Aggregate Queries

Caroline KyuMin Kim
College of Information and Computer Science
kyuminkim@umass.edu

12/15/2018

Advisor: Gerome Miklau
Second Committee Member: Marco Serafini
Research Type: Thesis

1. Introduction

Through this research, the most effective method to answer a set of large number of aggregate queries will be found. An aggregate query is a method of grouping the dataset to analyze it and produce single output for that group. This is a method that is used in daily life such as computing average value, or counting number of elements, finding maximum or minimum value, or computing sum or standard deviation. But for this project, the focus will only be on the predicate counting queries. Predicate counting queries can be used in many different areas to give a count of objects that satisfies certain condition such as counting the number of movie titles that begin with the letter “C” in a movie database for example. Aggregate queries like this are common in databases, and there are ways to speed the process up for a single query such as using indexes. But simultaneously answering many of them can be challenging.

To solve this problem, this research will be conducted to find how to efficiently run a large collection of aggregate queries on a database. Since the calculations done by aggregate queries can be used in many different fields, I hope to develop a system that will calculate aggregate functions without going through unnecessary steps as much as possible.

The goal of this research is to run all queries on a table when given the input of a large collection of aggregate queries Q and a database which will be a single table R in ways that will optimize time and memory space. And the size of Q can range from thousands to hundreds of thousands. Since there are different types of databases that people use to run aggregate queries, I will be approaching this problem in many different ways. The different ways will include using relational system performing single-query evaluation and grouped query evaluation, performing vector evaluation of queries, and performing map-reduce using Spark. In order to test these methods, I’ll be using the census as the sample dataset. Since the census has many different attributes such as gender, age group, race, or relationship in the household, there will be numbers of

different ways to write aggregate queries over this dataset. Some examples of these queries will be to count the number of males who are over 40 years old or the number of females who are the head of the household. With the census data, I will explore using relational database management system as well as system that is not relational. And by testing these methods, I hope to find ways to run all the queries in a way that will optimize time and memory space the most so that it can effectively be used when a large set of aggregate queries have to be run on a dataset no matter what kind of database management system the data is stored in.

2. Significance

Through this research, the most effective way of running large sets of aggregate queries can be found. There are many cases in different fields where they run numerous aggregate queries. So it will be helpful to find ways to optimize the space and time in running them. Also, this research will explore different methods to be applied to different types of datasets. So by looking at each evaluation, I will be able to determine which method works best in different situations. This research can be extended to be used in other aggregate queries other than the predicate counting queries using similar methods explored.

3. Background

Over the 499Y semester, works that are related to aggregate queries and implementations that I will be using have been reviewed over.

In the paper *Efficient Implementation of Data Cubes via Materialized Views*, Online Analytical Processing (OLAP) is described. OLAP is an approach for conducting multidimensional analysis of data. It is a relational database system, and it lets the users directly query the raw data. The multidimensional databases retain significant performance advantage. And performance in relational database systems can be improved dramatically by materializing the data cube into summary tables. Data cubes are special-purpose DBMS for storing multidimensional data and handling queries that aggregate over some dimensions. And views are projections of the cube onto some of its dimensions. Each queries have a natural view, but with views that group by more attributes, more can be answered. However, those views are larger and require additional cost. This paper explains the greedy algorithm for selecting optimal view is explained. This algorithm assumes that the top view is materialized. Then it selects additional views to materialize one at a time, until some total cost of selected views is reached. And at each step, it selects the view that most reduces the average cost of answering a query per unit space.

In the paper *The Matrix Mechanism: Optimizing Linear Counting Queries Under Differential Privacy,* the matrix mechanism, linear query, and the query matrix are explained. The matrix mechanism is an algorithm for answering a workload of linear

counting queries that adapts the noise distribution to properties of the provided queries. This mechanism uses a different set of queries which are answered using a standard Laplace or the Gaussian mechanism. Linear query is an aggregation query over a single relation that can be expressed as a linear combination of a set of database counts. And query matrix is a collection of m linear queries. Other details of privacy mechanisms in this paper are not relevant. But the vector representation of the table used in this research is important since it will be used in my research. This paper uses single query evaluation as dot product of query vector and data vector, and multiple query evaluation as matrix multiplication of query matrix and data vector.

The use of continuous queries is discussed in the *NiagaraCQ* paper. Continuous queries allow users to obtain new results from a database without having to issue the same query repeatedly. It can be useful to work with database that is frequently changing. It also introduces incremental grouping methodology. In this method, dynamic regrouping is used in cases of reduction in the overall performance of the system to re-establish their effectiveness because the quality of the group can deteriorate over time. Also, new queries are added to existing groups can optimize without having to regroup already installed queries.

A paper on Spark SQL was reviewed since I will be using Spark as one of the implementations in my research. This paper explained about the relational processing, declarative queries, and optimized storage. And it also showed how SQL users can call complex analytics libraries in Spark. Spark SQL supports relational processing both within Spark programs and on external data sources, provides high performance using established DBMS techniques, supports new data sources, and enables extension with advanced algorithms. It also explains the use of catalyst optimizer which makes it easier to add new optimization techniques and features to Spark SQL and enables external developers to extend the optimizer.

4. Methodology

The census data from the year of 2010 will be used in order to test different methods in this project. This dataset has two tables named ‘household’ and ‘person’. A ‘person’ in this dataset has attributes such as age, sex, race, hispanic, relationship with others, and more. And these people share the household link together if they belong in the same household. With this dataset, many different counting queries can be generated. The count for male, female, certain gender with specific age, or number of household heads who are female are examples of some of the counting queries that can be used. This dataset will be organized into different tables or matrices that match the descriptions from the “2010 Census Summary File 1”. And many queries will be generated from these tables so that they can be used during the research.

Four different methods will be explored throughout this research, and those methods consist of different evaluation strategies using different systems. For all four of

these methods, the metric will be the runtime of the system for calculating the aggregate queries and python will be used for coding. The four methods are:

1. Separate SQL Expressions Evaluation in RDBMS
The first method is evaluating separate SQL expressions in relational database management system. This method is the most basic approach to this problem. I will be executing each aggregate query in workload W, which is a separate SQL expression, and be adding up the runtime of all the queries. Since this is the most basic way of solving this problem, this can be used as a result that rest of the methods can be compared to in order to rank the efficiency.

2. Grouped Query Evaluation in RDBMS
The second method is the grouped query evaluation. This will also be in a relational system, and this method will require reformulation of the workload W into a smaller set W' of group-by queries or queries using cube operator. From these sets, each w in W will be able to be derived. I will have to research a way to go from W to W' without having to manually change all of them. Then each of the query in W' will be evaluated, and the runtimes will be added up. And from that result, I will have to derive an answer to each query in W. The challenge for this method will be to derive W' from W. And after that, it will be very similar to the first method. But ultimately, this method will show if using group-by queries or queries with the cube operator is more effective way of calculating large set of aggregate queries.

3. Vector Evaluation of Queries in Apache Spark
The third method for this research is the vector evaluation of queries. I will use Apache Spark to vectorize the dataset. For this method, the challenge will be to effectively vectorize the database table. The database will be put in to different kind of vector forms to explore which form of vector will be the most effective. Then, the dot product or matrix multiplication of these vectors will be performed to produce answer for the aggregate queries. The runtime of doing the dot product or matrix multiplication and giving the answer will be measured.

4. Map-Reduce in Apache Spark
In the last method, I will be using Map-Reduce. I will use Apache Spark to map the dataset. This process will need a system that can find the most common attributes in the set of queries to map the dataset to. Then, using Apache Spark again, these queries that were mapped will be grouped together with similar ones to be reduced down. And the runtime of this process will be evaluated to compare to other methods. This method will be the most challenging method to implement because it will require a system that

will find the most efficient way of picking attributes for the map-reduce given a large number of aggregate queries.

The four methods will be tested on the sets of aggregate queries with different sizes. Then the results will be evaluated to see which method works best in which environments. From this result, the conclusion will be drawn to match the most effective method to different cases.

5. Evaluation

- Milestone 1 - Create workloads and data generators
    - The workloads will be the aggregated queries
    - Workloads created during this milestone will be used to test different methods for the research
- Milestone 2 - Centralized relational database management system
    - Implement workload evaluation
    - Experiment
    - Interpret the experimental results
- Milestone 3 - Linear Algebra
    - Implement workload evaluation
    - Experiment
    - Interpret experimental results
- Milestone 4 - Spark
    - Implement workload evaluation
    - Experiment
    - Interpret experimental results
- Milestone 5 - Synthesize experimental findings
- Milestone 6 - First draft of the thesis
    - Submit to the advisor
- Milestone 7 - Second draft of the thesis
    - Submit to HPD
- Milestone 8 - Oral Defense
- Milestone 9 - Submission to the Honors College

For the first four milestones, the work will be evaluated on the effectiveness of the implementation. It will be evaluated and though over on if there is a better evaluation method or not. Other milestones will be discussed with the committee to see if there are anything that has to be fixed or be organized differently. The overall research will be evaluated on finishing all the milestones on the given due date. The committee will be

discussing the result of each milestone with me in every meeting and giving feedback on them.

6. Communication

There will be weekly meetings with the primary advisor. And as needed, the experimental results or the implementation strategies will be discussed through email. Each meeting will be about an hour long to discuss the process and the result for the experiments. For all the meetings, I will be expected to have everything that is due on that day. If nothing is due by that time, I will be expected to bring in the process of the current milestone I'm working on. And every week, I will be expected to work on this for 10 hours outside of the meeting time.

7. Timeline

- 01/29/2019 (week 1) - Milestone 1
- 02/12/2019 (week 3) - Milestone 2
- 03/05/2019 (week 6) - Milestone 3
- 03/26/2019 (week 9) - Milestone 4
- 04/09/2019 (week 11) - Milestone 5
- 04/16/2019 (week 12) - Milestone 6
- 04/23/2019 (week 13) - Milestone 7
- 04/30/2019 (week 14) - Milestone 8 and 9

8. References

Efficient Implementation of Data Cubes via Materialized Views

Harinarayan, Venky, et al. "Implementing Data Cubes Efficiently." Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data - SIGMOD '96, 1996, doi:10.1145/233269.233333.

Li, Chao, et al. "The Matrix Mechanism: Optimizing Linear Counting Queries under Differential Privacy." The VLDB Journal, vol. 24, no. 6, 16 July 2015, pp. 757–781., doi:10.1007/s00778-015-0398-x.

NiagaraCQ: A Scalable Continuous Query System for Internet Databases

Chen, Jianjun, et al. "NiagaraCQ." ACM SIGMOD Record, vol. 29, no. 2, Jan. 2000, pp. 379–390., doi:10.1145/335191.335432.

Introduction to Spark and Spark SQL Paper

Armbrust, Michael, et al. "Spark SQL." Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data - SIGMOD '15, 2015, doi:10.1145/2723372.2742797.

U.S. Census Bureau, 2010 Census of Population and Housing, Demographic Profile Summary File: Technical Documentation, 2011.