International Journal of Database Theory and Application
Vol.9, No.10 (2016), pp.363-378
http://dx.doi.org/10.14257/ijdta.2016.9.10.31

Query Evaluation on Probabilistic Databases Using Indexing and MapReduce

Kavita K. Beldar[1], M. D. Gayakwad[1], Debnath Bhattacharyya[2] and Hye-jin Kim[3]

[1]*Department of Information Technology,*
Bharati Vidyapeeth Deemed University College of Engineering,
Pune, India
{kavitakbeldar,gayakwad.milind}@gmail.com
[2]*Department of Computer Science and Engineering,*
Vignan's Institute of Information Technology,
Visakhapatnam-530049, India
debnathb@gmail.com
[3]*Sungshin Women's University,*
2, Bomun-ro 34da-gil, Seongbuk-gu,
Seoul, Korea
hyejinaa@daum.net
(Corresponding Author)

Abstract

Entity resolution technique is used for recognize the duplicate tuples which signify similar real world entities. Existing resolution technique is unable to solve the problems of higher level of heterogeneity and additional continual data alteration. Working on this type of database, there is necessitated to enumerate the integrity of data. The new approach is introduced here on probabilistic databases by unmerged duplicates for processing complex queries. This is achieved by using probabilistic databases. For competent access toward entity resolution data over a large collection of possible resolution worlds, new indexing technique is presented here. Also, a computation of query processing is reduced by using indexing structure. The focus is on set similarity relation on very big probabilistic database by using MapReduce technique. MapReduce is a popular paradigm that can process large volume data more efficiently. In this paper, different approaches proposed using MapReduce to deal with this task: 1. merge data set with MapReduce and merge data set without MapReduce, 2. Merge data set with MapReduce using Hadoop. This approaches implemented on windows and Hadoop framework and performed compressing experiments to their performances. Also the speedup ratio for both is tested.

Keywords*: Probabilistic Databases, Unmerged Duplicates, MapReduce algorithm, indexing*

1. Introduction

Now a day in IT based market a database takes a part in vital role. Some manufacturing and organizations depends on the accurateness of databases take out the operations. Hence, the superiority of the information stores up in database may have important cost proposition toward the system that impart on information to utility and carry out business. When the system is completely error free then it is said to be having clean data, the creation of a compressing inspection of such data consists of relating in the relational terms. Data frequently lack a distinctive overall identifier so as to would allow such functions. Many of the real world databases hold data whose accuracy is unsure. Work on

ISSN: 2005-4270 IJDTA

to this type of data there is necessitating computing the veracity of the data. Unsure databases in which the probable worlds have related possibilities are called a probabilistic database.

A probabilistic database is a general framework for managing imprecision and uncertainty in data. Building a general framework to manage uncertainty in data is a challenging goal because critical data-management tasks, such as query processing, are theoretically and practically more expensive than in traditional database systems. Handling queries efficiently and understanding huge set of unsure data is the most important test in probabilistic databases. This thesis demonstrates that it is possible to effectively manage large, imprecise databases using a generic approach based on probability theory. The logic of probabilistic merges hypothesis to handle ambiguity with the capability of deductive judgment to develop the structure. For the analysis purpose the probabilistic record linkage combines numerous databases in to single widespread database. Here two diagnostic queries that is aggregation and top-k queries are used. New indexing technique and dealing out algorithm presented which make possible effective query time for accessing the resolution information with merges and probabilities.

A MapReduce algorithm is developed for performing query efficiently on large probabilistic databases. Here, the performance of time on different scenarios such that on 52,000 and 2Lakh database size is checked. Queries on reduced data set merge with MapReduce and without MapReduce on 52,000 as well as 2lakh size database are performed. Same queries are run on Hadoop framework and compared the performance time on these different platforms. This technique throughout a wide-ranging evaluation by real-life databases of online shopping records of buyer and their orders are corroborated.

1.1. Key Technical Challenges and Goals

The key challenge is addressed in this paper is performance. Performance is challenging in probabilistic databases: in contrast to standard SQL processing, which is always theoretically easy, evaluating queries on a probabilistic database is theoretically hard (#P-hard). While hardness is a key challenge, it is also a golden opportunity: Optimization techniques from the database management literature can actually become more effective for probabilistic databases than they were for standard relational databases. For example, MapReduce using inverted indexing and materialized views are an effective technique in relational optimizers, which allows the system to recomputed information and then use this recomputed information to optimize queries. In probabilistic databases, this technique is even more effective. The following table shows the list of notations.

Table 1. List of Notations

Notations	Descriptions
R	Records in a table
E	Set of entities
L	Linkages
F	Factors
P	Probability
D	Probabilistic database
T	Deterministic relational tables
P^l	Linkage probability function
I	Indexing structure
(k,v)	Key and value

2. Motivating Example

Consider an employee's online shopping database where the data is integrated from place of correlated systems. Table 2 shows the Buyer table which is a small example of probabilistic database. This table contains the similar instances which depict the identical real world items. Table 3 shows the Order table, which are the orders of every customer. By using jaccard similarity technique on Buyer table, the result in associated probabilities that symbolize the probable matches in between the instances. The records r1,r6 and record r1,r9 describes the identical real world items, with the probabilities of 0.5 and 0.6. Accommodating this linkage creates a new entity $e_{1,6}$ and $e_{1,9}$, and it replaces to r1, r6 and r1, r9.The system integrates merge function for denoting representation of data of Last entity. Here the highest value of the year is considered. Therefore, after merging record r1, r6 and r1, r9, <$e_{1,6}$ "Matk", "Mandell", "F", 22-11-1991, USA, 2006 > and < $e_{1,9,}$ "Matk", "Mandell", "F,20-04-1990, PAK, 2014>.

Following table shows the probabilistic records of buyer and order customer details and their associated probabilities shown in Table 4.

Table 2. Buyer Table

Sr.no	Emp_id	First name	Last name	Gender	Birth date	Location	Year
r1	10001	Matk	Mandell	F	22-11-1991	USA	2006
r2	10002	Khalid	Mandell	M	15-12-1992	DUB	2005
r3	10003	Shai	Mandell	F	10-10-1956	IND	2009
r4	10004	Vidhr	Mandell	M	08-08-1991	JAP	2001
r5	10005	Leah	Mandell	F	07-11-1989	DUB	2010
r6	10006	Matk	Mandell	F	15-08-1998	ENG	2005
r7	10007	Leah	Mandell	F	07-11-1989	IND	2013
r8	10008	Khalid	Mandell	M	06-03-1992	DUB	2005
r9	10009	Matk	Mandell	F	20-04-1990	PAK	2014
r10	10010	Shai	Mandell	M	03-05-1991	IND	2003

Following Table 3 shows the order table of the buyer persons. It shows that how many items selected by each user and its total amount.

Table 3. Order Table

emp_id	emp_no	Items	Amount
r1	10001	2	127
r2	10002	1	90
r3	10003	5	120
r4	10004	3	113
r5	10005	2	150
r6	10006	1	100
r7	10007	3	250
r8	10008	1	130
r9	10009	2	245
r10	10010	2	380
r9	10009	1	120
r6	10006	2	200

2.1. Calculating Probability Using Jaccard Similarity

Here, probability is calculated using the jaccard similarity technique.Consider two sets x= {Matk,Mandell,F,22-11-1991, USA,2006} and y= {Matk, Mandell, F, 15-08-1998, ENG, 2005}. How similar are x and y? The jaccard similarity is defined

$$\text{Similarity}(X,Y) = JS(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} = \frac{|\{Matk, Mandell, F\}|}{|\{Matk, Mandell, F, 22-11-1991, USA, 2006, 15-08-1998, ENG\ 2005\}|} \quad (1)$$

$$= \frac{3}{6}$$

$$= 0.5$$

The attributes of first row with the attributes second row are compared and checks how many attributes of first row matches with the second row. In above buyer table record r1 compares with the record r6 and finds how many attributes are similar between r1 and r6.Out of six attributes three attributes matches that is 3/6=0, 5. So the probability is 0.5.

Table 4. Probability Table

Emp_no	Emp_id 1	Emp_id 2	Probability *P*
1	10001	10006	0.5
2	10001	10009	0.6
3	10002	10008	0.8
4	10003	10010	0.5
5	10004	10004	1
6	10005	10007	0.3

3. Review Area

In this section, the related works that include the overview of the entity linkages with uncertainty and ranking queries on probabilistic databases, MapReduce techniques, and Top-k ranking queries is introduced.

In 2014 Youzhong Ma, XiaofengMeng, proposed solution for set similarity joins using MapReduce algorithm on large probabilistic databases [1]. The authors projected two different approaches using MapReduce technique for performing these tasks: first map side pruning by Hadoop join and second reduce side pruning by Hadoop join. First uses which uses the sum of the existence probability to filter out the probabilistic sets directly at the map task side which have no any chance to be similar with any other probabilistic set. Second uses probability sum based pruning principles and probability upper bound based pruning principles to reduce the candidate pairs at reduce task side, it can save the comparison cost. Based on this approaches author proposed a hybrid solution that employs both map-side and reduce-side pruning methods.

In 2013 Maximilian dllya, Martin Theobald proposed approach for computing confidence bunds in support of top-k ranked queries in contingence databases [2]. This approach does not need to view very first for all query answers. Main purpose of this work is to identify effectively top-k with the majority possible results, with highest and lowest threshold on behalf of their possibilities, with no need to calculate the accurate possibilities. To achieve this goal author first consider combined data and assurance calculations on behalf of queries which do not consent to secure query procedure and so do not concur for efficient incremental query implementation. This method provides upper and lower bounds on the minor possibility of every single query results. This

approach permits for plug up for unusual schedulers. This plugging aims toward choose the sub goal for every query progression step used for top-k pruning.

In 2012 Dan Olteanu, Hongkai Wen proposed solution for issues in contingence databases for ranking query answers [3]. Author introduced a technique to make top-k exact answers for conjunctive queries. Statically obtained share plans and deterministic approximation used in this approach which revive upper and lower bounds probable query answers. To implement this technique SPROUT query engine has been use of a MayBMS that is the part of contingence databases. This mechanism is being based on additional near probability computation for result with common factors, probability computation by event decomposition, decomposition of event with common factors, ranking algorithms.

In 2011 Ming Hua, Jian Pei analyzed difficulty of answering contingence threshold top-k queries on unsure information that collects uncertain record [4]. This technique using a possibility of slightest p toward within the top-k catalog in which p is a customer defined possibility threshold. Author represented a perfect algorithm, based on Poisson estimation technique related algorithm and rapid sampling algorithm. Real and static data sets used to check ability of eventually threshold top-k queries and the capability of this method. Challenges in this approach are addressed below: What does an eventuality of top-k query mean? And how can a contingence threshold top-k query work efficiently? For this author apply efficient algorithm to avoid unfolding of all possible words. It shows the proper top-k probability for every tuple by using the sorted list of all tuples only once.

In 2010 Ekaterini Ioannou, Wolfgang Nejdl described a framework for entity interconnection with unsureness [5]. Here available interconnection has been used with data and its trust parameter instead of using the interconnection information to combine a-priori structures. For contingence interconnection new contingence query evaluation method has been used. This idea introduces below details: (i) it executes combinations at run time for not only on existing interconnection but also for query given; (ii) it permits solutions that includes structures, but generated as a result also for interconnections; and (iii) support and query conditions evaluation on integrated structures. This offers a functionality not currently supported by any traditional contingence databases.

3.1. Objectives

Effectively handling complex analytical queries in excess of probabilistic database through unmerged duplication and handling the linkage factors with large size which requires additional time for processing.

4. Proposed System

Figure 1 shows the proposed system architecture. The proposed system uses indexing algorithm and MapReduce algorithm for performing complex queries on massive amount of data which is a probabilistic database.

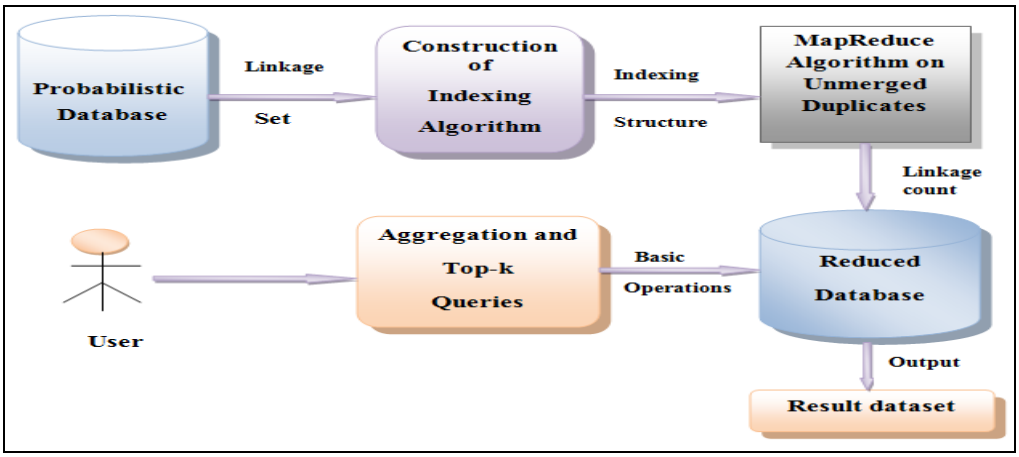

Figure 1. System Architecture Using MapReduce Algorithm

Modules of architecture are explained below in detail.

4.1. Probabilistic Database

A probabilistic database is nothing but an unsure data set in which the probable worlds contain associated possibilities. The values of some attributes of databases is unsure and recognized only with several possibilities is also called a probabilistic database. Applications in different areas like data integration, information extraction, data cleaning which generate huge amount of unsure data which is reproduced and processed by using probabilistic database. Above buyer Table 2 shows the example of probabilistic database. Here in buyer Table 2 there are duplicate records having different probability that is the possibility of occurrences. The last name Mandell is duplicate record for overall table. Handling queries on such databases is difficult task. Here indexing and MapReduce algorithms are performed on this database for efficient access of queries.

Definition 1:A probabilistic database P^d by duplicate records which are not merged $<t_1, t_2 \ldots t_n, R, L, P'>$, R is a table of duplicate records, $< t_1, t_2 \ldots t_n>$ are relational tables, L is the linkages in excess of the instance in table R and p^1 is the linkage function for probability calculation that is $p^1|L \rightarrow [0,1]$ |.

Unsure databases without merging duplicates records (definition 1) hold the linkage set L and the linkage function for probability calculation $p^{1,}$ which gives possibilities toward the linkages L. Hence it means that every linkage present with the possibility given by p^1, and do not with possibility $1\text{-}p^1$.The databases are taken from the online shopping customer's database which stores the information from various system resources. Here, databases with two different sizes are considered that are 52,000 which is given in existing system and another is 2,00,000 size database which is proposed here. Both the databases using indexing and MapReduce technique are filtered for the purpose of efficient query evaluation.

4.2. Construction of Indexing Structure

Indexing structure forms the base of competent processing of the supported query varieties. The main purpose of constructing an indexing algorithm is for decreasing the complexity of calculations required at the time of query processing. It is accomplished with the indexing structure which provides effective access to the all information encoded throughout the linkages. It also allows easy creation of probable worlds with fast retrieval of their possibilities. This indexing structure is constructed on employee tables. It provides random lookups and easy accessed ordered records in large databases. The indexing is done on last name column, so the last name is quickly accessed in large probabilistic databases. Here are some queries mentioned below:

- Index is created on "Last_name" column from the Buyer table

CREATE INDEX "first index"
On Buyer (Last_name)

- Index is created on a combination of two columns

CREATE INDEX "second index"
ON Buyer (Last Name, First Name)

The extended indexing structure handles the factors with large size that is the factors which contain a huge amount of linkages which requires additional time for processing. The indexing structure should contain data with respect to the factor length, which handles the time on behalf of processing minimum factors with respect to the processing for maximum factors. While constructing the algorithm of indexing structure, first it generates the factors f1 and f2.This is done by identifying connected components in the available group of linkages. Put all linkage in a factor, which are the instances of these linkages are simply submitted by linkages of identical factors. Two different factors *i.e*. male and female are created on buyer table. Dividing these all linkages hooked on two separate factors represents to just necessitate regarding to as generating entities via combining the instances which are participated in linkage factors f1 and f2.Separating linkages hooked on different factors has the optimistic results in the effectiveness. There is need to work out on linkage subsets of minimum sizes. After generating factors on a given group of linkages, processing is done on the linkages for every factor. For calculating the possibility ranges for all the probable worlds which are created using linkage of the particular factor. Generating the list of all instances participated in linkage a factor which considers distinct unification of the linkage and its instances and loads these linkage combinations which are generates from the given factors. The range of probability for factor f_i has a least value $\underline{v}f_i^p$ and highest value $\overline{v}f_i^p$.Suppose the probability p less than 0.3 then leaving out of these linkages that is two occurrences did not merge and it take place with possibility 1-p. The highest probability bound is a production each linkage probabilities and the lowest possibility bounds are the production of complements of the linkage possibilities. The range for factor $f_i$is

$$\underline{v}f_i^p = \prod_{l_{ri,rj} \in f_i} [1 - pl\,(l_{ri,rj})] \text{ and}$$

$$\overline{v}f_i^p = \prod_{l_{ri,rj} \in f_i} [pl(l_{ri,rj})]$$

Here how many numbers of linkages in factor f_i that number of linkage combinations are created. The number of linkage groups increases exponentially through the length of factors.

4.3. MapReduce

The MapReduce model is widely used for parallel programs and working on large amount of data in parallel. The MapReduce programming is a worldview for performing operations on big databases in distributed situations. The map function divides the documents into single words and for every word in the document it generates<key, value> pairs. For every words in the documents uses the function map (name, document)emit (word, count).The MapReduce technique is performed on first name of buyer table. Here it merges all first names which having the same first names into a separate linkage or link and gives it's all detail information into a same link. Because of this process the searching process time increases and run queries faster than without MapReduce technique.

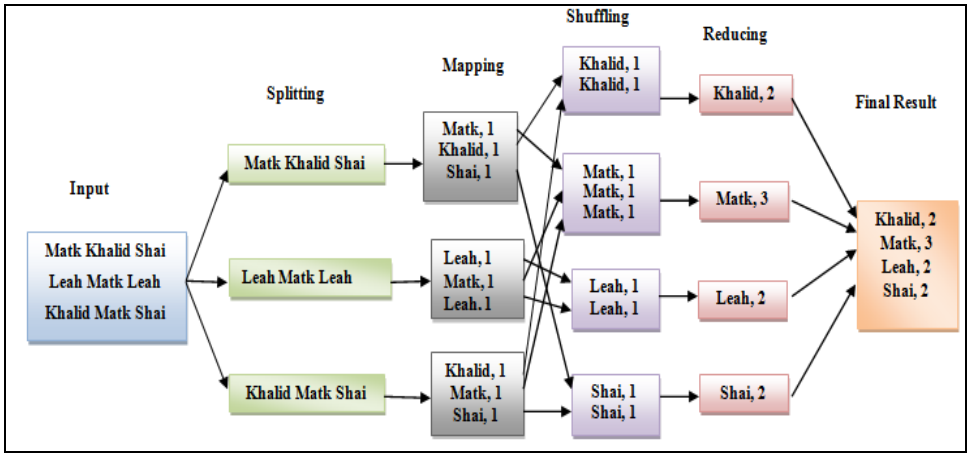

Figure 2. MapReduce Process

Figure 2 mentioned above shows the MapReduce framework.

1. **Input phase**: In this phase each record is translated in an input file using record reader and sends the converted data to the mapper function in the form of key and value pairs.
2. **Splitting:** splitting step takes input data set from source and divide into smaller sub-data sets. Here the all first names are splits into smaller subsets.
3. **Mapping**: Mapper takes a series of the key-value pairs and works every one of them to produce one or zero or more than zero key-value pairs. It maps the first names and converted into a format of<key, value> that is<Khalid,1><Matk, 1><Shai, 1>.
4. **Shuffling**: It performs two tasks that are merged and sort. Merging step combines all key-value pairs which have same keys. This step returns <key, List<value>> that is in above example<Khalid, list <2>>.The sorting step takes input from merging step and sorts all key-value pairs by using keys. The output of this step is sorted key-value pairs.
5. **Reducer**: The reducer function takes group of key-value paired data as input and run reducer function on every of them. The data is merged, cleaned, aggregated in different ways; it necessitates broad range for processing. After execution is completed it gives the result zero otherwise more than one key value pairs.
6. **Final result**: Here it counts the total number of occurrences of each first name on particular tables. For example, the above figure shows the final results with its total count, which is shown in above figure**.**

4.3.1. MapReduce Algorithm:

Input: Probabilistic database D, Indexing Structure I, Views v
Output: list (first_name, count)

1: Class mapper
2: method map (first_name f, integer i)
3: Emit (first_name f, pair (i, 1))
4: Class reducer
5: Method reducer (first_name f, pairs [(f1, c1) ;(f2, c2) ;::::])
6: sum ←0
7: count ←0
8: for all pairs (f,c) 2pairs [(f1,c1) ;(f2,c2);:::]do
9: sum=sum+c
10: count=count +c
11: iavg←sum=count // average of integer count
12: Emit (first_name f, integer iavg) // list of first_name with its total count

Map and Reduce are the main important tasks of a MapReduce algorithm. The map method takes input as a key and value pairs, the key is a first_name and value is its count in particular document. Here it gives the output is<Matk, 1><Khalid, 1><Shai, 1> shown in Figure 2. The reducer methods take input as a result of map function and merge them into another smaller group of tuples like (<f1, c1>, <f2, c2>…)herethe<f1, c1>isa first_name one and c1 is its count 1. Then it sorts alphabetically all first_name with its total count. The final result of all same first_name mapped together with its average counts is got, that is<Khalid, 2>, <Matk, 3>, <Shai, 2><Leah, 2> shown in Figure 2.

4.3.2. Probability Computation on the Set Level

Let $P^D$is a probabilistic database and R is a record set *i.e.* $R= \{r_1, r_2, r_3, r_4 \ldots r_{10}\}$, E isan entity set *i.e.* $E= \{e_1, e_2 \ldots, e_n\}$, f is a factors for male and female $f= \{f_i, fi\}$. P is the probability and L is set of linkages *i.e.* $L= \{l_{r1, r2}, l_{r3, r7, lr3, r10}, lr_{7,r10}, l_{r2,r5}, l_{r2,r9}, l_{r5,r9}\}$,

The probabilistic database P^D always contains the number of probabilistic sets, denoted as $P_{ri.}$ Every P_{ri} can be represented as r_{ni} set of instances r_{ni1}, r_{ni2}, r_{ni3}….. And r_{nin} .All the set instances P_{rik} of a probabilistic set P_{ri} are mutually exclusive. Each instance of P_{rik} is associated with an existence probability P_{rik}.$P \in (0; 1]$, where $\sum_{k=1}^{r_{ni}} Prik.P \leq 1$.

The probabilistic database P^D gives two probabilistic set databases that are buyer B^P and order O^P, a similarity threshold $\delta \epsilon (0,1]$ and probabilistic threshold $\beta \epsilon (0,1]$, a probabilistic set similarity join is to find all the pairs (b_i, o_j) from B^P and O^P, with probability larger than or equivalent to threshold δ, that is:

$$P^D = \{< b_i, o_j > \mid b_i \in B^P \text{ and } o_j \in O^P \text{and } P_{ri} \{\text{sim } (b_i, o_j) \geq \delta \} \geq \beta\} \quad (2)$$

By using the equation (1) shown above, the probability is calculated

$$\text{JS (A, B)} = \frac{|A \cap B|}{|A \cup B|}$$

$$P_{ri} \{\text{sim } (b_i, o_j) \geq \delta\} = \sum_{\forall b' \in bi} b' \sum_{\forall o' \in oj} o', b'.P.o'.P.X \text{ (sim } (b', o') \geq \delta) \quad (3)$$

Where b' and o' are the instances of $b_i$and o_j respectively. X (sim $(b', o') \geq \delta$) is a Boolean function.

$$X \text{ (sim } (b', o') \geq \delta) = \begin{cases} 1 \; if sim(b', o') \geq \delta istrue \\ 0 \qquad otherwise \end{cases} \quad (4)$$

The following table shows the result of mapper and reducer.

Query 1: All records using MapReduce

Table 5. MapReduce Results

First Name	Mapper					Reducer
	Emp_id	Last name	Birth date	location	Year	
Matk	10001	Mandell	22-11-1991	USA	2006	3
	10006	Mandell	15-08-1998	ENG	2005	
	10009	Mandell	20-04-1990	PAK	2014	
Khalid	10002	Mandell	15-12-1992	DUB	2005	2
	10008	Mandell	9-3-1992	DUB	2005	
Shai	10003	Mandell	10-10-1956	IND	2009	2
	10010	Mandell	3-05-1991	IND	2003	
Leah	10005	Mandell	7-11-1989	DUB	2010	2
	10007	Mandell	7-11-1989	IND	2013	
Vidhr	10004	Mandell	8-8-1991	JAP	2001	1

4.4. Reduced Data Set

The reduced data set contains the data which is filtered by the processing of indexing structure and MapReduce algorithm using inverted indexing. The database records are divided into two separate factors that are male and female. The above Table 2 shows the reduced data set example. That is all same first name records are merged into together in a same link. The searching is done faster than the original probabilistic database. The reduced data set contains duplicate instances which are merged together without deleting them. So the performance time increases automatically.

4.5. Basic Operations

First select the any last name from the large amount of database and then perform basic operations and different queries.

Query 2: select *from buyer where last name=" Mandell"

It displays all the records whose last name is Mandell. The result of query 1 is same as Table 1 above.

4.5.1. Retrieving by Groups

Here the group of locations are retrieved the by identifying the instances for every location which satisfy the conditions of queries. Then those instances are grouped together. The groups of same locations are G1= {r3, r7, r10} for "IND", G2= {r2, r5 r9} for "DUB", G3= {r4} for "JAP", G4= {r6} for "ENG", G5= {r9} for "PAK". Aggregation queries are used for group by clause. For example:

Select column1, column2
From buyer
where location =" IND"
Group by column1, column2

Query 3: SELECT * FROM buyer **Where location =" IND''**

Table 6. Results of IND Locations

10003	Shai	Mandell	F	10-10-1956	IND	2009
10007	Leah	Mandell	F	07-11-1989	IND	2013
10010	Shai	Mandell	M	03-05-1991	IND	2003

Query 4: SELECT * FROM buyer **Where location =" IND"**

Table 7. Results of DUB Locations

10002	Khalid	Mandell	M	15-12-1992	DUB	2005
10005	Leah	Mandell	F	07-11-1989	DUB	2010
10008	Khalid	Mandell	M	06-03-1992	DUB	2005

4.5.2. Retrieving of Factors

Using the indexing structure, the factors are created that are male and female. The indexing allows retrieving the records which are grouped together by their genders... For example group$G1= \{l_{r3,r7}, l_{r3,r10}, l_{r7,r10}\}$, $G2=\{l_{r2,r5}, l_{r2,r9}, l_{r5,r9}\}$ are the two different groups, which are grouped by sing their factors, that is record as male and record as female.

Query 5→SELECT *FROM Buyer where gender=" male"

Table 8. Result of All Male Records

First name	Last name	Gender	Birth date	Location	year
Khalid	Mandell	M	15-12-1992	DUB	2005
Vidhr	Mandell	M	08-08-1991	JAP	2001
Khalid	Mandell	M	06-03-1992	DUB	2005
Shai	Mandell	M	03-05-1991	IND	2003

Query 6→SELECT *FROM Buyer where gender=" female"

Table 9. Result of All Female Records

First name	Last name	Gender	Birth date	Location	year
Matk	Mandell	F	22-11-1991	USA	2006
Shai	Mandell	F	10-10-1956	IND	2009
Leah	Mandell	F	07-11-1989	DUB	2010
Matk	Mandell	F	15-08-1998	ENG	2005
Leah	Mandell	F	07-11-1989	IND	2013
Matk	Mandell	F	20-4-1990	PAK	2014

4.5.3. Top-k Query

Query for first 3 top records from the database. It gives the top 3 highest probability results.

Query 7→ SELECT Top 3* FROM buyer

Table 10. Results of Top-3 Records

Sr. no.	Emp_id 2	Emp_id 2	Probability
1	10001	10009	0.6
2	10002	10008	0.8
3	10003	10010	0.5

4.5.4. Merge Query

Here retrieving the merge results, compare the instance 1 to another instance and find their probability by considering the how many attributes of instance 1 matches to another instance.

Query 8: Retrieving all records of merge table

Table 11. Result of Merge

ID	Instance 1	Instance 2	Probability	Total Amount
I (10001,10006)	10001	10006	0.5	427
I (10001,10009)	10001	10009	0.6	372
I (10002,10008)	10002	10008	0.8	220
I (10003,10010)	10003	10010	0.5	500
I (10004,10004)	10004	10004	1	113
I (10005,10007)	10005	10007	0.3	250

5. Experiments and Results

5.1. Data Set

Data sets containing 2, 00,000and 50,000entries are considered. Each entry contains duplicate records as well as unique records. This data is generated using MySQL server by taking real data values of shopping mall. Time performance is tested on both databases performing same queries.

5.2. Methodology

In this data set map reduce algorithm on both databases that is 2, 00,000 and size 52,000 databases are performed. The performance is checked on windows and Hadoop framework and did the comparison of merge using MapReduce algorithm and merge without MapReduce algorithm. In this data set up to 2, 00,000 entries are performed on both MapReduce and without MapReduce in two different platforms that is Hadoop and windows xp/7. Performance is decreases whenever the database size increases more and more. An existing system uses indexing structure for performing query result fast. While the database size increases more and more the performance of indexing structure decreases.

Here, MapReduce algorithm is used to overcome the drawbacks of indexing structure. Mapping is done on first name that is all same first names are mapped together and merge them. So records are search quickly. Same queries on both databases that is 2, 00,000 and 50,000 using MapReduce algorithm and without MapReduce algorithm are executed. With MapReduce algorithm the processing time is less as compare to without MapReduce algorithm.

Following Table 12 shows the performance of queries on 50,000 and 2,00,000 databases using MapReduce algorithm and without MapReduce algorithm.The time performance is high using MapReduce algorithm as compare to without MapReduce algorithm. According to linkages size the time performance is varies between low and high.

5.3. Results

The following table shows the time performance of each query.Here , same queries are perofrmed on two different sizes of databases that is 52,000 and 2,000,00. Also did the comparision between them with respect to time that is first, database with size 52,000 merge with MapReduce and without MapReduce and second database with size 2,00,000 merge with MapReduce and without MapReduce. With the database size 2,00,000 the time performance is better than database size 52,000.

Table 12.Time Performance on Two Different Data Sets

Query number	52,000 (Times in milisecond)		2,00,000(Times in milisecond)	
	Without MapReduce	Merge with MapReduce	Without MapReduce	Merge with MapReduce
1	3387	2710	4277	3433
2	2449	2035	2870	1420
3	2480	1698	2824	1936
4	2475	1573	2340	1710
5	2450	1489	2309	2035
6	2793	1590	2808	1698
7	2449	2198	2450	2182
8	1681	1245	3247	1681

Figure 3 shows the graphical representation of Table 12. The graph shows the different performance level of each query in milliseconds. The time is in millisecond which shows on X axis and the size of data entries shows on Y axis.

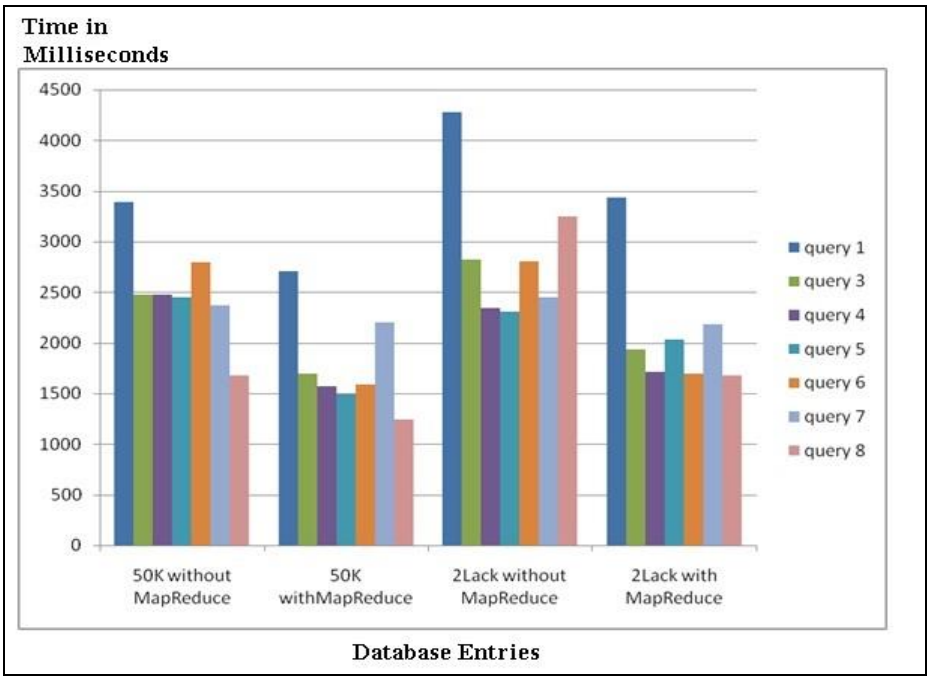

Figure 3. Processing Queries Time vs. Database Size

Figure 3 shows the comparison between time and database size. The time measured in milliseconds and database size is in thousands and lacks. The comparison is done between four parameters that is 50,000 with MapReduce 50,000 without MapReduce and 2,00,000 merge with MapReduce and 2,00,000 without MapReduce. On both databases the efficient performance is seen on merge with MapReduce method as compare to without MapReduce.

5.4 Contributions

1) Novel indexing structure is proposed that allow well-organized query instance right to use to the declaration information its combinations, possibilities.
2) New methods are provided here that make use of competent processing top-k and aggregation queries with no necessitate to turn up the probable worlds.
3) MapReduce algorithm is proposed that enable the mapping of similar data sets and reduce them. It also helps the balancing the processing time for small and large data sets.
4) These methods are validated throughout a wide spread development (using real life data sets).

6. Conclusions and Future Work

The problem of resolution throughout a general outline for processing difficult queries in excess of unmerged duplicates is addressed here. These approaches considerable probabilistic database which contains duplicate items, probable association between duplicate items and other related tables. First the indexing structure is introduced which provides effective access on the probable entity merges and its probabilities. It also reduces the complexity of query processing. Second the MapReduce algorithm is used for managing the large probabilistic database easily. MapReduce technique takes the set of input data and converts it into another reduced format of data with its list and total counts. Different approaches proposed here using the MapReduce that is merging with MapReduce and without MapReduce. The experimental evaluation is performed on real data set with two different sizes *i.e*. 52,000 and 2,00,000 respectively. A different technique for efficient processing of aggregation and top-k queries in excess of duplicate items is introduced here. Here assessment investigated the resolution and characteristic of queries for time processing, also verifies the approach is effective with high efficiency. In future this approach will try to find the solution for balancing the time for large linkage factor of attribute and for small linkage factor of attribute, when the database size is increases more than 2, 00,000.

References

[1] K. K. Beldar, M. D. Gayakwad, D. Bhattacharyya and T. H. Kim, "A Comparative Analysis on Contingence Structured Data Methodologies", International Journal of Software Engineering and its Application under ISSN, vol. 10, no.5, pp. 1738-9984.
[2] K. K. Beldar, M. D. Gayakwad and M. K. Beldar, "Optimizing Analytical Queries on Probabilistic Databases with Unmerged Duplicates Using MapReduce", IJIRCCE, vol. 4, no. 5, **(2016)**.
[3] Y. Ma and X. Meng, "Set similarity join on massive probabilistic data using MapReduce", 2014 Springer Science + Business Media New York, **(2013)**.
[4] M. Dylla, I. Miliaraki and M. Theobald, "Top-k Query Processing in Probabilistic Databases with Non-Materialized Views", University of Antwerp, **(2013)**.
[5] D. O. H. Wen, "Ranking Query Answers in Probabilistic Databases: Complexity and Efficient Algorithms", EPSRC EP/G069557/1 FRESNEL project, **(2012)**.
[6] M. Hua, J. Pei, W. Zhang and X. Lin, "Ranking Queries on Uncertain Data: A Probabilistic Threshold Approach", SIGMOD'08, **(2011)**.
[7] E. Ioannou, W. Nejdl and C. Niederee, "On the Fly Entity Aware Query Processing in the Presence of Linkage", 36th Inter. Conf. on Very Large Data Bases, **(2010)**.
[8] E. Ioannou and M. Garofalakis, "Query Analytics over Probabilistic Databases with Unmerged Duplicates", IEEE Trans. Knowledge Data Eng., vol. 27, no. 8, **(2015)**.
[9] N. Dalvi and D. Suciu, "Efficient Query Evaluation on Contingence Databases", Proceedings of the 30th VLDB Conference, Toronto, Canada, **(2004)**.
[10] M. A. Soliman, I. F. Ilyas and S. B. David, "Supporting ranking queries on uncertain and incomplete data", The VLDB Journal, vol. 19, **(2010)**, pp.477–501.
[11] N. Dalvi and D. Suciu, "Efficient Query Evaluation on Probabilistic Databases", USA Proceedings of the 30th VLDB Conference, Toronto, Canada, **(2004)**.
[12] N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," Proc. VLDB Endowment, vol. 16, no. 4, **(2007)**, pp. 523-544.

[13] R. Fink, L. Han, and D. Olteanu, "Aggregation in probabilistic databases via knowledge compilation," Proc. VLDB Endowment, vol. 5, no. 5, **(2012)**, pp. 490-501.
[14] T. Ge, "Join queries on uncertain data: Semantics and efficient processing," in Proc. Int. Conf. Data Eng., **(2011)**, pp. 697-708.
[15] D. Wang, M. Franklin, M. Garofalakis, and J. Hellerstein, "Querying probabilistic information extraction", Proc. VLDB Endowment, vol. 3, no. 1, **(2010)**, pp. 1057-1067.
[16] M. Akdere and U. Cetintemel, "Continuous probabilistic data association with constraints", Brown Univ., RI, USA, Tech. Rep. CS-11-02, http://cs.brown.edu/~makdere/ papers/vsn-tr.pdf, **(2011)**.