UDC 004.415.2:004.451.88
DOI https://doi.org/10.15589/znp2026.1(503).2.1

METHODS FOR TESTING NON-VISUAL DATA USING SQL QUERIES IN BACKEND SYSTEMS

МЕТОДИ ТЕСТУВАННЯ НЕВІЗУАЛЬНИХ ДАНИХ ЗА ДОПОМОГОЮ SQL-ЗАПИТІВ У БЕКЕНД-СИСТЕМАХ

Kateryna Yu. Horshchar
kate.horshchar@gmail.com
ORCID: 0009-0002-4008-6278

К. Ю. Горщар,
магістр

National University of Water and Environmental Engineering, Rivne
Національний університет водного господарства та природокористування, м. Рівне

Abstract. The *purpose* of the article. The purpose of the article is to study methods for testing non-visual data in backend systems using SQL queries to ensure comprehensive validation of the functionality of information systems at the database level. The study focuses on the use of SQL as a tool for verifying the creation, updating, and deletion of data, validating business logic, identifying orphan records, and ensuring the integrity of data that is not displayed in the user interface.
Scientific novelty. The scientific novelty lies in the systematization and classification of SQL testing methods for Backend QA and Database Testing, including data preparation (preconditions), record creation verification, reference integrity validation, orphan record detection, and business logic control. An architectural scheme for integrating SQL into automated testing processes is proposed, which considers the multi-level nature of backend data validation. The study also emphasizes the automation of SQL query generation through modern methods such as machine learning models and emphasizes the importance of SQL query security to avoid vulnerabilities such as SQL injection.
Results. A classification of SQL testing methods has been developed, which includes data preparation, validation of create/update operations, orphan record detection, business logic validation, reference integrity control, and data consistency checking. An architectural scheme for integrating SQL into Backend QA is presented, which ensures consistent validation at all stages of the test cycle. Practical testing in Google Colab using SQLite demonstrated the effectiveness of SQL for checking record creation, orphan record detection (0 detected), business logic validation (0 errors), and reference integrity (100% valid references). The results confirm SQL's ability to detect critical errors that are not captured by UI testing.
Conclusions. SQL is a powerful tool for comprehensive backend data validation, allowing you to identify inconsistencies, orphan records, and errors in business logic that go unnoticed in traditional user interface testing. The systematic use of SQL through preconditions and postconditions provides a controlled test of data functionality and integrity. Integration of SQL into automated tests requires specialized solutions for connecting to the database and securely executing queries. *Prospects for further research* include automating the generation of SQL tests based on the database structure and business logic, as well as adapting methods for industry-specific information systems.
Key words: database testing; SQL queries; backend testing; automated testing; business logic testing; SQL integration; data integrity.

Анотація. *Метою статті* є дослідження методів тестування невізуальних даних у бекенд-системах за допомогою SQL-запитів для забезпечення комплексної валідації функціональності інформаційних систем на рівні бази даних. Дослідження зосереджується на використанні SQL як інструменту для перевірки створення, оновлення та видалення даних, валідації бізнес-логіки, виявлення сирітських записів і забезпечення цілісності даних, що не відображаються в користувацькому інтерфейсі.
Наукова новизна. Наукова новизна полягає в систематизації та класифікації методів SQL-тестування для Backend QA і Database Testing, що включають підготовку даних (preconditions), перевірку створення записів, валідацію референсної цілісності, виявлення сирітських записів і контроль бізнес-логіки. Запропоновано архітектурну схему інтеграції SQL у процеси автоматизованого тестування, яка враховує багаторівневий характер валідації бекенд-даних. Дослідження також акцентує на автоматизації генерації SQL-запитів через сучасні методи, такі як моделі машинного навчання, та підкреслює важливість безпеки SQL-запитів для уникнення вразливостей, таких як SQL-ін'єкції.

Результати. Розроблено класифікацію методів SQL-тестування, що охоплює підготовку даних, перевірку операцій створення/оновлення, виявлення сирітських записів, валідацію бізнес-логіки, контроль референсної цілісності та перевірку консистентності даних. Представлено архітектурну схему інтеграції SQL у Backend QA, яка забезпечує послідовну валідацію на всіх етапах тестового циклу. Практичне тестування в Google Colab з використанням SQLite продемонструвало ефективність SQL для перевірки створення записів, виявлення сирітських записів (0 виявлено), валідації бізнес-логіки (0 помилок) та референсної цілісності (100% валідних референсів). Результати підтверджують здатність SQL виявляти критичні помилки, які не фіксуються при тестуванні через UI.

Висновки. SQL є потужним інструментом для комплексної валідації бекенд-даних, що дозволяє виявляти невідповідності, сирітські записи та помилки в бізнес-логіці, які залишаються непоміченими при традиційному тестуванні користувацького інтерфейсу. Систематичне використання SQL через preconditions і postconditions забезпечує контрольовану перевірку функціональності та цілісності даних. Інтеграція SQL в автоматизовані тести потребує спеціалізованих рішень для підключення до бази даних і безпечного виконання запитів. Перспективи подальших досліджень включають автоматизацію генерації SQL-тестів на основі структури бази даних і бізнес-логіки, а також адаптацію методів для галузевих інформаційних систем.

Ключові слова: тестування баз даних; SQL-запити; бекенд-тестування; автоматизоване тестування; перевірка бізнес-логіки; інтеграція SQL; цілісність даних.

PROBLEM STATEMENT

Modern information systems are characterized by a complex architecture, where the user interface displays only a small part of the data processed in backend systems. SQL is a powerful tool for checking backend data, capable of determining whether records have been created, whether the data meets the established requirements, and whether there are orphan records in the system. The data displayed in the user interface is only the tip of the iceberg of the overall information system, and errors may not be in visualization, but in incorrect database queries, incorrectly calculated aggregates, or residues from previous actions.

Traditional testing approaches that focus primarily on visual UI elements do not provide full validation of data at the database level. The problem is compounded by the fact that a significant portion of the data critical to the functioning of the system remains unrepresented in the visual representation, which creates gaps in test coverage and can lead to serious functional disruptions in the production environment due to incorrect operations with backend data.

ANALYSIS OF RECENT RESEARCH AND PUBLICATIONS

Research in the field of Database Testing and Backend QA demonstrate the growing interest of the scientific community in validation issues backend systems via SQL queries. Modern approaches cover a wide range of techniques from automatic query generation to verify record creation and update to comprehensive data integrity verification and orphan record detection, reflecting the complexity of the tasks of testing non-visual aspects of modern information systems.

Yun J., Lee S. [1] investigate the improvement of the efficiency of natural language to SQL query conversion through automatic generation of evidence base, which demonstrates the relevance of integrating SQL into automated testing processes to verify the creation of records after actions in the system and validate the compliance of data with business requirements. Ma P., Zhuang X., Xu C., Jiang X., Chen R., Guo J. [2] develop a model for training natural language to SQL conversion using reinforcement learning, which opens up new opportunities for automating the creation of test queries for validating business logic and checking relationships between tables in backend systems.

Xu K., Luo Y., Yuan Y., Yao AC-C. [3] investigate automated formal verification of backend systems using large language models, which emphasizes the importance of a systematic approach to validating non-visual system components to detect data that should not exist after delete or modify operations. Oriol X., Teniente E. [4] propose incremental verification of SQL statements in relational database management systems, which allows for the effective use of SQL in pre / post conditions to monitor compliance with business rules in real time and detect data integrity violations immediately after they occur.

Jyothi, Murthy TS [5] present a methodology for converting domain -specific queries into SQL using embedded data balancing techniques, demonstrating the potential of adapting SQL tools to specific Backend QA requirements for checking relationships between tables and validating complex business scenarios through automated queries. Nguyen D.-C., Ha M.-H., Do M.-T., Chen OT-C. [6] develop a lightweight graph– based SQL injection detection model convolutional neural networks, which indicates the need to ensure the security of SQL queries in Database test scenarios Testing when preparing prerequisites for validation backend -data.

Gong Y., Lei C., Qin X., Vaidya K., Narayanaswamy B., Kraska T. [7] create a comprehensive framework for detecting and correcting errors in text-to-SQL conversion, which confirms the complexity of ensuring the correctness of automatically generated test queries to check the state of backend data after operations via UI or API and detect inconsistencies in business logic. Kashyap A.,

Jana A. [8] conduct a comprehensive review of methods for detecting and preventing SQL injection attacks, which raises the issue of safe use of SQL in Backend QA to detect orphan records and validate data integrity without compromising the system.

Febrian DW, Huwae RB, Mardiansyah AZ [9] analyze vulnerabilities of SQL injections and other types of attacks in web applications of educational institutions, demonstrating the practical significance of the correct use of SQL queries in Database Testing to verify the compliance of backend data with system requirements and detect incorrect database queries. Pavlenko I., Boiko O., Mykolaiets D., Moskalenko O., Shrol T. [10] investigate the development of technologies in Ukrainian educational settings, which expands the context of the use of SQL testing in educational information systems for the validation of data creation, update and deletion operations through automated and manual approaches.

Pasichnyi R., Serhieiev V., Shevchenko S., Petrukha N., Hryvnak B. [11] consider the digital transformation of higher education as a driver of Ukraine's integration into the European educational space, which emphasizes the need to ensure the quality of information systems through a comprehensive Database Testing, including validation of backend data at the database level to detect incorrectly calculated aggregates and residuals after previous actions in the system.

The analysis shows that current research is aimed at creating automated methods for generating and executing SQL queries for Backend QA, ensuring the security of test procedures, and adapting SQL testing to specific industry requirements. At the same time, there is a need to develop a comprehensive approach to integrating SQL validation into traditional Database processes. Testing to ensure full coverage of non-visual aspects of backend systems through the systematic use of preconditions and postconditions.

SEPARATION OF PREVIOUSLY UNRESOLVED PARTS OF THE GENERAL PROBLEM

Despite a significant amount of research in the field of software testing, approaches to the systematic use of SQL queries specifically for validating non-visual data of backend systems that are not directly reflected in the user interface remain underdeveloped. Methods for integrating SQL testing into automated Backend QA processes, as well as mechanisms for formalized verification of business logic, referential integrity, and data consistency in real information systems, require further scientific substantiation.

The purpose of the article is – research into methods for testing non-visual data using SQL queries in backend systems to ensure comprehensive validation of the functionality of information systems at the database level.

METHODS, OBJECT AND SUBJECT OF THE STUDY

The study uses methods of analysis and generalization of scientific sources, comparative analysis of approaches to SQL testing, as well as methods of modeling and practical experiment based on the SQLite database. The object of the study is the process of testing backend systems and databases within the framework of software quality assurance. The subject of the study is methods and tools for validating non-visual data using SQL queries, aimed at checking business logic, referential integrity and data consistency.

PRESENTATION OF THE MAIN MATERIAL

Database methodology Non-visual data testing using SQL queries is based on a systematic approach to validating the database state before, during, and after functional operations in the system. Yun J., Lee S. [1] demonstrate that automatic generation of evidence base for SQL queries can significantly improve the efficiency of Backend QA to verify the creation of records after user actions via API or user interface, especially when working with complex multi-level queries to validate the compliance of backend data with established requirements.

SQL as a powerful tool for backend data validation allows for comprehensive validation by checking data creation, update, and deletion after actions in the UI or API, which is not available when testing only through the graphical interface. Ma P., Zhuang X., Xu C., Jiang X., Chen R., Guo J. [2] emphasize the importance of using trained models to generate SQL queries, which allows creating more complex test scenarios for validating business logic and detecting hidden inconsistencies in backend data, including incorrectly calculated aggregates and residues after previous operations.

Preconditions in Database Testing via SQL queries ensures data preparation before executing test scripts, creating a controlled environment for validation backend functionality. Xu K., Luo Y., Yuan Y., Yao AC-C. [3] emphasize the importance of formal verification of backend systems through automated creation of prerequisites for testing, which includes initialization of test tables, creation of reference relationships and preparation of control data sets for further verification of the correctness of operations with backend data.

Table 1 below systematizes the main SQL testing methods according to the specifics of Backend QA and Database Testing. The classification takes into account SQL's capabilities for testing various aspects of backend data, including orphan record detection, business logic validation, and table relationship testing, as well as the specifics of integrating SQL queries into pre / post autotests.

Table 1 demonstrates the specialization of SQL testing methods for Backend QA, where each approach has a specific purpose in validation. backend data through compliance checking, orphan record detection, and record creation control. The gradation of complexity from simple data preparation operations to complex consistency verification reflects a phased approach to implementing SQL as a powerful Database tool Testing.

Postconditions via SQL queries are a critical component of Backend QA, as they allow you to verify the correctness of changes to backend data after performing functional operations via the user interface or API. Oriol X., Teniente E. [4] develop an incremental method for checking SQL statements to detect violations of business rules immediately after changes are made, which can be integrated into automated tests for effective validation of new record creation and modification of existing backend data.

Verifying relationships between tables is a fundamental aspect of Database Testing, which ensures the correctness of referential relationships and prevents the creation of orphan records due to incorrect delete or modify foreign key operations. Jyothi, Murthy TS [5] demonstrate that domain -specific SQL queries can be optimized for specific Backend QA scenarios, including complex integrity checks to detect data that should not exist after business operations are performed.

Figure 1 below illustrates the architectural diagram of integrating SQL testing into the Backend QA and Database process. Testing. The scheme reflects the systematic use of SQL as a powerful tool for testing backend data at all stages of the test cycle, from data preparation through preconditions to comprehensive validation through postconditions.

The architectural diagram in Fig. 1 emphasizes the multi-level nature of SQL validation in Backend QA, where each stage has a specific purpose in ensuring the integrity of backend data. The sequence of stages provides a comprehensive check from the preparation of the test environment through preconditions to the final validation through postconditions, which allows you to detect whether the records were created, whether the data meets the requirements, and whether there are orphaned records in the system.

Validation of business logic through SQL queries in Database Testing allows you to check complex conditions and rules that may not be directly reflected in the user interface, including automatically calculated aggregates, balances, and derived indicators of the backend

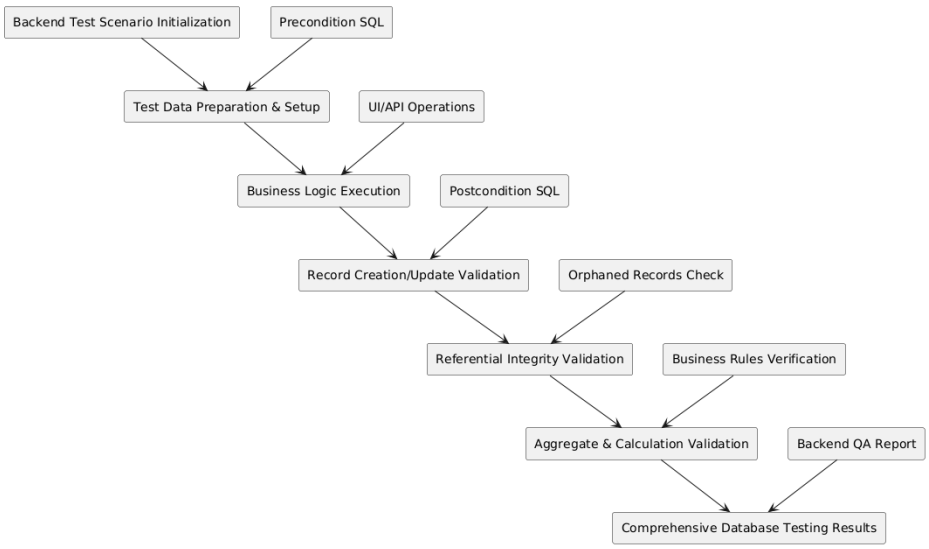

Fig. 1. SQL Integration Architecture for Backend QA and Database Testing
Source: author's development in PlantUML online editor.

Table 1. Classification of SQL testing methods for Database Testing and Backend QA

SQL Testing Method	Backend Data Validation	Pre/Post Integration	Implementation Complexity
Precondition SQL Setup	Data preparation and initialization	Pre-test conditions	Low
Record Creation Validation	Verify insert/update operations	Post-test verification	Medium
Orphaned Records Detection	Identify records without parent links	Post-cleanup validation	Medium
Business Logic Validation	Complex rules and aggregates verification	Pre/Post conditions	High
Referential Integrity Check	Foreign key constraints validation	Post-operative check	Medium
Data Consistency Verification	Cross-table consistency validation	Pre/Post comprehensive check	High

system. Nguyen D.-C., Ha M.-H., Do M.-T., Chen OT-C. [6] emphasize the need to ensure the security of SQL queries in Backend QA, especially when working with automatically generated scripts to verify the correctness of computational operations and detect incorrect database queries.

Gong Y., Lei C., Qin X., Vaidya K., Narayanaswamy B., Kraska T. [7] develop a framework for automatic detection and correction of errors in SQL, which can be adapted to create self-correcting Database Systems Testing, capable of detecting discrepancies between expected and actual results of operations with backend data and automatically adjusting test queries to increase validation accuracy.

Orphan record detection is a specific aspect of Backend QA that allows identifying data that has lost its connection to its parent records due to incorrect delete or update operations of foreign keys in the backend system. Kashyap A., Jana A. [8] emphasize the importance of a systematic approach to detecting such anomalies through regular execution of specialized SQL queries that can be integrated into automated tests to continuously monitor the integrity of backend data.

Consistency check backend data includes validation of information consistency between different tables and modules of the system, which is critically important for maintaining the integrity of business processes in complex multi-module architectures. Febrian DW, Huwae RB, Mardiansyah AZ [9] demonstrate the practical significance of such verification through Database Testing, where inconsistency backend data can lead to serious functional violations and create conditions for the emergence of incorrectly calculated aggregates.

Integrating SQL into automated tests as a powerful Backend QA tool requires the development of specialized components for connecting to the database, executing queries on pre / post conditions, and interpreting results in the context of validating non-visual aspects of the system. Pavlenko I., Boiko O., Mykolaiets D., Moskalenko O., Shrol T. [10] consider the technological aspects of integrating various tools, which can be extrapolated to the implementation of SQL testing into corporate systems to ensure comprehensive validation of operations with backend data.

Optimizing SQL query performance in Database Testing is critical to ensure the speed of execution of automated test suites, especially when testing large amounts of backend data or complex multi-level queries for business rule validation. Pasichnyi R., Serhieiev V., Shevchenko S., Petrukha N., Hryvnak B. [11] emphasize the importance of efficient use of technological resources, which includes optimizing Backend QA test processes to ensure their scalability and practical applicability in validation. backend data via SQL as a powerful validation tool.

To demonstrate the practical application of SQL as a powerful Backend QA and Database Tool Testing is shown in the graph in Fig. 2, which illustrates the results of the comprehensive validation backend -data in Google Colab. The graph displays test execution, including verification of record creation, orphan record detection, and data compliance, using SQLite as an in-memory database for isolated testing.

The graph in Fig. 2 displays the results of SQL testing as an effective tool for Backend QA and Database Testing conducted at Google Colab using SQLite in-memory database with foreign key constraints enabled (PRAGMA foreign_keys = ON). Testing covers four key stages: record creation verification (2 active users, 3 correct calculations, 2 valid completed orders), orphan

Fig. 2. Backend QA & Database Testing Results

Source: author's development at Google Collaborate.

record detection (0 orphans, 1 active user with no orders), business logic validation (0 calculation errors, 1 active user with no completed order, 0 high-value orders), and referential integrity verification (0 violations, 100% valid references). The automated approach includes preconditions (schema initialization and data insertion with IntegrityError handling for orphaned records) and postconditions (result validation), with visualization via Matplotlib with Seaborn fallback style. All tests passed successfully, as confirmed by the "Passed" marks on the graph.

CONCLUSIONS

Research into non-visual data testing methods using SQL queries in backend systems confirms that SQL is a powerful tool for backend data validation, capable of providing comprehensive validation by checking record creation, data compliance, and orphan record detection. Using SQL queries as the primary tool for Backend QA and Database Testing allows you to detect critical errors and inconsistencies that go unnoticed during traditional testing through the user interface.

Systematic use of SQL in automated tests through preconditions for data preparation and postconditions for checking the state after actions provides controlled validation of backend functionality and reliable verification of the results of operations via UI or API. Methods for detecting orphan records, checking relationships between tables, and validating business logic through SQL queries create a multi-level system of backend data quality control, allowing you to detect data that should not exist after business operations are performed.

Integrating SQL into automated testing frameworks as a powerful Backend QA tool requires the development of specialized architectural solutions that ensure efficient database connectivity, safe execution of queries on pre / post conditions, and interpretation of results in the Database context. Testing. The ability to use SQL both manually and in automated mode makes it a versatile tool for validating non-visual aspects of backend systems.

Prospects for further research include the development of automated tools for generating SQL tests for Backend QA based on the analysis of the database structure and business logic of the system, as well as the creation of adaptive Database methods. Testing for specific industry applications, taking into account their unique requirements for backend data integrity. The development of SQL testing methods will contribute to increasing the reliability of modern information systems and ensuring their compliance with business requirements through comprehensive validation of incorrectly calculated aggregates and residues after previous actions in backend systems.

REFERENCES

[1] Yun, J., & Lee, S. (2025). SEED: Enhancing text-to-SQL performance and practical usability through automatic evidence generation (Version 1). arXiv. DOI: https://doi.org/10.48550/ARXIV.2506.07423. Retrieved from https://arxiv.org/abs/2506.07423v1

[2] Ma, P., Zhuang, X., Xu, C., Jiang, X., Chen, R., & Guo, J. (2025). SQL-R1: Training natural language to SQL reasoning model by reinforcement learning (Version 3). arXiv. DOI: https://doi.org/10.48550/ARXIV.2504.08600. Retrieved from https://arxiv.org/abs/2504.08600v1

[3] Xu, K., Luo, Y., Yuan, Y., & Yao, A. C.-C. (2025). Towards automated formal verification of backend systems with LLMs (Version 1). arXiv. DOI: https://doi.org/10.48550/ARXIV.2506.10998. Retrieved from https://arxiv.org/abs/2506.10998v1

[4] Oriol, X., & Teniente, E. (2025). Incremental checking of SQL assertions in an RDBMS. Information Systems, no. 133, pp. 102550. DOI: https://doi.org/10.1016/j.is.2025.102550. Retrieved from https://www.sciencedirect.com/science/article/pii/S0306437925000353

[5] Jyothi, & Murthy, T. S. (2025). Domain specific question to SQL conversion with embedded data balancing technique (Version 1). arXiv. DOI: https://doi.org/10.48550/ARXIV.2504.08753. Retrieved from https://arxiv.org/abs/2504.08753v1

[6] Nguyen, D.-C., Ha, M.-H., Do, M.-T., & Chen, O. T.-C. (2025). Towards lightweight model using non-local-based graph convolution neural network for SQL injection detection. Egyptian Informatics Journal, no. 30, pp. 100684. DOI: https://doi.org/10.1016/j.eij.2025.100684. Retrieved from https://www.sciencedirect.com/science/article/pii/S1110866525000775

[7] Gong, Y., Lei, C., Qin, X., Vaidya, K., Narayanaswamy, B., & Kraska, T. (2025). SQLens: An end-to-end framework for error detection and correction in text-to-SQL (Version 1). arXiv. DOI: https://doi.org/10.48550/ARXIV.2506.04494. Retrieved from https://arxiv.org/abs/2506.04494v1

[8] Kashyap, A., & Jana, A. (2025). A survey: On detection and prevention techniques of SQL injection attacks. International Journal of Information and Computer Security, no. 26(4), pp. 332–371. DOI: https://doi.org/10.1504/ijics.2025.146528. Retrieved from https://www.inderscienceonline.com/doi/10.1504/IJICS.2025.146528

[9] Febrian, D. W., Huwae, R. B., & Mardiansyah, A. Z. (2025). Analisis kerentanan SQL injection, cross site scripting, dan insecure direct object reference pada website perguruan tinggi di Nusa Tenggara Barat menggunakan metode pengujian penetrasi. Jurnal Bumigora Information Technology (BITe), no. 7(1), pp. 25–38. DOI: https://doi.org/10.30812/bite.v7i1.5032. Retrieved from https://journal.universitasbumigora.ac.id/bite/article/view/5032

[10] Pavlenko, I., Boiko, O., Mykolaiets, D., Moskalenko, O., & Shrol, T. (2024). Advancements in STEM education and the evolution of game technologies in Ukrainian educational settings. Multidisciplinary Reviews, no. 7, pp. 2024spe007. DOI: https://doi.org/10.31893/multirev.2024spe007. Retrieved from https://www.multidisciplinaryreviews.com/10.31893/multirev.2024spe007

[11] Pasichnyi, R., Serhieiev, V., Shevchenko, S., Petrukha, N., & Hryvnak, B. (2024). Digital transformation of higher education as a driver of Ukraine's integration into the European educational space. Cadernos De Educação Tecnologia E Sociedade, no. 17(se4), pp. 232–245. DOI: https://doi.org/10.14571/brajets.v17.nse4.232-245. Retrieved from https://brajets.com/brajets/article/view/1905

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] Yun, J., & Lee, S. (2025). SEED: Enhancing text-to-SQL performance and practical usability through automatic evidence generation (Version 1). arXiv. URL: https://doi.org/10.48550/arXiv.2506.07423

[2] Ma, P., Zhuang, X., Xu, C., Jiang, X., Chen, R., & Guo, J. (2025). SQL-R1: Training natural language to SQL reasoning model by reinforcement learning (Version 3). arXiv. URL: https://doi.org/10.48550/arXiv.2504.08600

[3] Xu, K., Luo, Y., Yuan, Y., & Yao, A. C.-C. (2025). Towards automated formal verification of backend systems with LLMs (Version 1). arXiv. URL: https://doi.org/10.48550/arXiv.2506.10998

[4] Oriol, X., & Teniente, E. (2025). Incremental checking of SQL assertions in an RDBMS. *Information Systems*, № 133, pp. 102550. URL: https://doi.org/10.1016/j.is.2025.102550

[5] Jyothi, & Murthy, T. S. (2025). Domain specific question to SQL conversion with embedded data balancing technique (Version 1). arXiv. URL: https://doi.org/10.48550/arXiv.2504.08753

[6] Nguyen, D.-C., Ha, M.-H., Do, M.-T., & Chen, O. T.-C. (2025). Towards lightweight model using non-local-based graph convolution neural network for SQL injection detection. Egyptian Informatics Journal, 30, 100684. URL: https://doi.org/10.1016/j.eij.2025.100684

[7] Gong, Y., Lei, C., Qin, X., Vaidya, K., Narayanaswamy, B., & Kraska, T. (2025). SQLens: An end-to-end framework for error detection and correction in text-to-SQL (Version 1). arXiv. URL: https://doi.org/10.48550/arXiv.2506.04494

[8] Kashyap, A., & Jana, A. (2025). A survey on detection and prevention techniques of SQL injection attacks. *International Journal of Information and Computer Security*, Vol. 26, № 4, pp. 332–371. URL: https://doi.org/10.1504/IJICS.2025.146528

[9] Febrian, D. W., Huwae, R. B., & Mardiansyah, A. Z. (2025). Analisis kerentanan SQL injection, cross site scripting, dan insecure direct object reference pada website perguruan tinggi di Nusa Tenggara Barat menggunakan metode pengujian penetrasi. *Journal Bumigora Information Technology* (BITe), Vol. 7, № 1, pp. 25–38. URL: https://doi.org/10.30812/bite.v7i1.5032

[10] Pavlenko, I., Boiko, O., Mykolaiets, D., Moskalenko, O., & Shrol, T. (2024). Advancements in STEM education and the evolution of game technologies in Ukrainian educational settings. *Multidisciplinary Reviews,* № 7, pp. 2024spe007. URL: https://doi.org/10.31893/multirev.2024spe007

[11] Pasichnyi, R., Serhieiev, V., Shevchenko, S., Petrukha, N., & Hryvnak, B. (2024). Digital transformation of higher education as a driver of Ukraine's integration into the European educational space. *Cadernos de Educação Tecnologia e Sociedade*, № 17(se4), pp. 232–245. URL: https://doi.org/10.14571/brajets.v17.nse4.232-245

Дата першого надходження статті до видання: 22.01.2026
Дата прийняття статті до друку після рецензування: 17.02.2026
Дата публікації (оприлюднення) статті: 22.05.2026