Learning Databases Interactively: An SQL Adventure

Nicole Burgstaller
Claudia van der Rijst
Michael Morak
Claudia Steinberger
Universität Klagenfurt
Austria
nicole.burgstaller@aau.at
claudia.rijst@aau.at
michael.morak@aau.at
claudia.steinberger@aau.at

Abstract: Teachers and students face multiple challenges when learning and practicing the Structured Query Language (SQL) in database education. While hands-on practice is important for learning SQL, teachers often lack interesting sample data and exercises for students to practice with, and students may not have access to suitable devices or software to practice on. In this paper, we propose a web-based learning environment called aDBenture, which addresses these challenges by providing students with intelligent and adventurous exercises, and teachers with the ability to reuse, define, and analyze exercises and student work. aDBenture aims to support the teaching and learning of relational databases in an interactive and engaging way. The authors provide an overview of the current research in the field, as well as the requirements and features of the aDBenture tool and their first evaluations.

Introduction and Motivation

The subject area of databases and, in particular, relational databases, is part of many curricula at schools and universities. The correct use of Structured Query Language (SQL) (Molinaro 2005) forms a large part of a typical database course. As with learning any programming language, learning SQL involves a lot of hands-on practice. The more interesting the tasks in practical exercises are, the greater the motivation of the learners to deal with them (Boyle et al. 2016, Papastergiou 2009, Xinogalos & Satratzemi 2022). Mobile access to an exercise environment also makes practicing more accessible.

However, teachers often lack interesting sample data and exercises for their students to practice on. Usually, widely used example data is downloaded from the Web (see Sample Databases) or a very small set of sample data is created. Based on this data, teachers design exercises in the form of documents for the students mostly without embedding them into a motivating context. Then, students are expected to install a special database management system on their workplace computers and upload the sample data. Hence, students often need to prepare their isolated environment and to solve the given exercises there. What problems and successes the students experience over the course of their SQL exercises and whether and how much practice is done is not apparent to the teachers in this way.
For students, the installation of the database management system requires their own workplace computer with specific installation rights. Mobile devices like phones or tablets are not suitable to do the hands-on practices. Existing SQL editors like pgadmin (pgAdmin) or phpMyAdmin (phpMyAdmin) are also often too complex for database beginners and do not provide enough feedback when developing solutions for given exercises.
This situation should be improved by learning environments which (i) enable students in the frontend to learn based on intelligent and adventurous exercises and (ii) enable teachers in the backend to reuse or define adventurous exercises and analyse students work.

Several approaches exist in the literature that try to overcome these problems and to improve database learning environments. In the tools we could find, often only one predefined database is used. Game based approaches are

fixed to one adventure and the games are stateless. That means no results are stored. None of the tools offers adventures as reusable open learning resources, which can be reused as learning objects in an own learning context. Other approaches are tutorial oriented without any game-based touch. A teacher's backend is available almost never.

In this paper, we propose aDBenture, a learning environment that tries to overcome these weaknesses. To this end, we developed a web-based learning environment for SQL, which allows learners a mobile and adventurous approach to learning SQL. In addition, in this web-based learning environment they also receive helpful feedback on the correctness of their results as well as on errors that are still present in comparison to stored sample solutions. Teachers can use interesting sample data (e.g., a criminal database) and existing adventures (e.g., criminal cases) to let their students practice with SQL. Moreover, they can also create their own adventures and upload their own sample data. The motivation of aDBenture is to support the teaching and learning of relational databases in an interactive and engaging way.

The remainder of the paper is structured as follows. First, we deal with current research work in database learning environments. Then we define requirements for our approach including security aspects that are quite essential for our web-based approach. We continue by describing the learner's frontend of the aDBenture tools, as well as the teacher's backend. After a short description of the technical background, we present some preliminary evaluation results, future plans, and finally offer some conclusions.

Current Research

Todays, a number of tools exist that allow for managing and querying databases over the Web (phpMyAdmin) or via a special administration platform (pgAdmin). These tools are often used in the context of SQL education as well. Learners are expected to install a database management system locally on an available workstation and upload given sample data there. The installation process requires time, installation knowledge, and installation rights. Mobile devices, such as phones or tablets, are not suitable for practical exercises conducted in this fashion. Also, these tools provide a comprehensive set of functions which aren't needed in the SQL education process and often cause confusion for the learner. Often, such tools are targeted towards database administrators.

Working with these tools, students have to practice SQL in an isolated environment. Often, they do not get enough feedback about errors or correct result sets when developing SQL queries on the database. Also, it is not apparent to the teacher what problems and successes the students have in their SQL exercises and whether (and how much) practice is done at all. Furthermore, such database management tools are not well suited to be used in an exam situation.

To improve this situation, some learner or teacher-oriented solutions have been developed which focus on learning or assessing SQL. In the following, we give an overview and focus on approaches that offer free access, do not require any installation process for the learner to perform SQL exercises, and offer responsive web interfaces to enable mobile learning.

SQL Island (Xinogalos & Satratzemi 2022, SQL-Island.de) is a text adventure learning game for the database language SQL. The game is about a person who survived a plane crash on a desert island. SQL island offers the learner exactly one adventure and one database. The goal of the game is to find a way to escape the island. This way can be found by solving tasks using SQL queries. In case of an invalid SQL query, immediate feedback is given to the learner. If the query contains an error, a corresponding message is displayed. Required SQL commands are also shown during the game. If the SQL query is valid but returns an unexpected result, the learner receives a hint about the correct result or about incorrect tuples in the result table. The game is stateless, which means that the learner must start from the beginning when a game session is closed. The responsiveness of the game is also limited. It is also not possible for teachers to access the students' SQL statements, since no corresponding backend for teachers exists.

In SQL Murder Mystery (Canale & Farinetti 2022) the learner has to solve a crime mystery in SQL city. SQL Murder Mystery is designed to be both a self-directed lesson to learn SQL concepts and commands and a fun game for experienced SQL users to solve an intriguing crime. It is based on SQLite as a database management system and

offers the learner exactly one adventure and one database. In search for the murder, there are no predefined tasks to be fulfilled using SQL. The learner is expected to query the database to find out who committed the murder using his or her knowledge about the database schema. The game is stateless. A backend for the teacher does not exist.

SQLZoo (Cigas & Kushan 2010) is an online platform for writing and running SQL queries against a live database. It offers prepared tasks that serve as tutorials but also some that have to be solved by the learners themselves. Since the tool is not game based, there is no score to save. Feedback is provided by the database management system in case of syntactic and semantic errors. If the result of a query is not as expected, there is no detailed information on how to correct the error. It is not possible for a teacher to see the queries that have been entered by the learners. Other sets of sample data and tasks cannot be uploaded. The game is stateless.

SQL-Fiddle (SQLFiddle.com) and DB-Fiddle (DB-Fiddle) are platforms that allows a user to upload and query data in different database management systems. This spares the learner the installation of the database management system on their own workstation. So, the platforms do not provide a database to be queried or an adventure or tasks to be solved. They give no feedback on entered queries other than the error messages from the database management system. Both of them are stateless.

Coderunner (Lobb & Harlow 2016) is a plugin for the learning management system Moodle which can be used to train and assess programming skills. Next to SQL, CodeRunner currently supports also the programming languages Python, C, C++, Java, PHP, JavaScript, Octave and Matlab. It can be used for practicing programming skills as well as for exams in the context of Moodle Quizzes. When creating a SQL question as part of a Moodle quiz, the teacher must define a textual query, upload a (small) database schema plus associated data, and define the expected result. The learner must then find an SQL solution that matches the expected result when completing the quiz. The expected tuples of a question can be viewed by the learner, which is helpful in developing a correct SQL query. The learners' queries are stored and can be analyzed by the teacher after the assessment. Coderunner does not work on a live database but only in the SQLight environment of the Moodle Plugin. Thus, it is only usable in a Moodle environment and also requires several access rights to the course in which the quiz was set up.

The W3Schools online platform (W3schools SQL Tutorial) explains SQL using simple query examples, which users can try out in an online editor. Here, however, the user does not get any feedback, but has to be satisfied either with the already learned pattern query or the results table. In addition, the number of examples is limited to one simple example per subject area and there is only one database to practice.

In summary, the approaches and tools we have encountered mostly use one a predefined database. The investigated game-based approaches are fixed to one adventure and their games are stateless. When errors occur the user feedbacks of the investigated tools are unsatisfactory and inflexible. None of the tools offers adventures as reusable open learning resources, which can be used as learning objects in tutorials, MOOCs or private online courses. The development of new adventures by teachers is also not possible. Other approaches are tutorial oriented without any game-based touch. A teacher's backend is almost never available. Only Coderunner enables a teacher to define queries by themselves and to see the learner’s results. But this plugin also has its weaknesses, as reported above.

aDBenture – teaching and learning SQL interactively in an adventure setting

In the next sections, we will describe the aDBenture platform. Its goal is to overcome the problems mentioned in the overview of current research above. The name “aDBenture” stands for “Adventure Databases”. We developed a web-based and responsive tool called aDBenture. Students should be able to practice SQL in a game-based way via the frontend of aDBenture.Teachers should be able to create and edit their own SQL adventures and reuse existing adventures via the backend of aDBenture.

Based on the analysis of existing approaches described above, we defined the following requirements:

1. An adventure can be developed by a teacher (= owner), using the aDBenture backend. An adventure can be private or public. The reuse and adaptation of public adventures is possible across the aDBenture teacher community.

2. Adventures should take place in a certain domain and be designed as a series of tasks to be solved by the player (= learner) with the help of SQL.
3. A task consists of a goal, a number of points that should be awarded on successful completion, and several sample solutions that are used to evaluate the player's solution attempts. These sample solutions also represent the knowledge to later provide intelligent feedback to the player in case of incorrect solution attempts.
4. Using the aDBenture frontend, a player can try to solve the tasks of an adventure and train their SQL skills anonymously or while logged in. The player can immediately see if their queries are correct or deviate from the expected results.
5. The aDBenture frontend provides intelligent and helpful feedback to the player. Intelligent error messages are issued based on sample solutions rather than just the result of the SQL query. Through the aDBenture backend, the teacher can set the level of feedback provided to the player (e.g., accurate hints for errors, deviations from the expected solution, no hints).
6. The results of logged in players can be analysed via the backend by the teacher.
7. Adventures can take place in different domains. Hence, several different databases must be able to exist in the background, on which the adventures are built. Teachers can reuse existing databases and upload their own. Multiple relational database systems should be supported.
8. The frontend and the backend of aDBenture must be secured against possible cyberattacks.
9. aDBenture should also be usable in assessment situations. Teachers should be able to create adventures as exams, accessible via a complex URL and a randomly generated key.

Figure 1 presents a use case diagram of aDBenture.

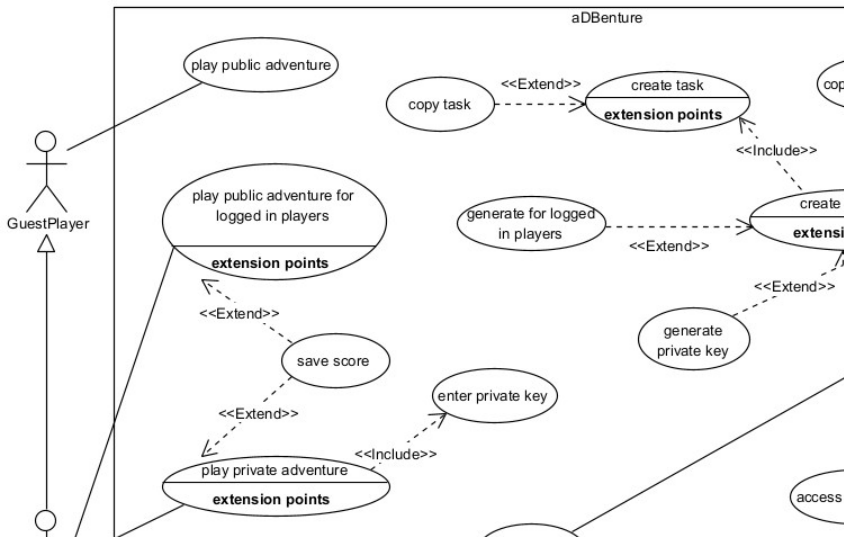

Figure 1: Use Case Diagram of aDBenture

In aDBenture the following user roles are available:

- GuestPlayer: For users who do not want to register, some adventures are offered. Their scores cannot be saved and teachers are not able to see the queries they have entered. For this user, aDBenture essentially is a stateless SQL adventure game.
- Player: Learners who register in the system are provided with an expanded selection of adventures. When playing through an adventure, the status of the tasks, the entered queries, and their scores are saved.
- Teacher: Teachers can create adventures and tasks. They can copy and reuse existing adventures or design entirely new ones. They can also upload their own database instances for this purpose. It is possible to create public adventures or private ones that are protected by a key. Furthermore, it is possible for teachers to analyze the game results, that is, access the information saved about registered players and their attempts.
- Admin: the admin can assign the teacher role and has access to all stored databases and adventures.

Figure 2 shows an overview of the sematic data model of aDBenture, which is used to store the data about the adventures and users.

The class *Adventure* describes the adventures and their attributes. Each adventure has a name, a story and a URL to directly reference the adventure e.g., from a tutorial, an exercise sheet or a MOOC.
The other attributes are necessary to customize an adventure (e.g., to make it private or public, to set its feedback intensity or to generate a solution key). Each adventure is owned by one *Teacher*. It is based on one *Database*, which has a certain *Schema*. It is possible to also store a corresponding *UML* class diagram explaining the conceptual model of the database.
An *Adventure* includes one or many *Tasks*. A *Task* has one or many *SampleSolutions* and a certain number of points. It relates to one or many *Tables* of the domain database. The domain database used of an adventure is not stored in the adventure database but referenced from here. An *Adventure* can be played by *Players*. The *Score* of each *Task* they play is stored as well as their last query.

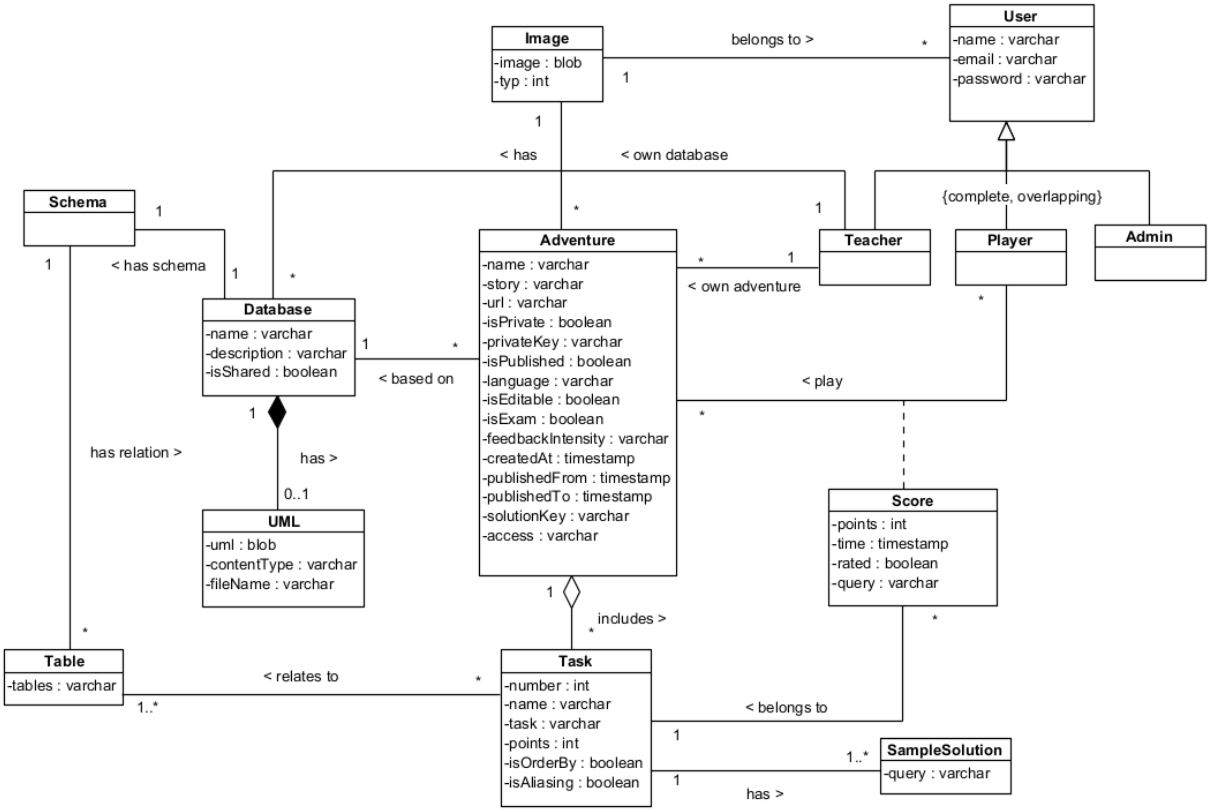

Figure 2: aDBenture semantic data model

aDBenture was implemented as a microservice. It was written in Java and implemented with the Spring framework. Angular was chosen as the framework for the user interface, and Codemirror was used for the SQL editors. The communication between the web-based user interface and the server-side code is done via REST. The application is built on a MySQL database but for the domain databases multiple relational database systems are supported. The connection from server-side code to the database is done via JDBC on the server.

In the following, the GUI of the frontend and the backend of aDBenture are briefly introduced. Security aspects as well as the intelligent feedback are also discussed. We reference the above requirements where appropriate.

aDBenture Frontend for Players

aDBenture can be used by guest players to easily and anonymously train one's SQL skills. There are several public adventures available for this. However, in order to be able to save an adventure score or to get access to further, non-public adventures, registration and login is required.
Figure 3 shows the aDBenture landing page for a player. It displays some predefined adventures as well as adventures that have already been played. More public adventures can be looked for using the filter icon at the top-

right of the toolbar. Registered players also have access to private adventures that can only be played with a key. The keys are generated by the teachers and announced to the players. When a player moves the mouse over a title picture, they receive brief information about the number of tasks, the achievable number of points and on which database instance the adventure is based. If a player is also a teacher, they can access the teacher's backend clicking on the hat icon.

Figure 3: aDBenture Landing Page

If a player clicks on an adventure, they get a more detailed overview of the adventure. There is information about the story, and registered players can see their previous score, if they already played the adventure. They can see the number of tasks and the achievable number of points. If they have solved tasks correctly, they can also see the achieved score (see Figure 4).
When the player starts or continues the adventure, they start with the first task immediately and they can try to solve it using an SQL query (Requirement 2). Each task has a certain number of points that a player can receive if they solve it correctly. Below the amount of achievable points, the goal of the task is described (see Figure 5).

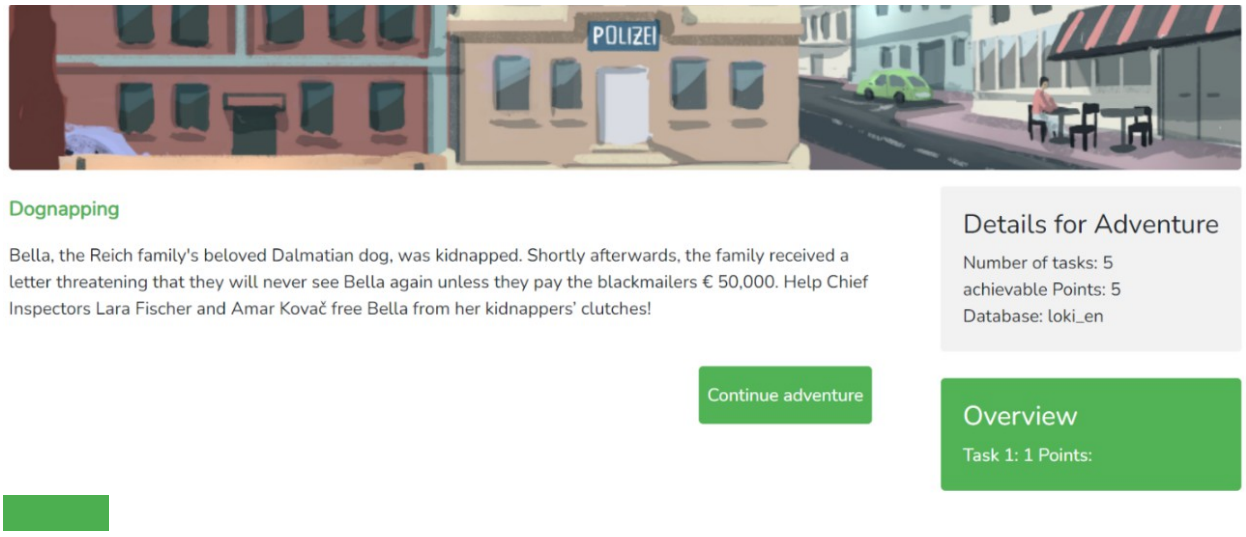

Figure 4: Introduction to the adventure "Dognapping"

On the right-hand side in Figure 5 the player finds the database schema with the tables, attributes and data types that the teacher has recommended for solving the task. It is also possible to display the UML diagram of the database schema. Below, the player finds an editor to enter SQL queries to solve the given task. Depending on the feedback level the teacher has set for the relevant adventure, the player also has the option to display the expected result (like in Figure 5). When the player checks a query, they get feedback about its correctness (Green in Figure 5 -> the result is correct; or Red in Figure 6 -> a problem was detected; Requirement 4).

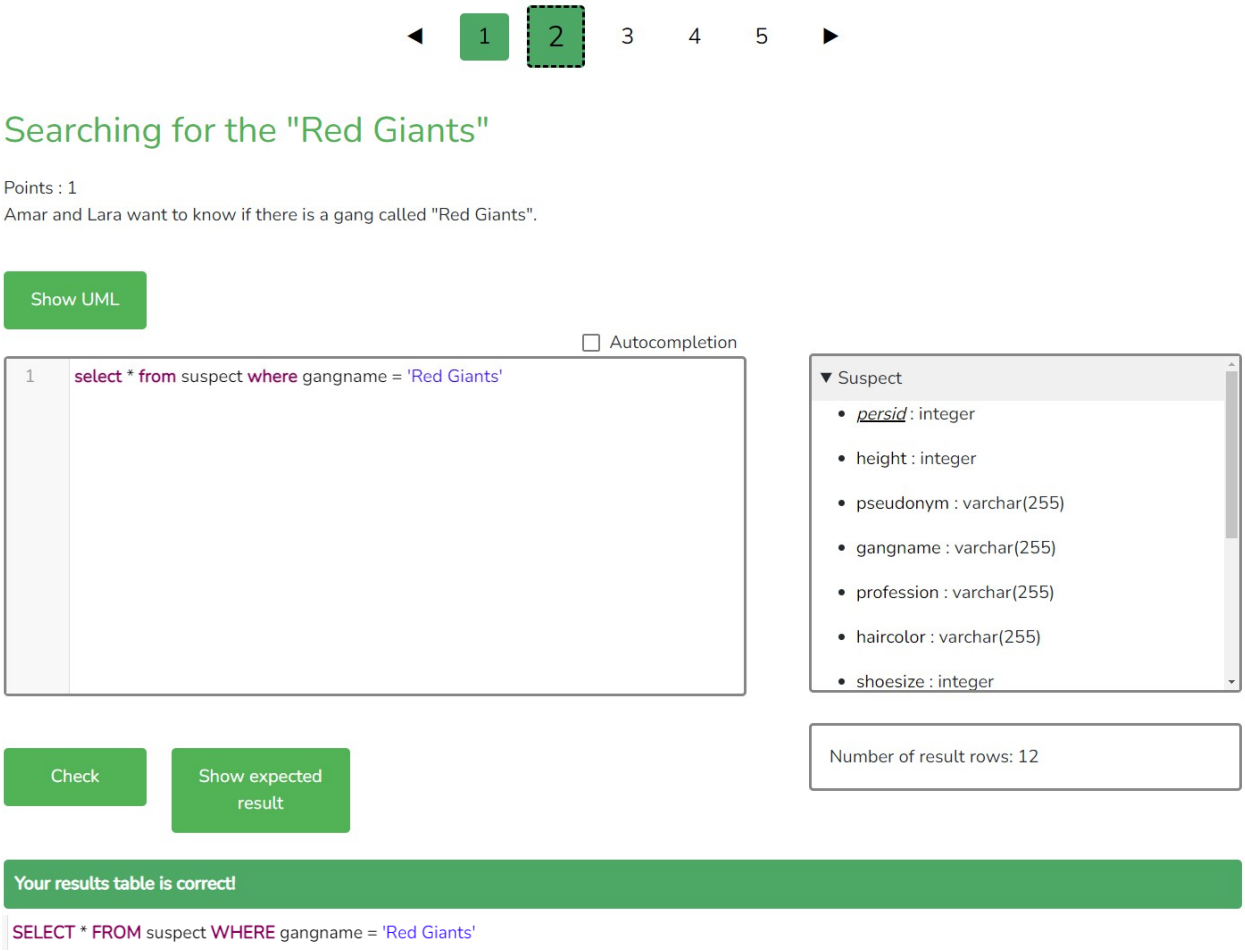

Figure 5:The Editor with positive feedback and the schema.

Intelligent feedback for the player is one of the core requirements which were defined for aDBenture (Requirement 5). When checking whether a task has been solved correctly, several steps are performed by the aDBenture feedback algorithm:

First, the player's query is compared with all the sample solutions stored by the teacher. If the player's query exactly matches one of them, the check is terminated, and the entered query is marked as correct. If the player's query returns the same result table as the first sample solution, but has no similarity with one of the sample solutions, the task is considered to be solved, but the query should be checked manually by the teacher in the backend. If the teacher confirms the correctness of the query, it can be added to the list of sample solutions. In this way, aDBenture becomes a learning system. We will now describe how our tool provides feedback for incorrect queries entered for a given task:

If the sample solution result table and the player's query result differ from each other the feedback algorithm proceeds in the following way: First, the sample solution that seems most like the player's query is determined. The SELECT clause of this sample solution and the player's query are examined and compared. Based on this, the player is told which columns are missing or unnecessary, or feedback is given that aliases should be used. If the SELECT clauses match, the next step is to compare the number of their result rows. If the number of result rows do not match, the query is considered to be incorrect. If the number of result rows match, the results are now compared to each other. If the result tables differ from each other, the player's query is evaluated in more detail. The feedback algorithm compares the FROM clauses and tells the player which tables are not required, or which tables are missing. If the tables correspond, the WHERE clause is examined more closely. This step is also performed for

queries that were considered correct so far. For the evaluation of the WHERE-conditions all stored sample queries are consulted. For this, the WHERE clause is split into smaller conditions, using keywords like AND, OR, LIKE, BETWEEN as a guide. Then the feedback algorithm checks which conditions match and which are unnecessary or missing (see Figure 6). It evaluates also missing or unnecessary columns, tables, and conditions. It is also possible to detect missing sorting criteria and inform the player when an alias is expected. If each of these conditions can be satisfied by the given query, that query is considered correct by the system.

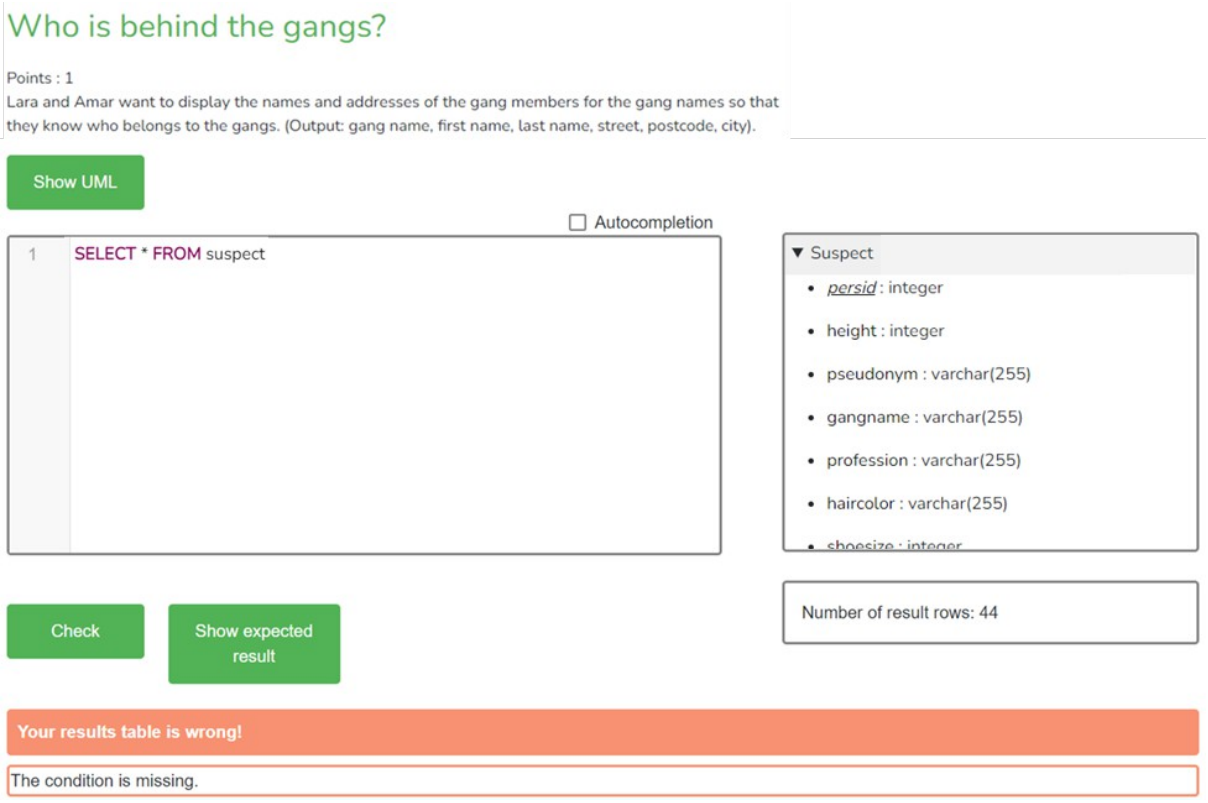

Figure 6: Feedback on an incorrect players query

To secure the front end of aDBenture against possible cyberattacks, players are prevented from modifying or deleting the database with DROP, ALTER TABLE or INSERT statements. A strict database user management is applied and aDBenture works with different database connections for different tasks (requirement 8). Access rights have been implemented as strictly as possible. Thus, players are only allowed to perform SELECT statements. Information about the schema of the used database is also not accessible to players. All information about the schema structure is retrieved via a separate database connection, apart from the one used to access the adventures themselves.

aDBenture Backend for Teachers

aDBenture's backend allows teachers to create and edit their own SQL adventures or to reuse and adapt them (Requirement 1). As shown in Figure 7, the teacher must enter a title and a short story describing the adventure. The teacher now has to choose the domain in which the adventure will take place. To do this, they can select a database of an existing domain (e.g., loki_en, selected in the figure, is an English crime case database) or upload their own (Requirement 7). The link to the domain is generated by aDBenture automatically. This link can be used to access an adventure directly, for example if it is used in a tutorial or a massively open online course (MOOC).

An adventure can be made accessible to different player groups. This is done making an adventure public or private and via the login requirement of the player. In case of a private adventure, only logged in players can play it. Private adventures are only accessible via a key generated automatically (Requirement 1). Depending on how much support

the player should receive, the teacher can adjust the level of feedback. This ranges from no feedback at all to very detailed feedback (Requirement 5). If access to a new adventure is not restricted, the adventure is also available for guest players.

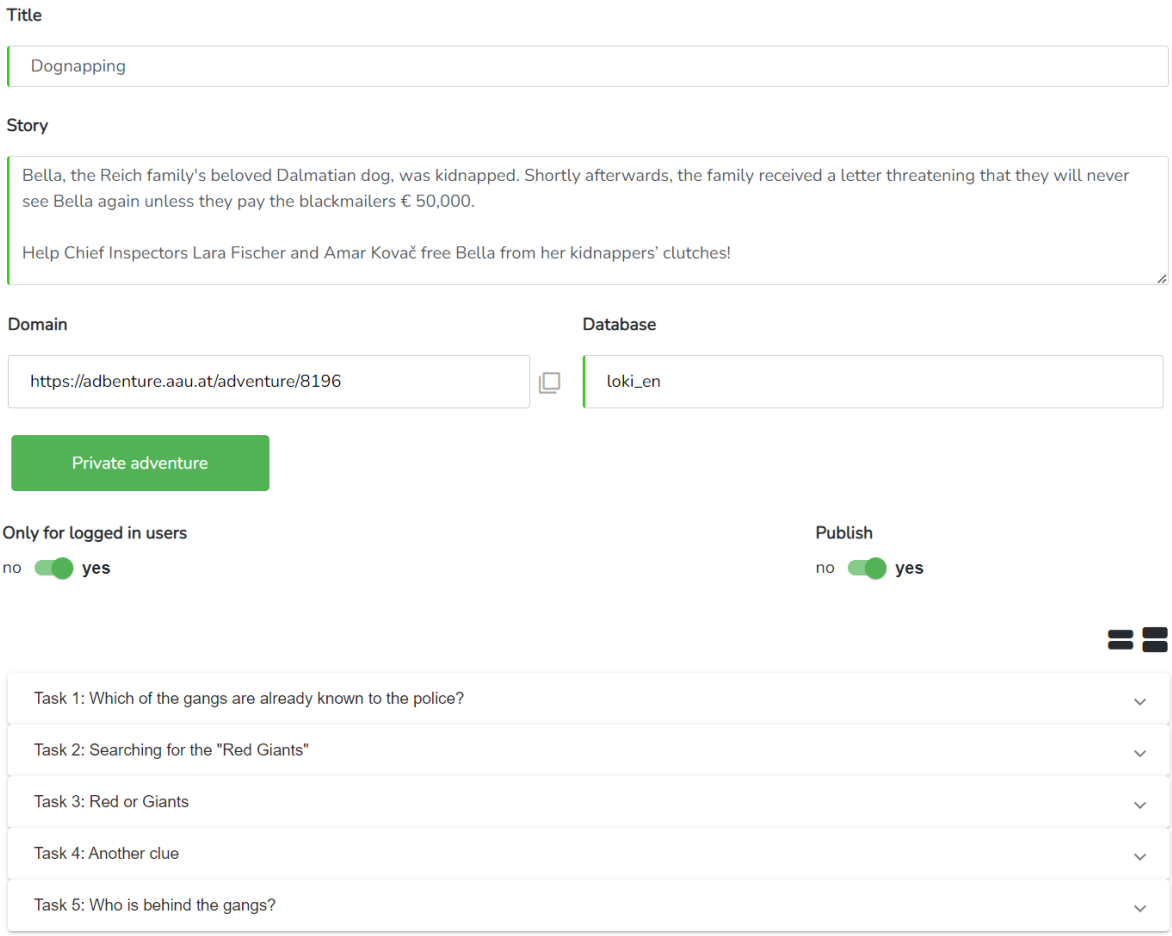

Figure 7: aDBenture's backend for creating/editing an adventure

An adventure consists of one or more tasks to be solved. E.g., the adventure in Figure 7 contains 5 tasks.

Figure 8 shows how to create and edit a task. It is obligatory that the teacher enters the title and the task description. They can select also some tables which are displayed then to the player as a schema view in the frontend. The teacher must also assign points to the task and assign at least one sample solution. Several samples can be separated by semicolons. To ensure a good usability the teacher can also check their sample solutions here in the backend (Requirement 3).

The backend of aDBenture offers also the opportunity to copy public adventures. Making them private afterwards enables the teacher to adapt copied tasks (Requirement 1). When the teacher publishes the adventure, it can be played. The player's queries are saved, and the results can be analysed by the teacher. The game results include the player's name, the entered query, the date and time of entry and the score (Requirement 6).

Building an adventure, teachers can use an existing the database, but they can also upload their own. When creating a new database instance within aDBenture, the name of the database and a brief description of the database domain are required. Teachers upload a UML diagram of the domain, an according relational schema. If the database could

be created successfully, corresponding tuples can be uploaded (Requirement 7). A database can be kept private or made public for other teachers.

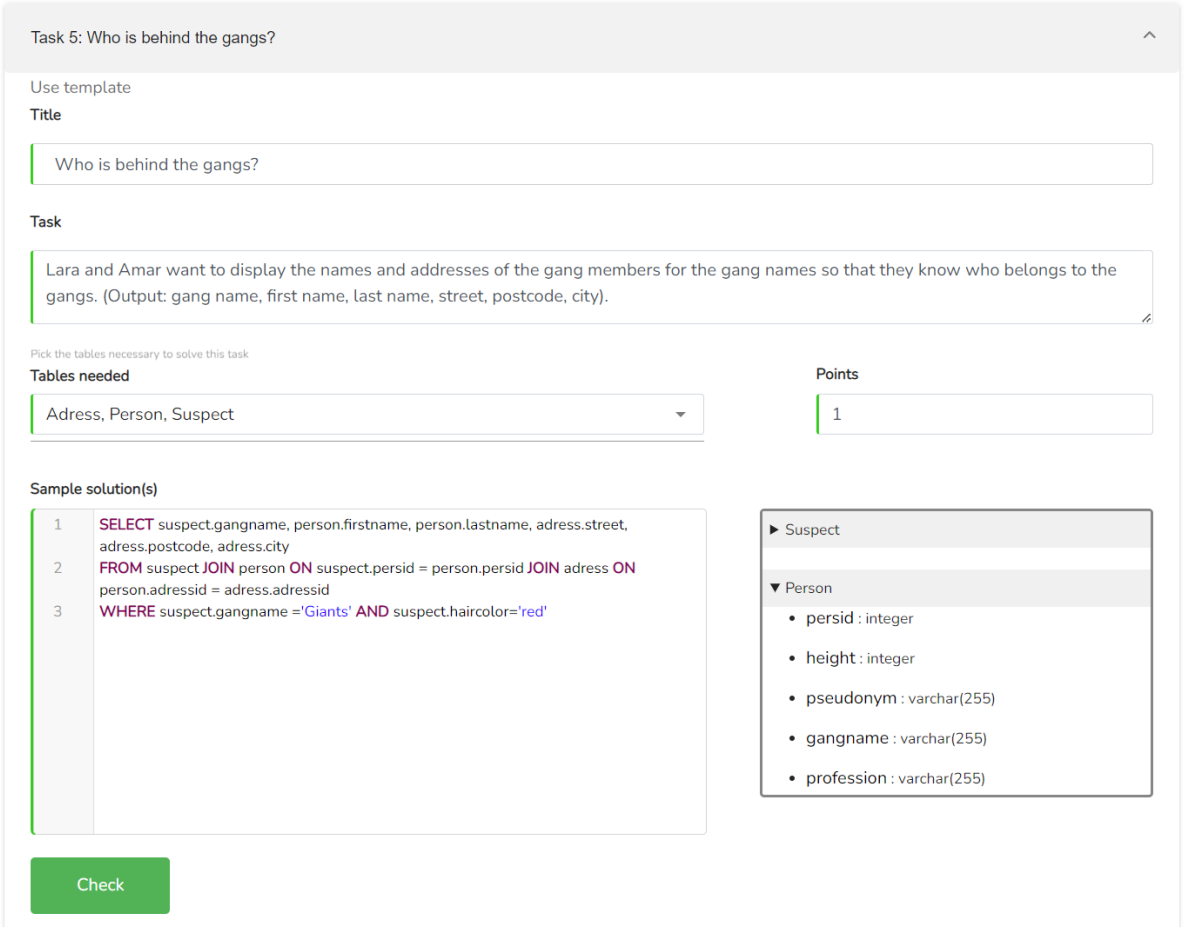

Figure 8: aDBenture's backend for creating/editing the tasks of an adventure

Since aDBenture can be used by students to practise SQL, it makes sense to use aDBenture also in assessment situations (Requirement 9). For this reason, teachers have the option to mark an adventure as an exam and use it, e.g., in a Safe Exam Browser (Safe Exam Browser). To integrate aDBenture into the Safe Exam Browser, the adventure must not contain any link that can be used to leave the exam. It also must not be possible to access another adventure and close when the assessment time has expired. To secure the backend of aDBenture against cyber-attacks, some restrictions were defined, e.g., an upper limit of databases to be uploaded and a size limit for the databases is set. Once the own database schema has been uploaded, the schema cannot be changed by the teacher anymore. In addition, the teacher is not allowed to access system tables (Requirement 8).

Evaluation

The development of aDBenture started in Summer 2021. The first prototype was tested and evaluated in a database course in spring 2022 with approximately 30 students. For this purpose, we used PostgreSQL as the exercise database because it was already being used in the original database course the year before. This evaluation was aimed to integrate the target group into the development process of aDBenture and to improve its usability. While a sample size of 30 students is not enough to get representative statistics, we did notice that students showed more enthusiasm about practicing SQL.

In autumn 2022 aDBenture was presented in a conference workshop (ISSEP) to collect some teachers' feedback. As the feedback from both students and teachers was very positive, we integrated suggestions for improvements and

developed a new version. The authors are also working on the development of a MOOC on databases (iMooX) which will be available online via iMooX (Ebner 2021). This database MOOC is going to use aDBenture in a form that students can interactively practice their SQL skills in all SQL lessons. SQL exercises will be provided there as aDBenture adventures. The first version of this database MOOC will be evaluated in spring 2023 with 120 students within a university-level database course. We plan to publish these evaluation results soon. The current version is available in German, English and Spain at adbenture.aau.at.

Resumé and further work

This paper investigated approaches to learn SQL. To overcome the weaknesses, we identified in the current state-of-the-art, a list of requirements for an innovative tool to practice SQL was defined. We then presented a tool, called aDBenture, to satisfy these requirements. Learners can practice SQL via the responsive frontend of aDBenture, teachers can also customize their own domains and exercises and analyze their students result via the backend of aDBenture. aDBenture can be used standalone by learners to practice SQL. However, adventures from aDBenture can also be embedded flexibly in SQL tutorials, MOOCS, and secure exam environments.

As next steps we are going to work on further game-based features for aDBenture, e.g., adventures that allow synchronous or asynchronous SQL competitions and security aspects of databases. (Schildgen & Rosin 2022)

References

Boyle, E.A., et al. (2016). An update to the systematic literature review of empirical evidence of the impacts and outcomes of computer games and serious games. *Computers & Education,* 94, 178–192.
Canale, L., & Farinetti L. (2022). SQL Murder Mystery: a serious game to learn querying databases. *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE.
Cigas J., & Kushan B. (2010). Experiences with online SQL environments. *Journal of Computing Sciences in Colleges*, 25 (5), 251–257.
DB-Fiddle, URL: https://www.db-fiddle.com/, last access 20.01.2023.
Ebner, M. (2021). iMooX-a MOOC platform for all (universities). *2021 7th International Conference on Electrical, Electronics and Information Engineering (ICEEIE)*. IEEE.
iMooX, Kurs: Tauch‘ ein in die Welt der Datenbanken, URL: https://imoox.at/mooc/course/view.php?id=217, last access 01.06.2023.
ISSEP (2022), The 15th International Conference on Informatics in Schools – A step beyond digital education. URL: https://issep2022.ifs.tuwien.ac.at/
Lobb, R., & Harlow J. (2016). Coderunner: A tool for assessing computer programming skills. *ACM Inroads,* 7 (1), 47–51.
Molinaro, A. (2005). *SQL Cookbook: Query Solutions and Techniques for Database Developers*. O’Reilly Media, Inc.
Papastergiou, M. (2009). Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Computers & education,* 52 (1), 1–12.
pgAdmin, PostgreSQL Tools, URL: https://www.pgadmin.org/ last access 20.1.2023.
phpMyAdmin, Bringing MySQL to the web, URL: https://www.phpmyadmin.net/ last access 2.2.2023.
Safe Exam Browser, URL: https://safeexambrowser.org, last access 22.01.2023.
Sample Databases, URL: https://github.com/morenoh149/postgresDBSamples last access 20.1.2023.
Schildgen, J., & Rosin, J. (2022). Game-based Learning of SQL Injections. In *1st International Workshop on Data Systems Education,* 22–25.
SQLFiddle.com, URL: http://sqlfiddle.com, last access 20.01.2023.
SQL-Island.de, URL: https://sql-island.informatik.uni-kl.de last access 20.01.2023
W3schools SQL Tutorial, URL: https://www.w3schools.com/sql/, last access 20.1.2023.
Xinogalos, S., & Satratzemi M. (2022). The Use of Educational Games in Programming Assignments: SQL Island as a Case Study. *Applied Sciences,* 12 (13), 6563.

Acknowledgements

“aDBenture” has been supported by the KWF (https://kwf.at/) under grant number KWF-3520|31870|45842.