Proposal

One Query-Multiple Databases: An interface to mutually access Hadoop/Oracle databases

Group number: 5

Group members: Arnab Saha (asaha), Arjun Sharma (asharm24), Nishtha Garg(ngarg)

Course Name: Data Intensive Computing

Date: 09/21/2015

Abstract

In today's era of digitalism, where a number of technologies to maintain and analyze the huge amounts of data have evolved, querying and analyzing data from multiple data repositories with different data mechanics is challenging; esp. when one is RDBM system(like Oracle) and the other is NoSQL system(like HBase). For instance, if a company (which uses HBase) expands its product into a new location, by acquiring a similar product by another company (which uses Oracle) then processing the entire data using a single interface is an intricate task.

In this project, we intend to solve the aforementioned problem in two steps. As the databases pertain to the same product, they have similar information but not necessarily similar structure. So we intend to provide an automated mapping technique for all the similar attributes in both datasets using domain mapping. Secondly we aim to create a common "SQL-like" API to access data from both datasets simultaneously. The ultimate goal is to provide a cost effective alternative to the approach of migrating the entire dataset from one technology to another.

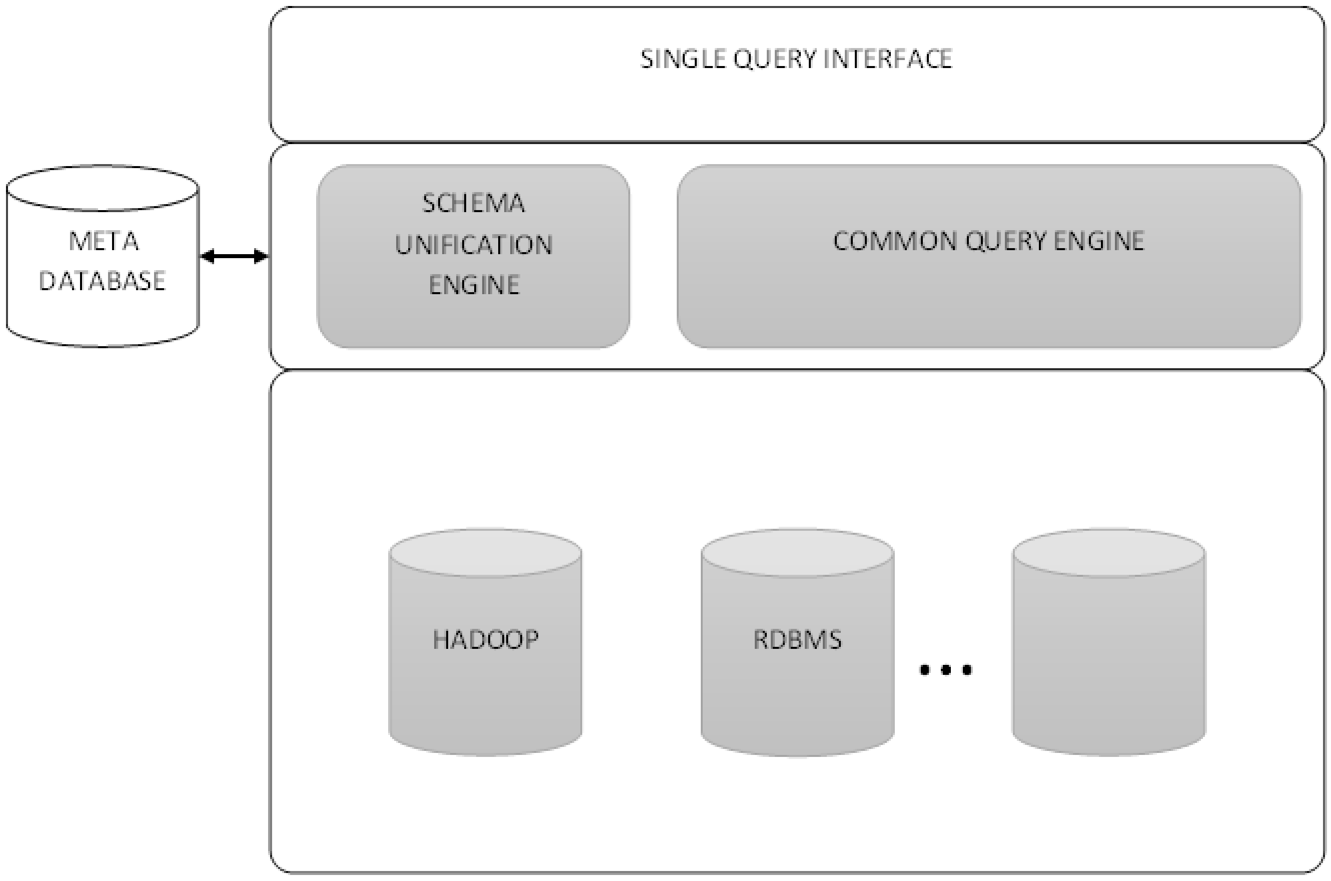
Overview

We aim at solving the problem of accessing data from heterogeneous data sources by firstly creating a single query interface (SUI) which is the abstraction point of One Query-Multiple Databases (OQMD). Users will be able to use this API to read data from various sources using a single SQL-like query. However, we need to build an execution engine using which the query can be de-multiplexed to different data sources and further results can be multiplexed back to SUI. A high-level design of the execution engine can be broken into two components:

- *Schema Unification Engine:* This component will be responsible for domain mapping i.e. identifying pairs of attributes (consisting of similar data) obtained from databases pertaining to different schemas, and storing this information in the meta-datastore.

  The engine will automate this task by taking a sample from the entire data set pertaining to k rows in the RDBMS and similarly k entries from any other non-sql system (like Hadoop). Then for each attribute in RDBMS, "patterns" of entries can be searched for, in all the attributes of the other database. This pattern may belong to a set of predefined domains as in the case of “Country” or to a generic regular expression as in the case of “Name”. This can be repeated for n samples until each attribute in RDBMS can be matched (with a certain confidence factor) to an attribute in the NoSQL system. Once matching is done, all the mapping information will be stored in the meta-data store.

- *Query Engine*: This engine will process the query and de-multiplex it further (by using mapping information from the meta-data store) to direct it to individual data sources. Further, the results from different sources will be compiled back to display in a table-like format.



Architecture of OQMD

Justification

The motivation for choosing the project is to build a novel system with marked up efficacy in all aspects and to gain knowledge in the field. Firstly we aim at providing a system which can replace the costly methods of data migration concept/ETL. Data migration is a process of transferring the data between systems or different storage formats, which is a non-trivial task.

Complexity of data migration is high which leads to cost overrun and delays with go-live. Overall this system will help to achieve the efficiency in terms of low cost and no downtime.

Main point of focus is schema unification using the automated mapping process which will help to reduce the cost and time taken by the organizations to transfer the data between the different storage systems at the time of expansion. This will also help to get rid of the problem of downtime i.e. a proportion of users at a time face the delay of services.

Also the learning process of various query languages by the user is a time and cost intensive task. So OQMD will enable users to work with just one query language across multiple data formats and not distress about learning various technologies.

Also we will be able to gain a pool of knowledge in the course of this project by learning about various technologies and database storage systems like HBase, Oracle, Big table etc.

Project Management

Goals

The ultimate goal of the project is to create a common SQL interface to access different types of databases. We have divided the project into two phases:

- *Single Query Interface*: This will have the look and feel similar to any SQL interface where you can run a Select/Update query but it is capable of fetching and updating data simultaneously from disparate databases like HBase and Oracle.
- *Schema unification*: The main challenge in using different databases to understand what an attribute in say database1 correspond to in say database2. So we are planning to develop the Schema Unification Engine which will automatically map the attributes from one database to another. For this step we are assuming that the data in both the databases are similar and therefore consist of similar attributes.

Milestones/Tasks

The goals are further divided into several milestones/tasks as follows:

- *Learning and setup phase*: This is the fundamental phase which will pave the path for the development of the project. This phase consists of learning about the architecture and SQLs of the different DBs and testing it on their native environment. *[Duration: 1 week, Depends on None]*
- *Sample data creation*: Simple testing data needs to be created on different DBs before development which will have similar schemas. *[Duration: 1 week, Depends on i]*
- *Manual mapping*: Though the final step in the development will be to achieve automation in schema unification, we will initially provide the attribute mapping manually and store it in the meta-database. *[Duration: 3 weeks, Depends on ii]*

- *Query interface creation*: This step marks the end of Phase I. At the end of this phase we will be able to fetch/update data from multiple databases with the same query. The data though spread across various databases will behave like sharded data on single database. *[Duration: 3 weeks parallel with iii, Depends on ii]*
- *Automated mapping*: The principal part of the project is to develop Schema unification Engine which will automate the mapping of the attributes across diverse databases. *[Duration: 1 month 1 week, Depends on iv]*
- *Testing and efficiency improvement*: Our intention is to do these two steps in parallel to improve our efficiency. While our primary focus is to improve the efficiency of the schema unification engine, we plan to reduce the bugs in the latest version. *[Duration: 1 month 1 week parallel with v, Depends on v]*

Verification

The first phase of the project is to develop Single Query Interface using manual mapping. We will test it with five test data ranging from simple schemas to complex schemas (We are yet to decide on the structure of the schemas). Once it runs successfully on the five test data we will move on to the second part of the project.

The second phase of the project deals with automated mapping across different databases. As domain profiling and mapping is not foolproof, we are not certain about the efficiency we might be able to achieve. Like most real world problems, absolute optimality in automated mapping might not be achieved thereby we are applying Pareto optimality for the requirements.

Background

With the advent of NoSQL technologies most Data Warehouse practitioners agree that both DBMS and NoSQL paradigms have advantages and disadvantages for various business applications and thus both paradigms must co-exist [1]. In order for this co-existence to work ETL(Extract, Transform and Load), the process of extracting data from various sources and storing it in another system , is the primary mechanism which has since been used. A number of works relevant to this range from extraction and storage of data using manual mapping of schemas to transformation and storage of the extracted data using automated schema mapping procedures.

Cupid [2] has been applied for mapping XML as well as relational data models. For two given schema, the system does an initial linguistic matching and then does a bottom-up structure matching to determine similar attributes. COMA [3] is a schema mapping system that relies on a large database of individual matching and a set of rules to govern them. It also reuses results from previous match operations. MapOnto [4] is a semi-automatic tool that assists users to discover possible relationships between database schemas.

Bibliography

[1] Stonebraker, Michael, et al. "MapReduce and parallel DBMSs: friends or foes?." Communications of the ACM 53.1 (2010): 64-71.

[2] Madhavan, Jayant, Philip A. Bernstein, and Erhard Rahm. "Generic schema matching with cupid." VLDB. Vol. 1. 2001.

[3] Do, Hong-Hai, and Erhard Rahm. "COMA: a system for flexible combination of schema matching approaches." Proceedings of the 28th international conference on Very Large Data Bases. VLDB Endowment, 2002.

[4] An, Yuan, Alex Borgida, and John Mylopoulos. "Inferring complex semantic mappings between relational tables and ontologies from simple correspondences." On the move to meaningful internet systems 2005: CoopIS, DOA, and ODBASE. Springer Berlin Heidelberg, 2005. 1152-1169.