ISSN (e): 2250 – 3005 || Volume, 05 || Issue, 03 || March – 2015 ||
International Journal of Computational Engineering Research (IJCER)

Efficient Data Distribution in CDBMS based on Location and User Information for Real Time Applications

Tejas Bhatt[1], Mr. Manjunath H[2]

1 PG Scholar, Dept. of Computer Science & Engineering, Mangalore Institute of Technology, Karnataka, India,
2 Associate Professor-I & Head(ISE), Dept. of Information Science & Engineering, Mangalore Institute of Technology, Karnataka, India

ABSTRACT:

In today's world users need to store data in every application and applications has a lot number of users data stored at databases. The retrieval time of data depends on how many records a database contains and not the configuration of the server. The less the number of records the less the retrieval time. Many methods are available when it comes to data partitioning, each with its own functionality for a specific application and requirement. This paper focus on the methods of data partitioning and the effective algorithm to partition the data equally between N numbers of nodes running on the cloud having database with same entities which contain the distributed data of user used in real time user application. Proposed method concentrate on data distribution in real time "user" application where user activities are primary requirement.

Key Words: *Data, Database, Data Distribution, Data Partitioning, Horizontal Partitioning, Vertical Partitioning, Cloud Database, User Information, Database Server, Nodes, CDBMS, OLAP, Fragmentation, Index.*

I. INTRODUCTION

In real time application with user interaction needs to store many records of user and all data of user is distributed among the databases in order to reduce data retrieval time. [1] CDBMS is a cloud database management system where databases are running at different nodes which are running over one or more clouds. This provides data availability and prevent failure from node crash. The challenge over here is to distribute user data in such a way that all databases gets same amount of data in the form of records and since the size of database is equal, it is easy to distribute workload among the cloud nodes. It means with every insertion of record the data of user has to be got spited in such a way that the possibility of equal records in all databases is more. The outcome of this approach is all nodes have same size of database which helps to distribute workload and the data retrieval time also gets reduced as the number of record in one data table is lesser compared to without distribution.

Data distribution approaches changes with the change of the application and the requirement. A real time user application has two main modules Signup and Login. To speed up the user operations it is likely to distribute data in such a way that each module needs only specific data and other data can be ignored. While signup, user actually insert the data which has two type of information, one is personal info and one is login info. The idea here is to separate this two info so that while searching other users the login info can be ignored and while login to the application the personal info can be ignored.

There are two main methods of data partitioning

1) Horizontal Partitioning: here the data is partitioned based on row attributes or we can say row based partitioning.
2) Vertical Partitioning: here the data is partitioned based on dimensions or we can say column based partitioning.

Example:

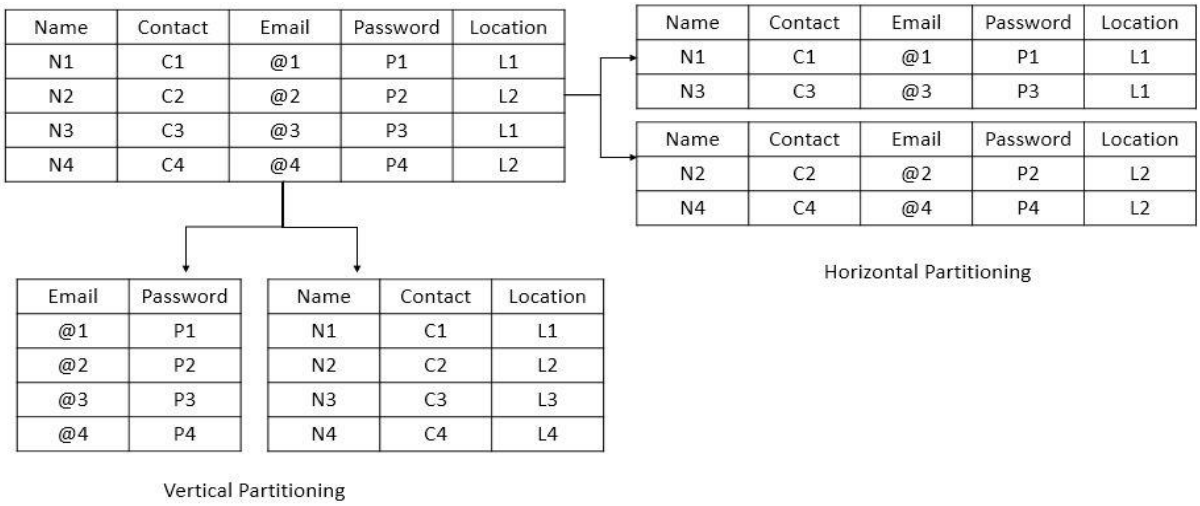

Fig 1: Partitioning Methods

II. LITERATURE SURVEY

[2] There are four main partitioning strategies available. First one is range partitioning where the records which comes under same range stored in a particular partition. For example age is an attribute which can be used as rang partitioned data such as all users with age 21 to 30 get stored in same partition. Second is list partition where records get stored in same partition which has attribute value defined in the list such as user's records with Asian countries gets store in same database (horizontal partitioning based on location). Third is hash partitioning where a hash function returns a value for a record to get store in particular partition. And fourth is composite partitioning which is the mix of all the above partitioning methods. [2] Many techniques of databases are implemented such as (1) selectively based partitioning, where the relation is partitioned based on selective joins on which sub queries are computed and merged to get the whole query. (2) Second technique is of fragmenting data which is used in data warehousing where minimum response time is required and in order to do that parallel execution of queries is necessary. In this approach the data distribution is carried out by generating star schemas which contains fact table (table with key attributes) and dimension tables (tables with attributes). To develop "best" dimension tables greedy algorithm is used. Such kind of queries are called OLAP (On-Line Analytical Processing) queries. (3) Mars, this is an approach where data distribution techniques applied to support reload in main memory database system. To prevent failure, database from archive memory (AM) to main memory (MM) is required. In this approach the partitioning approach (Horizontal or Vertical) is based on what kind of operation is needed. For deletion and updating horizontal partitioning is appropriate whereas when it comes to insertion, single vertical partitioning is better. (4) Genetic algorithm based clustering approach, in this approach clusters of data are created based on access of data. Since accessing data from database returns only a subset of the database it is more likely to put related searched data together in one Server(s)/Site(s) which forms clusters of data. (5) Physical and Virtual Partitioning in OLAP, as On-Line Analytical Processing (OLAP) requires high performance by reducing response time, intra query parallelism used where each query executes on every subsets of query tables. Virtual partitioning is a process where clusters of data subsets created on actual physical data and on that virtual partitioned of data query parallelism occurs. Combining physical and virtual partitioning makes it more flexible for intra query parallelism. (6) Near Uniform Range Partitioning, for databases with mass data it is more likely to partition the data based on range. Traditional range partitioning approach doesn't work on increased partitioning. Increased partitioning algorithms allowed the data being partitioned in each range which makes it easy to maintain partitions automatically, effectively and rapidly. (7) Dynamic vertical partitioning, in this approach the data is partitioned vertically based on information, queries and data attributes without manual work of DBA (Database Administration). An active rule-based statistic collector gathers information which can be used for vertical partitioning and based on those information data is partitioned vertically automatic and without intervention of DBA. There are chances that fragmented data can be again re-fragmented based on upcoming new data attributes and user queries. (8) Fine-grained Indexing, websites where user insertion operations are more than the retrieval or updating operations uses Shinobi system which indexes the database horizontally and create indexes based on frequently accessed data and drop those indexes which are infrequently accessed which provides great space management and fast query processing.

III. PROPOSED METHOD

Existing system works on distributing data based on location in list partitioning approach but it becomes non useful when users from one locations are more than the users from other location. Proposed method is for the real time application where the data gets distributed as it arrives to the system. This method focus on equal distribution of data when the probability about the number of users from specific location cannot be determined. Proposed method uses horizontal and vertical both partitioning approaches to equalize the records between number of databases when users with one location are more than the users with other location.

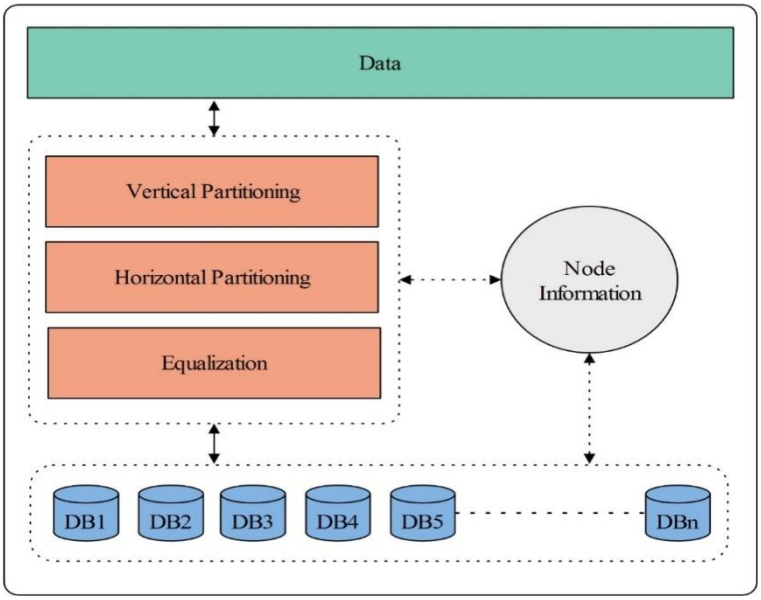

Fig 2: Overall Architecture

Step 1: Horizontal Partitioning based on location

As shown in Fig 3 when three records gets horizontally partitioned between two different databases based on location, record 1 gets inserted into DB1 and record 2 and 3 gets inserted into DB 2 as record 2 and 3 have same location. Now there is no equal number of records in databases. (DB1: 1 record, DB2: 2 records)

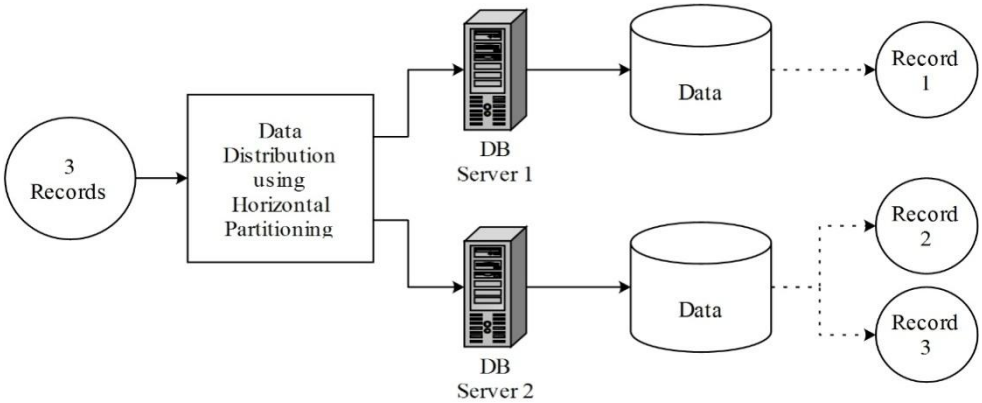

Fig 3: Horizontal Partitioning

Step 2: Apply Vertical Partitioning on each record stored at different database server

To prevent unequaled distribution vertical partitioning is applied on records stored at each database. As shown in Fig 4 it results in double in number of records. Initially DB1 had one record which got partitioned vertically and hence now there are 2 records in DB1. Initially DB2 had two records which got partitioned vertically and hence now there are total 4 records in DB2. (DB1: 2 record, DB2: 4 records)

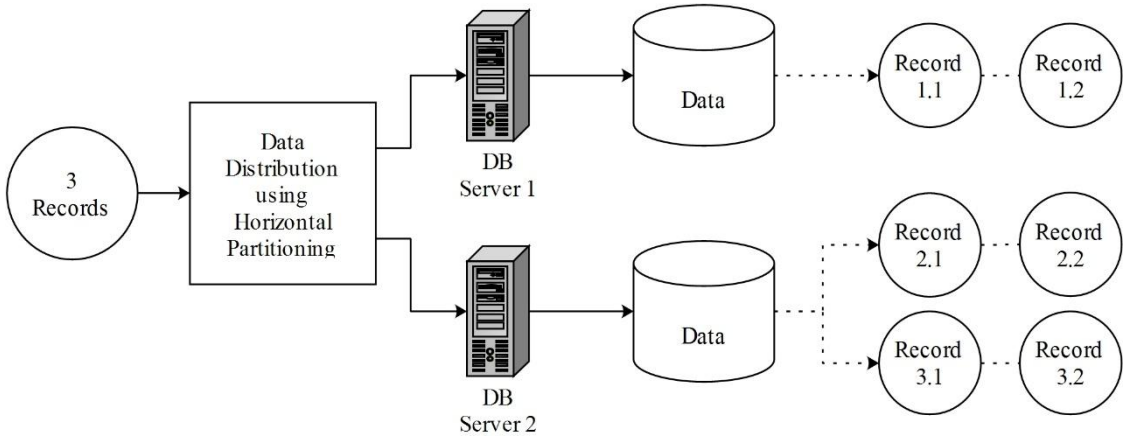

Fig 4: Vertical Partitioning

Step 3: Exchange the vertically partitioned data between both nodes

The final step is the equalization by exchanging the vertically partitioned records which in this case record 1.2 in DB1 moves to DB2 and records 2.2 and 3.2 in DB2 moves to DB1. Finally there are equalized databases where both databases have same number of records which in this case DB1 has 3 records and DB2 also has 3 Records as shown in Fig 5. (DB1: 3 record, DB2: 3 records) – Equalization

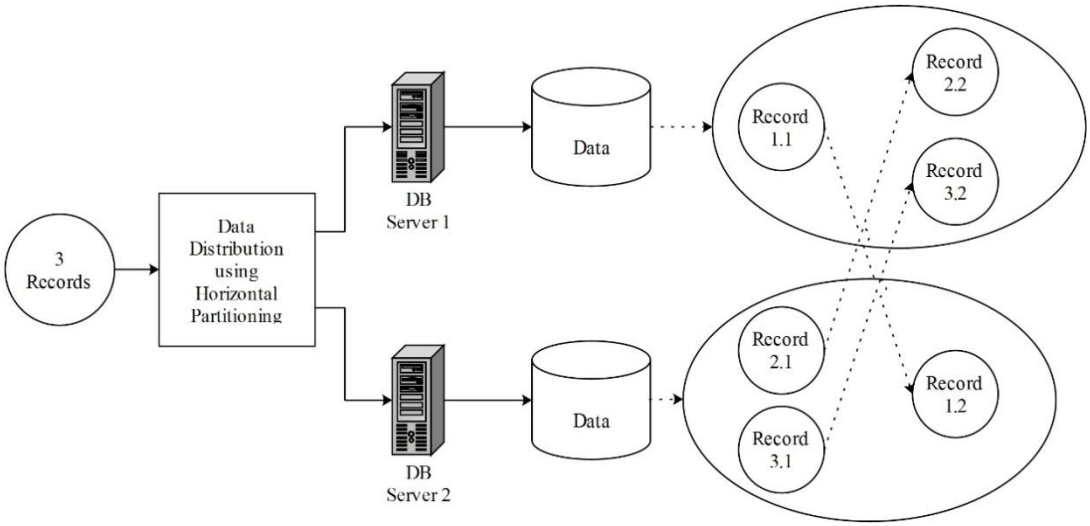

Fig 5: Equalization

General Method for N number of databases

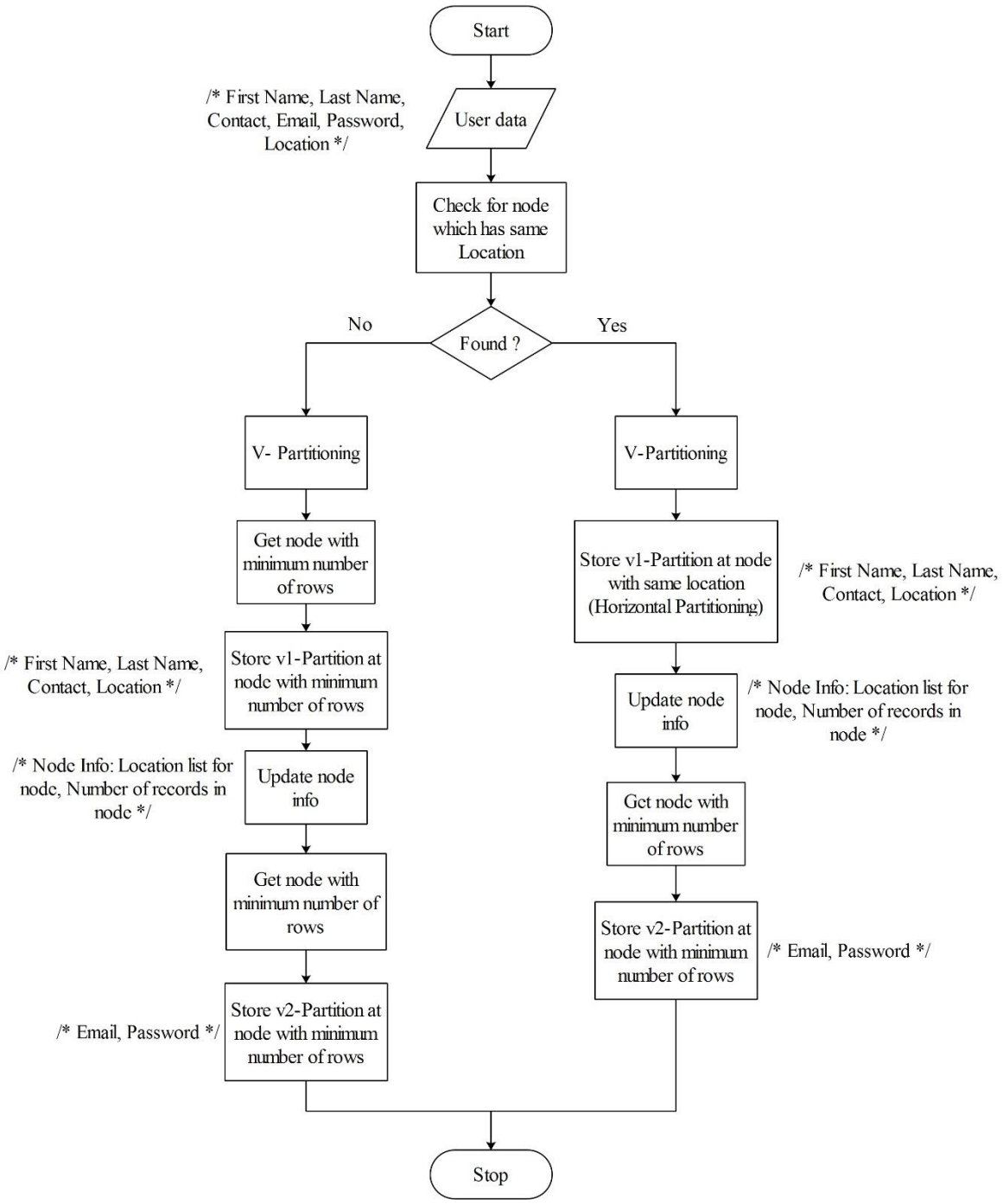

Fig 6: General Method

As shown in Fig 6, first the availability of user location is checked into node information, if it found then the data is partitioned vertically and one part of partition stored into the node having records with same location which is nothing but horizontal partitioning of data and other part of the partitioned data stored at the node with minimum number of records (rows). After storing first partition the node info gets updated to determine the node with minimum number of records later time. If node with same location not found in node info means if the arriving record is with totally new location then first it gets vertically partitioned and the first partition gets inserted into the node with minimum number of records and the node info gets updated and the second partition gets stored into the node with minimum number of record at that time. Which node has which location and how many records the node contains, that information is stored in node info table which gets updated whenever a partitioned record gets inserted in any node. As any real time application which interact

with users have two main modules that are Signup and Login. During signup module the data distribution takes place. Since only user email and password has to be checked and other data is not necessary during login module, vertical partition splits the data inn such a way that email and password attributes treated as one vertical partition and other data treated as another vertical partition. This approach of dividing login and user data reduces time of data retrieval and increase the performance by balancing the workload. Specialty of this approach is the user data and login data of a same user can not be in the same database which indirectly improves the security.

IV. CONCLUSION

Note: Insertion of a record actually insert two records in two different databases which are nothing but two vertical partitions of main record.

If the number of databases is an odd number 1, 3, 5 …. n then every n insertion gives equalized distribution of data.

Equalization factor E for Odd Number of Nodes		
Number of Nodes (Databases)(n)	Equalized Factor E (n)	Description
1	E = 1	Equalized data per 1 transaction
3	E = 3	Equalized data per 3 transaction
5	E = 5	Equalized data per 5 transaction
7	E = 7	Equalized data per 7 transaction
9	E = 9	Equalized data per 9 transaction
n	E = n	Equalized data per n transaction

If the number of databases is an even number 2, 4, 6 …. n then every n/2 insertion gives equalized distribution of data.

Equalization factor E for Even Number of Nodes		
Number of Nodes (Databases)(n)	Equalized Factor E (n/2)	Description
2	E = 2/2 = 1	Equalized data per 1 transaction
4	E = 4/ 2 = 2	Equalized data per 2 transaction
6	E = 6/2 = 3	Equalized data per 3 transaction
8	E = 8/2 = 4	Equalized data per 4 transaction
10	E = 10/2 = 5	Equalized data per 5 transaction
n	E = n/2	Equalized data per n/2 transaction

Above readings shows that it is more likely to have even number for this approach as it takes more transactions for fully equalized distributed data if the number of nodes are in odd number but it takes half number of transactions if the number of nodes are in equal number.

Almost equal distribution can be achieved using this approach as there are some exceptional case that records with a particular location increase continuously which leads to unequaled distribution. Probability of this exceptional case can be reduced by taking the smallest non unique attribute for horizontal distribution. For example in location based horizontal distribution it would be appropriate to take city or state as a parameter to compare and not country or continent. Doing this we are actually reducing the probability that more than one user can be from one location. The probability that more than one users can be from same location reduced with the level of location reduced in the manner of Continent -> Country -> State -> City.

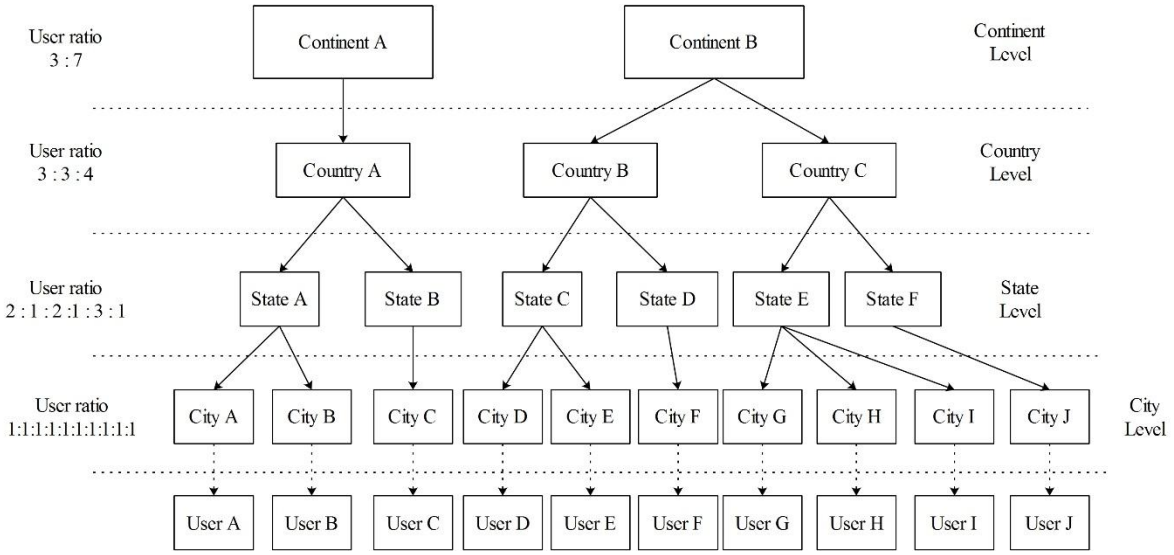

Fig 7: Probability of users from same location

As shown in Fig 7 For example, 10 users can be from same country which leads to unequaled distribution but there are more probability that those 10 users might be from different cities of the same country which increase the probability of more equalized data distribution.

Using this approach however data can be distributed based on location but also there are more chances that all the databases having same number of records which helps to reduce workload as all nodes can share the workload equally.

REFERENCES

[1]. "CLOUD DATABASE DATABASE AS A SERVICE", Waleed Al Shehri , Department of Computing, Macquarie University Sydney, NSW 2109, Australia. International Journal of Database Management Systems (IJDMS) Vol.5, No.2, April 2013.

[2]. "A Survey Paper on Database Partitioning", Dipmala T. Salunke, Girish P. Potdar, International Journal of Advanced Research in Computer Science & Technology (IJARCST 2014)