in LearnSQL. Currently, the corrector is prepared to ask students to program in Java and the JDBC API, but the programming language could be changed by changing the scorer configuration.

Optimization using Materialized views. A way to reduce the cost of SQL queries is the use of materialized views as partial aggregation repositories, especially for multidimensional-related queries. Students must learn and practice the conditions for query rewriting and how to choose the materialized views that are best suited for a given context. No other system but LearnSQL was found to support this skill.

Optimization using Indexes. This is a very important skill that students must acquire in order to perform well as database designers or administrators. They must learn the most common used kinds of indexes and know which one is the most appropriate given a specific context. In the exercises this context consists usually in a simple situation of one SQL query over a small number of tables. LearnSQL provides support for this skill by evaluating the index creation DDL provided by the student in response to the context described by the statement. No other system covers this skill.

Optimization of Workloads. Students must be also prepared to solve complex situations where a workload of several SQL queries involving several tables must be globally optimized. LearnSQL is the only system that covers exercises that require students to be able to combine several data structures on several tables to obtain the best global cost for a given workload under certain constraints (e.g., a maximum threshold of disk space to be used). Currently this kind of exercises supports the creation of indexes, clustered tables and materialized views.

Cost Estimation. Given a SQL query, students must be able to compute an estimation of the cost of any possible access plan. In exercises that support learning this skill, the statement gives a number of tables and indexes as inputs to a graphically represented process tree. The nodes of the tree are physical operations like selections and joins. As the response the students must provide the cost of the best access plan corresponding to the process tree computed using certain formulas. Only LearnSQL covers this skill.

2.3 Use of LearnSQL

This section presents a description of the steps that a teacher follows when he/she defines an exercise and creates a test. It also describes how the exercises are presented to the students.

First of all, the teacher defines the general attributes of an exercise. The concepts in the definition of an exercise are the same independently of the type of exercise (see a simplified UML diagram that repre-

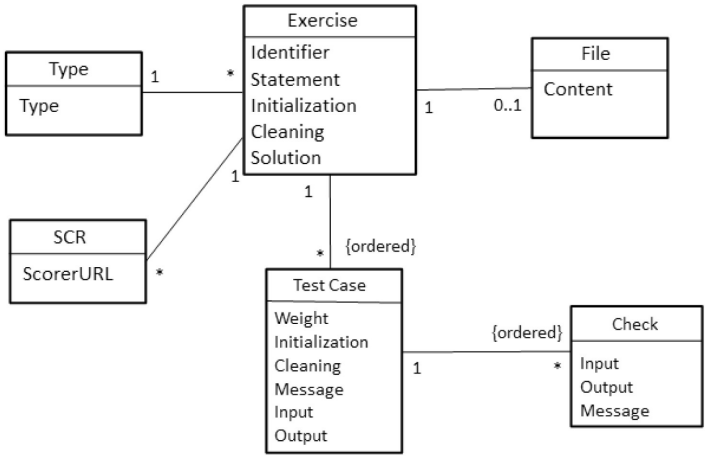


Fig. 2. Concepts in the LearnSQL Item Bank.

sents these concepts in Fig. 2). The most relevant are the following: the exercise *Statement* in natural language, the *Initialization* and *Cleaning* sentences (SQL CREATE sentences that establish the database schema for the exercise and DROP sentences that clean the database after the correction of the exercise), a *File* provided to the student that contains different material depending on the type of exercise, and the SCR associated to the exercise.

Next, the teacher introduces the different *Test Cases* that will be tested during the correction. Each test case corresponds to a different state of the database that is relevant for the exercise and has a *Weight* that is assigned by the teacher based on the total weight of the exercise. In the definition of each test case it is necessary to define the *Initialization* and *Cleaning* sentences (SQL INSERT sentences that establish the state of the database corresponding to the test case and DELETE sentences that clean the state). In some types of exercises (e.g., SQL Ins/Del/Upd exercises), the teacher needs to define one or more *Checks* for each test case in order to query the state of the database after the changes produced by the execution of the student's response. The *Input* of the check establishes a query to be executed; for instance, for the SQL Ins/Del/Upd exercises that modify the content of the database, the queries defined for each input will check the content of the tables after the modifications. The teacher is responsible for the introduction of the *Message* associated to each test case that will be the feedback provided to the student if the test case or check is not passed successfully. In some types of exercises (e.g., Triggers or SQL/PSM exercises), the teacher also needs to define an *Input* for each test case; for instance, for the Trigger exercises, *Input* correspond to SQL sentences that cause the execution of the trigger defined in the exercise responses.

To finish the definition of one exercise the teacher introduces a correct *Solution* (i.e., the teachers' solution) for the exercise. The solution is executed by the SCR associated to the exercise (the SCR configuration states the DBMS and database where it is executed) to generate correct *Outputs* for each *Test Case* and each *Check*.

After one or more exercises are defined, the teacher groups the exercises in a remote test. During the definition of a test the teacher states the position of each exercise in the test and its weight with regard the grade of the test.

Figure 3 shows the presentation of an exercise to the students. The students see: the weight of the exercise in the test (in the example, 30% of the total grade of the test); the statement that defines what the student is required to do (write an SQL sentence); the result of executing the teacher's solution for the public test case provided in the attached file (the teacher Toni); a text area to enter the student's response; the "Submit" button to correct the response and obtain feedback; the link to a pop-up window with the student's response that has been graded; and the current grade, the number of the next submission, and the total number of submissions allowed (grade for the submission is 8, if the student retries the exercise next submission will be the second one, 10 submissions allowed). Finally, the feedback of the correction process of the last response that has been submitted ("TC5: you have probably missed a join", therefore just test case TC5 failed).

The correction approach of LearnSQL is multiple test case based (as stated in Subsection 2.2.1), but not all type of exercises are corrected in the same way. By means of a polymorphic implementation of the SCR, the skill assessed by an exercise determines which one of the five different correctors is used (see Table 3). Appendix B describes how LearnSQL corrects students' responses depending on the type of exercise. Specifically, examples of feedback provided to the students are included in order to show

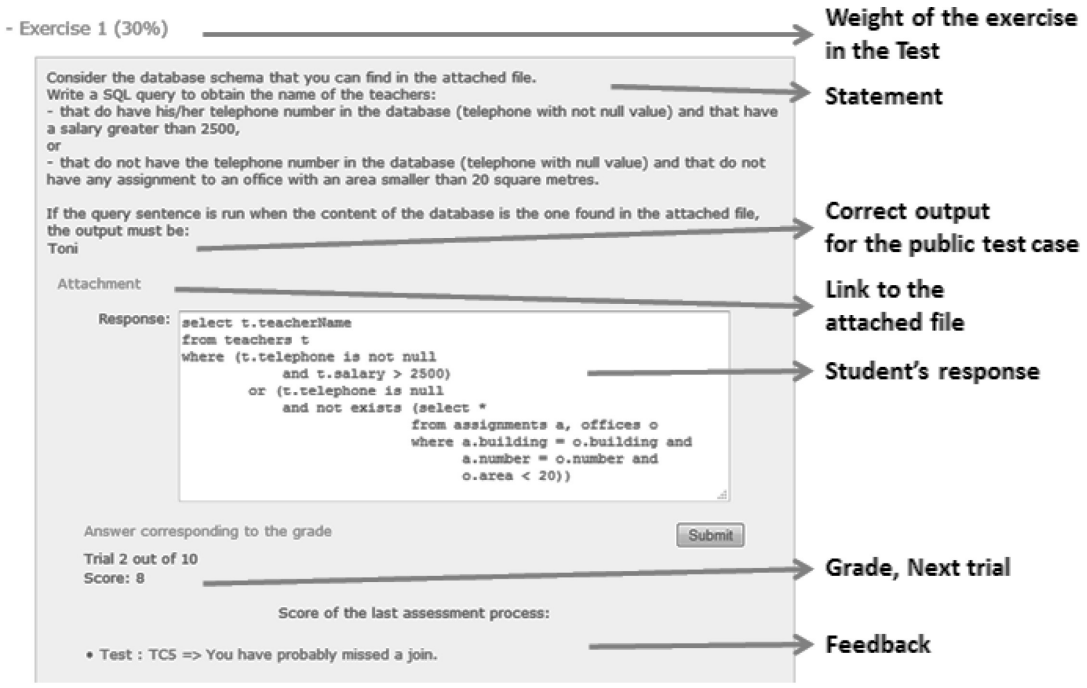


Fig. 3. SQL Query exercise example.

Table 3. LearnSQL correctors

LearnSQL Correctors	Skills
Corrector 1	SQL Queries SQL Ins/Del/Upd DDL Logical Design Normalization Relational Algebra Multidimensional Operations Optimization using Materialized views
Corrector 2	SQL/PSM Triggers
Corrector 3	Programming with SQL (JDBC)
Corrector 4	Optimization using Indexes Optimization of Workloads
Corrector 5	Cost Estimation

how the SCR messages can help students to improve their responses and therefore to learn.

3. Results

This section presents quantitative and qualitative analyses of the results of the use of LearnSQL. These results show the successful adoption of the system from the point of view of the results and of the satisfaction of the students.

3.1 Quantitative Analysis

In this subsection, the authors include data that was obtained from the use of the system during 10 semesters; from the autumn semester of 2009 (0910AS) when the system was completely deployed in all database courses and for all types of exercises to the spring semester of 2014 (1314SS). The goal of this subsection is to show the technical quality of the system and the capacity of the system to help students acquire relational database skills.

The first aspect analyzed was the number of submissions to the system during the 10 semesters. This can be seen in Table 4, where the number of submissions is distributed by type of exercise. In this table, the number of submissions is different depending on the type of exercise. The reason for this is that there are types of exercises used in courses with fewer students, as is the case for Optimization, and Cost Estimation exercises. In other types of exercises, the variation in the number of submissions is caused by the lower number of exercises proposed to students. This is the case for exercises of Programming with SQL, since these exercises take students more time.

Table 4 also includes the number of exercises that are currently defined in the Item Bank distributed by type of exercise, and the average of test cases defined for exercises of each type. As can be observed there are important differences among the number of test cases depending on the type of exercise. Specifically, it is important to justify the exercises for Cost Estimation, which have an average of only one test case; this is because of the type of the correction done for this type of exercise (see Appendix B, Corrector 5).

Table 4. Statistics per type of exercise in 10 semesters

Type of Exercise	Total number of submissions	Number of exercises	Average of test cases per exercise
SQL Queries	42,278	159	5.7
SQL Ins/Del/Upd	9,351	63	3.4
DDL, Logical Design, Normalization, Optimization using Materialized Views	13,330	212	7.8
Relational Algebra	29,371	57	4.4
SQL/PSM, Triggers	29,976	92	5.6
Programming with SQL	6,574	61	8.5
Optimization using Indexes, Optimization of Workload	2,034	51	8.6
Cost Estimation	2,385	102	1.2

The second aspect is the average time that it takes LearnSQL to correct one submission. Fig. 4 includes these times by type of exercise. The average response times were obtained for the submissions of the 10 semesters considered. The submissions were processed by just one SCR that accessed just one Item Bank using a pool of 5 connections and accessing 5 Correction Databases.

As can be observed, there is one type of exercise (Programming with SQL) that has a response time that is considerably higher than the rest (between 7 and 8 seconds). The reason for this is that, for each correction of a student's response, the SCR unzips the file that contains the Package where the Java class developed by students has to be included, the Java class has to be compiled, and after, executed requiring a new connection to the Corrections Database. Another type of exercise that has a high average response time is Optimization using Indexes and Optimization of Workloads (around 2 seconds). The reason for this is that, the SCR has to connect and query the student's database (see Appendix B, Corrector 5).

The third aspect analyzed is the number of submissions by semester. In Table 5 the submissions are first classified as Valid and Invalid (columns 2 and 3). Invalid submissions are the ones that include a syntactic error, invalid characters, forbidden sentences, or that submit a response that does not correspond to the type of exercise required; valid submissions are the rest and obtain a grade between 0 and 10. In the same table, the total submissions are classified as having been done in Individual Self-study at home or as Team submissions during face-to-face classes (column 4 and 5). The number of students and teams of students per semester are also included in order to know the number of users of LearnSQL per semester.

Table 5 also shows that there are important differences in the number of submissions per semester (column 6). The reasons for this variation are changes in the number of students accessing FIB studies, changes in the number of students that passed the previous programming courses in the curriculum, and also changes in the FIB curriculum. For instance, changes in the FIB curriculum caused an increase in the number of students (e.g., 263 enrolled students in the autumn semester of 1011AS).

In order to analyze the evolution of the number of submissions during blended learning, the authors did the normalization of the number of Individual Self-study Submissions assuming that each semester had 100 students (see Fig. 5). There are two situa-

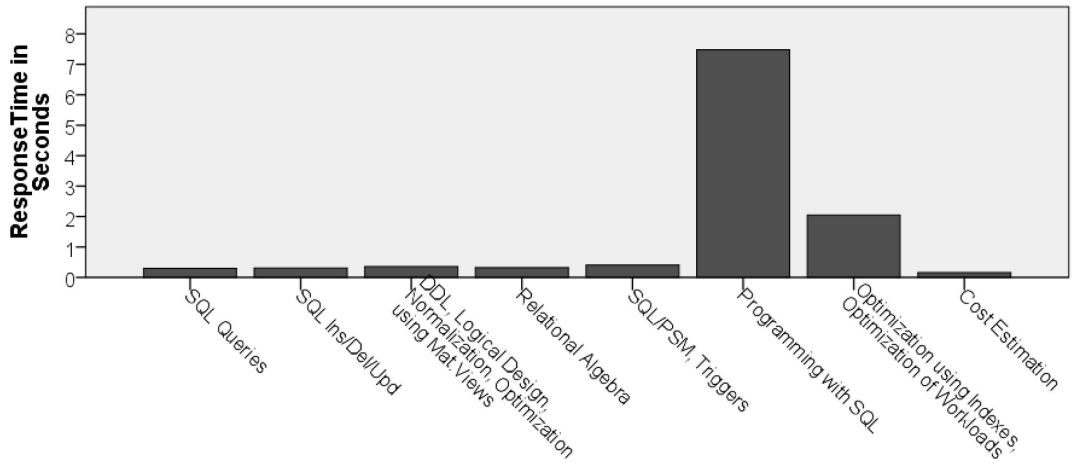


Fig. 4. Average Time that corrections take per type of exercise.

Table 5. Number of submissions and students in 10 semesters

Semester	Total Submissions	Valid Submissions	Invalid Submissions	Individual Self-study Submissions	Team Submissions during Classes	Number of Students	Number of Teams
0910AS	8,595	6,601	1,994	5,926	2,669	149	73
0910SS	11,594	8,649	2,945	8,557	3,037	162	84
1011AS	20,883	15,877	5,006	16,509	4,374	263	137
1011SS	14,711	10,865	3,846	11,148	3,563	178	91
1112AS	11,263	7,871	3,392	7,721	3,542	125	64
1112SS	12,791	8,782	4,009	8,675	4,116	129	68
1213AS	10,641	7,269	3,372	7,167	3,474	116	57
1213SS	11,646	7,768	3,878	8,560	3,086	157	83
1314AS	16,615	11,980	4,635	11,342	5,273	147	72
1314SS	16,560	12,286	4,274	11,023	5,537	158	82

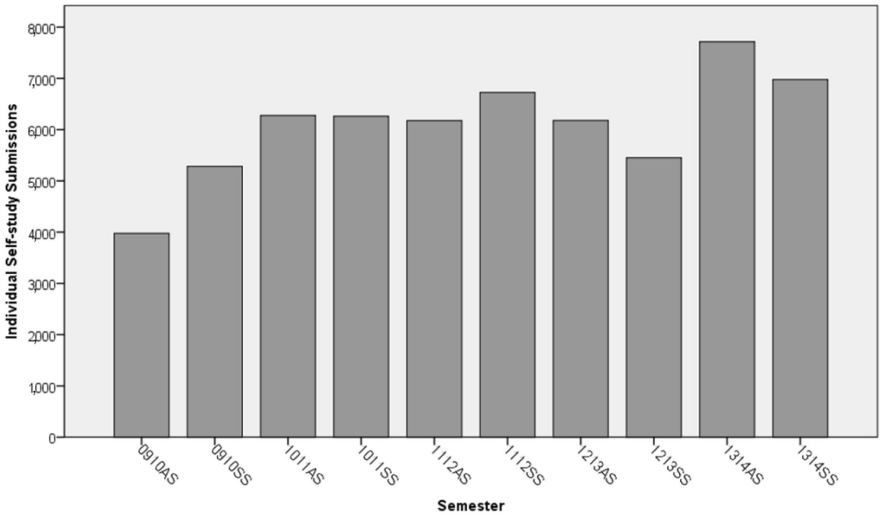


Fig. 5. Number of submissions in 10 semesters per 100 students.

tions that need to be noted. The first one is the progressive increase in submissions in the first semesters considered. This increase is mainly due to the fact that blended learning was introduced in the FIB studies during those semesters. This means that before semester 1011AS, the teachers allowed LearnSQL to be used from home, but it was not encouraged and no exercise was proposed to be solved from home. The second situation is the increase in the use of the system that was observed the last two semesters. The authors think that the reason is the increase in the last few years of the access grade for students entering FIB school, specifically for students that enrolled from autumn semester of 2013–14 (1314AS).

The following analysis demonstrates that the system helps students to acquire relational database skills. Fig. 6 shows the improvement in the students' responses submitted to LearnSQL for each semester. In order to show this increase, the authors obtained the average grade of the first submission and the average grade of last submission for each exercise. The grading system of our country grades from 0 to 10. The possible penalty for retries is not included in the grades. It can be observed that, with the exception of the first semesters (when students were less used to automatic correction systems) there are no relevant differences. In the latter semesters, the average grade for the first submission ranges between 5 and 6, and the average grade for the last submission ranges between 7.5 and 8.2. Consequently, it can be concluded that the students' responses improve from the first submission to the last one for a specific exercise, and that the feedback offered by LearnSQL is useful to students.

Finally, in order to know if the grades assigned by the LearnSQL system have a correlation with the grades that students obtain for exams, the correlation of the average of the grade obtained by students in LearnSQL exams and the grade obtained by the same students in the non-LearnSQL written exam was calculated. In the non-LearnSQL exam, students have no access to any kind of material and have exercises that are similar to the ones practiced in LearnSQL. The analysis was done using the whole population of 1584 students. The null hypothesis to revoke is that there is no correlation between the grades of students in the two exams. Table 6 shows the results obtained from doing the bivariate Pearson correlation analysis. The result is

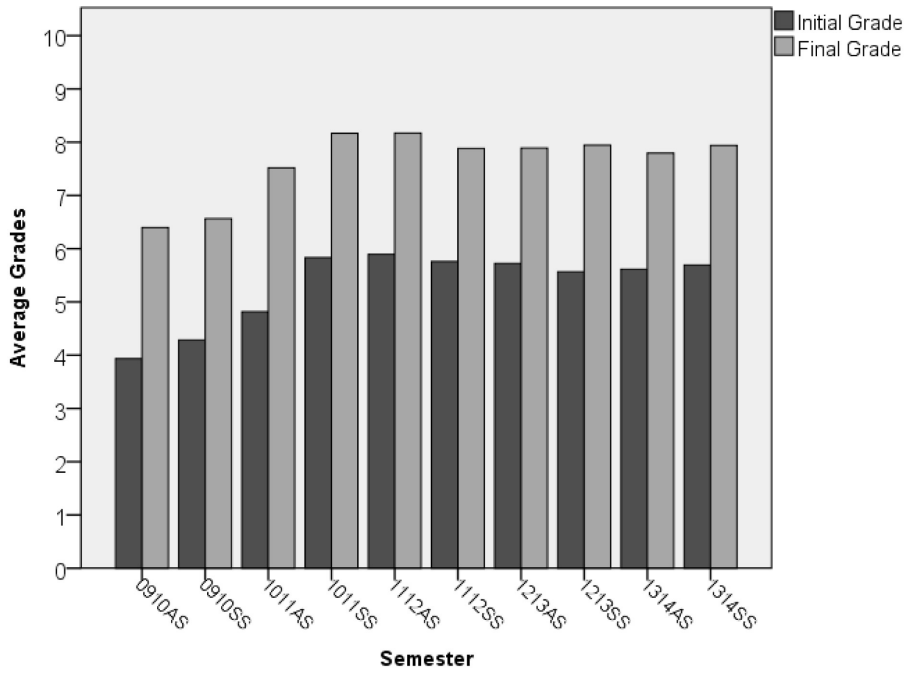


Fig. 6. Average grades for first/last submission for 10 semesters.

Table 6. Correlation result of LearnSQL versus non-LearnSQL exam grades

		Non-LearnSQL Exam	Average Grades Of LearnSQL Exams
Non-LearnSQL Exam	Pearson Correlation	1	0.471*
	Sig. (2-tailed)		0.000
	N	1,584	1,584
Average Grades Of LearnSQL Exams	Pearson Correlation	0.471*	1
	Sig. (2-tailed)	0.000	
	N	1,584	1,584

* Correlation is significant at the 0.01 level (2-tailed).

that there is a Pearson correlation of 0.471 based on the whole population with a level of significance of 0.01, meaning that the null hypothesis is revoked and that there exists a significant correlation.

The scatter plot of Fig. 7 shows graphically the relationship between the two grade variables. As the pattern of dots indicates, the graphic shows the positive correlation identified in the Pearson correlation analysis. The line of best fit or trend line is also drawn in Fig. 7 and shows (as expected) a trend of students obtaining better grades on LearnSQL exams than on non-LearnSQL exams. The reason is that the feedback and retries of LearnSQL exams help the students to improve their grades.

3.2 Qualitative analysis

A qualitative analysis was also carried out from data obtained from the autumn semester of academic year 2009–10 to the spring of academic year 2013–14. The data correspond to the results of a survey questionnaire that was completed by the students since the LearnSQL system was introduced in databases courses of UPC.

The survey given to the students is traditionally completed during the last class of the database courses, with the goal being to measure their satisfaction with their learning experience using LearnSQL. Four statements are proposed to the students.

- Statement 1: The feedback provided is useful and motivates me to analyze my errors.
- Statement 2: The option to retry during exams helps me to improve the grade.
- Statement 3: Having the tool available online helps me learn.
- Statement 4: LearnSQL is a good system for learning relational database subjects.

For each statement, the authors ask the students to state their agreement following a Likert-type

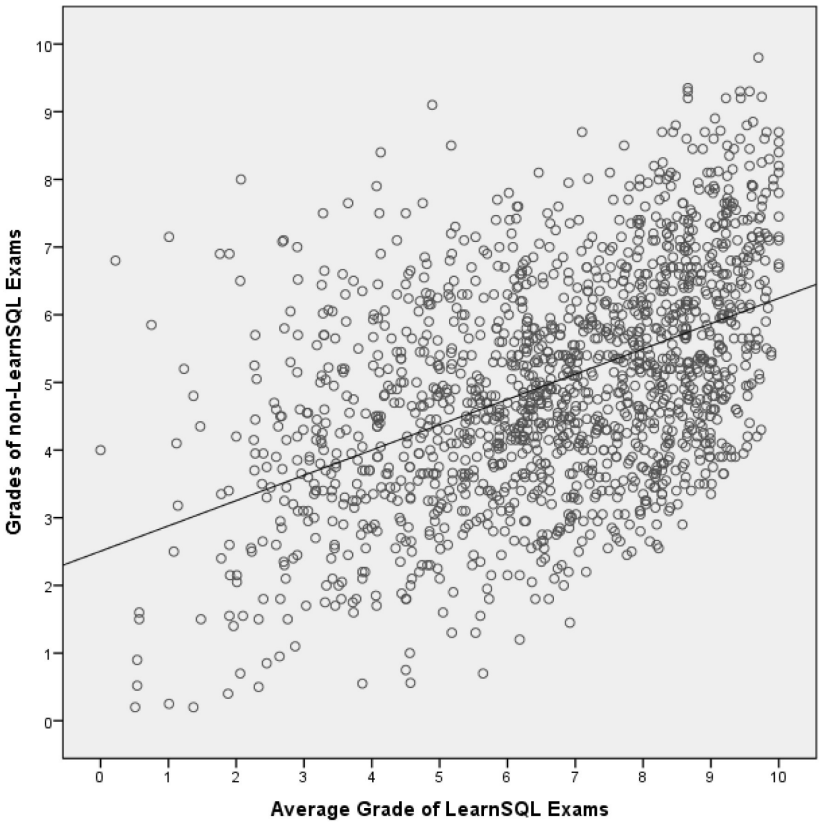


Fig. 7. Scatter Plot between the grades of exams.

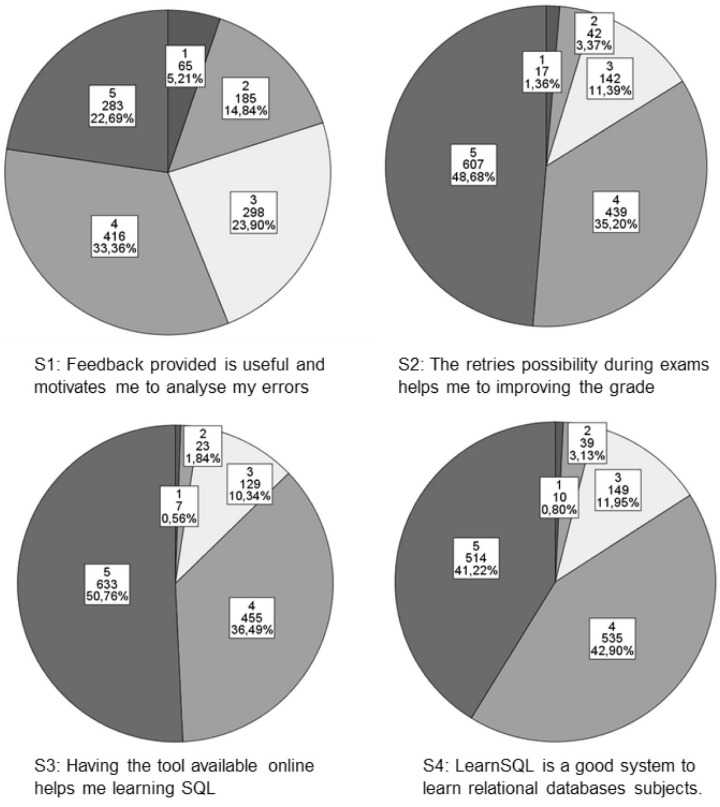


Fig. 8. Survey results.

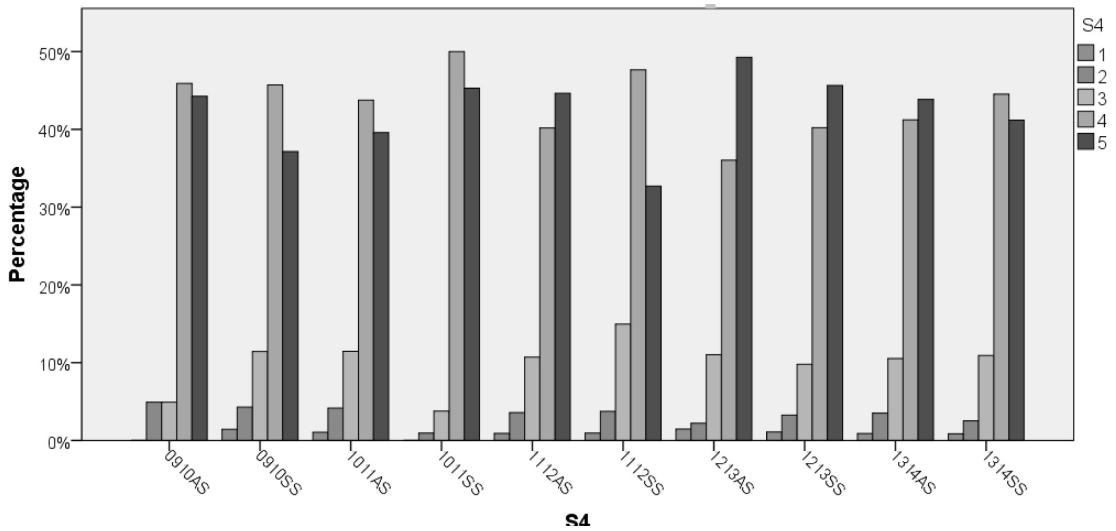


Fig. 9. The evolution of survey results for Statement 4.

scale (1 stands for maximum disagreement, 5 for maximum agreement) [44]. Fig. 8 shows for each statement, the number of responses for each value on the scale and the percentage this number represents in relation to the total number of answers. The authors collected 1013 valid answers from students between 2009 and 2014.

In Statements 2, 3 and 4 more than 80% of the students indicated a level of agreement of 4 or 5, and less than 5% of the students indicated a level below 3. Based on the results for Statement 1, which is related to the helpfulness of feedback, the authors consider it to be the weakest point of the system. However, the results are not so bad since 55% of the students indicated a level of agreement for the usefulness greater than 4, and only 20% thought that the feedback provided by LearnSQL was not useful for them or did not motivate them to solve the errors in the exercises.

In order to show that these results are maintained throughout the semesters, Fig. 9 shows the evolution of the percentage of each agreement level for Statement 4 per semester. As can be observed, most of the students agreed or totally agreed that LearnSQL is a good tool for learning relational database subjects.

4. Discussion

This section explains the main advantages and drawbacks of LearnSQL from the teachers' and students' point of view. This information is based on the results of the qualitative and quantitative analyses presented in Section 3 and on the authors' opinions as teachers of the FIB database courses. The authors also provide a set of lessons learned based on their experience that may be helpful to teachers considering introducing e-assessment in their courses.

From the students' perspective, LearnSQL is a useful system for learning relational database topics and for helping them in their learning process. The system can be used during autonomous learning periods because it provides feedback and grading in real time. It is also a useful tool for automatically assessing the students' expertise in certain relational database skills, and their exam grades improve thanks to the feedback and the possibility of retries. Finally, LearnSQL also prevents unfairness since it applies the same evaluation criteria to all answers.

There are two aspects that students do not consider to be very positive. The first one is that during the exams each retry has a penalty. They do not think that this penalty is fair. From the teachers' point of view this penalty is applied to prevent the students (during exams) from finding the correct response to an exercise just by retrying. The second one is that sometimes the feedback provided by the system does not help them find the errors. This is one of the aspects that the authors are working on in order to improve the system.

From the teachers' point of view, LearnSQL has several advantages. One of the most important ones is that the system automatically corrects the responses sent by the students in real time. Although teachers must do a manual post-analysis of these responses to give feedback to the students about some quality aspects of their responses (e.g., redundant joins or unnecessary subqueries), LearnSQL saves a lot of time since the post-analysis is performed once the system correction is finished and the teachers know its result during the post-analysis. Another important advantage of the system is that it provides clues of the types of exercises that students

find the most difficult and that need to be explained further or practiced longer.

One of the main disadvantages of the system is that it requires a lot of work and staff to start. One of the most time-consuming processes consists of populating the Item Bank with new exercises. Creating a new exercise requires a considerable amount of time because the teacher must define a complete set of test cases to cover all the possible mistakes that students might make in their responses. In the authors' experience, this process has been iterative. When an error has been detected in the definition of an exercise, it has been corrected for the next semester.

The authors' experience can be interesting for teachers who are in charge of database courses and are considering extending an existing learning system to introduce the automatic correction of database exercises or developing an automatic corrector. Even though the start-up is complex and time consuming, in the end, the automatic correction process saves a lot of time, especially in massive courses. When considering extending a LMS (e.g., Moodle) to include an automatic correction module, the main advantage is that the LMS already includes basic course management functionalities (e.g., student enrolment or resource and activity management). However, the LMS also has an important drawback, namely that each time a new version of the LMS is released, the automatic corrector must be adapted. This is an important issue to be considered from the beginning. Teachers that want a system that does not require periodic adaptations should choose or implement an independent automatic corrector. Nevertheless, in this case, if teachers want to have course management functionalities, these functionalities must be developed from scratch. The architecture of the system is another important issue. The different subsystems may benefit from a low coupling for maintenance and extension reasons.

Finally, teachers that are considering introducing e-assessment may benefit from the following lessons learned:

- From the authors' point of view, the e-assessment system is very useful because they can quickly determine where students are having trouble, how they are progressing, and how much work they are doing. Based on this information the teachers can make the needed adjustments throughout the course. This information is also used in laboratory classes because teachers can show students the common errors they have made on the different types of exercises.
- Students find LearnSQL to be very useful for autonomous learning and self-assessment.
- Students find it useful for teachers to provide personalized comments on each of their responses during exams. These comments are introduced by using LearnSQL in a manual post-analysis process. In the authors' experience, this is not a very time-consuming process and is well valued by students.
- Computer-based exams introduce difficulties in guaranteeing authorship due to the all-digital nature of e-assessment systems. This aspect is very relevant in automatic grading systems. Exams that are done using LearnSQL require supervised computer rooms or laboratories that help prevent cheating (to ensure that the test is taken by the intended students). This approach has drawbacks since laboratory classrooms are usually small. The authors have tried two alternatives to solve this problem. One solution requires several rooms. In this case, enough staff is needed to handle the examination session in parallel. The second solution requires several sessions. This approach requires different exams, which may have an impact on the fairness of the evaluation process (if one exam is easier than the other).
- Technical problems may arise in computer-based exams. However, in the authors' experience, no exam has ever been cancelled due to technical problems, which demonstrates the maturity of the LearnSQL system.

In summary, the authors think that the LearnSQL system discussed in this paper is very convenient and useful in database subjects. Even though the start-up of LearnSQL could initially require a lot of effort and staff, it is worth to introduce it in database courses because it encourages students in their learning process, it frees up the teachers in the classroom allowing teachers more time to engage students directly. It also allows more meaningful teacher-student interactions thanks to the knowledge provided by student's feedback.

5. Conclusions

This paper has provided a detailed overview of LearnSQL, which is a software tool that allows the automatic and efficient e-learning and e-assessment of relational database skills. LearnSQL has been analyzed both from a technological and educational perspective.

The technical quality in the implementation of LearnSQL has been guaranteed by means of the use of specifications and standards of software quality, as it has been explained in Subsection 2.1. Furthermore, the conducted quantitative analysis (see

Subsection 3.1) supports this assertion regarding technical quality.

The capacity of LearnSQL as an educational tool has been analyzed by comparing LearnSQL with other similar systems (see Subsection 2.2). The comparison has been performed at two levels: first, the general characteristics attached to any e-assessment system have been discussed. Second, the e-assessment functionalities of relational database skills provided by LearnSQL have been analyzed. In addition, the authors have shown several real examples of the use of LearnSQL in Subsection 2.3, as well as the correction techniques applied by the system (see Appendices A and B). It is important to note that the analysis has required an extensive and systematic process of literature review and constitutes a survey of educational tools in the database field. Although this work focuses on the database field, many of presented ideas and findings can be extrapolated to other fields of Computer Engineering, and they can serve as a basis for teachers interested in the development or adoption of this kind of educational software tools in their courses.

The comparative analysis also shows, on the one hand, that the main weakness of LearnSQL is that it does not provide adaptive learning. On the other hand, the main strength of LearnSQL is the broad set of knowledge and practical skills in relational database that it allows to acquire. The impact of LearnSQL and its successful adoption as an educational tool have been proven by means of a quantitative and qualitative analysis (see Section 3) that includes data from the use of the system since it was completely deployed in all database courses. The analysis is complemented with the lessons learned from the use of the system (see Section 4).

As future work, on the one hand, authors plan to include an automatic generator of test cases for each type of exercise in LearnSQL. Extending LearnSQL with this generator will be an important improvement of the system since it will make the work of the teacher even easier. On the other hand, authors want to introduce the possibility of having adaptive learning in the system. This extension could be based on the data that has been collected during the 8 years of use of LearnSQL (that shows the difficulty that students had during the resolution of the exercises) and on the particular topic addressed by each failed test case in an exercise.

Acknowledgements—LearnSQL has been developed with the support of different projects granted by the Generalitat de Catalunya and UPC: 2009MQD 00251, MQD00202.

This work has been partially funded by the Ministerio de Economía y Competitividad, under projects: TIN2013-45303-P-"ICT-FLAG" (Enhancing ICT education through Formative assessment, Learning Analytics and Gamification); TIN2014-52938-C2-2-R-ELASTIC: Especificación, verificación y simulación de modelos para los procesos de servitización de las TIC; TIN2013-44641-P-EOSSAC: Engineering Open Source Services and Apps for the Citizens; TIN2012-38584-C06-01-SKATER: Scenario Knowledge Acquisition by Textual Reading. It has also been partially funded by the Secreteria d'Universitats i Recerca de la Generalitat de Catalunya, under 2014 SGR 1534; and by the Lifelong Learning Program of the European Union, under project 519141-LLP-1-2011-1-ES-KA3-KA3MP—TRAILER: Tagging, Recognition and Acknowledgment of Informal Learning ExpeRience.

References

1. Joint Information Systems Committee UK, Effective Practice with e-Assessment: An overview of technologies, policies and practice in further and higher education, Publications of *JISC*, 2007.
2. Joint Information Systems Committee UK, Effective Assessment in a Digital Age A guide to technology-enhanced assessment and feedback, Publications of *JISC*, 2010.
3. J. Cook and V. Jenkins, Getting Started with e-Assessment, *Report University of Bath*, 2010.
4. J. Bull and C. Mckenna, *Blueprint for Computer-Assisted Assessment (Vol. 2)*, Routledge, Falmer, London, 2004.
5. C. Daly, N. Pachler, Y. Mor and H. Mellar, Exploring formative e-assessment: using case stories and design patterns, *Assessment & Evaluation in Higher Education*, **35**(5), 2010, pp. 619–636.
6. G. Crisp, Interactive E-Assessment—Practical Approaches to Constructing More Sophisticated Online Tasks. *Journal of Learning Design*, **3**(3), 2010, pp. 1–10.
7. J. W. Gikandi, D. Morrowa and N. E. Davis, Online formative assessment in higher education: A review of the literature, *Computers & Education*, **57**(4), 2011, pp. 2333–2351.
8. E. de Bruyn, E. Mostert and A. Schoor, Computer-based Testing—The Ideal Tool to Assess on the Different Levels of Bloom's Taxonomy, in *proceedings of 14th International Conference on Interactive Collaborative Learning (ICL),* Piestany, Slovakia, September 2011, pp. 444–449.
9. S. Gruttmann, D. Böhm and H. Kuchen, E-assessment of Mathematical Proofs: Chances and Challenges for Students and Tutors, in *proceedings of International Conference on Computer Science and Software Engineering (CSSE)*, Wuhan , China, December 2008, vol. 5, pp. 612–615.
10. T. A. Majchrzak and C. A. Usener, Evaluating the Synergies of Integrating E-Assessment and Software Testing, in R. Pooley, J. Coady, C. Schneider, H. Linger, C. Barry and M. Lang (eds), *Information Systems Development Reflections, Challenges and New Directions*, Springer, 2011.
11. J. Petit, O. Giménez and S. Roura, Jutge. org: an educational programming judge, in *Proceedings of 43rd ACM technical symposium on Computer Science Education (SIGCSE)*, Atlanta, USA, March 2012, pp. 445–450.
12. J. L. Fernandez Alemán, Automated assessment in a programming tools course, *IEEE Transactions on Education*, **54**(4), 2011, pp. 576–581.
13. L. de la Fuente-Valentín, A. Pardo and C. Delgado-Kloos, Addressing drop-out and sustained effort issues with large practical groups using an automated delivery and assessment system, *Computers & Education*, **61**, 2013, pp. 33–42.
14. D. Baneres, R. Clariso, J. Jorba and M. Serra, Experiences in Digital Circuit Design Courses: A Self-Study Platform for Learning Support. *IEEE Transactions on Learning Technologies*, **7**(4), 2014, pp. 360–374.
15. A. Y. Al-Zoubi, S. Jeschke, N. M. Natho, J. Nsour and O.F. Pfeiffer, Integration of an online digital logic design lab for IT education, *in proceedings of 9th ACM Special Interest Group for Information Technology Education Conference (SIGITE)*, Cincinnati, USA, October 2008, pp. 237–242.
16. A. A. Saleh, H. El-Bakry and T. T. Asfour, Design of adaptive elearning for logic operations, *International Journal of Education and Information Technologies*, **4**(2), 2010, pp. 49–56.
17. E. Hettiarachchi, M. A. Huertas and E. Mor, E-Assessment

System for Skill and Knowledge Assessment in Computer Engineering Education, *International Journal of Engineering Education*, **31**(2), 2015, pp. 1–12.
18. A. Huertas, Ten years of computer-based tutors for teaching logic 2000-2010: lessons learned, in *proceedings of the Third International Congress on Tools for Teaching Logic (TICTTL)*, Salamanca, Spain, June 2011, pp. 131–140.
19. K. M. Ala-Mutka, A survey of automated assessment approaches for programming assignments, *Computer science education*, **15**(2), 2005, pp. 83–102.
20. C. Douce, D. Livingstone and J. Orwell, Automatic test-based assessment of programming: A review, *Journal on Educational Resources in Computing*, **5**(3), 2005, num. 4.
21. M. Cvetanovic, Z. Radivojevic, V. Blagojevic and M. Bojovic, ADVICE—Educational system for teaching database courses, *IEEE Transactions on Education*, **54**(3), 2011, pp. 398–409.
22. A. Abelló, M. E. Rodríguez González, T. Urpí, X. Burgués, M. J. Casany, C. Martín and C. Quer, LEARN-SQL: Automatic assessment of SQL based on IMS QTI specification, in *proceedings of the 8th International Conference on Advanced Learning Technologies (ICALT)*, Santander, Spain, July 2008, pp. 592-593.
23. A. Abelló, T. Urpí, E. Rodríguez and M. Estévez, Extensión de Moodle para facilitar la corrección automática de cuestionarios y su aplicación en el ámbito de las bases de datos, in *proceedings of the MoodleMoot 2007*, Cáceres, Spain, October 2007.
24. A. Abelló, X. Burgués, M. J. Casany, C. Martín, C. Quer, E. Rodríguez, T. Urpí. LEARN-SQL: *Herramienta de gestión de ejercicios de SQL con autocorrección*, *in proceedings of 15th Jornadas de Enseñanza Universitaria de la Informática (JENUI)*, Barcelona, July 2009.
25. X. Burgués, C. Quer, C. Martín, A. Abelló, M. J. Casany, E. Rodríguez and T. Urpí, Adapting LEARN-SQL to Database computer supported cooperative learning, *in proceedings of workshop Methods and Cases in Computing Education (MCCE)*, Cadiz, Spain, July 2010, pp. 22–29.
26. LearnSQL Interaction—Students Role, https://youtu.be/TqbiEQmDUfA, Accessed February 2016.
27. LearnSQL Interaction—Teachers Role, https://youtu.be/Z9riXOIQZ7o. Accessed February 2016.
28. IMS Global Learning Consortium, IMS Question & Test Interoperability Specification. https://www.imsglobal.org/. Accessed February 2016.
29. C. McDowell, L. Werner, H. E. Bullock and J. Fernald, Pair programming improves student retention, confidence, and program quality, *Communications of the ACM*, **49**(8), 2006, pp. 90–95.
30. International Organization for Standardization and the International Electrotechnical Commission, *ISO/IEC 25010:2011. Systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE)—System and software quality models*, Published by ISO/IEC, 2011.
31. J. Sitthiworachart, M. Joy and E. Sutinen, Success factors for e-assessment in computer science education. In *proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (E-Learn)*, Las Vegas, USA, November 2008, pp. 2287–2293.
32. J. G. Paradis and M. L. Zimmerman, *The MIT guide to science and engineering communication*. MIT Press, 2002.
33. LearnSQL Web Site, https://www.upc.edu/learn-sql/related-systems, Accessed February 2016.
34. D. R. Sadler, Opening up feedback, in S. Merry, M. Price, D. Carless and M. Taras (eds), *Reconceptualising feedback in higher education: Developing dialogue with students*, Routledge, 2013, pp. 54–63.
35. C. F. Timmers, J. Braber-van den Broek and S. M. van den Berg, Motivational beliefs, student effort, and feedback behaviour in computer-based formative assessment, *Computers & Education*, **60**, 2013, pp. 25–31.
36. J. Soler, I. Boada, F. Prados, J. Poch and R. Fabregat, An automatic correction tool for relational Algebra Queries, in *proceedings of International Conference on Computational Science and Its Applications (ICCSA)*, Kuala Lumpur, Malaysia, August 2007, pp. 861–872.
37. A. Mitrovic, An intelligent SQL tutor on the web. *International Journal of Artificial Intelligence in Education*, **13**(2), 2003, pp. 173–197.
38. C. Pahl, C and C. Kenny, Interactive correction and recommendation for computer language learning and training, *IEEE Transactions on Knowledge and Data Engineering*, **21**(6), 2009, pp. 854–866.
39. R. Dollinger and N. A. Melville, Semantic evaluation of SQL queries, in *proceedings of International Conference in Intelligent Computer Communication and Processing (ICCP)*, August 2011, pp. 57–64.
40. Stanford Online DB-Course. https://lagunita.stanford. educourses/Engineering/db/2014_1/about, https://class.coursera.org/db. Accessed June 2015.
41. A. Mitrovic, S. Ohlsson and D. K. Barrow, The effect of positive feedback in a constraint-based intelligent tutoring system, *Computers & Education*, **60**, 2013, pp. 264–272.
42. P. Suraweera and A. Mitrovic, KERMIT: A constraint-based tutor for database modeling, in *proceedings of International Conference on Intelligent Tutoring Systems (ITS)*, Biarritz, France, June 2002, pp. 377–387.
43. A. Mitrovic, NORMIT: A web-enabled tutor for database normalization, in *proceedings of International Conference on Computers in Education (ICCE)*, Auckland, New Zealand, December 2002, pp. 1276–1280.
44. R. Likert, A technique for the measurement of attitudes. *Archives of psychology*, **22**, 1932, p. 55.

Appendix A

For relational database assessment tools, the main approaches for automatically correcting exercises are the following:

Single Test Case Based. This correction approach consists in the definition of one test case for each exercise. The idea is to test if the result of executing the student's response for the test case is the same as the one obtained executing the teacher's solution. One advantage of this approach is that the execution is done in a specific DBMS. This allows specific exercises to be defined for specific DBMS. The main drawbacks are the difficulty to create a test case that is complete enough to discard any wrong response and the incapacity to provide the student with the specific feedback for each mistake in the response. Another reported drawback is that, in SQL and other types of database exercises, different correct responses may exist that all pass the test case but that have different quality according to a specific point of view (e.g., existence of redundant joins in SQL queries). In these cases, a manual post-analysis is needed to further assess the different responses.

Multiple Test Case Based. This correction approach is a version of the Single Test Case where a test case is defined for each possible mistake in the student's response. In this approach, specific feedback can be provided for each mistake, solving one of the main drawbacks of the previous approach.

Constraint Based. In this approach, students' responses are not executed, but they are analyzed to check whether or not they fulfill a set of constraints. The constraints are general to any exercise. Some state SQL language constraints (e.g., the From clause in an SQL query has to contain one or more tables), and others are based on differences that are observed between the student's response and the stored teacher's solution (i.e., the From clause in an SQL query in the student's response does not have to include more tables than the one in the teacher's solution). Again, the main drawback of this approach is the completeness of the defined set of constraints to prevent that fake responses or partially correct responses from being considered as totally correct by the scorer. Another drawback is that a correct student's response that has a different structure from the teacher's solution violates several constraints (i.e., a SQL query with a join that is implemented as a subquery will be considered a wrong response, when it could be totally correct taking into account the result of its execution).

Syntactical Analysis Based. There are two complementary syntactical based approaches. The first one consists in sending the student's response to be executed in a specific DBMS in order to know if it passes the syntactical check done by the system. The second one consists in making a syntactic analysis of the student's response to see if it matches the teacher's solution. In the first approach, the identified errors are related to the misspelling of table names, etc. In the second approach, the problem is the same as in constraint-based approaches; a correct response is not considered to be correct if its syntax is different from the syntax used in the teachers' solution.

Transformation Based. This is a variant of the syntactical analysis-based approaches that try to solve their limitations. It consists in doing a normalization process of students' responses and the teacher's solutions by means of transformation patterns to check that both are equivalent solutions of the exercise. The systems that use this approach can use information about the transformation patterns applied to give feedback to the students in order to improve their responses.

Appendix B

This appendix includes an example of correction of an exercise for each corrector implemented in the SRC of LearnSQL giving examples of feedback provided to the students.

Corrector 1

In this corrector, when a student submits a response, the SCR executes that response for each test case and for each check, and compares the output of the student's response with the output stored in the Item Bank during the exercise definition (that corresponds to the teacher's solution output). If both outputs are identical, then the experiment succeeds. If some of the test cases do not succeed, the message associated to the test case is shown to the student in order to give him/her feedback.

Figure 10 illustrates how the exercise in Fig. 3 of the paper, corresponding to SQL Queries skill, is corrected in the DBMS PostgreSQL. The attached file that the student obtains with the exercise contains the database schema and the public test case. Fig. 10 shows the correction of the public test case and TC5. In the public test case correction, the outputs for the teacher's solution and for the student's response are identical, but they are different when TC5 is corrected. Therefore, Test Case TC5 does not succeed. The reason for this is that a join between the teachers and offices tables is missing in the subquery. The feedback to the student tries to give clues so that he/she can improve the response.

Corrector 2

In this corrector, the student's response creates one or more components in a database and does not generate any output directly. In order to correct exercises of this type, when a student submits a response, the SCR executes that response once. After that, the input corresponding to each test case is executed and the student's output for each test case and the check is compared with the correct output stored in the Item Bank by the teacher.

Figure 11 shows an example of this type of exercise (corrected in the DBMS PostgreSQL). In this SQL/PSM exercise, students must write a function (Stored Procedures). The procedure must list the teachers that have a current assignment to an office of the building given as input parameter. It must also update the area of other offices in the same building. If an exception arises during the execution of the procedure, a specific message must be shown to the user.

Two test cases failed in the student's response of Fig. 11. These two test cases are shown in Fig. 12. In test case TC4, the students' response obtains the teacher Toni twice, and one of them is related to an office to which

-- Database Schema

```
create table teachers
(teacherId char(50),
teacherName char(50) unique,
telephone char(15),
salary integer not null check(salary>0),
primary key (teacherId));

create table offices
(building char(5),
number char(5),
area integer not null check(area>12 and area<50),
primary key (building,number));

create table assignments
(teacherId char(50),
building char(5),
number char(5),
initialInstant integer check (initialInstant > 0),
finalInstant integer check (finalInstant > 0),
primary key (teacherId, building, number, initialInstant),
foreign key (teacherId) references teachers,
foreign key (building, number) references offices,
check(initialInstant < finalInstant));
-- there is a row for assignment of a teacher to an office that begins at initialInstant
-- finalInstant has null value in case it is a current assignment.
```

-- Public Test Case

```
insert into teachers values('111','Toni','66777888',3500);

insert into offices values('Omega','120',20);

insert into assignments values('111','Omega','120',345,null);
```

-- Teacher's Solution Output stored in the Item Bank

Toni

-- Student's Response Output obtained during the correction

Toni

-- Test Case TC5

```
insert into teachers values('111','Toni',null,100);
insert into teachers values('222','Anna',null,50);

insert into offices values('Nexus','120',16);
insert into offices values('Omega','120',30);

insert into assignments values('111','Nexus','120',109,344);
insert into assignments values('222','omega','120',401,null);
```

-- Teacher's Solution Output stored in the Item Bank

Anna

-- Student's Response Output obtained during the correction

No Rows

Fig. 10. Database and correction of two test cases for the SQL Query example.

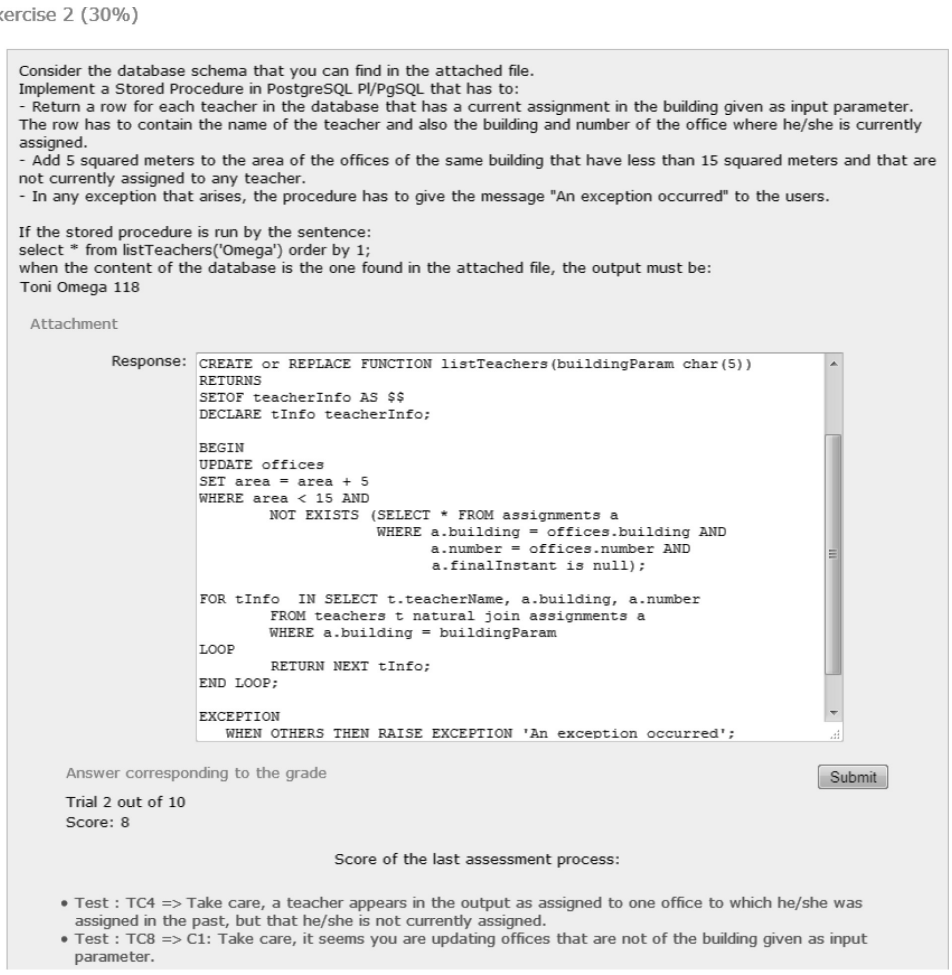


Fig. 11. SQL/PSM exercise example.

```
-- Test Case TC4
insert into teachers
values('111','Toni','666777888',30000);

insert into offices values('Omega','118',16);
insert into offices values('Omega','128',20);

insert into assignments
values('111','Omega','118',1,null);
insert into assignments
values('111','Omega','128',109,200);

-- Input
select * from listTeachers('Omega') order by 1;

-- Check C1
select * from offices order by 1;

-- Teacher's Solution Output TC4 (in the Item Bank)
Toni Omega 118

-- Student's Response Output TC4
Toni Omega 118
Toni Omega 128

-- Teacher's Solution Check C1 (in the Item Bank)
Omega 118 16
Omega 128 20

-- Student's Response Check C1
Omega 118 16
Omega 128 20
```

```
-- Test Case TC8
insert into teachers values('111','Toni','666777888',30000);
insert into teachers values('222','John','666888999',25000);

insert into offices values('Omega','118',16);
insert into offices values('Nexus','128',14);

insert into assignments values('111','Omega','118',1,null);
insert into assignments values('222','Nexus','128',109,200);

-- Input
select * from listTeachers('Omega') order by 1;

-- Check C1
select * from offices order by 1;

-- Teacher's Solution Output TC8 (in the Item Bank)
Toni Omega 118

-- Student's Response Output TC8
Toni Omega 118

-- Teacher's Solution Check C1 (in the Item Bank)
Omega 118 16
Nexus 128 14

-- Student's Response Check C1
Omega 118 16
Nexus 128 19
```

Fig. 12. Failed test cases for the SQL/PSM example.

the teacher is not currently assigned (Omega 128). The reason is that the student made a mistake in the Select sentence of the For block in the Stored Procedure (see Fig. 5). In test case TC8, the check C1 fails. The content of the table offices after executing the student's response is different from the one obtained after executing the teacher's solution. The reason is an error in the update sentence of the Stored Procedure.

Corrector 3

In the exercises corresponding to this corrector, the teacher has to provide the students with a framework of classes that the students must extend and also provide the API of the libraries in this framework that the students will need to use to solve the exercise. The students' responses are compiled and executed together with the whole framework. In these exercises, the inputs of test cases are input parameters of the program. The correction is done by comparing the outputs for the execution of the program for each test case, each input parameter value, and each checks.

Figure 13 shows an example of an exercise of Programming with JDBC (as well as for a correction in the DBMS PostgreSQL). For each teacher given in the input parameters, the program must list the identifier and the number of assignments to offices that the teacher had. The student's response is wrong because the SQL sentence in the program always gives a 1 as a result. For the public test case the output is the same, but not for TC1 where a teacher with two offices exist. A clue is provided to the student so that he/she could find the error.

Corrector 4

In the exercises corresponding to this corrector, and specifically in the Optimization of Workload exercises, the statement describes the exercise context (which is provided in the attached file as SQL sentences of creation and population of tables) and the specific constraints about the parameterization of tables and indexes (e.g., the load factor of tables or B+ indexes, the amount of space used, etc.) that the student's responses must satisfy. The teacher introduces the teacher's solution of optimization of the stated context, which is executed by the SCR to compute its cost. This cost is stored in the Item Bank database for future comparison with the cost of the student's response. Finally, the teacher defines the checks that will be used during the SCR correction in order to compare the cost of the student's response with a reference cost that is between the optimal one (the cost of the teachers' solution with the best database configuration expected) and the worst one (the cost of not defining any additional structure, just the heap tables). The students' response must consist of several index structures, clustered tables and materialized views in order to optimize the context to achieve a workload cost that is as close as possible to the optimal one.

Fig. 13. Programming JDBC exercise example.

Exercises of this type may take too much time to be corrected (i.e., if the SCR had to create and populate the database tables and the additional structures proposed by the students). In order to improve the time efficiency of the correction, instead of requiring the students to specify a database configuration to be created and assessed, they only need to provide the username and password of their database (see Fig. 8). This way, the SCR can access the student's database to correct the exercise (see the average time of correction of this type of exercise in Subsection 3.1). In order to guarantee that the student's database satisfies the constraints and parameters required in the exercise statement, the SCR accesses the student's database catalog to confirm these constraints.

Figure 14 shows an example of an exercise on workload optimization in Oracle. The workload to optimize consists of the four SQL queries given in the statement, each of which must to be executed a percentage of times. The constraint required for the student's database is that the space used in the database must be below 1740 disk blocks. The feedback after the correction states that the index structures created in the student's database failed the check corresponding to the 89% threshold, but that they have successfully passed the previous checks (18%, 35%, 49%, 62%, and 73% improvement for this particular exercise).

As in the previous correctors, the student can do several submissions and the SCR stores the response provided by the student in each submission. In this case, however, since the student does not provide the response through the RTM (but just writes his/her username and password), the SCR must obtain a snapshot of the database content, which consists of a list of tables, materialized views and indexes together with the relevant information about them (e.g., the number of rows, which table is indexed through each index, etc.).

For the second type of exercises corresponding to this corrector (the Optimization using Index exercises), the statement uses the same elements as in the Optimization of Workload exercises and so the student must provide the response in the same way (i.e., giving the username and password of his/her database). However, the way the SCR assesses the student's solution is different. In this case, after checking the fulfillment of constraints related to the creation of structures and space, the SCR simply queries the database catalog and compares the structures created by the student with those created in the teachers' solution.

Corrector 5

Exercises about cost estimation ask the student to compute the cost of a process tree that corresponds to the execution of an SQL query and that is represented as an image in the file that is attached to the exercise. The

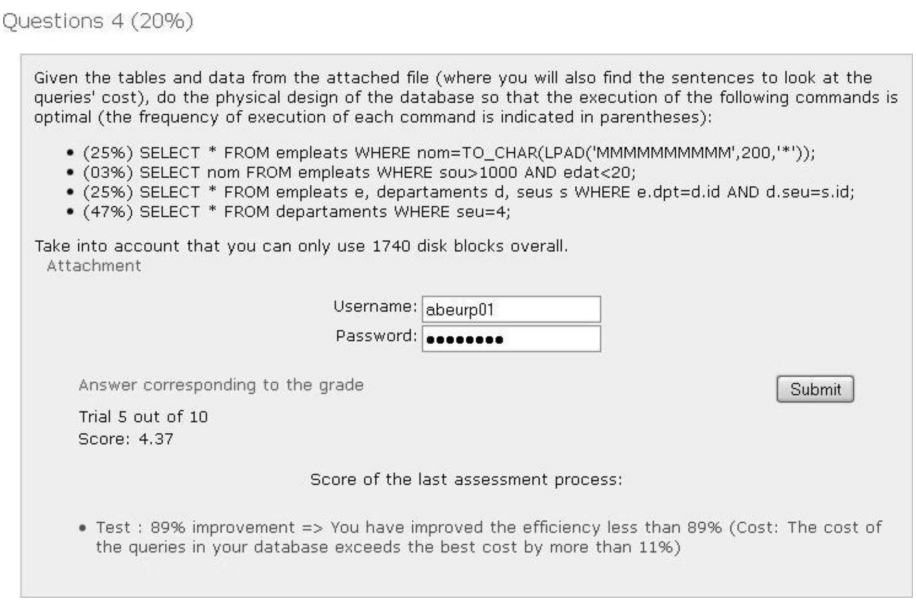
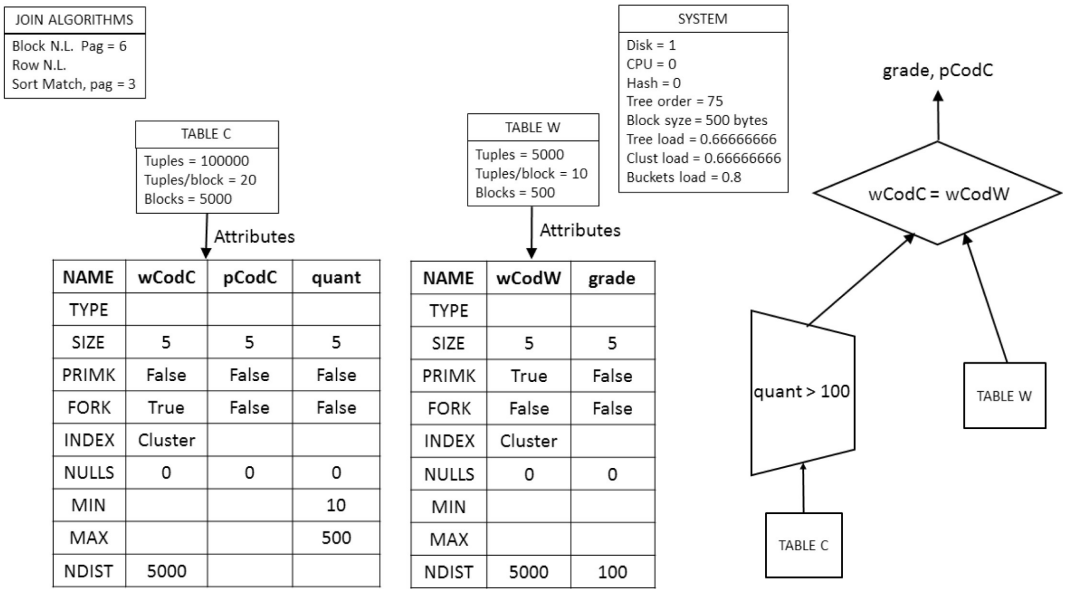


Fig. 14. Optimization of Workload exercise example.

tree can contain selections (using operators like <, >, =, etc.) and equi-joins. Fig. 15 is an instance of such a tree. The attached file also includes additional information that is relevant to the computation required by the exercise: System parameters (e.g., tree order, block size of intermediate tables, load factor for trees, clustered tables, and hash structures); Join algorithms to be considered (e.g., nested loops, index join, sort-match and hash join); Table statistics (e.g., number of rows, rows per block, and number of blocks); Attribute statistics (e.g., size, maximum and minimum values, number of distinct values, whether or not it is a primary or foreign key, number of rows with null value, kind of index over this attribute, if any). Multi-attribute indexes are not supported. Structures that are known by the scorer are B+ trees, hash index, clustered, index and bitmaps.

The SCR computes the number of rows in the query output (cardinality) and discovers the best access plan. It then computes the corresponding cost and stores the results in the Item Bank database.

The student's response to the exercise must be the cardinality issued by the query and the cost of the best execution plan. The SCR tells the student whether or not the cardinality is correct. If it is correct, the SCR proceeds with the cost assessment. If it is not the expected one, the SCR tries to guess the mistake made by the



NAME	wCodC	pCodC	quant
TYPE			
SIZE	5	5	5
PRIMK	False	False	False
FORK	True	False	False
INDEX	Cluster		
NULLS	0	0	0
MIN			10
MAX			500
NDIST	5000		

NAME	wCodW	grade
TYPE		
SIZE	5	5
PRIMK	True	False
FORK	False	False
INDEX	Cluster	
NULLS	0	0
MIN		
MAX		
NDIST	5000	100

Fig. 15. Cost estimation exercise example.

student (e.g., trying to use an index over the result of a selection on an indexed table, not taking into account the load factor of a clustered table, etc.).

Alberto Abelló is tenure-track 2 professor at UPC. He received his PhD from UPC in 2002. He is also a member of the MPI research group (Grup de recerca en Modelització i Processament de la Informació) at the same university. His research interests are databases, database design, data warehousing, OLAP tools, ontologies and reasoning. He is the author of articles and papers presented and published in national and international conferences and journals on these subjects (e.g., TKDE, ISJ, DKE, IJDWM and KAIS). He has also served in the program committee of VLDB, EDBT, DaWaK and DOLAP among others. Currently, he is the local coordinator at UPC of the Erasmus Mundus PhD program IT4BI-DC, and the postgraduate course on Big Data Management & Analytics

Xavier Burgués received the B.Sc. degree in computer science from UPC in 1986. He is currently professor at the ESSI of UPC. He is responsible of the Database Design course at the FIB. His research interests include conceptual modeling and metamodeling concepts, methodologies and tools.

Maria José Casany received her PhD from UPC in 2013. She has participated in the development of several LMS and authoring tools. She is a collaborating professor at the ESSI department of UPC. Since, 2004 she has taught databases at the UPC. Her currently research interests include e-learning, mobile learning and free and open source software.

Carme Martín received the B.Sc. and Ph.D. degrees in computer science from UPC in 1991 and 2005, respectively. She is currently collaborating professor at the Service and Information System Engineering department (ESSI) of UPC. She is the coordinator of the cross-curricula competence *Appropriate Attitude Towards Work* of the undergraduate and master's studies of the FIB of UPC. Her research interests include database systems, temporal databases and integrity constraint checking. Current research interests are involved with computer science education and databases for natural language processing.

Carme Quer is professor at UPC. She is a member of the GESSI research group (Group of Software Engineering for Information Systems). Her research interests are Requirement Patterns and Reuse in Requirements Engineering (http://www.upc.edu/gessi/PABRE/), Software Quality and Relational Databases e-Learning. She is participating in the LEARN-SQL project that has as aim the use of new techniques for improving the learning in the databases area.

M. Elena Rodríguez received the B.Sc. and Ph.D. degrees in computer science from UPC and University of Alcalá in 1993 and 2012, respectively. She is currently professor at the Universitat Oberta de Catalunya (UOC) where she coordinates undergraduate and master's database courses. Her current research interests deal with knowledge engineering with particular application in technology enhanced learning.

Oscar Romero is currently a tenure-track 2 lecturer at the ESSI department (UPC). He also coordinates the IT4BI Erasmus Mundus Master at UPC and the Big Data Management and Analytics postgraduate course at UPC School. His main interests are business intelligence, Big Data and the semantic web. His PhD thesis focused on data warehousing but since then, he has been working on many other topics such as NOSQL (and any technology beyond relational databases), bridging Big Data management and analytics, open data platforms (mostly at the database level), recommendation systems and semantic-aware systems (based or exploiting semantic formalisms such as ontology languages or RDF). He is also interested in agile methodologies/formalisms to incorporate non-technical people in the design, maintenance and evolution of database systems.

Toni Urpí received the B.Sc. and Ph.D. degrees in computer science from UPC in 1985 and 1993, respectively. He is currently professor at the ESSI department of UPC. He is currently the head of the ESSI department. His research interests include deductive databases, database updates and integrity constraint maintenance. Current research interests are involved with conceptual modeling, schema validation, abduction and query containment.