A Tensor Based Data Model for Polystore

An Application to Social Networks Data

Éric Leclercq
LE2I - EA 7508 - University of Bourgogne
9, Avenue Alain Savary
F-21078
Dijon, France
eric.leclercq@u-bourgogne.fr

Marinette Savonnet
LE2I - EA 7508 - University of Bourgogne
9, Avenue Alain Savary
F-21078
Dijon, France
marinette.savonnet@u-bourgogne.fr

ABSTRACT

In this article, we show how the mathematical object tensor can be used to build a multi-paradigm model for the storage of social data in data warehouses. From an architectural point of view, our approach allows to link different storage systems (polystore) and limits the impact of ETL tools performing model transformations required to feed different analysis algorithms. Therefore, systems can take advantage of multiple data models both in terms of query execution performance and the semantic expressiveness of data representation. The proposed model allows to reach the logical independence between data and programs implementing analysis algorithms. With a concrete case study on message virality on Twitter during the French presidential election of 2017, we highlight some of the contributions of our model.

CCS CONCEPTS

• **Information systems → Data model extensions**; *Information integration*; *Distributed storage*;

KEYWORDS

Polystore, Multi-paradigm Storage, OLAP, Tensor, Associative Array, Multi-relational Networks

ACM Reference Format:
Éric Leclercq and Marinette Savonnet. 2018. A Tensor Based Data Model for Polystore: An Application to Social Networks Data. In *IDEAS 2018: 22nd International Database Engineering & Applications Symposium, June 18–20, 2018, Villa San Giovanni, Italy.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3216122.3216152

1 INTRODUCTION

Data from social networks, especially those of Twitter, are increasingly used in applied research projects, in social sciences for example. These data, rich in information about interactions among individuals, allow researchers to understand the digital society's communication models and the interactions between digital social networks, traditional media and citizens. Results of these researches are relevant to many fields such as marketing, journalism, public policies study or political communication, as well as reactions to health crises, environmental issues, etc. However, to address their research questions, social scientists need: 1) to gain control over the data, namely to contextualize them; 2) to analyze selected data using several algorithms, each puts light on some aspects of the question and; 3) to interpret their results according their knowledge on the subject. For example, the study of political communication on Twitter requires the understanding of viral phenomena, the spread of fake-news and also the role of bots in the dissemination of information.

Several types of algorithms can be used, for example, to detect communities [14], events [3], influential users [2, 41], to simulate or study message propagation [20]. Algorithms hinge on various data models such as graphs, adjacency matrices, multidimensional arrays, time series. In addition, algorithms do not use the data in the same way, for example graph algorithms can optimize a function and/or perform a random walk on the graph, or detect the edges in a graph through which the number of shortest path between a pair of nodes is the most important (see figure 1).

Recent algorithms for social data analysis are rarely implemented in DBMS and matrix operations and associated factorizations (LU, SVD, CUR, etc.) [22, 32] are not directly supported by storage systems. Only a few NoSQL systems like Neo4j offer quite advanced data graph analysis tools. However, Neo4j does not allow to manage very large amount of data with attributes as the column-oriented systems would do [23]. The situation is almost similar for machine learning algorithms and tools. Only some recent systems such as Vertica[1], SAP HANA platform and its Predictive Analysis Library (PAL)[2], MLib in Apache Spark[3], or SciDB[4] support standard machine learning algorithms as black-box. Their data models are close to notion of relations and therefore the integration of new machine learning algorithms is complex. On the other hand some libraries such as Tensor-Flow[5] or Theano[6] have been developed to design machine learning tools using data structures close to algorithms. As a results these systems require to develop complex, hard to reuse and often error-prone programs for loading and transforming data.

In recent years we have witnessed the convergence of two separate research fields: High Performance Computing (HPC) and databases towards Data Intensive HPC. One of the concerns of the

IDEAS 2018, June 18–20, 2018, Villa San Giovanni, Italy

ACM ISBN 978-1-4503-6527-7/18/06.
https://doi.org/10.1145/3216122.3216152

[1] https://www.vertica.com/product/database-machine-learning/
[2] https://www.sap.com/community/topic/hana.html
[3] https://spark.apache.org/mllib/
[4] https://www.paradigm4.com/
[5] https://www.tensorflow.org/
[6] http://deeplearning.net/software/theano/

Data Intensive HPC is to be able to quickly feed the algorithms with data, as a result, some approaches try to combined several types of storage systems (HDFS distributed file systems, column-oriented databases, etc.) to build an efficient multi-paradigm storage system also called *multistore*, *polystore* or *polyglot storage* [17]. In such systems, data can be partitioned and stored in the model that best fits both the data structure and the algorithms required for their analysis; a partial duplication is also possible. Works on polystores focus mainly on the unification of access by using a common language, few systems propose a model-based approach and try to achieve physical data independence by using associative array [24, 26].

Our objective is to carry out logical data independence in an expressive model that can link models of different data stores while simplifying models transformations and gain performance by leveraging of different systems for computation that there are design to. Our contribution is the definition of a data model based on tensors for which we add the notions of typed schema using associative arrays. We show how the model constructs take place in a mediator/wrapper like architecture. We also define a set of manipulation and analysis operators on tensors.

The remainder of the paper is organized as follows. While section 2 discusses about OLAP and multi-paradigm storage systems including multistore, polyglot systems and polystores, sections 3, 4 and 5 present the software architecture and the tensor based data model and its operators. Section 6 presents different experiments and results obtained in context of TEP 2017 project which studies of the use of Twitter by candidates at the french presidential election in 2017.

2 STATE OF THE ART

In this section, we describe different approaches to social data warehousing: OLAP (*OnLine Analytical Processing*) data model and multi-paradigm storage. We end-up this section with a discussion of the strengths and weaknesses of these approaches.

2.1 OLAP Model and Systems for Social Data

Since 2010, several works have proposed a multidimensional star schema for building repositories of tweets. Some works are generic and others are directed towards specific analysis.

In [8] the authors have proposed an adapted measure of TF-IDF, where the most significant words are identified according to levels of dimensions in a cube. Their case study deals with evolution of diseases. Other works [37], [36] have built a warehouse dedicated to the sentiment analysis of tweets with a specific schema.

In [31, 35, 40], conceptual models for Twitter data from both OLTP and OLAP point of views are proposed. Models focus mainly on the relationships between tweets and users or among tweets (i.e., retweet, response).

In [11], the authors highlight the need to contextualize the data to help analysis. Enrichment requires a formal knowledge of the domain and is usually dependent on the purpose of analyses. It can be done by linking data to ontology terms and/or by using exploratory analyzes that characterize the data.

In a seminal paper in 2008 [9] the authors describe the *Graph OLAP* approach. They show that traditional OLAP technologies cannot handle efficiently network data analysis because they do not consider links among individual data tuples. They have developed a Graph OLAP framework to define multi-dimensional and multi-level views over graphs. Given a network dataset with nodes and edges associated to different attributes, a multi-dimensional model can be built so that any portions of a graph can be generalized/specialized dynamically, offering versatile views of the data. For example, from a citation graph the operation *roll-up* will produce a graph of institutions according to the author dimension. Favre et al. [16] propose another approach which consists in enriching the graph by using cubes associated to nodes and edges.

2.2 Multi-paradigm Data Storage

Ghosh states in [19] that storing data the way it is used in an application simplifies programming and makes it easier to decentralize data processing. Data storage and processing systems span over several families: relational, NoSQL, array, and distributed file systems. However, transforming various data into a single model may have a significant impact on performance of queries but also on capabilities to apply different algorithms. As stated by Stonebraker in [45**?**] "one size fits all" is not a solution for modern applications. As a result, several research projects have been inspired by previous work on distributed databases [39] in order to take advantage of a federation of specialized storage systems with different models[7]. Multi-paradigm data storage relies on multiple data storage technologies, chosen according to the way data is used by applications and/or by algorithms [43].

In [46] authors propose a survey of such systems and a taxonomy in for classes:

- Federated databases systems as collection of homogeneous data stores and a single query interface;
- Polyglot systems as a collection of homogeneous data stores with multiple query interfaces;
- Multistore systems as a collection of heterogeneous data stores with a single query interface;
- Polystore systems as a collection of heterogeneous data stores with multiple query interfaces.

We adopt a slightly different classification based on models and languages by: 1) considering a unique multidatabase query language approach [33] instead of federated systems to better represent the autonomy of data sources and pragmatic existing systems; 2) replacing homogeneity of data model systems by isomorphic models[8], for example for JSON and the relational model [4, 13] and; 3) instead of using query interface or query engine terms as a criterion we prefer query language. So our classification is the following: multidatabase query language (unique language), polyglot systems including data models isomorphic to relational model (with multiple languages), multistore, polystore. For each of these classes we describe some of the most significant representatives systems.

Spark SQL[9] is the main representative of multidatabase query language. It allows to query structured data from relational like

[7] http://wp.sigmod.org/?p=1629

[8] To be isomorphic two models must allow two way transformations at the structure level but also support equivalence between sets of operators. For example graph data model and relational data model are not isomorphic because relational data model does not support directly transitive closure.

[9] https://spark.apache.org/sql/

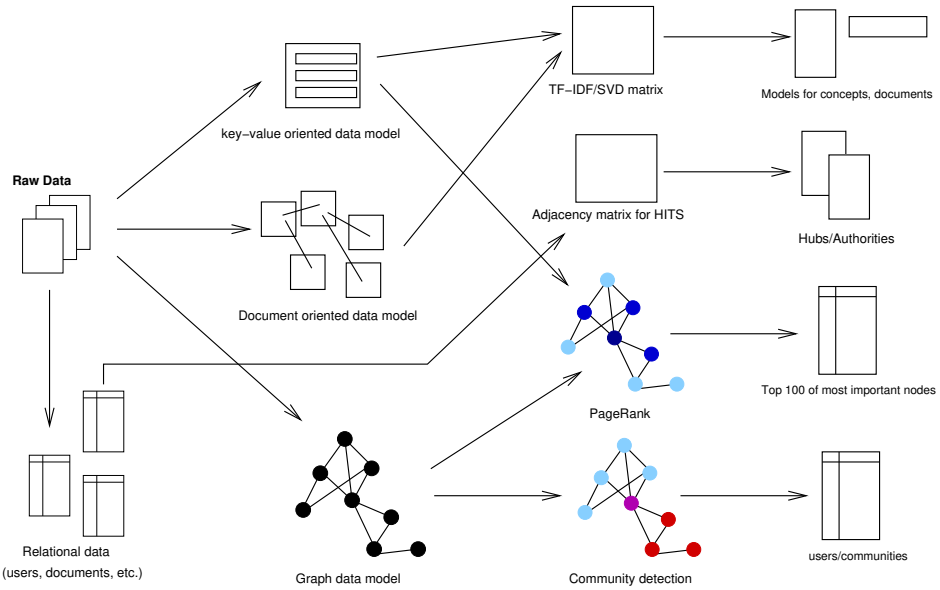

Figure 1: Models, models transformations, algorithms.

data sources (JDBC, JSON, Parquet, etc.) in Spark programs, using SQL.

According to our classification, CloudMdsQL [30] is more a polyglot systems than a multistore system. CloudMdsQL is a functional SQL-like language, designed for querying multiple data store engines (relational or NoSQL) within a query that may contain subqueries to each data store's native query interface. SQL++ which is a part of the FORWARD platform[10], is a semi-structured query language that encompasses both the SQL and JSON [38].

HadoopDB [1] coupled to Hive[11] is a multistore, it uses the map-reduce paradigm to push data access operations on multiple data stores. It can connect to non relational data store such as Neo4j. D4M (Dynamic Distributed Dimensional Data Model) [26] is a multistore that provides a well founded mathematical interface to tuple stores. D4M allows matrix operations and linear algebra operators composition and applies them to the tuple stores. D4M reduces the autonomy of data stores to achieve a high level of performance [27].

The BigDAWG system [15] is a polystore allowing to write multi-database queries with reference to islands of information, each corresponding to a type of data model (PostgreSQL, SciDB and Accumulo). Myria [48] supports multiple data stores as well as different data computing systems such as Spark. It supports SciDB for array processing, RDBMS, HDFS. The RACO (Relational Algebra COmpiler) acts as a query optimizer and processor for MyriaL language. Myria also supports user data functions in different other languages such as Python.

[10] http://forward.ucsd.edu/
[11] https://hive.apache.org/

2.3 Discussion

The general limitations of OLAP approaches for social network data are, on one side the performance and on the other side the schema evolution capabilities.

From a performance viewpoint, data models can induce numerous and expensive queries if we need to apply graph algorithms, e.g. to search path with a specific sequence of vertices, compute shortest path between two nodes or for the construction of adjacency matrix. Furthermore, the models transformations are usually expensive, e.g. to transform relational data into a graph or a graph into time series as well as the aggregation of data (*roll-up*) that uses a transitive closure to find all the potential relationships between users linked to an institution using users' citations in tweets.

From the evolution viewpoint, the issue does not concern data transformation operation but rather the knowledge required to determine the relevant dimensions for analyses. Indeed, the goal of data modeling for data warehousing is to build descriptive models to support analyses and then understand phenomena and produce new knowledge. Usually, social data analysis requires exploratory steps to discover or reveal data properties, to express hypotheses, and then to perform specific analyses on subsets of data, i.e., an iterative approach producing incrementally the knowledge. By comparison enterprise data are related to a context including a well defined semantics (sale of product, organization of the company, etc.) while for the social data, the semantic variability is very important. Analyses are not necessarily conducted by the same set of algorithms and each of them can require specific schema. Thus, a data warehouse model must support a set of operators to allow users to define views (materialized or not) close to the shape of the data expected by the algorithms (e.g., adjacency matrix, time series). For example, although there are few operators (retweets, mentions, hashtags, URLs) in Twitter, and a maximum size of 280 characters per tweet, the datasets generated by Twitter are relatively complex

to model and to process: A mention at the beginning of tweet is considered as an inquiry or an answer while multiple mentions at the end of a tweet are interpreted like the desire to expose the tweet to other users (media for example). Retweet is much more complex to interpret because it can involve either a membership or an opposition (for a satiric tweet for example), consequently its interpretation depends clearly on the content and on the context.

Multi-paradigm data storage is less considered in the approaches of data warehousing which stay still mainly in a traditional vision of the DBMS. The column-oriented or value-oriented NoSQL systems brought another vision which dissociate the features of DBMS by considering systems as storage engines for which the usual properties of the RDBMS are elastic. For example, it belongs to the programmer, depending of the NoSQL storage engine, to implement constraints checking in the application layer. The described polystore approaches roughly share the same principle by using a common language to access to storage engine. Some works on scientific data propose a different approach similar to physical data independence by using a generic model based on associative array to subsume relational, graph and matrix models [18, 24]. Instead, our approach tackles the logical data independence issue and explores the expressiveness of a model based on the mathematical object tensor.

3 OVERVIEW OF APPROACH AND ARCHITECTURE

Our approach is designed under the following assumption: preserve the local autonomy of the storage systems without considering updates of the data except those consisting in materializing results of models transformations or analyses. It is a polystore approach where it is possible to use either the native mode of each system or a tensor-based pivot data model to express queries. Tensor pivot model insures the decoupling between programs and data (*logical data independence*).

In terms of analysis tools, we have selected two types of languages: R and Spark. Tensor flow is similar to libraries supporting tensors in R or Spark, but it has been design with a workflow orientation and not with a data model orientation. Thus, tensor is rather a structure of exchange among processes of a complex workflow than a model to represent real data.

In the following sections we will clarify the notion of tensor and describe our polystore architecture but, at first, we use the analogy of a tensor with a multidimensional array or an hypermatrix, that is a family of elements indexed by N sets.

3.1 From Tensor Mathematical Object to a Data Model

Tensors are very general abstract mathematical objects which can be considered according to various points of view. Tensors can be seen as multi-linear applications or as the result of the tensor product. A tensor is an element of the set of the functions from the product of N sets $I_j, j = 1, \ldots, N$ to $\mathbb{R} : \boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, N is the number of dimension of the tensor or its order or its mode.

A tensor is often defined of as a generalized matrix, 0-order tensor is a scalar, 1-order one is a vector, 2-order one is a matrix, tensors of order 3 or higher are called higher-order tensors. More formally, a N-order tensor is an element of the tensor product of N vector spaces, each of which has its own coordinate system.

Tensor operations, by analogy with the operations on matrices and vectors, are multiplications, transpose, unfolding or matricization and factorizations (also named decompositions) [10, 29]. The most used tensor products are the Kronecker product denoted by $\otimes$, Khatri-Rao product denoted by $\odot$, Hadamard product denoted by $\circledast$, external product denoted by $\circ$ and n-mode denoted by $\times_n$.

In the rest of the article, we use the boldface Euler script letters to indicate a tensor $\boldsymbol{\mathcal{X}}$, for matrices the boldface capital letters $\mathbf{M}$, the boldface lowercase letters to indicate a vector $\mathbf{v}$, and an element of the tensor or a scalar is noted in italic, for example x_{ijk} is ijk-i-th element of 3-order tensor $\boldsymbol{\mathcal{X}}$.

Our objective is to provide the mathematical object tensor with operators to perform data manipulation (transformations) and analysis as well to set up the notion of schema and views to build a real tensor data model (figure 2). Moreover, as tensor will play of role of pivot among different data models, we will define constructs in the model to support links to data sources.

3.2 Architecture

The figure 3 describes how data which fill tensors values are obtained from *wrappers* which express the queries in the native language of each data store.

Queries for tensor construction are submitted to the *wrappers* and have the same shape: they send back $N + 1$ attributes where N first attributes are the dimensions and the last one serves as value for the elements of the tensor (obtained with `GROUP BY`-like queries on the attributes which represent the dimensions). This feature allows us to implement *wrappers* having all the same structure and so to simplify the models transformations. For R language, the *wrappers* are implemented using the packages R DBI, RNeo4j[12], RMongo, RCassandra et RHBase[13]. In Spark, we work with the SQL layer, data frame and RDD (Resilient Data Sets) abstractions. To represent indexes used in each dimension of tensors we use associative arrays, their values are sets of identifiers (unique keys) that map sets of values of a specific data type to natural numbers, for example to associate each Twitter users, or hashtag to a natural number. Associative arrays are maps from $K \rightarrow \mathbb{N}$ where K, a dimension, is a set of atomic types (real, integer, string, etc.). Associative arrays are translated into specific queries, sent to a storage system and materialized in the tensor model layer (figure 3). In Spark, they are then used in a multi-sources join to obtain values for a tensor. In R, associative arrays are stored as specific structure in the data source from which values are retrieved.

4 TDM: A TENSOR DATA MODEL

In this section, we briefly recall motivations and works around tensors in databases and complex networks fields. Then we present our tensor data model (TDM) and give some illustration examples.

4.1 TDM Motivations

In a database context, tensors are rather multidimensional arrays [5]. For example, in SciDB [44], the data model is based on arrays where

[12] https://github.com/nicolewhite/RNeo4j
[13] https://github.com/RevolutionAnalytics/rhbase

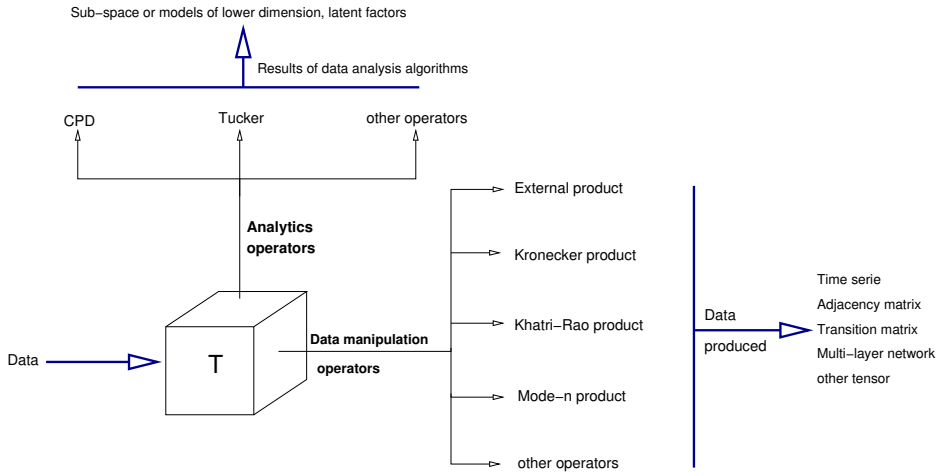

Figure 2: Tensor Model and operations

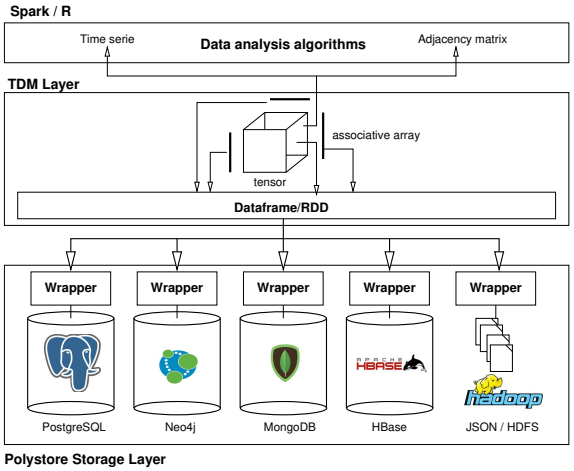

Figure 3: Tensor and associative arrays roles in the architecture of a polystore system

each cell can be a vector of values, dimensions can be either integers or user-defined types. These models are useful for data produced in domains such as earth-sciences that manipulate matrices and time series.

Complex networks, and network science field of research aim at providing tools to model the structural complexity of a variety of complex systems such as transportation, biology, communication networks and online social networks. Although adjacency matrices are popular structures to model networks, such representation is insufficient for representing heterogeneous multi-layer networks. Real complex networks are heterogeneous i.e., nodes and edges have different types and edges types can be semantically grouped. Achieving a deep understanding of such complex network data requires generalizing the traditional network theory. The concepts of multi-layer networks, multiplex networks, multi-relational networks or network of networks have been introduced recently to provide an expressive model for real-world complex networks [28]. In [12] and [28] the authors use adjacency tensor as a model for multiplex networks[14] and study popular analysis measures such as degree of centrality, eigenvector centrality, clustering coefficients, random walks, modularity on multiplex network representation. They also discuss the relationships among multiple models such as graph, multigraph, hypergraph and linear and multi-linear algebras. In [42] the authors use a 3-order tensor as a model of multi-relational network. One of the dimensions is used to represent the layers i.e. the different types of relationships, for example if there are n nodes (individuals) and r relationships between nodes (professional, family, friendship) the size of the 3-order tensor will be $n \times n \times r$ and ijk-i-th element has 1 for value if the node i maintains a relationship of type k with the node j, each relationship corresponding to a front slice of the tensor. Multi-layer networks and their representation in tensors allow to model complex relationships among different dimensions, without having a fine knowledge and then understanding their links by means of an appropriate decomposition.

4.2 TDM Formalization

In TDM, tensor dimensions are represented by associative arrays. In the general case, an associative array is a map from a key space to a value space.

Definition 4.1 (Associative Array). *An associative array is a map that associates keys to values as:*

$$A : K_1 \times \cdots \times K_N \rightarrow \mathbb{V}$$

where $K_i, i = 1, .., N$ are the sets of keys and $\mathbb{V}$ is the set of values.

The definition given in [25] restricts $\mathbb{V}$ to have a semi-ring structure and the associative array to have a finite support. In TDM we use associative arrays for two cases. First, we use different associative arrays denoted by $\mathbf{A}_i$ for $i = 1, .., N$ to model dimensions

[14] A multiplex network is a special type of multi-layer network in which the only possible types of inter-layer connections are ones in which a given node is connected to its counterpart nodes in the other layers.

of a tensor $\boldsymbol{\mathcal{X}}$, in this case the associative array has only one set of keys associated with natural numbers $\mathbf{A}_i : K \rightarrow \mathbb{N}$. Second, an associative array is used to represent the values in each tensor. A N-order tensor $\boldsymbol{\mathcal{X}}$ maps keys (n-uples) to a space of values (string, real, integer, etc.).

Definition 4.2 (Named Typed Associative Array). *A named and typed associative array of a tensor $\boldsymbol{\mathcal{X}}$ is a triple $(Name, \mathbf{A}, T_A)$ where $Name$ is a unique string which represents the name of a dimension, $\mathbf{A}$ is the associative array, and T_A the type of the associative array i.e. $K \rightarrow \mathbb{N}$.*

According to the previous definition, the signature of a named typed associative array is $Name : K \rightarrow \mathbb{N}$.

Definition 4.3 (Typed tensor). *A typed tensor $\boldsymbol{\mathcal{X}}$ is a tuple $(Name, D_A, V, T)$ where:*

- *$Name$ is the name of the tensor;*
- *D_A a list of named typed associative arrays i.e., one named typed associative array per dimension;*
- *V is an associative array that store the values of the tensor;*
- *T is the type of the tensor, i.e. the type of its values.*

V handles the sparsity of tensors. Sparse tensors have a default value (e.g. 0) for all the entries that not explicitly existing in the associative array. The signature of a typed tensor is $Name : D_A \rightarrow T$. A TDM schema is a set of typed tensors signatures.

4.3 Examples with Twitter Data

Even if Twitter data seem to have a simple structure they are actually very rich. Their corresponding relational schema has only few relations for example, some tables are used to represent entities (i.e. tweets, hashtags, etc.), or social relationships (i.e. followers, etc.), foreign keys and association tables represent the use of operators such as RT, @, #, URL. Nevertheless, the relational schema is not so easy to process because most analytic queries require multiple joins and auto-joins. Moreover, they usually contain complex group by clauses, sometimes with timestamps or ranking. Finally, some queries are recursive or require to compute the transitive closure of relations.

The social data retrieved from Twitter are by nature a multiplex network where the nodes are heterogeneous (users, tweets, hashtags, etc.) and the relationships too (retweet, publish, follow, mention, etc.). Moreover, different dimensions are existing such as users interactions (retweet, follow), users actions (publish, like), tweet structure (content, mention, hashtag, URL), so a richer model like multi-layered network model [28] should be used to perform meaningful analysis.

To illustrate the potential use of tensor, let start with a multi-layer network defined as $GM = (V, E, L)$ where $V = \{V_1, V_2, \ldots, V_n\}$ is a partitioned set of nodes, $E = \{E_1, E_2, \ldots, E_k\}$ is partitioned set of edges, with $E \subseteq V \times V$ and $E_i \subseteq V_l \times V_m$ for $i \in \{1, \ldots, k\}$ and $l, m \in [1, n]$. L is a partitioned set of layers, $L = \{L_1, L_2, \ldots, L_p\}$ where $L_i \subseteq E$, with $L_i \cap L_j = \emptyset, \forall i, j$ modeling the dimensions.

Binary relationships are subsets of $E \times E$ and their associated functions R_i can be used to associate values to edges or to count the number of edges between vertices. Binary relationships can be represented by matrices or tensor slices. For example, V_1, E_1 and R_1 can model users and mentions (@ operator) and the number of mentions between two users. But all relationships in the social data do not have the same signature. Let us denote the set of users by V_1, the set of tweets by V_2 and take from example different types of relationships that do not have the same signature:

- *mention*, $R_1 : V_1 \times V_1 \rightarrow \mathbb{N}$
- *retweet*, $R_2 : V_1 \times V_2 \rightarrow \mathbb{N}$
- $retweet_U$, $R_3 : V_1 \times V_1 \rightarrow \mathbb{N}$, i.e. aggregated retweets according to source user
- *publish*, $R_4 : V_1 \times V_2 \rightarrow \mathbb{N}$
- $follow$, $R_5 : V_1 \times V_1 \rightarrow \mathbb{N}$

Let's look in more details the relationships *mention*, $retweet_U$ and $follow$, their associated tensor is $\boldsymbol{\mathcal{T}}$ from $V_1 \times V_1 \times \{R_1, R_3, R_5\} \rightarrow \mathbb{N}$. For example $\boldsymbol{\mathcal{T}}(u_1, u_2, R_1) = 5$ if the user u_1 has mentioned five times the user u_2 in its tweets. Thus, a relationship between users can be modeled by a 2-order tensor and the set of users for their relationships will be represented by a 3-order tensor[15].

Let's now take another example with three sets of nodes *users* defined by V_1, *tweets* defined by V_2 and *time* defined by V_3 for representing the relationship *publish*, i.e. a user publishes a tweet on a given day. The different named typed associative arrays are given on figure 4. Figure 5 depicts the tensor $\boldsymbol{\mathcal{X}}$ and its values. The value 1 at coordinate (1,3) in the front slice means that the user u3 has posted the tweet t1 on day 18-03-08.

However, it is not so easy to model the presence of several operators in the same tweet, for example the co-occurrence of hashtags can be used to explain the meaning of the first one if it is very general term and a mention can be added at the beginning of the tweet as a call to an answer. These kinds of specific relationships can be modeled by using hyperedge in an hypergraph that can also be transformed into a tensor that encompasses both simple relationship and complex ones like $\boldsymbol{\mathcal{T}} : V_1 \times V_2 \times V_4 \times V_4 \times V_1 \rightarrow \mathbb{N}$, where V_4 is the hashtags used in the tweets.

This example highlights the following elements:

- When a user writes only one mention in a tweet without hashtags, the dimensions that represent hashtags should also include a *null* value. This *null* value is easily supported by introducing a specific value in associative arrays;
- As the order of tensor increases the sparsity also increases, at first it seems to be natural to define or adapt the theory of normal forms from relational data model to tensor[16]. Using the *null* value a tensor can model both simple relationships and more complex relationships this can also contribute to gain in performance by allowing materialized joins.

5 TDM'S OPERATORS

To carry out a wide range of queries it should possible to define several of the standard operators from relational algebra in terms of tensor operations. In [24, 25] the authors define a model and operators over associative arrays to unify relational, arrays, and key-value algebras. Our operators are defined to provide programmers with a logical data independence layer i.e. to bridge the semantic gap between analysis tools and storage systems. Our set of operators

[15] If all the relationships to be represented are homogeneous, that is to say if the associated functions have the same signature.

[16] In the previous example it will probably drive us to define 3 or more 2-order tensor for users and hashtags, users and mentions, users and tweets.

user	u1	u2	u3	...
i	1	2	3	...

tweetID	t1	t2	t3	t4	...
j	1	2	3	4	...

time	18-03-08	18-03-07	18-02-28	18-02-26	...
k	1	2	3	4	...

Figure 4: Named Typed Associative Arrays representing tensor dimensions

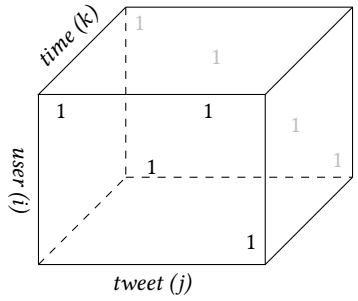

Figure 5: Associated tensor $\mathcal{X}$

works at two different levels: at the associative array level and at the tensor level.

5.1 Data Manipulation Operators

A fiber of a tensor $\mathcal{X}$ is a vector obtained by fixing all but one $\mathcal{X}$'s indices: $\mathcal{X}_{:jk}$, $\mathcal{X}_{i:k}$ et $\mathcal{X}_{ij:}$. Fibers are always assumed to be column vectors, analogue of matrix rows and column. A slice of a tensor $\mathcal{X}$ is a matrix obtained by fixing all but two of $\mathcal{X}$'s indices: $\mathcal{X}_{i::}$, $\mathcal{X}_{:j:}$ et $\mathcal{X}_{::k}$.

A project operator can be generalized by the Hadamard product of a N-order tensor with a boolean tensor of the same order that contains 1 for the elements to be selected: $\mathcal{X} \circledast \mathcal{B}$.

For example, for a 3-order tensor, $\mathcal{X}_1$, representing users, hashtags used (their number of occurrences in the tweets of a user) and the time, to select all hashtags used by a user u_i, the result will be in a 2th-order tensor such as: $\mathcal{X}_2 = \mathcal{X}_1 \circledast \mathcal{B}_1$ with $\mathcal{B}_{1_{i::}} = 1$. To obtain a time series reflecting the use of a hashtag, the sum of the columns of the 2th-order tensor obtained is carried out.

A select operator can act on two levels: 1) on the values contained in the tensor (equivalent to a selection on a single relational attribute) or 2) on the values that are in associative arrays $\mathbf{A}_i$, $i = 1, .., N$. The select operator σ is written with two conditions, the first on the dimensions, the other on the values:

$$\sigma[cdt\ dim][cdt\ val]\mathcal{X}$$

The condition on the dimensions is made by the product of Hadamard with Boolean tensor whose elements which have 1 for value correspond to the elements of the $\mathbf{A}_i$ selected. The following example selects tweets published by the user $u1$ and time is comprising between 18-03-08 and 18-02-28, from the tensor $\mathcal{X}$:

$$\sigma[U =' u1' \wedge T \geq' 18-02-28' \wedge T \leq' 18-03-08'][=1]\mathcal{X}$$

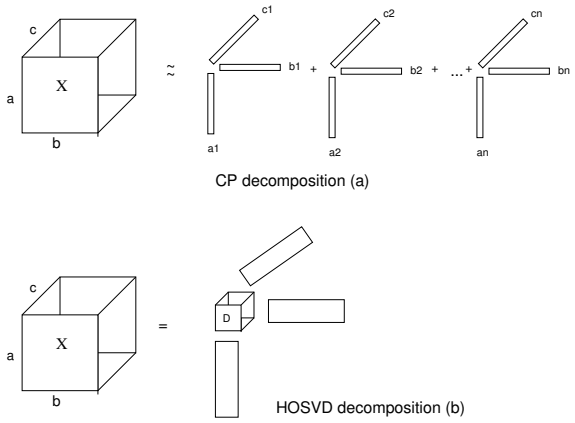

Figure 6: CP and HOSVD decompositions

5.2 Analytical Decomposition Operators

Tensor decompositions such as CANDECOMP/PARAFAC (CP), Tucker, HOSVD are used to perform dimensionality reductions and to extract latent relations [29]. Since tensor representations of data are multiple and their semantics are not explicit, the results of tensor decompositions are complex to interpret. However, by analogy with the matrix decompositions it is possible to determine the decomposition associated with an objective, or to answer to a question, by using a proper tensor model. For example, to express each space generated by one of the dimensions of the tensor independently of the others, but depending on the global space, we can use a decomposition CP (figure 6(a)). To determine user models based on hashtag or recurring patterns of behavior we prefer to use a HOSVD decomposition (figure 6(b)). Then models produced by HOSVD decomposition can be used in recommendation systems.

6 SCIENTIFIC APPLICATION USE CASE

In this section, we validate our approach by a proof-of-concept, showing how TDM is used in the context of a real project (TEP 2017) involving collaborations with social scientists. We will not dwell here on the results of the interpretations made by social scientists.

The data we are working on, are a part of 50M tweets corpus collected during the french presidential election in 2017. Data are stored in JSON file format in HDFS, most important attributes of tweets are stored in a relational database in a unique table and in another database with a schema in third normal form. The main research objective of this project is to better understand where and how discourse relating to emerging political issues circulates on the French Twitter-sphere, during presidential election campaign in 2017.

We choose to focus our study on virality, the initial list of potential viral tweets (i.e. tweets that have been propagated a lot) reduces the corpus of 50M of tweets to thousand, giving the possibility to social scientists to validate our experiments by a qualitative study. According to Bessi and al. [7], robots played an important

role during the American presidential election of 2016. The authors estimated that about 400,000 robots are engaged, responsible for roughly 3.8 million tweets (19% the entire conversation).

Our objective is to understand how bots artificially relieved tweets and to analyze this phenomenon. From the model point of view and more precisely, the construction of the tensor, we wish to observe the quantity and complexity of code necessary to transform the data in two cases: 1) using R directly for the analyses; 2) using the TDM model like intermediate.

6.1 Viral Tweets

Virality can be defined by three parameters summarized by Beauvisage and al. [6]: temporal concentration of the attention to the content, traffic of this content and mechanisms of the contagion from an individual to the other one. Unlike tweets that make buzz, the start of their broadcast is slower and it lasts longer in time. Various metrics is taken into account to determine tweet virality [21, 34, 49]. We have lexical metrics with study of the contents (presence of URL and hashtags, construction of the hashtag from several words, etc.) and contextual metrics with the activity of the account, its community, etc. as well as the time. Number of metrics makes difficult the interpretation of obtained results.

We reduced the global corpus to the period between two ballots of the election (from April 24th, 2017 till May 7th, 2017) and calculated the number of retweets for every tweet. A sample of the most popular tweets that is, in our case, the most propagated by retweets was obtained by selecting tweets retweeted at least 1 000 times over the period. This sample contains 1,123 tweets among which some were retweeted more of 20 000 times. The list of tweets id is available on GitHub[17] for a reproducibility of results. For each tweet, we then studied the time series for the frequency i.e. the number of retweets by period of one hour, 4 hours, 8 hours, 12 hours and 24 hours, calculated the speed of propagation and extracted, by use of the algorithm breakout[18], the intervals of time in which activity was important.

Queries which produced the analysis results are expressed in the native language of the storage system, here SQL. In both cases, queries are launched from R on the PostgreSQL database with normalized schema and produce data for the algorithms. It takes in less than a minute to produce tensor dimensions and less than 5 minutes to produce tensor values. It is approximately the same execution time for build the dataset without using the tensor. In the tensor (users, hashtags, time) case the code is divided in several small queries (dimensions, values). In the other case a unique, complex to read, query with more than 15 lines of SQL produces the data set.

6.2 Bots Detection

In order to understand the diffusion mechanisms of supposed viral tweets, we first studied the share of activity related to bots or rather accounts whose behavior is similar to humans.

Several methods have been proposed to detect robots [47], they most often aggregate a large number of features to produce a predictive model based on a learning algorithm such as *random-forest*. An experiment using the OSoMe API[19] to obtain a probability of automated behavior (of robot type) leds us to note that the values of the probabilities were not enough significant to detect bots during the studied period. One of the assumptions is that it is hybrid accounts of users assisted by algorithms. However, simple criteria such as the maximum number of tweets published in one hour makes it possible to unambiguously find some accounts with automated behavior, confirmed by the manual study, which will serve as a marker for the other analyzes. Based on the observation that real (human) accounts publish tweets on a regular basis, basic statistics such as the average of tweets or retweets sent per hour do not make it easy to extract robots.

Bots do not retweet randomly, so from the tweets contents (hashtags) we built a 3-order tensor modelling the users U having retweeted a supposed viral tweet, the hashtags H contained in these tweets and the time T (14 days of observation). We got a tensor containing potentially $1,077 \times 568 \times 336$ items; the values of the tensor therefore represent the number of occurrences of each hashtag per user per hour by considering only the retweets of supposed viral tweets. The research space being very large, then we performed a CP decomposition to reduce the user space based on their behavior. The decomposition CP produces n groups of three 1-order tensors, here vectors U, H, T (see figure 6(a)). We then apply the k-means clustering algorithm to identify groups of users. The retained value of n is the one from which there is no more modification of the clusters. Experimentally, we get $n = 8$ and therefore a user is described by a point in an 8-dimensional space. The k-means algorithm applied to this data determines 4 groups of users: a group of one account already detected as a robot, a group of two accounts, a group of about thirty accounts comprising more than half users with a probability of being a robot greater than 0.6 and a last group containing other users. The group of two accounts, revealed after manual study, to be linked (same behavior and hashtags) and assisted by an algorithm that retweets messages against the Macron candidate. These accounts had evaded other analysis techniques.

The tensor construction from the data and the tensor decomposition in R take each less than 5 minutes, a thorough study of performance will be required.

7 CONCLUSION

In this article, we have proposed a new architecture for social networks data warehousing based on a polystore system and a tensor-based pivot data model. The tensor model makes it possible to generalize matrix representations (adjacency matrix, time series, etc.) as well as graphs including multigraphs and hypergraphs and to take into account models of complex networks (multi-layer networks, multi-relational networks, etc.). We also presented some data manipulation and analysis operators on the tensor model.

The work has been experimented with Twitter data. We detected viral tweets by focusing on time series. In order to understand the information dissemination mechanism, we also studied the significance of social bot activity in diffusion. Our results have been validated by social scientists and researchers in communication

[17] https://github.com/EricLeclercq/TEE-2017-Virality

[18] The algorithm is based on the method E-Divisive with Medians (EDM), it uses a statistical measure of energy to detect differences of the average https://github.com/twitter/BreakoutDetection.

[19] https://botometer.iuni.iu.edu/

sciences. This experiment demonstrated the tensor modeling capabilities and the relevance of the architecture according to the ease of implementation of model transformation in analyses.

The perspectives concern the complete formalization of a set of operators as well as the study of the properties of the algebraic structure that they generate (semi-ring for example). In parallel, we want to develop a real prototype of architecture in order to study queries optimization including tensor operators. Nevertheless, semi-ring structures give operators good properties for distributed implementation, which suggests a good potential for scaling up.

REFERENCES

[1] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, and A. Rasin. Hadoopdb: an architectural hybrid of mapreduce and dbms technologies for analytical workloads. *Proceedings of the VLDB Endowment*, 2(1):922–933, 2009.

[2] M. A. Al-Garadi, K. D. Varathan, S. D. Ravana, E. Ahmed, G. Mujtaba, M. U. S. Khan, and S. U. Khan. Analysis of online social network connections for identification of influential users: Survey and open research issues. *ACM Computing Surveys (CSUR)*, 51(1):16, 2018.

[3] F. Atefeh and W. Khreich. A survey of techniques for event detection in twitter. *Computational Intelligence*, 31(1):132–164, 2015.

[4] M.-A. Baazizi, H. B. Lahmar, D. Colazzo, G. Ghelli, and C. Sartiani. Schema inference for massive json datasets. In *Extending Database Technology (EDBT)*, pages 222,233, 2017.

[5] P. Baumann and S. Holsten. A comparative analysis of array models for databases. In *Database Theory and Application, Bio-Science and Bio-Technology*, pages 80–89. Springer, 2011.

[6] T. Beauvisage, J.-S. Beuscart, T. Couronné, and K. Mellet. Le succès sur Internet repose-t-il sur la contagion ? Une analyse des recherches sur la viralité. *Tracés. Revue de sciences humaines*, (21):151–166, 2011.

[7] A. Bessi and E. Ferrara. Social bots distort the 2016 U.S. Presidential election online discussion. *First Monday*, 21(11), 2016.

[8] S. Bringay, N. Béchet, F. Bouillot, P. Poncelet, M. Roche, and M. Teisseire. Towards an on-line analysis of tweets processing. In *Database and Expert Systems Applications*, pages 154–161. Springer, 2011.

[9] C. Chen, X. Yan, F. Zhu, J. Han, and S. Y. Philip. Graph OLAP: Towards online analytical processing on graphs. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 103–112. IEEE, 2008.

[10] A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[11] P. Costa, F. F. Souza, V. C. Times, and F. Benevenuto. Towards integrating online social networks and business intelligence. In *IADIS international conference on Web based communities and social media*, volume 2012, 2012.

[12] M. De Domenico, A. Solé-Ribalta, E. Cozzo, M. Kivelä, Y. Moreno, M. A. Porter, S. Gómez, and A. Arenas. Mathematical formulation of multilayer networks. *Physical Review X*, 3(4):041022, 2013.

[13] M. DiScala and D. J. Abadi. Automatic generation of normalized relational schemas from nested key-value data. In *Proceedings of the 2016 International Conference on Management of Data*, pages 295–310. ACM, 2016.

[14] A. Drif and A. Boukerram. Taxonomy and survey of community discovery methods in complex networks. *International Journal of Computer Science and Engineering Survey*, 5(4):1, 2014.

[15] J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, J. Kepner, S. Madden, D. Maier, T. Mattson, and S. Zdonik. The BigDAWG Polystore System. *ACM SIGMOD Record*, 44(2):11–16, 2015.

[16] C. Favre, W. Jakawat, and S. Loudcher. Graphes enrichis par des Cubes (GreC) : une approche innovante pour l'OLAP sur des réseaux d'information. In *Actes du XXXVème Congrès INFORSID, Toulouse, France*, pages 293–308, 2017.

[17] V. Gadepally, P. Chen, J. Duggan, A. Elmore, B. Haynes, J. Kepner, S. Madden, T. Mattson, and M. Stonebraker. The BigDAWG polystore system and architecture. In *IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6, 2016.

[18] V. Gadepally, J. Kepner, W. Arcand, D. Bestor, B. Bergeron, C. Byun, L. Edwards, M. Hubbell, P. Michaleas, J. Mullen, et al. D4m: Bringing associative arrays to database engines. In *IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6, 2015.

[19] D. Ghosh. Multiparadigm Data Storage for Enterprise Applications. *IEEE Software*, 27(5):57–60, Sept 2010.

[20] A. Guille, H. Hacid, C. Favre, and D. A. Zighed. Information diffusion in online social networks: A survey. *ACM Sigmod Record*, 42(2):17–28, 2013.

[21] T.-A. Hoang, E.-P. Lim, P. Achananuparp, J. Jiang, and F. Zhu. On modeling virality of twitter content. In *International Conference on Asian Digital Libraries*, pages 212–221. Springer, 2011.

[22] L. Hogben. *Handbook of linear algebra*. Chapman and Hall/CRC, 2013.

[23] J. Hölsch, T. Schmidt, and M. Grossniklaus. On the performance of analytical and pattern matching graph queries in neo4j and a relational database. In *EDBT/ICDT 2017 Joint Conference: 6th International Workshop on Querying Graph Structured Data (GraphQ)*, 2017.

[24] D. Hutchison, B. Howe, and D. Suciu. LaraDB: A minimalist kernel for linear and relational algebra computation. In *Proceedings of the 4th Algorithms and Systems on MapReduce and Beyond*, page 2. ACM, 2017.

[25] H. Jananthan, Z. Zhou, V. Gadepally, D. Hutchison, S. Kim, and J. Kepner. Polystore mathematics of relational algebra. In *IEEE International Conference on Big Data (Big Data)*, pages 3180–3189, Dec 2017.

[26] J. Kepner, W. Arcand, W. Bergeron, N. Bliss, R. Bond, C. Byun, G. Condon, K. Gregson, M. Hubbell, J. Kurz, et al. Dynamic distributed dimensional data model (d4m) database and computation system. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 5349–5352. IEEE, 2012.

[27] J. Kepner, W. Arcand, D. Bestor, B. Bergeron, C. Byun, V. Gadepally, M. Hubbell, P. Michaleas, J. Mullen, A. Prout, et al. Achieving 100,000,000 database inserts per second using accumulo and d4m. In *High Performance Extreme Computing Conference (HPEC), 2014 IEEE*, pages 1–6. IEEE, 2014.

[28] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter. Multilayer networks. *Journal of Complex Networks*, 2(3):203–271, 2014.

[29] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[30] B. Kolev, C. Bondiombouy, P. Valduriez, R. Jiménez-Peris, R. Pau, and J. Pereira. The CloudMdsQL Multistore System. In *SIGMOD*, 2016.

[31] M. B. Kraiem, J. Feki, K. Khrouf, F. Ravat, and O. Teste. Modeling and olaping social media: the case of twitter. *Social Network Analysis and Mining*, 5(1):47, 2015.

[32] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

[33] W. Litwin, A. Abdellatif, A. Zeroual, B. Nicolas, and P. Vigier. Msql: A multidatabase language. *Information sciences*, 49(1-3):59–101, 1989.

[34] Z. Ma, A. Sun, and G. Cong. On predicting the popularity of newly emerging hashtags in twitter. *Journal of the American Society for Information Science and Technology*, 64(7):1399–1410, 2013.

[35] S. Mansmann, N. U. Rehman, A. Weiler, and M. H. Scholl. Discovering OLAP dimensions in semi-structured data. *Information Systems*, 44:120–133, 2014.

[36] I. Moalla and A. Nabli. Towards Data Mart Building from Social Network for Opinion Analysis. In *15th International Conference Intelligent Data Engineering and Automated Learning (IDEAL)*, pages 295–302, 2014.

[37] L. G. Moya, S. Kudama, M. J. A. Cabo, and R. B. Llavori. Integrating web feed opinions into a corporate data warehouse. In *Proceedings of the 2nd International Workshop on Business intelligencE and the WEB*, pages 20–27. ACM, 2011.

[38] K. W. Ong, Y. Papakonstantinou, and R. Vernoux. The SQL++ Query Language: Configurable, Unifying and Semi-structured. Technical report, UCSD, 2015.

[39] M. T. Özsu and P. Valduriez. *Principles of distributed database systems*. Springer Science & Business Media, 2011.

[40] N. U. Rehman, S. Mansmann, A. Weiler, and M. H. Scholl. Building a data warehouse for twitter stream exploration. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1341–1348. IEEE Computer Society, 2012.

[41] F. Riquelme and P. González-Cantergiani. Measuring user influence on twitter: A survey. *Information Processing & Management*, 52(5):949–975, 2016.

[42] M. A. Rodriguez and J. Shinavier. Exposing multi-relational networks to single-relational network analysis algorithms. *Journal of Informetrics*, 4(1):29–41, 2010.

[43] J. Sharp, D. McMurtry, A. Oakley, M. Subramanian, and H. Zhang. *Data Access for Highly-Scalable Solutions: Using SQL, NoSQL, and Polyglot Persistence*. Microsoft patterns & practices, 1st edition, 2013.

[44] M. Stonebraker, P. Brown, D. Zhang, and J. Becla. Scidb: A database management system for applications with complex analytics. *Computing in Science & Engineering*, 15(3):54–62, 2013.

[45] M. Stonebraker and U. Cetintemel. " one size fits all": an idea whose time has come and gone. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 2–11. IEEE, 2005.

[46] R. Tan, R. Chirkova, V. Gadepally, and T. G. Mattson. Enabling query processing across heterogeneous data models: A survey. In *Big Data (Big Data), 2017 IEEE International Conference on*, pages 3211–3220. IEEE, 2017.

[47] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini. Online Human-Bot Interactions: Detection, Estimation, and Characterization. *CoRR*, abs/1703.03107, 2017.

[48] J. Wang, T. Baker, M. Balazinska, D. Halperin, B. Haynes, B. Howe, D. Hutchison, S. Jain, R. Maas, P. Mehta, et al. The myria big data management and analytics system and cloud services. In *CIDR*, 2017.

[49] L. Weng, F. Menczer, and Y. Ahn. Predicting Successful Memes using Network and Community Structure. *CoRR*, abs/1403.6199, 2014.