Relational Thinking Meets NoSQL: Challenges in Learning MongoDB and Cassandra

Nea Peltola
nea.peltola@tuni.fi
Tampere University
Tampere, Finland

Hilkka Grahn
hilkka.grahn@jyu.fi
University of Jyväskylä
Jyväskylä, Finland

Mikko Nurminen
mikko.nurminen@tuni.fi
Tampere University
Tampere, Finland

Katriina Vartiainen
katriina.vartiainen@tuni.fi
Tampere University
Tampere, Finland

Bingxiang Chen
bingxiang.chen@tuni.fi
Tampere University
Tampere, Finland

Toni Taipalus
toni.taipalus@tuni.fi
Tampere University
Tampere, Finland

Abstract

NoSQL data models such as document databases and wide-column databases have gained popularity in industry and education in the last two decades. The database design principles and query languages of different NoSQL systems differ from those of the relational model and SQL, which can cause challenges in transitioning from relational systems to NoSQL systems. In this mixed-methods study, we explore the challenges novices experience with database design and querying with two NoSQL systems: MongoDB and Cassandra. Our results show that NoSQL database design is perceived more difficult than relational due to access-pattern-first design patterns, old habits, and the limits imposed by database distribution. Querying, however, is not perceived as more difficult or easier due to similarities of query languages to previous experiences with SQL and programming. These results are applicable in the classroom in determining the teaching order of novel data models, and in focusing teaching on the challenging aspects in document and wide-column databases.

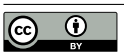
CCS Concepts

• **Information systems** → **Query languages for non-relational engines**; Semi-structured data; • **Theory of computation** → **Data modeling**; • **Applied computing** → *Education.*

Keywords

querying, database design, NoSQL, MongoDB, Cassandra, MQL, CQL, data modeling, wide-column database, document database

ACM Reference Format:
Nea Peltola, Hilkka Grahn, Mikko Nurminen, Katriina Vartiainen, Bingxiang Chen, and Toni Taipalus. 2026. Relational Thinking Meets NoSQL: Challenges in Learning MongoDB and Cassandra. In *Proceedings of the 31st ACM Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2026), July 10–15, 2026, Madrid, Spain.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3803400.3809335



ITiCSE 2026, Madrid, Spain

ACM ISBN 979-8-4007-2634-7/2026/07
https://doi.org/10.1145/3803400.3809335

1 Introduction

NoSQL is an umbrella term for several novel, non-relational data models. NoSQL systems have established themselves in domains where flexible database schemas, low latency, and high data availability are valued [7], yet where relational database management systems (RDBMSs) fail to meet these criteria [5]. NoSQL courses has received less attention than relational databases in academic curricula [13, 15] as well as in computing education research [cf. 20, 21], most likely due to their relative novelty.

In this study, we examine the challenges in logical database design and querying with two NoSQL systems: MongoDB and Cassandra. Quantitatively, for both systems, we study the perceived difficulty in database design and querying when compared to SQL and relational databases. Qualitatively, and again for both systems, we analyze what are the most and least challenging aspects in database design and querying. The results show that database design with both MongoDB and Cassandra is perceived more difficult than relational database design, but querying with both MongoDB and Cassandra is perceived to be of similar difficulty as SQL. Our qualitative insights provide more information on the *why*.

The rest of this study is structured as follows. In the next section, we provide theoretical background on wide-column and document database design and querying, as well as on educational research on NoSQL systems. In Section 3, we detail our survey instrument, the participants, and our data analysis methods. Section 4 contains the results, and Section 5 practical implications and threats to validity. Section 6 concludes the study.

2 Theoretical background

2.1 Relational and NoSQL fundamentals

Relational databases rely on relatively fixed schemas, normalization theory, and join operations to maintain consistency and support flexible queries. Database design follows a normalization-first process, where minimizing redundancy and preserving referential integrity are central goals. These features make relational systems predictable and well-suited for a wide range of tasks [17]. On the other hand, NoSQL data models emerged to address scalability and flexibility challenges faced by relational systems. They often relax schema constraints, support horizontal scalability, and distribute data across multiple nodes. Instead of normalization, NoSQL

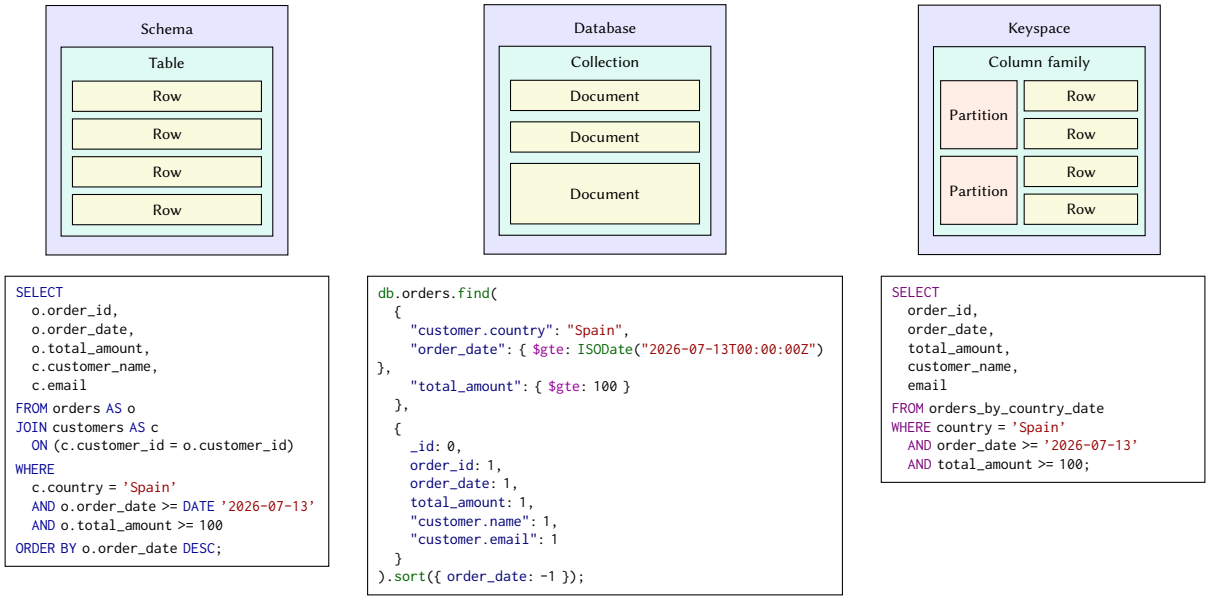


Figure 1: General data structures in a relational database (left), and their rough equivalents in a document database (center), and in a wide-column database (right); below the data structures, query equivalents in SQL (left), MQL (center), and CQL (right); the MQL and CQL queries presuppose that the database has been designed to accommodate this particular data access pattern

systems commonly adopt an access-pattern–driven design philosophy. Many NoSQL systems do not directly support joins, and their query capabilities depend on efficiently designing database schemas [9, 17].

NoSQL databases are commonly categorized into four data models: key-value, document, wide-column, and graph [5, 21], out of which document and wide-column models are relevant to this study. DBMSs using the document model (e.g., MongoDB and CouchDB) represent data using flexible JSON documents. Wide-column DBMSs such as Cassandra store data logically in table-like structures, but physically in a column-oriented fashion [19], i.e., by storing the values of a column physically together. This differs from relational, row-oriented databases, which store the values of a row physically together.

2.2 Database design in MongoDB and Cassandra

MongoDB is a document-oriented NoSQL database that stores data as JSON-like documents with flexible and potentially heterogeneous schemas. On a high level of abstraction, documents correspond to rows in a relational database, and documents of a same type or intent are stored in a collection [5], which roughly corresponds to a table [18] (cf. Fig. 1). Documents can contain nested objects and arrays, enabling developers to model complex entities without the need for pre-defined data structures [22]. A central design decision involves embedding related data within a document or referencing it across multiple documents. Embedding supports efficient read operations, whereas referencing encourages modular and normalized structures [10]. Unlike relational databases, MongoDB encourages denormalized structures designed around expected read patterns.

Cassandra is a wide-column DMBS optimized for availability and horizontal scalability. From a human perspective, the data structures consist of rows and column families, the latter corresponding to tables. Data are organized using a partition key and optional clustering keys. The partition key determines data placement across physical nodes, while clustering keys define row ordering inside a partition. Cassandra does not support join operations, so schema design must begin with clearly defined access patterns. Furthermore, the boundaries between logical and physical database design are blurred, as Cassandra strives to avoid queries that do not efficiently utilize the chosen partition (and possible clustering) keys [22].

2.3 Querying in MongoDB and Cassandra

MongoDB query language (MQL) operates with JavaScript methods. Typically, schemas are not defined, but created through insert operations, i.e., an insert with whichever fields defines the structure of a particular document, yet the structure of that document is neither necessarily based on pre-existing documents, nor does it define the structure for subsequently inserted documents. Querying is based on defining a JSON document in the query, which is then compared to the documents in the database, and matches are returned. MongoDB supports complex queries through its aggregation pipeline, a multi-stage framework for filtering, transforming, and combining documents. The pipeline performs functions analogous to joins in relational databases but requires procedural reasoning and familiarity with pipeline stages [10].

Cassandra query language (CQL) is similar in syntax to SQL, yet missing joins and including operators for multi-valued columns such as lists and dictionaries. Queries must strictly follow the chosen partition keys, and filtering outside a partition is typically not permitted. This restriction is arguably the biggest departure from

SQL, where arbitrary joins and selections are allowed despite the physical structure of the database.

2.4 NoSQL education

As stated in the Introduction, NoSQL education has received less scientific attention than relational databases, although the scientific trend is on the rise [21]. We are not aware of empirical studies on the education of document or wide-column database design. Some studies provide practice-oriented design guidelines for document databases [4], or more general frameworks suited for several NoSQL data models [6].

On the other hand, querying with NoSQL systems has been studied in computing education research. It has been shown that students struggle with transferring abstract data operation concepts when learning a new database language, even if they have already practiced those concepts in a different language [12]. There have also been attempts to mitigate these challenges in the classroom with tools such as `SQL2X`, which illustrates how SQL queries translate to, e.g., MongoDB queries [24] and `TriQL`, which helps in learning multiple query languages [1].

Regarding querying with MQL, some studies have identified the most challenging aspects from novice perspectives, such as aggregation [2, 8] and document linking [3]. Additionally, MQL does not seem particularly intuitive for students [14]. Regarding CQL, one study found that as CQL cannot express joins, several tasks forced students into conceptual dead-ends [14], meaning that as database design is closely connected to querying, formulating a query correctly sometimes requires schema redesign. Recent studies have also explored query formulation errors in learning NoSQL languages such as Cypher [16] and MQL [25].

3 Methodology

3.1 Survey instrument

We assessed the perceived difficulty of designing and querying MongoDB and Cassandra databases on an 11-point scale (0..10), where 5 represented the perceived difficulty of SQL databases. That is, responses greater than 5 indicated how much the participant perceived the task to be *more difficult* than an equivalent task in an SQL database, and responses less than 5 indicated how much *easier* the task was perceived. For example, a response of 2 regarding the question of the difficulty of designing a document database in MongoDB indicated that is was much easier for the participant to design a document database than a relational database. Additionally, we asked *what* were the most challenging and the least challenging aspects of designing and querying with the chosen NoSQL systems.

The survey instrument included questions regarding participant background: age, gender, major subject, highest degree, programming experience, and a list of maximum of three programming languages the participant felt most prominent in.

The survey was shared on an online course platform, and the data was collected by the course instructor. Prior to answering, the potential participants were shown a full privacy statement regarding how their data would be handled. Answering the survey was voluntary, yet answering yielded course points towards a better grade. Consenting to participating in the research was also voluntary, and carried no positive or negative consequences. That is, a potential participant could answer the survey yet decline to participate. In such cases, course points were given, and the survey answers were not stored or used in this study. Of the 94 students, 68 chose to participate – a response rate of 72%. As participation was based on informed consent, the potential participants were not minors, and participating carried no risks or threats to safety, no ethical approval was needed for data collection, as per national board on research integrity guidelines.

3.2 Participants

The participants (N = 68) were recruited from the two universities of the authors, representing a total of four cohorts. Cohorts #1 (12 participants), #2 (13 participants) and #3 (8 participants) were recruited from university A, and cohort #4 (35 participants) from university B. Approximately 22% of the participants were female. All the cohorts studied remotely.

The course was intended for third year information technology students, and the required prerequisite courses included relational databases and programming. Approximately 42% of the participants majored in software engineering, approximately 27% in management information systems, and approximately 8% in computer science. The rest (23%) majored in other subjects such as cyber security or knowledge management. Prior to answering our survey, the participants were taught NoSQL database design, querying, and database distribution with Cassandra and MongoDB (cohorts #1, #2 and #3). Cohort #4 was taught Neo4j and Redis in addition to MongoDB and Cassandra. For all cohorts, teaching consisted of weekly lectures (approximately 2 hours per week) and practical, self-guided work (approximately 5 hours per week).

3.3 Analyses

We visually inspected the assumptions of normality for all four dependent variables (difficulties in designing and querying with MongoDB and Cassandra), and found the assumptions to be approximately satisfied. Therefore, we used parametric paired samples t-test in the analyses.

In addition to the quantitative answers, the participants were given a chance to explain *what aspects* of querying or database design were easy and challenging in free-form text fields. There were a total of eight of such text fields (easy/challenging × MongoDB/Cassandra × design/querying), and we received a total of 412 responses for qualitative data analysis. All participants did not answer to all qualitative questions, and some participants voiced more than one challenging or easy aspect. In the analyses, we utilized both conventional content analysis and directed content analysis [11]. First, the sixth author coded 10% of all qualitative data using conventional content analysis, question-per-question. Second, these initial codes were used for directed content analysis by the first, third, fourth, and sixth author who coded all the data, two questions each. Finally, the first and sixth author convened to appropriately name and combine the codes into categories reported in the results section.

Additionally, we tested whether the collected background variables had an effect on the results. Uniformly, there were no statistically significant differences in the dependent variables between genders, universities, or major subjects. Furthermore, age, highest

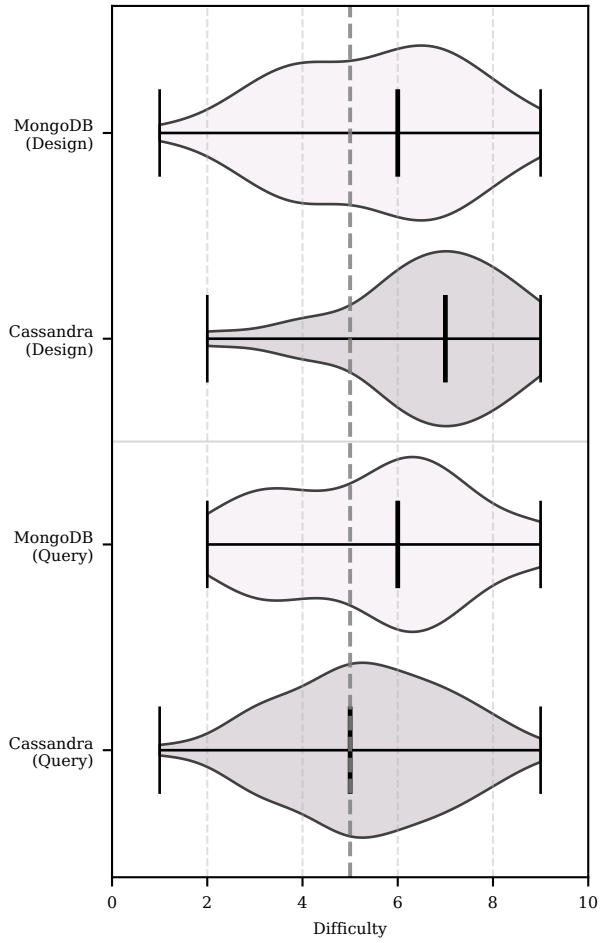


Figure 2: Violin plots of the difficulty of database design and querying in MongoDB and Cassandra in relation to SQL databases; the dashed vertical line represents the perceived difficulty of SQL databases, i.e., values higher than 5 indicate that a task with a NoSQL system is more difficult than the corresponding task with an SQL system; lines at the ends of the violins represent min and max values, and lines in the middle represent medians

degree, or programming experience had no statistically significant effect on any of the dependent variables. We chose an alpha level of .05 for all the statistical tests in this study.

4 Results

4.1 Database design with MongoDB

When compared to relational database design (i.e., the baseline of 5), database design in MongoDB was considered more difficult. One-sample t-tests revealed a statistically significant difference, $t(67) = 2.31$, $p = .024$, with a small effect ($d = 0.28$). The first violin plot in Fig. 2 shows slight bimodality, where some participants deemed MongoDB database design somewhat more difficult, and others somewhat less difficult.

Table 1: The challenging and easy aspects in designing a MongoDB database, and the number of mentions

#	Challenging
25	I am accustomed to relational database design.
14	Designing access-pattern-first and predicting future needs.
12	Finding the optimal structure for shard keys.
11	Working around the lack of joins.
9	An overall lack of design rules.
6	Working with nested documents.
4	Learning a new syntax for implementation.
3	Optimization and secondary indices.
	Easy
28	Flexible nature of the database schema.
6	No need to avoid redundancy.
6	No need to think about joins.
4	Familiar data types.
3	JSON syntax.
3	Less to worry about than in relational database design.
3	All data about one object is in one document.

Table 2: The challenging and easy aspects in designing a Cassandra database, and the number of mentions

#	Challenging
23	Choosing the partition and clustering keys.
22	Understanding new database structures.
8	Designing based on access patters.
3	I am accustomed to other data models.
	Easy
15	Familiarity with the basic data structures.
11	Imagining the queries for the structure.
8	No need to avoid redundancy.
6	Flexible nature of the schema.

The results of the qualitative analysis, i.e., which aspects of document database design with MongoDB were the most and least challenging are listed in Table 1.

4.2 Database design with Cassandra

When compared to relational database design (i.e., the baseline of 5), database design in Cassandra was considered more difficult. One-sample t-tests revealed a statistically significant difference, $t(67) = 8.43$, $p < .001$, with a large effect ($d = 1.02$). The second violin plot in Fig. 2 shows that rather uniformly, most of the participants deemed Cassandra database design somewhat more difficult.

The results of the qualitative analysis, i.e., which aspects of wide-column database design with Cassandra were the most and least challenging are listed in Table 2.

4.3 Querying with MongoDB

When compared to querying with SQL (i.e., the baseline of 5), querying in MongoDB was not considered more difficult. One-sample t-tests approached statistical significance, $t(67) = 2.00$, $p = .050$, with

Table 3: The challenging and easy aspects in querying with MongoDB, and the number of mentions

#	Challenging
36	The syntax of queries.
13	Aggregation pipeline logic.
7	I am accustomed to other query languages.
5	The lack of joins.
5	Querying requires knowledge of JavaScript.
	Easy
13	I am familiar with JavaScript and JSON.
12	Basic operations are easy to apply.
3	The flexible schema makes inserts easy.
3	The structure of the database is intuitive.
2	Overall query logic.
2	Aggregation pipeline logic.

Table 4: The challenging and easy aspects in querying with Cassandra (CQL), and the number of mentions

#	Challenging
28	Understanding how partition keys affect queries.
20	Understanding the limitations of CQL.
4	The syntax of CQL.
3	The fact that querying is tied to database design.
3	Understanding the database structure.
	Easy
28	The syntax of CQL.
8	The logic of CQL.
7	Applying basic operations.

a small effect (d = 0.24). The third violin plot in Fig. 2 shows slight bimodality, similar to MongoDB database design difficulty.

The results of the qualitative analysis, i.e., which aspects of querying with MongoDB were the most and least challenging are listed in Table 3.

4.4 Querying with Cassandra

When compared to querying with SQL (i.e., the baseline of 5), querying in Cassandra (CQL) was not considered more difficult. No statistically significant difference was found (p = .102). The last violin plot in Fig. 2 shows a bell curve indicating that rather uniformly, the participants rated querying to be of similar difficulty in CQL as in SQL. The results of the qualitative analysis, i.e., which aspects of querying with Cassandra were the most and least challenging are listed in Table 4.

5 Discussion

5.1 A summary of results

Document database design is difficult due to previous mental models pertaining to relational database design, and because designing access-pattern first and predicting future access-patterns is challenging. This seems justified, as relational database design can be seen as *one size fits all*, regardless of the access-pattern. That is, we can use normalization theory to design a relational database, and rely on the fact that whatever our queries will eventually be, they are at least adequately compatible with the database structure. The flexible nature of the schema was seen as the main point in making document database design easy.

Wide-column database design is difficult, rather uniformly, and even more so than document database design. The primary challenges are the selection of partition and clustering keys that directly affect which queries can be executed, as well as new database structures such as multi-valued data types. Compared to relational database design, the lack of need to design foreign keys was not seen a contributing factor. On the other hand, familiarity of basic database structures such as tables (i.e., column families) and columns was seen as the main facilitator towards making database design easy.

Querying with MongoDB is not difficult or easy. Compared to SQL, some participants perceived querying easier and some more challenging (Fig. 2). The most challenging factor was the syntax of queries, which is arguably rather syntax-heavy with three types of brackets, and other special characters such as quotation marks, colons, commas, periods and dollar-signs. On the other hand, some participants voiced that their familiarity with JavaScript made querying easier.

Querying with CQL is not difficult or easy. Compared to SQL, most of the participants deemed querying with CQL of similar difficulty. According to the qualitative analysis, this was due to the syntax of CQL, which was familiar to the participants in the form of SQL. Despite the fact that CQL does not support table joins, which would arguably make querying easier due to simplicity of CQL, CQL was not deemed easier than SQL. According to the qualitative analysis, query formulation was not difficult because of syntactical or logical intricacies of CQL, but because it was challenging to understand which queries Cassandra allows to be executed, as the execution is based on the chosen partition and clustering keys.

5.2 Comparison with prior findings

Our findings align with and extend prior research on teaching MongoDB and Cassandra. In MongoDB education, earlier work shows that students frequently struggle with operations that conceptually resemble relational joins. For example, aggregation and join-like operations have been reported as the most difficult aspects based on the number of semantic errors students commit [2]. Additional evidence indicates that the most common MongoDB querying errors arise when dealing with arrays, embedded documents, and linked documents queried via cursors [3]. Importantly, learning querying with MongoDB tends to yield more semantic errors than SQL [2]. This pattern is consistent with studies showing that students generally perceive SQL as simpler and more relevant than MongoDB's aggregation pipeline [8]. Recent analyses also show that students find complex tasks such as theta-joins in MongoDB particularly challenging, whereas tasks related to handling missing values or issuing range queries are perceived as easier [14]. These prior findings parallel the patterns observed in our study, where MongoDB design and querying were also considered relatively challenging by many participants, when compared to similar tasks in SQL databases.

Similar trends appear in wide-column database education, although scientific evidence is more scarce. Earlier research suggests that the absence of joins in CQL contributes substantially to students' perceived task difficulty [14]. When confronted with problems that intuitively call for join operations, students often experience uncertainty, especially when tasks expect an explanatory text answer rather than a direct query [14]. More broadly, NoSQL querying has been noted to often require additional programming skills not typically needed for SQL, adding another layer of conceptual load for learners [23]. On one hand, our results align with these observations: while students did not perceive querying with Cassandra to be more difficult than SQL overall, the challenges they identified were closely tied to Cassandra's data modeling foundations and the constraints imposed by its query language. On the other hand, while MongoDB querying was considered more challenging than SQL, the qualitative analysis showed that familiarity with programming in fact makes MongoDB querying easier.

5.3 Practical implications

Designing NoSQL databases presents a set of conceptual and practical challenges that fundamentally differ from those encountered in relational database design. From a computing education perspective, understanding these difficulties contributes to both curriculum development and pedagogical research. Unlike relational systems, NoSQL databases lack universally accepted design methodologies and require designers to reason about data organization in relation to specific access patterns and scalability constraints. These complexities arguably increase the cognitive load on learners, who must shift from well-established database normalization principles toward context-sensitive decision-making that balances multiple competing system goals. Investigating how students navigate these challenges helps identify the conceptual gaps and misconceptions that arise when they transfer knowledge from traditional data modeling paradigms to the NoSQL context.

Furthermore, exploring the difficulty of NoSQL design provides insight into broader questions of learning and problem-solving in computing education. Research in this area can reveal how students develop abstract reasoning about data systems, how they construct mental models for distributed storage, and what instructional approaches best support the acquisition of such knowledge. As industry increasingly relies on polyglot and distributed data systems, preparing students to engage effectively with these technologies requires a deeper pedagogical understanding of their complexity. Therefore, studying the challenges of NoSQL database design and querying facilitates the development of teaching strategies.

5.4 Limitations and threats to validity

We recognize several threats to validity in our study design. *Construct validity*: it is possible that participants may have interpreted survey questions differently from what we intended, reducing the accuracy with which the instrument measures the intended constructs. We tried to mitigate this by phrasing the questions clearly. It is also possible that we misinterpreted qualitative answers, which may lead to incorrect mapping between participant responses and theoretical constructs. We mitigated this by using both conventional and directed content analysis, involving multiple coders (cf. Section 3.3), and conducting coder discussions to reconcile category naming and interpretation.

Internal validity: the four cohorts may differ in motivation, prior experience, or informal learning environments, which can influence outcomes in unseen ways. Furthermore, variations in guidance or pedagogical style may affect how students learn and perform. The choice of involving two universities instead of one emphasizes this concern, yet there were no statistically significant differences between the participants from the two universities. Additionally, teaching resources that vary across cohorts may introduce systematic differences in learning outcomes. We mitigated this by harmonizing core content (MongoDB and Cassandra), ensuring all cohorts received the fundamental material despite some variations. The data were collected across four years, which might induce unforeseen changes in study program structure, student preparation, or technology familiarity. We mitigated all the above by statistically testing whether the collected background variables influenced the results (they did not), as reported in Section 3.3. Finally, small cohorts may increase sensitivity to outliers or reduce the stability of findings. We mitigated this by combining four cohorts (N = 68) and reporting effect sizes, not only significance tests.

External validity: variations in major subjects may limit the generalizability of the results to a specific type of student profile. We mitigated this by collecting detailed background information and statistically examining whether major subject influenced outcomes (it did not, cf. Section 3.3). Similar to the above, major subject differences may affect prior knowledge or familiarity with database concepts. Further, collecting data from two institutions may introduce institutional differences (teaching culture, student demographics), affecting generalizability. We mitigated this by recruiting comparable cohorts and statistically testing for university-level effects (none were found). Additionally, conducting the study within a single national context limits generalizability to international populations. Finally, we have consciously limited our scope to MongoDB and Cassandra, and not to document or wide-column databases in general. Our results may not be applicable in different document databases (such as CouchDB) or different wide-column databases (such as ScyllaDB), because the design principles and query languages may differ even within a single NoSQL data model.

6 Conclusion

NoSQL systems have established a foothold in the data systems landscape, and non-relational data models are a part of a modern data systems student's toolkit. In this study, we sought to understand what makes data modeling and querying with MongoDB and Cassandra challenging and what makes these tasks easy. These results are applicable in the classroom in focusing on challenging topics, preparing for common mistakes, and designing course prerequisites. For future work, and given the current gap in the literature, we specifically call for in-depth research on the design, implementation, and evaluation of educational approaches to *NoSQL database design*.

Acknowledgments

We thank all study participants for their valuable contributions.

References

[1] Abdussalam Alawini, Peilin Rao, Leyao Zhou, Lujia Kang, and Ping-Che Ho. 2022. Teaching data models with TriQL. In *Proceedings of the 1st International Workshop on Data Systems Education*. 16–21.

[2] Ridha Alkhabaz, Zepei Li, Sophia Yang, and Abdussalam Alawini. 2023. Student's learning challenges with relational, document, and graph query languages. In *Proceedings of the 2nd International Workshop on Data Systems Education: Bridging education practice with education research*. 30–36.

[3] Ridha Alkhabaz, Seth Poulsen, Mei Chen, and Abdussalam Alawini. 2021. Insights from student solutions to MongoDB homework problems. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*. 276–282.

[4] Michael Carey, Wail Alkowaileet, Nick DiGeronimo, Peeyush Gupta, Sachin Smotra, and Till Westmann. 2025. Towards Principled, Practical Document Database Design. *Proceedings of the VLDB Endowment* 18, 12 (2025), 4804–4816.

[5] Ali Davoudian, Liu Chen, and Mengchi Liu. 2018. A survey on NoSQL stores. *ACM Computing Surveys (CSUR)* 51, 2 (2018), 1–43.

[6] Alfonso de la Vega, Diego García-Saiz, Carlos Blanco, Marta Zorrilla, and Pablo Sánchez. 2018. Mortadelo: A model-driven framework for NoSQL database design. In *International Conference on Model and Data Engineering*. Springer, 41–57.

[7] Miguel Diogo, Bruno Cabral, and Jorge Bernardino. 2019. Consistency models of NoSQL databases. *Future Internet* 11, 2 (2019), 43.

[8] Jens Ehlers. 2024. Teaching multiple data models and query languages. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*. 234–240.

[9] Adity Gupta, Swati Tyagi, Nupur Panwar, Shelly Sachdeva, and Upaang Saxena. 2017. NoSQL databases: Critical analysis and comparison. In *2017 International conference on computing and communication technologies for smart nation (IC3TSN)*. IEEE, 293–299.

[10] Cornelia Győrödi, Robert Győrödi, George Pecherle, and Andrada Olah. 2015. A comparative study: MongoDB vs. MySQL. In *2015 13th international conference on engineering of modern electric systems (EMES)*. IEEE, 1–6.

[11] Hsiu-Fang Hsieh and Sarah E Shannon. 2005. Three approaches to qualitative content analysis. *Qualitative health research* 15, 9 (2005), 1277–1288.

[12] Zepei Li, Sophia Yang, Kathryn Cunningham, and Abdussalam Alawini. 2023. Assessing student learning across various database query languages. In *2023 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.

[13] Margaret Menzin, Sriram Mohan, David R Musicant, and Raja Sooriamurthi. 2020. NoSQL in undergrad courses is noproblem. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 962–963.

[14] Vanessa Meyer, Lena Wiese, and Ahmed Al-Ghezi. 2024. Analyzing Learning Patterns, Task Difficulty and Usability in a Digital Database Learning Tool. In *2024 21st International Conference on Information Technology Based Higher Education and Training (ITHET)*. IEEE, 1–10.

[15] Sriram Mohan. 2018. Teaching NoSQL databases to undergraduate students: A novel approach. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. 314–319.

[16] Amedeo Pachera, Stefania Dumbrava, Angela Bonifati, and Andrea Mauri. 2025. Understanding student errors in graph query formulation. *ACM Transactions on Computing Education* 25, 3 (2025), 1–35.

[17] Kosovare Sahatqija, Jaumin Ajdari, Xhemal Zenuni, Bujar Raufi, and Florije Ismaili. 2018. Comparison between relational and NOSQL databases. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 0216–0221.

[18] Abdul Sattar, Torben Lorenzen, and Keerthi Nallamaddi. 2013. Incorporating nosql into a database course. *Acm Inroads* 4, 2 (2013), 50–53.

[19] Christof Strauch, Ultra-Large Scale Sites, and Walter Kriha. 2011. NoSQL databases. *Lecture Notes, Stuttgart Media University* 20, 24 (2011), 79.

[20] Toni Taipalus and Ville Seppänen. 2020. SQL education: A systematic mapping study and future research agenda. *ACM Transactions on Computing Education (TOCE)* 20, 3 (2020), 1–33.

[21] Nirnaya Tripathi. 2025. NoSQL database education: A review of models, tools and teaching methods. *Journal of Systems and Software* (2025), 112391.

[22] Harley Vera-Olivera, Ruizhe Guo, Ruben Cruz Huacarpuma, Ana Paula Bernardi Da Silva, Ari Melo Mariano, and Maristela Holanda. 2021. Data modeling and nosql databases-a systematic mapping review. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–26.

[23] Hai Wang and Shouhong Wang. 2023. Teaching tip: Teaching NoSQL databases in a database course for business students. *Journal of Information Systems Education* 34, 1 (2023), 32–40.

[24] Wensheng Wu. 2021. SQL2X: Learning SQL, NoSQL, and MapReduce via translation. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 590–596.

[25] Sophia S Yang. 2025. *Bridging the gap: Understanding SQL learning challenges through quantitative analyses and qualitative insights*. Ph. D. Dissertation. University of Illinois Urbana-Champaign.