International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 04 | Apr -2017 www.irjet.net p-ISSN: 2395-0072

Development of a Middleware Layer for CouchDB NoSQL System for Providing Transactional Properties

Rashmi K[1], Sushmitha S[2], Dr. Kavitha K S[3], Dr. Kavitha C[4]

[1]*PG Student, Department of CSE ,Global Academy of Technology , Bengaluru, Karnataka, India*

[2]Assistant Professor, *Department of CSE ,Global Academy of Technology, Bengaluru, Karnataka, India*

[3]Professor, *Department of CSE ,Global Academy of Technology, Bengaluru, Karnataka, India*

[4]Professor & HOD, *Department of CSE ,Global Academy of Technology, Bengaluru, Karnataka, India*

Abstract - *NoSQL will not follow standard principles of normalization, the de-normalized data results in weaker consistency and atomicity. NoSQL systems have increasingly been used in large scale applications that need high availability and efficiency but with weaker consistency. Various approaches have been proposed to address transaction management in NoSQL databases. However, because of the diverse flavors and kinds of NoSQL databases, there has been no accepted standard approach of managing transactions in NoSQL databases. This paper proposes a model which will provide NoSQL system a transaction management layer. The strategy is to supplement current NoSQL architecture with an extra layer that manages transactions. The proposed model is validated through a prototype system using CouchDB. Preliminary experiments show that it ensures stronger consistency.*

***Key Words*: NoSQL databases, transactions, consistency and atomicity.**

1. INTRODUCTION

Many NoSQL stores compromise consistency in favor of availability, partition tolerance, and speed. Barriers to the greater adoption of NoSQL stores include the use of low-level query languages, lack of standardized interface. Most NoSQL stores lack true ACID transactions.

The idea of Enormous Information has prompted a presentation of another set of databases utilized as a part of the distributed computing condition, that veer off from the qualities of standard databases. The outline of these new databases grasps new components and procedures that bolster parallel handling and replication of information. Information are disseminated over various hubs and every hub is in charge of handling questions coordinated to its subset of information. Every subset of information overseen by a hub is called shard. This strategy of information stockpiling and preparing utilizing numerous hubs enhance execution and accessibility [1]. The design of these new frameworks, otherwise called NoSQL (Not Just SQL) databases, is intended to scale over numerous frameworks.

The essential target of NoSQL frameworks is to guarantee high proficiency, accessibility and versatility in storing and processing huge Information. NoSQL frameworks don't guarantee more grounded consistency what's more, uprightness of information. They in this manner don't execute ACID(Atomicity, Consistency, Isolation, Durability) transaction. Be that as it may, it is essential to give more grounded consistency and uprightness of information while keeping up suitable levels of efficiency, availability and scalability.

In this paper we propose another model that takes into account value-based guideline of standard database frameworks. The goal is to give consistency and to keep up the ACID properties while thinking about the accessibility and proficiency of NoSQL databases. The proposed model separate transaction processing from underlying data storage and to ensure transparency and abstraction. So as to execute simultaneousness, the proposed approach exploits snapshot isolation technique [3].

The potential contributions of the proposed model are summarized as follows.

• The design of a new transaction model for NoSQL systems that maintains ACID properties of transactions in order to ensure stronger consistency.

•Development of an architecture that separates the transactional logic from underlying data thus ensuring transparency and abstraction.

•Development of a prototype system using real NoSQL system, CouchDB.

1.1 NoSQL database systems issues

The following are the issues in NoSQL database system.

- No support for normalization which leads to lack of consistency and atomicity.

- No support for join operation, It lacks the power and flexibility of designing useful queries.
- ACID transactions are not supported in NoSQL databases..
- NoSQL database will not provide schema information.

1.2 Related Work

Different methodologies have been proposed to address transaction management in NoSQL databases. There has been no accepted manageable transaction in NoSQL database.

Deuteronomy [4] is an approach towards transaction processing in NoSQL databases. Deuteronomy isolates the transaction component (TC) from the data component(DC). The TC oversees exchanges and exchanges can traverse different DCs. Rather than the approach proposed in this paper, Deuteronomy makes utilization of locking system to oversee simultaneousness and guarantee consistency. Locking is valuable however it negatively affects the performance of transaction.

G-Store [5], acquaints a key gathering convention with gathering keys for applications that need multi-row transaction. Group inside G-store are dynamic and have a life expectancy. In this manner group will be erased after their life expectancy. Exchanges are constrained to inside a gathering and G-Store can't give exchanges over group.

Megastore [2], utilizes substance bunches arrangement like G-store. Be that as it may, in Megastore, amass development is static and an substance has a place with a solitary gathering for the duration of the life expectancy of that substance. All things considered, ACID exchanges can just occur inside indicated group.

COPS (Cluster of Order Preserving Servers)[6], introduces two variables called dependencies and versions to preserve order across keys. It is implemented using a distributed key value NoSQL database. CloudTPS, like Deuteronomy, make use of two layers architecture which includes LTM(Local Transaction Manager) and the cloud storage. Transactions are replicated across LTMs to preserve consistency in the presence of failures.

2. PROPOSED SYSTEM

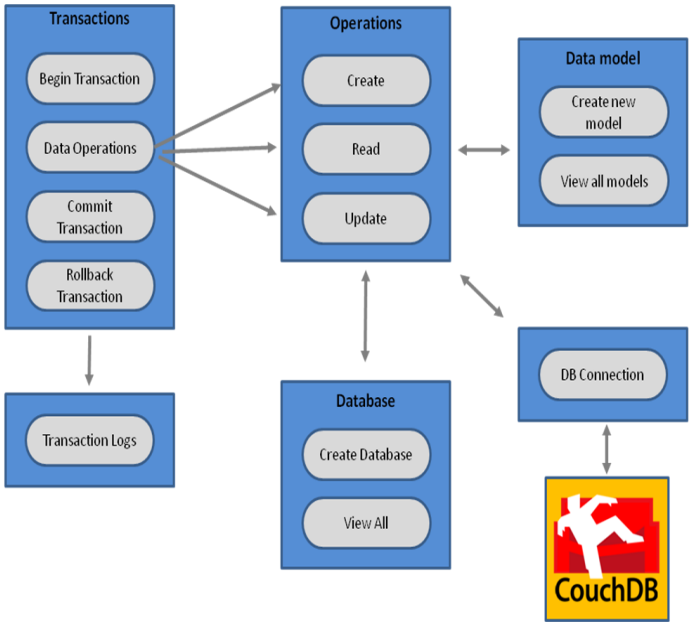

Fig -1: Proposed System Architecture

The architecture splits into following modules as shown in Fig -1.

2.1 Data Access Layer:

Data access layer is the one which exposes all the possible operations on the data base to the outside world. It will contain the DAO classes, DAO interfaces, POJOs, and Utils as the internal components. All the other modules of this project will be communicating with the DAO layer for their data access needs.

2.2 Account Operations:

Account operations module provides the following functionalities to the end users of our project.

- Register a new seller/ buyer account
- Login to an existing account
- Logout from the session
- Edit the existing Profile
- Change Password for security issues
- Forgot Password and receive the current password over an email
- Delete an existing Account

Account operations module will be re-using the DAO layer to provide the above functionalities.

2.3 Connection to Couch DB and Databases:

Here, the end user can create a connection to the couch DB by specifying the host name and the port number of the installed instance. The end user can also connect to a remote couch db that is present in a different geographic location by just entering its host's IP address and the port number. The default port number will be 5984. However, the user can modify this during the couch DB installation. The user can create a new data base or view the list of all existing databases he/she have created using this module. The user can also grant the permission on the database to the other users of the transaction layer. By granting the permission, the user is allowing the other participant to perform any of the transaction operations on the database he/she have created. In addition to that, the user can delete the database or the user can also remove the permission he/she had granted to the other users.

2.4 Operations:

Here, the end user can perform various data operations. The possible data operations include the write access, read access, update, or the delete access. Before the user can perform any of these mentioned data operations, they will have to select the database against which the data operations must be performed. The data operations performed on this module will not be logging anything unlike the Transaction module.

2.5 Transactions:

Here, the end user can initiate a new transaction and perform various data operations on as discussed in the previous division. Before the end user can perform the transaction, he/she will have to select the database against which the transaction has to be executed. The database the user is going to select will be either the one, he/she created him/herself or the one which the other users have granted the access to it. Each and every single operation in the transaction session will be logged in the local mysql table and will be available to view in the GUI. The end user can either rollback or commit the transaction after all the data operations he/she have performed.

2.5 Transaction protocol:

1. As soon as client initialize the transaction request, the transaction layer generates the ID for that transaction and sends the Id and related information to CouchDB.
2. CouchDB by receiving id and related information, CouchDB comes to know what all the data entities the NoSQL transaction going to access, then it will send the id and data entities related information to transaction log.
3. The transaction log saves the transaction details and responds with start time.
4. The CouchDB releases the data entities to transaction layer where actual transaction take place.
5. Once NoSQL transaction is complete, transaction layer sends update, that is if user had updated or deleted data then CouchDB needs to reflect data in data store to ensure data consistency.
6. Transaction log checks whether other transaction has updated after start time of this transaction, if so the transaction log aborts the transaction and response to client through transaction layer. If not continues and sends commit timestamp to CouchDB.
7. Then CouchDB saves data in data store and responds to transaction layer with commit timestamp that is commit is successful.
8. If user requested for rollback, transaction layer contacts transaction log to know the list of operations performed, then reverse operation is performed, say if user deleted data means reverse operation is add data.

The transaction log maintains user details who committed or rollback the transaction. The proposed system allow user to create their own database and datamodel of their interest. The data will be saved in database in encrypted format, which provides security to data.

3. CONCLUSION

We proposed a new model for NoSQL database systems. It provides NoSQL databases with standard ACID transactions support that ensures consistency of data. The project described the design of the proposed model and the architecture within which it is implemented. As a proof of concept the proposed approach is implemented using real Couch DB database system.

4. FUTURE WORK

Generalize the transaction layer to operate across multiple nosql systems so that the adoption of this layer is easy. Integrate the transaction protocol with the NOSQL system provided as a service on the cloud.

REFERENCES

[1] D. DeWitt and J. Gray, "Parallel Database Systems: The Future of High Performance Database Systems," Commun. ACM, vol. 35(6), Jun. 1992.

[2] J. Baker, C. Bond, J. Corbett, and J. Furman, "Megastore: Providing Scalable, Highly Available Storage for Interactive Services.," Proc. of the Conference on Innovative Data system Research (CIDR 2011), 2011.

[3] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P. O'Neil, "A Critique of ANSI SQL Isolation Levels", 2007.

[4] J. J. Levandoski, "Deuteronomy_: Transaction Support for Cloud Data," Conf. on Innov. Data Systems Research (CIDR), California, USA.vol. 48, 2011.

[5] S. Das and A. El Abbadi, "G-Store_: A Scalable Data Store for Transactional Multi key Access in the Cloud," In: Proc. of the 1st ACM symposium on Cloud computing. Indianapolis, USA, ACM, 2010.

[6] W. Lloyd, M. J. Freedman, M. Kaminsky, and D. G. Andersen, "Don't Settle for Eventual_: Scalable Causal Consistency for Wide-Area Storage with COPS. In: Proc. of the 23rd ACM Symposium on Operating Systems Principles. Cascais, Portugal. 2011.