GRANT Privileges, REVOKE Risk: Safe and Scalable Teaching of Database Administration with Isolated Containers*

Andrzej Wójtowicz[1], Maciej Prill[1]

[1]Adam Mickiewicz University, Faculty of Mathematics and Computer Science, Uniwersytetu Poznańskiego 4, 61-614 Poznań, Poland

Abstract

Database administration is an essential but often underrepresented area in academic curricula due to the complexity and infrastructural demands of teaching elevated-privilege operations. Existing LMS-based autograders support only restricted SQL practice and cannot accommodate administrative tasks such as user management, backups, or performance tuning. To address these limitations, we developed an extension to CodeRunner in Moodle that enables full administrative interaction with Microsoft SQL Server, MySQL, and PostgreSQL using temporary, isolated Docker containers. This architecture provides each student with a safe, fully privileged environment for engaging in realistic, hands-on work. We deployed this solution in a new Database Administration course and evaluated it through surveys, task completion data, and performance tests. Students reported high satisfaction with the system's usability and realism, confirming the feasibility and effectiveness of the proposed approach, while also indicating areas for future refinement.

Keywords

database administration, student assessment, Microsoft SQL Server, MySQL, PostgreSQL

1. Introduction

Standard introductory database courses focus primarily on classical topics: SQL syntax covering queries (Data Query Language, DQL), data definition (Data Definition Language, DDL), and modification (Data Manipulation Language, DML), data modeling, transactions, triggers, and basic query optimization. In more advanced courses, some curricula also cover analytics topics such as window functions, data warehouses, or NoSQL systems [1, 2, 3, 4]. In contrast, hands-on experience with database internals or building a database engine from scratch is much rarer and typically limited to specialized systems courses [5, 6].

The general structure of database courses can be outlined in the ACM/IEEE-CS/AAAI program recommendations *Computer Science Curricula 2023* [7]. Based on them, it is clear that apart from cases where the purpose of the course is to create a simplified database engine, most of the classes focus on teaching how to use databases from a regular user's perspective.

In computer science education, autograders play a significant role, allowing instructors to automatically evaluate hundreds of small programming tasks while providing students with the ability to practice independently, outside of class hours. Most existing autograders were designed with the end-user in mind, i.e., someone who submits queries but does not manage the database [8, 9, 10]. As a result, issues related to data security, administration, and configuration of database systems are rarely taught in practice. In this paper, we describe a practical approach to overcoming this limitation by enabling full administrative access for students in a secure, isolated environment that supports hands-on learning of real-world database administrator (DBA) tasks.

DataEd'26: 5th International Workshop on Data Systems Education

* Published in the Proceedings of the Workshops of the EDBT/ICDT 2026 Joint Conference (March 24-27, 2026), Tampere, Finland

andre@amu.edu.pl (A. Wójtowicz); mprill@wmi.amu.edu.pl (M. Prill)

0000-0003-1385-6572 (A. Wójtowicz); 0009-0009-6280-1843 (M. Prill)

Moreover, the preparation of a database autograder itself is a challenge. In the case of tasks that require assigning temporary administrator rights to a student, the problem becomes even more difficult. A completely different architecture is needed, allowing for the safe execution of commands in an isolated environment. Administrative tasks are not limited to SQL alone; they often require operating system shell support. Unlike lightweight, embedded databases (such as SQLite), typical administrative environments rely on online, client-server systems.

Practical teaching of security issues is becoming particularly important as the IT industry puts more focus on them. According to the WEF and Accenture report [11], 66% of organizations see the impact of artificial intelligence on security, 72% notice an increase in the number of cyberthreats, and two-thirds of companies report a lack of employee skills to meet their data security requirements. Meanwhile, the OECD report [12] shows that the demand for cybersecurity specialists in Europe is growing several times faster than in other professions. As Kohnke points out [13], database environments remain a critical attack vector, and their protection requires specialized knowledge in configuration, auditing, access control, encryption, and cross-platform security management. Furthermore, the issue is of such strategic importance that national agencies, including the U.S. Department of Defense and the U.K. Department for Work and Pensions, have published formal standards and reference designs mandating hardening, access controls, encryption, and secure DevSecOps practices for database systems [14, 15]. Additionally, the U.S. Federal CIO Council emphasizes that *"data management is critical to making Zero Trust a reality"*. It prescribes comprehensive measures such as data categorization, encryption, identity-centric access control, continuous monitoring, and auditing across the data life-cycle to secure databases within a zero-trust architecture [16].

To validate the relevance of such a course for DBAs, we surveyed first-year computer science students ($n = 36$, multiple-choice question). Half of them (50%) expressed interest in a dedicated database administration course – more than any other higher level database topic SQL and query optimization (39%), implementation of a database engine (31%) or data warehouses (19%).

In this paper, we want to provide instructors with a tool that will make it easier to create courses in the field of database administration, combining the industry's growing needs with students' real interest. Our solution is based on the Moodle (https://moodle.org) Learning Management System (LMS) extended with the CodeRunner plugin [17]. This open-source autograder uses an external Jobe sandbox server (https://github.com/trampgeek/jobe), enabling safe compilation and execution of programs in multiple languages. Although CodeRunner was originally designed for programming tasks, its flexibility also extends to functions needed for teaching database administration. CodeRunner is not limited to Moodle and can be integrated with any platform through the Learning Tools Interoperability (LTI) standard.

In the following parts of the paper, we present the work related to our contribution (Section 2), describe the design and implementation of our solution (Section 3), the results of its evaluation (Section 4), limitations and plans for further development (Section 5), and the conclusions (Section 6). We hope this paper can serve as a practical reference for educators looking to introduce realistic and secure database administration exercises in their curriculum.

2. Related Work

Although DCL (Data Control Language) commands such as GRANT and REVOKE are included in official programming guidelines, they are rarely used in teaching practice in database courses [18]. Several reports on experiences with teaching database administration can be found in literature, but almost all of them are based on the Oracle system. For example, Mata-Toledo et al. presented a sample syllabus for a course based on Oracle [19]. Thomas and Udoh described a more extensive course that included significant application components but lacked specialized tools for teaching administrative tasks [20, 21]. Simmonds provided students with virtual machines (VMs), but also without using integrated teaching tools [22]. On the other hand, Mir described a short, two-week module dedicated to security in Microsoft SQL Server without dedicated technical support [23].

The high interest in the Oracle-based course reflects its historical popularity in academic environments.

According to the DB-Engines ranking for 2025, Oracle is the most popular database engine; however, the next three are MySQL, Microsoft SQL Server, and PostgreSQL [24]. Interestingly, the annual Stack Overflow survey results show that these three systems are among the leading database engines of the client-server type used by developers [25]. These discrepancies may stem from long-term enterprise adoption of Oracle, where large organizations invested early in robust vendor support. In contrast, newer deployments often favor free and open-source alternatives.

Autograders have been used in computer science and computing education for many years [8]. The latest review of available solutions [9] shows that instructors prefer tools that provide high flexibility and the possibility of integration with other systems. The most commonly used evaluation methods are approaches based on analyzing the program's functionality or output. The necessity of further development of existing solutions was also emphasized, to better support educational goals and the growing needs of teachers.

In the realm of SQL instruction, work on autograders has a long tradition [10, 26, 27]. Intensive research is also being conducted on methods of assessing the correctness of solutions, not only in terms of the final result, but also in terms of the language concepts used. For example, whether the student applied the required SQL concept in the solution is analyzed [28], and sometimes artificial intelligence techniques are used for this purpose [29]. Other studies focus on the methods of allocating partial points [30]. The effectiveness of formative assessment using practice tests has also been demonstrated: an observational study leveraging an SQL autograder showed a causal link between engagement with ungraded practice tests and improved performance on graded assessments [31].

Some autograders provide integration with LMS [32]. The CodeRunner plugin for Moodle [17] is particularly popular, and in its basic version only supports SQLite. Recent work has expanded it to support client-server engines such as MySQL, Microsoft SQL Server, or PostgreSQL [28], and DuckDB [33].

From the perspective of preparing tasks related to database administration, one of the main difficulties is the lack of support for DCL commands such as GRANT or REVOKE. Their use requires elevated privileges during evaluation, which generates security risks on the server side that executes the student-provided code. Additionally, many administrative tasks, e.g., creating backups, go beyond the scope of SQL and require running scripts in the system shell. Solution described in [28] uses containerization only for the permanent hosting of a shared database server, not in a dynamic and isolated manner for each test case of a given SQL task. As a result, it does not allow for granting of temporary administrative rights to students. From a technical standpoint, VMs and containers are the best solution for such educational applications. The literature suggests that VMs are well-suited for educational activities, while containers are more suitable for autograder environments [34].

Despite recent efforts, no existing solution offers full administrative control in a secure, scalable manner for teaching purposes. So far, no integrated, multi-vendor, open-source environment supports the creation of database administration tasks in autograders. The solution we propose in this paper fills this gap.

3. Design and Implementation

Teaching database administration in a scalable, safe, and realistic way requires architectural choices that go beyond traditional autograding setups. Our system provides full administrative privileges to students in temporary, isolated environments that mimic real-world deployments. In this section, we describe the architecture of our system and the way tasks are defined within the course. The source code of the solution has been made available in a public Git repository (https://github.com/andre-wojtowicz/coderunner-dba-dataed2026) under the MIT license.

3.1. System Design

Our solution extends the architecture proposed in [17, 35] based on integrating Moodle with the CodeRunner plugin and the Jobe sandbox server. In the traditional model, the student's answer is

combined with a question template code and data from test cases. Afterwards, the combined code is sent to the Jobe server, which runs it in a secure sandbox environment and returns the result to Moodle for grading. Our extension involves adding special question templates and an extra layer on Jobe that allows the utilization of Docker images containing appropriate database systems and dynamic creation of containers launched only for the duration of a test case evaluation, starting from a predefined state.

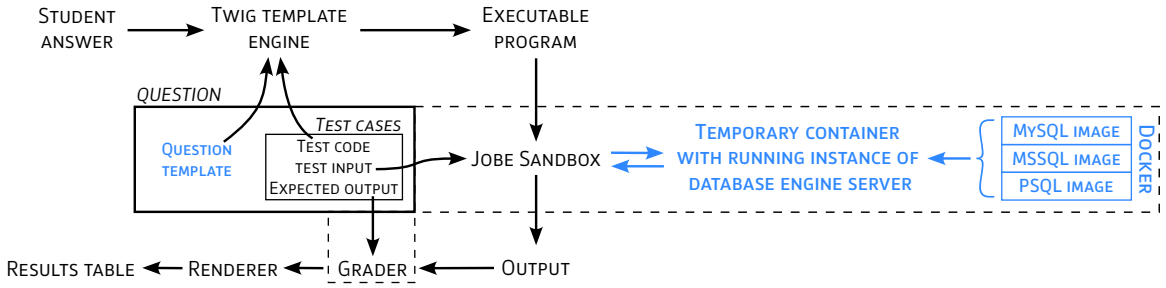

Figure 1: The extended CodeRunner system architecture with dynamic running of Docker containers for safe execution of administrative routines in an isolated environment. Adapted from [35].

The solution's architecture is illustrated in Figure 1. Each test case is run in a separate container for a given task, in which a full database server operates with available administrator privileges. This approach allows the student to execute DCL commands and administrative operations in the shell, such as creating backups or restoring databases. Our solution significantly extends the one presented in [28], where a common, containerized database server operated for the entire course, and only separate databases were dynamically created for testing DQL, DML, or DDL queries.

Dynamic creation of an entire database server for each test case allows for granting administrative privileges within the container, which enables safe testing of administrative scenarios. The isolated container ensures a strong security boundary: even if a student misuses administrative privileges, the risk is contained within the temporary sandbox, with no access to host resources or other student environments. Additionally, Jobe enhances security through Linux *cgroups*, which limit resource usage for each container. Moreover, our solution is scalable, as the number of containers launched simultaneously depends on the number of Jobe workers and available hardware resources, such as CPU and RAM.

3.2. Question Design

The system is based on CodeRunner question templates. A separate template has been prepared for each supported database system (MySQL, Microsoft SQL Server, PostgreSQL) to define questions. The questions are divided into two main types. The first type involves writing SQL code, which is executed using client tools (e.g., the `mysql` command in the shell). Figures 2 and 4 present an example definition and execution of this type of question, respectively. The second type is system tasks, which involve completing the appropriate parameters of shell commands, such as `mysqlpump`. Figures 3 and 5 show an example definition and execution of this type of question, respectively.

Each template contains an API implemented in Python, described in the template documentation. The instructor can use the API in the "Extra template data" section for a given test case to prepare a server instance, preprocess the student's response, and execute the code to check the correctness of the solution. Attaching additional files (e.g., SQL code creating databases) to the test case is also possible. The correctness assessment is made by comparing the textual result of the program's operation with the expected pattern.

CodeRunner automatically generates the "Expected output" field based on the correct answer entered by the instructor in the "Answer" field, which significantly streamlines the question creation process. It is recommended that the instructor enter sample correct answers, as it allows for collective testing of all questions in the course in the event of an update to the Docker image (e.g., to a newer version of the database system), to check whether the way the answers are generated has changed.

Although the system's main purpose is to enable safe practice of administrative commands, it also

supports classic SQL commands from the DQL, DML, and DDL groups. They are often necessary for administrative tasks, such as checking whether the user has the appropriate permissions.

Since each task is defined as Python code, more advanced mechanisms can be used to evaluate correctness. For example, before displaying the results on the screen, one can use regular expressions to replace dynamic values (e.g., current date and time) with fixed placeholders. This approach allows for an effective comparison of the program's output with the expected text pattern. The flexibility of processing intermediate and final results makes it possible to fully adapt the evaluation mechanism to

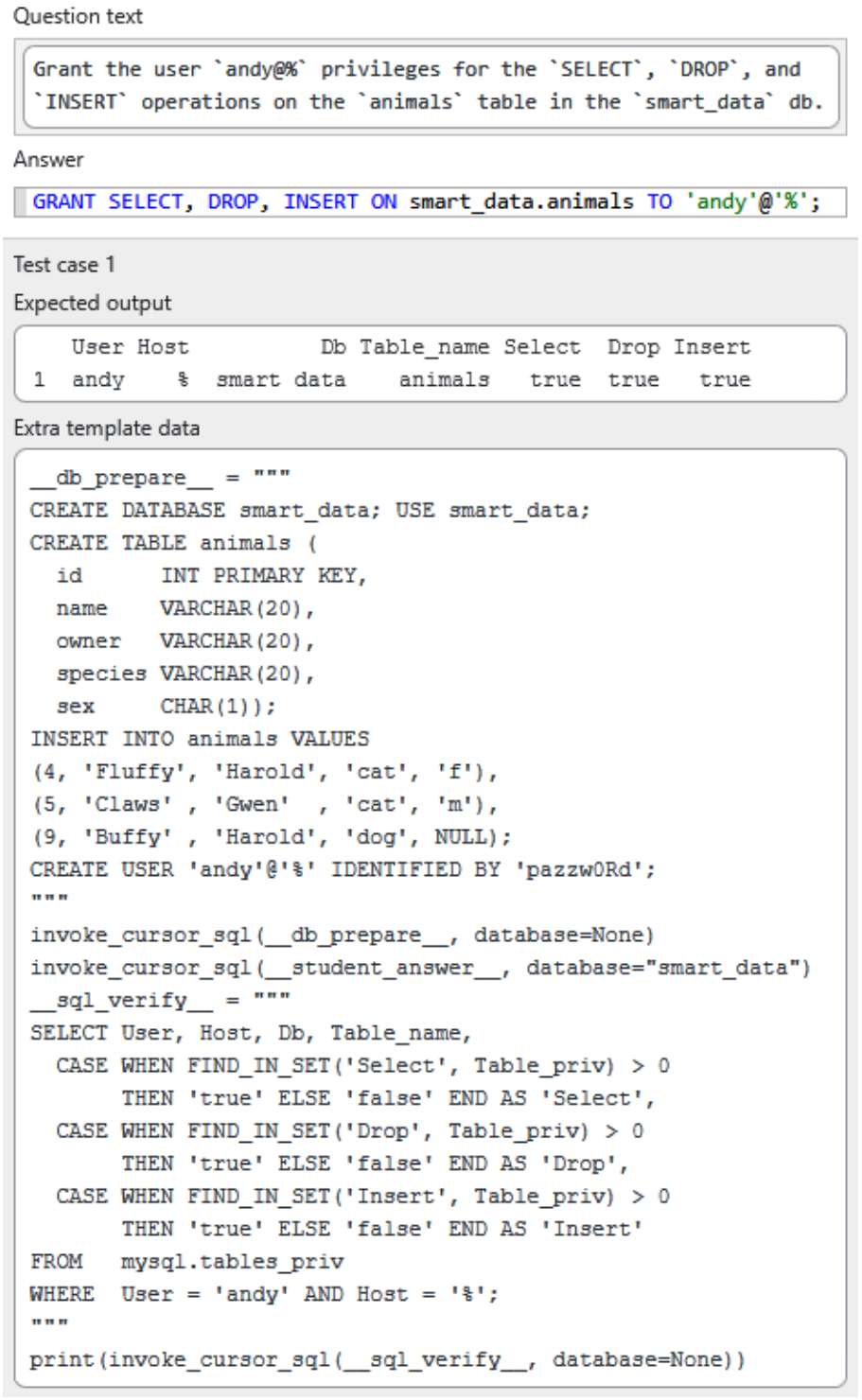

Figure 2: The sample definition of a SQL DCL question, such as GRANT, including a model answer and test case data.

Question text

```
Use `mysqlpump` to create a copy of the data stored in the tables
of the `academy` database whose names start with `ye`.
```

Answer

```
mysqlpump --default-parallelism=1 --skip-tz-utc
--skip-set-charset --skip-watch-progress
--include-databases=academy --include-tables=ye%
--no-create-db --no-create-info --skip-routines
```

Test case 1

Expected output

```
-- comment
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS;
SET @OLD_SQL_MODE=@@SQL_MODE;
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
SET @@SESSION.SQL_LOG_BIN= 0;
INSERT INTO `academy`.`ye19` VALUES (1);
INSERT INTO `academy`.`ye20` VALUES (1);
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
SET SQL_MODE=@OLD_SQL_MODE;
```

Extra template data

```
__student_answer__ = __student_answer__.strip()
if not __student_answer__.startswith('mysqlpump'):
    sys.stderr.write("Command must start with 'mysqlpump'\n")
elif __student_answer__.count('\n') > 0:
    sys.stderr.write("Multiline command attempt\n")
else:
    __server_prepare__ = """
    INSTALL PLUGIN auth_socket SONAME 'auth_socket.so';
    SET SESSION sql_log_bin = OFF; SOURCE academy.sql;
    SET PERSIST local_infile = ON;
    """
    invoke_commandline_sql(__server_prepare__, database=None)
    clt = ['/usr/bin/mysqlpump'] + COMMAND_LINE_TOOL[1:4]
    clt = clt + __student_answer__.split()[1:]
    cmd_output = invoke_commandline_sql(None,
                 database=None, command_line_tool=clt)
    import re
    print(re.sub("^--.*", "-- comment", cmd_output,
                 flags=re.MULTILINE))
```

Support files

academy.sql 6/02/25, 15:09 654 bytes text/x-sql

Figure 3: The sample definition of a shell routine question that involves calling an administrative command, such as `mysqlpump`.

educational needs.

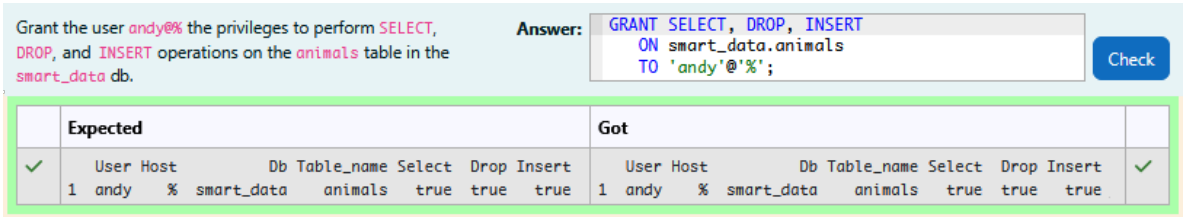

Figure 4: The results of evaluation of the SQL instruction GRANT.

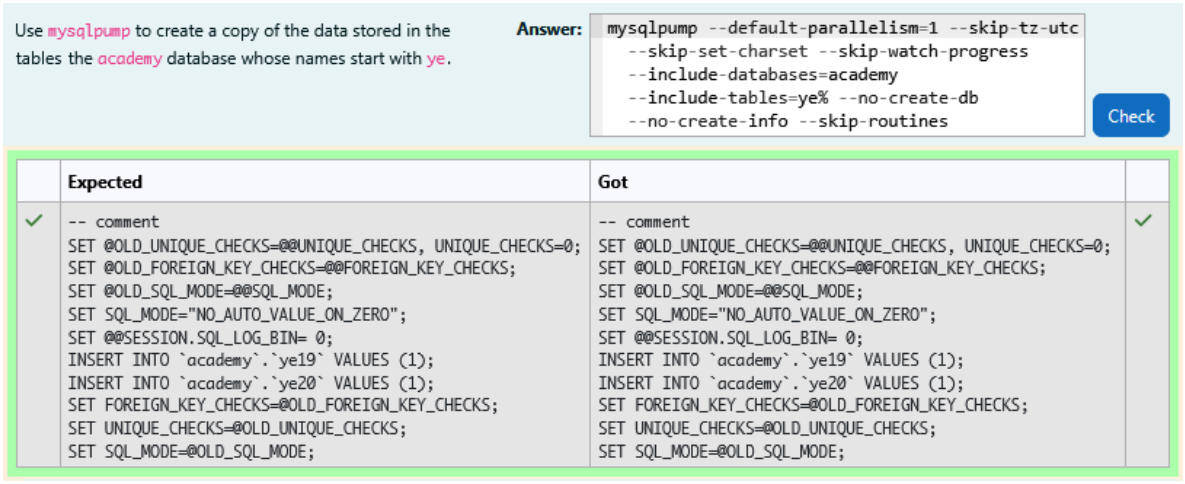

Figure 5: The results of evaluation of the shell command `mysqlpump`.

4. Evaluation

We implemented the proposed solution as part of the Database Administration elective course designed for students in the sixth semester of engineering studies in computer science. The course spanned 15 weeks and covered administrative topics for three database systems: MySQL, Microsoft SQL Server, and PostgreSQL. Five analogous but thematically independent modules have been developed for each of these engines. Despite similar thematic structures, each system has its configuration and syntactic nuances, which require separate discussion and practice. Including multiple database engines helps prevent educational vendor lock-in by ensuring that students develop broadly applicable administrative skills rather than becoming narrowly specialized in a single technology. The topics covered: (1) *server installation and client tools*; (2) *users, roles, and permissions management*; (3) *password and data schema management*; (4) *file operations, backups and recovery*; (5) *data partitioning*.

Each weekly problem set was allocated 2 hours of lab time, and all autograded tasks together accounted for 75% of the final grade (the remaining 25% came from a project).

We provided students with a VM running Ubuntu 22.04 system, enabling flexible local work. The database engines were based on MySQL 8.0.41, MSSQL 2019-CU30, and PostgreSQL 14.16. After completing the exercises and graded tasks locally, students could test and send their final solutions in Moodle for verification with the autograder.

The course was attended by 17 students, 16 of whom completed the final survey. The questionnaire covered both the content and structure of the course, as well as the technical solution used for task verification. We briefly summarize the responses below.

Several aspects of the course proved particularly effective. In the post-course survey (six-level Likert scale; 1 – very negative; 6 – very positive), students were generally satisfied with the course structure and content ($\mu = 5.13$, $\sigma = 0.72$). The subject of the classes was rated as interesting and useful ($\mu = 4.94$, $\sigma = 1.12$). The difficulty level of the questions was rated as moderate ($\mu = 3.31$, $\sigma = 0.60$), which indicates a good balance between challenge and accessibility.

However, some areas still require further refinement based on task performance. In each of the fifteen

topics, students solved several autograded tasks, and completing a topic awarded up to 5 course points (0–5). Based on per-topic points, the lowest-scoring topics were *data partitioning* ($\bar{p} = 4.43$, $s_p = 1.52$) and *users, roles, and permissions management* ($\bar{p} = 4.28$, $s_p = 1.50$). The respondents suggested increasing the number of real-life scenarios in these topics. This is in line with the observation that the appropriate selection of database content and its usage context influences engagement during learning [36].

On the technical part of the course, the speed of the CodeRunner/Jobe environment (16 CPU cores and 64 GB of RAM) for each of the engines was rated highly; even though Microsoft SQL Server container execution takes at average more than two times longer than two other database engines (10.0 s vs 4.0 s), it did not affect the students' subjective assessment of the autograder's speed as they perceived them as balanced. In the open-ended question about using the provided practice VM, most students declared regular use, mainly to solve examples, test solutions, and work with course materials; in the context of integrating three different database engines, opinions were positive.

In a separate survey question regarding the quality and clarity of the CodeRunner/Jobe environment feedback (1 – very low, 6 – very high), the results suggest some need for further improvement ($\mu = 4.50$, $\sigma = 0.89$). Two students reported technical problems, mainly concerning the lack of clarity of feedback in cases of their incorrect solutions. These results are consistent with the findings indicating the importance of feedback quality in database education [30].

5. Limitations and Further Work

Despite positive results, the current system has limitations that motivate further work.

Firstly, the system does not provide automated, context-aware hints for incorrect submissions, which can frustrate students [30]. Future work could integrate syntax/semantics analysis and optionally RAG-enabled language models to generate targeted guidance [37].

The second limitation is the lack of support for partial grading, which could better reflect students' progress in complex tasks. There are already several promising technical proposals [38, 30], which can inspire further work on implementing this function in the system.

Currently, the system supports three popular database engines. However, there is a lack of support for Oracle, which is still significant in large enterprises. Incorporating the Oracle engine may prove challenging. The free version has noteworthy resource limits [39], and use of this engine in *spawn-container-per-test-case* manner may require a commercial license. Initial research found this server's start times unacceptably long for this use – further configuration tuning might solve this deficiency.

Questions are defined as separate, isolated tasks. Unlike Jupyter Notebook environments, where building a multi-stage solution in one go is possible, each question represents an independent environment launch. Suppose an instructor wants to provide a running example through teaching material. In that case, this may require duplicating setup code across multiple questions, potentially reducing maintainability and instructional clarity.

Another limitation is that most tasks validate procedures in a controlled environment, whereas real DBA work often includes diagnosing failures and handling unexpected or adversarial user behavior. Future work could add fault-injection and "misbehaving user" scenarios inside containers to increase realism while preserving isolation.

The system is also not adapted to check tasks requiring long-term data processing (e.g., operations lasting a minute or longer), which would require queuing and task scheduling. Moreover, due to the design based on single instances of database servers, the current architecture does not support scenarios requiring communication of multiple servers (e.g., tasks related to data replication or distributed queries). Their implementation would require the dynamic creation of mini-networks of containers, potentially significantly extending the time it takes to evaluate a single test case.

Finally, having a safe and scalable environment that records real student submissions enables broader research into how students learn administrative database concepts. The system could identify common misconceptions related to DCL operations, expanding on prior work in this area [40, 41], particularly now that AI-based code-generation tools have become increasingly accessible [42].

6. Conclusions

We presented a scalable, open-source extension to the CodeRunner autograder that enables safe, automated evaluation of database administration tasks involving elevated privileges, such as DCL commands. The tool is publicly available, enabling easy adoption and further development by instructors and institutions.

A key system innovation is the per-test-case containerization, granting students full administrative rights in a controlled and temporary environment. This allows for practical training in administrative tasks. Replacing the container image is sufficient to update database versions.

The course evaluation showed that the students' perception of the system's operation, and speed is generally positive and sufficient for effective learning. The obtained survey data and task performance metrics are a valuable source of information for course coordinators, as they can help identify more difficult topics and adjust the syllabus and teaching materials accordingly.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: Grammar and spelling check. After using this tool, the authors reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] H. Garcia-Molina, J. D. Ullman, J. Widom, Database Systems: The Complete Book, 2 ed., Prentice Hall Press, USA, 2008.

[2] R. Elmasri, S. B. Navathe, Fundamentals of Database Systems, 7th ed., Pearson, USA, 2015.

[3] A. Silberschatz, H. Korth, S. Sudarshan, Database Systems Concepts, 7 ed., McGraw-Hill Education, USA, 2019.

[4] D. Miedema, T. Taipalus, V. V. Ajanovski, A. Alawini, M. Goodfellow, M. Liut, S. Peltsverger, T. Young, Data Systems Education: Curriculum Recommendations, Course Syllabi, and Industry Needs, in: 2024 Working Group Reports on Innovation and Technology in Computer Science Education, ITiCSE 2024, ACM, 2025, p. 95–123. doi:10.1145/3689187.3709609.

[5] E. Sciore, Database Design and Implementation: Second Edition, Springer International Publishing, Cham, Switzerland, 2020. doi:10.1007/978-3-030-33836-7.

[6] M. S. Rehman, A. Elmore, R. C. Fernandez, Not-So-Bitter Pill to Swallow: Slipstreaming Memory Safe Programming via Rust as part of a Database Systems Course, in: Proceedings of the 4th International Workshop on Data Systems Education: Bridging education practice with education research, DataEd '25, Association for Computing Machinery, New York, NY, USA, 2025, pp. 41–46. doi:10.1145/3735091.3737532.

[7] A. N. Kumar, R. K. Raj, S. G. Aly, M. D. Anderson, B. A. Becker, R. L. Blumenthal, E. Eaton, S. L. Epstein, M. Goldweber, P. Jalote, D. Lea, M. Oudshoorn, M. Pias, S. Reiser, C. Servin, R. Simha, T. Winters, Q. Xiang, Computer Science Curricula 2023, Association for Computing Machinery, New York, NY, USA, 2024. doi:10.1145/3664191.

[8] J. C. Paiva, J. P. Leal, A. Figueira, Automated Assessment in Computer Science Education: A State-of-the-Art Review, ACM Transactions on Computing Education 22 (2022). doi:10.1145/3513140.

[9] B. Ruth, J. R. Hott, Auto-grading in Computing Education: Perceptions and Use, in: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1, SIGCSETS 2025, Association for Computing Machinery, New York, NY, USA, 2025, p. 1008–1014. doi:10.1145/3641554.3701900.

[10] J. D. Ullman, Gradiance On-Line Accelerated Learning, in: Proceedings of the Twenty-Eighth Australasian Conference on Computer Science - Volume 38, Australian Computer Society, Inc., AUS, 2005, p. 3–6.

[11] World Economic Forum, Accenture, Global Cybersecurity Outlook 2025, Insight report, World Economic Forum, Geneva, Switzerland, 2025. URL: https://www.weforum.org/publications/global-cybersecurity-outlook-2025/.
[12] OECD, Building a Skilled Cyber Security Workforce in Europe: Insights from France, Germany and Poland, OECD, Paris, France, 2024. doi:10.1787/3673cd60-en.
[13] A. Kohnke, Managing Security Across Disparate Database Technologies, ISACA Journal 2022 (2022) 32–37. URL: https://www.isaca.org/resources/isaca-journal/issues/2022/volume-4/managing-security-across-disparate-database-technologies.
[14] Chief Information Officer, DoD Enterprise DevSecOps Reference Design, Technical Report, U.S. Department of Defense, 2019. URL: https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf.
[15] Chief Security Office, Security standard SS-005: Database Management System, Technical Report, U.K. Department for Work and Pensions, 2026. URL: https://assets.publishing.service.gov.uk/media/69806ab9ec71a16669612daf/ss_005.pdf.
[16] U.S. Federal CIO Council, Federal Zero Trust Data Security Guide, Technical Report, CISO & CDO Councils, 2024. URL: https://www.cio.gov/assets/files/Zero-Trust-Data-Security-Guide_Oct24-Final.pdf.
[17] R. Lobb, J. Harlow, Coderunner: A Tool for Assessing Computer Programming Skills, ACM Inroads 7 (2016) 47–51. doi:10.1145/2810041.
[18] T. Taipalus, V. Seppänen, SQL Education: A Systematic Mapping Study and Future Research Agenda, ACM Trans. Comput. Educ. 20 (2020). doi:10.1145/3398377.
[19] R. A. Mata-Toledo, C. A. Reyes-Garcia, A model course for teaching database administration with personal Oracle 8i, J. Comput. Sci. Coll. 17 (2002) 125–130.
[20] R. Thomas Mason, A Database Practicum for Teaching Database Administration and Software Development at Regis University, Journal of Information Technology Education: Innovations in Practice 12 (2013) 159–168. doi:10.28945/1858.
[21] E. Udoh, Teaching database in an integrated Oracle environment, SIGCSE Bulletin 38 (2006) 71–74. doi:10.1145/1189136.1189174.
[22] T. Simmonds, Teaching database administration in the world of Big Data and Small Budgets, in: Proceedings of TLAD 2013: 11th International Workshop on Teaching, Learning and Assessment of Databases, TLAD, HEA, Birmingham, United Kingdom, 2013, pp. 1–7. URL: http://gala.gre.ac.uk/id/eprint/19913/.
[23] M. Hasan, J. Elarde, B. Bruster, Teaching Database Security in an Undergraduate Database Administration Course Serving Computer Science, Information Technology and Cybersecurity Students, in: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2, SIGCSE 2023, Association for Computing Machinery, New York, NY, USA, 2023, p. 1406. doi:10.1145/3545947.3576350.
[24] DB-Engines, DB-Engines Ranking, https://db-engines.com/en/ranking, 2025.
[25] Stack Overflow, Stack Overflow Developer Survey 2024: Technology – Databases, https://survey.stackoverflow.co/2024/technology#1-databases, 2024.
[26] C. Kleiner, C. Tebbe, F. Heine, Automated Grading and Tutoring of SQL Statements to Improve Student Learning, in: Proceedings of the 13th Koli Calling International Conference on Computing Education Research, ACM, New York, NY, USA, 2013, p. 161–168. doi:10.1145/2526968.2526986.
[27] B. Chandra, A. Banerjee, U. Hazra, M. Joseph, S. Sudarshan, Automated Grading of SQL Queries, in: 2019 IEEE 35th International Conference on Data Engineering (ICDE), IEEE, Macau, China, 2019, pp. 1630–1633. doi:10.1109/icde.2019.00159.
[28] A. Wójtowicz, M. Prill, Relational Database Courses with CodeRunner in Moodle: Extending SQL Programming Assignments to Client-Server Database Engines, in: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1, SIGCSETS 2025, Association for Computing Machinery, New York, NY, USA, 2025, p. 1239–1245. doi:10.1145/3641554.3701830.

[29] L. Xiang, SQL Query Evaluation with Large Language Model and Abstract Syntax Trees, in: Proceedings of the 55th ACM Technical Symposium on Computer Science Education, SIGCSE 2024, Association for Computing Machinery, New York, NY, USA, 2024, p. 1890. doi:10.1145/3626253.3635408.

[30] C. Kleiner, F. Heine, Enhancing Feedback Generation for Autograded SQL Statements to Improve Student Learning, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, ITiCSE 2024, Association for Computing Machinery, New York, NY, USA, 2024, p. 248–254. doi:10.1145/3649217.3653579.

[31] A. Wójtowicz, A. Stachowiak, E. Pankowska, Train Hard, Score High: The Impact of Practice Tests on SQL Learning, in: Proceedings of the 4th International Workshop on Data Systems Education: Bridging education practice with education research, DataEd '25, Association for Computing Machinery, New York, NY, USA, 2025, pp. 35–40. doi:10.1145/3735091.3737573.

[32] K. Danutama, I. Liem, Scalable Autograder and LMS Integration, Procedia Technology 11 (2013) 388–395. doi:https://doi.org/10.1016/j.protcy.2013.12.207, 4th International Conference on Electrical Engineering and Informatics, ICEEI 2013.

[33] L. Behme, G. Dusella, R. P. Lemaitre, A. Borusan, V. Markl, Teaching Large-Scale Data Management to Large Cohorts of Undergraduate Students, in: Proceedings of the 4th International Workshop on Data Systems Education: Bridging education practice with education research, DataEd '25, Association for Computing Machinery, New York, NY, USA, 2025, p. 1–5. doi:10.1145/3735091.3737571.

[34] G. Tuparov, Sandbox Databases in SQL University Courses – State of the Art, in: Proceedings of the 15th International Conference Education and Research in the Information Society, CEUR, Plovdiv, Bulgaria, 2022, pp. 82–89. URL: http://ceur-ws.org/Vol-3372/paper09.pdf.

[35] R. Lobb, The Architecture of CodeRunner, https://trampgeek.github.io/moodle-qtype_coderunner/#the-architecture-of-coderunner, 2025.

[36] T. Taipalus, D. Miedema, E. Aivaloglou, Engaging Databases for Data Systems Education, in: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1, ITiCSE 2023, Association for Computing Machinery, New York, NY, USA, 2023, p. 334–340. doi:10.1145/3587102.3588804.

[37] H. Kumar, M. Reza, J. Thomas-Mitchell, I. Musabirov, L. Zhang, M. Liut, Understanding Help-Seeking Behavior of Students Using LLMs vs. Web Search for Writing SQL Queries, in: Proceedings of the 4th International Workshop on Data Systems Education: Bridging education practice with education research, DataEd '25, Association for Computing Machinery, New York, NY, USA, 2025, pp. 23–28. doi:10.1145/3735091.3737569.

[38] M. Fabijanić, I. Mekterović, Partial SQL Query Assessment, in: 2023 46th MIPRO ICT and Electronics Convention (MIPRO), IEEE, Opatija, Croatia, 2023, pp. 1317–1322. doi:10.23919/MIPRO57284.2023.10159706.

[39] Oracle, Database 26ai Free, https://www.oracle.com/database/free/, 2025.

[40] D. Miedema, E. Aivaloglou, G. Fletcher, Identifying SQL Misconceptions of Novices: Findings from a Think-Aloud Study, in: Proceedings of the 17th ACM Conference on International Computing Education Research, ICER 2021, Association for Computing Machinery, New York, NY, USA, 2021, p. 355–367. doi:10.1145/3446871.3469759.

[41] D. Miedema, G. Fletcher, E. Aivaloglou, Expert Perspectives on Student Errors in SQL, ACM Trans. Comput. Educ. 23 (2022). doi:10.1145/3551392.

[42] A. Laaksonen, K. Korpimies, M. Luukkainen, Trends in Students' SQL Queries in the Era of Generative AI, in: Proceedings of the 25th Koli Calling International Conference on Computing Education Research, Koli Calling '25, Association for Computing Machinery, New York, NY, USA, 2025, pp. 1–7. doi:10.1145/3769994.3770024.