Enabling Global Big Data Computations

Damianos Chatziantoniou, Associate Professor (Presenter)
Panos Louridas, Associate Professor
Dept. of Management Science and Technology
Athens University of Economics and Business

Outline

- Introduction
- Motivating Example
- Concepts, Theoretical Framework
- DataMingler , A Mediator Tool for Big Data
- Conclusions

Until Recently…

- Relational systems were ubiquitous, everything was modeled as a relational database, in practice, no other data models existed (since mid-90s)
- SQL was the only data manipulation language – the output was always a relation
- Everyone and everything was retrieving and updating a relational database (through ODBC)
- Data Integration == Data warehousing (i.e. extract data from data sources and transform/clean/integrate into a new relational schema)

However, Once Upon A Time…

- Relational systems was not ubiquitous and other data models existed (and used) – network, hierarchical, object-oriented
- Even relational systems and SQL greatly varied from vendor to vendor
- Federation, mediators, virtual databases, interoperability, connectivity were popular terms and hot research topics. Data Integration was associated to these.

Big Data Era – One Size Fits All is Gone!

- New applications require data management systems implementing different data models:
  - Key-value (Redis), graph (Neo4j), semi-structured (MongoDB)
- *Different* data models ➔ *Different* query languages, producing results in different formats
  - SQL, APIs, Javascript, Cypher
- Programs such as Python/R or CEP engines manipulate structured/unstructured/stream data and produce output, in different formats too

➔ High heterogeneity in data manipulation tasks

Research Questions

- How one can represent/standardize the output of all the previous data manipulation tasks in order to use it in some query formulation?
- How one can intelligently/efficiently organize these data manipulation tasks into one conceptual schema?
- Beckman Report challenges:
  - Coping with diversity in the data mgmt landscape
  - End-to-end processing and understanding of data

High Level Goals

- Provide an easy to use conceptual schema enterprise's (and beyond) data infrastructure in order to:
  - make data preparation easier for the analyst
  - hide systems' specifics and data heterogeneity
  - allow the simple expression of dataframes (for data mining):
    - involving transformations and aggregations in different PLs
    - an efficient and optimizable algebraic framework for evaluation
  - offer better data governance
  - share/export/join parts of the schema to global schemata, ability to "crawl" the schema for automated feature discovery
  - contribute to end-to-end processing

Motivating Example: Churn Prediction (1)

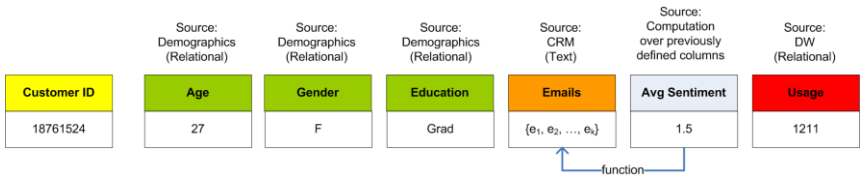
- Churn Prediction at Hellenic Telecom Organization
  - first big data project at HTO (end of 2014)
  - implementations so far involved only structured data
  - goal was to use both structured and unstructured data
  - a predictive model had to be designed and implemented taking into account the many possible variables (features) characterizing the customer – structured and unstructured

Motivating Example: Churn Prediction (2)

- Possible data sources
  - a traditional RDBMS containing customers' demographics
  - a relational data warehouse storing billing, usage, traffic
  - flat files produced by statistical packages such as SAS and SPSS, containing pre-computed measures per contract key
  - CRM data containing metadata of customer-agent interactions, including agent's notes (text) on the call
  - email correspondence between customers and the customer service center of the company (text)
  - audio files stored in the file system, containing conversations between customers and agents (audio)
  - measures on the graph of who is calling who

Motivating Example: Churn Prediction (3)

- The (data management) goal was to equip the data analyst with a simple tool that enables fast and interactive experimentation
  - select *easily* features from multiple data sources
  - define transformations and aggregations over these, possibly using different query/programming languages for each
  - combine *efficiently* into a tabular structure (a dataframe) to feed some learning algorithm



Features - Requirements

- Provide a set of “features” to the business analyst
  - Each feature is associated with an entity ➔ notion of the key
  - Features should be somehow organized ➔ conceptual model
  - Features should be generated using different DM systems and programming languages in a standardized manner
  - One or more features could be transformed to another feature, using some computational process in any programming language and well-defined semantics ➔ algebra over features
  - Features should exist anywhere, locally/remotely, and should be easily accessible (addressable), participating in global schemas
  - The “outer join” of a set of features defined over the same entity (= same key) is a **dataframe** (which is also a feature)

KL-Columns – Definition (1)

- A KL-column is a collection of (key, list) pairs

  $A = \{(k, L_k): k \in K\}$

- Examples:

CustID	Emails
162518	[text1, text2, …]
526512	[text1, text2, …]

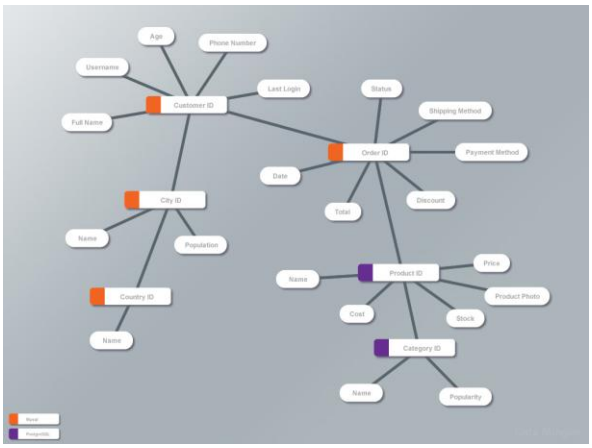
CustID	Age
162518	[25]
526512	[48]

- A KL-column is essentially a multimap, where values mapped to a key are organized as a list

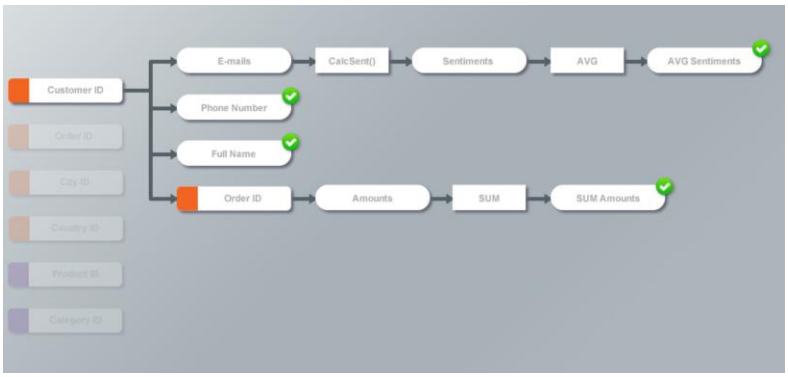
KL-Columns – Definition (2)

- A KL-column will be used to denote a Feature
- A KL-column will be populated by key-value computations, a stream of (key, value) pairs (mapping)
- A dataframe will be the “outer join” of KL-columns
- Columns may be distributed among different machines. That means that a dataframe can comprise data residing in different machines, and the data is joined on the fly to create an integrated dataframe
- One can define several operators over KL-columns, forming an algebra (e.g. selection, reduce, apply, union)

DataMingler Tool: Data Canvas



DataMingler Tool: Query Formulation



Conclusions

- We know how to store, process, analyze big data – in an ad hoc, individual manner
- We do not know how to *manage/model* big data infrastructures
- A conceptual schema, a mediator, could be the answer
- Analysts work on that layer to form input for machine learning algorithms and visualization tasks, to see stream data, to share features, to define access rights ➔ data governance, end-to-end processing