Journal of Engineering Sciences
Vol 11, Issue 1, JAN/2020
ISSN NO:0377-9254

A NO SQL DATABASE MODEL WITH NESTED TRANSACTIONS USING MULTI-TENTANT ARCHITECTURE

D.S.C. REDDY[1], KANAMATHAREDDY RESHMA REDDY[2]

[1]Professor & HOD, Dept. of CSE, PBR Visvodaya Institute of Technology and Science, AP, India.

[2]M.Tech, Dept. of CSE, PBR Visvodaya Institute of Technology and Science, AP, India.

Abstract – As the world is frantically moving towards digitization, handling huge volumes of data is becoming complex and difficult to optimize. This is mainly because of increasing dimensions of data, its unstructuredness and increased need for storage space. To increase the efficiency of the cloud environment, many service providers have moved towards multitenant architecture. In a multi-tenant architecture, a number of individual applications termed as tenants work under a shared environment. In such a scenario, it is imperative that data of one tenant be isolated from that of other tenants. Further, it is also desirable to optimize the use of storage space. With these considerations, this paper proposes multi-tenant database architecture for applications using a column-based NoSQL data store. The architecture has been implemented with the NoSQL database, Cassandra.

Keywords – multitenant architecture, Cassandra, Queries Processing.

I. INTRODUCTION

With overall increase in the measure of data rise, cost of planning the data relies already increased hugely. Considering this, many application services have affected or have strategic now been moving against distributed environment. In the several years the distribution of distributed computing and also the study of distributed systems have magnified gradually. It is also not possible to storage, compute the humungous data on a single machine; thereby leading to the need for distributed and cloud based answers for data storage.

Data being kept in distributed environment [1] or booming a stratus is of large rates and requires storage space to deal with it. it has been observed that triplex customers of cloud based applications, such as certain exploitation software as a service role model; share whatever park data. Storage of the data every application in very cloud ends in data that is typically redundant, because of multiple

copies of the same figures. for instance, consider a text file of 2MB that gives been shared within 10 people over a network. instead of storing this text file for all 10 users, a instance of this file can be shared inside all applications; thus, saving a lot of memory space which is one of the most critical instruments in the cloud environment. this architecture in which multiple applications, i.e., tenants can share single data copy is termed as multitenancy.

Multitenancy, in the conditions of data storage, is employed to minimize misuse of supplies. Moreover, this leads to employee benefit given lowering the requirements of maintenance and storage. However, it is also needed that data of multiple tenants is isolated separately other. This isolation is required for security as well as protection from unwarranted modifications of data. Thus, a successful implementation of multitenant data store should ensure optimal use of storage space as well as isolation of data of tenants from each different. One more necessity is the need for flexibility in data syntax's. This is necessary a tenant to possess some detail data attributes that may not use by other tenants.

This paper recommends the practice of a multi-tenant data store using a column based No SQL database. No SQL firewalls are gradually getting used through cloud based applications since they offer features of data sharing, flexibility of schema design and dealing with of unstructured nature of data [2, 3]. One important family of No SQL databases is column based databases which additionally will be appropriate because handling sparse data. Amanda [4] has been employed for practice of the proposed multi-tenant data store. However, the system might be applied to use columnar database.

II. PROPOSED SYSTEM

Normal relational databases will not be well equipped to address high volume of unstructured data. It is which really No SQL databases relate. The present play uses apache Cassandra; a column based No SQL database management system which is a distributed, expansible and highly available data store. Does not have a single point of loser. The design of a multitenant architecture using prophetess underdog adorned in the overall fig 1 and delineate after fetching an event booking system as a SaaS application.

A. extracting data from the data source and design table schema:

The proposed system continues to follow a shared database, same schema multitenancy model because an effect booking system.

Because this application efficient could be different tenants, namely movie, conference/workshop, adventure parks etc. the creation of the shared out method almost all feasible attributes (columns) using the query language of prophetess, CQL is illustrated in fig 2.

```
CREATE TABLE
ticketbooking.events (
event_code int,
actor text,
venue_code int,
director_name text,
duration int,
Movie_genre text,
Event_type text,
EVENT_title text,
occu_perc decimal,
ticket_price decimal,
tickets_sold int,
title_year int,
total_sales decimal,
location_venue text,
event_organisors text,
event_startingdate date,
event_enddate date,
email text,
start_time text,
end_time text,
facebookpage text,
Primary Key(event_code,Event_type);
```

Fig. 2: Events table

B. Import the extracted data into the Cassandra data store:

The data for the multiple tenants is imported into the Cassandra database from the application.

C. Isolate tenants' data by creation of Materialized Views (MV):

Cassandra gives a feature to create took place views. This feature is used by the expected system to isolate a tenant's data from other properties. Use of one MV for one tenant also consists of trait in study being used by each tenant. Further, Cassandra is apt at dealing with sparsity in data. This is required as some columns values can be null for a particular tenant. For instance, for an event type "conference", value of columns Movie_genre, director_name, duration, actor will be null. Similarly, for an event type "movie", value of columns facebookpage, email, event organists are null. Thus, each row can have multiple null ethics, to handle such sparse data we have used Cassandra's storage engine. here, being a column family No SQL data store, took place views can be discovered which only store the articles present in that row for each tenant. Each tenant's MV can be observed below.

Materialized View for Tenant MOVIE

create MATERIALIZED VIEW Movie

AS Select event_code, actor_1_name, actor_2_name, actor_3_name, director_name, duration, movie_genre,event_title, event_type, location_venue, occu_perc, start_time, ticket_price, tickets_sold, title_year, total_sales, venue_code from events where event_code is not null and venue_code is not null and event_type is not null primary key (event_code,event_type,venue_code);

Materialized View for Tenant Conference/Workshop

create MATERIALIZED VIEW Conference AS Select event_code, event_title, event_type, location_venue, start_time, email, end_Time, event_enddate, Event_Organisors, event_startingdate, event_title, FacebookPage, venue_code from events where event_code is not null and venue_code is not null and event_type is not null primary key((event_code,venue_code),event_type);

Materialized View for Tenant Music Festival

create MATERIALIZED VIEW MusicFestival AS Select event_code, event_title, event_type, location_venue, start_time, email, end_Time, event_enddate , Event_Organisors, event_startingdate, event_title, FacebookPage, venue_code, ticket_price, tickets_sold, occu_perc from events where event_code is not null and event_type is not null and event_title is not null primary key((event_code,event_type),event_title);

III. RESULT ANALYSIS

The materialized perspectives set up will be then used to outgrowth surveys for different properties in Cassandra data store engine, thus giving a way of insulation to each tenant. Effects had been tabulated in fig 4.

Unlike MySQL, Cassandra does not explicitly show time taken to practice a question, control tracing on/off is used to compute the timestamp in Cassandra. Cassandra promote sets of data, for small diagrams Cassandra is much slower than MySQL [9].

The effects obtained prove that multitenant architecture give desired results for large set of data in less time but it is not recommended to use multitenant architecture for small set of data. Set of data used in the proposed system comprised of 360 rows, among these some were sparse i.e. maximum values of columns were null.

Table 1: Sample Queries with processing time

S. No	QUERIES	PROCESSING TIME (ms)
1.	Select * from movie ;	0.336
2.	Select event_title, director_name, location_venue from movie where event_type='Movie' allow filtering;	0.409
3.	select * from conference where event_type =	0.31

	’Conference’ allow filtering;	
4.	select Sum(tickets_sold) from Movie where event_code =735 and event_title=’The Departed’;	0.26
5.	select Max(ticket_price) from MusicFestival where event_code =139 and event_type=’Music’;	1.10
6.	Select event_title, location_venue, start_time, event_startingdate from conference where venue_code in (1022,1061) and event_code in (731) ;	1.21
7.	select event_title, director_name,actor_1_name from Movie where movie_genre=’ Crime|Drama|Thriller’ and director_name=' Joss Whedon' Allow filtering;	0.98
8.	select count(*) from MusicFestival where event_code =735 and event_type=’Music’;	0.78

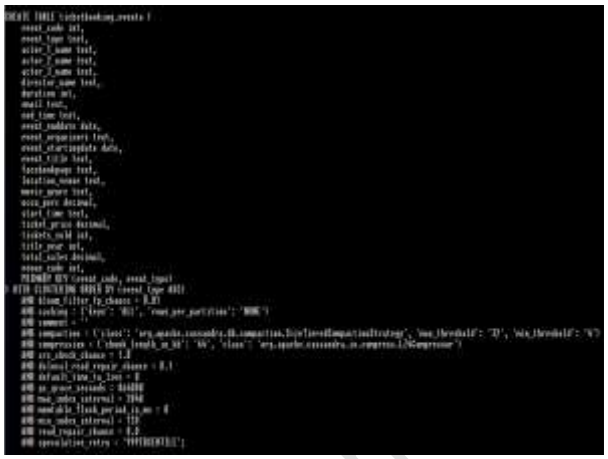

Fig 4: Schema of Table Events

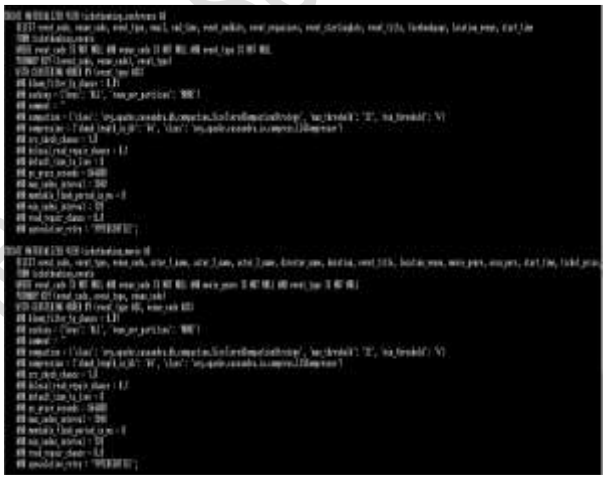

Fig 5: Materialized View for different Tenants

IV. CONCLUSION

The paper has given No SQL based architecture for a multi-tenant data store. a column founded No SQL family database, Cassandra has been used for` the practice of the data store. The advantage of the projected system lies in its ease of implementing data isolation among tenants while allowing flexibility of schema design or extension.

Results of implementation show so the projected multitenant architecture supplies wanted data isolation to its tenant along with operates considerably for giant datasets but for small datasets, query interval will increase.

REFERENCES

[1] W. Zeng, Y. Zhao, K. Ou, and W. Song. “Research on cloud storage architecture and key technologies”. Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, 2009.

[2] J. Han, H. E, G. Le and J. Du. “Survey on NoSQL database”. 6th international conference on pervasive computing and applications IEEE , pp. 363-366, 2011.

[3] R. Burtica, E. M. Mocanu, M. I. Andreica and N. Tapus. ”Practical Application and Evaluation of NoSQL Databases in Cloud Computing”, in Proceedings of Systems Conference (SysCon), IEEE International., Vancouver, 2012, pp. 1-6.

[4] G. Wang and J. Tang. “The nosql principles and basic application of cassandra model”. International Conference on Computer Science and Service System, pp. 1332-1335. IEEE.2012.

[5] G. Karataş, F. Can, G. Doğan, C. Konca and A. Akbulut, "Multi-tenant architectures in the cloud: A systematic mapping study". International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, pp. 1-4,2017.

[6] A. Jumagaliyev, J. Whittle and Y. Elkhatib, "Using DSML for Handling Multi-tenant Evolution in Cloud Applications," IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Hong Kong, pp. 272-279, 2017.

AUTHORS

D.S.C. REDDY working as a Professor & HOD, Dept. of CSE, PBR Visvodaya Institute of Technology and Science, AP, India.

Kanamathareddy Reshma Reddy has received her B.Tech Degree in Computer Science and Engineering from Lakireddy Balireddy College of Engineering, JNTU Kakinada in 2014 and M.Tech in Computer Science from PBR VITS, JNTU Anantapur in 2019.