Requirements analysis for data model selection

Hitasha Ghai
Dept. of Electrical and Computer Engineering
University of Victoria
Victoria, BC
hitashaghai@uvic.ca

***Abstract* — Designing a database is an important task within an organization and choosing the correct data model is trivial for future performance. There are different types of data models available and this report analyses and compares three of them considered important and contrasting each other.**

Keywords—Relational database, Graph database, Key-Value database

I. INTRODUCTION

The first step in the data design process is to decide which data model to use. The available models can be divided into two major groups: SQL and NoSQL, as described in Section II. The report aims to describe and compare three data models (one SQL and two NoSQL) based on the following questions:

(i) How complex are the queries to be performed on the data?
(ii) Is the consistency necessary for the requirements or could the eventual consistency be sufficient?
(iii) How much storage capacity is available?
(iv) What is the maximum latency acceptable?
(v) Should the model be flexible?
(vi) Is scalability a requirement?

The report is organized as follows: Section II gives a small introduction to SQL and NoSQL databases; Section III describes the three data models used for comparison; Section IV explains the criteria for a good data model selection.

II. SQL AND NOSQL

SQL databases follow property of ACID [9]:

a) Atomicity: each transaction is considered as a single 'unit' that succeeds or fails completely
b) Consistency: a transition brings the system from one consistent state to another
c) Isolation: it is possible to execute the transitions in parallel as they were executed sequentially
d) Durability: the effects of a transition are permanent

On the other hand, NoSQL databases are non-relational, open source, and horizontally scalable.

The main benefits gained by the application of a NoSQL model can be synthesized in 5 key points [6]:

1. Easy design development without a normalization process
2. Possibility of handling different forms of data
3. Management of huge amounts of data
4. Traffic scaling
5. Flexibility toward new requirements

There are 4 NoSQL models [10]:

(i) Graph: entities represented by nodes and relationships as links or edges between nodes
(ii) Column-family: columns of a table are divided into families representing related data
(iii) Document: the database is divided into documents with their collection of keys and values
(iv) Key-value: it stores data in values referenced by keys

III. MODELS

This section reports a short definition with an example for each of the three models chosen for the analysis. All examples are constructed from the Entity-Relationship diagram in fig.1.

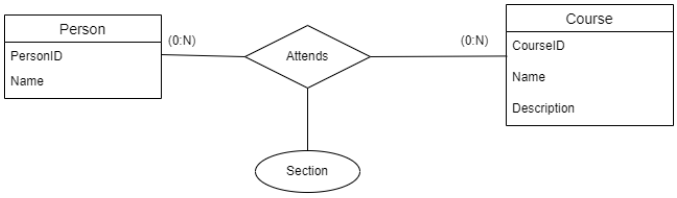

Fig.1 ERD of Person attending multiple courses

A. Relational database

A relational database, as Oracle defines [1], is a data model consisting of tables and relations between them. Each table has a special column, named "primary key", which uniquely determines all the other columns (attributes) in the same table.

The relational model obtained from fig.1 is the following:

Person (PersonID, Name)

Course (CourseID, Name, Description)

Attendance (*CourseID, PersonID*, Section)

The third table, "Attendance", has been added since in the original model we had a many-to-many relationship which at the logical level is transformed with an extra table containing the references to the two primary keys. This represents a foreign key relationship [7]

B. Graph Database

Following Diestel's definition [9], a graph G is a pair G =(V, E) of sets such that:

- V is the set of vertices
- $E \subseteq [V]^2$
- $V \cap E = \emptyset$

The graph database is a type of NoSQL model in which the data is not stored in tables, but in nodes that represent the entity of the graphs [2].

The example is shown in fig.2.

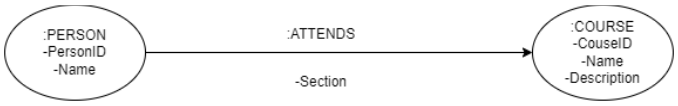

Fig. 2 LPG for Person attending Courses

C. Key-Value database

In the key-value database, also known as the key-value store, data is stored in values referenced by keys. The keys and values could assume any type (from String to complex objects) [3].

Broadly, we can affirm that a key-value store can be considered as a relational table but with only two columns: the first column represents the primary key, while the second column contains all other attributes [6].

The key-value store associated with the ERD in fig.1 is reported in fig.3.

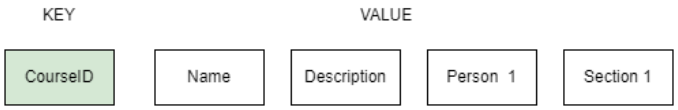

Fig. 3 KV for People attending courses.

As shown in fig.3, it has been decided to use the CourseID as the key and save the rest in the value, together with the people attending that course and the corresponding section.

This case is an extreme example of denormalization which could have been avoided or adapted in reality according to the system requirements.

IV. MODEL COMPARISON CRITERIA

The three presented models serve different functions according to the system requirements. The relational database is a normalised model, while graphs and key values are mostly de-normalised. The choice among the two types depends on the system requirements and the operations to be performed.

If there will be constant changes in the data, e.g., insertions, updates, and deletes, then having a normalised database is the most suitable optional, since we can avoid insertion, update, and deletion anomaly. In fact, the data is not duplicated, therefore we need to make only one change. This is the case with OLTP databases.

On the other hand, if we will use the data only for reading, e.g., for data analytics purposes, then a denormalized database is the fastest in terms of queryability. This is the case, for example of OLAP databases.

Table I Parameters comparison between models [15]

	RELATIONAL	**GRAPH**	**KEY-VALUE**
Data model complexity	Medium	High	Low
The breadth of data model applicability	Low	High	Medium
Ease of schema change	Low	Very high	Very high
Performance	Medium	High variable (depending on query)	Very high
Scale-out cost	High	Depends on architecture	Low
TCO for very large operational volumes	High	Depends on architecture	Low

A. Consistency

According to Brewer's CAP theorem [14], a distributed database can only have two of the following properties:

a) *Consistency*: all read operations return the same information.
b) *Availability*: all data is accessible but not updated.
c) *Partition tolerance*: the system is working even in case of network failures.

Since we are talking about distributed systems, it is not possible to compromise on partition tolerance. Therefore, we are left only with the option of choosing between Availability or Consistency. In the case of choosing Consistency, we might get errors whenever we try to access data that has not been updated. Hence, the system is not available. On the other side, if we allow the system to be always available, we are at some point not giving the right information and, hence, ignoring consistency.
In general, all NoSQL databases tend to prioritize availability over consistency [8].
However, they try to achieve what we call eventual consistency.

B. Flexibility

In a relational model, all data needs to fit in the diagram [4]. For example, the creation of a relationship that is the same as

existing but with different entities will require the creation of an extra table. Moreover, relational models are thought for structured data which needs to be inserted/updated in a predefined schema for easy access [5].

The graph model is schema-less: each node has its properties which can vary over time. In addition, a node can have an indefinite number of relationships that do not require to be determined priorly. This second model also offers the possibility of reusing nodes without the necessity of creating duplicate nodes for equivalent, but separate relationships.

Overall, NoSQL databases are more flexible than relational databases, normally used for storing semistructured and unstructured data.

C. Scalability

NoSQL databases were developed in the Internet era when SQL models were revealed to be incapable of handling the huge amount of data required by web pages [6].

Relational databases are not suitable for horizontal scaling, because this will mean having different tables on different servers and the cost of joins between servers is relatively huge. It is not possible to conduct horizontal scaling or scaling in the relational databases, e.g., adding more machines for better load balancing. This is because a relational data model guarantees consistency, which could be loosed when scaling the database across multiple machines.

D. Accessibility

The two non-relational data models presented in this report have some differences as well. First of all, key-value databases are fast but not as flexible as graph databases, which are not that fast.

Key-value stores are the simplest non-relational model thanks to the two-column structure. In this way, it is possible to access the data stored in values quickly through the key related to it.
Graph databases are mostly used for representing relationships clearly. The queryability, in this case, is excellent, due to data de-normalization, but requires an increased complexity in terms of speed.

V. Examples of DBMS

A. MySQL

As DuBois describes in his book [11], MySQL is a relational database management system based on a SQL client/server model. It consists of an SQL server, clients for accessing the server, administrative tools, and a programming interface.
The main advantages of MySQL, cited in the guide by Suhering [12], are available on different operating systems, stability, and well-written documentation available on the official website.
It is possible to use this DBMS directly from the command line or in the user-friendly view of the workbench.

B. Neo4J

Neo4J is an open-source NoSQL [2] that implements a graph model at the storage level. The key points of this model are:

(i) Cypher, query language similar to SQL
(ii) Constant time traversals
(iii) Flexibility
(iv) Drivers for common programming languages (Java, JavaScript, .NET, Python, etc.)

C. Amazon DynamoDB

DynamoDB is a NoSQL cloud database service provided by Amazon. This service is not only available to the customers but it is applied by Amazon for some of its services, e.g., Alexa.
The key characteristics of DynamoDB are those reported by Usenix [13]:

(i) The service is fully cloud-based.
(ii) Data from different customers is saved on the same physical machine to maximize space utilization and minimize costs for customers.
(iii) Unlimited amount of data stored in tables.
(iv) Responses with low latency.
(v) High availability.
(vi) Flexible use cases.

VI. Conclusion

In conclusion, the choice of one model or the other depends on the specific use case.
Even though NoSQL databases have a completely different structure from the tables used in the relational model, they present some huge differences among them.
The relational model is mostly used when data needs to be constantly updated, for example, in bank service.
Graph databases are commonly used for fraud detection, social networks, or knowledge graphs.
While KV is a good option if we require something more similar to a relational model but schemaless.

References

[1] Oracle, “What is Relational Database (RDBMS)?” – https://www.oracle.com/ca-en/database/what-is-a-relational-database/
[2] Neo4J Documentation, “What is a Graph Database?” – https://neo4j.com/developer/graph-database/
[3] Amazon, “What is a key-value database?” - https://aws.amazon.com/nosql/key-value/
[4] G. Jordan, “Practical Neo4J”
[5] M. Drake, “A Comparison of NoSQL Database Management Systems and Models” – https://www.digitalocean.com/community/tutorials/a-comparison-of-nosql-database-management-systems-and-models
[6] MongoDB, “Understanding the Different Types of NoSQL Database” – https://www.mongodb.com/scale/types-of-nosql-databases
[7] MySQL Documentation, “13.1 17.5 FOREIGN KEY Constraints” – https://dev.mysql.com/doc/refman/5.6/en/create-table-foreign-keys.html
[8] M. Drake, “A Comparison of NoSQL Database Management Systems and Models” –

https://www.digitalocean.com/community/tutorials/a-comparison-of-nosql-database-management-systems-and-models

[9] V. Sharma, M. Dave, “SQL and NoSQL Databases”

[10] W. Kriha, C. Strauch, “Ultra-Large Scale Sites: NoSQL Databases (Lecture notes)”

[11] P. DuBois, “MySQL”

[12] S. Suehring, “MySQL Bible”

[13] A. Elhemali, N.Gallagher, N. Gordon, J. Idziorek, R. Krog, C. Lazier, E. Mo, A. Mritunjai, S. Perianayagam, T. Rath, S. Sivasubramanian, J. C. Sorenson III, S. Sosothikul, D. Terry, A. Vig, Amazon Web Services, “Amazon DynamoDB: A Scalable, Predictably Performant, and Fully Managed NoSQL Database Service”.

[14] S. Salomé, “Brewer’s CAP Theorem”

[15] A. Fowler, E.Ciurana, “NoSQL & Data Scalability”