

Graph-based Relational Data Visualization

Daniel Mário de Lima*, José Fernando Rodrigues Jr.[†] and Agma Juci Machado Traina[‡]

Instituto de Ciências Matemáticas e de Computação

Universidade de São Paulo

São Carlos, Brazil

Email: {,†,‡}danielm, †junio, ‡agma}@icmc.usp.br*

Abstract—Relational databases are rigid-structured data sources characterized by complex relationships among a set of relations (tables). Making sense of such relationships is a challenging problem because users must consider multiple relations, understand their ensemble of integrity constraints, interpret dozens of attributes, and draw complex SQL queries for each desired data exploration. In this scenario, we introduce a twofold methodology; we use a hierarchical graph representation to efficiently model the database relationships and, on top of it, we designed a visualization technique for rapidly relational exploration. Our results demonstrate that the exploration of databases is deeply simplified as the user is able to visually browse the data with little or no knowledge about its structure, dismissing the need for complex SQL queries. We believe our findings will bring a novel paradigm in what concerns relational data comprehension.

Keywords—relational databases; hierarchical visualization; visual analytics;

I. INTRODUCTION

Over the last decades, a huge amount of information has been generated, making large databases common in several kinds of applications. Examples of this growth are found in industry-generated data, where information from clients, products, and transactions of multiple types are stored in relational manner. In these databases, the entities are described by attributes and refer to each other in relationships that define a strong structural cohesion. Relational Database Management Systems (RDBMS's) are the ultimate solution for such structured data; they provide intelligent storage, and powerful querying capabilities in applications that range from commerce to education. And, despite the fact that non-relational solutions have popped-up in the market – such as NoSQL databases, RDBMS's still answer for the greatest market-share [1], [2]. But, although RDBMS's are wonderfully engineered for data storage, they are not adequate for visual analytics – this is the point we tackle in this study.

In this context, storing and retrieving data is only part of the problem when the goal is to obtain useful information. Reasoning about voluminous and complex data can be very difficult, a shortcoming that demands analytical processes in order to find patterns, uncommon arrangements and relationships, and other kinds of hidden knowledge that may aid in decision support. Such processes are commonly performed in exploratory fashion according to which the analyst does

not know a priori what to look for. In these circumstances, a rich and responsive visual environment can provide preferred results, notably for structured data.

A straightforward approach to investigate structured data is to use graph representations with node-link visualizations, according to which nodes and edges correspond, respectively, to entities and relationships of the Entity-Relationship (ER) database model [3]. Within these considerations, here we experiment with a graph representation based on *hierarchical partitioning*, a technique that improves the scalability of graph-based visualizations. And so, our proposal uses the ER structure in order to generate an initial graph that is hierarchically partitioned according to the entities, attributes, and values found in the database. This hierarchically-partitioned graph, then, gives rise to a multiple-level visualization comprised of nodes, groups of nodes, edges, and summarizations over which interactive querying and aggregation take place.

In this study, given a relational database, we are interested in answering the following questions:

- How are the data (entity instances) distributed over the relations of the database?
- How are the entities on the database related to each other?
- How do the several attributes of the database influence the relationships of the entities?
- How do we quickly and intuitively browse the relational data, considering its complex structure?

We answer these questions by using hierarchical graph partitionings created from both the structure and the data found in the database to be analyzed. Over that, we define a novel visual/interactive scheme instantiated in a fully operational prototype. Our contribution makes the exploration of the relationships between data entities intuitive and computationally fast, even if considering very large databases. According to our technique, the database structure can be browsed through exploration paths in which the user can visualize entities and their relationships without explicitly defining queries. Our study relies on the SuperGraph concept and the Graph-Tree structure [4], and we derive the GMine system [5], originally designed for sole graph analysis, to the realm of visual analysis of relational data; hence, our

new systematization is named RMine system.

Coming next, Section II reviews the related literature, followed by our proposed methodology in Section III, where we will define: (a) a relational-based hierarchical partitioning for databases; (b) how this partitioning is represented in a Graph-Tree structure together with associated algorithms; and (c) the visualization environment for this data structure. In Section IV, our experiments will show the main aspects of our approach and how it simplifies the data manipulation process. Then, in Section V we finish with a brief discussion of the main achievements, outlining ideas for future improvements.

II. RELATED WORK

Although there are many studies on database visualization, these studies do not explore the structural information of the data, but rather they concentrate on table representation in cube-like schemes. We introduce a novel perspective, one that permits to explore how multiple tables relate to each other.

One of the most referenced studies that aim at visually exploring databases is Polaris [6], [7]. It follows the well-known data cube approach, which is broadly used in a number of decision support systems in businesses and organizations, mainly in the form of OLAP (on-line analytical processing) services [8]. The system provides an interface to develop and interact with *visual specifications*; a visual specification is the assignment of the attributes of a table for each of the axes of a data cube, along with the required definitions for: selection of records, data transformations, aggregations, partitionings, sortings, and visualization properties. Polaris' data-cube is organized so that each cell presents the visualization (scatterplots, or barcharts) of specific data ranges over selected attributes at different granularities. Different from our study, Polaris is not designed for the inspection of relational structure data.

In another study [9], Stolte et al. propose *Zoom Graphs*, a general multiscale visualization of hierarchically structured data. Their study defines a formal notation to design zoom-graph visualizations; the notation uses four possible patterns used to describe the core structure of the most common designs. Although this method can model complex schemata by using multiple hierarchies, it restricts the user's interaction to follow a single exploration path through a previously chosen hierarchy.

On a different line, Maniatis et al. [10] employ the Table Lens technique [11] over data cube visualizations. They adopt the *Cube Presentation Model* to split the presentation components of the logical data layer, allowing the user to explore sections of a fact table by choosing the desired values of the attributes being presented. Similarly, Techapichetvanich and Datta [12] introduce the *Hierarchical Dynamic Dimensional Visualization* (HDDV) to explore hierarchically structured data from data cubes. Their approach maps the

cube dimensions to the levels of an exploration tree whose visualization exposes the exploration path. In the tree, each level is a bar stick with dividing marks for splitting either attribute ranges or nominal labels of a given dimension. This method allows the user to visually build cube queries and quickly switch to a different hierarchical path, but it is still able to show only a single path at a time.

Mansmann and Scholl [13] improve the visualization of multiple hierarchies with the *Enhanced Decomposition Tree*, which combines the Cube Presentation Model [10] with different space-filling visualization techniques. In their scheme, the exploration tree is allowed to have sub-trees comprising different attributes and being visualized in parallel.

Wang et al. [14] present a client-server visualization system named *Zoom Tree*. Their system allows a schema-based data cube navigation similar to the schema of Mansmann and Scholl [13], with the difference that dimensions with too many values are presented according to hierarchies organized by ranges of values. Selected portions of the data are visualized in a table layout inspired by Polaris [6], and the user's navigation (zooming into the table data) is stored in the Zoom Tree, thus forming a navigation history.

ASK-GraphView et al. [15] offers a hierarchical graph structure which closely resembles the SuperGraph concept [4]. Their approach partitions the graph in antichains whose associated views can be processed in the main memory, then reducing the depth by retracting combs and reducing the fan-out by separating siblings under new subtrees. With this structure, they apply a recursive clustering pipeline to find subgraph components and create the hierarchical visualization. Differently from their approach, the proposed method employs the database structural information to partition the graph and build the hierarchy.

A recurrent concern of these studies is the scalability regarding both response time and visual cluttering, fundamental demands for interactive systems. The data cube visual interface, as used in these studies, is well-established, providing a promising environment for exploratory analysis. However, these studies are tied to quantitative analysis, demanding the analyst to know the right attributes and operations in order to compose the visualizations; such schemes provide little or no support for exploring relationships between entities.

Besides, all these previous studies are transactions-oriented, focusing on the datacube metaphor; differently, our study does not focus on transactions, but on the multiple relationships that emerge from the structure of operational databases.

III. THE PROPOSED APPROACH

A. SuperGraph overview

The method we propose is based on the concept of SuperGraphs, a formalization that abstracts a hierarchically partitioned graph. A SuperGraph is recursively comprised

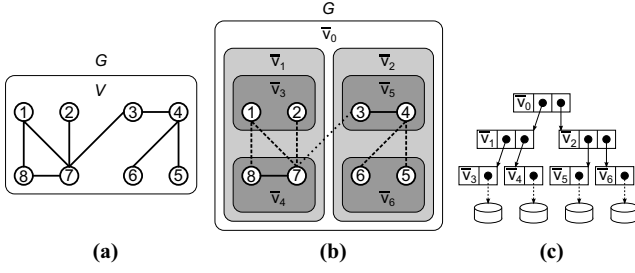


Figure 1. SuperGraph obtained from a partitioned graph.

of nodes, SuperNodes (groups of nodes), edges, and SuperEdges (groups of edges), and is defined as follows:

Definition 1: [SuperGraph] Given a finite undirected graph $G = \{V, E\}$, with neither loops nor parallel edges, a SuperGraph is defined as $\bar{G} = \{\bar{V}, \bar{E}\}$, where \bar{V} is a set of SuperNodes \bar{v} , and \bar{E} is a set of SuperEdges \bar{e} .

Definition 2: [SuperNode] A SuperNode \bar{v} is recursively defined as a set \bar{V}' of SuperNodes or graph nodes (if it is a leaf), plus a set \bar{E}' of SuperEdges \bar{e}_{ij} . As follows:

$$\begin{aligned} \bar{v} = \{\bar{V}' &= \{\bar{v}_0, \bar{v}_1, \dots, \bar{v}_{(|\bar{V}'|-1)}\}, \\ \bar{E}' &= \{\bar{e}_{ij} = \{(\bar{v}_i, \bar{v}_j) | \bar{v}_i, \bar{v}_j \in \bar{V}'\}\} \end{aligned} \quad (1)$$

Definition 3: [SuperEdges] A SuperEdge represents all the edges $(u, v) \in E$ that connect graph nodes from a SuperNode \bar{v}_i to graph nodes from SuperNode \bar{v}_j . Formally, the SuperEdge between SuperNodes \bar{v}_i and \bar{v}_j is defined as follows:

$$\text{SuperEdge}(\bar{v}_i, \bar{v}_j) = \bar{e}_{ij} = \{e = (u, v) | (u, v) \in E, u \text{ is enclosed in } \bar{v}_i \text{ and } v \text{ is enclosed in } \bar{v}_j\} \quad (2)$$

Definition 4: [Weight of a SuperEdge] The weight of a SuperEdge is the sum of the weights of its edges.

Figure 1 exemplifies the SuperGraph abstraction. Figure 1(a) shows graph G , defined as $G = \{V = \{1, 2, 3, 4, 5, 6, 7, 8\}, E = \{(1, 7), (1, 8), (2, 7), (3, 4), (3, 7), (4, 5), (4, 6), (7, 8)\}\}$.

From graph G , it is possible to conceive the hierarchical partitioning presented as SuperNode \bar{G} in Figure 1(b). This partitioning is composed of SuperNodes \bar{v}_0 through \bar{v}_6 and corresponding SuperEdges:

$$\begin{aligned} \bar{v}_0 &= \{\bar{V}' = \{\bar{v}_1, \bar{v}_2\}, \bar{E}' = \{\bar{e}_{12} = \{(3, 7)\}\}\} \\ \bar{v}_1 &= \{\bar{V}' = \{\bar{v}_3, \bar{v}_4\}, \bar{E}' = \{\bar{e}_{34} = \{(1, 7), (1, 8), (2, 7)\}\}\} \\ \bar{v}_2 &= \{\bar{V}' = \{\bar{v}_5, \bar{v}_6\}, \bar{E}' = \{\bar{e}_{56} = \{(4, 5), (4, 6)\}\}\} \\ \bar{v}_3 &= \{\bar{V}' = \{1, 2\}, \bar{E}' = \{\}\} \\ \bar{v}_4 &= \{\bar{V}' = \{7, 8\}, \bar{E}' = \{\bar{e}_{44} = \{(7, 8)\}\}\} \\ \bar{v}_5 &= \{\bar{V}' = \{3, 4\}, \bar{E}' = \{\bar{e}_{55} = \{(3, 4)\}\}\} \\ \bar{v}_6 &= \{\bar{V}' = \{5, 6\}, \bar{E}' = \{\}\} \end{aligned}$$

Figure 1(c), in turn, presents the corresponding Graph-Tree structure, which reflects the hierarchical partitioning of

the SuperGraph. In the figure, one can see that the Graph-Tree is designed so that leaf SuperNodes are selectively loaded from disk. The main feature of the Graph-Tree is its ability to dynamically determine the edges that interconnect either nodes or SuperNodes. This feature implies that:

- 1) given a node, one can determine all the edges that connect to this node without having to check all the partitions and levels of the graph hierarchy;
- 2) given any two SuperNodes, one can determine all the edges that connect these two groups of nodes.

These two features are the key to the present study as they allow for the dynamic inspection of the structural data in relational database. In order to provide these features, the Graph-Tree uses specific algorithms whose details are out of the scope of this study.

B. Relational-based hierarchical partitioning

In previous studies, the Graph-Tree data structure was used to process and to visually interact with graphs that were automatically partitioned. The problem with this application is that the number h of levels in the hierarchy is determinant in interpreting the graph hierarchical partitioning; however there are no algorithms to automatically determine these values from a given graph. To solve this problem, in this study, we define the number of hierarchy levels as the number of attributes of interest in a given database relation. As it will be explained further on, this defines a semantically-rich hierarchy organized according to the values of the attributes found in the data.

Together with this approach, we use the information given by the relationships between the different entities of the database to instantiate graph-like data. Therefore, we were able to produce hierarchically-partitioned graphs that incorporate the information of entire databases considering the semantics given by their attributes and the structure given by their relationships. This approach is totally different if compared to former database visualization techniques, which are centered on quantitative transactional values, disregarding the important information represented by the structure of the database.

First level

Following our line of thought, the first level of the hierarchy is determined by entities that define many-to-many relationships. For example, consider the simple database schema shown in Figure 2 – in this schema, entity Person has a many-to-many relationship to entity Publication. Following the SuperGraph representation, the first level of the hierarchy will have one partition (or SuperNode) for each corresponding entity – Person and Publication, in this case. These SuperNodes are added as children of the root node.

By using this initial partitioning, the correspondent database hierarchical visualization will present the database entities in its first level, as illustrated in Figure 3(a). In this

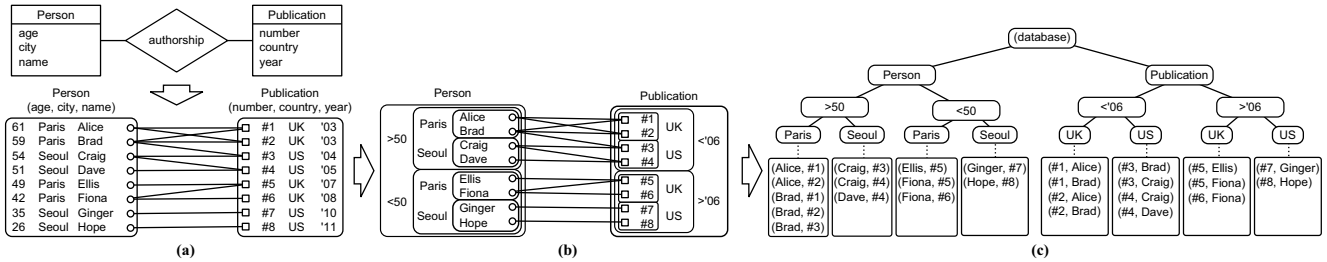


Figure 2. Super-Graph built by partitioning two entity relations linked by a many-to-many relationship.

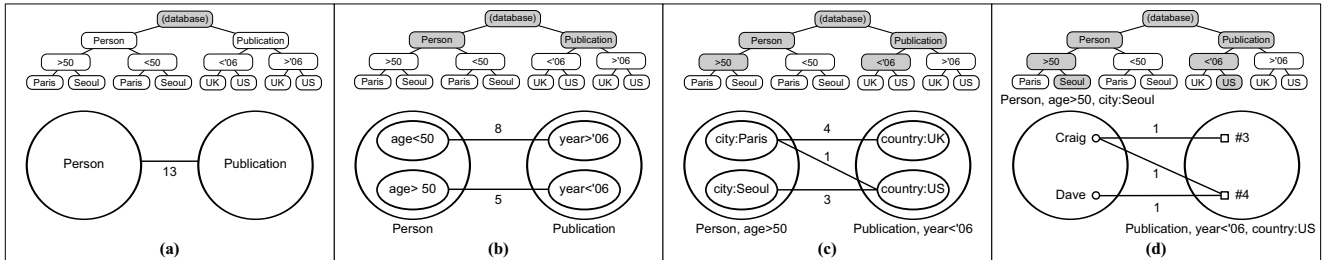


Figure 3. Hierarchical SuperGraph visualization.

first level, the edges between Person and Publication defines a SuperEdge composed by all the edges between nodes in these partitions. The first level of the visualization, hence, provides an overview of how the data is structured and how the different entities manifest in the database. In system RMine, this visualization is interactive. Besides visualizing the SuperNodes, it is possible to retrieve further details by inspecting the SuperEdges of the visualization. And so, one can double click a SuperEdge of interest and the underlying Graph-Tree will present which nodes interact with each other.

Deeper-levels

Following, the idea is to have the possibility of refining the information of each SuperNode entity of the first level. That is, each SuperNode had to be partitioned into another set of SuperNodes, a level below in the hierarchy. The problem here is how to determine the partitioning in a given level, and how many levels to use. In order to answer these two questions, our method considers the attributes of each entity as the information to guide the lower levels of the partitioning. We proceed considering two actions:

- the number and meaning of the levels are given by the attributes of each entity, one level per attribute;
- the number of partitionings in a given level is given by the values of that attribute as found in the database.

Action a) implies that the most representative attributes of a given entity must be considered. In the example, entity Person can be represented by age and city, determining two levels below the first level; and entity Publication can be represented by country and year, again two levels below the

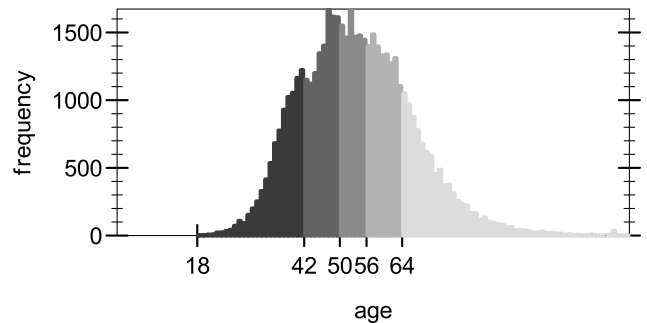


Figure 4. Percentile statistical partitioning. In this case, the 5 quintiles of entity Person as determined by attribute Age.

first level - illustrated in Figure 2. Action b) is a little more complicated; it demands to search the database and verify the distribution of values for each attribute.

There are different kinds of attributes; the main kinds are categorical, nominal, and numeric. Categorical attributes with small numbers of categories, as *gender* for example, will determine one partition for each distinct category. For a greater number of categories, it is interesting to evenly partition the data following its distribution. For nominal and numeric attributes, the number of partitions can be determined by considering the distribution of the values. Figure 2(b) shows an example with each attribute being partitioned in two groups – attribute *age* partitions the *Person* SuperNode in two SuperNodes “age > 50” and “age < 50”; and then each SuperNode is again partitioned in two SuperNodes, “city: Paris” and “city: Seoul”. This method

results in the Graph-Tree structure illustrated in Figure 2(c), which can be used in a hierarchical SuperGraph visualization as illustrated in Figure 3. According to such visualization, the user can descend in different paths and levels of the tree to inspect the SuperEdges between the various arrangements of SuperNodes.

Still for nominal and numeric attributes, one can consider a greater number of k partitions; which can be achieved with traditional statistic percentile analysis. Since we are dealing with a visual interactive ensemble, the default value of k is set around the limits of human working memory, which is 7 ± 2 elements according to Miller’s Law [16]. This restriction leads to a number of partitions per level that does not overload the user’s cognition.

As an example, Figure 4 shows the distribution of ages of a real-world database (detailed in Section IV), in which we define $k = 5$ partitions. We obtain quintiles $QU_1 = [18, 41]$ (20 % of the observations), $QU_2 = [42, 49]$ (40 % of the observations), $QU_3 = [50, 55]$ (60 % of the observations), $QU_4 = [56, 63]$ (80 % of the observations) and $QU_5 = [64, 162]$ (100 % of the observations).

Following the percentile statistical approach, we end up with partitions with nearly the same size in number of entity nodes. This is a straightforward procedure intended for automatic analysis; yet, our methodology can also benefit from user-defined parameters (such as a greater k) for specific analytical goals.

Design criteria

The design of our methodology was defined to satisfy four features:

- 1) the structural information of the database becomes represented in the graph hierarchy, an information formerly ignored;
- 2) the semantic information given by the attributes and values found in the database are kept for further analysis;
- 3) it becomes possible to deal with very large databases while maintaining its semantic meaning and interpretability;
- 4) the hierarchical representation adheres to hierarchical visualization techniques, permitting to visually explore the database.

In Section IV, we demonstrate these features together with additional quantitative criteria by means of a broad set of experiments.

C. Database representation and preprocessing

In order to have the database represented as a semantic-rich hierarchical graph, as described in the preceding section, it is necessary to preprocess it based on choices of representative attributes. This step involves selecting the relevant relations (entities) and relationships, as required by the analyst, tracing its set of attributes and their respective distributions.

Once the database is preprocessed, its data is scanned and a Graph-Tree is created following a hierarchical partitioning that conforms to the data properties. In an instantiated Graph-Tree there are three kinds of information:

- the nodes at the leaves, which correspond to the tuples of the relations;
- the SuperNodes along the hierarchy up to the first level, which correspond to sets (groups) of nodes and, recursively, to sets of SuperNodes; at the first level, they correspond to the relations of the database;
- structural information that distributed along the structure permits to calculate the relationships (edges) between any pair node-node, node-SuperNode, or SuperNode-SuperNode.

From the storage point of view, a small portion of the Graph-Tree is kept in the main memory, while a bigger portion is kept on the disk. This organization permits to have the data loaded on demand, saving on processing and memory resources.

D. Hierarchical Graph Visualization

Our method generates semantic-rich SuperGraph for a given database, having as a product, a Graph-Tree that organizes and manages data on demand. The conception of this product, as it has been described, adheres to hierarchical visualization techniques of any kind (as those listed by Schulz [17]), and as a proof-of-concept, we use a canonical node-link visualization as implemented in GMine. Originally, this visualization has been used to visualize graphs in general; for the database domain, we have adapted it so to bring the specificities of relational data to an interactive visual environment.

The visualization we propose is based on the following operations, which can be thought as specialized cases of drill-down and roll-up operations [18]:

- *expand SuperNode*: digs one level down the hierarchy;
- *contract SuperNode*: moves one level up in the hierarchy;
- *SuperNode-to-SuperNode connectivity*: as one expands or contracts a given SuperNode, its correspondent SuperEdges must reflect the operation, that is, subsets/supersets of SuperEdges must be calculated for every pair of SuperNodes;
- *hide/show SuperEdge*: as the number of SuperEdges varies, the user must be able to hide or show SuperEdges of interest;
- *expand SuperEdge*: under user’s demand, the set of edges that determine a given SuperEdge is presented in a separate visualization, in which details can be observed;
- *filter edges*: as a SuperEdge is expanded, the number of edges presented to the user may be way too much for visualization; for this, our scheme supports filtering

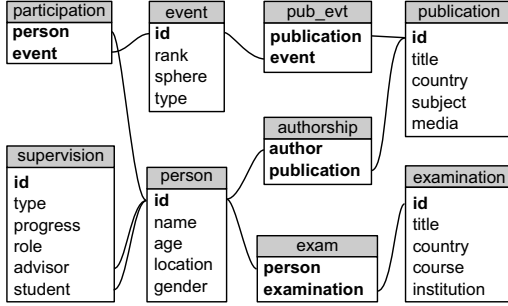


Figure 5. Tycho-USP database.

edges based on their weight, that is, the number of times two nodes have interacted in the database.

All these features are integrated in a novel environment for database visualization, as illustrated in the next section. In this environment, once the database data are loaded, the user can verify and refine the information that characterizes each entity, at the same time that he/she can track the relationships within these entities.

IV. EXPERIMENTS

To demonstrate the potential of the proposed approach, this section covers some experiments. Given a database and some exploratory tasks, we illustrate each task together with its corresponding SQL queries and processing times. The intent is to demonstrate that our method substitutes complex and costly SQL operations that, otherwise, would demand time to be written and to be processed. Our method, instead, allows for the same operations satisfying to interaction-time restrictions.

A. Database and test setup

The data used in the following experiments come from the Tycho-USP database. Tycho-USP is a general academic database from the University of São Paulo, containing data about students, professors, and their academic work. The data are collected from various information systems in the university and merged with external data from scientific agencies of the Brazilian government. It is structured in a schema with five main entities: Event (352,400 nodes), Examination (382,890 nodes), Publication (691,083 nodes), Research Supervision (26,237 nodes) and Person (50,779 nodes); with relationships Person-Examination (851,168 edges), Person-Event (247,516 edges), Person-Publication (691,083 edges), Person-Supervision (52,439 edges) and Publication-Event (26,237 edges); a total of 1,503,389 nodes and 1,868,443 edges.

As illustrated in Figure 5, the four entities of Tycho-USP were considered according to their attributes of interest as follows: Person (age, location, gender), Publication (year, country, subject), Event (rank, sphere, type), Supervision (type, progress, advisor_role) and Examination (country,

course, institution). The structure of the database was considered according to the relationships linking Person to all the other entities, and Publication to Event. The attributes were used to determine the levels of the hierarchy, and the relationships were used to determine the edges of the underlying graph.

For our experiments, we use parameter $k = 5$. The nominal attributes are divided into the k most frequent classes plus an “others” class with the remaining elements; and numeric attributes are partitioned in k percentiles with approximately the same cardinality. The first task is to build the data structure; to this end, our method receives a configuration set containing the entities and relationships of interest and builds an empty Graph-Tree. Then, the construction procedure writes the nodes and edges in the leaf SuperNodes, and fills up the upper levels with connectivity SuperEdges. This initial step creates a persistent Graph-Tree on disk, which can be loaded in the visualization system later.

All time measurements are averaged wall-clock timings, taken on a personal computer equipped with an AMD Phenom II X4 850 processor, 4 GB of DDR3 main memory, a single 500 GB SATA hard disk and Microsoft Windows 7 Professional x64 operating system.

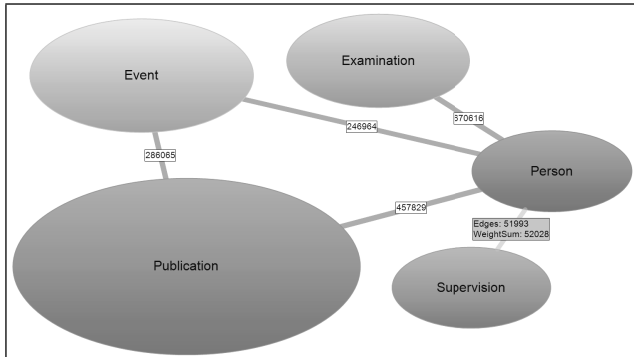
B. Visual analysis

In this section, we demonstrate how our method can be used for visually inspecting a relational database. We carry the following tasks:

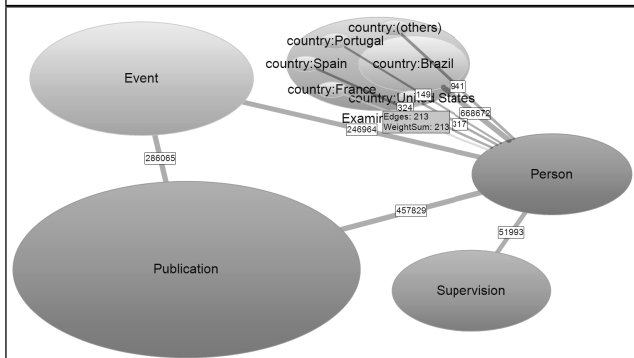
- 1) Visualize the distribution of the entities and of their relationships: how many people, publications and events are there and how are they arranged?
- 2) Visualize the relationship between Person and Examination: which countries are preferred by exchange students?
- 3) Which courses are preferred in Brazil?

When the Graph-Tree is ready, RMine presents the first level under the root, one SuperNode for each database entity. As noted in Figure 6(a), by selecting one of the SuperNodes, RMine calculates and presents the SuperEdges that connect this SuperNode to the other entities. The size of the SuperNodes are proportional to the number of nodes they represent, and the thickness of the SuperEdges are $c \cdot \log(n)$ to the number n of edges they represent and a configurable constant c . Therefore, one can intuitively tell which are the largest relations (Publication and Event) and which are the most intense relationships (Person-Examination and Person-Publication), just as addressed by question 1). Figure 6 also presents the correspondent SQL commands that would be necessary to produce the same information as that being visually observed.

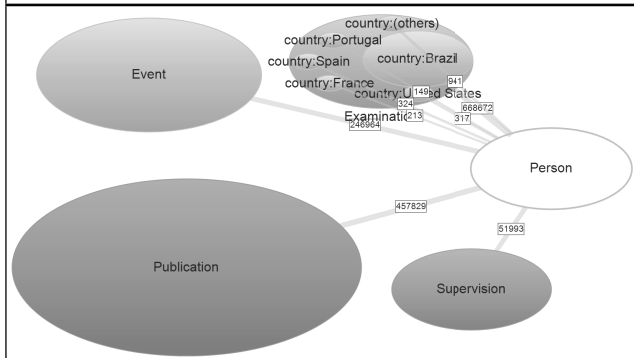
The next interaction step is to expand one SuperNode of interest. This action triggers the next level of partitioning, according to the first attribute. Each SuperNode expansion triggers a number of connectivity calculations



(a) First level of the Tycho-USP GraphTree. The selected SuperEdge represents (Person × Supervision) and corresponds to:
`SELECT p.name, s.id FROM Supervision s JOIN Person p ON p.id = s.advisor OR p.id = s.student`



(b) Expanding the SuperNode Examination. The selected SuperEdge represents (Person × Examination:Country=France) and corresponds to:
`SELECT p.name, x.title FROM Exam x JOIN Person p ON p.id = x.person JOIN Examination e ON e.id = x.examination AND (e.country = 'France')`



(c) Hiding unselected SuperEdges. SuperEdges (Person × Examination:Country=?), corresponding to parametric SQL:
`SELECT p.name, x.title FROM Exam x JOIN Person p ON p.id = x.person JOIN Examination e ON e.id = x.examination AND (e.country = ?)`

Figure 6. Super Nodes and correspondent expansions.

that assign new SuperEdges between the newly exposed children SuperNodes and the remaining SuperNodes in the visualization. In our example, we expand SuperNode Examination according to the country where the examination

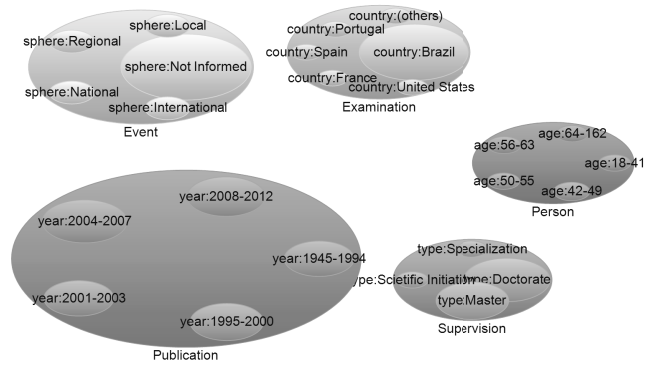


Figure 7. All the SuperNodes expanded in their first partitioning level.

occurred – see Figure 6(b), which highlights the connections to France, and Figure 6(c), which emphasizes entity Person and the parametric aspect of the corresponding SQL. And so, for the Person-Examination relationship, we can answer question 2) by simply reading the weights of the connectivity SuperEdges – they sum up the number of USP students in all countries, pointing to partitions: *Brazil, Spain, United States, France, Portugal, and others*.

After expanding more entities, the visualization will look like Figure 7, in which a subset of partitions is presented in deeper levels of the hierarchy. Looking closely at SuperNode Person, now partitioned by *age*, we see that the automatic partitioning makes each one of the *Person by age* SuperNodes hold a percentile range with an approximately equal number of objects. The figure shows that roughly 20% of the people in this database are older than 42 years old and younger than 49 years old. Now, considering entity Publication, as partitioned by year, one can see that the ranges of the partitions tend to shorten; since the partitioning followed the percentile approach, it means an increase in number of publications along the time.

At this point, by inspecting SuperNode Examination by country, we can answer question 3). For this task, we expand SuperNode Examination:Country=Brazil – Figure 8(a), so that we can view the most representative *courses* together with connectivity SuperEdges to Person by age – Figure 8(b). The visualization shows that Business, Architecture, Law, Nursing, and Education are the preferred courses, and for a given selected course, we can inspect how are the preferences by age. For Architecture course, we can see that younger people (18 through 41 years) answer for the smallest fraction of the people (professors and students) related to this specific course.

From another point of view, we might be interested in analyzing the behavior of the younger fraction of people in relation to all the other courses – Figure 9. The produced visualization points out that Law is the course to which younger professionals and students are most connected to. In RMine, we can go deeper in our data inspection

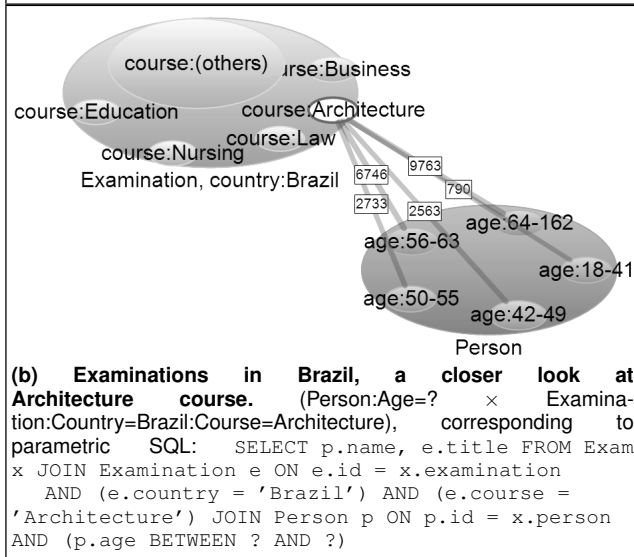
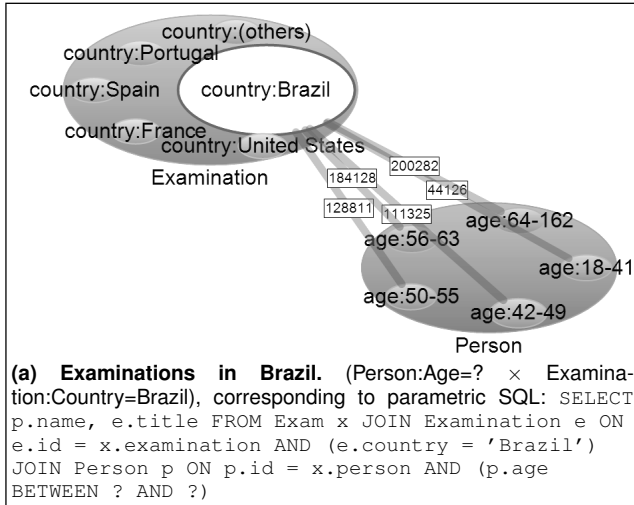


Figure 8. Exploring author-publication relationships.

by retrieving the specific instances that correspond to the observed SuperEdges. For example, if we are interested in knowing details about People:Age=[18-41] and Examination:Country=Brazil:Course=Business, a simple click on its correspondent SuperEdge will bring us the visualization presented in Figure 10. There, in a bipartite graph, one can see the names of the people and the titles of the examinations that are related in the specific context of their correspondent SuperNodes.

This section demonstrated that our method allows for complex aggregation tasks to be performed intuitively in interaction-time constraints. We showed that, not only the user is spared the need to write complex SQL code, but also that the processing requirements are significantly reduced. The cost of these benefits is the preprocessing time that, also, is within acceptable time constraints; especially if we consider the fact that the Graph-Tree structure is persistent

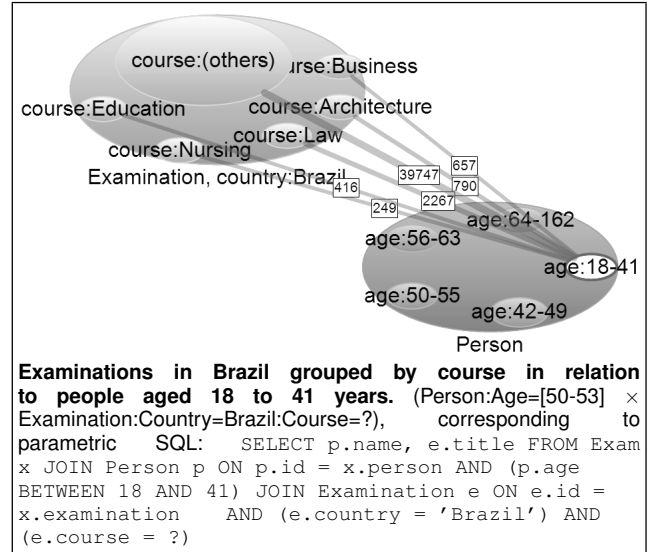


Figure 9. Exploring author-publication relationships.

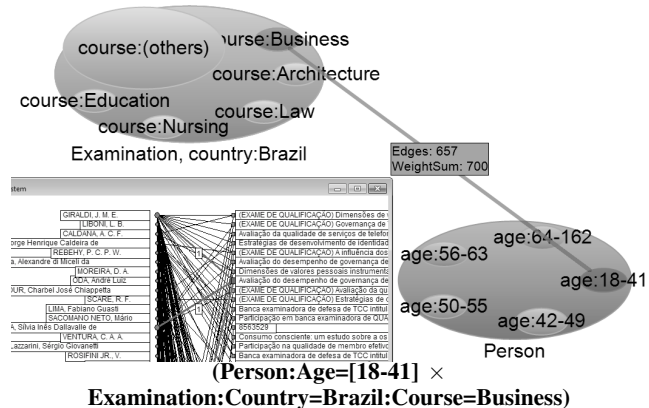


Figure 10. Connectivity of SuperNodes.

on disk.

C. Time measurements

In the previous section, we demonstrated the visual interactive appeal of our method, which lends itself to database exploratory analysis; we also showed how our method prevents users from writing complex SQL queries by means of the Graph-Tree features. Besides these contributions, our method permits costly relational aggregations to be executed in a fraction of the time a relational database would take for the same task. In this section, we demonstrate this feature by comparing the wall-clock timing both for the database execution and for the RMine processing. To this end, we will use 374 different connectivity (SuperNode-to-SuperNode) computations, each one corresponding to a SuperEdge in RMine; and each one corresponding to a complex SQL aggregation query. For our time measures, we use commodity database PostgreSQL 9.2.1 amd64 with

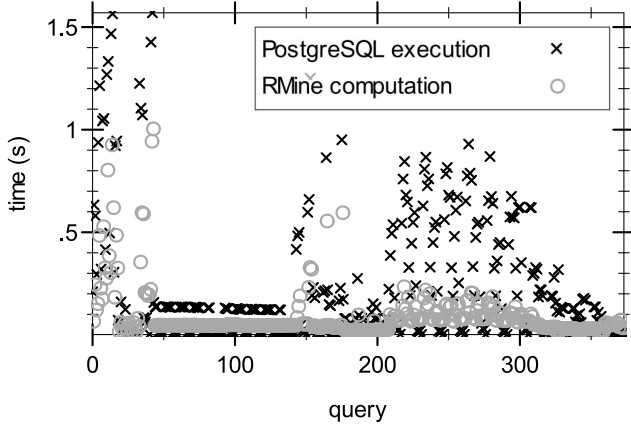


Figure 11. Wall-clock timings for each of 374 RMine connectivity computations and 374 equivalent PostgreSQL aggregations.

a schema tuned with indexes for the column combinations in the following queries.

Consider the connection SuperEdges between *Examination by country by course* and *Person by age*. This information can be accomplished interactively or using an SQL query. As an illustration, the connectivity between SuperNode Examination:Country=Brazil:Course=Business and SuperNode Person:Age=[18-41] can be selected with the following SQL query:

```
SELECT person.name, examination.title FROM
exam
JOIN person ON person.id = exam.person
AND (person.age BETWEEN 18 AND 41)
JOIN examination
ON examination.id = exam.examination
AND (examination.country = 'Brazil')
AND (examination.course = 'Business');
```

The same holds for the 374 connectivity calculations proposed in this section. The calculations vary, both for the commodity RDBMS and for our method, depending on the intensity of the relationship between the SuperNodes involved in the analysis. Hence, we show that the time we take is significantly smaller than the time an RDBMS would take; note that we are not considering the time a user would take to design the SQL queries, this time is supplanted by the interactive aspect already presented in the former section.

In Figure 11 we show the time for each of the queries considered in the experiment. The figure shows that our method achieves the answer faster than the RDBMS in a great number of cases, mainly in heavier steps. The database caching also plays a clear role, showing slower queries interleaved by faster ones. The figure also shows that the time taken varies depending on the attributes involved in the query; some nominal attributes determine unbalanced partitionings, and thus, unbalanced result sets. In other

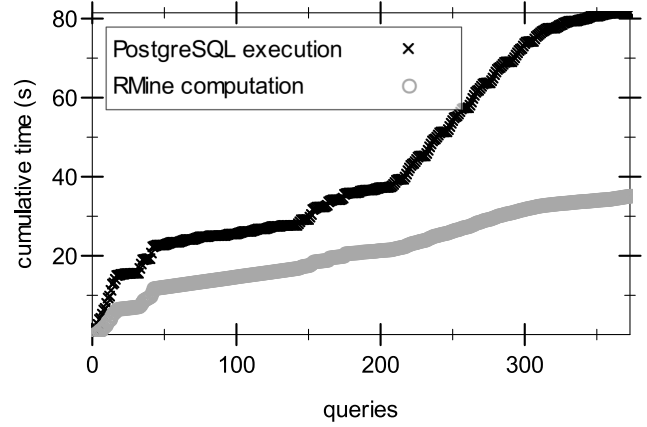


Figure 12. Cumulative wall-clock timings for 374 RMine connectivity computations and 374 equivalent PostgreSQL aggregations.

Table I
CUMULATIVE TIME MEASURED FOR EACH SUPERNODE EXPANSION (SECONDS)

Expand SuperNode	Load	Conn	SQL
(initial loading)	7.330	-	-
Person	0.042	5.955	12.833
Person:50-55	0.014	1.731	2.752
Examination	0.276	7.133	9.513
Examination:U.S.A.	0.000	2.571	2.598
Event	0.168	4.420	9.253
Event:Regional	0.003	0.398	0.713
Publication	0.244	8.584	29.989
Total	8.321	39.376	67.651

words, some queries return more data than others. Figure 12 presents the accumulated time for the queries. The figure shows that, for sequences of queries, our method progresses arithmetically better than the RDBMS; this was a need in the design of our method because exploratory interaction asks for long sequences of trial and error steps.

In Table I we list some times taken by our method and by the RDBMS. Column *Load* corresponds to the time RMine takes to load data from disk; column *Conn* corresponds to the time an expansion operation (connectivity calculation) takes in RMine – an expansion triggers a set of connectivity calculations from the expanded SuperNode to the other SuperNodes in the scene; and column *SQL* corresponds to the time the RDBMS takes to do the same *expand* operations. Still in Table I, we consider rows for the initial load of the preprocessed Tycho-USP Graph-Tree, and rows for *expand* operations considering entities Person, Examination, Event, and Publication. All times are in seconds. The totalizations in the table demonstrate that all steps of the visual interaction were computed faster in RMine than the corresponding relational queries. In fact, RDBMS's are not designed for exploratory data inspection, and this is the point we attack.

V. CONCLUSION

We have defined and experimented on a novel approach for analyzing the structure, the data, and the relationships as defined in relational databases. Our solution was based on the Graph-Tree structure and its related algorithms, which provided an efficient way of storing, retrieving, and calculating the relationship information of the database, features that are key to our method. Over the Graph-Tree, we defined a procedure for reading and organizing the database information according to a semantic-rich hierarchical graph partitioning. The Graph-Tree, then, was used as the basis of RMine, an operational prototype for relational visual analysis.

We worked with a visual graph-based approach that demonstrated to be intuitive in respect to visual exploration, and that proved to be efficient in terms of computational cost. The visual exploration spares the analyst of the need to write complex SQL queries, meanwhile the computational cost benefits from the efficient relationship features provided by the Graph-Tree. For future studies, we envision the coupling of analytical features to aid the user in summarizing the meaning of the multiple data presented over the visualization; also, we consider the possibility of having the Graph-Tree to be dynamically altered according to analytical parameters on the fly.

ACKNOWLEDGMENT

This study received support from the following funding agencies: Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes).

REFERENCES

- [1] R. Agrawal, A. Ailamaki, P. A. Bernstein, E. A. Brewer, M. J. Carey, S. Chaudhuri, ..., and G. Weikum, "The Claremont report on database research," University of California at Berkeley, Sep. 2008. [Online]. Available: <http://db.cs.berkeley.edu/claremont/claremontreport08.pdf>
- [2] G. Anthes, "Happy birthday, RDBMS!" *Comm. ACM*, vol. 53, no. 5, pp. 16–17, May 2010.
- [3] P. P.-S. Chen, "The entity-relationship model – toward a unified view of data," *ACM Trans. Database Syst.*, vol. 1, no. 1, pp. 9–36, Mar. 1976.
- [4] J. Rodrigues, H. Tong, J.-Y. Pan, A. Traina, C. Traina, and C. Faloutsos, "Large graph analysis in the GMine system," *IEEE TKDE*, vol. 25, no. 1, pp. 106–118, jan. 2013.
- [5] J. F. Rodrigues, Jr., H. Tong, A. J. M. Traina, C. Faloutsos, and J. Leskovec, "GMine: a system for scalable, interactive graph visualization and mining," in *Proceedings of the 32nd international conference on Very large data bases*, ser. VLDB '06, 2006, pp. 1195–1198.
- [6] C. Stolte, D. Tang, and P. Hanrahan, "Polaris: a system for query, analysis, and visualization of multidimensional relational databases," *IEEE TVCG*, vol. 8, no. 1, pp. 52–65, jan/mar 2002.
- [7] —, "Query, analysis, and visualization of hierarchically structured data using Polaris," in *ACM SIGKDD '02*, 2002, pp. 112–122.
- [8] E. Thomsen, *Olap Solutions: Building Multidimensional Information Systems*, 2nd ed. New York, NY, USA: John Wiley & Sons, Inc., 2002.
- [9] C. Stolte, D. Tang, and P. Hanrahan, "Multiscale visualization using data cubes," *IEEE TVCG*, vol. 9, no. 2, pp. 176–187, april-june 2003.
- [10] A. S. Maniatis, P. Vassiliadis, S. Skiadopoulos, and Y. Vassiliou, "Advanced visualization for OLAP," in *ACM DOLAP '03*, 2003, pp. 9–16.
- [11] R. Rao and S. K. Card, "The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information," in *ACM SIGCHI '94*, 1994, pp. 318–322.
- [12] K. Techapichetvanich and A. Datta, "Interactive visualization for OLAP," in *ICCSA 2005*, ser. Lecture Notes in Computer Science, O. Gervasi, M. Gavrilova, V. Kumar, A. Laganà, H. Lee, Y. Mun, D. Taniar, and C. Tan, Eds. Springer Berlin / Heidelberg, vol. 3482, pp. 293–304.
- [13] S. Mansmann and M. H. Scholl, "Exploring OLAP aggregates with hierarchical visualization techniques," in *ACM SAC '07*, 2007, pp. 1067–1073.
- [14] B. Wang, G. Chen, J. Bu, and Y. Yu, "Zoomtree: Unrestricted zoom paths in multiscale visual analysis of relational databases," in *Computer Vision, Imaging and Computer Graphics. Theory and Applications*, ser. Communications in Computer and Information Science, P. Richard and J. Braz, Eds. Springer Berlin Heidelberg, 2011, vol. 229, pp. 299–317.
- [15] J. Abello, F. van Ham, and N. Krishnan, "ASK-GraphView: A large scale graph visualization system," *IEEE TVCG*, vol. 12, no. 5, pp. 669–676, sept.-oct. 2006.
- [16] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information," *Psychological Review*, vol. 63, no. 2, pp. 81–97, 1956.
- [17] H.-J. Schulz, "Treevis.net: A tree visualization reference," *IEEE CGA*, vol. 31, no. 6, pp. 11–15, nov.-dec. 2011.
- [18] N. Elmqvist and J.-D. Fekete, "Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines," *IEEE TVCG*, vol. 16, no. 3, pp. 439–454, may-june 2010.