RESEARCH ARTICLE OPEN ACCESS

Amplification Misplaced Answers to Crowd DB Using Fuzzylogic and K-Means Clustering Algorithm

[1]Ms.Pavithra P., [2] Ms. Abarna N.

*1M.Phil Research Scholar, PG & Research Department of Computer Science &Information Technology Arcot Sri Mahalakshmi Women's College , Villapakkam, Tamil Nadu, India

*2.Assistant Professor, PG & Research Department of Computer Science &Information Technology Arcot Sri Mahalakshmi Women's College, Villapakkam, Tamil Nadu, India

----------**********----------

Abstract:

Due to the fact that existing database systems are increasingly more difficult to use, improving the quality and the usability of database systems has gained tremendous momentum over the last few years. Data clustering is a process of arranging data into groups or a technique for classifying a mountain of information into some manageable meaningful piles. The goal of clustering is to partitions a dataset into several groups such that the similarity within a group is better than among groups. K- means is one of the basic clustering algorithm which is commonly used in several applications, but it is computationally time consuming and the quality of the resulting clusters heavily depends on the selection of initial centroids. We can remove first limitation using the Enhanced K- Means algorithm. This paper represents the comparison analysis of basic K-Means clustering algorithm and Enhanced K- Means clustering algorithm which shows Enhanced K-Means algorithm more effective and efficient than Basic K-means algorithm. we use the query-refinement method. That is, given as inputs the original top-k SQL query and a set of missing tuples, our algorithms return to the user a refined query that includes both the missing tuples and the original query results. Case studies and experimental results show that our algorithms are able to return high quality explanations efficiently.

Keywords—**Missing answers, Top-K, SQL, Usability, K-Means Clustering, fuzzy logic set,**

----------**********----------

I. INTRODUCTION

A search engine can locate information sources based on keywords which help a user to answer a question. Explaining the features of missing tuples in database queries is called why-not questions. In recent years it has received more attention [1]. In the database community preference query computation has received more attention. To invoke a common approach, a user's preference which give an arrangement of objects to user, and based on his/her decision of the objects; this are strongly introduce the correct weights. For this order-based representative skyline is used and selects representatives which are represented in the order [5].

A why- not question is occurred when a user need to know why her expected tuples are not appeared in the result. Recently, research was done on answering why-not questions on traditional SQL queries However, on preference queries like top-k queries none of those can answer why-not questions yet. The query auto completion concept is used to help a user for formulating the SQL query, but it is complex process to user. The why-not questions are helpful to users to seek clarification on missing tuples from the result.

For answering the why-not questions on top-k queries problem, there are two algorithms to answer such kind questions efficiently. First is a why-not top-k question and second is a why-not top-k dominating question. If user can specify the weighting value, the input query is given to why-not top-k question otherwise the query is given to why-not top-k dominating question. In the missing tuples are already known to the user and explained why those objects are missing [3]. The two algorithms are used which takes user query as an input. In the first algorithm user specifies the weighting value for the query and also uses the scoring function as any monotonic function. The second algorithm is same as Why-Not Top-K question and there is no needto provide weighting value. After the executions of this algorithms result is returned to user by

using Query Refinement Approach which will provide a missing result to user.

This technique defines a best refined query should be (a) similar - there are minimum edit operations than original query and (b) precise - in the result some extra tuples, including the missing objects and original result [3].

Approaches to database preference queries may be classified into two categories according to their qualitative or quantitative nature. In the qualitative approach, preferences are defined through binary preference relations. Among the representatives of this family of approaches, let us mention an approach based on CP-nets, and those relying on a dominance relation, e.g. Pareto order, in particular Skyline queries. In the quantitative approach, preferences are expressed quantitatively by a monotone scoring function [7][3].

Algorithm 1: Basic K-Mean Clustering [5]:

- Choose k points as initial centroid
- Repeat
- Assign each point to the closest cluster centre,
- Recompute the cluster centres of each cluster,
- Until convergence criterion is met.

B. Limitations of K-Means[6][7]:

- It is computationally expensive and requires time proportional to the product of the number of data items, number of clusters and the number of iterations.
- The quality of the resulting clusters heavily depends on the selection of initial centroids which causes it to converge at local optimum.
- Empty clusters problem, which occur to defined fixed cluster in staring of the algorithm.

II. RELATED WORK AN EFFICIENT ENHANCED K-MEANS CLUSTERING ALGORITHM

The aim of following approach makes that K-means algorithm more effective and efficient by removing the first limitation i.e. it limits the number of computations to some extent. This algorithm is easy to implement, requiring a simple data structure to keep some information in each iteration to be used in the next iteration. The idea makes k-means more efficient, especially for dataset containing large number of clusters. Since, in each iteration, the k-means algorithm computes the distances between data point and all centers, this is computationally very expensive especially for huge datasets. Therefore, we do can used from previous iteration of k-means algorithm. We can calculate the distance for each data point to nearest cluster. At the next iteration, we compute the distance to the previous nearest cluster. The point stays in its cluster, if the new distance is less than or equal to the previous distance, and it is not required to compute its distances to the other cluster centers. This saves the time required to compute distances to k-1 cluster centers.

Crowd DB primarily deals with integrating the inherent capabilities of humans into a database system, hence enabling it to answer otherwise difficult queries. Human input is obtained by using *crowd sourcing.* Crowd DB models crowd input as relations in a database. Once federated in Crowd DB, the data from electronic data sources and from people complement each other to make it possible to answer traditionally difficult queries.

Crowd DB taps the underlying design principles of traditional databasesCrowd DB requires the implementation of several extensions to traditional databases.

Crowd DB is essentially a database system it is possible to use a declarative programming language such as SQL to access the federated information.

Operators need to be provided that permit interleaving of query processing and crowd sourcing. Additionally, Crowd DB needs to ensure performance of the system is not adversely impacted while interacting with humans

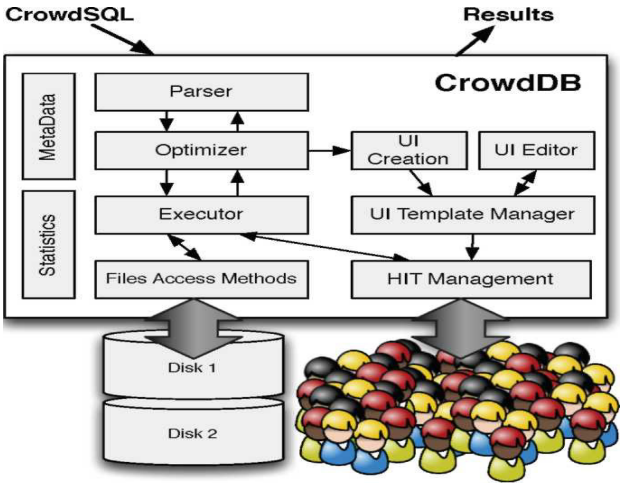

Fig2.1: CrowdDB architecture

A recommender system that supports interactive database exploration. This system aims at assisting non-expert users of scientific databases by tracking their querying behavior and generating personalized query recommendations. The system is supported by two recommendation engines and the underlying recommendation algorithms. The first identifies

potentially "interesting" parts of the database related to the corresponding data analysis task by locating those database parts that were accessed by similar users in the past. The second identifies structurally similar queries to the ones posted by the current user. Both approaches result in a recommendation set of SQL queries that is provided to the user to modify, or directly post to the database system will enable users to query and get real-time recommendations from the Sky Server database, using user traces collected from the Sky Server query log.(QueRIE)

Optimization based interactive query relaxation framework for queries that return no answers. Given an initial query that returns an **empty answer set**, our framework **dynamically computes** and suggests **alternative queries with less conditions than those the user has initially requested**, in order to help the user arrive at a query with a **non-empty answer**, or at a query for which **no matter how many additional conditions are ignored, the answer will still be empty.** Our proposed approach for suggesting query relaxations is driven by a novel probabilistic framework based on optimizing a wide variety of application-dependent objective functions. We describe optimal and approximate solutions of different optimization problems using the framework. We analyze these solutions; experimentally verify their efficiency and effectiveness (DavideMottin, Alice Marascu, SenjutiBasu Roy)

Explaining missing answers in queries that include selection, projection, join, union, aggregation and grouping (SPJUA). Explaining missing answers of queries is useful in various scenarios, including query understanding and debugging. We present a general framework for the generation of these explanations based on source data. We describe the algorithms used to generate a correct, finite, and, when possible, minimal set of explanations. These algorithms are part of **Artemis**, a system that assists query developers in analyzing queries by, for instance, allowing them to ask why certain tuples are not in the query results. Missing tuples at a space that allows developers to effectively use them for query analysis(Melanie Herschel, Mauricio A. Hernandez).

III. PREVIOUS IMPLEMENTATIONS

A fuzzy query involves linguistic terms corresponding to gradual predicates, i.e., predicates which are more or less satisfied by a given (attribute) value. In addition, these various terms may have different degrees of importance, which means that they may be connected by operators beyond conjunction and disjunction. For instance, in the context of a search for used vehicles, a user might say that he/she wants a *compact* car *preferably French,* with a *medium* mileage, *around* 6 k$, whose color is *as close as possible* to light grey or blue. The terms appearing in this example must be specified, which requires a certain theoretical framework.

Goal of the project is to introduce gradual predicates inside database query languages, thus providing flexible querying capabilities. Algebraic languages as well as more user-oriented languages are under consideration in both the original and extended relational settings.

The notion of an uncertain database covers two aspects: i) attribute uncertainty: when some attribute values are ill-known; ii) existential uncertainty: when the existence of some tuples is itself uncertain.

Even though most works about uncertain databases consider probability theory as the underlying uncertainty model, some approaches rather rely on possibility theory. The issue is not to demonstrate that the possibility-theory-based framework is "better" than the probabilistic one at modeling uncertain databases, but that it constitutes an interesting alternative inasmuch as it captures a different kind of uncertainty

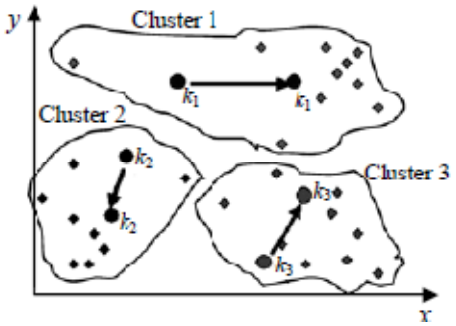

Fig3.1: Recalculating the position of the centroids

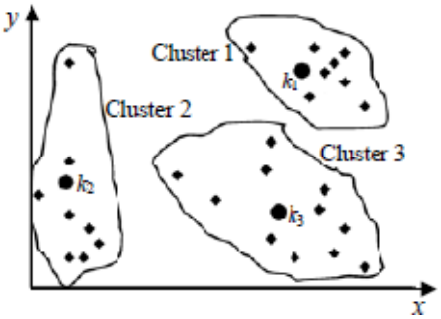

Fig 3.2: Final position of the centroids

3.1 Fuzzy sets

Fuzzy sets were introduced in order to model sets or classes whose boundaries are not sharp. This is particularly the case for many adjectives of the natural language which can be hardly defined in terms of usual sets (e.g., high, young, small, etc.), but are a matter of degree.

A fuzzy (sub)set F of a universe X is defined thanks to a membership function denoted by $[_F$which maps every element x of X into a degree $[_F$(x) in the unit interval [0,1]. When the degree equals 0, x does not belong at all to F, if it is 1, *x* is a full member of F and the closer $[_F$(x) to 1 (resp. 0), the more (resp. less) x belongs to F. Clearly, a regular set is a special case of a fuzzy set where the values taken by the membership function are restricted to the pair {0, 1}.

Fuzzy set F is defined as the (regular) set of elements whose degree of membership is greater than or equal to *a* and this concept bridges fuzzy sets and ordinary sets.

Similarly to a set A which is often seen as a predicate (namely, the one appearing in the intentional definition of A), a fuzzy set F is associated with a gradual (or fuzzy) predicate. For instance, if the membership functions of the fuzzy set *young is given* by:

$$[young(X) = 0\ for\ any\ x > 30, [young(x) = 1\ for\ any\ x < 21, [young(21)0.9[young(22)0.8,.$$
$$[young(29)0.1, young\ old, is\ young([young(26)0.4).$$

Fuzzy set theory starts with a strongly coupled definition of union and intersection which rely on triangular norms (T) and co-norms (_L) tied by de Morgan's laws. Then:

$$[A \cap B(x) = T([a(x), [B(x)) \quad [A \cup B(x) = L([a(x), [B(x))$$

The complement of a fuzzy set F, denoted by F, is a fuzzy set such that: [p(x) = *neg*($[_P$(x)), where *neg is* a strong negation operator and the complement to 1 is generally used. The conjunction and disjunction operators are the logical counterpart of intersection and union while the negation is the counterpart of the complement

3.2 Properties

- Minimum and maximum are the most commonly used norm and co-norm because they have numerous properties among which: potency and double distributive), except excluded-middle and non-contradiction laws, the unit interval
- These three operators given, others can be extended to fuzzy sets, such as the difference:

$$FE — F(x) = T(x), ftp(x))$$

And the Cartesian product:

$$jeXf(x, y) = T(J\ E(x), (y)).$$

The inclusion can be applied to fuzzy sets in a straightforward way:

$$E\ C\ F\ VX, jE(X) < fF(x)$$

Which one is based on the notion of a fuzzy implication (the usual logical counterpart of the inclusion).

The starting point is the following definition valid for sets:

$$E\ C\ F\ 4^{\wedge VX,X\ e\ E} == X\ e\ F$$
$$\deg(E\ C\ F) = T_x(jE(x) == f\ fiF(x))$$

Where =f is a fuzzy implication whose arguments and result take their value in the unit interval. Different families of such implications have been identified (notably R-implications and S-implications) and the most common ones are:

- Kleene-Dienes implication : $a ==_{K—D} b$ = max(1 — *a, b),*
- Rescher-Gaines implication: $a ==_{R—G} b$ = 1 if *a* <*b* and 0 otherwise,
- Godelimplicat ion : *a ==Go b = lis a* <*b* and b otherwise,
- Lukasiewiczimplication: $a ==_{Lu} b$ = min (1, 1 — *a* + b).

Fuzzy sets can also be combined in many other ways, for instance using mean operators, which do not make sense for classical sets.

3.4 Possibility theory

Possibility theory is a theory of uncertainty which aims at assessing the realization of events. The main difference with the probabilistic framework lies in the fact that it is mainly ordinal and it is not related with frequency of experiments

Probabilistic case, a measure (of possibility) is associated with an event.

$$n(x) = 1$$

$$n(0) = 0$$
$$\cap (A \cup B) = \max(n(A), \cup (B))$$

Where X denotes the set of all events and A, B are two subsets of X. If n(A) equals 1, A is completely possible (but not certain), when it is 0, A is completely impossible and the closer to 1 n(A), the more possible A. From the last axiom, it appears that the possibility of *A,* the two Boolean values is:

$$Max(n(A), n(A)) = 1$$

In other words, if A is completely possible, nothing can be deduced for n(A). This state of fact has led to introduce a complementary measure *(N),* called necessity, to assess the certainty of A. N(A) is based on the fact that A is all *the more* certain as A is impossible.

$$N(A) = 1 - n(A)$$

And, in general:

$$n(AnB) < \min(n(A), ll(B)),$$
$$N(A \cup B) > \max(N(A), N(B)).$$

In the possibility setting, a complete characterization of an event requires the computation of two measures: its possibility and its certainty. It is interesting to notice that the following property holds:

$$n(A) < 1 == N(A) = 0$$

It indicates that if an event is not completely possible, it is excluded that it is somewhat certain, which makes it possible to define a total order over events: first, theevents which are somewhat possible but not at all certain (from $(\Pi = N = 0$ *to* $n = 1$ *and* $N = 0)$, then those which are completely possible and somewhat certain (*from* $\Pi = 1$ *and* $N = 0$ *to* $n = N = 1)$. This favorable situation (existence of a total order) is valid for usual events, but if fuzzy ones are taken into account, this is no longer true (because $A \cup A = X$ is not true in general when A is a fuzzy set) and the only valid property is: $V\,A$, $\Pi\,(A) > N\,(A)$.

IV. PROPOSED ANALYSIS

4.1 CrowdDB taps the advantages of being based on traditional relational databases.

Semantic Checker

Input The semantic checker takes, as input, the parse tree representation of the query

Function The semantic checker involves ensuring that the query is semantically correct.Some examples of semantic checks that may be performed in a traditional database system include type checking, ensuring no inconsistencies exist in the query and ensuring that the projected attributes exist in the queried relations. In some cases, the semantic checker also adds details to the internal representation of the query. Additional details include primary and foreign key constraints, correlations between sub queries, etc. If not semantically correct, the checker returns the appropriate error to the application and terminates query execution.

Extensions Apart from the functions listed above, the CrowdDB semantic checker performs additional checks on queries directed at crowd sourced relations. The additional conditions under which the semantic checker throws an error in CrowdDB.

Output As a result of semantic checking, a possibly enhanced parse tree representation is produced.

Code Generator

Input The code generator takes a query plan as input.

Function In a traditional database management system, the code generator is responsible for generating executable code.

Extensions The Crowd DB code generator component extends the same functionality to cater to crowd-specific operations as well.

Output The code generator produces an iterator tree, which is then accessed by the run-time system

4.2 Algorithm 1: An Efficient Enhanced k-Mean

Clustering Algorithm[8] : First Function

Function distance()

//assign each point to its nearest cluster

a. For i = 1 to n

b. For j = 1 to k

c. Compute squared Euclidean distance d2(xi, mj);

d. endfor

e. Find the closest centroid mj to xi;

f. mj = mj+xi; nj = nj+1;

g. MSE = MSE + d2(xi, mj);

h. Clusterid[i] = number of the closest centroid;

i. Pointdis[i] = Euclidean distance to the closest centroid;

j. endfor

k. For j = 1 to k

l. mj = mj/nj;

m. endfor

First, calculate the distances between point number I and all k centriods in Line 3. And determine the nearest centroid to data points in Line 5.and Line 6 adds point number i to cluster number j, and increase the count of points in cluster j by one. Line 8 and 9 are keep records of number of closest centroid and Euclidean distance to the closest centroid.and again recalculated the new centroids in Line 12. The rest of the algorithms are shown in algorithms 3, is called distance_new(). Line 1 finds the distance between the current point i and the new cluster center assigned to it in the previous iteration, if the computed distance is smaller than or equal to the distance to the old center, the point stays in its cluster that was assigned to in previous iteration, and there is no need to compute the distances to the other k-1 centers. Lines 3~5 will be executed if the computed distance is larger than the distance to the old center, this is because the point may change its cluster, so Line 4 computes the distance between the current point and all k centers. Line 6 searches for the closest center, Line 7 assigns the current point to the closest cluster and increases the count of points in this cluster by one, Line 8 updates mean squared error. Lines 9 and 10 keep the cluster id, for the current point assigned to it, and its distance to it to be used in next

call of that function (i.e. next iteration of that function). This information is kept in Line 9 and Line 10 allows this function to reduce the distance calculation required to assign each point to the closest cluster, and this makes the function faster than the function distance in

4.3 Algorithm 2: An Efficient Enhanced k-Mean Clustering Algorithm[8] :Second Function
//Assign each point to its nearest cluster
a. For i = 1 to n

Compute squared Euclidean distance
d2 (xi, Clusterid[i]);
If (d2 (xi, Clusterid[i]) <= Pointdis[i])
Point stay in its cluster;
b. Else
c. For j = 1 to k
d. Compute squared Euclidean distance d2(xi, mj);
e. Endfor
f. Find the closest centroid mj to xi;
g. mj = mj+xi; nj = nj+1;
h. MSE = MSE + d2(xi, mj);
i. Clustered[i] = number of the closest centroid;
j. Pointdis[i] = Euclidean distance to the closest centroid;
k. Endfor
l. For j = 1 to k
m. mj = mj/nj;
n. endfor

V.EVALUATION RESULTS

Comparisons performed by the crowd as a result of the CROWDEQUAL function are cached for future use in auxiliary tables. The processing of the CROWDEQUAL function is explained in further CROWDEQUAL only functions on existing tuples for the purpose of subjective comparison or entity resolution. Its usage does not lead to the crowd sourcing of new tuples. Hence, the LIMIT clause is not mandatory, whether the table is regular or crowd sourced.

		Regular table	**Regular table with Crowd columns**	**Crowd table**
Table 2	**Regular table**	Traditional join	Incomplete tuples in Table 1 may be crowdsourced	Incomplete tuples and new tuples in Table 1 may be crowdsourced
	Regular table with Crowd columns	Incomplete tuples in Table 2 may be crowdsourced	Incomplete tuples in both tables may be crowd-sourced	Incomplete tuples in both tables may be crowd-sourced, New tuples in Table 1 may be crowdsourced
	Crowd table	Incomplete tuples and new tuples in Table 2 may be crowdsourced	Incomplete tuples in both tables may be crowd-sourced, New tuples in Table 2 may be crowdsourced	Incomplete tuples and new tuples in both relations may be crowdsourced

Table1: Two-way joins in CrowdDB

In the presence of referential integrity constraints, CrowdDB executes a functional join query. In addition to completing missing information using the semantics specified in Table 5.1, CrowdDB takes special care to ensure that the missing references that are crowd sourced do not violate any referential integrity constraints. If the referencing column is crowdsourced, it is possible that it contains CNULL values. This denotes a missing reference which can be completed as a side-effect of query processing. The options available to the crowd for completing missing references depend on whether the referenced table is regular or crowdsourced. If the referenced table is regular, the crowd completes the missing reference by ensuring it refers to an existing key. Alternately, if the referenced table is crowdsourced, the crowd can complete the missing reference by making it refer to an existing key or by adding anew key to the referenced table. Summarizes the additional restrictions on the completion of missing references during functional join execution based on the nature of the referenced and referencing columns.

The first function is the basic function of the k-means algorithm, that finds the nearest center for each data point, by computing the distances to the k centers, and for each data point keeps its distance to the nearest center. The first function is shown in Algorithm 2, which is similar to that in Algorithm 1, with adding a simple data structure to keep the distance between each point and its nearest cluster. This function is called distance().

		Referencing Table		
		Regular table	Regular table with Crowd columns	Crowd table
Referenced Table	Regular table	Traditional join	Incomplete foreign keys must refer to *existing* primary keys	Incomplete foreign keys must refer to *existing* primary keys
	Regular table with Crowd columns	Traditional join	Incomplete foreign keys must refer to *existing* primary keys	Incomplete foreign keys must refer to *existing* primary keys
	Crowd table	Traditional join	Incomplete foreign keys can refer to *existing or new* primary keys	Incomplete foreign keys can refer to *existing or new* primary keys

Table 2: Additional restrictions on two-way functional joins in CrowdDB

In the original K-means algorithm, before the algorithm converges the centroids are calculated many times and the data points are assigned their nearest centroids[10,11,12]. since complete new redistribution of the data points takes place according to the new centroids,this takes O(nkl),where n is the number of data objects, k is the number of clusters and l is the number of iterations. In enhanced K-meams algorithms, to obtain initial clusters, this process requires O(nk).here, some data objects remains in its cluster while the others move to other clusters depending on their relative distance from the new centroid and the old centroid. This requires O(1) if a data-object stays in its cluster, and O(k) otherwise. As the algorithm converges, the number of data objects moving away from their cluster decreases with each iteration. Assuming that half the data objects move from their clusters, this requires O(nk/2). Since the algorithm converges to local minimum, the number of points moved from their clusters decreases in each iteration .so we expect the total cost is nkΣ 1/n.

Even for large number of iteration,

i=1;

So, its cost is approximately $O(n_k)$, *not* $O(n_{kl})$.

5.1 SAMPLE SCREENS

Fig5.1: suvey of Prediction

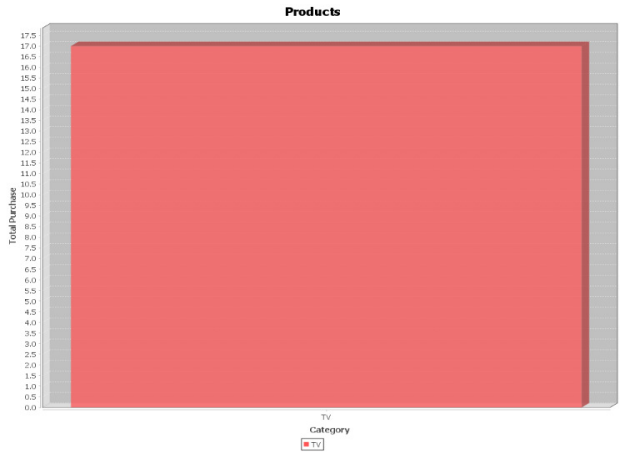

Fig5.2: Category wise Data analysis

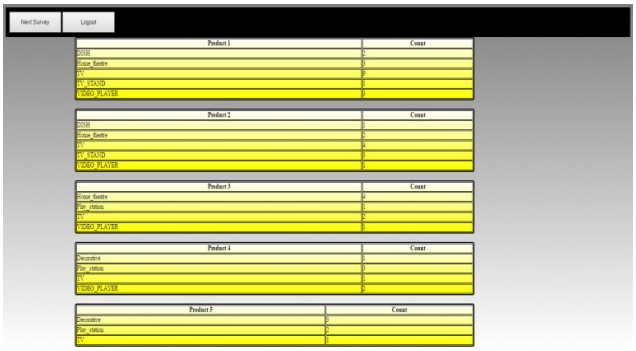

Fig 5.3: Review of each product iteams

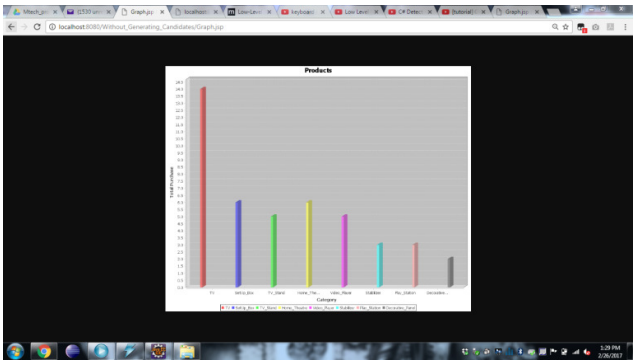

Fig5.4 : Performance of Product details

6. CONCLUSION

To solve the scalability and load balancing challenges in the existing parallel mining algorithms for frequent itemsets, we applied the MapReduce programming model to develop a parallel frequent itemsets mining algorithm. 1) A straight information structure, CAUL, is proposed, which focuses on the underlying driver of the two phase, applicant era approach embraced by earlier calculations, that is, their information structures can't keep the unique utility data. 2) A high utility example development methodology is introduced, which incorporates an example identification technique, pruning by utility upper jumping, what's more, CAUL. This essential methodology beats earlier calculations strikingly. 3) Our methodology is upgraded essentially by the look ahead methodology that distinguishes high utility examples without specification. Later on, we will take a shot at high utility successive example mining, parallel and disseminated calculations, and their application in huge information investigation. We improve the performance of FiDoop by balancing I/O load across data nodes of a cluster.

6.1 FUTURE WORK

While the two-phase, candidate generation approach first generates high TWU patterns (candidates) and then identifies high utility patterns from high TWU patterns, our approach directly finds high utility patterns without generating any high TWU patterns (candidates).

6.2 ADVANTAGES:

- Original utility information in raw data set is easily retrieved without generating candidate's server.
- The number of high utility patterns with item set pair is easily identified

REFERENCES

[1] S. Agrawal, S. Chaudhuri, and G. Das, "DBXplorer: A System for Keyword-Based Search over Relational Databases," in ICDE, 2002, pp. 5–16.

[2] H. Wu, G. Li, C. Li, and L. Zhou, "Seaform: Search-As-You-Type in Forms," in PVLDB, vol. 3, no. 2, 2010, pp. 1565–1568.

[3] J. Akbarnejad, G. Chatzopoulou, M. Eirinaki, S. Koshy, S. Mittal, D. On, N. Polyzotis, and J. S. V. Varman, "SQL QueRIE Recommendations," in PVLDB, vol. 3, no. 2, 2010, pp. 1597–1600.

[4] M. B. N. Khoussainova, Y.C. Kwon and D. Suciu, "Snipsuggest: Context- Aware Autocompletion for SQL," in PVLDB, vol. 4, no. 1, 2010, pp. 22–33.

[5] A. Chapman and H. Jagadish, "Why not?" in SIGMOD, 2009, pp. 523– 534.

[6] J. Huang, T. Chen, A.-H. Doan, and J. F. Naughton, "On the Provenance of Non-Answers to Queries over Extracted Data," in PVLDB, 2008, pp. 736–747.

[7] M. Herschel and M. A. Hern´andez, "Explaining Missing Answers to SPJUA Queries," in PVLDB, 2010, pp. 185–196.

[8] Q. T. Tran and C.-Y.Chan, "How to ConQueR Why-not Questions," in SIGMOD, 2010, pp. 15–26.

[9] Z. He and E. Lo, "Answering Why-Not Questions on Top-K Queries," in ICDE, 2012.

[10] I. Md. Saiful, Z. Rui, and L. Chengfei, "On Answering Why-not Questions in Reverse Skyline Queries," in ICDE, 2013, pp. 973–984.

[11] Z. He and E. Lo, "Answering why-not questions on top-k queries," IEEE Trans. Knowl.Data Eng., vol. 26, no. 6, pp. 1300–1315, 2014.

[12] A. Motro, "Query Generalization: A Method for Interpreting Null Answers," in Expert Database Workshop, 1984, pp. 597–616.

[13] A. Motro, "SEAVE: A Mechanism for Verifying User Presuppositions in Query Systems," ACM Trans. Inf. Syst., vol. 4, no. 4, pp. 312–330, 1986.

[14] F. Zhao, K.-L. T. G. Das, and A. K. H. Tung, "Call to Order: A Hierarchical Browsing Approach to Eliciting Users' Preference," in SIGMOD, 27-38, p. 2010.

[15] A. Vlachou, C. Doulkeridis, Y. Kotidis, and K. Nørv°ag, "Reverse Top-K Queries," in ICDE, 365-376, p. 2010.