The Quest for a Database Selection and Design Method

Noa Roy-Hubara

Ben-Gurion University of the Negev
nro@post.bgu.ac.il

Abstract. New types of database have emerged over the last decade, aimed at answering new requirements in the Big Data era. The new databases, in additional to the Relational model, may fit to specific types of applications. Therefore, new challenges have also emerged, including the issue of which database model to select for a given application, and how to design the database based on the selected model. To the best of our knowledge, these two challenges have not been addressed by any systematic method. In this research we plan to devise a structured method for database model selection and design based on variety of factors, including data-related requirements, functional requirements, and non-functional requirements. Based on these requirements the method will recommend which database models are the most appropriate for that application and will suggest a design for the recommended models.

Keywords: Database Selection, Database Design, Database Models, NoSQL

1 Introduction

The last decade had brought new advancements to the database domain with a tremendous growth in the field with new data models and providers. These advancements have arrived after more than four decades of the dominance of the relational model which still remains a leading database solution[1]. Over the years, other database models and systems emerged, including object-oriented (OO), and in the last decade, the NoSQL and the NewSQL databases. Object-Oriented databases are integrated with object-oriented programming languages, and thus overcome the gap between the Relational database and the OO programming languages. NoSQL databases are flexible, horizontally scalable databases that aim at overcoming the Relational databases rigidity. NewSQL databases are a combination of the Relational and NoSQL databases, aimed to converge the advantages of both technologies.

With the new technologies, new challenges have risen. Since many database models are available nowadays, there is a challenging need to select the most suitable database model (or models) and systems for a specific application. In addition, the new DBMSs are less studied and therefore lack structured design methods for their creation and design, as exist for the relational databases. These two main challenges may result in an unfitting database and/or database design for an application, a fact that might lead to

[1] https://db-engines.com/en/ranking

many changes and rollbacks in the development lifecycle. These changes, rollbacks and detours in the quest for the right database system and its design consume both time and money.

Addressing the selection process in research is done in studies which mostly compare different DBMSs based on technical aspects such as replication type, atomicity type, data model, etc. These studies are useful for becoming familiar the new databases; however, they hardly deal with how well the various databases fit with the specific requirements of an application. In addition, practitioners who deal with the selection problem hardly perform an analysis of the problem (data analysis, goal analysis) for finding most fitting DBMSs. In regarding the second challenge, of database design, several new methods were proposed. However, it seems that none of the methods is widely adopted, and practitioners design new databases based on best practices and trial and error processes.

In this research we plan to propose and implement a method for a database selection and design. The sought method will emphasize the users' requirements, including data-related requirements, functional requirements and non-functional requirements. The proposed method would take into account all needed requirements in the database selection and design process and will assist practitioners to choose from the variety of database models and solutions. The sought method will support the much-needed analysis that practitioners require, but lack to perform.

The rest of the paper is structured as follows. In Section 2 we review the state-of-the-art. In Section 3 we elaborate on the research methodology. In Section 4, we briefly describe our initial suggestion for the method. Finally, in Section 5 we summarize and elaborate on our plans for future research.

2 Current Status

2.1 In database selection

To the best of our knowledge, no structured method for database selection is available. Various surveys, such as [4, 5, 8, 10, 11, 18, 20], analyze characteristics, capabilities and benefits of various database technologies. These characteristics include supported query languages, index implementation, availability, consistency, durability, security, support for transactions, and license types. Yet, these surveys usually do not deal with the issue of how to select a database technology based on the users' needs and the requirements of the application.

We found studies that refer to the issue of database selection to a limited extent. For example, [1] compared different graph databases and their features, including storing features, querying features and data structures. [7] and [9] conducted empirical comparisons of different types of workloads, such as data insertion time and traversal time for different databases. The authors of [19] compared the performance of five NoSQL databases. They excluded graph database providers from their study since they claim that its use cases are different from the other three NoSQL database models. They defined three types of workloads and tested execution time and throughput for the five databases. While the study involved DBMSs of specific providers, the authors deduce

that "Document databases, followed by Column-family databases, have a good average performance since they own both efficiency and scalability". In [3] six database systems were compared based on different, divided into functional and non-functional requirements, and techniques. With respect to functional requirements, the authors checked supported types of queries, such as sorting, joins, transactions, etc. With respect to non-functional requirements, they compared latency and availability. With respect to techniques, they looked at technical aspects such as replication, logging and analytic framework. The authors also provided a decision tree that maps some of the aspects to the different database providers.

While all presented surveys provide a valuable understating to different models and their characteristics, they do provide a structured way to assist programmers choose the right model/technology based on their application requirements.

2.2 Database design

To fully understand the current situation of the new databases' design, we surveyed several design methods [14]. In addition, we performed a systematic literature review [15], in which we found 24 new methods and assessed them based on different criteria. We found the field is definitely gaining more attention and generated several interesting findings.

First, most methods used a known conceptual model to represent the data such as ERD or UML class diagram. These models are widely accepted and used, and helpful in describing a domain, probably even one less structured, in an understandable manner. The new methods defined a set of rules and/or definitions to utilize the conceptual models appropriately. It seems that a usage of a known model to define the data structure makes a method more appealing to users since it would require less effort in learning new conceptual models.

Second, there is tradeoff between a method generality and its complexity. When a method is tailored to a specific database provider, it is not usable in other databases of the same model. However, if a method is fitted to all types of databases, then it is more complex and harder to learn and use. Hence, most studies choose to focus on one specific NoSQL database type, which is more inclusive than a method that is tailored to one specific provider.

Functional and Non-functional requirements are addressed in new methods to a limited extent. Functional requirements (i.e., queries) are very important in the NoSQL world. NoSQL databases do not support some concepts that exist in the relational databases such as joins, nested queries, etc. [2]. Due to this fact, when designing NoSQL databases, it is crucial to consider the needed queries. A design that would not take the queries into effect might be ineffective in answering the desired queries. In regarding the Non-functional requirements, we believe it is important to address them as well, mostly when choosing a database solution. Since most methods assume that a database was chosen a-priori, most NFRs are not addressed. We believe that choosing the right type of databases for specific tasks is a crucial step in the design process. Such decision would save time and money and would reduce the need to redesign databases.

3 Research Methodology

As in this research we aim at devising new selection and design methods, we plan to adopt the design science approach [6]. Figure 1 presents the context and the main steps we are planning to carry in this research.

Following the design science approach, we performed a requirement analysis of what is needed for the tasks at hand. In particular, we started with the research point of view and systematically reviewed related studies. We further created and distributed a questionnaire to assist in understanding the current situation. The questionnaire aims at understating the current status practitioners facing regarding the processes of database selection and design. We believe that insights form such a survey will provide the basis for shaping the needed methods.

Based on practitioners' answers to the questionnaire we further plan to interview and ask experts and practitioners for their experience in database selection and design. Our initial analysis and the added information from the experts will facilitate the creation of a comprehensive set of requirements for the sought methods. Based on this set of requirements we plan to devise proper models, rules, and guidelines. These will be later implemented in a software tool that will support all artifacts. We further plan to evaluate the artifacts using various techniques. We plan to implement and evaluate the proposed method and implementation. Based on the results to further refine it. Table 1 elaborates on the guidelines adapted from [6].

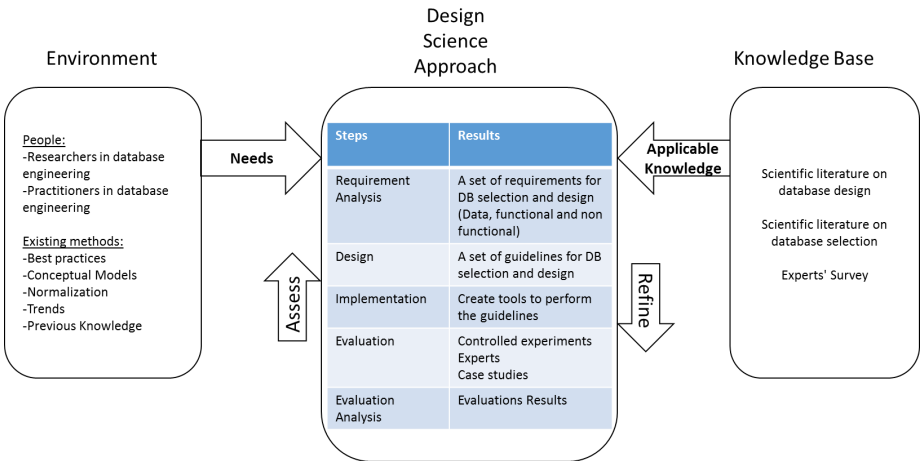

Fig. 1. Research Methodology adapted from [6]

Table 1. Guidelines on design science research and their instantiation

Name	Instantiation in the context of this research
Design as an artifact	In our research we have several artifacts, models for specifying the requirements, guidelines and rules for database selection, database fragmentation and database design.
Problem relevance	The problem we address is the process of database selection for an application and/or domain and designing the database in the best possible manner. This is appealing as the variety of database solutions increases and the need to select the right ones for specific applications.
Design evaluation	The method as a whole and the different artifacts will be evaluated and re-evaluated in order achieve the best possible solution. It will be evaluated with controlled experiment, use-cases and by experts' reviews.
Research contributions	The research will tackle an almost unreferenced process of creating a database from start to finish: from choosing to designing, using new methods for the entire process.
Rigorous research design	The research approach will follow the design science research approach.
Design as a search process	We plan to try several approaches in order to achieve the best possible results.
Communication of research design	We plan to publish our work throughout the process. Currently, SLR is under review.

4 Preliminary Method Proposal

The proposed method for selecting and designing database models and systems considers various types of users' requirements. It consists of the following steps:

1. Gather and specify the **data-related** requirements and express them using a conceptual data model. In this work we use the UML class diagram, chosen based on its widespread use.
2. Gather and specify the **functional** requirements that are related to database operations, i.e., data retrievals and updates operations. Hereafter we call them queries.

3. Gather and specify the **non-functional** requirements (NFRs) that are related to the data requirements and the queries.
4. Based on the above, the method considers dividing the conceptual data model into fragments, each of which has different characterizations (such as different access frequency, different performance requirements, and different consistency requirements).
5. Select the most suitable database model/system for each fragment. This will be based on a general-purpose pre-defined profile for each database model. A pre-defined profile consists of a set of non-functional properties associate with each database model.
6. Design the selected database model/system for the different fragments.

Due to space limitation we will focus on steps 4-6, which constitute the main steps of the proposed method. A preliminary example for steps 1-5 can be found on [16].

Step 4 – Dividing the conceptual data model into fragments (fragmentation)
Currently, relatively large applications with diverse needs may need to be implemented with more than one database model; a method for database selection should also take such possibility into account. In this step all the gathered requirements (data, functional and non-functional) are weighted and considered into the decision if and how to divide the conceptual data model.

This step requires two levels of prioritizing (i.e., weighting) between the different requirements. The first level is in between the different NFRs. At this stage, our method takes into account six NFRs: consistency, integrity, flexibility, volume, velocity and veracity. It is a fair assumption that when addressing two classes' non-functional similarity and fragmentation, not all NFRs have equal weights. For example, if two classes require different consistency levels (e.g., eventual or strong) they are less likely be fragmented together than in the case that two classes that require different volume (e.g., very high and very low). Therefore, when calculating the NFRs distance between two classes, each NFR receives a weight based on our perceived importance of the NFR.

The second level is between the different types of requirements. Another fair assumption is that different requirements (data, functional, non-functional) might also have different impact on the fragmentation process. For example, the fact that two classes are connected in the conceptual model does not necessarily mean that they will be fragmented together. In fact, this is probably less important than if the classes are frequently queried together or share similar non-functional characteristics. Therefore, each type of requirements will receive a different weight in the process.

The weights were chosen based on analytical hierarchy process (AHP) [17]. The pairwise comparisons, which compose the input for the AHP, were chosen based on previous knowledge and surveys on the different databases. In the future we plan to let practitioners set the weights throughout the process. However, since our method is aimed to ease this process, and decrease the need to be familiar with different technologies and their aspects, we plan to make this plan optional.

Once all the requirements were weighed into the classes' similarity, the fragmentation process is done by a clustering algorithm: DBSCAN[2]. DBSCAN receives as input a distance matrix and finds the best number of clusters and their contents.

Step 5 – Data model selection

In order to select the most suitable database model for each fragment, we defined a database profile. A profile is a numerical description for the different NFRs for the databases. In order to perform the selection, we calculate the similarity between the profiles of the fragments and the different data models, based on Manhattan distance function[3], and choose/recommend the most similar data model.

We use the NFRs for this process since they best differentiate between the different data models. For example, NoSQL databases support eventual consistency and low integrity, as opposed to Relational databases that support high consistency and integrity (as part of the ACID concept). All NFRs, apart from query complexity and volume, received a numeric value based on the scale of the NFR.

The fragment's profile is calculated as an average of each of the non-functional requirements of the classes that constitute the fragment. The selection of the appropriate database model will be based on the weighted distance among the fragments' profiles and the database models' profiles (as in the previous step, we assume that NFR are of the different importance for the selection process, hence the weighting). The chosen database model is the one with the minimal distance, i.e., with the most similar profile. The result is one or more fragments and the most suitable database models for their implementation.

Step 6: Design the fragments

The previous step resulted in one or more fragments required for the application, and the most suitable model for implementing them. However, even a most fitting model has to be designed carefully and with accordance to the different requirements. In this step each fragment is designed as an individual unit according to its most suitable database model; each model has an appropriate method for its design.

The output of this step is a set of model blocks for each model. A block is a database model unit, e.g. a document, a table, a node, etc. These blocks constitute as a type of schema for each of the fragments. We plan to suggest design for the different blocks based on existing methods as much as possible. In a literature review we performed [15] we found some possible methods, that with some adjustments, the process would result in a complete, sound and adequate design.

More specifically, we have in mind two specific methods. The first is GDBS [13], a method we developed. The method creates a schema for graph database based on an ERD. The method requires changes in order to take into account query considerations and non-functional properties. Another method is NoSE [12] that with some changes would suit well to our process. NoSE is used for designing column stores, based on an EER diagram and needed queries. The method is based on innovative concepts such as

[2] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

[3] https://xlinux.nist.gov/dads/HTML/manhattanDistance.html

query decomposition and machine learning, guarantying a sound schema (i.e. column families). As GDBS, with small modifications it will fit our design process.

5 Summary and Future Work

As many database models and supporting system have emerged during the last decade, it is important to select the most fitted ones for specific applications. In this research we plan to devise a method for selecting the most fitting database model(s) and system(s) for a set of requirements and proposing a design for the recommended models. The method contains six steps beginning with a comprehensive requirement elicitation and terminating with a design of the most fitting database models for the sought application.

Currently, the work focuses on the selection process – steps 1-5 in the suggested process. The process was demonstrated and adjusted on a rather small but extensive case study. We currently work on examining the process on a large-scale case-study, based on a real system. In addition, in future we plan to formalize the sixth step, the design step, by adopting and adjusting suggested design methods, and creating new design methods when needed. We also plan to automate the method in order to facilitate its usage and allow developers to overcome the current limitations of selecting and designing database models.

Acknowledgements

This research is advised by Dr. Arnon Sturm and Prof. Peretz Shoval.

References

1. Angles, R. (2012, April). A comparison of current graph database models. In Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on (pp. 171-177). IEEE.
2. Chebotko, A., Kashlev, A., & Lu, S. (2015). A big data modeling methodology for Apache Cassandra. In Big Data (BigData Congress), 2015 IEEE International Congress on (pp. 238-245). IEEE.
3. Gessert, F., Wingerath, W., Friedrich, S., & Ritter, N. (2017). NoSQL database systems: a survey and decision guidance. Computer Science-Research and Development, 32(3-4), 353-365.
4. Han, J., Haihong, E., Le, G., & Du, J. (2011, October). Survey on NoSQL database. In Pervasive computing and applications (ICPCA), 2011 6th international conference on (pp. 363-366). IEEE.
5. Haseeb, A., & Pattun, G. (2017). A review on NoSQL: Applications and challenges. International Journal of Advanced Research in Computer Science, 8(1).

6. Hevner, A., & Chatterjee, S. (2010). Design science research in information systems. In Design research in information systems (pp. 9-22). Springer, Boston, MA.
7. Jouili, S., & Vansteenberghe, V. (2013, September). An empirical comparison of graph databases. In Social Computing (SocialCom), 2013 International Conference on (pp. 708-715). IEEE.
8. Khazaei, H., Fokaefs, M., Zareian, S., Beigi-Mohammadi, N., Ramprasad, B., Shtern, M., ... & Litoiu, M. (2016). How do I choose the right nosql solution? a comprehensive theoretical and experimental survey. *Big Data and Information Analytics (BDIA)*, *2*, 1.
9. Kolomičenko, V., Svoboda, M., & Mlýnková, I. H. (2013, December). Experimental comparison of graph databases. In Proceedings of International Conference on Information Integration and Web-based Applications & Services (p. 115). ACM.
10. Kumar, R., Parashar, B. B., Gupta, S., Sharma, Y., & Gupta, N. (2014). Apache Hadoop, NoSQL and NewSQL solutions of big data. International Journal of Advance Foundation and Research in Science & Engineering (IJAFRSE), 1(6), 28-36.
11. Lourenço, J. R., Cabral, B., Carreiro, P., Vieira, M., & Bernardino, J. (2015). Choosing the right NoSQL database for the job: a quality attribute evaluation. Journal of Big Data, 2(1), 18.
12. Mior, M. J., Salem, K., Aboulnaga, A., & Liu, R. (2017). NoSE: Schema design for NoSQL applications. IEEE Transactions on Knowledge and Data Engineering, 29(10), 2275-2289.
13. Roy-Hubara, N., Rokach, L., Shapira, B., & Shoval, P. (2017). Modeling graph database schema. *IT Professional*, *19*(6), 34-43.
14. Roy-Hubara, N., & Sturm, A. (2018). Exploring the Design Needs for the New Database Era. In *Enterprise, Business-Process and Information Systems Modeling* (pp. 276-290). Springer, Cham.
15. Roy-Hubara, N., & Sturm, A. (2019). Design Methods for the New Database Era: A Systematic Literature Review. Submitted to SOSYM, 2019
16. Roy-Hubara, N., Sturm, A., & Shoval, P. (2019). A Method for Database Model Selection. In *Enterprise, Business-Process and Information Systems Modeling*. Springer, Cham.
17. Saaty, T. L. (2008). Decision making with the analytic hierarchy process. International journal of services sciences, 1(1), 83-98.
18. Storey, V. C., & Song, I. Y. (2017). Big data technologies and management: What conceptual modeling can do. Data & Knowledge Engineering, 108, 50-67.
19. Tang, E., & Fan, Y. (2016, November). Performance comparison between five NoSQL databases. In Cloud Computing and Big Data (CCBD), 2016 7th International Conference on (pp. 105-109). IEEE.
20. Tudorica, B. G., & Bucur, C. (2011, June). A comparison between several NoSQL databases with comments and notes. In Roedunet International Conference (RoEduNet), 2011 10th (pp. 1-5). IEEE.]