ISSN 2186-7437

NII Shonan Meeting Report

No. 2017-6

Language-integrated queries: toward standard logics for big analytics

Laurent Daynès
George Fletcher
Wook Shin Han

May 29 - June 1, 2017

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo, Japan

Language-integrated queries: toward standard logics for big analytics

Organizers:
Laurent Daynès (Oracle Labs, France)
George Fletcher (TU Eindhoven, Netherlands)
Wook Shin Han (Pohang University of Science and Technology, South Korea)

May 29 - June 1, 2017

1 Introduction

Database management systems (DBMSs) are typically optimized for a particular data model (e.g., relational, semi-structured, graph-based) and interfaced with a unique query language (e.g., SQL, HiveQL, XQuery, JSONiq, SPARQL). In contrast, database applications are written in general-purpose programming languages that offer developers a large choice of libraries (e.g., to simplify presentation to the end-users, the writing of business logic, etc.).

For various architectural reasons, database applications execute in an environment distinct from that of the DBMS, i.e., on a client machine (e.g., a connected mobile device) or on a middle-tier, or even within the database itself. This situation causes two main problems that have been the focus of research for several decades: (1) how to better integrate database querying with application programming languages to eliminate impedance mismatch while retaining the full power of the database querying capabilities; and (2), how to minimize the traffic, both in terms of number of interactions and volume of data exchanged, between the database and its applications.

1.1 Current State of the art

With respect to integrating querying with programming languages, the industrial landscape is currently dominated by ORM solutions (Hibernate, Ruby-on-Rails, SQLAlchemy, DJango, Propel, RedBeanPHP, to name the most prominent). These frameworks are based on popular architectural patterns (e.g., active records, data mappers) that greatly simplify application development by wrapping database operations in type-safe object-oriented interfaces, allowing developers to write code completely in the host language.

Unfortunately, these solutions encourage developers to write code that iterates over collections of objects representing database records using idioms of the language to perform bulk operations like filters and joins that would be better done by the database. This often results in poor performance as it both increases traffic with the database and can overwhelm the application?s memory with very large intermediate results.

To avoid these problems, a number of language-integrated query techniques for embedding queries into general-purpose programming languages have emerged. These techniques seek to reconcile the goals of type-safety, uniform programming idioms, on one hand, and better capturing of querying intent to optimize interactions with databases, on the other. Two directions are being investigated: (1) use some form of static analysis or type system to identify part of programs that can be turned into queries; (2), extend conventional language with explicit quotation or surface syntax for expressing queries more directly.

This second approach has gained popularity with Microsoft?s LINQ which offers programmers a unified API for querying arbitrary data providers and supports facilities to extend the syntax of the host language to add query constructs.

Unfortunately, this approach is too restricted for a couple of reasons. First, although queries appear integrated syntactically, a driver still needs to translate the query into a form that can be shipped for execution at the back-end where the actual data resides. In practice, the query is just translated back to SQL query. Consequently, deep integration of language expressions and queries is

missing, and complex queries featuring user-defined functions often require several round-trips between the database and the runtime to exchange intermediate results.

Second, a heavy burden is put on the data provider designer, who has to resort to either (1) developing a superficial provider, that only implements the most basic query primitives or, (2), investing a considerable amount of time, effort and expertise in developing a sophisticated data provider capable of analyzing the syntactic representation of queries and of translating it into one or more requests that can be executed by the back-end. In all cases, sub-expressions from the host language that participate in the query must be translated into an equivalent expression in the interface of the data provider (e.g., SQL). This translation may not always be possible, or requires a substantial amount of work, such as, providing an equivalent stored procedure at the database side for all user-defined functions used in the application queries. When this isn?t possible, a complex query expression must be split into multiple queries and intermediate results must be materialized at the application side in order to apply the host language?s sub-expressions. This is a trait shared by all of the solutions mentioned above, and one that cannot be solved as long as data providers fail to offer a querying interface that can accept foreign language expressions.

1.2 Needs for an Algebra for data analytics?

Since its introduction in the 1970's, Codd's relational algebra (RA) has served as an indispensable workhorse in the engineering of relational database systems. As the mediating layer between specification of queries by clients in their host language, on one hand, and compilation of optimized physical query execution plans, on the other, the RA is arguably one of the key technologies which led to the rise of practical data management solutions in the 1980's. Generalizations and extensions of RA played an analogous role in the 1990?s and 2000?s, to address new challenges arising, for example, in the management of object-based and semi-structured data collections.

In the last decade, we have witnessed a continued explosion of research and development of data intensive systems and languages for big data analytics. These range, for example, from distributed computing frameworks such Apache Spark and Apache Flink to document-centric data stores such as MongoDB or Microsoft Azure DocumentDB, to acceleration engine for graph data analytics like PGX. To bridge the gap between the specification of analytic tasks by clients of these systems, on one hand, and compilation of optimized execution plans, on the other, an analogue of the relational algebra for big data analytics processing is called for. Although each of the systems in the contemporary data engineering landscape to some degree realizes its own flavor of a query algebra, there is currently no recognized logical language which serves this role. Recent efforts such as Apache Calcite are a step in the right direction, but are still focused on the relational paradigm.

As discussed above, on the other end of the spectrum there have been efforts on integrating native data querying capabilities into languages (aka language integrated querying) such as .NET, Java, PHP, and JavaScript. Such efforts extend the various languages by the addition of query operators and expressions which often go beyond the expressiveness of the relational algebra.

A broad community discussion of the features and design of extended algebras for big data analytics, as integrated in general-purpose programming languages, is crucial to bring big data analytics solutions to the next level of maturity.

As data processing platforms equip themselves with support for new programming languages (e.g., JavaScript support in Postgres and Microsoft Azure DocumentDB; Python support in Amazon Redshift; R in Oracle, Microsoft SQL server, IBM dashDB, SPARK, etc.), the need for a data provider interface capable of accepting multi-lingual expressions in queries regardless of particular query syntax will only continue to grow in importance.

1.3 Workshop organization and outcomes

The goal of this meeting is to take the first steps towards elaborating solutions for (1) a standard language-, data-model-, and platform-independent declarative interface to data providers which is able to leverage available multi-lingual capabilities of data providers; and, (2) corresponding compilation and execution strategies. For this broad discussion, we aimed to bring together relevant leading researchers from both academia and industry, across the domains of programming languages, data management systems, and distributed and parallel systems specialized in data processing (e.g., Graph Analytics, Machine Learning, etc.).

The workshop was organized in two parts: a first part in which selected participants were invited to present their views and experiences on a particular aspect of the problem space; the second part was organized into working groups that discussed in depth several key topics. Additional impromptu presentations where given throughout the workshop. Three working groups formed to address the following topics:

1. How should a language agnostic intermediate representation (IR) for data analytics programs look like? How to compile such an IR into a form executable over an heterogeneous, multi-model data processing architecture;

2. Remaining challenges for language-integrated querying, taking into account the need for integrating new forms of data processing (e.g., machine learning) and new data models (e.g., graph).

3. What is the state of end-user interfaces for data analytics and can we better support data exploration.

Notes from sessions are available from the workshop web-site in the form of google doc. One main outcome from these groups is a decision from several participants to initiate collaborative work, starting with the writing of a Horizon 2020 EU grant proposal in the area of big Data technologies and extreme-scale analytics. A second outcome is a collaborative effort to write an up to date survey in the form of a tutorial on the state of language-integrated query techniques that can be presented in upcoming important main database research venues (VLDB, SIGMOD).

Overview of Talks

Chimera are useful ? Embedding scripting languages in data management systems and vice-versa

Hannes Mühleisen, CWI

Data Science takes place in specialized scripting environments such as Python or R. These environments were never designed to handle huge datasets, but are now routinely used for precisely that task. Data management is an issue, reading large dataset from flat files or through a socket connection involves huge overheads and thus increases the time-to-analysis to impracticable amounts. In this talk, we will discuss the concepts and innovations required to embed complex data management systems into a statistical scripting environment.

Strymon: Queryable Online Simulation for Modern Datacenters

Vasiliki Kalavri, ETH Zurich

A modern enterprise datacenter is a complex, multi-layered system whose components often interact in unpredictable ways. Yet, to keep operational costs low and maximize efficiency, we would like to foresee the impact of changing workloads, updating configurations, modifying policies, or deploying new services. In this talk, I will share our research group?s vision for Strymon; a multi-purpose platform that aims to provide a cross-layer model for modern datacenters and support automatic reconfiguration through online simulation. Driven by a real-use case from our industrial partners, I will highlight the need for Strymon to support a diverse set of data representations, input sources, query languages, and execution models. Finally, I will share our initial design decisions and give an overview of Timely Dataflow; our platform of choice for Strymon?s core implementation.

Language-integrated query: state of the art and open problems

James Cheney, University of Edinburgh

In this talk I will give a brief and broad overview of approaches to incorporating queries (or other forms of heterogeneous computation) into programming languages, focusing on three general strategies:

- using query operator APIs (e.g. .NET LINQ operators, Delite, FlumeJava),
- directly embedding the syntax of SQL or other query languages (e.g. C#, SML#, Ur/Web),
- overloading language constructs or extending the language with query operations (e.g. Kleisli, F#, Database Supported Haskell, or Links)

and comparing and contrasting them, briefly highlighting areas for future research.

Managing the Proliferation Problem – Oracle Labs perspective on language-integrated querying & data analytics

Hassan Chafi, Oracle

SQL Extension for Complex OR Mapping in HTAP,

Kihong Kim, SAP

OR mapping of analytical objects to HTAP database tables often leads to complex relational models, consisting of thousands of joins and unions, and thus efficient query processing is very difficult. However, since humans can consume a small number of values at a time, each analysis report requires a subset of such complex objects and thus many of the joins can be eliminated per query basis. This work presents SQL extensions to easily simplify complex OR mapping models from SAP applications.

A new DAWN for data analysis

Kunle Olukotun (Stanford)

DAWN is a new Stanford lab to create infrastructure for usable machine learning. We observe that ML algorithms are now ?good enough? for many applications, but the bottlenecks to real-world use are tasks around the algorithm, such as data labeling, data augmentation, and robust serving. We are developing runtimes, algorithms and serving systems to tackle these problems.

A principled approach to language integrated queries

Kim Nguyen (LRI, Universit Paris-Sud)

BOLDR is a modular framework that enables the evaluation of queries containing application logic (and, in particular, user-defined functions) in databases. BOLDR detects the boundaries of queries present in an application, translates them into an intermediate representation together with the relevant language environment needed to evaluate them, rewrites them in order to avoid query avalanches and to make the most out of database optimizations, and converts the database(s) results back to the application.

Flare: Native Compilation for Heterogeneous Workload

Tiark Rompf (Purdue University)

The need for modern data analytics to combine relational, procedural, and map-reduce-style functional processing is widely recognized. State-of-the-art systems like Spark have added SQL front-ends and relational query optimization, which promise an increase in expressiveness and performance. But how good are these extensions at extracting high performance from modern hardware platforms? While Spark has made impressive progress, we show that for relational workloads, there is still a significant gap compared with best-of-breed query engines. And when stepping outside of the relational world, query optimization techniques are ineffective if large parts of a computation have to be treated as user-defined functions (UDFs). We present Flare: a new back-end for Spark that brings performance closer to the best SQL engines, without giving up the added expressiveness of Spark. We demonstrate order of magnitude speedups both for relational workloads such as TPC-H, as well as for a range of machine learning kernels that combine relational and iterative functional processing. Flare achieves these results through (1) compilation to native code, (2) replacing parts of the Spark runtime system, and (3) extending the scope of optimization and code generation to large classes of UDFs.

Modeling Ice on Windmills: a use-case for multi-model / multi-engine data processing

Wolfgang Lehner (TU Dresden)

List of Participants

- Laurent Daynès, Oracle Labs
- Prof. George Fletcher, Eindhoven University of Technology
- Prof. Wook-Shin Han, Pohang University of Science and Technology
- Prof. Toshiyuki Amagasa, University of Tsukuba
- Dr. Sihem Amer-Yahia, CNRS
- Prof. Kim Nguyen, Université Paris-Sud
- Dr. Matthias Brantner, Oracle Labs
- Dr. Hassan Chafi, Oracle Labs
- Prof. Chee-Yong Chan, NUS
- Prof. James Cheney, University of Edinburgh
- Prof. Adam Chlipala, MIT
- Prof. Sebastian Erdweg, TU Delft

- Dr. Sungpack Hong, Oracle Labs
- Prof. Wolfgang Lehner, TU Dresden
- Dr. Sebastian Maneth, University of Bremen
- Prof. Ioana Manolescu, INRIA
- Prof. Guido Moerkotte, University of Mannheim
- Dr. Hannes Mühleisen, CWI
- Prof. Makoto Onizuka, Osaka University
- Prof. Tiark Rompf, Purdue University
- Prof. Eelco Visser, TU Delft
- Prof. Nikolay Yakovets, Eindhoven University of Technology
- Dr. Giuseppe Castagna, CNRS - Université Paris Diderot
- Dr. Vasiliki Kalavri, ETH Zurich
- Dr. Parke Godfrey, York University
- Dr. Shin-ichiro Okamoto, Yahoo Japan
- Dr. Kihong Kim, SAP Labs Korea
- Prof. Kunle Olukotun, Stanford University
- Prof. Yuqing Melanie Wu, Pomona College
- Mr./Ms. Yukyoung Lee, Pohang University of Science and Technology (POSTECH)

Meeting Schedule

Check-in Day: May 28 (Sun)

- Welcome Banquet

Day 1: May 29 (Mon)

- Introduction movie of NII Shonan meeting
- Workshop introduction by organizers
- "Chimera are useful: Embedding scripting languages in data management systems and vice-versa", Hannes Mühleisen (CWI)
- "Strymon: Queryable Online Simulation for Modern Datacenters", Vasiliki Kalavri (ETH Zurich)
- "Language-integrated query: state of the art and open problems?, James Cheney (University of Edinburgh)
- "Managing the Proliferation Problem – Oracle Labs perspective on language-integrated querying & data analytics", Hassan Chafi (Oracle)
- "SQL Extension for Complex OR Mapping in HTAP", Kihong Kim (SAP)
- "A new DAWN for data analysis", Kunle Olukotun (Stanford)
- Demonstrations / Posters
  - *"Gelly-Stream: Continuous Single-Pass and Iterative Graph Processing on Unbounded Data"*, Vasiliki Kalavri
  - *"Language-Integrated Query in a Proof Assistant"*, Adam Chlipala
  - *"Deep embedding of JavaScript in the Oracle Database"*, Matthias Brantner, Laurent Daynès
  - *"IceDust 2: Derived Bidirectional Relations and Calculation Strategy Composition"*, Eelco Visser
  - *"Integrating BD-DSLs into Scala"*, Wolfgang Lehner
- Organizing Working Groups

Day 2: May 30 (Tue)

- Day 2 introduction
- Working groups session 1
- Plenary progress report
- Working groups session 2
- Working groups end of day report

Day3: May 31 (Wed)

- Day 3 Introduction

- "A principled approach to language integrated queries", Kim NGuyen (LRI, Université Paris-Sud)
- "Flare: Native Compilation for Heterogeneous Workloads in Apache Spark?, Tiark Rompf (Purdue University)
- Working Groups session 3
- Excursion and Main Banquet

Day4: June 01 (Thu)

- Day 3 introduction
- Working Groups Wrap up