Analysis of Multiple Account Holders in Various Banks Using HADOOP Technology

Abinaya U¹, Divyabharathi V², Pavithra N³, Geetha Rani S⁴
UG Scholar¹,²,³, Assistant Professor⁴
Department of Information Technology
Dhanalakshmi College of Engineering, India

Abstract:
In this paper, User’s banking data is partitioned into multiple Tuples and stored in different sets of Database. We have designed an Application to track multiple accounts maintained in different banks by the same user and who have transacted more than 50000 per day using big data now a day’s pan card number is very essential for banking sector. For tracking the users unique ID (pan card) is used.

Keywords: Balanced partition algorithm, big data, Range aggregate queries, Map reduce technique.

I. INTRODUCTION:
Big data is a broad term for data sets so large or complex that traditional data processing applications are inadequate. Challenges include analysis, capture, duration, search, sharing, storage, transfer, visualization, and querying and information privacy. Big data analysis can provide various social aspects and preference of every individuals. This give new opportunity to rise fundamental questions about the complex world[1][2][3]. For example, Preis et al[2] analyzed the massive behavioral data sets related to and yield a profit higher than that of a random investment strategy, to build an efficient investment strategy. Choi and Varian[3] presented the sketches to forecast the economic indicators such as automobile sale, social unemployment and travelling. Now a days it is very important to provide efficient and effective methods and tools for analyzing the big data. There is an application example for big data is DIDS. The anomaly activities and strange patterns on the network level are monitored and reported by DIDS. DIDS detect the anomalies using statistics information of summarizing the results from diverse sensors to detect the attacks. This scenario motivates the range aggregate query problem[4] within the given query range. RAQ is the most important tool for online suggestion, decision management, and trend estimation. It is very difficult to obtain the range aggregate queries results in big data environment. Big data involves increasingly large amount of data, and the tuples stored in different files or blocks. On other hand, real time system aim to provide relevant results within the second on massive data analysis [5]. To boost the performance of range aggregate queries Prefix-sum Cube (PC) [4][6] is used in OLAP. The numerical values are sorted and any range aggregate query on data can be answered in constant time. If any new tuple is added into the cube, it has to be recalculate the prefix sum. Online aggregation is important approach to speed up the range aggregate queries. OLA provide estimated results while background process is still running. The accuracy is improved in further stages. The sampling and histogram approaches have been utilized in database to support the selectivity estimation.

In this paper, users banking data is partitioned into multiple tuples and stored in different sets of database. We have designed an application to track the multiple accounts maintained in different banks of same user and who have transact more than 50000 per day using big data. Now a day’s pan card number is very essential for banking sector. For tracking the users unique ID (pan card) is used.

II. METHODOLOGY AND MATERIALS
In this paper, we propose users banking data is partitioned into multiple tuples and stored in different sets of database. We have designed an application to track the multiple accounts maintained in different banks of same user and who have transact more than 50000 per day using big data. For tracking the users unique ID (pan card) is used. Now a days pancard number is very essential for banking sector.

1. USER ACCOUNT CREATION
Here first the User create an account and they are allowed to access the Network. Once the User creates an account, they are login to their account and request to the Service Provider. Based on the User’s request, the Service Provider will respond to them. The Service Provider database contains all the user information. In this paper, we will design the User Interface Frame to Communicate with the Server over Network Coding using the programming Languages like Java. By sending the request to Server Provider, the User can access the demanded data if they authenticated by the Service Provider.
Algorithm 1. Partitioning(R, VP)
Input: (R, VP);
R: an input record;
VP: the partition vector set.
Output: PID;
PID: a partition identifier for partition p.
1: Parse the input record R into different column-families
By the defined schema;
2: Compute the GID with its value from aggregation
column
By algorithm 2;
3: Get the partition vector Vpi from VP with the GID, and
Let Vpi \( \frac{1}{4} \text{GID}; Vr \gt ;
4: Set target partition identifier,
PID \( \frac{1}{4} \text{GID}; \text{random} \frac{1}{2} \); Vpi: Vr _ >;
5: Build the sample in partition PID, such as:
CounterPID counterPID \( \text{r} 1\); 
==counterPID is the number of record;
SumPID sumPID \( \text{r} N\);
//N is value of aggregation attribute from R;
Sample ID sum k; l; m; r=counterPID;
6: RID Hash\( \text{PID}; \text{counterPIDP};
//RID is the unique record identifier for R;
7: Send R to partition PID;
8: return PID.

1.1. RBI SERVER
Bank Service Provider will contain information about the
user in their Data Storage. Also the Bank Service provider
will maintain the all the User information to authenticate
when they want to login into their account. The User
information will be stored in the Database of the Bank
Service Provider. To communicate with the Client and the
other modules of the Company server, the Bank Server will
establish connection between them. For this Purpose we are
going to create a User Interface Frame.

Algorithm 2. FastRAQuering (Q)
Input: Q;
Q: select sum (AggColumn) other Colname where
li1<ColNamei<li2 opr lj1<ColNamej<lj2.
Output: S;
S: range-aggregate query result.
1: Deliver the request Q to all partitions;
2: for each partiioni in partitions do
3: Compute the cardinality estimator of range li1 <
ColNamei < li2 from the local histogram, and
let CEi be the estimator of the ith dimensions;
4: Compute the cardinality estimator of range lj1 <
ColNamej < lj2 from the local histogram, and let
CEj be the estimator of the jth dimensions;
5: Merge the estimators CEi and CEj by the logical
Operator Opr, and compute the merged cardinality
Estimator CEmerged;
6: Counti _h\( CEmergedP;\)
==_h is a function of cardinality estimation.
7: Compute the sample for AggColumn, and let
Sample i be the sample;
8: SUMi Counti _ Samplei;
==SUMi is a local range-aggregate query result;
9: end for
10: Set the approximate answering of FastRAQ as S. Let
S PM
IV. CONCLUSION
In this paper, we propose FastRAQ—a new estimated answering tactic that acquires accurate estimations quickly for range-aggregate queries in big data environments. We believe that FastRAQ provides a good starting point for developing real-time answering methods for big data analysis.

V. FUTURE ENHANCEMENT
First, FastRAQ can solve the 1:N format range-aggregate queries problem. Second, FastRAQ is now running in homogeneous environments; we will further explore how FastRAQ can be applied in heterogeneous environments or even as a tool to boost the performance of data analysis in database. Third, we can provide more security and verify the originality of unique ID.

V. REFERENCES


