# New project - BIG DATA

Mirian, Martin, Jacques, Simone

No Institute Given

**Abstract.** Notes (some of them copied from articles and books) to be used as basis for future work and project.

## 1 Introduction

Nowadays, large volumes of graph-structure data are widespread in many different domains such as social network, scientific computing, telecoms network, linked open data, organization management, finance chains and so on ([**?**]).

As discussed in [**?**], graph-structured data differs from ordinary big data in its deeper focus on relationships between concepts or entities. Relationships are as important as entities, and we need not only to model attributes or constraints on them but also to easy query and analysis over them.

In this project we consider a data graph distributed in many different machines. One machine contains the specification of this graph (schema + constraints).

Our goals are:

1. Choose and extend a schema language (possibly including some constraints)
2. Use a resolver to infer new constraints; test consistency.
3. Propose a user-friendly query language over the data graph. First step: a datalog-like language.
4. Consider fragmentation/distribution according to usual queries.
5. Propose query execution plans for efficient query evaluation.

## 2 Schema Language

The graph schema is described by using a descriptive logic (DL), to be translated into RDFS or OWL for implementation. In our context, the schema is a Tbox (description of concepts, relationships and some kinds of constraints). The Abox is the graph, basically described by concepts and relationships. Tbox is not distributed while Abox is.

We start by considering the DL-LITE fragment.

### 2.1 DL-LITE

The family of DL-LITE has been designed for capturing the main modelling primitives of conceptual data model such as E-R or UML while remaining tractable in the presence of inclusion statements and a certain form of negation ([**?**]).

- The constructs allowed in DL-LITE are unqualified existential restrictions on roles ($\exists R$) and on inverse roles ($\exists R^-$) and the negation.
- The axioms allowed in a Tbox of DL-LITE are concept inclusion statements of the form $B \sqsubseteq C$ or $B \sqsubseteq \neg C$ where $B$ and $C$ are atomic concepts or existential restriction ($\exists R$ or $\exists R^-$).
- Negation is only allowed in the right-hand side of inclusions.
- We have 2 types of inclusion: negative inclusion (NI) and positive inclusion (PI).
- A DL-LITE schema may contain axioms corresponding to those proposed in RDFS. Besides it may contain other axioms of three kinds: PI, NI and key constraints which are not expressible in RDFS (see OWL, so).
- There are two dialects of DL-LITE namely :
  - A DL-LITE$_\mathcal{R}$: Tbox allows role inclusion statements $P \sqsubseteq Q$ or $P \sqsubseteq \neg Q$ where $P$ and $Q$ are atomic roles or inverse atomic roles. Thus it is obtained by extending axioms of RDFS with PI and NI.
  - A DL-LITE$_\mathcal{F}$: Tbox allows functionality statements on roles of the form $funct P$ of $funct P^-$. To express keys. Thus it is obtained by extending axioms of RDFS with key constraints, PI and NI, but *excluding inclusion between properties*. Due to this exclusion, RDFS is not included in DL-LITE$_\mathcal{F}$.
- Remark: Why not simply extend RDFS with three kinds of axioms? Because functional constraints interact with inclusion constraints in intricate way.
- Two fundamental differences between query answering in the context of RDFS and DL-LITE knowledge bases:
  - Inconsistency: RDFS does not permit negation, but DL-LITE allows some kind of negation - so, testing consistency is important.
  - Incompleteness: RDFS rules are safe (allowing the simple bottom-up algorithm to answer queries). However axioms in DL-LITE may correspond to unsafe rules (cf. blanc nodes). We have to use top-down approach for evaluating queries !!!!
- Due to all these remarks... We start by using DL-LITE$_\mathcal{R}$

### 2.2 Schema

The schema describes a graph. For defining a schema as a DL knowledge base we need:

- A vocabulary composed of a set **C** of atomic concepts $A, B, \ldots$, a set **R** of atomic roles $P, Q, R, \ldots$ and a set of atomic individuals $a, b, c \ldots$.
- A set of constructs used for building tuples, complex concepts and roles from atomic individuals, concepts and roles.
- A language of axioms that can be stated for constraining the vocabulary in order to express domain constraints.
- Atomic individuals are all names used to denote singular entities (be they persons, objects or anything else) in our domain of interest.
- If $a_1, \ldots, a_n$ are atomic individuals, then $\langle a_1, \ldots, a_n \rangle$ is a n-ary individual tuple. In this document we only deal with binary tuples.
- A language of axioms that can be stated for constraining the vocabulary in order to express domain constraint.

**Definition 1 (DL Lite Syntax).** *Let $A \in \boldsymbol{C}$ be an atomic concept name, $P \in \boldsymbol{C}$ be an atomic role and $P^-$ the inverse of $P$. Valid general concepts are defined by the abstract syntax:*

$$B ::= A \mid \exists R \qquad R ::= P \mid P^-$$
$$C ::= B \mid \neg B \qquad E ::= R \mid \neg R$$

*$B$ denotes a basic concept, i.e., an atomic concept or a concept of the form $\exists R$ and $R$ denotes a basic role, i.e., a role that is either an atomic role or the inverse of an atomic role. Finally, $C$ denotes a (general) concept, which can be a basic concept or its negation, whereas $E$ denotes a (general) role, which can be a basic role or its negation.*

## 3 Data Graph

## 4 Query Language: sequential version

## 5 Query Language: distributed version

## 6 Data distribution

## 7 Query Plan

Develop a general algorithm for planing query execution.

We start with a query in OUR datalog-like language, where indexes are indicated. One idea is to fix as indexes terms involved in joins, since this can contribute to the reduction of the communication steps.

Let us consider the query $Q([x], sum(y)) \leftarrow A(x, [z]), B([z], y)$ to be executed. We suppose that relations A and B are distributed in 4 processors.

We propose general algorithms for constructing datalog query plans... according to the indexes "positions".

We extend our first propositions to queries containing intentional predicates and we consider recursive queries.

–

## 8 Les objectifs et l'originalité de la proposition

## 9 Projet de recherche

## 10 Objectifs en terme de formation

## 11 L’infrastructure, la pertinence du partenariat et les projets associés

## 12 Les retombées espérées à la fin du projet

## 13 Plan de travail et ressources humaines

## 14 Plan d’application des ressources

## 15 Description du partenariat et projets associés des partenaires