Demonstration of the CloudMdsQL Multistore System
Boyan Kolev, Carlyna Bondiombouy, Patrick Valduriez, Ricardo Jiménez-Peris, Raquel Spain, José Pereira

To cite this version:

HAL Id: lirmm-01408621
https://hal-lirmm.ccsd.cnrs.fr/lirmm-01408621
Submitted on 5 Dec 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License
The blooming of different cloud data management infrastructures, specialized for different kinds of data and tasks, has led to a wide diversification of DBMS interfaces and the loss of a common programming paradigm. This has turned multistore systems to a major topic in the nowadays cloud landscape.

In this demonstration, we present Cloud multidatabase query language (CloudMdsQL) [1], a functional SQL-like language, designed for querying multiple heterogeneous databases (e.g. relational and NoSQL) within a single query containing nested subqueries. Each subquery addresses directly a particular data store and may contain embedded invocations to the data store’s native query interface. Thus, the major innovation is that a CloudMdsQL query can exploit the full power of local data stores, by simply allowing some local data store native queries to be called as functions, and at the same time be optimized based on a simple cost model, e.g. by pushing down select predicates or using bind join.

One of the major challenges in front of the CloudMdsQL language/engine is to allow joins across heterogeneous data stores, by pushing down select predicates or using bind join.

This demonstration concentrates on a CloudMdsQL use case scenario: a social network analysis tool for marketing companies. The use case aims at finding the communities in a social network, for a specific set of topics, with their top influencers. Marketing companies are interested in discovering the people they need to convince about the quality of a specific brand. The dataset of this use case is a sample of Twitter, but it allows working with other social networks like Facebook or blogs. The application runs a Twitter listener of a set of topics in real-time; it modifies the database for each tweet it receives. The schema of this application contains a generic entity called Document to store text-items (tweets, messages, etc.), which can appear relevant to the keywords *k*.

For the execution plan, the query optimization plays an important role to assign the bind join method to all the join operations. The reason is that the selected communities relevant to the keywords *k1*, *k2* and *k3* are always a few, and thus the Sparksee query is evaluated only for a few communities, which significantly reduces the number of executions of expensive graph computations.

1. REFERENCES