Journal of Applied Research and Technology

www.jart.icat.unam.mx

Journal of Applied Research and Technology 20 (2022) 306-319

Original

A test model for database architectures: an assessment for job search engine systems

Mary Carlota Bernal[a]* • Yeimer Molina[b]

[a]*Universidad Simón Bolívar, Facultad de Ingenierías, Cúcuta, Colombia*
[b]*Universidad Nacional Experimental del Táchira,*
Laboratorio de Investigación y Desarrollo en Informática, San Cristóbal, Venezuela

Received 08 21 2020; accepted 11 16 2021
Available 06 30 2022

Abstract: Information systems are increasingly complex structures due to the diversity of processes involved and the big data generated, hence data management is essential. NoSQL databases adopt new approaches to data management differing from relational structures. In this study, three databases were designed, a relational database using PostgreSQL and two NoSQL databases made in MongoDB applied to operation of a job offer system, with the aim of comparing its operation and efficiency. A method was proposed for the metric-guided evaluation of database models using functionality and efficiency criteria according to Systems and Software Standard Quality Requirements and Evaluation (SQuaRE). Testing cases were created considering the International Software Testing Qualifications Board (ISTQB) best practices. Relational data model was selected as a pattern, for this reason, to populate NoSQL databases a reference framework was applied for data migration from one environment to another, thus the tests were performed under the same hardware, software and data conditions. This study determined that the SQL schema provides greater functionality, ensuring transaction support and data integrity. On the other hand, the NoSQL schemas are more efficient in response to big data processing, although they have a certain level of data duplication, transaction support fails and some join operations are not support.

Keywords: Relational database, NoSQL, functionality, efficiency, test model, job search engine systems

*Corresponding author.
E-mail address: m.bernal@unisimonbolivar.edu.co(Mary Carlota Bernal).

1. Introduction

Relational databases have played a significant role within organizations, offering features that allow redundancy control, consistency, sharing, integrity, security, and accessibility of data (Marqués, 2011), this complemented with the regular support of transactions, atomicity and data independence (Date, 2000) give support and stability to business operations.

Despite the characteristics above and the popularity that relational systems have had, (Han et al., 2011) describe that these are usually insufficient for the constant big data processing and analysis requirements generated in today's environments. In attention to the previous situation, the databases called NoSQL (Not only SQL), work in order to face some of the situations generated by relational systems taking advantage of features such as simplified design, horizontal scalability, and greater control over availability of the data (Zafar et al., 2017) .

The most recent comparative database studies are focused on big data systems, in which the characteristics of the databases are analyzed with the use of unstructured data that is generated from the various interactions in cloud services (Kumar & Jayagopal, 2018). However, database management systems (DBMS) are also essential elements for daily practice in small and medium-sized companies, in which the current dynamics have increased the digital transformation of their processes, requiring new ways of organizing structured data for its use (Sokolova et al., 2020).

With the massive growth in data volume, many companies have migrated towards NoSQL database adoption to store high amounts of data, and both analyzing and managing data. However, many organizations, based on the nature of their operations, use relational databases or hybrid data storage model to manage their processes. Therefore, this situation leads to new movement from relational and object-relational databases to NoSQL databases. Even so, the data models are entirely different between these two types of databases, on the fact that NoSQL are distributed databases without join operations and with schemas dynamic (Schreiner et al., 2020). Consequently, model transformation is fundamental to ease migration and recognize the advantages and disadvantages of its implementation.

In this context, this study aims to compare a relational database with a non-relational database: a document database (MongoDB), a system that requires easy data recovery and data consistency. For this paper, a data structure was developed for a job search system. We selected MongoDB database management system because it is the most popular open-source document-oriented database, and it is a suitable solution for both, distributing data and managing the balance between instances (Fraczek & Plechawska-Wojcik, 2017). On the other hand, among the relational databases, we chose PostgreSQL, another open-source but also commercially available database.

The main contributions of this paper are outlined as follows:

• Definition of a relational and non-relational data model for the operation of a job offer system, based on requirements oriented to base operations in this kind of system and showing two-design strategies to know the scope and limitations of each solution.

• The development of testing strategies for both relational and non-relational databases, constructing a reference framework for a migration from one environment to another.

• A metric-guided evaluation for tracking results and provide recommendations for the different database schemas implemented.

The rest of this paper is organized as follows. Section 2 presents an overview of relational and NoSQL data models. Section 3 includes the evaluation method developed in this work. Section 4 describes both the comparisons and the metrics obtained and the paper is concluded in Section 5.

2. Job search engine systems

A job search engine is an online platform that helps people to find opportunities for employment. With the help of job search systems, persons can create and share their resumes and search for vacancies, while employers are able to post job offerings and look for suitable candidates. To achieve this purpose, these systems must be characterized by:

• Flexibility. The platform should allow different types of job search and posting. That is, allow employers and recruiters the ability to do things like post multiple ads and offer users search tools that include things like the ability to filter results.

• Accuracy. The platform should allow to develop searches with precise results according to the registered profiles.

• Relevancy. System information retrieval should allow search results to be relevant according to characteristics and profiles to attract and retain users in their different roles

A platform that complies with these specifications requires an adequate database design that allows the registration and retrieval of data to satisfy each functionality. Relational and non-relational approaches can be proposed for this goal. This study explores and evaluates design alternatives considering the characteristics of flexibility, accuracy and relevance typical of a search system and evaluates the behavior of these databases in terms of functional architecture when they are the object of basic operations for the normal performance of this system, considering the structure of the data, the query pattern and the scale.

3. An overview in databases evaluation

3.1. Relational database systems. architecture and levels of abstraction

A relational database is an organized collection of related data (Date, 2000), whose relationships form a logical structure, which contains not only the data but how they are also related. These structures are called metadata and make the independence logical-physical data possible (Paredaens et al., 1989). The basic unit of work in a relational database environment is the transaction. A database transaction symbolizes a unit of work performed within a database management system against a database and is treated coherently and reliably, independent of other transactions. A transaction generally represents any change in a database. (Elmasri Ramez & Navathe Shamkant, 2022). Integrity in the database is achieved by the guarantee of its ACID properties (Atomicity, Consistency, Isolation, Durability), where the atomicity allows operations to develop all or none, denoting completeness. The consistency ensures that the database passes from a valid state to another valid one maintaining its integrity and isolation to maintain the independence and durability of the operations and therefore guarantee, once the operation has been developed, persist. Access to registered information in the relational database is done by the Structured Query Language -a high-level language- that allows both, the retrieval and structured data management, transforming it into information.

These primary characteristics define the functional architecture in which data is organized and retrieved in this type of environment, and that combined with the rules to define what a database administration system requires (Codd, 1971) establishes the operating engine and the levels of abstraction present in the relational database management systems. PostgreSQL is an open-source object-relational database that has earned a strong reputation for reliability, feature robustness, and performance (The PostgreSQL Global Development Group, 2013). For this reason, PostgreSQL was chosen for relational data representation in the present study.

3.2. Non-relational database systems (NoSQL). functional architecture

The NoSQL approach refers to all those databases or data warehouses that do not follow the basic principles listed above and are related to extensive databases that are regularly-accessed. These characteristics describe a wide diversity of products and technologies having some relationship with each other (simplified design, horizontal scalability, and greater control over data availability) (Imam et al., 2018), but the method of data storage and manipulation is different. NoSQL databases do not follow a concept of relationships between records, therefore, the data is presented in a denormalized form. This approach contains all the data in the same structure, allowing them to handle long read and write flows, but frequently affecting other features of the data (Bugiotti et al., 2014).

NoSQL database architectures are classified according to the way data store, including categories such as key-values, document-oriented databases, BigTable implementations, and graph-oriented databases (Ercan & Lane, 2014). For this study, the analysis of a document-oriented architecture was considered.

Document-oriented databases

This NoSQL approach type is based on the fact that any data or entity can be stored as a document (Hows et al., 2014). The documents in this type of database refer to the set of key-value pairs stored, which use JSON (JavaScript Object Notation) notation to a data record and the architectural structure is a set of documents called "collection" (Scherzinger et al., 2013). The main characteristics of this implementation establish that:

•In these databases, the documents use a hierarchical tree structure implemented through maps, collections, and scalar values. All documents are stored in the same way, but the content in each one is different.

•Each document can be read and entirely written in one transaction.

•The documents are independent, improving the database performance, and reduce the concurrency effects.

It is important to note that data may require in each operation, a single document depending on the design implemented, eliminating joins or transactions between objects. Thus, the design is essential for modeling information requirements. In this study, for the NoSQL representation, MongoDB was used. MongoDB is an agile and scalable NoSQL database, based on a document-oriented model, whose data is stored in separate documents within a collection (www.mongodb.com).

4. Methods

Requirements for a job search system were considered to structure different data models. These data models were implemented and populated using a migration strategy in order to consolidate a useful environment for the test's development, allowing the evaluation of both functionality and the efficiency of each data storage. Python was used for: the work with the MongoDB database through the Pymongo library, access to PostgreSQL features through the Psycopg library and development, and execution of test strategies (www.python.org). Measurement was made according to key indicators to analyze the performance of the data models. Figure 1 illustrates this methodological process, starting with the conception of the model considering the needs of a job search engine to the evaluation of database environments.

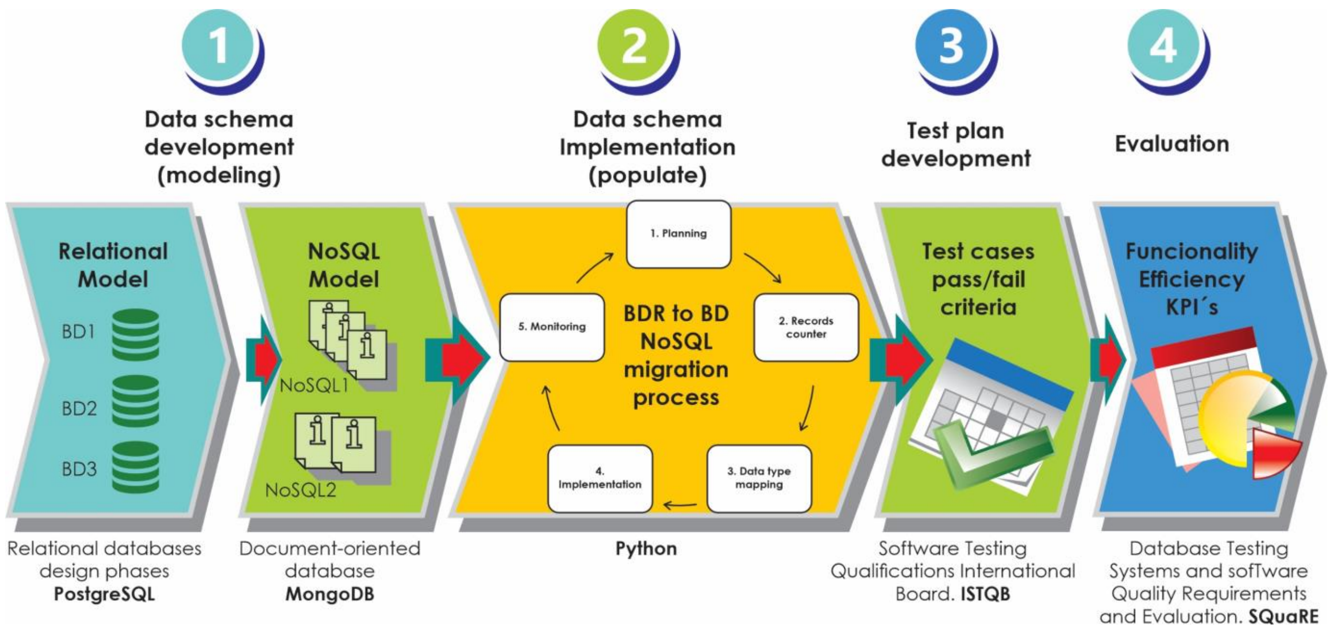
1
Data schema development (modeling)
2
Data schema Implementation (populate)
3
Test plan development
4
Evaluation
Relational Model
BD1
BD2
BD3
NoSQL Model
NoSQL1
NoSQL2
1. Planning
2. Records counter
3. Data type mapping
4. Implementation
5. Monitoring
BDR to BD NoSQL migration process
Test cases pass/fail criteria
Funcionality Efficiency KPI´s
Relational databases design phases PostgreSQL
Document-oriented database MongoDB
Python
Software Testing Qualifications International Board. ISTQB
Database Testing Systems and software Quality Requirements and Evaluation. SQuaRE

Figure 1. Development methodology.

4.1. Data model definition

In order to accomplish our objectives, three data schemas were constructed: The first one was based on relational systems, and the other two were based on non-relational systems. The two NoSQL data models development were made to assess both advantages and disadvantages in models with the same functionality but different structuration.

Relational data schema development: Schema design

The software requirements were determined for the relational model, and then, it was structured in entities, attributes, and relationships through a logical design. In the last step, the relational data fields mapping was developed, completing all aspects of the physical implementation for PostgreSQL.

The relational model sets the data type of each column, the foreign keys, and the business rules implemented using constraints. This data model was selected as a pattern for the subsequent models. Figure 2 shows the relational model obtained for the job search application implementation.

For the relational model, we created three databases: DB0, DB1 and DB2. For this, a script was developed that contains the SQL statements for the creation of each table, constraints and necessary business rules. These databases were populated through data insertion scripts, subsequently the data in each table was counted and the size of the tables in each database was recorded. The summary of this information is shown in Table 1.

4.2. Mapping strategies

Both relational and document-oriented MongoDB are heterogeneous databases supported by different technologies. For this reason, a rules transformation between these two environments (PostgreSQL and MongoDB) is necessary (Fouad & Mohamed, 2019). A first approach was made considering the relational model developed and the system requirements (availability, consistency, scalability) as a whole,

CRUD operations, entities and the number of records possible to process (Imam et al., 2019).

Development of NoSQL data schemas.

The NoSQL data scheme was made through data conversion from a traditional relational database to MongoDB (Mearaj et al., 2019). For NoSQL structure design, document-oriented approach was conducted, with MongoDB system, representing all the relationships through references and embedded documents. Two functionality strategies were defined to evaluate performance considering two design alternatives:

Strategy 1. Three different collections

For this data schema, the information was modeled in three different collections, as shown in Figure 3.

• JOBSEEKER Collection. A collection where all the information with respect to users is stored for those who are looking for job offers. Education, work experience, skills, and applications are structured as an array of embedded-

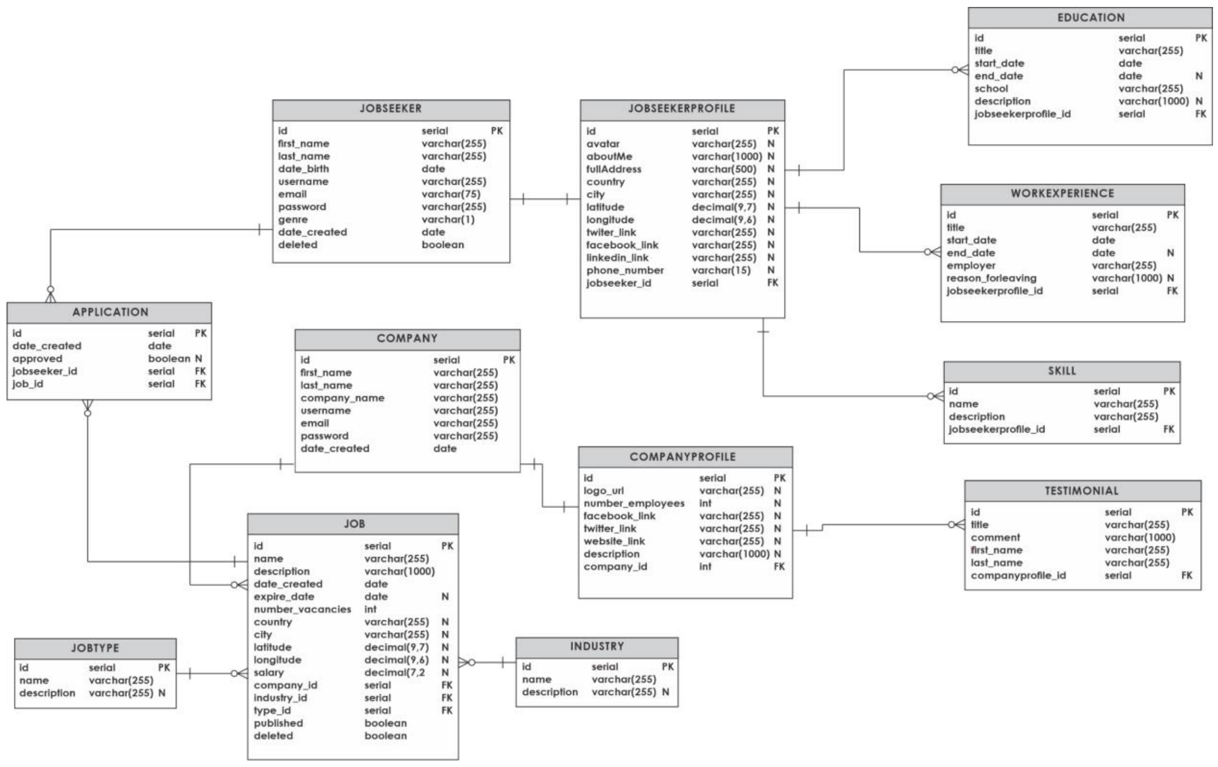

Figure 2. Relational data model.

documents because the number of the maximum element possible to contain is between 10 and 20.

• COMPANY Collection. This collection stores information regarding companies. The job identifiers created by the company are included. Each element within the array is referenced by the _id field of the JOB collection.

• JOB Collection. This collection stores information regarding all job offers created by companies.

For this data schema, we model the information in two collections named JOBSEEKER and COMPANY. Documents structure in both collections is similar to those presented in strategy 1, but documents stored in JOB collection are embedded in the COMPANY collection in this new data schema, as seen in Figure 4.

4.3. The Evaluation method

In order to achieve our goals, the next step was to compare models. We started with selecting the tools and database engines. Then, we prepared data sets and test use cases. Finally, we analyzed the results of the executed tests.

4.3.1. Tools and database engines

All implementations were tested in the same run configuration to obtain the most adequate comparison. The machine employed in the tests had:

• Intel i7-3630QM 2.4 Ghz quad-core processor,
• 8 GB RAM,
• 1 TB of solid-state storage,
• Ubuntu 14.04 LTS

The test uses the following databases:
• PostgreSQL 9.3
• MongoDB 3.2.0

Utilities:
• Programming language: Python 2.7
• PostgreSQL connection driver: Psycopg 2.4.5
• MongoDB connection driver: Pymongo 3.1.1

4.3.2. Test preparation

The objective of the tests was to evaluate the presence of conditions and necessary elements in the three database designs and to verify their performance. To do this, we considered the following database options:

• Use of database indexes. An index is a database structure that can be used to improve the performance of database activity.

• Query Development and Derived Structures. A query is a request for data or information from a database. Here we evaluate the possibility of information retrieval with the use of various structures and operators.

• Design of scalable databases. Scalability is the ability of a database to handle growth in the amount of data and users. To do this, structural tests are handled, evaluating the different design options.

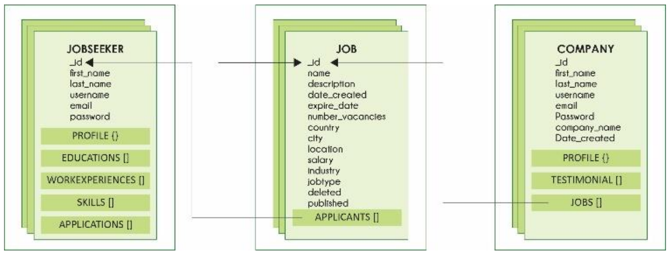

Figure 3. NoSQL1 schema.

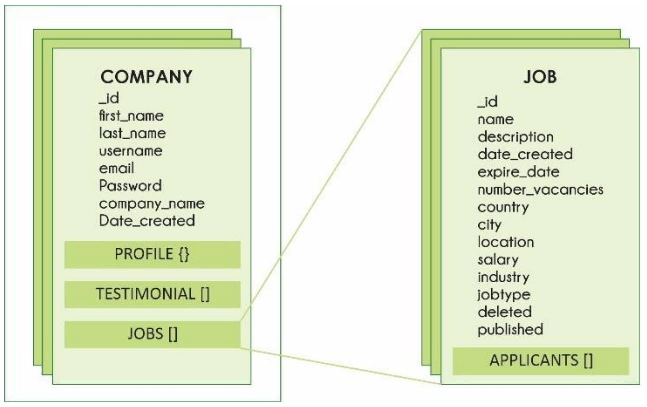

Figure 4. NoSQL2 schema.

Three Database tests were developed: structural tests, functional tests, and non-functional tests. For the relational model, the evaluation strategy was oriented to the data organization in order to develop standard tasks for a job search engine: 1) loading of job offers, 2) loading of candidate data, 3) offers search, and 4) candidates search. SQL clauses provided the benchmark for relational and non-relational environments testing. The non-functional tests were concentrated around adequacy, precision and efficiency metrics (Tziatzios, 2019).

Besides, for the NoSQL models, four points were emphasized:

• Technology knowledge on how data was stored (Document)

• Understanding about different file formats used (JSON)

• How the data can be accessed (collections)

• Test strategy in terms of both data conversions and comparisons

For this last step, is required to populate the NoSQL database; for this reason, a migration strategy was developed. Data migration is the process of data transfer among data storage systems, data formats, or computer systems (Ghotiya et al., 2017). In a NoSQL database, data migration not only includes data transfers from one database to another but also requires adaptation of structures and models that fit the final database without affecting data accuracy and integrity (Wijaya & Akhmadarman, 2018).

For data migration strategy, the following procedure were developed:

I.Planning. In this step, it was determined how to develop the migration process, for this, the entity–relationship model and the NoSQL database schemas proposed in each strategy were used. Python scripts were built to fill each collection.

II.Records counter. The records that required to be migrated were counted from the SQL database and it was chosen in what collection should be stored inside the NoSQL database.

II.Data type mapping. All SQL schema fields were formatted into a valid data type, to be later migrated into non-relational schemas.

IV.Implementation. Migrated-scripts execution in Python.

V.Monitoring. Verification in the information subject to migration. The number of records in the SQL databases was compared with the number of records migrated. Also, the records in both systems were consulted to study whether the information returned was the same.

Additionally, a workload-based approach was employed (Beach et al., 2020), by a set of predefined queries, representative of the tasks performed in the job-seeking systems.

4.3.3. Workloads

Once each data model was designed, an implementation process was developed for each database technologies respectively. To compare the characteristics that are the object of study, three data repositories were created for each model; each database was differently according their number of records stored. For each PostgreSQL database, all the information was migrated to the schemas created in MongoDB (Antaño et al., 2014). Table 1 shows data distribution in the relational databases.

For the implementation of the non-relational schemas, we created three databases for each NoSQL data model in MongoDB. To populate the databases in MongoDB, the process of migrating data from SQL systems to NoSQL described above was performed. Table 2 shows data distribution in the non-relational databases after migration.

For models comparative analysis, a test plan was designed according to the method for software testing proposed by the Software Testing Qualifications International Board (ISTQB® International Software Testing Qualifications Board, 2019). The procedure followed to develop the tests is described in Figure 5.

Table 1. Data distribution in relational databases.

Table name	DB0		DB1		DB2	
	# Records	Mb	# Records	Mb	# Records	Mb
Job seeker	50.000	14	500.000	138	1.500.000	422
Job seeker profile	50.000	27	500.000	266	1.500.000	798
Education	100.261	34	850.380	284	2.349.904	785
Work experience	99.487	33	675.467	226	1.675.350	560
Skill	150.267	45	1.000.088	301	2.999.892	902
Application	100.964	10.9	635.467	63	3.051.520	309
Company	10.000	3.2	200.031	64	725.058	234
Company profile	10.000	4.7	200.031	91	725.058	329
Testimonial	19.854	6.6	299.629	97	1.087.813	350
Job	50.268	19	1.000.000	419	3.000.000	1258
Job type	5	0.031	11	0.031	11	0.031
Industria	62	0.031	5062	0.56	5062	0.56

Table 2. Data records distribution in NoSQL databases.

Collections	nosql#1-0	nosql#1-1	nosql#1-2	nosql#2-0	nosql#2-1	nosql#2-2
Job seeker	50.000	500.000	1.500.000	50.000	500.000	1.500.000
Company	10.000	200.031	725.058	10.000	200.031	725.058
Job	50.268	1.000.000	3.000.000			

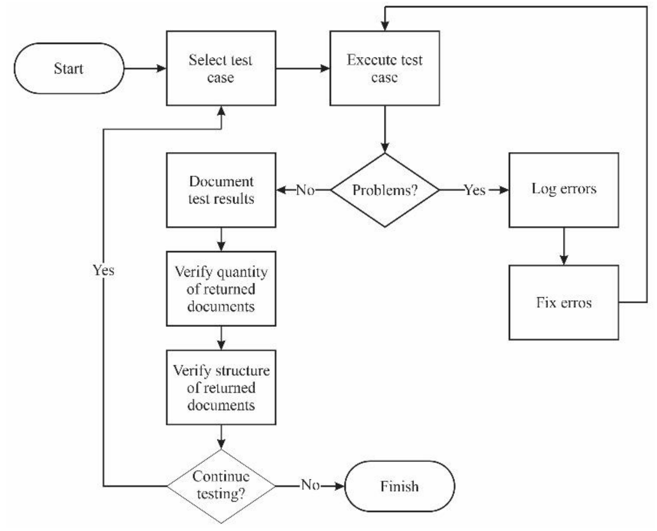

Figure 5. Testing procedure.

In this procedure, each test case is selected and run. In case of problems in its execution, the failure is documented and corrected. The procedure also included the document verification process in terms of quantity and structure, to assess the integrity of the data returned in both architectures.

4.3.4. Metrics

Testing cases were oriented for both functionality and efficiency criteria established by the Systems and software Standard Quality Requirements and Evaluation (SQuaRE) (ISO, 2014):

• Functionality defined as the satisfaction of need expression and measured by the adequacy and accuracy.
• Efficiency, associated with the response time exhibited by the different data models

The test cases were built using the operations allowed in the structured query language. Table 3 lists the scope of the test plan established for the study and shows the implementation used to evaluate each functionality in every single data model.

In each test case it was verified:

• The application stores the transaction information in the application database and displays them correctly to the user.

• No information was lost in the process.
•Neither partially-performed nor aborted operation information was saved by the application.

For this reason, the pass / fail criteria applied to test cases were:

• Functionality tests:
◌ Step: The number of records returned in the queries made by each used data schema, match. In the case of non-coincidence, a check of the results set of records was developed, verifying the returned information is the same.
◌Failure: If the number of elements returned in the schemas differ or the information returned is not the same.
• Efficiency tests:
◌ Step: If the query executed is successful.
◌ Failure: If the query executed has some error.

5. Results

When cases test execution ran, metrics were performed as key performance indicators for both model evaluation and comparison. Table 4 shows the formal specification of the metric. We presented the goal, the method followed for its measurement and the calculation formula.

After measuring, we obtained the following results:

• Adequacy, 95% of the evaluated functionalities comply with the NoSQL schemas concerning the SQL schema, and there are certain limitations regarding joins, specifically in inner joins between collections. Table 5 shows the results.

In Table 5, (A) represents the number of cases that failed out of the number of possible cases (B). We can see that MongoDB presented limitations in the union operations. This limitation was related to the design strategy that handles more than one collection.

◌ Results accuracy measured with the AC1 metric. In Table 6 and Table 7, (A) represents the number of test cases where the number of records does not match and (B) the number of test cases developed.

Results in Table 6 and Table 7 shows that:

◌ Schema-based in three collections (NoSQL #1): there was a difference in the unions, observing that 33% of returned records was the same concerning the SQL schema. These results were achieved with $lookup operator, which, together with the operator $match, recreate an inner join between collections. The missing percentage, 66.7%, is due to: a) absence of records when performing a left join between collections and b) an inner join was simulated when performing two independent queries.

◌ Schema-based on two collections (NoSQL #2): only in the unions, the number of returned records does not match those returned by SQL schema (according to the queries executed in the test cases), and it was observed the consulted records are returned in a single document due to their modeling form.

◌ For both schemas, in terms of record aggregation, row modification, logical and comparison operators, the number of returned records is the same for SQL schema.

• Accuracy measured through AC2 metric (returned records) are shown in Table 8:

◌ In the case of combine rows from two or more tables, based on a related column between them (joins) for the three-collection schema (NoSQL #1), 66.7% of the records match those returned by the SQL schema. The remaining 33.3% was due to the returned records differed in the left join between collections.

Table 3. Test plan scope.

Category	SQL statements and operators	MongoDB implementation
Logical operators	And, or, not	$and, $or, $not
Record aggregation	Group by, having, count, avg	$group, count, $sum, $avg, $match
Comparison operators	Between, >,<,>=,>=, in, not in, like	$gte, $lte, $gt, $eq, $in, $nin, $regex
Unions	Inner join, left join	Two independent consultations | operators $lookup y $match
Record modification	Lower, upper, trunc	$toLower, $toUpper, $trunc
DML statements	Insert, update, delete	Insert, update, update_many, Delete, delete_many
Record ordering	Order by	$sort
Restrictions	Constraints	Document validation
Database indexes	Database indexes	
Row selection		Projection

Table 4. Metrics specification.

Goal	Method	Test metric
Adequacy (AD): Verify how complete is the functionality provided by each database system.	Count the number of missing operators in the evaluation and compare with the operators specified in the test case	$1 \leq n \leq 12 \;\; n = \frac{\lvert S \rvert}{2} \;\; y\, S \subseteq \mathrm{C}$ C={and, or, not, =, >= , <= ,<, >,, group by, avg, having, order by, in, like, lower, upper, trunc, inner join, left join, insert, update, delete, index, constraints}
Accuracy (AC1): Verify that the number of records returned are the same in each data model	Count the number of test cases where the number of returned records do not match and compare with the number of test cases performed	$X = \left(1 - \frac{A}{B}\right) * 100$ X= Percentage of test cases where the number of returned records is the same. A= number of test cases where the number of records do not match B= number of test cases developed
Accuracy (AC2) Verify that the returned records are the same in each data model	Count the number of test cases where the returned records do not match and compare with the number of test cases performed	$X = \left(1 - \frac{A}{B}\right) * 100$ X= Percentage of test cases where the returned records are the same. A= number of test cases where the records do not match. B= number of test cases developed
Efficiency (EF): Obtain response times by testing certain functionalities under some conditions	T = Time calculated in microseconds	Direct metric

◌ On the other hand, the two-collection schema (NoSQL #2) exhibited a 100% match with the SQL-schema results because NoSQL#2 structure makes a collection's joint an unnecessary step since information is returned as a single document.

• In terms of efficiency:

◌ In the relational database, different response times can be obtained depending on the number of fields returned in the query. When compared to the NoSQL # 1 model, this behavior did not happen; that is, the differences between the response times are minimal.

◌ The insertion of records in the NoSQL # 2 schema is faster compared to the relational schema and NoSQL # 1; this is because all the information is inserted in a single document

◌ To sort the fetched data, must be specified a limit of records to return or define an index on the field on which sort is being performed

◌ The response times obtained during the sorting of records in the NoSQL schemas are considerably shorter compared to those obtained in the relational model.

◌ In the relational model, response times when grouping records can vary considerably depending on the number of returned records grouped and the number of fields by which they are grouped. When records exceed 50k, the processing time decreases considerably between SQL schema and NoSQL.

Results from test cases are summarized in Table 9.

We did an analysis of the database transactions through the atomicity, consistency, isolation and durability properties, we found that MongoDB guarantees these properties when applications tend to handle all the information within a single document. When it comes to transactions involving multiple documents and multiple collections, MongoDB does not fully secure ACID properties. NoSQL databases follow BASE (Basically Available, Soft State, Eventual consistency) principles. This fact means that, in the event of a failure, two scenarios can occur: in the middle of some writing operation, the changes applied up to the point of failure cannot be reversed, and the users who are doing reading operations can obtain a certain degree of inconsistency in the results they are displaying. This behavior was evidenced in the three-collections strategy model because the information of the companies and the job offers are in different collections.

Table 5. Adequacy results.

Functionality	PostgreSQL			MongoDB		
	A	B	X	A	B	X
Logical operators	0	3	100%	0	3	100%
Comparison operators	0	5	100%	0	5	100%
Aggregation functions	0	4	100%	0	4	100%
Records modification	0	3	100%	0	3	100%
Unions	0	2	100%	1	2	50%
Constraints	0	1	100%	0	1	100%
DML sentences	0	3	100%	0	3	100%
Database indexes	0	1	100%	0	1	100%
Total	0	22	100%	1	22	95%

Table 6. Accuracy (AC1) metric results SQL Schema and NoSQL #1 schema.

Functionality	SQL Schema			NoSQL #2 schema		
	A	B	X	A	B	X
Logical operators	0	1	100%	0	1	100%
Comparison operators	0	4	100%	0	1	100%
Aggregation functions	0	4	100%	0	3	100%
Records modification	0	2	100%	0	1	100%
Inner Joins	0	3	100%	1	1	0%
Total	0	14	100%	1	7	85.7%

Table 7. Accuracy (AC1) metric results SQL Schema and NoSQL #1 schema.

Functionality	SQL Schema			NoSQL #1 schema		
	A	B	X	A	B	X
Logical operators	0	1	100%	0	1	100%
Comparison operators	0	4	100%	0	4	100%
Aggregation functions	0	4	100%	0	4	100%
Records modification	0	2	100%	0	2	100%
Inner Joins	0	3	100%	2	3	33.3%
Total	0	14	100%	2	14	85.7%

Table 8. Accuracy (AC2) metric results.

Functionality	SQL Schema			NoSQL #1 schema			NoSQL #2 schema		
	A	B	X	A	B	X	A	B	X
Logical operators	0	1	100%	0	1	100%	0	1	100%
Comparison operators	0	4	100%	0	4	100%	0	1	100%
Aggregation functions	0	4	100%	0	4	100%	0	3	100%
Records modification	0	2	100%	0	2	100%	0	1	100%
Inner joins	0	3	100%	1	3	66.7%	0	1	100%
Total	0	14	100%	1	14	92.8%	0	7	100%

Table 9. Results efficiency test cases.

Functionality	Number of records returned	Time SQL schema	Time NoSQL #1 schema	Time NoSQL #2 schema
AND, OR , NOT	22.857	1.17	2.52	5.25
	69.219	3.39	8.02	15.65
>=, <=, >, <	659.756	11.48	7.75	15.02
	1.978.208	35.83	24.89	53.16
IN	94.455	1.59	2.04	Not Apply
	281.299	4.34	6.13	
LIKE	99.550	4.82	14.34	Not Apply
	300.182	5.61	19.92	
AVG, GROUP BY 1 variable	155	0.56	0.92	3.56
	155	1.47	2.83	10.54
AVG, GROUP BY 2 variables	55.682	32.75	2.07	4.58
	55.682	112.33	6.39	13.12
HAVING	258	0.56	2.62	Not Apply
	8.332	2.32	5.56	
ORDER BY	327.319	17.42	error	Not Apply
	1.057.029	63.70		
ORDER BY	1000	5.58	1.12	Not Apply
	1000	21.00	3.36	
SELECT all columns	1000	1.60	1.23	Not Apply
	1000	6.07	3.53	
SELECT de 7 columns	1000	0.75	1.15	Not Apply
	1000	2.00	3.75	
INSERT de 50k records	Not Apply	48.53	47.18	28.80

6. Conclusions

In this study, we present a test model for database architectures. To do this, we create relational and non-relational database designs applied to an employment-oriented online service. We use a structured method in four phases: modeling (data schema development), population (data schema implementation), development of a test plan and metric evaluation. Our main contributions are in the development of each phase.

Regard the data modeling strategy, initially we show a relational design considering the requirements for the basic operations of such a system. Then, two document-oriented design strategies were proposed to know the scope and limitations of each solution. For the relational data schema was defined tables, attributes, relationships and constraints to avoid redundancy and guarantee data integrity. In the case of NoSQL schema, there is some flexibility in data modeling, since it is not mandatory to have a predefined schema and each document can have a different structure. However, to improve its performance, it was important to define aspects such as number of collections, unions between collections, frequency of read and write operations on documents, data duplication and advantages and disadvantages of embedded documents.

For the development of test strategies for both relational and non-relational databases, we use a framework for migration from one environment to another. The migration strategy for populating the NoSQL databases used Python scripts to populate each collection. Monitoring was developed through a record counter and a verification of the migrated information.

Finally, we have conducted a metric-guided evaluation to track results and provide recommendations for the different database schemas implemented. In this evaluation, the performance metrics used verify consistent transactions, consistent data retrieval operators and flexibility in schema design. The most important findings show that the queries executed in NoSQL data schemas can have a different structure and can depend on several factors: operators that are implemented, modeling of each data schema, information that the user wants to return in each query, the need or not to link between collections to get the required records.

As a result of testing, the SQL schema provides increased functionality by ensuring transaction compliance and data integrity.

Regard the NoSQL schema structured in three collections, certain limitations appear when making union between collections and transaction limitations. However, in response to bigdata processing, response times were very favorable, although this model also has some level of data duplication.

The NoSQL schema based on two collections avoids the need to perform join operations to get the result record set. As all the information is practically in a single document, it is possible to comply any transaction. Additionally, the inserts in this model are fastest than the other designs.

The choice of the model that best suits the job-oriented online service proposed for this study depends on the existing needs, in case transactions and data integrity are essential and response time can be sacrificed in some degree, it is recommended to use the SQL schema. . If response times are a fundamental part of the application, transactions are not 100% essential and data duplication is accepted, it is recommended to use any of the NoSQL schemes, considering the differences that exist between them. Another strategy suggests migrating to a hybrid model that combines SQL and NoSQL databases. According to (Sokolova et al., 2020), this approach adds flexibility, mobility and efficiency to the exposed data management system.

Conflict of interest

The authors have no conflict of interest to declare.

Financing

The authors received no specific funding for this work.

References

Antaño, A. C. M., Castro, J. M. M., & Valencia, R. E. C. (2014). Migración de Bases de Datos SQL a NoSQL. Tlamati, Especial 3, 144-148. CICOM.

Beach, P. M., Langhals, B. T., Grimaila, M. R., Hodson, D. D., & Engle, R. D. L. (2020). A Methodology to Identify Alternative Suitable NoSQL Data Models via Observation of Relational Database Interactions. *Theses and Dissertations*. 4339.

Bugiotti, F., Cabibbo, L., Atzeni, P., & Torlone, R. (2014). Database design for NoSQL systems. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8824*, 223–231. https://doi.org/10.1007/978-3-319-12206-9_18

Codd, E. F. (1971). A data base sublanguage founded on the relational calculus. In *Proceedings of the 1971 ACM SIGFIDET (now SIGMOD) workshop on data description, access and control* (pp. 35-68). https://doi.org/10.1145/1734714.1734718

Date, C. J. (2000). An Introduction to Database Systems: eBook Addison-Wesley Longman Publishing Co., Inc. (7th Ed.).

Ercan, M. Z., & Lane, M. (2014). An evaluation of NoSQL databases for EHR systems. In Proceedings of the 25th Australasian Conference on Information Systems. Auckland University of Technology, School of Business Information Systems.

Fouad, T., & Mohamed, B. (2019). Model Transformation From Object Relational Database to NoSQL Document Database. In *Proceedings of the 2nd International Conference on Networking, Information Systems & Security* (pp. 1-5). https://doi.org/10.1145/3320326.3320381

Fraczek, K., & Plechawska-Wojcik, M. (2017). Comparative analysis of relational and non-relational databases in the context of performance in web applications. In *International Conference: Beyond Databases, Architectures and Structures 716*, 153-164. https://doi.org/10.1007/978-3-319-58274-0_13

Ghotiya, S., Mandal, J., & Kandasamy, S. (2017). Migration from relational to NoSQL database. In *IOP Conference Series: Materials Science and Engineering*, 263(4), 042055. https://doi.org/10.1088/1757-899X/263/4/042055

Han, J., Haihong, E., Le, G., & Du, J. (2011). Survey on NoSQL database. In *2011 6th International Conference on Pervasive Computing and Applications*. 363-366. IEEE. https://doi.org/10.1109/ICPCA.2011.6106531

Hows, D., Membrey, P., & Plugge, E. (2014). MongoDB Basics. In *MongoDB Basics*. Apress. https://doi.org/10.1007/978-1-4842-0895-3

Imam, A. A., Basri, S., Ahmad, R., & González-Aparicio, M. T. (2019). Schema proposition model for NoSQL applications. *Advances in Intelligent Systems and Computing*, *843*, 30–39. https://doi.org/10.1007/978-3-319-99007-1_3

Imam, A. A., Basri, S., Ahmad, R., Watada, J., & González-Aparicio, M. T. (2018). Automatic schema suggestion model for NoSQL document-stores databases. *Journal of Big Data*, *5*(1), 46. https://doi.org/10.1186/s40537-018-0156-1

ISO. (2014). *ISO/IEC. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE*. https://www.iso.org/obp/ui/#iso:std:iso-iec:25000:ed-2:v1:en

ISTQB® International Software Testing Qualifications Board. (2019). *Downloads - ISTQB® International Software Testing Qualifications Board*. https://www.istqb.org/downloads.html

Kumar, M. S., & . Jayagopal, P.. (2018). Comparison of NoSQL Database and Traditional Database-An emphatic analysis. *JOIV: International Journal on Informatics Visualization*, *2*(2), 51. https://doi.org/10.30630/joiv.2.2.58

Marqués, M. (2011). *Bases de datos* (Publicacions de la Universitat Jaume I. Servei de Comunicació i Publicacions (Ed.); Primera edición).

Mearaj, I., Maheshwari, P., & Kaur, M. J. (2019). Data conversion from Traditional Relational Database to MongoDB using XAMPP and NoSQL. In *2018 Fifth HCT Information Technology Trends (ITT)* (pp. 94-98). IEEE. https://doi.org/10.1109/CTIT.2018.8649513

Elmasri Ramez, & Navathe Shamkant (2022). *Fundamentals of Database Systems. eBook. In S. Dissano (Ed.), USA (7th ed.). Pearson.*

Paredaens, J., De Bra, P., Gyssens, M., & Van Gucht, D. (1989). Relational Database Model. In The Structure of the Relational Database Model. *EATCS Monographs on Theoretical Computer Science*, vol 17. https://doi.org/10.1007/978-3-642-69956-6_1

Scherzinger, S., Klettke, M., & Störl, U. (2013). Managing schema evolution in NoSQL data stores. https://doi.org/10.48550/arxiv.1308.0514

Schreiner, G. A., Duarte, D., & dos Santos Mello, R. (2020). Bringing SQL databases to key-based NoSQL databases: a canonical approach. *Computing*, *102*(1), 221-246. https://doi.org/10.1007/s00607-019-00736-1

Sokolova, M. V., Gómez, F. J., & Borisoglebskaya, L. N. (2020). Migration from an SQL to a hybrid SQL/NoSQL data model. *Journal of Management Analytics*, *7*(1), 1–11. https://doi.org/10.1080/23270012.2019.1700401

The PostgreSQL Global Development Group (2013). *The world's most advanced open source database*. https://www.postgresql.org/

Tziatzios, D. (2019). Model-based Testing for SQL Databases. http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-269424

Wijaya, Y. S., & Arman, A. A. (2018). A framework for data migration between different datastore of NoSQL database. In *2018 International Conference on ICT for Smart Society (ICISS)* (pp. 1-6). IEEE.
https://doi.org/10.1109/ICTSS.2018.8549944

Zafar, R., Yafi, E., Zuhairi, M. F., & Dao, H. (2016, May). Big data: the NoSQL and RDBMS review. In *2016 International Conference on Information and Communication Technology (ICICTM)* (pp. 120-126). IEEE.
https://doi.org/10.1109/ICICTM.2016.7890788