IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, June 2022
Impact Factor: 6.252

Migration of Relational Database to Document Oriented Database

Anna Benchy[1] and Gibin George[2]
Student, Department of Computer Science, Santhigiri College of Computer Sciences, Vazhithala[1]
Assistant Professor, Department of Computer Science, Santhigiri College of Computer Sciences, Vazhithala[2]

Abstract: *Relational databases that came into existence a long time ago have many advantages and disadvantages. To get rid of this limitation in relational databases NoSQL databases were introduced that mainly focus on high operations speed and flexibility in storing the data. To store large volumes of semi-structured data over distributed nodes without a predefined schema and query and aggregate the data very quickly, a document-oriented database can be used which is a type of NoSQL database. Some organizations that are currently dealing with relational databases would sometimes wish to migrate from relational to document-oriented databases. Hence there is a need for integrating the relational databases into NoSQL databases.*

Keywords: Relational Database, Document-Oriented Database, NoSQL, Migration, Relationship, Linking, Schema, Data, etc.

I. INTRODUCTION

Relational databases came into existence a long time ago. It stores the normalized data in interrelated tables and ensures data integrity. In a relational database, the data is structured and scaling is difficult. Hence, they have many advantages and disadvantages as well. To get rid of the limitations of relational databases, NoSQL databases were introduced that mainly focus on two things: high operations speed and flexibility in storing the data. Some organizations that are currently dealing with the relational database would sometimes wish to migrate from relational to NoSQL databases. Hence there is a need for integrating the relational databases into NoSQL databases. In my paper, I would like to discuss the integration of relational databases into a document-oriented database which is a type of NoSQL database.

II. RELATIONAL DATABASE

A relational database uses tables to store data. Tables are used to store the information about the objects that are in the database. Multiple tables are linked together by a common field in a relational database. A table is called a relation that represents real-world objects such as a person, place, or thing. A table consists of rows and columns. Each horizontal entity in a table(row) is termed as a record that has a unique id known as a primary key. The vertical entity or the columns of the table store the attributes of the data. A Relational Database Management System or RDBMS is a software program to store, create, and manage the data in a relational database.

An RDBMS acts as an interface for the users to interact with the database to manage the storage of data and access to the data. SQL or Structured Query Language can be used to access the database in the relational database management system. An RDBMS helps to maintain data integrity and data consistency. The primary key in a table can act as the foreign key in another table using which multiple tables can be linked. Hence there won't be the need for repetition of data which can reduce data redundancy. In order to maintain consistency before and after a transaction relational database follows ACID properties. ACID stands for Atomicity-Consistency-Isolation-Durability. Atomicity ensures that either a transaction occurs completely or it doesn't take place at all, i.e., a transaction won't stop in the midway.Consistency ensures that the information in the database is consistent before and after every single transaction.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, June 2022
Impact Factor: 6.252

Isolation ensures that multiple transactions occur independently without being affected by each other. Durability ensures that changes of a successful transaction will be reflected in the database even if the system fails.

III. WHY NOSQL DATABASES?

The relational data model is inflexible. It has a well-defined schema that will restrict the structure of the data. We have to define a fixed structure like the tables columns and rows and define their relationships which is hard to change in a relational database. Also, if the domain is deeply nested using a relational model to traverse it will require a lot of tables and joins. Relational databases have poor horizontal scalability. they can be scaled vertically by adding more resources like CPU and ram. But they cannot be scaled horizontally i.e., connect multiple machines and form a cluster.

This is due to the consistency requirements. A document-oriented database solves some of these issues faced by the relational database. It is schema-less and doesn't restrict the data to have a structure i.e., it can store structured unstructured or semi-structured data. It is flexible since the data is not structured. They can store data in a simple and straightforward form. It can be scaled horizontally. That means the database can be partitioned across several servers. Hence when we need to store unstructured or semi-structured big data across multiple servers NoSQL databases are a perfect choice. Hence NoSQL databases excel in terms of scalability, ease of use, and availability than relational databases. Hence relational databases are being replaced with NoSQL databases.

IV. DOCUMENT ORIENTED DATABASE

NoSQL databases are used to label the non-relational and distributed data stores. It usually stores the unstructured or semi-structured data as JSON documents or key-value pairs. JSON is a native language that is used to store and query data. NoSQL data stores are favored by high-volume services that require sub-second response time. The main types of NoSQL databases are document-oriented, key-value, wide-column, and graph.

A document-oriented database also known as a document store, is a computer program and data storage system designed for storing, retrieving, and managing document-oriented information. The document-oriented information is also known as semi-structured data. The central concept of a document-oriented database is that of a “document”. In the database, the documents can be addressed using a unique key that represents that document.

These documents are grouped into collections to form database systems. Relationships among the documents are represented through nested data in document-based databases. Documents encapsulate and encode data in some standard formats or encodings. The usage of an API or query language to retrieve documents based on their contents is an important defining characteristic of a document-oriented database. Document databases are designed for flexibility so they aren’t forced to have a schema. Therefore, it is easy to modify the document-based database since there are no restrictions on the format and structure of data storage. Document databases are a good option when an application requires the ability to store varying attributes along with large amounts of data. MongoDB and Apache CouchDB are examples of popular document-based databases.

V. WHY INTEGRATE A RELATIONAL DATABSE INTO A DOCUMENT ORIENTED DATABASE?

Relational Databases came into existence 30 years ago and since then they were the foundation of Enterprise Data Management. It has been the leading data storage technology for years but the changes in the demands for data processing have caused the emergence of new data storage, retrieval, and processing mechanisms. Document-based databases are one such newly emerged mechanism. Nowadays processing data, building applications, and analyzing results are becoming much more complex. The data will be incoming from a variety of sources, and organizations so it will be difficult to manage the increasingly complex user loads with traditional Relational Databases.

To match the highly flexible and powerful languages used for software development new world engineering applications demand the persistence of complex and dynamic forms of data which can be met using the Document-oriented databases. The Cloud computing solutions help in the storage of data efficiently,

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, June 2022
Impact Factor: 6.252

computingthe massive amount of data, and providing high scalability, to ensure high performance and availability at low costs. Hence operating expenses can be reduced with cloud computing solutions. NoSQL databases are more appropriate for cloud computing than relational databases. These are some of the reasons why it becomes necessarily important to migrate from a Relational Database to Document oriented database. Document databases use practical, intuitive modeling that reads the data faster than relational models and a much more flexible structure that can handle large amounts of unstructured data. There are many software's that are already using relational databases. Hence there is a need for integrating the relational database into Document-oriented databases.

VI. MIGRATING FROM A RELATIONAL DATABSE TO DOCUMENT ORIENTED DATABSE

Data migration can be defined as a "Tool-supported one-time process which aims at migrating formatted data from a source structure to a target data structure whereas both structures differ on a conceptual and/or physical level". Data migration has two important steps. The first is to restructure the source data according to the requirements of the target system, and the second is to transfer data from the source to the target database. There are several methods for dealing with these steps: schema conversion, meta-modeling approach, ETL (Extract, Transform, Load), program conversion, model-driven migration, and automated data migration.

Different methods can be used to migrate data from relational databases to document-oriented databases. Sometimes developers create their migration scripts which will transform the relational data in the source into a hierarchical JSON structure which can then be imported from relational databases to the document-oriented database. An extract transform load (ETL) tool can also be used to migrate the data. Some of the ETL vendors include Informatica, Pentaho, and Talend. They extract data from the source database, transform it into the target schema, and then load it into the target database. Another type of approach creates XML documents without redundancy and preserves all constraints. In this approach at first XML schema is generated from the source i.e., relational database, and then data is translated. Here the main emphasis is to preserve all RDB constraints and avoid redundancies.

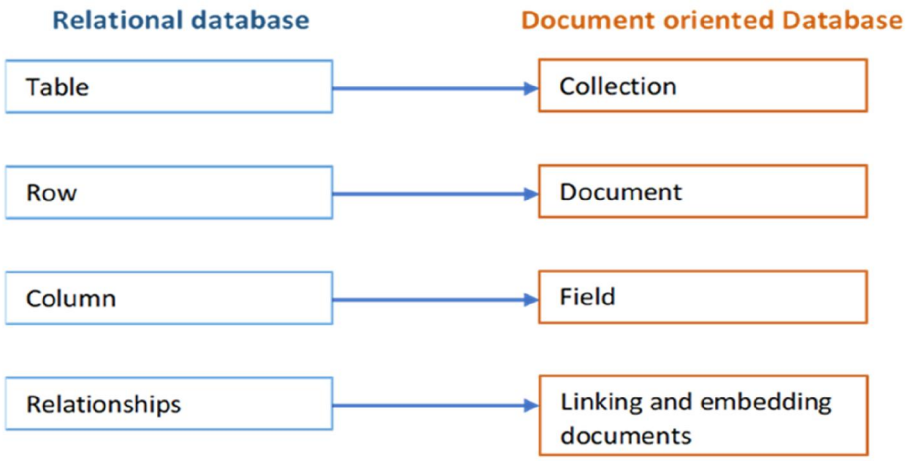

Figure 1: Data Mapping chart for migration from relational to document-oriented database

A data mapping chart can be used to summarise the migration from a relational database to a document-oriented database. Document-oriented databases consist of collections that are analogous to SQL tables in an RDBMS Database. Further, every collection stores data in the form of documents that correspond to rows in an RDBMS table. While an RDBMS row stores data in a set of columns, a document has a JSON-like structure known as fields which is equivalent to RDBMS columns.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, June 2022
Impact Factor: 6.252

In RDBMS relationships among the data are established by linking different tables using the primary key and foreign key with the help of a joint statement which allows you to combine two tables based on a condition. On the other hand, in document-oriented databases relationships are developed using embedded and linking documents.

VII. EXAMPLE

There are 3 tables namely Customers, Addresses, and BankAccounts in the relational data model

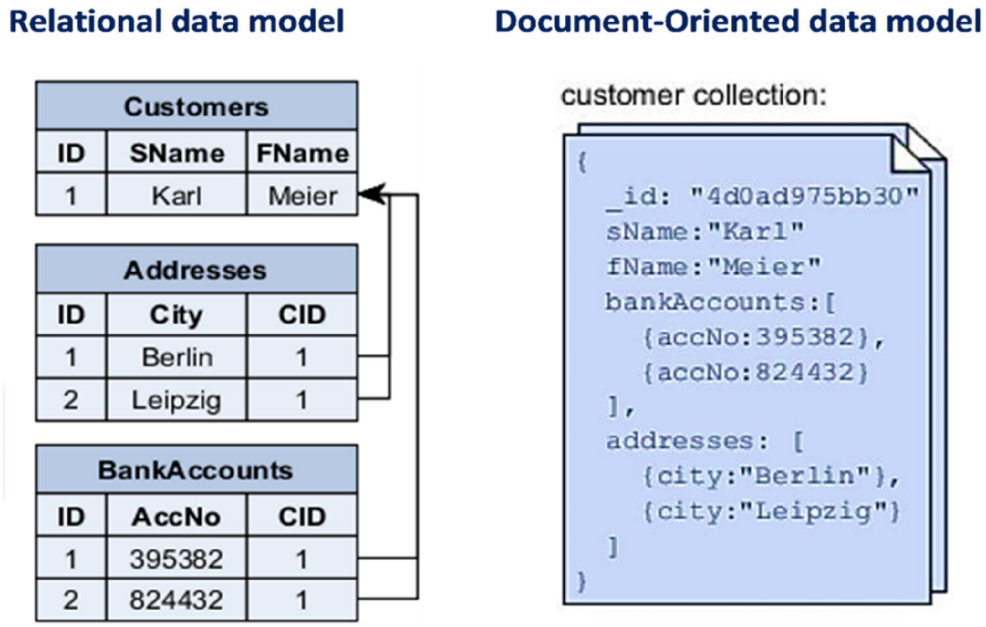

Figure 2: Pictorial Representation of tables

The table 'Customers' is used to store the individual information about different customers. It has three columns: ID to store the customer's unique identification number, SName to store the first name of the customer, and FName to store the second name of the customer. The 'Addresses' table stores the information about the cities from which the customers are. It has three columns ID to store the unique identification number given to a city, City to store the name of the city, and CID to store the customer id and link the 'Addresses' table to the Customers table.

The table 'BankAccounts' has three columns namely ID to store the unique identification number of an account, AccNo to store the account number of the customers, and CID to store the customer id and link the 'BankAccounts' table to the 'Customers' table. In the 'Customers' table ID act as the primary key. In both the 'Addresses' and 'BankAccounts' tables the primary key of 'Customers' table that is ID act as a foreign key. In both the table the customer ID is named as CID and is the foreign key that links these tables.

The table customers have only one and three the ID of the customer is 1, SName is Karl and FName is Meier. In both the 'Addresses' and 'BankAccounts' table we can only enter the value of CID According to the values of ID in the 'Customers' table. Here in the 'Customers' table there is only one entry and the ID is 1. Hence the only possible value for CID in both the 'Addresses' and 'BankAccounts' table is 1. The 'Addresses' table has 2 entries. The ID of the first address is 1 and the city is Berlin and the CID is 1. The ID of the second address is 2 and the city is Leipzig and the CID is 1. The 'BankAccounts' table also has 2 entries. The ID of the first account is 1 and the AccNo is 395382 and the CID is 1.

The ID of the second account is 2 and the AccNo is 824432 and the CID is 1. On mapping, the table 'Customers' will become customer collection. All the columns will be mapped as fields. Hence the fields of the

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, June 2022
Impact Factor: 6.252

collection customer will be id, sname, and fname. The relationship with both the 'Addresses' and 'BankAccounts' table is embedded in the customer collection itself. The embedded 'BankAccounts' table and 'Addresses' table will be like a field. But its value won't be a single value but would be represented by a set of values. The set of values for 'bankAccounts' includes the values of different account numbers. The set of values for 'addresses' includes the values of the different cities.

VIII. CONCLUSION

Even though relational databases offer large security to the data it also lacks many features. These drawbacks lead to the emergence of new databases like NoSQL which offered a solution to most of the problems faced by a relational database. But NoSQL also has its disadvantages. If the data is structured and needs a high degree of data integrity adhering to the principles of atomicity, consistency, isolation, and durability relational databases are the best option. If we need to store large volumes of semi-structured data over distributed nodes without a predefined schema and query and aggregate the data very quickly, use a document-oriented database. If an existing software that already uses the relational database needs the features offered by NoSQL databases the data can be migrated as discussed above. There is no ideal choice. According to the needs of the software, the database should be selected or an existing database should be migrated from one type to other.

REFERENCES

[1] Raghu Ramakrishnan, Johannes Gehrke, Database Management Systems, 3rd ed., McGraw Hill Education; Third edition.
[2] ElmasriRamez, NavatheShamkant, Fundamentals of Database Systems, Pearson Education; Seventh edition.
[3] Gaurav Vaish, Getting Started with NoSQL, Packt Publishing Limited.
[4] C. J. Date, An Introduction to Database Systems, Pearson India; 8th edition.
[5] https://www.flexsin.com/blog/nosql-and-rdbms-advantages-and-challenges/.
[6] https://www.mongodb.com/scale/nosql-vs-relational-databases
[7] https://www.ksolves.com/blog/nosql/advantages-of-nosql-over-rdbms