SUGAR: A Graph Database Fuzzy Querying System
Olivier Pivert, Olfa Slama, Grégory Smits, Virginie Thion

To cite this version:
Olivier Pivert, Olfa Slama, Grégory Smits, Virginie Thion. SUGAR: A Graph Database Fuzzy Querying System. IEEE International Conference on Research Challenges in Information Science (RCIS’16) Demos, Jun 2016, Grenoble, France. 10.1109/RCIS.2016.7549366. hal-01297185

HAL Id: hal-01297185
https://inria.hal.science/hal-01297185
Submitted on 22 Nov 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License
SUGAR: A Graph Database Fuzzy Querying System

Olivier Pivert, Olfa Slama, Grégory Smits, Virginie Thion
Rennes 1 University / IRISA – Lannion, France
Email: {Olivier.Pivert, Olfa.Slama, Gregory.Smits, Virginie.Thion}@irisa.fr

Abstract—Graph databases have aroused a large interest in the last years thanks to their large scope of potential applications. Defining a language allowing a flexible querying of graph databases may greatly improve usability of data. In this paper, we present a system for querying graph databases in a flexible way. The preferences are based on fuzzy set theory and may concern i) the content of the vertices and ii) the structure of the graph.

I. INTRODUCTION

Much work has been done about fuzzy querying of relational databases (see e.g. [1]). However, even though relational databases are still widely used, the need to handle complex data has led to the emergence of other types of data models. In the last few years, a new concept has started to attract a lot of attention in the database world, namely that of graph databases (see e.g. [2]), whose basic purpose is to efficiently manage networks of entities where each node is described by a set of characteristics (e.g. a set of attributes), and each edge represents a link between entities. Such a database model has many potential applications, e.g. for modeling social networks, RDF data, cartographic databases, bibliographic databases, etc. This model may be extended into the notion of a fuzzy graph database where a degree may be attached to edges. This degree may express the “intensity” of any kind of gradual relation between two nodes (e.g. likes, is friends with, is about). Graph databases, which may be fuzzy or not, raise new challenges in terms of flexible querying since two aspects may be involved in the preferences that a user may express: i) the content of the nodes and ii) the graph structure.

![Fuzzy data graph](attachment://fuzzy_data_graph.png)

Fig. 1. A fuzzy data graph $DB$ derived from an excerpt of DBLP data

We present SUGAR, a system that makes it possible to query a fuzzy graph database in a flexible way. This system is built on the Neo4j Graph Database Management System [3] and implements the FUDGE language introduced in [4].

II. GRAPH DATABASE FUZZY QUERYING

The last decade has witnessed a growing interest in preference queries (see e.g. [1]). Motivations for introducing preferences inside database queries are manifold. First, it has appeared to be desirable to offer more expressive query languages that can be more faithful to what a user intends to say. Second, the introduction of preferences in queries provides a basis for rank-ordering the retrieved items, which is especially valuable in case of a large set of items satisfying a query. Third, a classical query may also have an empty set of answers, while a relaxed version of the query may be matched by some items. In the graph database context, the need for having flexible querying tools is even more acute as users are in general unfamiliar with the representation of the data [2]. In the following, we focus on the fuzzy-set based approach to preference queries, which is founded on the use of fuzzy set membership functions that describe the preference profiles of the user on each attribute domain involved in the query.

Fuzzy set theory allows to model classes or sets whose boundaries are not clear-cut. For such sets, the transition between full membership and full mismatch is gradual rather than crisp (non fuzzy). Typical examples of such fuzzy classes are those described using adjectives of the natural language, such as young, short, cheap, etc. Formally, a fuzzy set $F$ on a referential $U$ is characterized by a membership function $\mu_F : U \rightarrow [0, 1]$ where $\mu_F(u)$ denotes the grade of membership of $u$ in $F$. In particular, $\mu_F(u) = 1$ reflects full membership of $u$ in $F$, while $\mu_F(u) = 0$ expresses absolute non-membership. When $0 < \mu_F(u) < 1$, one speaks of partial membership. Two crisp sets are of particular interest when defining a fuzzy set $F$: the core $C(F) = \{ u \in U | \mu_F(u) = 1 \}$, which gathers the prototypes of $F$, and the support $S(F) = \{ u \in U | \mu_F(u) > 0 \}$. In practice, the membership function associated with $F$ is often of a trapezoidal shape. Then, $F$ is expressed by the quadruple $(A, B, a, b)$ where is $C(F) = [A, B]$ and $S(F) = [A-a, B+b]$.

III. THE FUDGE LANGUAGE

The FUDGE language [4] is based on the CYPHER query language, used for querying graph databases in a crisp way in the Neo4j graph database management system [3]. FUDGE relies on a formal algebra defined in [4].

We focus here on selection queries in which fuzzy preferences may appear. A FUDGE selection query is composed of: (1) a list of define clauses for fuzzy term declarations. If a fuzzy term $f_{term}$ is defined by a trapezoidal function with the four positions (abscissa) $A$, $a$, $B$, $b$, then the clause has the form $\text{define } f_{term} \text{ as } (A-a, A, B-b)$. If $f_{term}$ is a decreasing membership function, then the clause has the form $\text{define } f_{term} \text{ as } (B-b, B-a)$ (and there is the corresponding define clause for increasing functions). (2) a match clause of the form $\text{match } pattern \text{ where condition}$ where $pattern$ is a fuzzy graph pattern and $condition$ is the possibly compound condition attached to the pattern. A graph pattern is defined $\text{a la ASCII art}$, in the CYPHER way, where the graphic symbol $\text{node}$ denotes a node, which may contain information...
of the form `query_variable:Type` concerning this node. The symbol `-[form]->` denotes a connection i.e. either an edge where `form` may denote the label of the edge, or a path where `form` is a fuzzy regular expression that the path has to satisfy.

The FUDGE Query 1 aims to retrieve information concerning authors (au2) who have, among their close contributors, an author (au1) who published a paper (ar1) in www and also published a paper (ar2) in Pods recently (ar2.year is recent). The `define` clauses define the fuzzy terms short and recent.

```
1 DEFINE short AS (3,5), recent AS (2010,2014)
2 MATCH
3 (ar1:Article)-[:part_of]->()>-[:series]->(s1),
4 (ar2:Article)-[:part_of]->()>-[:series]->(s2),
5 (ar1)-[:creator]->(au1:Author),
6 (ar2)-[:creator]->(au1:Author),
7 (au1)-[:Length]->(au2:Author)
8 WHERE s1.id=WWW AND s2.id=Pods AND ar2.year IS recent
```

Query 1. A FUDGE query

As illustrated here, fuzzy conditions may concern either the content of vertices and edges, e.g. the year of an article may be recent, or a path, e.g. a path going from an author to another one is required to be short.

IV. THE SUGAR SOFTWARE

The SUGAR software implements the FUDGE language. It is built from the Neo4j REPL Console RabbitHole [6] supporting CYPHER queries. SUGAR is a software add-on composed of two modules, which interact with a Neo4j crisp engine (see Fig. 2): the Transcriptor module and the Score Calculator one. The former aims to translate a FUDGE query into a (crisp) CYPHER one that retrieves all needed data. This query is then sent to the crisp Neo4j engine. Then the Score Calculator module extracts answers and calculates the satisfaction degree associated with each answer returned by the crisp engine, and ranks the answers.

```
FUDGE query (fuzzy terms)
```

Client (user)

```
SUGAR add-on
```

```
SUGAR Transcriptor
```

```
SUGAR Score Calculator
```

```
SUGAR software architecture
```

IV. THE SUGAR SOFTWARE

The SUGAR software implements the FUDGE language. It is built from the Neo4j REPL Console RabbitHole [6] supporting CYPHER queries. SUGAR is a software add-on composed of two modules, which interact with a Neo4j crisp engine (see Fig. 2): the Transcriptor module and the Score Calculator one. The former aims to translate a FUDGE query into a (crisp) CYPHER one that retrieves all needed data. This query is then sent to the crisp Neo4j engine. Then the Score Calculator module extracts answers and calculates the satisfaction degree associated with each answer returned by the crisp engine, and ranks the answers.

```
SUGAR Transcriptor
```

```
SUGAR Score Calculator
```

```
Crisp CYPHER query
```

```
SUGAR system can be downloaded at www.shaman.irisa.fr/fudge-prototype, which also proposes complementary screenshots.

Acknowledgement: This work has been partially funded by the French DGE (Direction Générale des Entreprises) under the project ODIN (Open Data INtelligence).

REFERENCES