CoherentPaaS

Coherent and Rich PaaS with a
Common Programming Model

ICT FP7-611068

Query Compiler for the Common Query v2 (Full Functionality)

D3.6

<February, 2016>

Document Information

Scheduled delivery	24.02.2016
Actual delivery	
Version	2.2
Responsible Partner	INRIA

Dissemination Level:

RE Restricted

Revision History

Date	Editor	Status	Version	Changes
05.07.2015	B. Kolev	Draft	0.1	Initial version
16.09.2015	B. Kolev	Revised	1.0	Addressed internal reviewers comments
28.01.2016	B. Kolev	Revised	2.0	Addressed reviewers comments: - Merged contents with D3.5 - Quality of deliverables
28.01.2016	Y. Zhang	Reviewed	2.1	Added structure to the introduction
23.02.2016	M. Patiño	Reviewed	2.2	Improved quality of the deliverable Made the deliverable self-contained

Contributors

Boyan Kolev (INRIA), Patrick Valduriez (INRIA)

Internal Reviewers

Marta Patiño (UPM), Ying Zhang (MonetDB)

Acknowledgements

Research partially funded by EC 7th Framework Programme FP7/2007-2013 under grant agreement n° 611068.

More information

Additional information and public deliverables of CoherentPaaS can be found at: http://coherentpaas.eu

1. Executive Summary

CloudMdsQL (Cloud Multidatastore Query Language) is the common query language of CoherentPaaS. It is a functional SQL-like language, capable of querying multiple cloud data stores (SQL, NoSQL, HDFS, etc.) within a single query that contains embedded invocations to each data store's native query interface. The query compiler parses a CloudMdsQL query and generates a query execution plan to be processed by the query operator engine.

The final version of the compiler, presented in this deliverable, implements the full functionality for processing CloudMdsQL queries. The compiler parses a query first into an abstract syntax tree (AST), then it transforms the AST into a query execution plan (QEP), represented as a directed acyclic graph, where leaf nodes are references to named tables and all other nodes represent relational algebra operations. The output of the query compilation is the JSON serialization of the generated QEP, which is further handled by the common query engine.

Upon decomposition of a SELECT query, each named table is mapped to a sub-plan, which is represented as a relational algebra tree, where the leaf nodes may be references to other named tables (for nested SQL sub-queries), references to data store tables (for named table expressions against SQL data stores), or native/Python expression definitions (for native/Python named table expressions). Further semantic analysis includes resolution of names of tables and columns and analyses of datatypes, cross-references, and WHERE clause predicates. The WHERE clause analysis may result in pushing selection operations down into the sub-plans for data stores, the executability of which is then validated against each data store's capability specification.

The compilation of data manipulation queries is also implemented. It allows for data, retrieved from one or more data stores, to be used for modification of data in one or more other data stores. Data manipulation queries make use of named action expressions to request modification of data in the data stores (e.g. insert, update, delete).

This deliverable also evaluates the performance of the query compiler using a micro-benchmarking that measures the duration of each phase of the compilation process for three queries of different complexity.