INTRODUCTION

Increasingly, management of data representing relationship structures across entities is an important issue in several application domains. Often, semantics are hidden in the way different entities are related, which may not be obvious by studying each entity in isolation. Querying on properties of such relationship structures are sometimes as important, if not more so, as querying for values of specific attributes.

Some application examples follow:

• In proteomics – or the study of proteins, it is well known that the conformation, or the 3D structure, of a protein is strongly correlated to the functioning of the protein (Ex: Graves et. al, 1995; Colinge and Bennett,
The conformation map of a protein is formed by interactions between the different atoms that make up the protein molecule. Studying these interaction patterns hence, is of immense importance.

- In the field of computer-aided design of electrical circuits, some interconnection patterns across electrical devices keep recurring in different problems. Managing data about electrical circuits and comparing similarities between two or more circuits is of significant importance.

- In the field of information security, a commonly occurring challenge is to look for anomalous behavior by one or more entities, like people, software, etc. in the system. Anomalies constitute deviations from the norm that are distinct from “noise” or “novel” deviations – both of which also constitute deviations from the norm (Chandola et. al 2009). A central aspect of anomaly detection constitutes studying and reasoning about the relationship patterns that an entity has with other entities in the system.

- Designers of public infrastructure like airports, hospitals, etc. face a number of conflicting requirements concerning safety, efficiency, reachability, cost, etc. Conflicting requirements like these are often modeled as network design and facility location problems (cf. Brandes and Erlebach, 2005) over graphs. Since designing a facility from scratch is often a hard problem, architects use design patterns depicting best practices and architectural features that have been recorded over time. Management of such datasets requires the database to support structural queries over graphs.

Management of graph structured data is not a new problem in the field of databases. Most data structures at the application level usually need to deal with some form of graph structure. Early approaches to data management did not have clear separations between logical and physical views of data. As a result, graph structured data from the application were stored directly as a networked set of records at the physical level. An example of this is the Network data model (Taylor and Frank, 1976).

Later approaches towards data management had clearer separations between the physical layout of records on disk and the logical structuring of the database schema. The most popular among these is of course, the relational data model (Codd 1983), which has become the standard for most commercial DBMS today. The relational data model gave mathematical underpinnings to the logical structuring of data, based on concepts of relations, functional dependencies and normal forms. Relations in a relational schema are captured in two different forms: an n-ary relation across several attributes of a given semantic entity; and relations across entities. The former kind of relation is represented in the form of a table – every column in a table is related to every other, by virtue of being attributed to a single entity. The second form of relation is represented by means of foreign keys across tables.

So, in a sense, every relational database represents a graph structured logical model for the data. However, conventional relational databases are inadequate for the challenges of managing graph data as illustrated in the application examples. They can be summarized as follows:

- A relational database represents several instances of a single graph structure that is captured by the schema. Graph databases on the other hand, may need to contend with several graphs with different structures as part of the same database.

- While a graph can be represented as a set of relations (for example, modeling each edge as a relation) and stored as a set of rows in a table, doing so would violate an implicit assumption in relational databases.
Related Content

An Open ECA Server for Active Applications
[www.igi-global.com/article/open-eca-server-active-applications/3392?camid=4v1a](www.igi-global.com/article/open-eca-server-active-applications/3392?camid=4v1a)

Leveraging Early Aspects in End-to-End Model Driven Development for Non-Functional Properties in Service Oriented Architecture
[www.igi-global.com/article/leveraging-early-aspects-end-end/52994?camid=4v1a](www.igi-global.com/article/leveraging-early-aspects-end-end/52994?camid=4v1a)

Using Harel's Statecharts to Model Business Workflows
[www.igi-global.com/article/using-harel-statecharts-model-business/3281?camid=4v1a](www.igi-global.com/article/using-harel-statecharts-model-business/3281?camid=4v1a)

Real-Time Object-Oriented Database Support For Program Stock Trading
[www.igi-global.com/article/real-time-object-oriented-database/51132?camid=4v1a](www.igi-global.com/article/real-time-object-oriented-database/51132?camid=4v1a)