03/14/2022

In today's class we discussed on sorting the tuples and accessing the tuples with less cost. It involves the accessing the Top N elements which is the most used thing for the current scenarios. As this is costly operation and we need to find an efficient way to obtain the needed tuples. Below is the approach followed to obtain it.

Firstly, we create and initialize the TopK and initialize it, we keep on adding the elements to the tuples involving a case to check if the size is less than K and if not, push an item into the TopK. Which is obvious case always. If we find the worse in topk we replace the worse tuple with the other in the topk. The need to find and store the TopK elements is always a need to access the most frequently used which helps in reducing the processing capabilities to process and use it. Also, with the fact it also uncovers several interesting data elements in the fields of the statistical analysis. Another important concept is to combine them with grouping things helps to extract important information. Also helps in coming up with the values to data mine the incomplete and ignore the noise data which may not contribute to process and use. There is an interesting behavior that once the topk is full of the sense that it is holding a high expectation that it occurs, and new tuples are not that competitive and contributes to the much fast access than the most selections made previously. With this interesting behavior we find many User defined aggregates can be used to access the data and is quite common in most RDBMS systems and restricting the disk/memory page to 4KB making it to produce one single tuple.

Based on the facts several observations are concluded that DBMS works slow when it is prompted to push and pull the data from it as a result DB Clients cannot deal with the large results. Also, we discussed the concepts of grouping based on topk elements using the Generalized Linear Aggregate functionality.

03/16/2022

Discussion started with understanding the Generalized Linear Algebra, we can define GLA using the UDA interface, while it can only be seen using Structured Query Language (SQL), where a user provides the immediate access to the current state of Aggregate. It needs to move the runtime address space to user application of memory space for the GLA to start functioning. Defined more details about Meta GLA and templates are used to define another GLA. Example includes <GLA1, GLA2...>

Implementation of grokit which inturn uses GLAs. A combination of PHP and C++. Type generation is looked into using grokit and generated the code for the same, gone through the code generation from the input of GLA.

03/18/2022

Based on the previous lecture on making the GLA and we started the discussion on the understanding the types of GLAs. Using the different operators such as Group-By, Top-K, Distinct. Several Meta-GLAs can also be concluded using Group-By Multiplexer. These are basically used in statistical analysis as discussed in the previous lecture for the computation of Count, Average, Variance, Min, Max, CountDistinct which are commonly used mathematical calculations on the data. We can also start several approximation elements too using Bloom Filters, Hyper Loglog for counting the distinct elements.

Several traditional and well-articulated data mining and Statistical analysis such as Linear regression in solving the linear equations in several dimensions and often use a lot of processing power to evaluate the

results and Logistic regression is also used to mine the data, General Linear regression can also be achieved using it. Dating back to 1970s, Expectation-maximization is made achievable using the Gaussian Mixture techniques. Following it K-Means, K-Medians, Generalized K-Means used to evaluate the Mahalanobis Distance. All of the statistical analysis included computation with vectors and matrices are handled with the usage of the above-mentioned techniques.

It requires large amounts of data to be transferred with almost no extra passes leading to the simple question How to achieve it? One such way is to carry the information between the iterations and maintain the constant state while scanning the data elements. Assuming that the state is stable and id allowed to alter with shouldIterate() which is true for the k-means algorithm by maintain the constant state of the cluster centers. On comparing the Data-Streaming wit GLAs, by definition it scans the data once and computes in the small space provided. By 2000s, all of the Data-streaming algorithms became to adopt the functionality that the Linear Generalized Aggregate followed. To be specific with the internals, it holds state of the data along with the functionality to additems and finalize the final result. With the intuition developed from the above facts, we can define GLA to be independent of order in which we access and provide the information, also it is distributive in nature to process quickly and evaluates to be true by performing the equivalence with it. One question come into picture related to the size of the memory is can GLA handle the scenarios where the state is larger when compared to the memory, we need to check on the same in the next class.