Research article Available online www.ijsrr.org ISSN: 2279–0543

International Journal of Scientific Research and Reviews

Fusion Architecture of Database for Large and Diverse Dataset

Sameera Khan

Department of Computer Science, Amity University Chhattisgarh, Raipur, skhan@rpr.amity.edu

ABSTRACT:

In past decades the data model used for a standard storage system is a relational data model. Predominantly it uses table and record structure to store data. Recent advancements in the nature and behavior of data in terms of volume, velocity and variety have led to the necessity of a new database model. Non-relational data models are one solution to it but at the cost of basic ACID properties. This large dataset consists of structured, semi-structured and unstructured data. All of them have to be treated differently. Both relational and non-relational data models have their respective advantages and disadvantages in data handling. This paper proposes a fusion architecture of database which combines the advantages of both and at the same time minimizing the disadvantages. The proposed architecture will be capable of handling diverse and voluminous data. Proposed approach intends to bridge a gap between relational and non-relational approaches and to provide an efficient data handling method for large and diverse datasets.

KEYWORDS: ACID properties, large datasets, NoSQL, relational model

***Corresponding author:**

Sameera Khan

Department of Computer Science,

Amity University Chhattisgarh,

Raipur,

Email: skhan@rpr.amity.edu

INTRODUCTION

Data engines based on relational data models displayed high performance while using small relational dataset. Such databases are able to handle limited volume and variety of data. Large volume and diverse variety of data cannot be properly handled using relational databases. Due to the ease of availability of handheld devices, introduction of social media, emphasis of using the Internet of things (IoT), Internet of Vehicles (IOV), increase in the users of portable devices, there is a sudden increase in data generation. The rate of generation of data is increasing exponentially. Such a large volume of data is not homogenous in nature. Since it is collected by using diverse data sources, they differ in their formats and forms. Some are structured data, some are semi-structured whereas others are unstructured data. Velocity and volume of data is the primary concern while handling large dataset.

In recent years spatial, temporal, spatiotemporal databases have been introduced to deal with large and diverse datasets. [1]With the increase in the use of social media, Facebook and Twitter are alone a big contributor as a data source. [2]Since the volume of data generated is increasing exponentially, data handling methods also need to be updated regularly. New approaches are required which are capable enough to deal with 6v's of big data- Volume, Velocity, Variety, Veracity, Variability and Value.[3]Data received is not just text now. It is in varied formats like voice, image, document, e-mails, objects and many others. Each of them has to be treated in a different manner.

Relational models are widely used model since the past decades. But with the constant change in the technological environment and type of data generated, somehow use of relational databases is not enough. As a result of this Non-relational database models are becoming popular for handling large datasets. Such database models sacrifice various basic requirements and also slower as compared to relational models. In this paper a hybrid approach for the data model is proposed in which proper amalgamation of both the data models will be done. The proposed system will exploit the advantages of both the data models and lessen the disadvantages by selecting the apt database each time before storing the data.

RELATIONAL DATABASE

Relational databases are the most commonly used approach for data storage. It uses tables for data storing in which columns represent attributes and rows represent records. This data model is largely in use for many decades. Almost all open source databases use this model. E.F. Codd has given twelve basic rules for describing the structure of a relational database.[4]Following are the highlights of Codd's rule-

- Data must be stored in a table where rows are records and columns are attributes.
- Every record must have a unique identifier known as a primary key which can be a single attribute or

group of attributes for uniquely identifying each record. To access content, table name, attribute name and primary key must be known.

- Null values must be treated systematically. It can be a case of missing value, unknown data or data not applicable. All three cases must be treated separately.
- The active online catalog must be supported by the database in the form of a data dictionary.
- The database must support data sublanguages for insertion, deletion and manipulation of data.
- After each operation or on user request database must be able to update views.
- All the basic operations like insertion, deletion and manipulation must be supported by the database. In addition to this it must also be able to support set operations like union, intersection, set difference etc.
- Any change in table structure must not affect user view i.e. it must support physical data independence.
- Any physical change must not affect the database structure i.e. it must support logical data independence.
- All the integrity constraints must be applied in the database and any change in these constraints must not affect the database.

Table 1- Tabular representation of data in relational database

Employee_ID	Employee_name	Department
1001	Sameera	Engineering
1002	Sana	Humanities
1003	Shahwar	Humanities

Above Table represents tabular representation of Relational database with three attributes Employee_ID, Employee_name and Department where Employee_ID is the primary key. Rows signify records. In the above representation shows three records. Relational Databases always follow Codd's rule. They also display ACID properties which are an acronym of Atomicity, Consistency, Isolation and Durability.

NON RELATIONAL DATABASES

Non-relational databases are usually object-oriented databases which do not ensure ACID properties. They are designed to handle diverse variety and a large volume of data. Non-relational databases support BASE properties. The BASE is an acronym of Basically Available, Soft state, Eventual consistency. Non-relational databases are basically of four types[5,6]Each of which are discussed briefly below-

- Key-value database- the data is stored in a key-value pair where the key represents a unique identifier and the value represents data. Value field can be a pointer, an object, an image or any other data format. Riak, Dynamo, Oracle NoSQL database are some of the examples of the key-value database.
- Column family database-It uses the concept of keyspace in which column families are stored. Each column family is equivalent to a table in the relational model. It has multiple rows and each row contains multiple data items. It is not mandatory to have the same data in each row. Usually a timestamp is attached with each data. Such databases are easily scalable and compressible. Cassandra, BigTable, HBase are some of the examples of this family.
- Graph family database- they are used to represent highly connected data and relationships. Usually they are schemaless in order to provide flexibility. It uses properties on the NoSQL structure while keeping the relational model alive. Neo4J is an example.
- Document database- Data is stored in the form of the key-value method but the significant difference lies in the method of representing values. Values are represented in the form of documents. These documents are encoded with XML, JSON or BSON encodings which provides structure to the document. MongoDB and CouchDB are common examples.

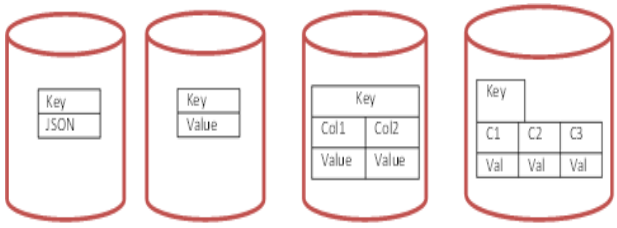

Figure 1 Architecture of various non-relational databases. Document-based, Key-value, Column family and graph family structures

Biggest advantages of non-relational databases are scalability and flexibility. Data can be stored in the system with non-uniform formats and varying length in non-relational databases unlike relational databases. They can be scaled up as and when required.

Complex transactions in such databases are difficult. Multiple document transaction is not supported by the non-relational database. Eliminating the implementation of ACID properties makes it difficult to perform a multi-document transaction. To attain the objective of scalability, the need for ACID properties was sacrificed.

COMPARISON BETWEEN RELATIONAL AND NON-RELATIONAL DATABASES

Relational and non-relational approaches in database architecture are entirely different from each other.[7] This section will compare both the approaches on the basis of several basic criteria like scalability, flexibility, technique, security issues, complexity, query languages used etc.

- Scalability- Relational databases support vertical scalability, that means if the volume of data increases, it can be accommodated by increasing the storage capacity and the processing power on the other hand non-relational databases support horizontal scalability. Horizontal scalability deals with an increase in data by adding more nodes and server to the network. Horizontal scalability is cheaper than vertical scalability.
- Performance- Relational databases support ACID properties whereas non-relational databases support BASE properties. It is almost impossible for a non-relational database to support ACID properties.
- Complexity- Since relational databases follow a predefined schema, they are less complex but due to the varying structure of the non-relational databases they are more complex in nature. More specifically graph-based databases are more complex in nature than other relational databases.
- Flexibility- With the introduction of dynamic data and diverse data sources it is always expected from a database to accommodate all the data which is not always possible. Since relational databases have fixed schema it is very difficult and expensive to change the schema frequently whereas non-relational databases are quite flexible in nature due to the use of dynamic schema. Non-relational databases can save a wide variety of data with varying data formats and size quite easily. Relational databases are capable of handling only structured data whereas non-relational databases can handle structured, semi-structured and unstructured data.

Table 2- Comparison of relational and non-relational databases

Data Model	Based on the Technique	Performance	Scalability	Complexity	Flexibility	Query Language	Security
Relational Database	Relational Algebra	Varies with content	Varies with content	Low	Low	SQL	High
Key-value database	Varies with content	High	High	Low	High	No Standardized language, Varies with system	Average
Column family database	Varies with content	High	High	Low	Average		Average
Graph family database	Graph Theory	Varies with content	High	High	High		Low
Document database	Varies with content	High	High	Average	High		Varies with content

- Query language- Relational database supports Structured Query Language(SQL) which is a powerful query language that can handle complex queries and provide advanced features like joins, aggregation and set operations whereas non-relational databases do not have any standard query tool. Depending on the application which is using non-relational databases query processing technique differs. Due to ever-changing query processing it becomes difficult to analyze a system initially.[8] A standard query processing system must be designed for non-relational databases as well.
- Security- Relational databases provide security features for the databases to ensure the security of services. Partitioning and distributivity are the significant characteristics of non-relational databases, it makes difficult to provide security features to the database. [9]Different levels of security mechanisms are implemented in different types of non-relational databases like authentication, authorization, role-based access control, encryption and auditing. There is no standard protocol for all types of non-relational databases to implement the solution of all the security issues. Every database deals with security issues in its own way.

RELATED WORK

With the change in the face of data, database technologies are also emerging with new challenges. Various approaches are being proposed and implemented in traditional relational and non-relational databases to improve the performance of the database and to increase the throughput as well as to decrease the computation overhead. One such approach was proposed in [10]by A.E.Lofty et al. They proposed a middle layer solution to be implemented between the web application and the database. This middle layer consisted of the load balancer, server detector, scalability manager, Transaction manager and data migrator. The middle layer solution offers to provide ACID properties in the non-relational databases. Another approach was presented by B.E James and P.O. Asagba in [11]. They implemented a hybrid database system with one SQL database and MongoDB as a non-relational database. This approach was for big data management. This approach combines the advantages of both relational and non-relational databases and reduces the disadvantages of both. Grolinger, Wilson, Tiwari and capretz proposed a methodology for using NoSQL databases for cloud environments in[12]. They proposed a combined approach of NoSQL and NewSQL together for cloud handling.

PROPOSED METHODOLOGY

Non- relational databases are useful in handling voluminous data but they fail to retain most of the significant properties of an ideal database which are always present in the relational database. A mid-way solution is proposed for the same so that a database can have advantages of both relational and

non-relational database while reducing the disadvantages of both. Block diagram of the proposed system is given in fig-2.

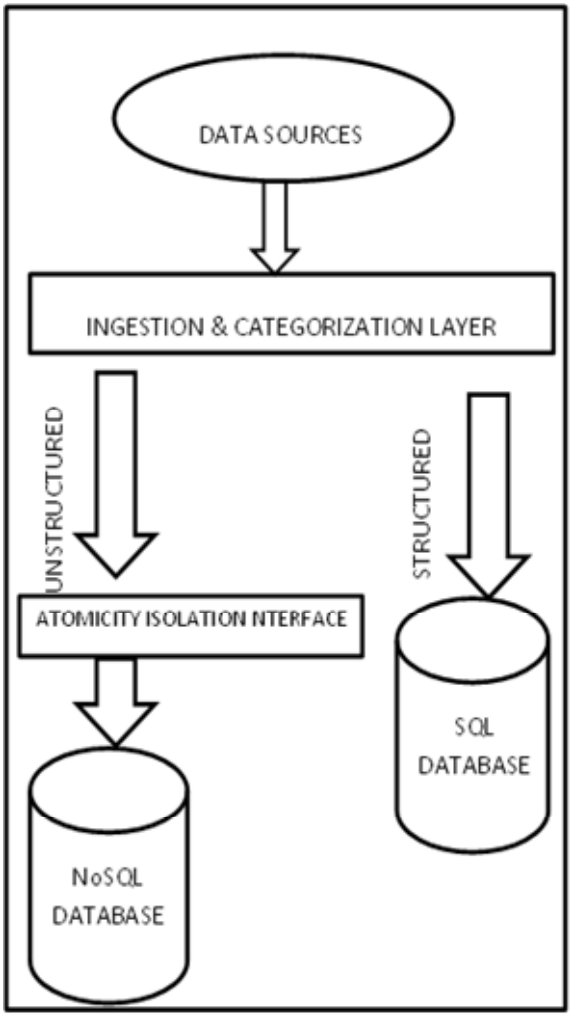

Figure 2 Block Diagram of the proposed system

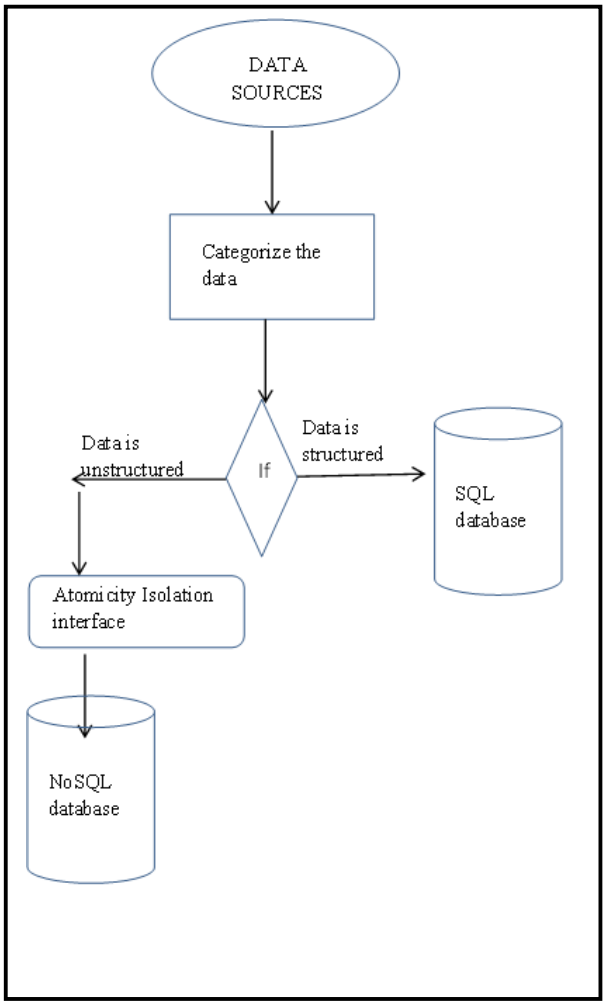

Figure 3 Flowchart of the proposed system

The proposed system has four basic blocks- Ingestion and categorization layer, SQL database, NoSQL database and an Atomicity-Isolation interface. All the data sources are connected to the data ingestion and categorization layer where data will be taken from the data sources and then it will be categorized as structured or unstructured data. When structured data is received it must be treated separately so as to save NoSQL processing overhead for already structured data. To implement this, ingestion and categorization layer is induced between data sources and the databases. This layer takes the input from diverse data sources and then categorization is performed. If structured data is received it is directly sent to the SQL database connected with this layer otherwise it will be sent to Atomicity-Isolation interface. This interface will help in implementing two of the ACID properties for NoSQL database.

This interface can be switched on and off as and when required so as to save extra processing overhead. The interface will use a locking protocol and queuing method for achieving the goal. If some transaction request on unstructured data arises the atomicity isolation interface will be activated. All the data elements involved in the transaction are locked beforehand. If any of the data is previously locked with other transaction, the current transaction will be added to the wait queue for the particular data element. The data elements are locked by their key elements only. As soon as the transaction starts the value of flag exec is set to zero. After the completion of the transaction this flag is set to 1. If the value is one then only the changes in the database will be durable. The flowchart for the proposed system is given in fig-3.

Atomicity isolation interface will be facilitated with a mechanism for scalability and load balancing to make the architecture useful for big data. Since big-data is characterized by large volume and high velocity, some mechanism to manage speed lag is needed. This will be incorporated by using scalability and load balancing mechanism. It will be integrated into the atomicity isolation interface. To reduce the extra computation time taken by this layer it is conditionally switched on as and when required. The algorithm proposed for this architecture is given below. The algorithm comprises of two parts first is the basic algorithm and the second is the atomicity isolation algorithm.

Basic Algorithm-

1. Ingest the data from various data source
2. Categorize the data as structured or unstructured data
3. If
   Data is structured
   Then store in SQL database
   Else
   Call Atomicity-Isolation algorithm
4. Update the database
5. Exit.

Atomicity-Isolation algorithm

1. Take input data
2. Identify the key values involved in the transaction
   Set exec=0
3. Lock all the key values involved in the transaction until the transaction completes
   If (any data is already locked by some other transaction)
   Add the current transaction in wait queue for the key value
   Else
   Grant lock immediately
   Perform transaction
   Set exec=1
4. If exec=1
   Commit the changes
   Else rollback
5. Unlock all the key values in the reverse order of acquiring them.

CONCLUSION AND FUTURE WORK

Many applications require a system to store diverse datasets with added ACID properties which are not possible if the relational or non-relational database will be used alone. The proposed system is a solution to implement ACID properties in a non-relational database. Structured data will be stored directly in the SQL server whereas unstructured data is processed via atomicity isolation interface and are stored in the NoSQL database. Due to the induced interfaces and layers, change in efficiency of the database is expected. Since the database designed is meant to handle voluminous data with high velocity, a remarkable decrease in efficiency will not serve the purpose. That is why data is categorized first so as to reduce the processing time of structured data in the non-relational database. As there are four types of non-relational database namely- key value, column family, document-based and graph family, each type of database can be used to study the change in efficiency of the architecture.

REFERENCES

1. .Demiryurek U., Banaei-Kashani F., Shahabi C., Transdec:A Spatiotemporal Query Processing Framework For Transportation Systems, IEEE 26th International Conference On Data Engineering (ICDE 2010) 2010;1: 1197-1200
2. Sharma S, Tim U S, Wong J, Gadia S, Sharma S,. A Brief Review on Leading Big Data Models; Data Science Journal; 2014; 13: 138-157.
3. Khalid M Y, Hui P H, Raman V, Exploratory Study For Data Visualization In Internet Of Things; 42nd IEEE International Conference On Computer Software & Applications :2018; 517-521
4. Codd, E. F..A Relational Model Of Data For Large Shared Data Banks; M.D. Computing : Computers In Medical Practice; 1970; 153(6): 377-387
5. Nayak, A., Poriya A, Poojary D. Type Of NOSQL Databases And Its Comparison With Relational Databases; 2013; 5(4): 16-19
6. Tudorica B G, Bucur C. A comparison between several NoSQL databases with comments and notes, 2011 RoEduNet International Conference 10th Edition: Networking in Education and Research 2011;1-5.
7. Sahatqija, K., Ajdari, J., Zenuni, X., Raufi, B., & Ismaili, F. Comparison Between Relational And NOSQL Databases. 2018 41st International Convention On Information And Communication Technology, Electronics And Microelectronics (MIPRO) 2018; 0216-0221.
8. Dindoliwala V J, Morena R D, Survey On Security Mechanisms In Nosql Databases, International Journal Of Advanced Research In Computer Science; 2017; 8 (5): 333-338

9. Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E., & Abramov, J. Security Issues In Nosql Databases. 2011IEEE 10th International Conference On Trust, Security And Privacy In Computing And Communications, 2011; 541-547
10. Lotfy A E.,Saleh A I, El-Ghareeb H A and Ali H A. A middle layer solution to support ACID properties for NoSQL databases 2016; 28: 133-145
11. James B E, Asagba P O, Hybrid Database System For Big Data Storage And Management, International Journal Of Computer Science Engineering And Applications (IJCSEA); 2017;7(3):15-27
12. Katarina G, Higashino W A, Tiwari A and Capretz M A M. Data management in cloud environments: NoSQL and NewSQL data stores. Journal of Cloud Computing: Advances, Systems and Applications ; 2013; 2: 1-24.
13. Bathla G, Rani R, and Aggarwal H. Comparative Study of NoSQL Databases for Big Data Storage,International Journal of Engineering & Technology 2018; 7: 83-87