International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 02 | Feb 2023 www.irjet.net p-ISSN: 2395-0072

AUTOMATIC TRANSFER OF DATA USING SERVICE-ORIENTED ARCHITECTURE TO NoSQL DATABASES

Manu Mishra[1], Mr. Wasif khan[2]

[1]M.Tech, Computer Science and Engineering, SR Institute of Management & Technology, Lucknow, India
[2]Associate Professor, Computer Science and Engineering, SR Institute of Management & Technology, Lucknow

---

Abstract - *Since a few years ago, there has been a meteoric increase in the number of databases that are not genuine relational databases, which may be described as exponential growth. No one term accurately describes these kinds of databases; rather, they can only be characterised by a collection of qualities that are shared by all of them, such as the lack of a predefined schema, inherent scalability features, high performance, data, and so on. NoSQL is the acronym that has been used to refer to these types of databases. Many businesses are beginning to see the value of NoSQL databases and are interested in making the transition to using them. On the other hand, they struggle to move their data since the process requires a great deal of research and thought. The nomenclature and query language used with each kind of database is unique to that database. These businesses will be able to effortlessly move to the NoSQL databases of their choice with the assistance of our innovative automated migration approach, which makes use of the power of service-oriented architecture. We make use of web services that encapsulate some of the most well-known NoSQL databases, such as MongoDB, Neo4j, and Cassandra, among others. This allows us to conceal the inner workings of these databases while still enabling the efficient migration of data with very little or no prior knowledge of how these databases function. To demonstrate the viability of the idea, relational data was successfully moved from the Apache Derby database to the NoSQL databases MongoDB, Cassandra, Neo4j, and DynamoDB, respectively. Each vendor represents a unique NoSQL database type.*

Key Words: NoSQL, Service Oriented Architecture, Cassandra, MongoDB, Neo4j, Amazon

1. INTRODUCTION

Relational databases have been an integral part of many of our web-based programmes, websites, and apps for quite some time. This custom has been around for quite some time. Problems with relational databases include impedance disparity (the gap between what application developers need as data and how data is stored on a disc), the inability to naturally support clusters, and the difficulty in horizontally scaling as data growth occurs. When there is a mismatch between what application developers need and how data is stored on the relational database, a problem known as the impedance gap arises. NoSQL (which stands for "not only SQL") databases were coined to indicate that they did not primarily use the SQL query language, and eventually many other companies followed suit with their solutions.

The International Data Corporation estimates that by 2022, micro-service architectures will be used in the development of 90% of new applications. Current tendencies were used to construct these forecasts. When microservice architecture and the DevOps methodology are used together, the software may be developed with more speed and adaptability. As a result, companies may more quickly introduce their digital goods and services to highly competitive marketplaces. Businesses are starting to modernise their outdated monolithic systems by splitting them up into smaller, more manageable microservices so that they can keep up with the competition and safeguard their competitive edge. While academics and engineers have looked at the problem of migrating monolithic software to a microservice architecture, not as much attention has been paid to how databases should change to accommodate this change. This is a big gap in the knowledge base. Part of the investigation has gone missing in this area, and it must be pieced back together as soon as possible. Experts in the field agree, however, that data management is one of the microservices' most significant limitations. Microservice discovery within monolithic programmes and microservice dissection of source code have been the primary research foci. This topic has been the focus of a great deal of research. The research will be focused on these two primary goals. While transitioning from a monolithic design to a microservice architecture, there is little to no guidance on how to modify current data storage to accommodate the new architecture provided by any of the existing migration approaches. This is because the revised architecture was not considered during the development of any migration strategies. The author believes that, except A. No migration techniques have investigated the adaption of data storage to micro-service architecture outside the one proposed by Levcovitz et al. That's in contrast to A's method. It was proposed by Levcovitz et al. that micro-services be extracted from large, centralised enterprise systems. Unlike strategy A, though. For extracting micro-services from monolith business systems, this strategy departs from the recommendations of Levcovitz and others. Contrary to A.'s assertions, Levcovitz and his group have discovered this to be true.

1.1. NoSQL

Several innovative methods of database administration have emerged in the last decade. Due to the ever-evolving nature of information, relational databases have struggled to keep up with the likes of Google, Amazon, Facebook, and Twitter since the rise of Web 2.0. Social media and other internet records are one such example of the kind of data we are talking about here. These companies have come to this decision because relational databases simply cannot keep up with the volume and velocity of information. The sheer amount of data mined from social media and other online activity logs simply overwhelms traditional relational databases. Moreover, the relationship between the items is not presented straightforwardly by relational databases, and a significant number of join queries are needed to collect all of the information that is related to the objects. Keep in mind that horizontal expansion is not a natural feature of relational databases. You should keep this in mind. Web 2.0 businesses now almost always employ clusters as their default backend infrastructure. The integration of a relational database into the cluster was described as one of the primary challenges that needed to be met. This was a major contributor to the success of NoSQL database systems.

1.2. Aggregate Data Modeling

Data in relational databases is often organized into rows (tuples). No rows may be included inside one another, nor may a row contain a list of values. Can't have it both ways, sorry. Both of the alternatives you listed are not possible. This is because a row can only process a finite amount of information at once. The use of nested data is crucial in disciplines like biology, which deals with living things. Aggregate orientation should be considered as a solution. The aggregate perspective broadens our view of the data so that we may examine its underlying structures, rather than its component records. In this way, the data is more easily digestible. This finding strongly suggests that nesting could be a possibility.

From its origins in Domain-Driven Design, the word "aggregate" is used to describe a collection of interconnected parts that are often treated as a whole. Research of this kind is where the term originates. Databases on a cluster are greatly facilitated by seeing data in aggregates since they are the natural units for sharding and replication. As a result, there is more database compatibility. Aggregates are the most natural method to structure massive datasets, hence they work well for this purpose. Considered in this view, it also facilitates the handling of enormous data sets. Aggregates are easier for programmers to work with than tuples. The information in many NoSQL databases is aggregate-oriented [5], but each of these databases has its unique way of storing this information. MongoDB is a good example of this type of database.

Over the last several years, the popularity of document databases known as "MongoDB" has skyrocketed. A document in MongoDB serves a similar purpose to a row in a relational database. A collection in MongoDB is the equivalent of a table in a relational database, just as a schema is the equivalent of a database and a table is the equivalent of a collection. Because of its use of replica sets, the MongoDB database can provide very reliable availability. The master-slave setup is used for replicating data inside clusters. Adding new nodes to a graph database has no impact on the database's functionality, unlike relational databases. All sorts of enterprises, large and small, have this need.

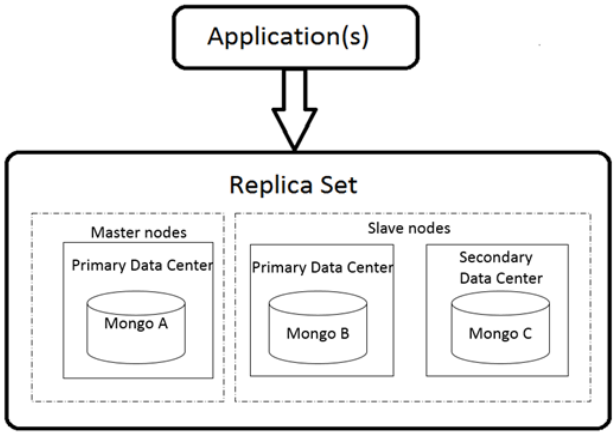

Figure 1: Replica Set in MongoDB in a master-slave configuration

2. SERVICE-ORIENTED ARCHITECTURE

In no way is it a revolutionary idea to place greater importance on service. It achieves this by using the time-tested strategy of "split and conquer" and the principle of "code reuse." Applications built on top of a service-oriented architecture (SOA) may be seen as a set of interrelated services. In contrast, these services don't need to be directly owned by the same corporation, which is a key distinction. You can't overlook this one bit of information. To this point, solutions to each issue may be classified into one of two groups.

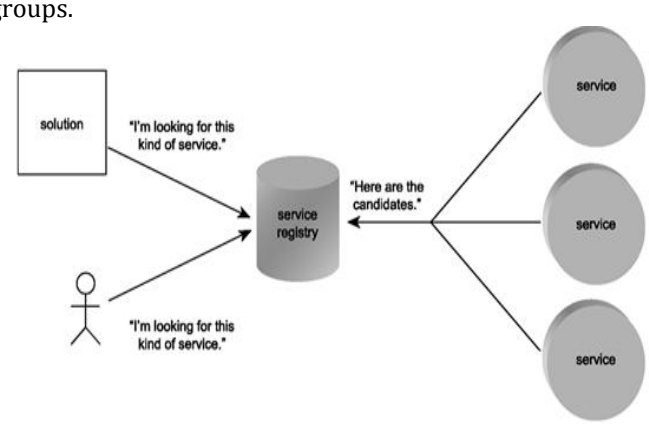

Figure 2: Service registry for candidate services

3. DATA MIGRATION MODEL

With the end aim of creating a migration model in mind, the approach of using the construction technique known as Service Oriented Architecture was selected. During the plan's implementation, a broad range of service options will be drawn from a common pool.

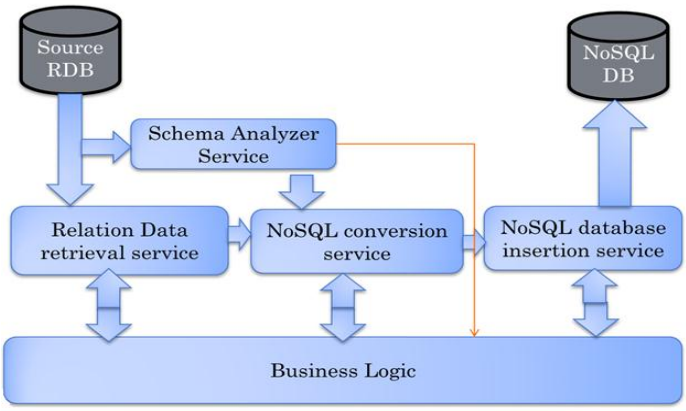

Figure 3: Relational-NoSQL Migration Model

Algorithm 1 ReadFromDB web service

Input: U //Source table URL location, L //List of Tables

Output: S // database in JSON XS:String format

Begin

1: Analyze tables to find whether join operation can be applied on list of tables using primary-foreign key relationship information from source table
2: **for** each table T in L **do**
3: **if** If primary-foreign key exists in T with rest of tables of L **then**
4: Execute join select query on table and store in resultSet D
5: **else**
6: Excecute select query on table and store in resultSet D
7: **end if**
8: Add column header information of T into D
9: **end for**
10: Convert D into JSONArray format J
11: Convert J into XS:String format for BPEL transfer
12: Exit

The algorithm-01 examines the database's structure and draws a judgement on the presence or absence of a primary key-foreign key relational link. If the relationship does exist, then the relevant tables should be joined together and the resulting dataset should include both the joined and individual results. If the connection is missing, the findings should be included in the dataset without any further processing. A modified JSON string containing this dataset will be sent over the internet.

4. IMPLEMENTATION OF MIGRATION MODEL

As a proof of concept, we developed a mechanism to move data from Apache Derby, a relational database, to MongoDB, Cassandra, Amazon DynamoDB, or Neo4j. The design of this system was inspired by the structure of the Apache Derby relational database. To show how well our system performs, we constructed this demonstration system. Figure 4 is a visual representation of the system in action and gives further context for the discussion that follows. Additional references may be found in [the following] During the actual implementation, OpenESB version 2.3 was used. This sample demonstrates the system's "Business Process Enterprise Logic." To condense the whole meaning of "Business Process Enterprise Logic." Information from the previous user is used to request the readDB service, which grants access to the database. This happens after you have acquired the information from the previous user. Next up in the procedure is this step, which follows right after the one before it. Neo4j, MongoDB, Cassandra, or AmazonDB may all be used as the underlying database and their respective insertion services may be used. The ultimate decision rests with the user. To put it simply, multi-service servers are not necessary at any time.

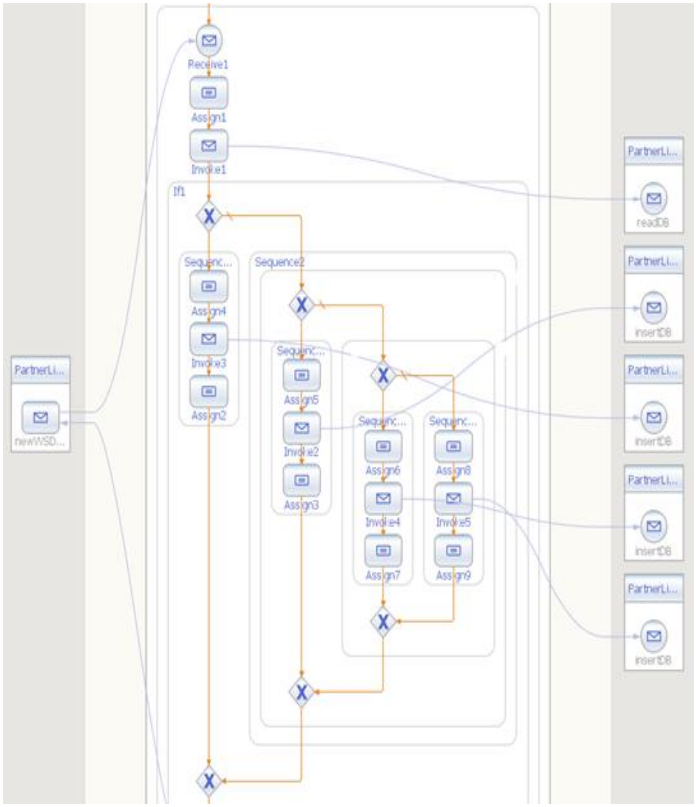

Figure-4: Design of migration models BPEL

5. CONCLUSION

Distributed computing, in which information is processed over a decentralised network, has gained prominence over the last decade as a promising approach to the creation of modern online applications. The value of the cloud computing market has skyrocketed in recent years. Because DBMSs are in charge of storing and displaying an application's data, information has become a crucial part of web-based programmes. It is the goal of this thesis to

provide a framework for designing a system that can migrate data from traditional relational databases to the most popular types of non-relational databases. The creation of such a system will also form the backbone of this thesis. The second objective of this thesis is to provide a tool for migrating data from relational to non-relational storage systems. This research analyses a strategy and approaches that might help software companies migrate their present relational databases to NoSQL providers utilising service-oriented architecture. The University of Washington conducted the study. Both methods and an overall plan are detailed in this study. There is an extra level of abstraction built into Asp. The intention was to facilitate the shift to NoSQL databases as much as possible. Web services have been created to examine the schema of relational databases and implement changes automatically. Through the use of web services, we were able to provide support for a variety of NoSQL databases, including MongoDB, Cassandra, Neo4j, and Amazon DynamoDB; this also allowed for multi-vendor compatibility. Some of the information that may be found in these databases includes: Roughly one-fiftieth of all software applications today are NoSQL systems, and there are over a hundred distinct organisations across the world that provide this service.

REFERENCE

1. K.Mehra, Y.Yan and D.Lemure, Automatic data migration to the cloud in the Sixth International Workshop on Cloud Data Management (CloudDB 2014)

2. Jing Han, E Haihong, Guan Le, and Jian Du. Survey on NoSQL database. In Pervasive computing and applications (ICPCA), 2011 6th international conference on, IEEE, 2011.

3. P. Howard and C. Potter., Bloor research: Data migration in the global 2000 - research, forecasts and survey results

4. Andre Calil and Ronaldo dos Santos Mello, Simplesql: a relational layer for simpledb In Advances in Databases and Information Systems, Springer,2012.

5. Sadalage P.J and Fowler .M, 2013, NoSQL Distilled, Pearson, p.99-109

6. J. Henrard , M. Hick, P. Thiran , and J. Hainaut . Strategies for data engineering. In Reverse Engineering, 2002. Proceedings. Ninth Working Conference on pages 211220. IEEE, 2002

7. M. A. Jeusfeld and U.A. Johnen. An executable meta-model for re-engineering database schemas. Springer, 1994.

8. J. H. Jahnke and J. Wadsack. Varlet: Human-centered tool support for database reengineering. In Proc. of Workshop on Software-Reengineering, 1999.

9. A. Maatuk, A. Ali , and N. Rossiter . Relational database migration: A perspective. In Database and Expert Systems Applications, pages 676683. Springer,2008.

10. K. Haller. Towards the industrialization of data migration: Concepts and patterns for standard software implementation projects. In Advanced Information Systems Engineering, pages 6378. Springer, 2009.

11. B. Bordbar, D. Draheim, M. Horn, I. Schulz, and G. Weber. Integrated model-based software development, data access, and data migration. In Model Driven Engineering Languages and Systems, pages 382396. Springer, 2005

12. Amazon. Amazon DynamoDB. "http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/".Retrieved on November 2014.

13. Apache Cassandra. http://docs.datastax.com/en/cassandra/2.0/cassandra/gettingStartedCassandraIntro html. Retrieved on December 2014.

14. Neo4j graph database. http://neo4j.com/developer/get-started/. Retrieved on December 2014.

15. MongoDB. http://docs.mongodb.org/manual/. Retrieved on June 2014.

16. A. Thakar and A. Szalay. Migrating a (large) science database to the cloud. In Proceedings of the 19th ACM International Symposium on High-Performance Distributed Computing, HPDC 10, pages 430434, New York, NY, USA, 2010.ACM.