1 Introduction

2 OLAP and Graph Databases

This chapter introduces the main concepts that forms the theoretical foundation necessary to better understand the work presented in this dissertation. Initially, we will explore the definition of Data Warehouse, Multidimensional Model and OLAP tools. Then, we will dive into concepts related to Graph and Graph Databases.

2.1 Data Warehouse

According to (INMON, 2005), a data warehouse (DW) is “a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management’s decisions”. The first important aspect of a data warehouse is that it is subject-oriented, which means that the information stored in the DW is related to the company’s subject. Consider a retail company for example: the main subjects can be product, sale, vendor and customer, therefore the data in the warehouse will be related to these entities.

The second characteristic of a data warehouse is that it is integrated, which means it contains data coming from multiple different sources. During the process of loading the warehouse, the data is converted, formatted, normalised and go through any other process to make the final information stored in the warehouse consistent. Another important aspect of a data warehouse is that it is nonvolatile, which means that the data in the warehouse does not get updated as operational environment. The warehouse is loaded in batches and it stores a snapshot, creating a history of the data. The final important aspect of a DW is that every record of data contains some sort of a time attribute to mark the moment in which the record is accurate.

Figure 2.1 shows the basic elements of a data warehouse (KIMBALL; ROSS, 2011). The Operational Source Systems capture and store the transactions of the business and they are considered elements outside of the data warehouse, since there is no control on the content or the format of the data that they store. The Data Staging Area is where the process of Extract-Transformation-Load (ETL) is made: the data is extracted from the operations source system, then they are transformed in order to integrate all the information in a consistent format, and finally the data is loaded into the presentation area.

The Data Presentation Area is where the data is organised, stored and accessed by analytical applications. According to (KIMBALL; ROSS, 2011), this area should be composed by a series of integrated data marts, which are repositories that present the data for a single process of the organisational business. A data mart stores atomic data and

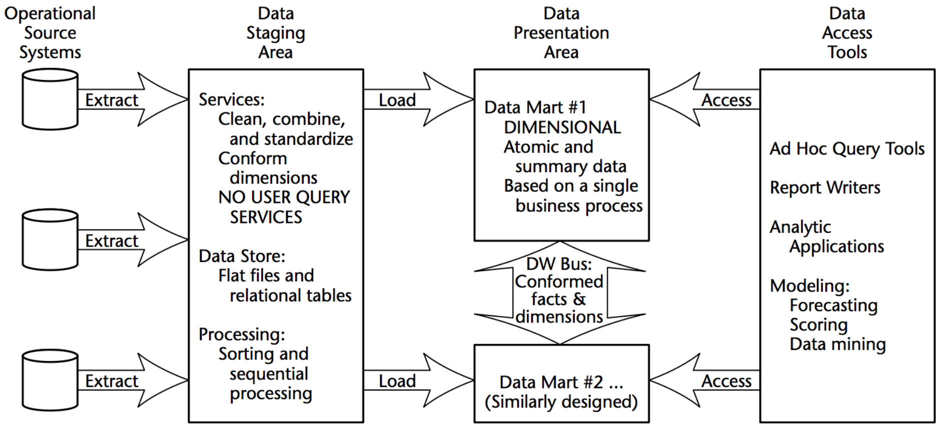


Figure 2.1 – Elements of a Data Warehouse (KIMBALL; ROSS, 2011)

organizes them in a model that is more legible to humans. The most popular technology used to implement data marts adopted by the industry is On-Line Analytical Processing (OLAP), that will be covered in more detail along this chapter.

Finally, the Data Access Tools query the data in the presentation area. This element is formed by a set of different applications, since simple ad hoc queries until complex data mining application.

2.2 Multidimensional Model

The main functionality of a data warehouse is to facilitate multidimensional analysis (OLAP COUNCIL, 1997). This type of analysis reduces the number of misinterpretations by aligning the data with the analyst's mental model of the business. The multidimensional analysis provides an easy navigation through the database, showing specific subsets of data in different orientations and performing analytical calculations.

In order to perform multidimensional analysis, the data must be organised in a multidimensional model, where information is stored in a multidimensional array called *hypercube* or *cube* (VASSILIADIS, 1998). A cube is composed by cells that store *measures* aggregated by the data *dimensions*. A *dimension* is a cube's structural attribute formed by a list of properties that are similar to each other according to the user's perception of the data. Measures are the values being analysed by the user. Figure 2.2 shows an example of the dimensions and measures extracted from a sales application.

As shown in Figure 2.2 , each dimension (Geography, Time and Item) can be associated with an hierarchy of data aggregation levels. This feature allows user to view the data from different levels of details, for example: the measure *Sales* aggregated by *Region* can be detailed by *Country* or even further detailed by *City*.

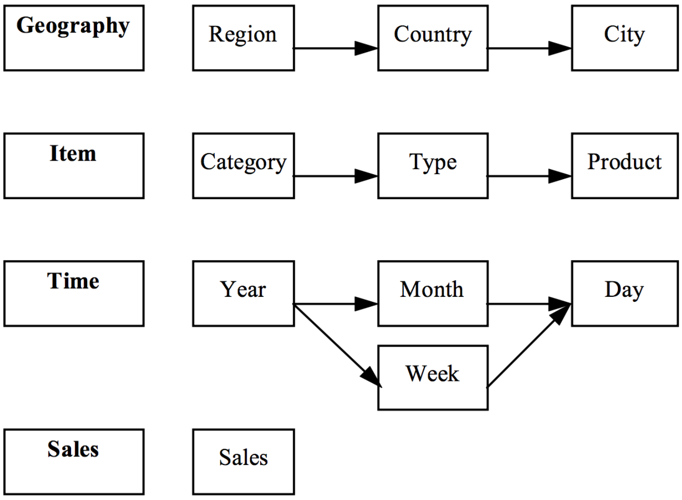


Figure 2.2 – Example of dimensions and measure for a sales application (VASSILIADIS, 1998)

2.3 OLAP

On-Line Analytical Processing (OLAP) is a category of technological tools used by analysts and managers to extract new knowledge from consolidated enterprise data (OLAP COUNCIL, 1997). In order to achieve this goal, data are organised in cubes and stored following multidimensional models. The access to this data should be fast, consistent and interactive and it should provide various views of information, reflecting the different dimensions of an enterprise as perceived by the user.

Usually, the data loaded into an OLAP system comes from a data warehouse. According to (INMON, 2005), the relationship between OLAP systems and data warehouse is complimentary: while OLAP offers control and flexible ways to explore data in different dimensions and hierarchy levels, data warehouse provides a robust data source for the OLAP system, where up-to-date data is available already extracted and properly integrated.

2.3.1 Types of OLAP Systems

There are two approaches for the physical model of an OLAP system (VASSILIADIS; SELLIS, 1999): Multidimensional On-Line Analytical Processing (MOLAP) and Relational On-Line Analytical Processing (ROLAP) Architectures.The MOLAP architecture provides a direct multidimensional view of the data. This approach stores the data in a Multidimensional Database Management System (MDBMS), which use n-dimensional arrays that contains the measures of the cube. This type of DBMS has a better performance than traditional Relational Databases, but it is more difficult to manage updates.

The ROLAP architecture is a multidimensional interface to relational data. This approach uses a traditional Relational Database Management System (RDBMS) to store the data organised in a star or snowflake schema. A star schema is formed by one or more dimension tables and one centralised fact table, which stores the measures of interest for the OLAP system. Figure 2.3 shows an example of a star schema, where the tables TIME, GEOGRAPHY, ACCOUNT and PRODUCT are dimension tables and SALES is the fact table.

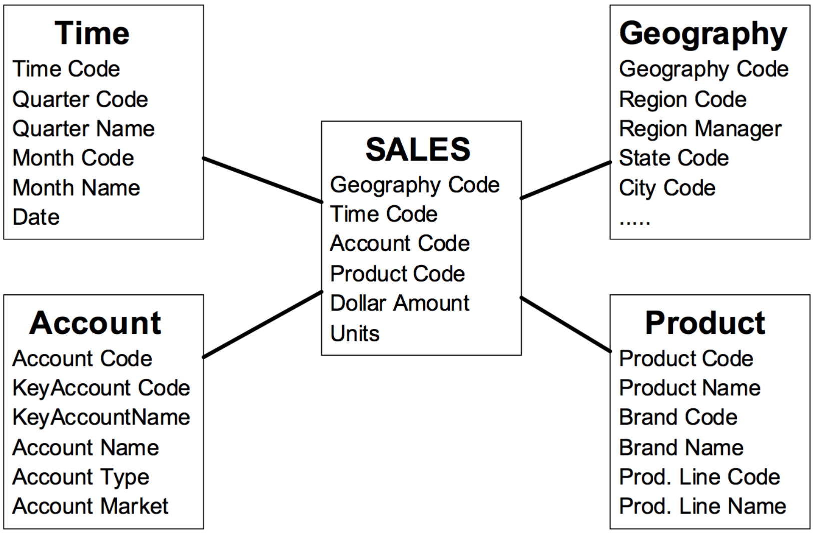


Figure 2.3 – Example of a Star Schema (VASSILIADIS; SELLIS, 1999)

Despite MOLAP architecture has the advantage of relying on a multidimensional native storage mechanism, the ROLAP approach allows easy integration with existing relational information systems and has the advantage of relational data being more efficiently stored than multidimensional data.

2.3.2 OLAP Operators

As mentioned earlier in this chapter, an OLAP system should provide a fast and interactive way for the user to explore the data stored in the cube. These are the main OLAP operations that can be used to navigate and manipulate different dimensions of the data (VASSILIADIS, 1998):

Roll-up Aggregates information along one dimension, summarising data on a higher level in its hierarchy. Consider the dimensions shown in Figure 2.2 and the measure of total dollar amount of sales per city, we can perform a roll-up operation to obtain the total dollar amount of sale per state.

Drill-down Detail information along one dimension, navigating the data from a higher to a lower hierarchy. Consider the measure of total dollar amount of sales per year, we can drill-down this query to obtain the total dollar amount of sales per month.

Slice Selects a slice of the cube according to user-specified dimension values. For instance, the user can select to view the total dollar amount of sales for the year of 2016.

Dice Selects a subcube from the original data according to user-specified conditions referred to more than one dimension. For instance, the user can select to view to total dollar amount of sales for the year of 2016 in the city of Recife.

Pivot Changes the orientation of dimensions in the cube, i.e. swap a row dimension to a column dimension.

There are other operations that can be performed in a OLAP system, but the ones shown above are considered the basic set of operations to support dynamic multidimensional analysis (INMON, 2005).

2.4 Graph

Graph is a data structure formed by a set of vertices (or nodes) $V = \{v_1, \dots, v_n\}$ and a set of pair of vertices called edges (or relationships) $E = \{v_1v_2, \dots, v_nv_m\}$. It can be represented graphically (where the vertices are shown as circles and edges are shown as lines, as shown in Figure 2.4) or mathematically in the form $G = (V, E)$ (TOBERGTE; CURTIS, 2013).

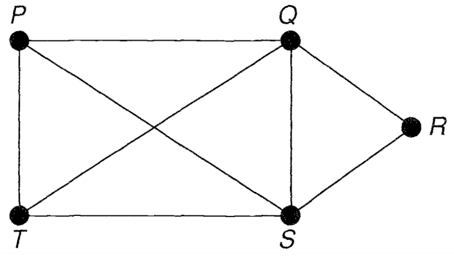


Figure 2.4 – Example of graph (TOBERGTE; CURTIS, 2013)

There are several real world applications that can take advantage of a graph model, specially the ones where the relationship between entities is an important information to be represented (MILLER, 2013). Consider, for example, a social network application similar to Twitter, where a user can follow another user. In this scenario, we can represent a user as a vertex and the relationship between users as an edge, as shown in Figure 2.5

2.4.1 Graph Theory

Graph Theory is a branch of Mathematics dedicated to the study of graph structures (TOBERGTE; CURTIS, 2013). Given the definition of a general graph explained above, there are several classifications and concepts associated with graph structures.

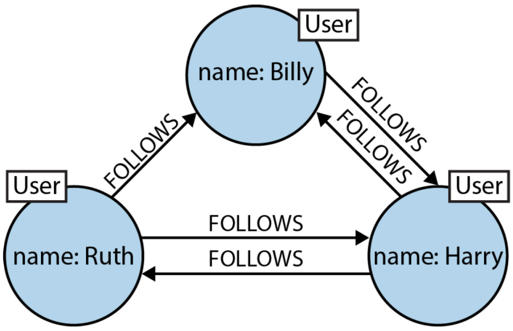


Figure 2.5 – Example of graph representation of a social network (MILLER, 2013)

Loop When an edge starts and finishes in the same vertex, as the one shown in Figure 2.6 starting at vertex T and finishing at vertex T.

Multigraph When a graph allows multiple edges connecting the same pair of vertices, as illustrated in Figure 2.6 with two edges connecting vertices Q and R.

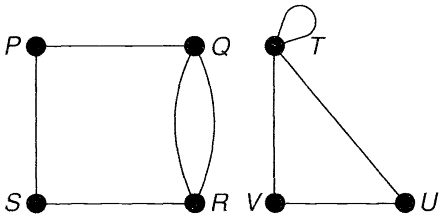


Figure 2.6 – Example of a multigraph with a loop (TOBERGTE; CURTIS, 2013)

Simple Graph When a graph does not have loops or multiple edges, as the one depicted in Figure 2.4.

Complete Graph When each pair of distinct vertices are connected to each other, as shown in Figure 2.7.

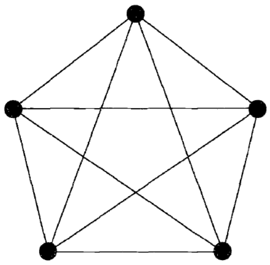


Figure 2.7 – Example of a complete graph (TOBERGTE; CURTIS, 2013)

Directed Graph When the edges have directions expliciting the start and end of the connection, as illustrated in Figure 2.8.

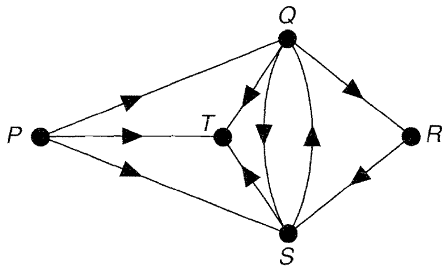
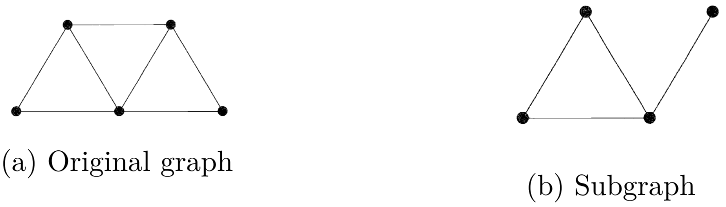


Figure 2.8 – Example of a directed graph (TOBERGTE; CURTIS, 2013)

Subgraph It is a graph obtained from another graph G, where its vertices are a subset of the vertices of G and its edges are a subset of the edges of G. Figure 2.9 shows in 2.9b a subgraph of the graph in 2.9a.

Path It is a subgraph obtained by a sequence of adjacent and distinct vertices.


(a) Original graph

(b) Subgraph

Figure 2.9 – Example of a subgraph (TOBERGTE; CURTIS, 2013)

There are other types of classifications of a graph and other concepts associated to it (TOBERGTE; CURTIS, 2013). These are the main definitions for the purposes of understanding the work presented here.

2.4.2 Network Analysis

There are different kind of analysis that can be done depending on the nature of the graph data. Due to the work presented here, we will explore the aspects of social network analysis, which focus on the structure of relationships (edges) between entities (vertices) in the graph. Several types of analysis can be performed in a graph, but the measure of a node's centrality is historically one of the most studied cases of analysis (FREEMAN, 1978). A common application for a centrality measure is to find out what are the focal points in a social network, i.e. who are the people that are the most connected to other people? There are three types of centrality measures (FREEMAN, 1978):

Degree Centrality This measure is given by the number of adjacent vertices. Considering a social network, a person (vertex) that has the greatest number of connections can be considered one of the focal points of the network, since it can directly read the most nodes in the graph. Given a graph with n vertices, the maximum value the degree centrality of a vertex in the graph can be is $n-1$. The degree centrality of a vertex is related to its potential communication activity.

Betweenness Centrality This measure is given by the frequency in which a vertex is present in the shortest path between other vertices. Considering a social network, a person with high betweenness centrality is capable of influencing others by intercepting connections between several vertices in the graph. The betweenness centrality of a vertex is related to its potential control of communication.

Closeness Centrality This measure is given by the inverse of the average distance of the shortest path of a vertex between other vertices in the graph. Considering a social network, a person with high closeness centrality do not depend as much on others to communicate with other people in the network. The closeness centrality of a vertex is related to its potential independency and efficiency to control communication.

2.5 Graph Databases

Graph Databases are an alternative to Relational Database Management Systems (RDBMS), which are commonly used in the industry since the early 1980's. Despite the popularity of RDBMSs, GDBs allow the storage of data in graph model, which is a more natural way to represent information for some applications, such as social networks, semantic web pages and recommendation systems (MILLER, 2013).

The most popular form of graph model is the labeled property graph model (ROBINSON; WEBBER; EIFREM, 2015). The main characteristics of a labeled property graph are:

- Nodes (vertices) and Relationships (edges) can contain properties, i.e. key-value pairs
- Nodes can be labeled with one or more labels
- Relationships are named and directed, i.e. always have start and end nodes

The formal definition for a labeled property graph is given by $G = (V, E, L_V, L_E, A_V, A_E)$, where:

- V is a set of vertices (nodes)
- $E \subseteq V \times V$ is a set of edges
- L_V is a set of node labels and L_E is a set of edge labels

- $A_V = \{a_1, \ldots, a_n\}$ is a set of node attributes, where $a_i = (k_i, m_i)$ is a key-value pair. Each vertex $v_i \in V$ is associated with an attribute vector. $A_E = \{b_1, \ldots, b_n\}$ is the set of edge attributes defined in the same way as node attributes.

An example of a labeled property graph was shown in Figure 2.5, where the nodes have the label "User" and the property "name" and the relationships are named "Follows" and have arrows indicating the direction of the edge.

2.5.1 Historical Overview

Scientific research related to graph data models were continuously published between 1980's and the first half of 1990's. Then, the database community attention turned to semistructured data model, due to the emergence of XML and the growth of hypertext document applications (ANGLES; GUTIERREZ, 2008). From this period, the main focus of the graph databases proposed was to establish a better way to represent data and methods to retrieve and manipulate data modeled as graph.

Recently this area has gained again attention from the community, due to the emergence of trendy projects (chemistry, biology, social network, semantic web, among others) where the importance of information relies not only in the entities but also in the relationship between them (ANGLES, 2012). Most recent implementations of graph databases are concerned with handling increasing amount of data and improving performance in retrieving and manipulating data.

The most popular graph database according to DBEngines[1] website is Neo4J[2]. It is an open source native graph storage database implemented in Java. Neo4J has its own query language called Cypher, which can be used to create, update and retrieve nodes and relationships. This graph database also provides high availability and scalability for big volume of graph data.

Besides Neo4J, there are other popular graph databases according to DB Engine site. OrientDB[3] is a NoSQL Multi-Model database, which means it stores data in the form of documents, key-value stores, objects, graph and others. Like Neo4J, OrientDB is also implemented in Java, but it uses an extended version of SQL that includes special operators to manipulate graph. This database allows the creation of a pre-defined data schema and the definition of complex data type, like dates, decimals and binary objects (BLOB).

TitanDB[4] is a native graph database implemented in Java. In order to establish connection with the hard disk, Titan needs to be linked to a data storage system -

[1] https://db-engines.com/en/system/Neo4j

[2] https://neo4j.com/

[3] http://orientdb.com/orientdb/

[4] http://titan.thinkaurelius.com/

ht]

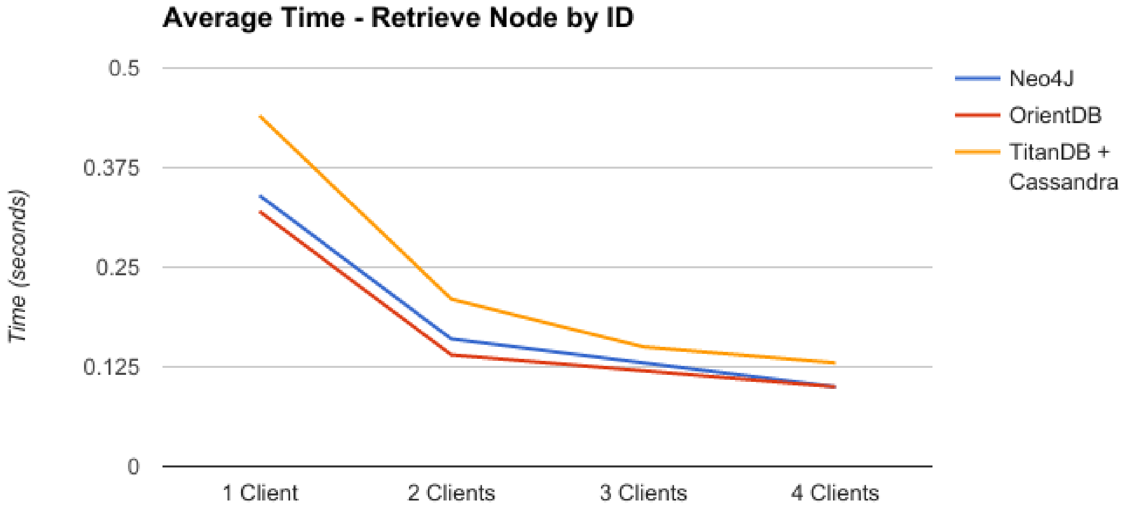


Figure 2.10 – Chart with mean time that each DB took to retrieve a node by its id

Cassandra[5], HBase[6] or BerkeleyDB[7] - that is suitable for the application. The creation of nodes, edges and the submission of queries can be done through a Java API or through a Gremlin server.

2.5.2 Comparing Graph Databases Performance

One of the basic tasks when manipulating a database is to retrieve an object given its id. The experiment described in (JOUILI; VANSTEENBERGHE, 2013) records the average time to retrieve a node by its id, given an 500000 nodes graph and 4 clients executing this action 200 times. This process was executed using Neo4J, OrientDB and TitanDB using Cassandra as backend.

According to Figure 2.10, Neo4J and OrientDB have similar performance, being OrientDB slightly faster to retrieve a node given its id. On the other hand, TitanDB using Cassandra as backend has a bad performance compared with the other DBs. However, TitanDB can be connected with other backend services, which can improve its performance for this task.

Another common concern among developers is the amount of memory taken by the database. In the experiment shown in (MCCOLL et al., 2014), a graph containing 32,000 nodes and 256,000 edges was stored in Neo4J, OrientDB and TitanDB and the amount of memory required by each database was measured.

[5] http://cassandra.apache.org/
[6] https://hbase.apache.org/
[7] http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/overview/index.html

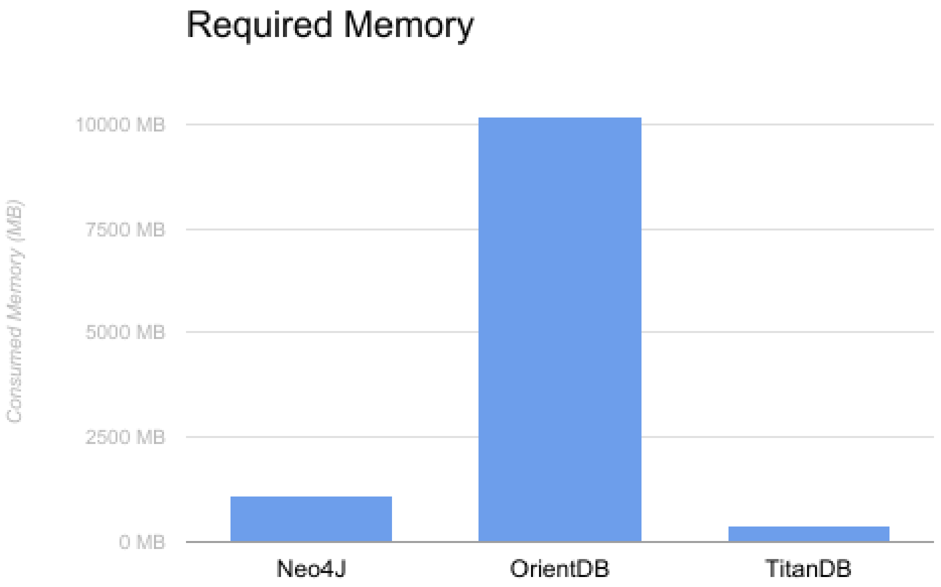


Figure 2.11 – Chart with the amount of memory required for each DB

From the Figure 2.11, we can notice that OrientDB is the one that consumes the most, reaching up to 10.156 MB of memory. On the other hand, TitanDB requires less memory, only needing 388 MB to store the graph.

2.6 Final Considerations

In this chapter, we discussed the main concepts related to the work presented in this dissertation. Initially, the definition of Data Warehouse and Multidimensional Model were introduced, followed by a detailed view on different types and operators an OLAP system can have. Then, we covered the main components of a graph and how it can be classified according to different characteristics. Finally, we explained the definition of a Graph Database and went through a historical overview of such systems, as well as a comparison of the most popular Graph DBs nowadays. In the next chapter, we will review the most important research published that implements a OLAP system using Graph Databases.

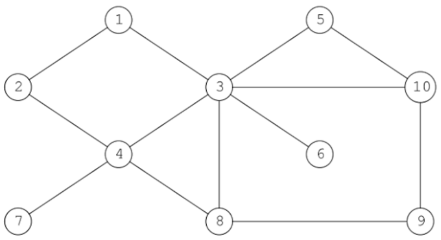
3 Graph Cubes: State of the Art

This chapter presents the most important research works published in the area of OLAP systems implemented using graph databases. The first work presented here dates back to 2011 and introduces several concepts - such as graph cube, aggregated graphs and cuboid queries - that serve as basis for other papers published since then.

3.1 Graph Cube: On Warehousing and OLAP Multidimensional Networks

One of the main works in graph analysis using OLAP methods is described in (ZHAO et al., 2011), and it introduces several concepts that will be used by other authors to describe their work in this area. Graph Cube is one of those concepts and it is defined as a multidimensional model that extends multidimensional networks to provide decision support features. A multidimensional network is a graph $N = (V, E, A)$, where V is a set of vertices, E is a set of edges and A is a set of vertex-specific attributes. Each vertex in the graph is a multidimensional tuple and the attributes of the vertex define the graph cube dimensions.

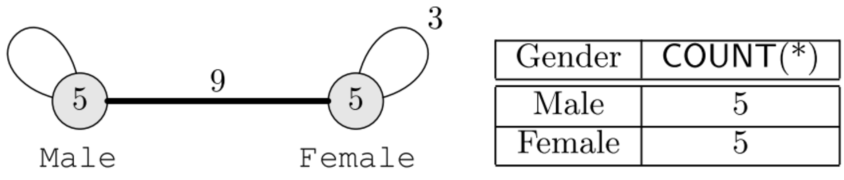
A graph cube is formed by all possible aggregate networks calculated from the original multidimensional network. An aggregate network (often called cuboid) is a summarisation of the original graph with respect to one or more dimensions, which is calculated by applying an aggregate function (e.g. COUNT, SUM, AVERAGE, among others) on the vertices attributes. Consider the multidimensional network illustrated in Figure 3.1.



ID	Gender	Location	Profession	Income
1	Male	CA	Teacher	$70,000
2	Female	WA	Teacher	$65,000
3	Female	CA	Engineer	$80,000
4	Female	NY	Teacher	$90,000
5	Male	IL	Lawyer	$80,000
6	Female	WA	Teacher	$90,000
7	Male	NY	Lawyer	$100,000
8	Male	IL	Engineer	$75,000
9	Female	CA	Lawyer	$120,000
10	Male	IL	Engineer	$95,000

Figure 3.1 – Multidimensional network (ZHAO et al., 2011)

The vertices of the graph presented in Figure 3.1 represent individuals and the edges represent the relationship between these individuals. The table on the right side of Figure 3.1 describes the attributes of each vertex: an unique *ID*, *Gender*, *Location*,



Gender	COUNT(*)
Male	5
Female	5

Figure 3.2 – Aggregate Network by *Gender* dimension (ZHAO et al., 2011)

Profession and *Income*. Figure 3.2 shows the cuboid obtained by applying the function COUNT on the attribute *Gender*: two nodes represent the possible values for the attribute (Male and Female) and they contain the number of nodes in the original graph with the respective *Gender* value. It is important to notice that the relationship between individuals was also aggregated in the resulting network, e.g. the aggregate network show that there are 9 relationships between male and female individuals in the original graph.

The dimension of a cuboid is given by the set of non-aggregate dimensions of the cuboid. For instance, the cuboid in Figure 3.2 has the dimension *Gender*. A cuboid A' is an ancestor of another cuboid A'' if $dim(A')$ contains $dim(A'')$. Given these definitions, all possible cuboids of a graph cube can be organised in a graph cube lattice, ordering the cuboids according to its ancestors. A multidimensional network N with n dimensions has 2^n cuboids in the graph cube. Figure 3 shows the graph cube lattice for the multidimensional network introduced in Figure 3.1, considering only *Gender*, *Location* and *Profession* as dimensions.

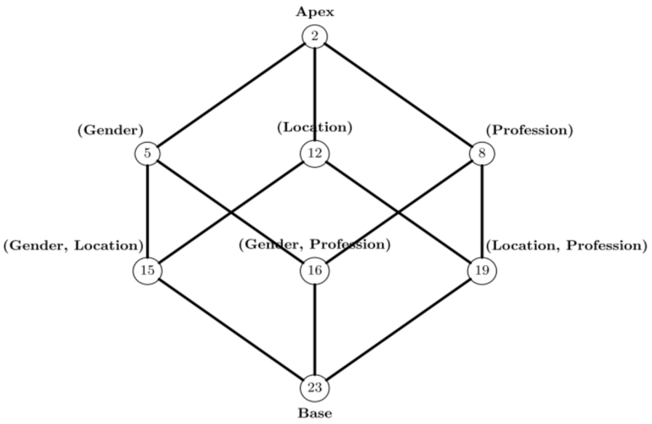


Figure 3.3 – Graph Cube Lattice (ZHAO et al., 2011)

The paper proposes two OLAP query models:

Cuboid Query Aggregate vertices and edges based on the dimension requested in the query and can work with any aggregate function (SUM, AVG, etc). For instance, consider a graph with nodes containing the attributes (*Gender*, *Location*, *Profession*).

A cuboid query for this graph can be (*Gender*, *, *) which will result in an aggregate graph showing all the vertices with the same Gender value aggregated and the edges also aggregate by the function COUNT.

Crossboid Query Returns the aggregate network between two or more cuboids structures. An example of a crossboid query can be the aggregate network between an user with $ID = 3$ and all the locations

Given that the size of the graph cube lattice is exponential with respect to the number of dimensions of the original multidimensional network, the paper proposes a partial materialisation in order to process queries. The partial materialisation is implemented using a greedy algorithm that selects k cuboids ($k < 2^n$) to be materialised according to the benefit of those cuboids to improve the cost for query evaluation.

The graph cube implementation described above is also studied by other works. In (DENIS; GHRAB; SKHIRI, 2013), a distributive approach is proposed using map reduce jobs to calculate each step of the aggregation process. In (KHAN et al., 2014), a new algorithm to compute the Graph Cube (iGraphCubing) is proposed, using a new prunning method based on the Structural Significance measure.This measure takes into account three factors:

- The diversity of the attribute value in the neighborhood for each vertex
- The clustering coefficient
- The density around each vertex

The work in (ZHAO et al., 2011) presents experiments evaluating the effectiveness and the efficiency of the method proposed. The effectiveness was evaluated by analysing OLAP queries submitted to the graph cube. The efficiency was evaluated by analysing the graph cube performance depending if it was fully or partially materialised in disk. In conclusion, the paper proposes an implementation of a graph cube obtained from an homogeneous attributed graph, but it does not consider heterogeneous networks neither attributed edges

3.2 Graph OLAP: Towards Online Analytical Processing on Graphs

The Graph OLAP framework proposed in (CHEN et al., 2008) takes as input a set of network snapshots, where each snapshot contains a graph and a set of informational attributes that describes the snapshot. In this scenario, the paper defines two dimension types: informational dimensions (formed by the informational attributes) and topological dimensions (formed by the attributes of the nodes and edges of the graph). The authors distinguish different semantics for OLAP operations in graphs, so these operations are categorised into Informational OLAP (I-OLAP) and Topological OLAP (T-OLAP).

The framework gives as output an aggregate graph with a summarised view of the snapshots set. The type of the aggregate graph returned by the framework can also be categorised in Informational Aggregate Graph (I-aggregate graph) and Topological Aggregate Graph (T-aggregate graph).

Informational Aggregate Graph is computed based on a set of network snapshots that have informational dimensions with same values. Figure 3.4 shows an example of an i-aggregate graph composed by snapshots describing the co-author relationship between individual authors. The co-author relationship is grouped by a certain conference (*sigmod*, *vldb*, *icde*), by a class of conferences (*db-conf*), by a specific period of time (*2004*, *2005*) and by a group of time periods (*all-years*). It is important to notice that the graph in Figure 3.4 shows different levels of aggregations for each co-author relationship.

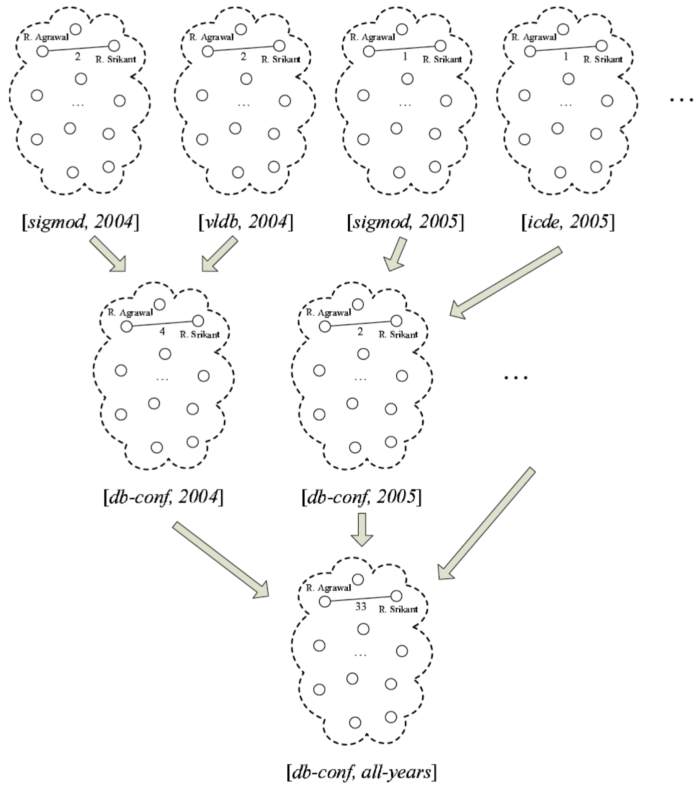


Figure 3.4 – Example of Informational Aggregate Graph (CHEN et al., 2008)

Classic OLAP operations in an i-aggregate graph can be interpreted as follows:

Roll-up Aggregate multiple snapshots to form a higher level summary of information

Drill-down Return to lower-level snapshots from aggregate graph

Slice / dice Select a subset of snapshots based on informational dimensions

Topological Aggregate Graph is obtained based on a single network, where the nodes are the result of applying the aggregate function to the nodes of the original network with the same attribute value. Figure 3.5 shows an example of a t-aggregate graph where the information about co-author relationship between individual authors in one snapshot was aggregated into co-author relationship between the institutions the authors belong to.

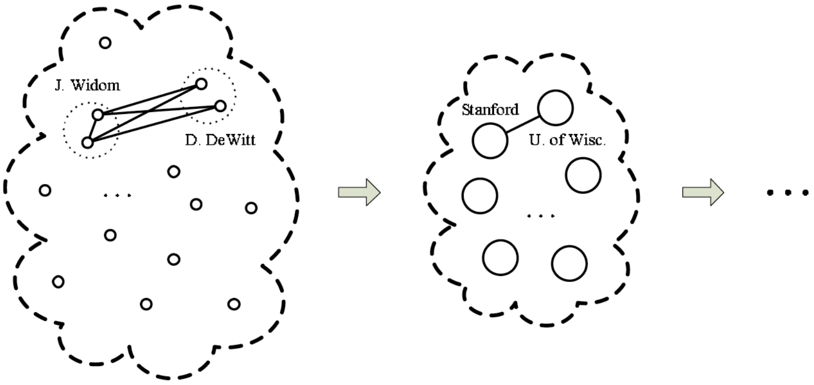


Figure 3.5 – Example of Topological Aggregate Graph (CHEN et al., 2008)

Classic OLAP operations in an t-aggregate graph can be interpreted as follows:

Roll-up Merge topological elements (nodes or edges) and replace them by corresponding higher-level elements

Drill-down Split merged elements into lower-level elements

Slice / dice Select a subgraph of a snapshot based on topological dimensions

The Topological OLAP is further explained in (QU et al., 2011). This work takes into consideration two properties (T-Distributiveness and T-Monotonicity) used to classify how different measures can be performed in an OLAP Graph. A measure function is considered T-Distributive if the result of the function applied to high-level nodes from the graph can be obtained by the computation of pre-computed results of the same function applied to lower-level nodes from the same graph. On the other hand, a measure is considered T-Monotone, if the data search space can be pruned given a user-defined threshold, by dropping node pairs with measures that don't satisfy the threshold. The paper shows experiments using common constraint function, proven to be T-Distributives and/or T-Monotones, such as SUM, MIN, MAX, Density, Degree Centrality, Closeness Centrality, among others.

3.3 HMGraph

An Heterogeneous and Multidimensional Graph OLAP framework (HMGraph OLAP) is proposed in (YIN; WU; ZENG, 2012). This framework uses a graph model similar to Graph OLAP (CHEN et al., 2008), but it adds the concept of Entity Dimensions due to the heterogeneity of the input graphs (Graph OLAP framework only handles homogeneous graphs). Figure 3.6 shows an example of a heterogeneous multidimensional network, highlighting the entity attributes of the graph.

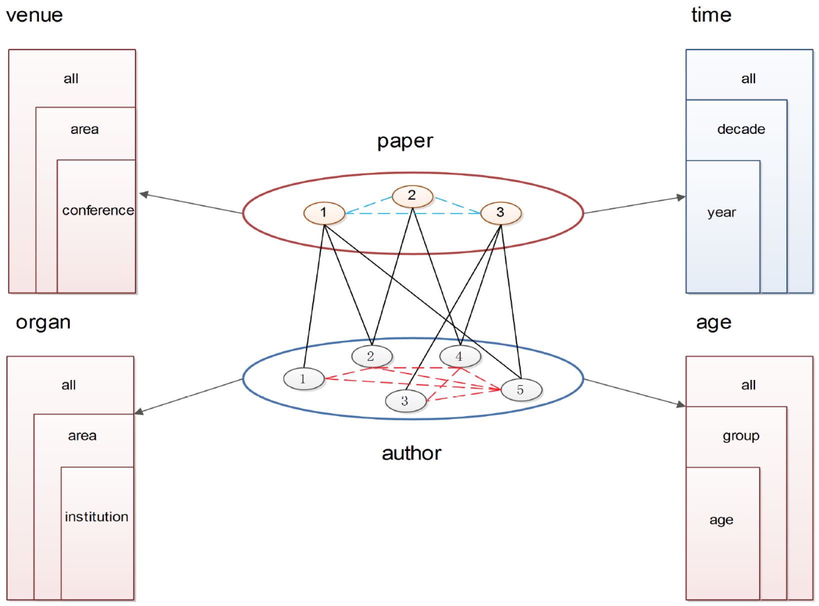


Figure 3.6 – Example of a heterogeneous multidimensional network (YIN; WU; ZENG, 2012)

Entity attributes are the attributes that describe the characteristics of an entity. In the graph illustrated in Figure 3.6, organ and age are entity attributes of author entity. Entity Dimension is related to the types of nodes in the graph.

Like Graph OLAP, HMGraph can perform I-OLAP and T-OLAP operations, but it can also perform rotate and stretch operations. The rotate operation is done by changing nodes into edges and edges into nodes, as shown in Figure 3.7. The stretch operation is done by changing edges into entities and adding edges between the recently created entity and the nodes previously connected to the transformed edge, as shown in Figure 3.8.

Even though the work presented in (YIN; WU; ZENG, 2012) draws attention to the importance of heterogeneous networks in real world application, the paper does not provide further implementation detail on how the framework can be used with real data.

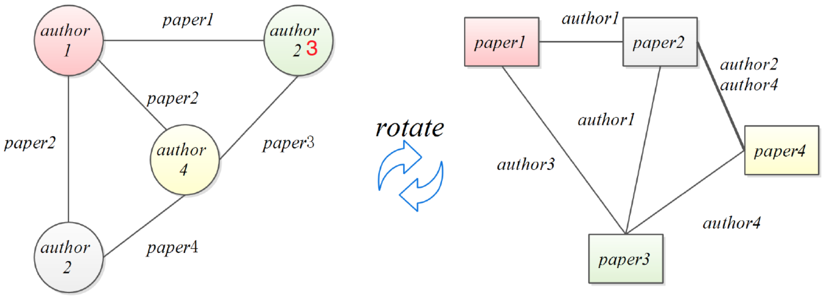


Figure 3.7 – Example of rotate operation (YIN; WU; ZENG, 2012)

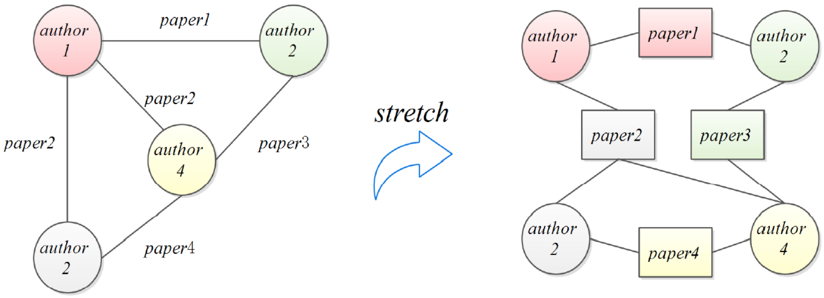


Figure 3.8 – Example of stretch operation (YIN; WU; ZENG, 2012)

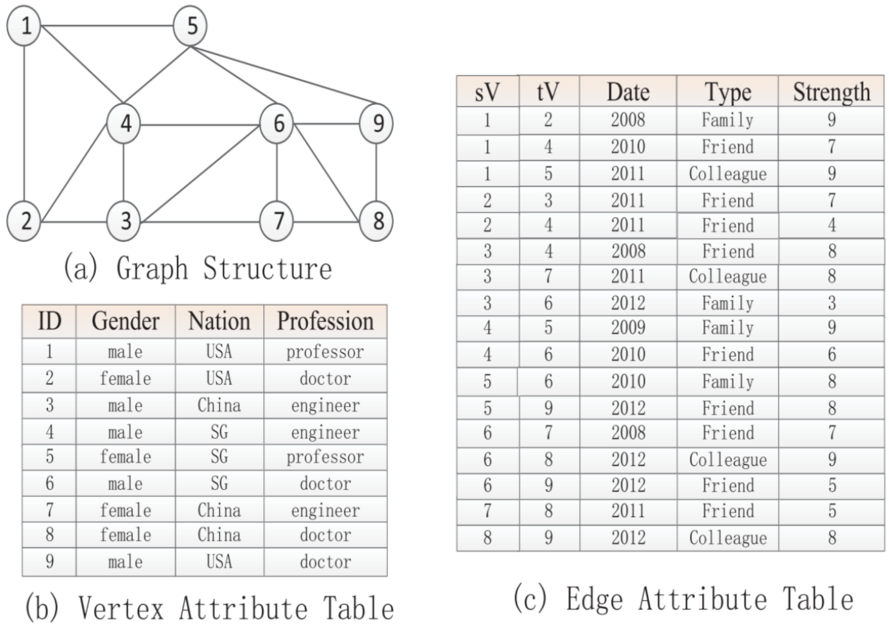
3.4 Pagrol: PArallel GRaph OLap over Large-scale Attributed Graphs

The work presented in (WANG et al., 2014) proposes a parallel Graph OLAP system, adopting the Hyper Graph Cube model that extends attributed graphs to support decision making services. The model proposed in this paper is similar to the Graph Cube described in (ZHAO et al., 2011), with the main difference being the presence of attributes also in the graph edges. In this scenario, there are two types of dimensions: vertex dimensions and edge dimensions. Figure 3.9 shows an example of an attributed graph.

Given an attributed graph with n vertex dimensions and m edge dimensions, the Hyper Graph Cube will contain 2^{n+m} aggregate graph obtained as described by the work of (ZHAO et al., 2011). This Hyper Graph Cube can be seen as the cartesian product between all the vertex-aggregate networks (when one or more vertex dimensions are aggregated) and the edge-aggregate networks (when one or more edge dimensions are aggregated). This cube arrangement can support the following query categories:

Category 1 Queries answered by information stored either in a vertex or in a edge attributes. For example:“How many relationships appeared in 2012?” or “What is the percentage of users in each different professions in this network?”

Category 2 Queries answered by integrating the knowledge stored in both vertex and edges attributes. For example: “What is the trend of the number of relations appearing



(a) Graph Structure

ID	Gender	Nation	Profession
1	male	USA	professor
2	female	USA	doctor
3	male	China	engineer
4	male	SG	engineer
5	female	SG	professor
6	male	SG	doctor
7	female	China	engineer
8	female	China	doctor
9	male	USA	doctor

(b) Vertex Attribute Table

sV	tV	Date	Type	Strength
1	2	2008	Family	9
1	4	2010	Friend	7
1	5	2011	Colleague	9
2	3	2011	Friend	7
2	4	2011	Friend	4
3	4	2008	Friend	8
3	7	2011	Colleague	8
3	6	2012	Family	3
4	5	2009	Family	9
4	6	2010	Friend	6
5	6	2010	Family	8
5	9	2012	Friend	8
6	7	2008	Friend	7
6	8	2012	Colleague	9
6	9	2012	Friend	5
7	8	2011	Friend	5
8	9	2012	Colleague	8

(c) Edge Attribute Table

Figure 3.9 – Example of attributed graph (WANG et al., 2014)

between USA and SG (Singapore) in the last 3 years?"

Category 3 Queries answered by an aggregate graph, that provides a summarised view of the data along some dimensions. For example: "What is the graph structure as grouped by users' gender as well as relationship type?"

The Hyper Graph Cube also supports roll-up and drill-down operations, along both vertex and edge dimensions. For instance, if we have an aggregate graph along Location dimension according to City value, we can roll up to obtain an aggregate graph according to State value.

The materialisation for the Hyper Graph Cube is done using Map-Reduce (MR) jobs. Since vertex and edges are stored in two different tables in the distributed file system (DFS), the materialisation is done in two steps: first the two tables are joined into one flat table containing all the dimensions and the second step performs the cube computation. This process is optimised using self-contained joins and batching techniques.

3.5 GRAD Graph Cubes

One of the most recent works in this area is presented in (GHRAB et al., 2015). This paper proposes a new technique for extracting multidimensional concepts and building OLAP cubes from heterogeneous property graphs. The authors propose a new classification of graph measures based on the aggregation type and computation algorithm:

Content-based measure Calculated based on graph's nodes and edges attributes. They are similar to traditional OLAP measures.

Graph-specific measure Obtained by applying graph algorithms. They capture topological properties of the graph.

Graph as measure Different aggregation levels of a graph can be considered measures.

Given a property graph with two distinct classes of nodes, the authors explore candidate dimensions, measures and cubes that can be obtained from the graph. The example used throughout the paper is a movie graph: it has movie nodes linked to nodes representing the actors that acted in the movie, as shown in Figure 3.10. The dimensions obtained by a subset of nodes and edges attributes are called inter-class dimensions.

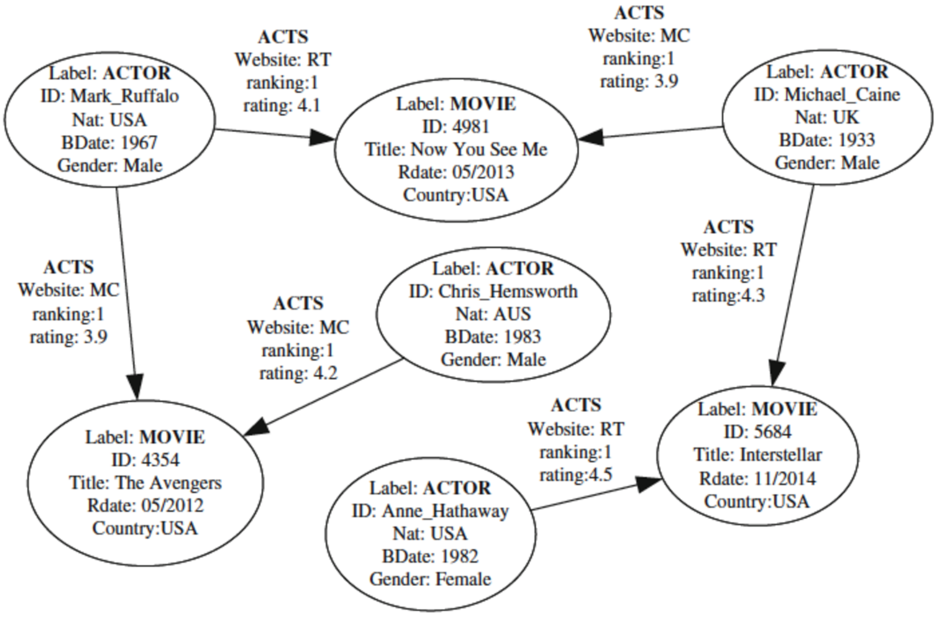


Figure 3.10 – Original movie graph (GHRAB et al., 2015)

Once the dimensions are selected, a graph lattice is defined by all possible OLAP aggregations obtained by aggregating the intra-class dimensions. The inter-class measures fall back in one of the aforementioned categories (content-based, graph-specific or graph as measure).

Figure 3.11 shows the aggregate graph obtained by grouping movies by their release date and actors by their birth date and gender. The graph shows the average ranking and rating of the ACTS relationship between grouped actors and movies.

This paper also proposes a technique for building OLAP cubes extracted from a graph modelled according to the analysis-oriented graph database model GRAD. This model provides advanced graph structures, integrity constraints and graph algebra. According to the authors, traditional property graphs only support OLAP analysis of inter-class

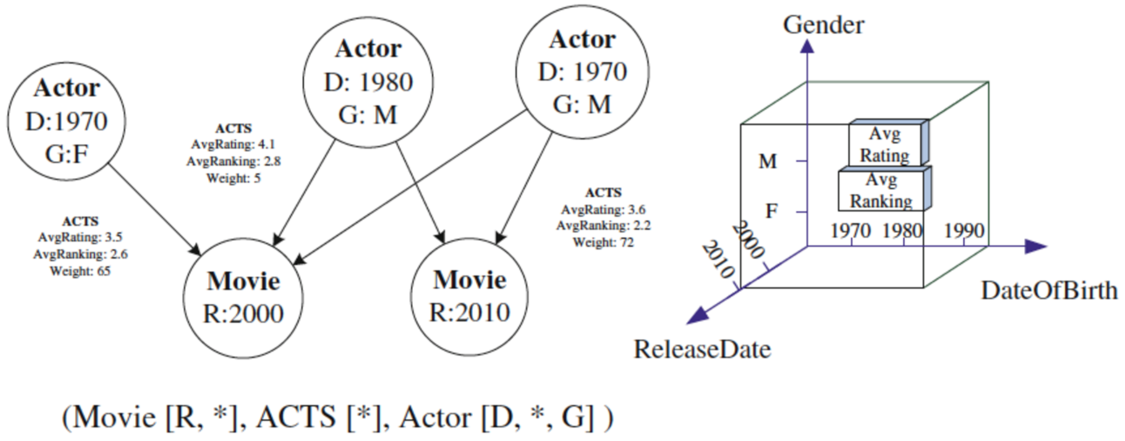


Figure 3.11 – Aggregate Graph for inter-class dimensions (GHRAB et al., 2015)

dimensions, while additional capabilities brought by GRAD allows the analysis of information stored within each node.

The GRAD model consider heterogeneous, attributed, labelled graphs and supports complex type attributes on the nodes. This model introduces special analytical structures called hypernodes, that represent real world entities grouped by classes. Each hypernode is a subgraph formed by an entity node - which contains the label and the identifier attributes - attributes nodes - linked to the entity node and represent the non-identifier attribute - and literal nodes - which stores the effective value of its attribute node. Figure 3.12 shows an example of a movie graph modelled with GRAD.

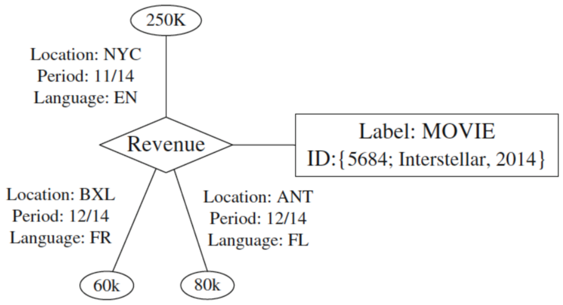


Figure 3.12 – Movie graph on GRAD model (GHRAB et al., 2015)

Based on this model, the paper defines Intra-class Dimensions as a subset of attributes nodes. The Intra-class Measures are identified by the attribute node label and are calculated in a similar way as the measures in property graph model. Figure 3.13 shows the result of applying aggregation function to the original GRAD graph in order to calculate the revenue measure, aggregating the Location according to the Country Name (CN) attribute.

This framework's implementation used Neo4J for the graph storage and HDFS

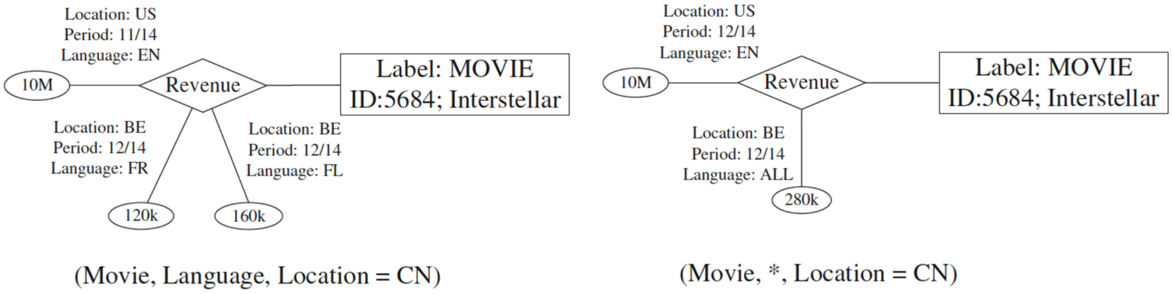


Figure 3.13 – Aggregate Graph for intra-class dimensionl (GHRAB et al., 2015)

(Hadoop Distributed File System) for distributed processing. The architecture is also composed by a middleware layer that is responsible for computing the aggregate graph and measures, using GraphX[1] library to calculate graph-specific measures.

3.6 Framework Comparison

The first works done in OLAP analysis on graph focused on homogeneous graph datasets and defined ground concept of this area, such as aggregate graphs and graph lattice. Several operators were proposed and, in general, three types of measures were taken into consideration:

Informational / Content-based / Attribute-based measure Similar to traditional OLAP measures, this information is obtained by applying an aggregate function on the node's attributes.

Topological / Aggregate Graph measure This type of measure gives information about the topology of the graph and is obtained by applying aggregate function on nodes and edges, generating a graph as a measure.

Graph-based / specific measure This type of measure is based on graph analysis theory and can be represented by a number or a subgraph.

One relevant characteristic of the works presented so far is the little explanation given on how the framework was indeed implemented, which made their understanding rather difficult. The Table 1 shows a comparison between all the frameworks presented in this chapter, regarding the type of graph, dimensions and operations supported by each one.

The work proposed here will be focused in heterogeneous graph datasets and will support the three types of measures described previously, in a similar way as presented by

[1] https://spark.apache.org/graphx/

Table 1 – Comparison of studied frameworks

Framework	Graph	Dimensions	Operations
Graph Cube	Homogeneous	Vertex Attributes	Cuboid and Crossboid Query
Graph OLAP	Homogeneous	Informational and Topological	I-OLAP and T-OLAP Operations
HMGraph	Heterogeneous	Informational, Topological and Entity	I-OLAP, T-OLAP, Rotate and Stretch Operations
Pagrol	Homogeneous	Vertex and Edge Attributes	3 Query Category and Roll-up/Drill-down Operations
GRAD Graph Cubes	Heterogeneous	Inter-class and Intra-class	-

the work on GRAD Graph Cubes. In addition to that, this work will also explore OLAP operations on graph cubes and will perform further experiments comparing graph OLAP with traditional relational OLAP.

3.7 Final Considerations

In this chapter, we presented the main research works published related to OLAP system using Graph Databases. The majority of the works available only supported homogeneous graph, but they introduced important concepts of the area, such as graph cubes and aggregate graph. The implementations that actually gave support to heterogeneous graph, proposed different graph models in order to answer analytical queries. In the next chapter, we will specify a simple OLAP system using Graph Database without the need to define a new graph model.

4 OLAP Analysis on Graph Database

In this chapter we will propose a system capable of performing OLAP analysis and that supports heterogeneous graphs, without the need to define a new graph data model. Initially, we will contextualise the proposed system and introduce a running example that will be used to better explain the system operation. Then, the system?s architecture is presented and its main components are further detailed in the following sections.

4.1 Contextualisation

In Chapter 3, we investigated some of the main works in the area of Graph OLAP. Most of them only give support to homogeneous graph (ZHAO et al., 2011)(CHEN et al., 2008)(WANG et al., 2014), while real world graph-like data contains different types of nodes and edges.The frameworks HMGraph (YIN; WU; ZENG, 2012) and GRAD Graph Cube (GHRAB et al., 2015) support heterogeneous graph, but they propose a new multidimensional model in order to perform OLAP analysis.

The objective of the system proposed here is to support OLAP analysis on graph databases without the need to re-model operational data. This will be done by adding a layer of pre-processed aggregate graph and an analytical query processor module on top of the operational graph database.

4.2 Running Example

Consider the DBLP dataset as the running example that will be used throughout this chapter to help explaining the main concepts of the system proposed. The DBLP[1] is an online computer science bibliography that, up until May 2016, indexes more than 3.3 million publications by more than 1.7 million authors. For this example, we will consider that the data was extracted and modelled according to the schema shown in Figure 4.1 and described as follows:

- Each publication becomes a node with label *Publication* and with the attributes *title*, *year* and *journal.*
- Each author becomes a node with label *Author* and with the attribute *name*
- Edges labeled *PUBLISHED* connect Author nodes to Publication nodes, representing the relationship between an author and their published work.

[1] http://dblp.uni-trier.de/faq/What+is+dblp.html

- Edges labeled *COAUTHOR* connect Author nodes to other Author nodes, representing the relationship between authors that have contributed to the same published work.

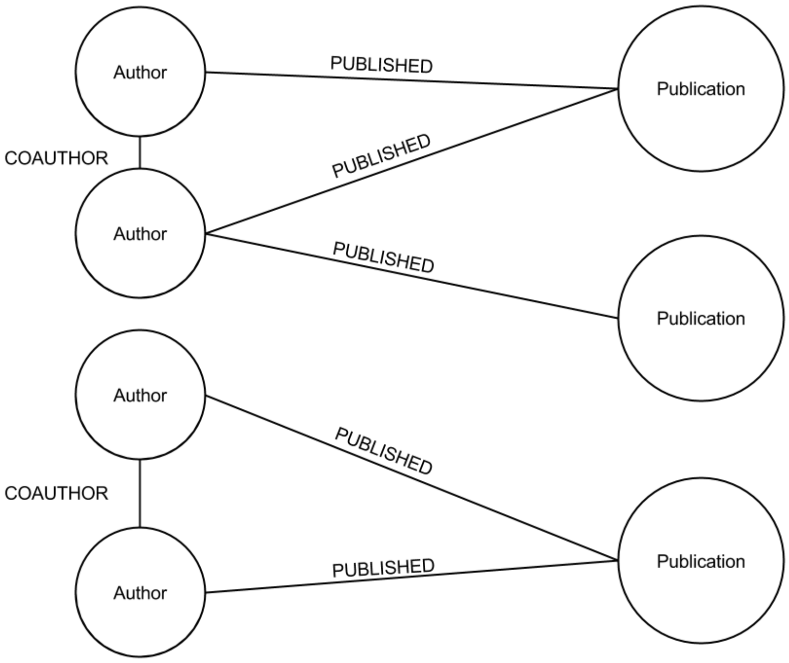
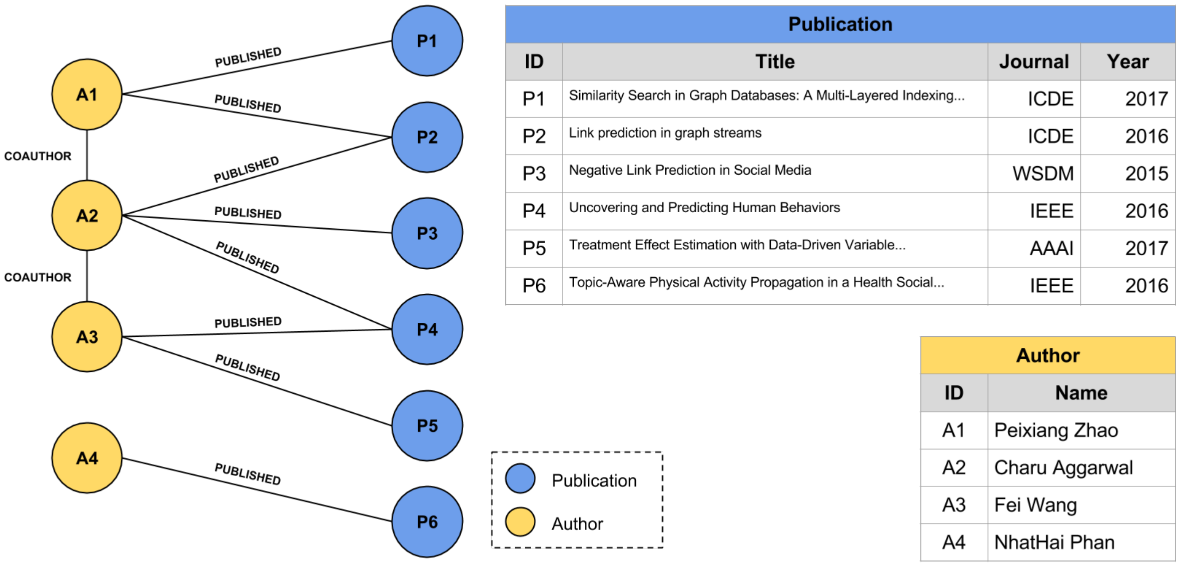


Figure 4.1 – Schema representation of the DBLP graph data

Figure 4.2 shows a subset of the original DBLP dataset, modelled according to the schema representation depicted in Figure 4.1. The following graph will be used as our running example throughout this chapter.



Publication			
ID	Title	Journal	Year
P1	Similarity Search in Graph Databases: A Multi-Layered Indexing...	ICDE	2017
P2	Link prediction in graph streams	ICDE	2016
P3	Negative Link Prediction in Social Media	WSDM	2015
P4	Uncovering and Predicting Human Behaviors	IEEE	2016
P5	Treatment Effect Estimation with Data-Driven Variable...	AAAI	2017
P6	Topic-Aware Physical Activity Propagation in a Health Social...	IEEE	2016

Author	
ID	Name
A1	Peixiang Zhao
A2	Charu Aggarwal
A3	Fei Wang
A4	NhatHai Phan

Figure 4.2 – Running Example with subset of DBLP dataset

4.3 Dimensions and Measures

As discussed in Chapter 2, an OLAP system is a tool that facilitates multidimensional analysis of the data. In order to perform such kind of analysis, it is necessary to define the dimensions and measures that will be considered during the multidimensional analysis:

Dimension Given a labeled property graph $G = (V, E, L_V, L_E, A_V, A_E)$, as presented in Chapter 2, a dimension is given by $d = (a, l)$, where $a \in A_V$ and $l \in L_V$. In our running example, we can define $d_1 = (journal, Publication)$ and $d_2 = (year, Publication)$, i.e. the *Publication* attributes *journal* and *year* are dimensions d_1 and d_2 in our OLAP system, respectively.

Measure Given a labeled property graph $G = (V, E, L_V, L_E, A_V, A_E)$, a measure is a calculation computed over the graph G using a function F, that can return the type of measures defined by (GHRAB et al., 2015):

Content-based measure For this kind of measure is calculated based on the nodes and edges attributes. In our running example, the total number of authors that published a work in 2007 is a content-based measure that is calculated using the function COUNT, which will count the number of *Author* nodes that have a relationship *PUBLISHED* to *Publication* nodes that have attribute *year* equals to 2007.

Graph-specific measure This type of measure is calculated by applying a network analysis algorithm over the graph. In our running example, the list of authors that most contributed with other authors is a graph-specific measure that is computed by applying the degree centrality measure to the *Author* nodes of the graph.

Graph as measure This kind of measure is given by different aggregation levels of a graph. In our running example, the network of authors and publications aggregated according to the journal the work was published in is a graph that represents a measure.

4.4 Architecture

The Graph OLAP system proposed in this work attempts to provide an efficient way to answer analytical queries without having to propose a new graph data model, which would imply changing the original data source model. The Figure 4.3 depicts the architecture of the system, illustrating its main components: Graph Aggregators, Aggregated Graphs and Analytical Query Processor.

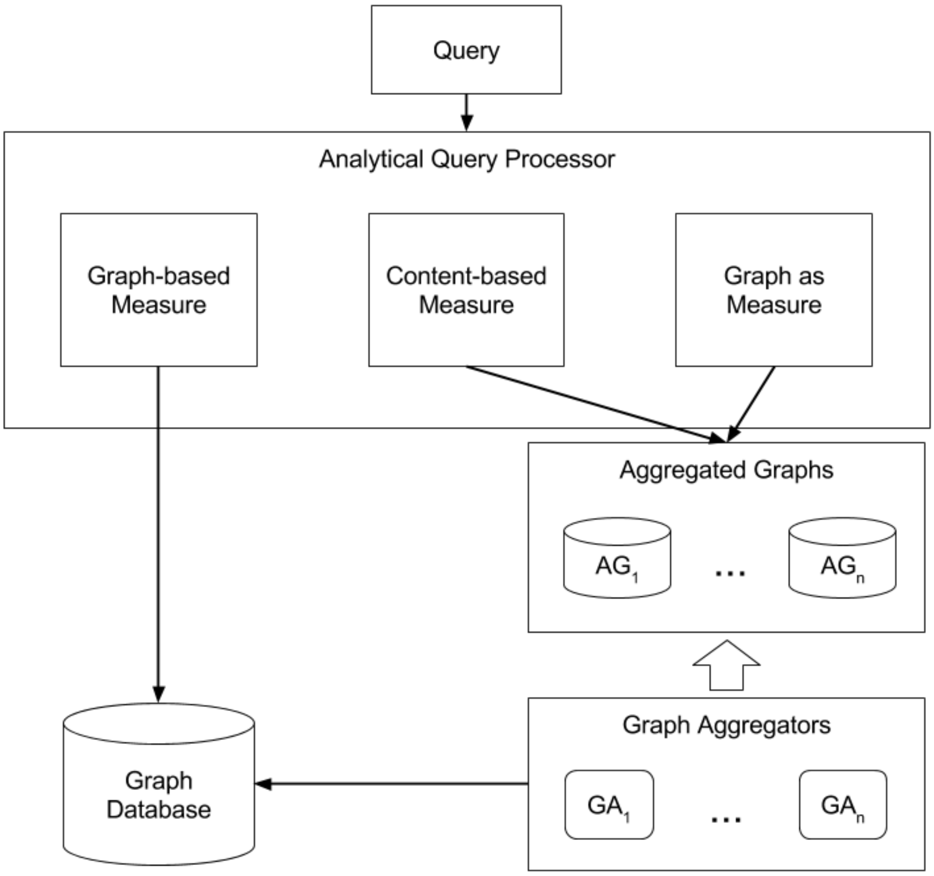


Figure 4.3 – OLAP Analysis over Graph Databases Architecture

The system?s input is an analytical query submitted by the user, that will be processed by the Analytical Query Processor (AQP). According to the type of measure expected by the user, the AQP will determine which specific processor will handle the query: graph-based, content-based or graph measures processors. If the user asks for a graph-based measure, the AQP will consume the original data stored in the graph database to calculate the measure. On the other hand, if the user requires a content-based or a graph as measure, the AQP will calculate the measure based on the data from Aggregate Graphs, which are also stored in graph databases.

The Aggregate Graphs are generated by the Graph Aggregators (GAs), which are defined during the design process of the system by the project designer. The project designer is responsible to define what are the dimensions considered in the OLAP system and, therefore, create the GAs that will generate all possible aggregate graphs for the dimensions. More details on Aggregate Graphs and Graph Aggregators are given in the following sections.

The data source considered for this system is a Graph Database that follows the labeled property graph model and supports heterogeneous graphs. This means that nodes can have one or more labels indicating different types of entities. Nodes and relationships can have properties. We assume that the data stored in the GDB is integrated, i.e. the data in the repository is consistent, well-formatted and normalised.

4.5 Aggregate Graph

Given a graph $G = (V, E, L_V, L_E, A_V, A_E)$ and a set of dimensions $D = \{d_1, \ldots, d_n\}$, where $d_i \in A_V \cup A_E$, an aggregate graph is generated by applying an aggregate function F to one or more dimensions. The result is a new graph $G_A = (V_A, E_A, L_{VA}, L_{EA}, A_{VA}, A_{EA})$, where:

- $VA = \{v_1^A, \ldots, v_n^A\}$ is a set of aggregate vertices, where each vertex v_i^A either (i) corresponds to the result of applying the function F to a set of vertices $V' \subseteq V$ that is associated with an attribute vector $[a_1, \ldots, a_k]$ containing one or more dimensions in D or (ii) corresponds to a vertex in V.
- $E_A \subseteq V_A \times V_A$ is a set of aggregate edges, where each edge e_i^A either (i) corresponds to the result of collapsing a set of edges $E' \subseteq E$ that connects one or more vertices in V that were aggregated in V_A or (ii) corresponds to an edge in E.
- L_{VA} is a set of aggregate vertex labels and L_{EA} is a set of aggregate edge labels
- $A_{VA} = \{a_1, \ldots, a_n\}$ is the set of attributes for the aggregate vertices, where $a_i = (k_i, m_i)$ is a key-value pair. Each aggregate vertex $v_i \in V_A$ is associated with an attribute vector. $A_{EA} = \{b_1, \ldots, b_n\}$ is the set of aggregate edge attributes defined in the same way as aggregate vertices attributes.

Consider the graph depicted in Figure 4.2 of our running example. Figure 4.4 shows the aggregate graph G_A obtained by applying the aggregate function COUNT to the dimension set $D = \{d\}$, where $d = (year, Publication)$. Notice that the resulting aggregate graph ends up with the same *Author* nodes as the original graph, since these vertices do not have attributes contained in the dimension set D. The *Publication* nodes were aggregated according to their *year* attribute, resulting in three nodes representing the works published in 2017, 2016 and 2015. The aggregate vertices also store the measure calculated using the function COUNT. The edges with the label PUBLISHED were collapsed, representing the connection between each author and the group of works published in a specific year. The collapsed edges also store the measure obtained by the use of COUNT function.

4.6 Graph Aggregators

The Graph Aggregators (GAs) are modules responsible for generating the aggregate graph that will be used to answer the analytical query submitted by the user. During the

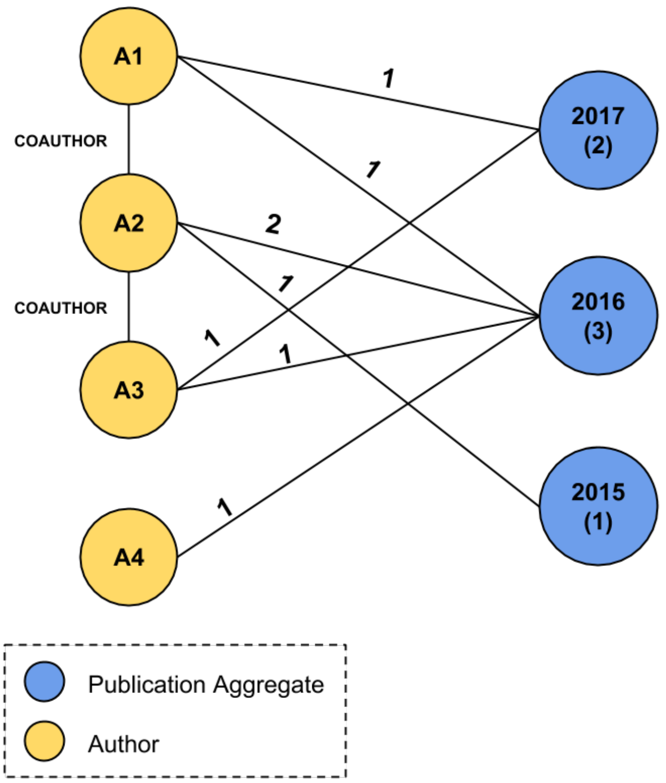


Figure 4.4 – Aggregate Graph obtained from running example graph

design process for the system, the Project Designer is responsible for building the GAs based on the dimensions and measures the system should be able to analyse. Each GA will receive as input the original graph G stored in the Graph Database, the set of dimensions to be aggregated D, the aggregate function F and should provide as output an aggregate graph as defined in the previous section. Algorithm 1 describes the process performed by a GA in order to generate an aggregate graph.

The GA algorithm starts by creating an empty set of aggregate vertices and edges that will compose the final aggregate graph that will be returned. Then, the GA will iterate over all the vertices (nodes) of the original graph G, checking for each node if it has the same label as the dimension being aggregated. If the node has the same label, the algorithm checks if already exists an aggregate node for the dimension value of the node. If the aggregate node exists, we collapse the value of the node to the value of the aggregate node using the function F and updating the aggregate node value. We also aggregate the edges that are connected to the node accordingly. If the aggregate node does not exist, we add a new aggregate node with the initial value equals to node?s dimension value and aggregate its edges accordingly. If the node does not have the same label as the dimension being aggregated, we only add the node as it is to the aggregate vertices set. Finally, we setup the aggregate graph and return it.

Once the aggregate graph is generated, it will be stored in a graph database and it

Algorithm 1 Graph Aggregator Process

```
function $generateAggGraph(G, F, D)$
    $aggVertices \leftarrow []$
    $aggEdges \leftarrow []$
    for $node$ in $G.vertices$ do
        if $node.label$ in $D$ then
            $dimValue \leftarrow getDimValue(node, D)$
            if $dimValue$ in $aggVertices$ then
                $aggregateVertices(dimValue, node, F)$
                $aggregateEdges(dimValue, node, F)$
            else
                $aggVertices.add(dimValue, node)$
                $aggregateEdges(dimValue, node, F)$
            end if
        else
            $aggVertices.add(node)$
        end if
    end for
    $aggGraph \leftarrow (aggVertices, aggEdges)$
    return $aggGraph$
end function
```

will be accessed by the Analytical Query Processor.

4.7 Analytical Query Processor

The Analytical Query Processor (AQP) is responsible for processing the query submitted by the user. This component contains three modules that will process the query differently based on the type of measure requested by the user:

Graph-based measure To calculate this measure, the AQP consumes the data from the original graph database and uses any existing library to perform network analysis on the data.

In our running example, we could submit a query asking for the *Author* node with highest centrality degree. The AQP calculates the centrality degree for each node in the original graph using an external library. Then, it should return the node with id **A2**, since it is the Author node with the highest number of connections with other nodes.

Content-based measure To calculate this measure, the AQP consumes the data from an aggregate graph and uses the aggregate function to give the resulting measure, which can be a single value, a list or a table of values. This module's response is similar to the response given by traditional OLAP systems.

From our running example, we can ask for the count of publications by year. In this case AQP consumes the data from the aggregate graph illustrated in Figure 4.4 and it would list the nodes with *Publication Aggregate* label, which already contains the count measure as attribute.

Graph as measure For this type of measure, the AQP also consumes data from an aggregate graph, but in this case, the measure is the aggregated graph itself. Therefore, this module does not need to perform other calculations.

For instance, the aggregate graph shown in Figure 4.4 can be considered a measure if the user requests a topological view of the original data grouping the publications by year

4.8 Final Considerations

Bibliography

ANGLES, R. A comparison of current graph database models. *Proceedings - 2012 IEEE 28th International Conference on Data Engineering Workshops, ICDEW 2012*, n. February, p. 171–177, 2012. Cited in page 11.

ANGLES, R.; GUTIERREZ, C. Survey of graph database models. *ACM Computing Surveys*, v. 40, n. 1, p. 1–39, 2008. ISSN 03600300. Disponível em: <http://portal.acm.org/citation.cfm?doid=1322432.1322433>. Cited in page 11.

CHEN, C. et al. Graph OLAP: Towards online analytical processing on graphs. *Proceedings - IEEE International Conference on Data Mining, ICDM*, p. 103–112, 2008. ISSN 15504786. Cited 5 times in pages 17, 18, 19, 20, and 27.

DENIS, B.; GHRAB, A.; SKHIRI, S. A distributed approach for graph-oriented multidimensional analysis. *Proceedings - 2013 IEEE International Conference on Big Data, Big Data 2013*, p. 9–16, 2013. Cited in page 17.

FREEMAN, L. C. Centrality in social networks conceptual clarification. *Social Networks*, v. 1, n. 3, p. 215–239, 1978. ISSN 03788733. Cited in page 9.

GHRAB, A. et al. A framework for building olap cubes on graphs. In: SPRINGER. *East European Conference on Advances in Databases and Information Systems*. [S.l.], 2015. p. 92–105. Cited 6 times in pages 22, 23, 24, 25, 27, and 29.

INMON, W. *Building the data warehouse*. [S.l.: s.n.], 2005. 428 p. ISBN 0471081302. Cited 3 times in pages 3, 5, and 7.

JOUILI, S.; VANSTEENBERGHE, V. An empirical comparison of graph databases. In: IEEE. *Social Computing (SocialCom), 2013 International Conference on*. [S.l.], 2013. p. 708–715. Cited in page 12.

KHAN, K. U. et al. OLAP on Structurally Significant Data in Graphs. p. 2–5, 2014. Cited in page 17.

KIMBALL, R.; ROSS, M. *The Data Warehouse Toolkit: the complete guide to dimensional modeling*. [S.l.]: John Wiley & Sons, 2011. Cited 2 times in pages 3 and 4.

MCCOLL, R. C. et al. A performance evaluation of open source graph databases. In: ACM. *Proceedings of the first workshop on Parallel programming for analytics applications*. [S.l.], 2014. p. 11–18. Cited in page 12.

MILLER, J. Graph Database Applications and Concepts with Neo4J. *Proceedings of the Southern Association for Information Systems Conference*, p. 141–147, 2013. Disponível em: <http://sais.aisnet.org/2013/MillerJ.pdf>. Cited 3 times in pages 7, 8, and 10.

OLAP COUNCIL. *OLAP AND OLAP Server Definitions*. [S.l.], 1997. Disponível em: <http://www.olapcouncil.org/research/glossaryly.htm>. Cited 2 times in pages 4 and 5.

QU, Q. et al. Efficient Topological OLAP on Information Networks.pdf. In: *International Conference on Database Systems for Advanced Applications*. [S.l.: s.n.], 2011. p. 1–15. Cited in page 19.

ROBINSON, I.; WEBBER, J.; EIFREM, E. *Graph Databases*. 2nd editio. ed. [S.l.]: O'Reilly Media, Inc., 2015. 238 p. ISBN 9781491930892. Cited in page 10.

TOBERGTE, D. R.; CURTIS, S. *Introduction to GraphTheory*. [S.l.: s.n.], 2013. v. 53. 1689–1699 p. ISSN 1098-6596. ISBN 9788578110796. Cited 3 times in pages 7, 8, and 9.

VASSILIADIS, P. Modelling multidimensional database, cube and cube operations. *Scientific and Statistical Database Management, 1998. Proceedings. Tenth International Conference on*, n. May 2000, 1998. Cited 3 times in pages 4, 5, and 6.

VASSILIADIS, P.; SELLIS, T. A survey of logical models for OLAP databases. *ACM SIGMOD Record*, v. 28, n. 4, p. 64–69, 1999. ISSN 01635808. Disponível em: <http://portal.acm.org/citation.cfm?doid=344816.344869>. Cited 2 times in pages 5 and 6.

WANG, Z. et al. Pagrol: Parallel graph olap over large-scale attributed graphs. *Proceedings - International Conference on Data Engineering*, v. 1, p. 496–507, 2014. ISSN 10844627. Cited 3 times in pages 21, 22, and 27.

YIN, M.; WU, B.; ZENG, Z. Hmgraph olap: a novel framework for multi-dimensional heterogeneous network analysis. *Proceedings of the fifteenth international workshop on Data warehousing and OLAP*, p. 137–144, 2012. Cited 3 times in pages 20, 21, and 27.

ZHAO, P. et al. Graph Cube: On Warehousing and OLAP Multidimensional Networks. *Sigmod '11*, p. 853–864, 2011. ISSN 07308078. Disponível em: <http://www.cs.uiuc.edu/{~}hanj/pdf/sigmod11{_}pzha>. Cited 5 times in pages 15, 16, 17, 21, and 27.