ORIENTAL JOURNAL OF
COMPUTER SCIENCE & TECHNOLOGY
An International Open Free Access, Peer Reviewed Research Journal
Published By: **Oriental Scientific Publishing Co., India.**
www.computerscijournal.org

ISSN: 0974-6471
June 2017,
Vol. 10, No. (2):
Pgs.271-275

Approximation Query Layer (AQLayer): Design and Architecture

TAMANNA SIDDIQUI and MOHAMMAD ALKADRI

Department Of Computer Science Aligarh Muslim University
Corresponding author Email: ja_zu_siddiqui@hotmail.com

http://dx.doi.org/10.13005/ojcst/10.02.03

(Received: May 13, 2017; Accepted: May 31, 2017)

ABSTRACT

Data Scientists need to manipulate with data (retrieve, aggregate, join, ….) - when they do their tasks - for that it will be very useful to build a layer which prepare the data to be convenient for analysing step; with an approximation processing and some error tolerant defined by the user, that layer will handle both inserting records or collections to the database and retrieving the information from that database. In this paper we will focus on the structure and the design of this layer, and dig deeper how this layer will translate the user's inquiring manner to a SQL statement suitable for approximation processing.

Keywords: Big Data, Approximation Query Processing.

INTRODUCTION

In the modern software design, almost company try to isolate development layer and database layer by using some language similar to SQL or pattern to handle this isolation task, like Active Record and Microsoft LINQ. By using active record on relational databases, the object data is stored in-memory with functions to do main tasks on database (select, insert, update and delete). Any table or view in database will be wrapped into class in active record architecture.

LINQ stands for Language-Integrated Query, it is a query language developed by Microsoft with syntax similar to SQL[1]. There are other existing LINQ like PHP (PHPLinq), ActionScript (ActionLinq), TypeScript (linq.ts), JavaScript (linq.js) but C# LINQ is the best between them because it designed as a part of the C# language for that we denote to it with LINQ. In LINQ they added some query expression which able the developer to retrieve and process the data from different sources, and some standard query operators (SQO)[2].

Decision support systems (DSS) requirements, the high query complexities and the strict response-time were the reason of rising Approximation Query Processing (AQP) where the user sacrifices the exactness to enhance the response-time[3,6]. AQP depends on synopses of

the huge amount of data to answering queries with a cost effective manner. There are three main techniques of AQP, they are[3,5]:

· Sampling-based techniques: In this technique, synopses are produced by using random samples on a huge amount of data[4]. Sampling technique can be grouped into (Uniform random sampling, Biased sampling, Icicles, Outlier indexing, Congressional sampling, Stratified sampling) depending on the selection criteria of sample data. There are many successful applications of random sampling in population surveys and selectivity estimation.

· Histogram-based techniques: The item value in this technique is divided (partitioned) into buckets, then the average frequency of the bucket will be the base for computation the frequency of items[5,7,8]. There are many types of histograms (Uniform, Equi-width, Equi-depth, End-biased, serial)[9,10] but the most used one is equi-depth[8,11].

· Wavelet-based technique: It is a mathematical tool (linear processing) for the hierarchical decomposition of functions, consists of detailed coefficients and good de-correlation features (properties). By using these coefficients with compression/decompression algorithms, it can compress and reconstruct the original data, that able it to join different kind of information in one wavelet decomposition without gaps[12,14]. This technique is very useful in image processing and signal application[3], and high-dimensional OLAP cubes. There are many families for wavelet like (Haar[12], Daubechies[15], Morlet[16], …).

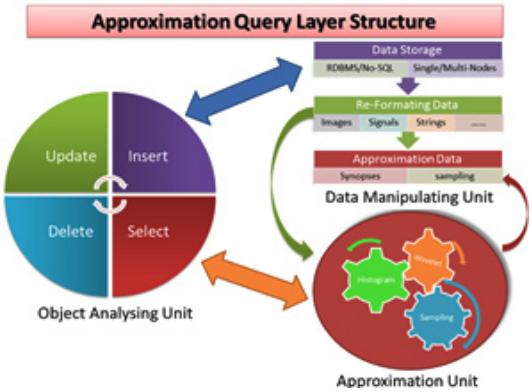

Fig. 1: Approximation Query Layer Structure

AQLayer Definition

Approximation Query Layer (AQLayer) is an isolation layer to separate between the data scientist and the database, which means enable the scientist to send an object and receive approximated or exact results, no direct SQL - or other - statement to the database or any of its elements. This layer could be included in any of the existed databases, either relational database (RDBMS) or No-SQL one. Other manner to make this layer useful is by make it like translator between the data scientist (user) and the database which this layer covers it.

This layer will be independent from the database system which is covered by it, no matter if it's relational database (RDBMS), or non-relational database (No-SQL). The user (data analyst) will prepare an object and send it through one of the main APIs of this layer (Select, Insert, Update and Delete), then this layer will convert that object with its specification to a suitable statement (SQL or No-SQL) depending on the covered database type.

In this article we will focus on relational database systems and mention No-SQL databases when it is needed. Form relational databases

```
class Customers extends DB_Parent
{
  private customer_id , first_name , last_name , company_id ;
        .....
  public Orders = array();    // array of children objects from Order class
}
```

prospective, this layer should be able to provide the process of storing, retrieving and approximating data with following features:

- Mapping tables:
- Any table should be mapped with a class.
- If there is a software or system - above this layer – with an OOP designing approach, then we should have a converter between Design classes and wrapper-classes.
- Representing any master-detail between tables.
- Select Query:
- The user should be able to select which fields he wants to return back in the result records.

Fig. 2: Insert Object Analyser

- Order the results depends on some fields.
- Sometimes users need to limit the results with Offset (start from) and Count (number of result he wants to retrieve) to manipulating with the data and check it.
- Filter the results by any collection of conditions (on both the main object and its children), these conditions can be any of:
- Equal and Not Equal.
- Greater than, Less than, Greater or equal, Less or equal, Between and Not Between.
- Contain, Not Contain, Start With, Not Start With, End With and Not End With.
- IN and Not IN.
- "Is Null" and "Is not Null".
- Specify the error tolerant (the ratio of error which the approximation result should not be more than it) that is accepted for this selection.
- Insert Data
- Add new objects (rows), whether it is a single object or parent-children objects collection.
- Produce the needed synopses data of the

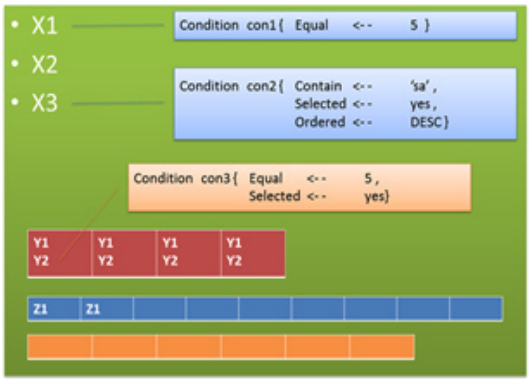

Fig. 3: Parent-Children Object with conditions

```
class Condition
 {
   public Selected, OrderBy, Limit, Between;
   public Equal, Contain, In, Grt, Less, GrtEq, LessEq, Null;
   public New_Value, StartWith, EndWith;
   public NotEqual, NotContain, NotStartWith, NotEndWith, NotIn, NotBetween;
 }
```

original data and change them according to the added and deleted records (collections of data).

Made any kind of approximation methods on data to reformat it in suitable shape, to facilitate retrieve it by appropriate manner in our approximation query system.

AQLayer Architect

This section is to demonstrate the structure of a novel layer which can be built above big data systems (like Hadoop or other systems) or as an included component in these systems, the purpose of this layer is to facilitate the communication between the data scientists and these systems when they are manipulating with them while they do their analysing tasks on the targeted data.

This layer consists of three partitions (Object Analysing, Data Manipulating, and Approximation Unit). Any table (collection) is mapped with a class; we call this class "WrapperClass", and for any master-detail relation between tables, the representation of this relation will be done by parent-children wrapper-classes in this layer, where array of the wrapper-class which represent detail table

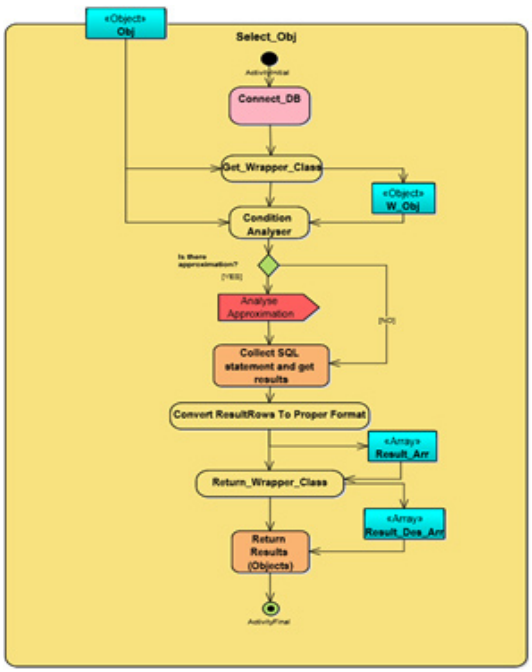

Fig. 4: Select Object Analyser

will be appended in the wrapper-class of the master table, similar to the next code:

Through this layer, the user has the ability to send single object or combination of parent object and its children objects, that will be true when inserting data or even when inquiring specific row in the database. While inserting data it is obvious that this layer will iterate same process (recursive method) on the parent object and its children object to create appropriate statement to insert these data in proper way, but in relational database systems one step more will be done, this step is providing the primary key's value of the parent object (in master table) to its children object (details tables) in their foreign key column. Figure2 illustrate the activity diagram of the recursive method for inserting data. Then the role of the approximation unit will start to create synopses or sampling data for the original data depending on the appropriate technique (sampling, histogram, or wavelet) which is chosen in this unit.

The most powerful part of this layer is selection method; this method will handle an object (collection of objects) through object analyser to create the targeted statement to retrieve appropriate results. The user should assign condition class to any data attribute after specifying the condition values for one or more of the condition class' attributes; the structure of the condition class will be like this:

Figure3 explains how the parent object can hold conditions in its attributes and also in the attribute of its children object, condition analyser will join all these condition, convert it to some suitable syntax depending on the database below AQLayer, and put it in the final statement.

In Select Object Analyser, there are many sub-routines or methods like Get_Wrapper_Class and Condition Analyser, and it communicate with Approximation Unit when the user send an object with an approximation results are desired, these activities are explained in Figure4.

CONCLUSION

In this paper we demonstrate the definition and the structure of the Approximation Query Layer which will be as a translation layer between the data users (Scientists or analysts) and the database storage systems, our explanation focus on relational database systems, but this layer is convenient for both relational and non-relational systems. As we illustrate though the paper, this layer will able the user to manipulate with data (select, insert, update, and delete) with an approximation manners to speed the process of storing and retrieving the data.

REFRENCES

1. A. Corbellini, C. Mateos, A. Zunino, D. Godoy, and S. Schiaf, "Persisting big-data/ : The NoSQL landscape," vol. **63**, pp. 1–23, 2017.
2. Microsoft, "standard_query_operators." 2007.
3. K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim, "Approximate query processing using wavelets," pp. 199–223, 2001.
4. B. Babcock, S. CHaudhuri, and G. Das, "Dynamic Sample Selection for Approximate Query Processing," 2001.
5. Y. S. Mehanna, M. Mahmuddin, and H. S. Abdelaziz, "Approximate Query Processing Concepts and Techniques," pp. 11–19, 2015.
6. S. Río, V. López, J. M. Benítez, and F. Herrera, "On the use of MapReduce for imbalanced big data using Random Forest," *Inf. Sci. (Ny)*., vol. 285, pp. 112–137, 2014.
7. A. Aboulnaga and S. CHaudhuri, "Self-tuning Histograms/ : Building Histograms Without Looking at Data," 1999.
8. H. Mousavi and C. Zaniolo, "Fast and Accurate Computation of Equi-Depth Histograms over Data Streams," 2011.
9. B. Yýldýz and B. Tolga, "Equi-depth Histogram Construction for Big Data with Quality Guarantees," pp. 1–13, 2016.
10. L. Chen and A. Dobra, "Histograms as statistical estimators for aggregate queries," *Inf. Syst.*, vol. **38**, no. 2, pp. 213–230, 2013.
11. J. Myung, J. Shim, J. Yeon, and S. Lee, "Handling data skew in join algorithms using MapReduce," vol. **51**, pp. 286–299, 2016.
12. P. Porwik and A. Lisowska, "The Haar – Wavelet Transform in Digital Image Processing/ : Its Status and Achievements," pp. 79–97, 2004.
13. M. Hartmann, "Building Wavelet Histograms on Large Data in MapReduce," no. 1, pp. 1–12, 2013.
14. M. Hariharan, C. Y. Fook, R. Sindhu, A. Hamid, and S. Yaacob, "Objective evaluation of speech dysfluencies using wavelet packet transform with sample entropy," *Digit. Signal Process.*, vol. 23, no. 3, pp. 952–959, 2013.
15. R. Maidstone, "Wavelets in a Two-Dimensional Context," pp. 1–17, 2012.
16. R. Angi, "WaveletComp/ : A guided tour through the R-package," pp. 1–38, 2014.