International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Optimized Crowdsourcing Systems for Advanced Structured Query Language Operators

S. Y. Shelar[1*], A. B. Rajmane[2*]

[*]Ashokrao Mane Group of Institutions, Vathar, Maharashtra, India
[1]seemashelar01@gmail.com
[2]amolbrajmane@gmail.com

Abstract: ***Query optimization plays an important role in crowdsourcing system. We study the optimized crowdsourcing system which is SQL opeartors based crowdsourceing system, in which user submits the query. Crowdsourcing system compiles query and creates the evaluation plans, then excutes selected plan on crowdsourcing platform. A submitted query has many execution plans but selected best query plan gives siginificant effect on overall performance of crowdsource system. For this purpose, we present OptimizedCrowd system in which query optimization is based on latency. In the developed system for query optimization purpose, we use select and join queries.***

Keywords: Crowdsourcing, Query Optimization, Human Intelligence Tasks (HIT), SQL.

1. Introduction

Crowdsourcing system is web based activity in which thousands of people simultaneously post and edit work e.g., Yahoo! answers [11] where users take the review of people in question-answer format. Crowdsourcing is software which solves the complex tasks that computer cannot solve easily so with the equal intention crowdsourcing system combines human logic with computer power and gets the accurate result. Recently, crowdsourcing system has been adopted in database system. The brief information of developed optimizedcrowd system which uses the SQL operators like select [2] [10], fill [7], count [3], max [4] and join [5] in the query optimization process is overviewed chronologically. Some crowdsourcing systems such as CrowdDB [6], Qurk [9], Deco [7] and CrowdOP [8] provide the SQL interface which is known to the database users. Crowdsourcing system gives the platform on which requester posts tasks and workers accept tasks and works on these tasks. In crowdsourcing system, queryoptimizer generates evaluation plans for submitted query and system selects the best query plan to generate the tasks. Crowdsourcing system has an immense impact of selection of best query plan. By considering this impact we produced the OptimizedCrowd system. Even query processing in the crowdsourcing system is as same as in the database, the query optimization has greater importance as far as the performance of the crowdsourcing system is concerned.

The following are some of the characteristics of the developed system,

1) Latency based optimization: Latency is parameter which is used to improve the performance of crowdsourcing system. Latency means how long people wait for results. We examine the recent crowdsourcing system such as CrowdDB [6], CrowdScreen [2] only works on cost and CrowdFind [10] that takes latency to find out best query plan for optimization purpose. Our system considers the latency into a query optimization purpose.
2) Multiple crowdsourcing operators are used in optimizations: Deco [7] works on the missing tuples from the database. Qurk [9] focuses on *join* and *sort* operators. CrowdOP uses the *fill, select* and *join* operators in query optimization technique. CrowdScreen and CrowdFind work for *select* operator. The proposed system works on the operators such as *select* which finds the specific tuples from relation, *fill* is used to place value which is unknown to database. Aggregation operation such as *count* [3] is used to estimate number of items in dataset that satisfies a predicate or special condition and *max* is used to find the highest ranked object or tuple in set. In this paper, we study the query optimization for various SQL operators by using latency. The remaining paper contains following parts, part 2 describes related works for existing system. Part 3 describes architecture of proposed model and parts 4 contain experimental results for *select* and *join* queries. Part 5 concludes the paper.

2. Related Work

Now a days, many works have been proposed to perform database operations in a crowdsourcing system such as *select, join* and aggregation operators such as *count, max* and *sort.* Recently many crowdsourcing systems have been developed to provide SQL like query interface to the crowd such as CrowdDB system solves the queries by using human answer which cannot be solved by database and uses SQL as query language. It is rule based optimization crowdsourcing system. Rule based is easy to implement but it generates very less effective evaluation plans. It uses *crowdprobe*, *crowdjoin* and *crowdcompare* operators in query optimization process. Deco is declarative crowdsourcing system in which SQL queries are solved. Deco works on missing value of database. Deco uses only *fill* operator. CrowdOP is the optimized crowdsourcing system which uses *Cselect*, *Cjoin* and *Cfill* operators in query optimization process. CrowdOP takes cost and latency into optimization process. Our developed system takes latency for optimization purpose because latency is an important key element in crowdsourcing system. This latency model is similar to CrowdOP and CrowdFind. But CrowdFind works only select operator and CrowdOP works for *Cselect, Cfill* and *Cjoin* queries while our work focus is on optimization with the help of some advance aggregation operators such as *count* and *max* with minimum latency.

3. Overview of Proposed System

The proposed system uses database operations in query optimization process so that there is need to use database query language in the system. In this section, we introduce the data model and query language that is used by the proposed system and also architecture of proposed system is described.

3.1 Data Model

The Proposed system uses relational data model. In this system data is considered as schema that consists of a set of relations. These relations are designated by schema designers. SQL query can be executed by relations and queries can be given by requester.

3.2 Query Language

The proposed system goes through the SQL query language over the database and result of the query is obtained by executing query by using the relations used in the data model. The proposed system considers following two types queries.

(1) Selection Query: A selection query uses one or more selection conditions over the records in single relation. It has many applications in crowdsourcing platform like filter item [2]. A simple example is find out cars having width is less than equal to 66.5 and length is greater than equal to 165.3 and wheel base is less than equal to 96.9. For this, we express the query as Q1 for three conditions,
*Q1: select * from car where width <= 66.5 AND length >= 165.3 AND wheel_base < 96.9*
We take another example of selection query for two conditions and express the query as Q2,
*Q2: Select * from car where wheel_base < 96.1 AND stroke >= 3.12*
(2) Join Query: Join query is used to combine records from two or more relations according to certain conditions. An example join query Q3 over relations *car* and *image* which combines records from *car* relation to *image* relation, which is presented as follows.
Q3: select c1., c2.* from car c1, image c2 where c1.carid=c2.carid and c1.symboling >= 2 and c1.bore >= 3.31 and c1. highway_mpg > 34*

3.3 Architecture of Proposed System

The proposed system is based on crowdsourcing platform which has some optimization in sense of SQL queries. The proposed system includes SQL queries like *select, join, fill, count*, and *max*. In this section, we describe overall architecture of proposed system. The workflow of proposed system is shown in figure 1.

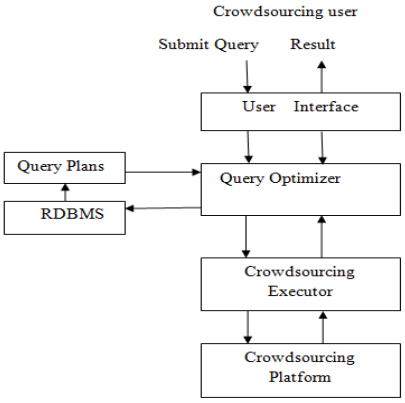

Figure 1: Block Diagram of the OptimizedCrowd System

The proposed system incorporates the traditional query compilation, optimization and execution components. We briefly describe each component as follows.

1) **User Interface:** By using user interface, user submits SQL query.
2) **Query Optimizer:** Submitted query is processed by query optimizer. In query optimizer, query first compiles and generates the optimized query plans which is based on latency i.e. execution time of query. The query optimizer selects the low latency query plan for execution purpose.
3) **Crowdsourcing Executor:** The selected query plan is evaluated by crowdsourcing executor and generates the tasks. Then publish these tasks on crowdsourcing platform. These tasks are solved by human workers and answers are given to these tasks. The system collects the answer from the workers. By using these answers, crowdsourcing executor, executes the query and accurate result is given to the user.

4. Experimental Results

In this section, initially we examine the effectiveness of our proposed optimization method for the *select, join* queries. For our experiments purpose, we use a real auto import dataset [1] which consists of specification of 205 cars to generate the *car* relation. This dataset is in text format. In preprocessing step, we convert text format into SQL database format so that system can easily process database queries. Then we manually generate relation *image* by adding image related with each car present in *car* relation. This image is taken from yahoo auto [12].

4.1 Evaluating *Selection* Query Optimization

In this section, we evaluate optimization method for selection queries. We vary the number of selection condition from two to three. For experimental purpose, we take example query Q1 with three conditions and Q2 with two conditions. In Query optimization process, query Q1 generates the query plans. In experimental study, we calculate execution time of each query plan on the basis of number of records fetch by each query condition which is used in the query. Performance of each of the query plan of query Q1 shown in the figure 2.

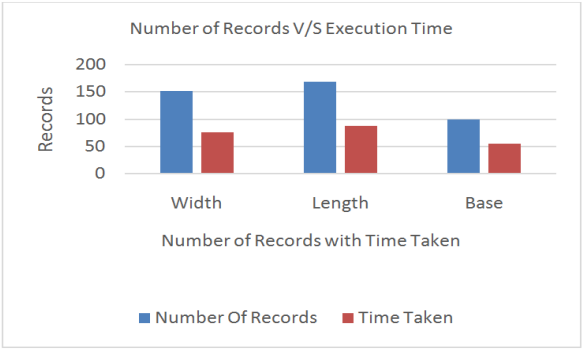

Figure 2: Performance of Query Q1 with Execution Time and Conditions

In figure 2, we take the count of records fetched from database on the basis of each condition used in query and takes execution time of each query plan. When system execute the query Q1 on the basis of wheel_*base* condition, the system return result in less time than other two conditions used in query Q1. Due to low execution time, system selects query plan as *select * from car where wheel_base < 96.9 and length >= 165.3 and width <= 66.5.* This query plan is used for tasks generation. We also take the experimental result for query Q2. Figure 3 represent execution time and records fetch by each condition used in query Q2. In experimental analysis, we take count of records fetch by each condition and execution time of each condition used in the query. We have observed that to display result of query Q2, system gives best results in comparatively minimum time when it executes the query on *wheel_base* condition. As a result the query with less execution time as "select * *from car where wheel_base < 96.1 and stroke >= 3.12"* is selected for tasks generation.

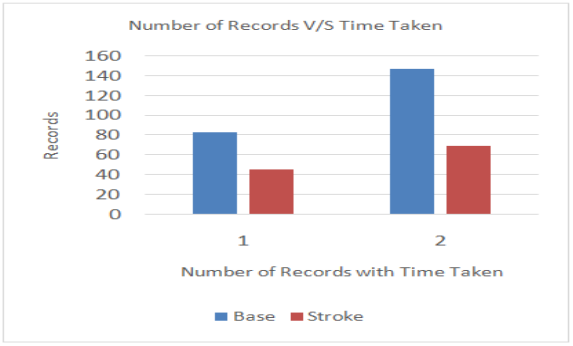

Figure 3: Performance of Query Q2 with execution time

4.2 Evaluating *Join* Query Optimization

In this section, we validate optimization approach for join query. For experimental purpose, we take Q3 query with three conditions. We join two relations *car* and *image* on basis of attribute *carid.* Query Q3 evaluate in three ways on the basis of conditions present in the query. This evaluation of each condition of query Q3 with execution time and records fetched by each attribute of the query is shown in figure 4. From figure 4, it clearly shows that when query Q3 evaluate on the condition *highway_mpg* then system gets result in very less time. Due to low execution time, system selects this query plan for task generation.

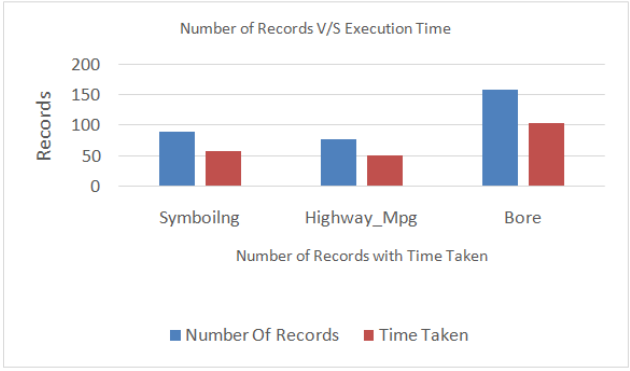

Figure 4: Performance of Query Q3 with execution time

From the above discussion, we have observed that selection of best query plan improves the performance of system in terms of execution time hence query optimization plays an important role in developed system.

5. Conclusion and Future Work

In this paper, we have presented a latency based query optimization which helps to optimization in sense of SQL queries. The system includes SQL queries like select, join, fill operations. In addition to these three operations some aggregate operators are used in query optimization purpose such as count and max. This optimization is effective for SQL query processing. In future, we intend to study complex join query for query optimization and also we use cost and latency combination in query optimization technique.

References

[1] https://archive.ics.uci.edu/ml/datasets/Automobile
[2] A.G. Parameswaran, H. Garcia-Milina, H. Park, N. Polyzotis, A. Ram and J. Windom, "CrowdScreen: Algorithms for filtering data with humans," in Proc ACM SIGMOD Int. Conf. Manage. Data, 2012, pp. 361-372.
[3] A. Marcus, D.R. Karger, S. Madden, R. Miller, and S. Oh, "Counting with the crowd", Proc. VLDB Endowment, vol. 6, no. 2, pp. 109-120, 2012.
[4] P. Venetis, H. Garcia-Molina, K. Huang, and N. Polyzotis, "Max algorithms in crowdsourcuing Environment",in Proc. 21st Int. Conf. World Wide Web, 2012, pp. 989-998.
[5] A. Marcus, E. Wu, D.R.Karger, S.Madden, and R.C.Miller, "Human- powered sort and joins", Proc. VLDB Endowment, vol. 5, no. 1, pp. 13- 24, 2011.
[6] M.J.Franklin, D.Kossmann, T. Kraska, S.Ramesh, and R. Xin, "CrowdDB: Answering queries with crowdsourcing", in Proc. ACMSIGMOD Int. Conf. Manage. Data, 2011, pp. 61-72.
[7] A.G.Parameswaran, H.Park, H.Garcia-Moline, N.Polyzotis, and J. Widom, "Deco: Declarative crowdsourcing", in Proc. 21st ACM Int. Conf. Inf. Knowl. Manage, 2012, pp. 1203-1212.
[8] Ju Fan, Meihui Zhang, Stanley Kok, MeiyuLu, and Beng Chin Ooi, "CrowdOP: Query Optimization for Declarative Crowdsourcing Systems", IEEE

Transactions on Knowledge and Data Engineering, vol. 27, no.8, pp. 2078-2092, August 2015.
[9] A. Marcus, E. Wu. S. Madden, and R.C.Miller, "Crowdsourced databases: Query processing with people", in Proc. 5th Biennial Conf. Innovative Data Syst. Res., 2011, pp.211- 214.
[10] A.D.Sharma, A. Parameswaran, H. Garcia- Molina, and A. Halevy, "Crowd-powered find algorithms", in Proc. IEEE 30th Int. Conf. DataEng., 2014, pp. 964- 975.
[11] Yahoo! answers, http://answers.yahoo.com
[12] https:// autos.yahoo.com/