Information Integration for Movies Data Using Graph Database

Vibhor Goyal*, Aman Agarwal*, E.S. Pilli* and Sameep Mehta†
*Department of Computer Science and Engineering
Malaviya National Institute of Technology, Jaipur, Rajasthan, India 302017
Email: vibhorgoyal20@gmail.com, aman.agarwal0008@gmail.com, espilli.cse@mnit.ac.in
† IBM India Research, New Delhi, India 110070
Email: sameepmehta@in.ibm.com

***Abstract*—The development of the Internet in the recent years has made it possible to access different information systems anywhere in the world. Information Integration is the merging of information from heterogeneous sources with differing conceptual, contextual and typographical representations. In this paper, we exploit Information Integration techniques for movies data from different sources over the web.**

Graphs are used to model many complex data objects and their relationships in the real world. In recent years, graphs have become increasingly popular in a variety of domains varying from Biology, Chemistry, Healthcare systems and computer vision to Business Intelligence and Social Media Analytics.

We have developed a system that integrates data about movies from various sources across the web and populated the TITAN graph database. This enables us to show that complex information can be retreived using simple queries using Gremlin, a graph query language.

Keywords*—*Information Integration, Data Fusion, Graph Database, TITAN, Gremlin.

I. Introduction

The late 1990s have witnessed huge proliferation of electronically accessible information that has led to a great deal of research and development in information retrieval to help users so that they can search and quickly retrieve relevant and meaningful information [1]. Recently, user demands for integrated searches over different information sources have been increasing rapidly. However it is not easy to seamlessly integrate heterogeneous information retrieval systems. In particular, resolving discrepancies in the structure of the databases that are to be integrated is one of the most difficult issue.

The immensity of web data valuable for various human needs has led to research on information extraction from the web [2]. With more and more information sources available via inexpensive network connections over the Internet, the desire to organize this information in a user friendly way has been the motivation behind the work presented here [3]. Much of the websites available over the web do not serve all the information on one platter so there is a need to extract information from different sources and apply information integration techniques to populate the data onto a common store. One of the advantages of information integration systems is that the user of such a system obtains a complete yet concise overview of all existing data without needing to access all data sources separately: complete because no object is forgotten in the result; concise because no object is represented twice and the data presented to the user is without contradiction. Information Integration possesses many applications in the present world. These may vary from comparison shopping, portals integrating data from multiple sources, electronic market places to medical genetics in integrating genomic data, Astrophysics to monitor the events in the sky and several other areas of Enterprise Data integration.

Graph databases have become increasingly popular for a variety of uses ranging from modeling online code repositories to tracking software engineering dependencies. These areas use graph databases because many of their problems can be expressed in terms of graph traversals. Recent work has applied graph databases to virtualization management, noting that many IT questions can also be expressed as graph traversals. Graph databases differ in that the data is the structure. This provides a level of flexibility and resilience that is a great match for todays fast-moving business and agile development methods. The data representation also differs fundamentally. Graph databases represent data as things (or nodes) and relationships between things. This comes much closer to the way we think about complex systems.

Relational databases, on the other hand, need to carry out a number of steps to determine whether and how things are connected, and then to retrieve related data records. Response times slow down as a relational database grows in volume, which causes problems as a business grows. However with a graph database, traversal speed remains constant, not depending on the total amount of data stored. This allows the database to naturally keep up with ones business as it grows. The reason for this vast difference in performance lies in how data and relationships are stored inside the database. Native graph databases use a technique called index-free adjacency. In simple terms, this means that each data element points directly to its inbound and outbound relationships, which in turn, point directly to related nodes, and so on. This technique allows million of related records to be traversed per second.

In this paper, we propose a system that integrates distributed and heterogeneous information of movies from various data sources across the web. We have developed a prototype that integrates information from four different sources viz. Internet Movie Database (IMDb) [5], Twitter [6], www.metacritic.com [7] and www.rottentomatoes.com [8]. The different components of the above data sources are deployed using a graph model. These componets fit into one another using nodes and edges which are the basic building blocks of any graph model. This graph model is further queried

978-1-4799-1823-2/15/$31.00 ©2015 IEEE

and information is retrieved from it in an efficient way. The rest of the paper is organized as follows: Section 2 contains related work done in the field of Information Integration and Graph Databases. Section 3 presents the proposed structure of our system. Section 4 gives results drawn from analysis of our observations. Section 5 gives the conclusion and discusses some of the future aspects related with the work.

II. Related Work

Many established practices exist for the traditional discipline of Data Fusion. However there is a need to expand the fusion view from sensor networks [9], intrusion detection [10], [11], collaborative alerts correlation [12], anomaly detection [13] to integration of information available on the web on to a common store. Moreover, most of the earlier Information Integration Systems were based on a relational data model, however in the present scenario with the increasing size and complexity of data, there is a need to use a more efficient and robust data model.

Jiangfan et al. [9] investigated the cognitive model of spatial information, the conceptual model of space phenomena, and the logic of spatial data management mode, leading to the integration of various spatial data sets from the view of model integration. Furthermore, they provided a method for the effective storage and management of massive and multi-source heterogeneous spatial data.

Zhao et al. [11] proposed a new computer information security protection system based on data fusion theory. Multiple detection measures were fused in their system, so that it has lower false negatives rate and false positive rate as well as better scalabilities and robust. Their experiments proved that by fusing, the final detected rate of the entire system to every intrusions was higher than or close to the best detected rate of all the basic detectors to them, which makes the entire system has a relative high detected rate to all intrusions, and makes up for the flaw that single detecting method cannot has good detecting result to all intrusions.

Zhuang et al. [12] presented an efficient and effective model for collaborative alerts analyzing. Their system enhanced the alert verification using assets contextual information. By applying alert fusion and using a precisely defined knowledge base in the correlation phase, they also provides a method to get general and synthetic alerts from the large volume of elementary alerts. They discussed the similarity function of alerts and proposed an algorithm which can do correlation among alerts from different security tools.

Kirk et al. [14] proposed Information Mainfold (IM) as a novel framework for browsing and querying of multiple networked information sources. A system that demonstrated the viability of knowledge representation technology for retrieval and organization of information from structured and unstructured sources was presented in their work.

Levy et al. [15] proposed a similar but a more sophisticated work to use the data stored on the web to answer complex queries that go beyond keyword searches. They used a Relational Data model to store the data of cars (model, price, year) from various sources and proposed querying method for this data. Experimental studies indicating the scalability ofInformation Mainfold architecture and algorithms to severalhundreds of information sources was also presented by them. Their system used the source descriptions to purane efficiently the set of information sources for a given query and to generate executable query plans.

Cirzvegna et al. [17] proposed a methodology to learn to extract domain-specific information from large repositories with minimum user intervention. Their system was used in mining websites of their Computer Science Department to find out who works in a specific subsection and extract the list of projects and to find out who works with whom for a given project and period.

Arens et al. [18] described a system that addresses the important problem of how to efficiently integrate information from multiple databases. They emphasized on the query planning problem i.e. the selection of appropriate data sources and ordering the access to them. They also stressed on reformation of queries i.e. the use of knowledge both about the domain and the databases to modify queries to make retrieval plans for them more efficiently.

Januja et al. [19] addressed the issues of incomplete and inconsistent semantic information and knowledge integration by using schemes based on argumentation. They discussed the Argumentation-enabled Information Integration Web-DSS along with its syntax and semantics for semantic information integration and devise a methodology for sharing the results of Web@IDSS in Argument Interchange Format format. They presented the algorithms for knowledge integration and the prototype application for validation of results.

Vicknair et al. [20] performed a comparison of the relative usefulness of the relational database MySQL and the graph database Neo4j to store graph data. The goal of this study was to determine whether a traditional relational database system like MySQL, or a graph database, such as Neo4j, would be more effective as the underlying technology for the development of a data provenance system.

Angles et al. [21] performed a survey study of different graph database models and analysed its performance and utility when compared with other database models. Applications that are successfully modelled using graph were also discussed in their work.

Bordoloi et al. [22] described a method for transforming a relational database to a graph database model. In this approach, the dependency graphs for the entities in the system are transformed into star graphs. This star graph model is transformed into a hyper graph model for the relational database, which, in turn, can be used to develop the domain relationship model that can be converted into a graph database model.

Thus using the erstwhile concept of Information Mainfold and taking into account the advantages of graph databases over the relational one, this paper gives a method to integrate the data of movies from various sources over the web and intends to use a graph data model to store and access the obtained integrated data. Here we have deployed the knowledge graph model unlike the existing information integration systems. The knowledge graph model is more robust and generic way to graphically store and analyse the data.

III. Proposed Technique

With the advent of Internet over the years, most of the information is available on web in a well defined way, but on different websites. For retrieval and analysis, this information needs to be integrated on a common platform.

We propose a technique to populate a database of movies in

form of a graph by gathering information from various sources on the web. We expose this graph database via some APIs and graph processing languages to extract useful information which is further queried and analysed.

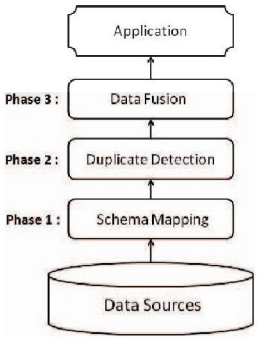

Fig. 1: Data Integration Process

We follow a standard 3 step data integration process [3] as shown in Fig. 1. Firstly, we need to identify corresponding attributes that are used to describe the information items in the source. The result of this step is schema mapping, which is used to transform the data present in the sources into a common representation. Secondly, the different objects that are described in the data sources need to be identified and aligned. In this way, multiple, possibly inconsistent representations of the same real world objects are found using duplicate detection techniques. In the last step, the data obtained is fused into a single representation, while resolving the inconsistencies in the data. This last step is referred to as Data Integration.

A. Technical Process

Based on the Data Integration process, we propose the following procedure that builds on top of it, and is described in Fig. 2. First, data stores, IMDb [5], Twitter [6], www.metacritic.com [7] and www.rottentomatoes.com [8] are selected and raw dumps of the data are obtained using urllib [24] package for python and other commands viz. Wget [25] and cURL [26]. The obtained data is parsed and refined further. The refined data is further processed and a key value based data structure is obtained which is uploaded over a graph database.

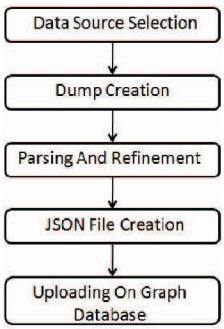

Fig. 2: Technical Process

B. Sources of Data

A large amount of data is available on the web for movies. We however restricted our work to four major sources:

1) **IMDb** [5]: The Internet Movie Database is an online store for information related to films.
2) **Twitter** [6]: Twitter is an online social networking and micro blogging service that enables us to know about the latest trends.
3) **Metacritic** [7]: Metacritic is a website that aggregates reviews of movies and keeps a numerical score for reviews.
4) **Rotten Tomatoes** [8]: Rotten Tomatoes is a website devoted to film reviews, news and information about films. It is widely known as a film review aggregator

C. Data Collection

Different commands and APIs are used to generate dumps of data from these various sources.

1) **urllib** [24]: It is a package that collects several modules for working with URLs.
2) **Wget** [25]: GNU Wget is an open source softwarepackage for retrieving files using HTTP, HTTPS and FTP, the most widely-used Internet protocols. It is a non-interactive command line tool, so it may easily be called from scripts, cron jobs, terminals without X-Windows support, etc.
3) **cURL** [26]: cURL is a tool to transfer data from or to a server, using one of the supported protocols (HTTP, HTTPS, FTP, GOPHER, DICT, TELNET, LDAP or FILE). The command is designed to work without user interaction.

D. Data Parsing and Refinement

After accumulating raw dumps of the data from the various sources using specific APIs and commands, the data is parsed and refined into a noise free format. Using the concept of regular expressions, suitable data structures, and other programming concepts in Python 2.7.2, data is obtained in tag free format.

Pythons built-in re [27] module is used for parsing which provides excellent support for regular expressions. The re module raises the exception re.error if an error occurs while compiling or using a regular expression. Functions like matching functions re.match(), re.findall(), search function re.search(), search and replace function re.sub(), split function re.split(), compile function re.compile() from this module are used for parsing the raw dumps.

E. JSON File Creation

The refined data is converted into a key value data structure which is stored in Python ordered dictionary OrderdDict. There are three keys in this ordered dictionary namely mode, vertices and edges. The value part of the vertices and edges key is also a dictionary. Thus it forms a nested dictionary type of data structure. Using JSON module from Python [28], this dictionary based data store is stored in a properly formatted JSON file [29]. The JSON file thus obtained is uploaded on to TITAN [30], a graph database via Gremlin, a graph query language [31].

F. Uploading on Graph Database

A set of gremlin queries for uploading a JSON file have been shown in Fig. 3 where storage backend is HBase, a non-relational, distributed database [32] and single machine environment is used for testing purpose.

```
gremlin> config = new BaseConfiguration()
==>org.apache.commons.configuration.BaseConfiguration@1b9f7e8
gremlin> config.setProperty("storage.backend", "hbase")
==>null
gremlin> config.setProperty("storage.hostname", "127.0.0.1")
==>null
gremlin> g= new TinkerGraph()
==>tinkergraph[vertices:0 edges:0]
gremlin> g.loadGraphSON('sample.json')
==>null
```

Fig. 3: Uploading JSON

Graph database refers to any storage system that can contain, represent and query a graph consisting of a set of vertices and a set of edges relating to a pair of vertices. Titan [30] is a scalable graph database optimized for storing and querying graphs containing hundreds of billions of vertices and edges distributed across a multi-machine cluster. It is open source with the liberal Apache 2 license. It supports storage Backends like Apache HBase [32], Apache Cassandra [33], Oracle BerkeleyDB [34] and Akiban Persistit [35]. Titan is also a transactional database and can support thousands of concurrent users executing complex graph traversals. A sample graph stored in TITAN is visualized as shown in Fig. 4.

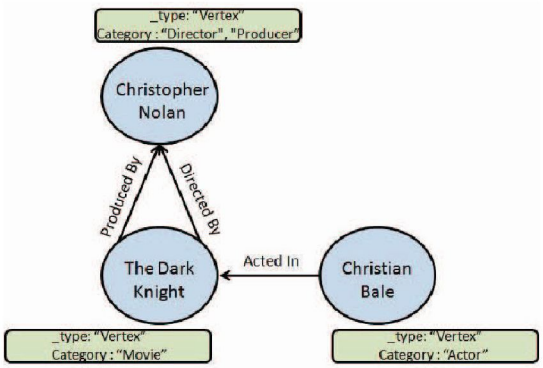

Fig. 4: A Sample Dependency Graph in TITAN

IV. Results

The system thus developed is experimented with top 250 movies taken from IMDb. Every movie is represented as vertex in this database and is associated with attributes like cast, budget, released dates, genre, runtime, metascore, IMDb rating, user reviews, rotten reviews, critic reviews, Twitter tweets etc. These attributes would need a number of relational tables to store the data to prevent anamolies and to maintain the database integrity. In a relational database model, we apply multiple JOIN queries to obtain the results which makes the system slow and less efficient. To overcome this, the presented approach stores complete data in a single graph with the relationships expressed as edges and supports all kinds of complex querying and traversal in linear time in a much efficient and hastle free way.

In the following presented gremlin queries g represents the graph, V represents Vertices, E represents Edges, outE represents out edges from a vertex, inV represents vertex to which an edge points , inE represents the incoming edges of a vertex.

Names of all persons who has been director and producer and actor

```
g.V.filter{"Director" in it.category
&& "Producer" in it.category
&& "Actor" in it.category}
```

Release Date of a movie in any country say USA

```
v.outE.filter{it.label=="Released On"
&& it.inCountry=="USA"}
```

Integrated Movie Rating of a movie combined from IMDb, Rotten Tomatoes and Metacritic

```
v.outE.filter{it.label=="Integrated
Rating"}.inV
```

Tweets in trend on www.twitter.com about a movie

```
v.outE.filter{it.label=="Tweet"}
```

Rating of rotten and www.rottentomatoes.com fresh of a movie from www.rottentomatoes.com

```
v.outE.filter{it.label=="Rotten Rating"
&& it.label=="Fresh Rating"}
```

All the movies with same rating as The Dark Knight Rises from www.imdb.com

```
g.v("The Dark Knight Rises").outE
.filter{it.label=="Rating"}.inV.inE
```

Reviews of a movie from www.rottentomatoes.com

```
v.outE.filter{it.label=="Rotten Reviews"}
```

Critic Reviews of a movie from www.metacritic.com

```
v.outE.filter
{it.label=="Metacritic critic reviews"}
```

Awards won by a movie from www.imdb.com

```
v.outE.filter{it.label=="Awards Won"}
```

The experimental results in this section suggests that a graph database can handle a wide range of graph queries even with big data. In a relational database, these queries require

heavy join operations even with full indexing. On the other hand, a graph database suited for this type of query processing avoids explicit intermediate and dummy table creation. The benefit of the graph database comes from not only its direct data representation and storage, but also its intuitive query structure. The compact representation of the graph query facilitated the management, user validation and exploration of the analytic intent of the query.

V. Conclusion and Future Work

With rapid increase in information on the web, there arises a need to integrate and make it available at a common store so as to avoid multiple searching. Concept of Information Integration came way back in 1995 when Information Mainfold was proposed as a novel framework for browsing and querying of multiple networked information sources. Since then, there had been a subtle progress in this field and researchers have come up with more optimum methods of Information Integration and Querying.

Building on top of the existing methodologies, we have developed a system that populates a graph database by integrating data of movies from different sources over the web. We thus obtained a graph database for movies by integrating information like cast, budget, shooting locations, genre, critic reviews, tweets etc. from four sources namely IMDb, Twitter, www.metacritic.com and www.rottentomatoes.com. Putting together the basic concept of Information Integration along with our proposed technique, we thus analysed and presented a graphical way of efficiently storing the data.

The work opens many areas where it can further be extended. An application can be build on top of this data store that could be used as a Meta Search Engine for movies and with use of Machine Learning Algorithms, a recommender system could also be proposed. The tweets and critic reviews that are currently incorporated in our work could be further exposed to Natural Language Processing and Semantic Analysis, to extract emotions from them. This will further lead to generation of a system which can predict the performace of a new movie based on twitter trends, IMDb rating, metacritic rating and rottentomatoes score using Machine Learning techniques.

References

[1] Romano, N.C., Roussinov, D., Nunamaker, J.F., Chen, H.: Collaborative Information Retrieval Environment: Integration of Information Retrieval with Group Support Systems. In: Proc. of the 32nd Hawaii International-Conference on System Sciences. . Maui, pp. 5 8. Society Press, Hawaii (1999)

[2] Chidlovskii, B.: Information Extraction from Tree Documents by Learning Subtree Delimiters. In: Proc. IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), pp. 38. ACM Digital Library, Acapulco, Mexico (2003)

[3] Bleiholder, J., Naumann, F.: Data fusion. ACM Computing Surveys 41(1), 141 (2008)

[4] Mashable, http://mashable.com/2012/09/26/graph-databases/

[5] IMDb: Internet Movie Database, http://www.imdb.com/

[6] Twitter, https://twitter.com/

[7] Metacritic, http://www.metacritic.com/

[8] Rotten Tomatoes, http://www.rottentomatoes.com/

[9] Jiangfan, F., Wei, W.: Multiple Spatial Model Fusion in Heterogeneous Sensor Networks. International Journal of Multimedia and Ubiquitous Engineering 9(2), 114 (2014)

[10] Bass, T.: Intrusion detection systems and multisensor data fusion. Communications of the ACM 43(4), 99105 (2000)

[11] Zhao, X., Jiang, H., Jiano, L.: A Data Fusion Based Intrusion Detection Model. In: First International Workshop on Education Technology and Computer Science, pp. 10171021. IEEE Press, Wuhan, China (2009)

[12] Zhuang, X., Xiao, D., Xuejiao, L., Zhang, Y.: Applying Data Fusion in Collaborative Alerts Correlation. In: International Symposium on Computer Science and Computational Technology, pp. 124127. IEEE Press, Shanghai, China (2008)

[13] Chatzigiannakis, V., Androulidakis, G., Pelechrinis, K., Papavassiliou, S., Maglaris, V.: Data fusion algorithms for network anomaly detection: classification and evaluation. In: Third International Conference on Networking and Services (ICNS07), pp. 5051. IEEE Press, Athens, (2007)

[14] Kirk, T., Levy, A.V., Sagiv, Y., Srivastava, D.: The Information Mainfold. In: Proc. of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments, pp. 8591. AAAI Digital Library, Palo Alto, California, USA (1995)

[15] Levy, A.Y., Rajaraman, A., Ordille,J.: Querying Heterogeneous Infomation Sources Using Source Descriptions. In: Proc. of 22th International Conference on Very Large Databases, pp. 251262. Morgan Kaufmann, Bombay, India (1996)

[16] Halevy, A., Rajaraman, A., Ordille,J.: Data Integration: The Teenage Years. In: Proc. of the 32nd international conference on Very large Databases, pp. 916. ACM Digital Library, Seoul, Korea (2006)

[17] Ciravegma, F.: Integrating Information to Bootstrap Information Extraction from Web Sites. In: Proc. IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), pp. 914. ACM Digital Library, Acapulco, Mexico (2003)

[18] Arens, Y., Knoblock, C.A.: Planning and Reforming Queries for Semantically-Modeled Multidatabase Systems. In: Proc. of the Second International Conference on Information and Knowledge Management, pp. 423432. ACM Digital Library, Washington, DC, USA (1993)

[19] Januja, N.K., Hussain, F.K., Hussain, O.K.: Semantic information and knowledge integration through argumentative reasoning to support intelligent decision making. Information Systems Frontiers 15(2), 126 (2013)

[20] Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., Wilkins, D: A comparison of a graph database and a relational database: a data provenance perspective. In: Proceedings of the 48th Annual Southeast Regional Conference (ACM SE10), pp. 16. ACM Digital Library.

[21] Angels, R., Gutierrez, C.: Survey of graph database models. In: ACM Computing Surveys (CSUR), Volume 40 Issue 1, February 2008, Article No. 1.

[22] Bordoloi, S., Kalita, B.: Designing Graph Database Models from Existing Relational Databases. In: International Journal of Computer Applications (IJCA13), Volume 74Number 1.

[23] Park, Y., Shankar, M., Park, B., Ghosh, J.: Graph databases for large-scale healthcare systems: A framework for efficient data management and data services. In: IEEE 30th International Conference on Data Engineering Workshops (ICDEW), 2014 ,pp. 1219

[24] Python urllib Library, https://docs.python.org/2/library/urllib.html

[25] Wget, https://www.gnu.org/software/wget/

[26] cURL project , http://cURL.haxx.se/

[27] Python re Module , https://docs.python.org/2/library/re.html

[28] Python JSON Module, https://docs.python.org/2/library/json.html

[29] JSON: JavaScript Object Notation, http://json.org/

[30] TITAN: Graph Database, http://thinkaurelius.github.io/titan/

[31] Gremlin Documentation, http://gremlindocs.com/

[32] Apache HBase, http://HBase.apache.org/

[33] Apache Cassandra, http://cassandra.apache.org/

[34] Oracle Berkely DB, http://www.oracle.com/technetwork/database

[35] Akiban Persistit, https://github.com/pbeaman/persistit