DEVELOPING A GENERIC APPROACH TO MAKE EASY THE DEVELOPER TASK USING MULTIPLE DATA STORES

Chimbili Pallavi[1], G. S. Udaya Kiran Babu[2]
[1]PG Scholar, [2]Associate Professor
Bheema Institute of Technology and Science, Adoni, Kurnool (Dt), AP, India.

ABSTRACT: Nowadays, either relational DBMSs or NoSQL DBMSs are on demand and the application developer must be familiar with the proprietary API of each type of DBMS. However, these APIs are heterogeneous on different levels: API level and Typing level. In order to satisfy different storage requirements, cloud applications usually need to access and interact with different relational data stores having heterogeneous APIs. This APIs heterogeneity induces two main problems. First it ties cloud applications to specific data stores restricting therefore their migration. Second, it requires developers to be familiar with different APIs. For this reason, in this paper we propose an integrated set of models, algorithms as well as requirements aiming at relieving developers task for developing, deploying as well as migrating multiple data stores applications in cloud. Our proposed scheme concentrates essentially on 3 points. First, we provide a unifying data model utilized by applications builders to keep in touch with heterogeneous relational data stores. Based on that, they express queries making use of OPEN-PaaS-DataBase API (ODBAPI), targeted relaxation API allowing programmers to write down their applications code independently of the target data stores. 2nd, we presents Virtual Data Stores, which act as a mediator and communicate with integrated data stores wrapped through ODBAPI. Ultimately, we present a declarative technique that makes it possible for to lighten the burden of the tedious and non-standard tasks.

I. INTRODUCTION

The success of the Database-as-a-service model has introduced a huge style of business and study approaches. Although many had been broadly studied, there is an absence of taxonomy. In the path of this work, we will distinguish between managed database offerings, proprietary database offerings and Backend-as-a-services (BaaS). In a managed database carrier a DBaaS supplier presents a cloud-deployed DBMS as well as automates operational tasks comparable to provisioning, multi-tenancy, backups, safety, entry manipulate, flexibility, scaling, performance tuning as well as replication. Proprietary database offerings build on recently designed database techniques (e.g. Google DataStore situated on Megastore) or combine exceptional databases to a polyglot persistence atmosphere to present them by way of provider-targeted protocols and APIs. The Backend-as-a-provider model enhances the DBaaS model by using utilizing including abstractions for backend concerns of mobile purposes and internet sites (e.g. Authentication, push notifications, data justification as well as property). DBaaS methods fluctuate inside the degree to which they furnish automation of operational responsibilities, the underlying expertise store, pricing gadgets, Server Level Agreements (SLAs), multi-tenancy techniques and, most significantly, their interfaces. Whilst Proprietary database choices offer supplier-unique interfaces, managed database choices present database-particular protocols that were not planned for cloud environments. As described by way of Fowler et al. This encompasses NoSQL databases that have a (possibly denormalized) combination as their primary unit of entry: rows in enormous-column and file stores, documents in file stores and key-value pairs in key-value stores. This aligns well with the valueless resource-oriented model of amusement. The key assertion and inspiration for this work is that mixture-oriented DBaaS programs may also be greatly elevated with the aid of supplying them by way of a single unified and scalable relaxation API. That is appealing as consumers can also be reused, understanding items be shared and applications be migrated. Customers drive into the computing Cloud with data as well as applications. Some Cloud programming models must be proposed for customers to adapt to the Cloud infrastructure. For the ease and convenient entry of Cloud offerings, the Cloud programming model, nonetheless, should not be too complicated or too progressive for end customers. The MapReduce is a programming model and a related implementation for processing and producing tremendous data units diagonally the Google worldwide infrastructures. The MapReduce model first of all includes applying a “map” operation to a few data files – a set of key/value pairs, after which techniques a “reduce” operation to the entire values that, shared the equal key. There have been large changes taking situation in predicament to internet progress, not too long ago most of structured databases are converted to unstructured databases. The cause behind this is, due to the innovation and use of many social media. This data which is extracting from one of a kind sources are in special type. They are stored in specific varieties of databases. Gaining access to these databases is way intricate given that the developer has got to know all these databases and their exact API. The complexity rises as the connectivity is made to distinct database. For example, to connecting a single database the developer is required to load its driver separately and select a correct API after which participate in the one-of-a-kind operation. Within the context to the heterogeneity of API, the proposed approach implemented a fashioned interface which is makes use of RESTful API to attach one-of-a-kind structured as well as unstructured data stores in cloud. On this procedure the interface helps to connect to the server and makes it possible for performing

the unique operations on the structured and unstructured databases. An API helps to realize a proper database as per developer's specification and makes connectivity to it. The distinct operations to the databases are carried out by means of the API. In this method the developer has to specify the operations and the execution is performed as per specification. The procedure makes use of one of kind databases like Mongo database, couch database that are unstructured databases. It will necessary for developer to access such new science without understanding their distinctive API which in flip will precious to control the development burden.

II. RELATED WORK

Dr Elaine Shi described a few enabling technologies closer to this vision. Principally, she told about 1) preserve users' data against probably compromised purposes; 2) the right way to safeguard customers' data against a probably compromised computation supplier; and three) the way to preserve users' data in opposition to a potentially compromised storage provider. She instructed about our ongoing effort at integrating these technologies to provide a cloud infrastructure which provides data protection at the platform level. In this method, users can advantage from the rich cloud purposes without stressful in regards to the privacy of their data; and utility developers can focus on setting up performance at the same time offloading the burden of offering safety and privacy to the cloud platform. Researchers have realized that supporting allotted transactions does no longer enable scalable and available designs. Therefore to satisfy the scalability requisites internet functions, designers have scarified the capacity to support dispensed transactions. This resulted in the design of easier data outlets based on the important key-value schema, the place tables are considered as a huge assortment of key value entries. Key-value stores akin to Bigtable, PNUTS, Dynamo, ecStore as well as their open source analogous, were the desired data outlets for applications within the cloud. The property common to all systems is the important key-value abstraction where data is considered as key-value pairs and atomic entry is supported most effective at the granularity of single keys. These methods are restricts access granularity to single key accesses, although providing minimal consistency and atomicity guarantees on multi-key accesses. At the same time this property works well for current functions, it's inadequate for the following generation internet applications which emphasize collaboration. In view that collaboration by means of definition requires constant entry to groups of keys; scalable and steady multi key access is valuable for such functions. Extra the notion of key-value stores and accesses on the granularity of single keys used to be put ahead as the sole means to achieve high scalability along with availability in such techniques. Based on these ideas, a quantity of key value outlets also called row stores like Bigtable, PNUTS, Dynamo, ecStore as well as Hbase had been designed and successfully implemented. Single key atomic entry semantics naturally makes it possible for effective horizontal data partitioning, and provide the basis for scalability and availability in these techniques. Nonetheless, all these key value stores even though enormously scalable, discontinue short of supplying transactional guarantees even on a single row. Thus to satisfy the scalability requirements of net functions, designers have scarified the capability to support allotted transactions.

III. FRAMEWORK

A. System Overview

As shown in Fig. 1, an interface is built through which different structured as well as unstructured data stores in cloud are connected.

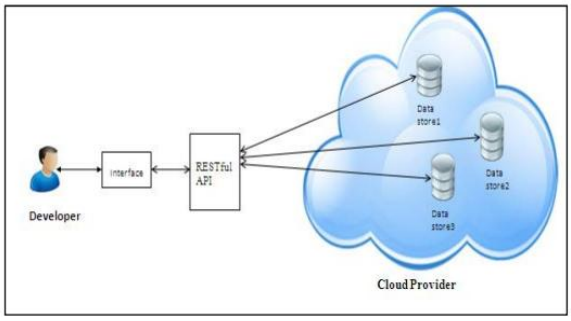


Figure1. System Framework

The RESTful API helps to connect with databases which are in a cloud. The API connects to the database as per specification of the developer. The proposed system performs CRUD (Create, Read, Update and Delete) operations on the data stores as per developer's requirement. After executing the specified query the result is sent back to the API. On the server side, it shows all the performed operations and its status. On the API it shows the status of a query, either execution of successful or unsuccessful with its code value.

B. REST API/Services

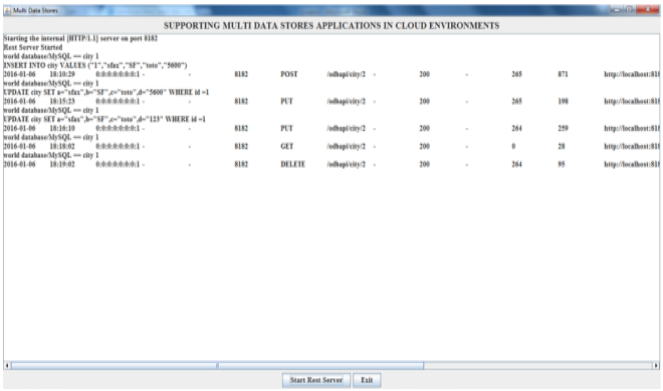
RESTful internet services are offerings which might be constructed to work best on the net. Representational State Transfer (REST) is an architectural form that specifies constraints, such as the uniform interface, that if utilized to a web provider result in desirable houses, corresponding to efficiency, scalability, and modifiability, that enable offerings to work first-rate on the internet. In the REST architectural style, data and performance are considered assets, and these resources are accessed utilizing Uniform Resource Identifiers (URIs), often hyperlinks on the internet. The resources are acted upon by way of using a suite of simple, well-outlined operations. The REST architectural form constrains architecture to client-server architecture, and is designed to use a stateless conversation protocol, most of the time HTTP. Within the REST architecture kind, customers and servers exchange representations of assets using a uniform interface and protocol. These principles are encourages RESTful purposes to be easy, lightweight, and have excessive efficiency. RESTful net services most often map the four principal HTTP ways to the operations they execute: create, retrieve, update, and delete. The following table indicates a mapping of HTTP ways to the operations they participate.

C. Virtual Data Stores

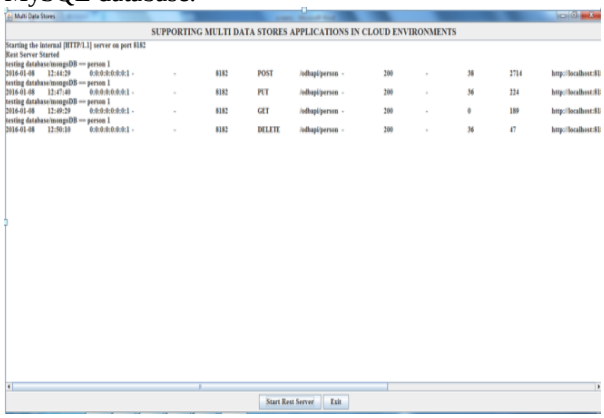
Wrapper REST services alter execution effortless queries over the concerned data stores. Nevertheless, they may be not meant to execute advanced queries (corresponding to become a join, union, and so forth.). In our technique, we have a tendency to recall Virtual Data Store (VDS) a selected detail responsible for execution queries submitted by means of multiple data store application. Multiple data store software submits CRUD and intricate queries to the VDS which might be liable of their execution by means of interacting with acceptable data store through their REST services. VDSs alter developers to distinctive their a part of queries over more than one data stores in the course of a declarative manner and take in charge the burden of their executions.

IV. EXPERIMENTAL RESULTS

To reduce the developer effort for developing tasks with multiple data stores in this project we proposed or developed ODBAPI. In our experiments, we used two databases such as MySQL and MongoDB. By using REST Client API we perform the CRUD operations such as insert record, update record, Delete record and Retrieve record on these two databases.



The above screen describes that the CRUD operations done on MySQL database.



The above screen describes that the CRUD operations done on MongoDB database.

V. CONCLUSION

In this paper we conclude that, to execute the complex queries with multi data stores, we proposed an integration of set of models, algorithms as well as tools. Through these we can achieve to task of developing, deploying as well as migration of multi data store applications in the cloud. The proposed system focused mainly on three considerations. These are (i) ODBAPI for CRUD Operations (ii) Virtual data stores for complex queries execution and (iii) Manifest for data stores discovery as well as automatic application deployment.

REFERENCES

[1] R. Sellami, S. Bhiri, and B. Defude, “ODBAPI: a unified REST API for relational and NoSQL data stores,” in The IEEE 3rd International Congress on Big Data (BigData’14), Anchorage, Alaska, USA, June 27 - July 2, 2014, 2014.

[2] P. Atzeni, F. Bugiotti, and L. Rossi, “Uniform access to nonrelational database systems: The sos platform,” in Advanced Data Systems Engineering - 24th International Conference, CAiSE 2012, Gdansk, Poland, June 25-29, 2012. Proceedings, 2012, pp. 160– 174.

[3] O. Cur´e and et al., “Data integration over NoSQL stores using access path based mappings,” in Database and Expert Systems Applications-22nd International Conference, DEXA 2011. Proceedings, Part I, 2011, pp. 481–495.

[4] N. Ghosh and S. K. Ghosh, “An approach to identify and monitor sla parameters for storage-as-a-service cloud delivery model,” in Workstores Proceedings of the Global Communications Conference, GLOBECOM 2012, 3-7 December, Anaheim, California, USA, 2012, pp. 724–729.

[5] D. Kossmann, “The state of the art in distributed query processing,” ACM Comput. Surv., vol. 32, no. 4, pp. 422–469, Dec. 2000.

[6] M. Sellami, S. Yangui, M. Mohamed, and S. Tata, “Paasindependent provisioning and management of applications in the cloud,” in 2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, June 28 - July 3, 2013, 2013, pp. 693–700.

[7] R. Sellami and B. Defude, “Using multiple data stores in the cloud: Challenges and solutions,” in Data Management in Cloud, Grid and P2P Systems - 6th International Conference, Globe 2013, Prague, Czech Republic, August 28-29, 2013. Proceedings, 2013, pp. 87–98.

[8] M. Pollack, O. Gierke, T. Risberg, J. Brisbin, and M. Hunger, Eds., Spring Data. O’Reilly Media, October 2012.

[9] P. Atzeni, F. Bugiotti, and L. Rossi, “Uniform access to nonrelational database systems: The sos platform,” in Advanced Data Systems Engineering - 24th International Conference, CAiSE 2012, Gdansk, Poland, June 25-29, 2012. Proceedings, 2012, pp. 160– 174.

