

INTRODUCTION

Graphs vs Relational Databases

Graph databases have become more common when it comes to managing large amounts of data. As a group, graph databases provide a clear benefit over relational databases for the following reasons:

- The conceptual data model translates directly into the graph database. This means there are no constraints on external knowledge to do joins, as with relational databases.
- Graph databases are flexible and easily expandable. This gives a possibility to add new nodes and relationships, without disturbing the existing graph.
- Metadata can be stored directly as part of the data. This makes it much easier for users to understand the data.
- Data can be evaluated in different contexts. This facilitates flexibility, since data can be explored starting from any node within the network.

Types of Graphs

All graph databases have one thing in common; they consist of nodes connected by edges. However, how the nodes and their relations are modelled and stored depends very much on the purpose of the database.

Two common graph database techniques are the RDF (Resource Description Framework) graph and the property graph databases.

- *RDF graph databases* model and store the data in an atomic way, where the nodes often are primitive data types like string, integer, date, etc.

- *Property graph databases* add properties to the nodes and relations. The properties can also be called attributes, fields, etc. Examples of property graphs are the Capish database and the Neo4j database.

COMPARISON - RDF GRAPHS VS PROPERTY GRAPHS

RDF Graph

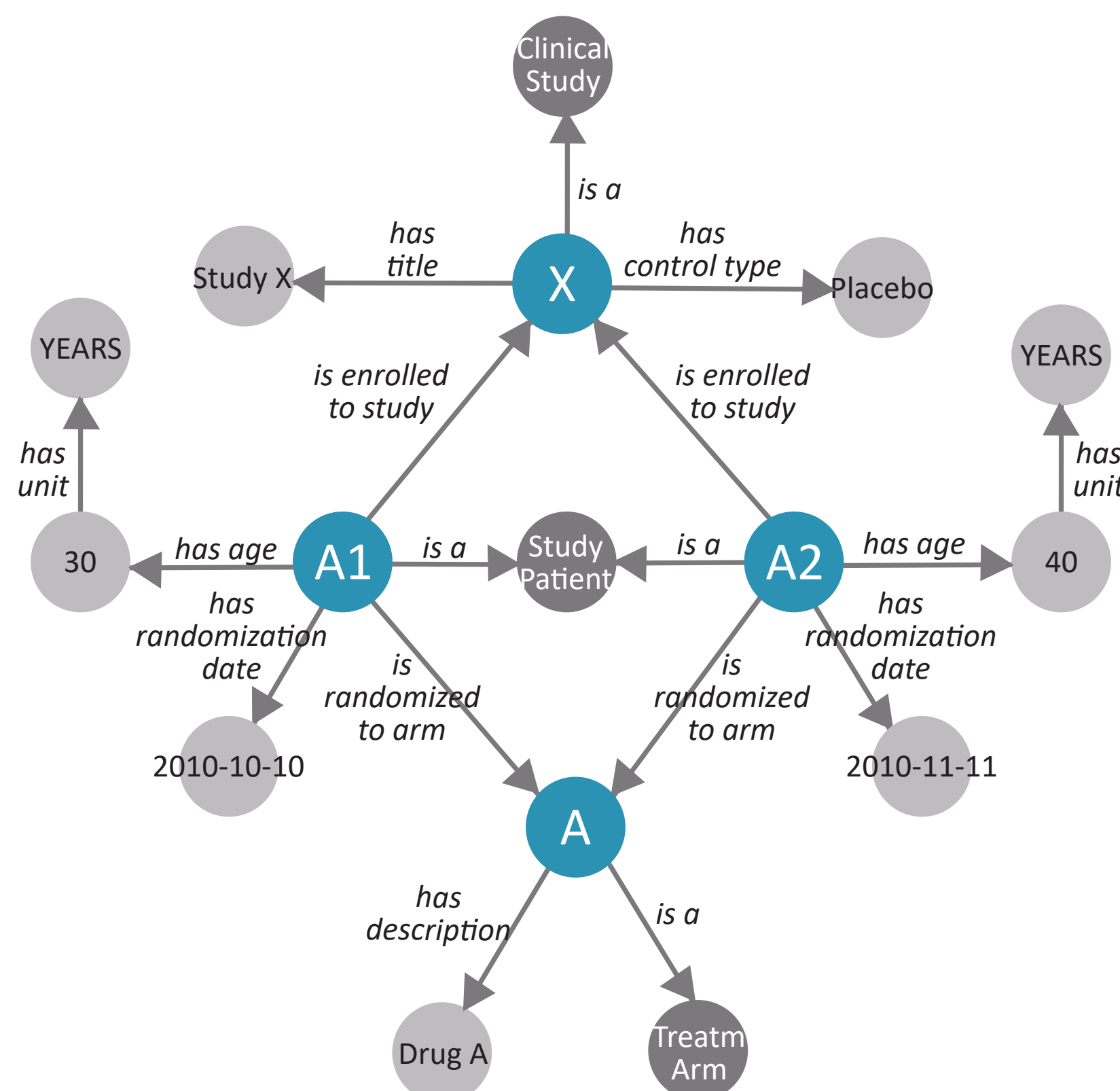
RDF is built upon W3C's linked data technology, which has been developed for the semantic web. For this reason, the items in an RDF database are identified by URIs, as a way of "name-spacing" the identifiers.

RDF is based on "triple stores", with statements comprising of a subject, predicate and object. The predicate in this statement represents a relationship between the subject and object.

Triple store examples:

URI_1 hasName Mary
URI_1 knows URI_2
URI_2 hasName John

- = Node corresponding to a Holon
- = Node corresponding to a Holon type
- = Node corresponding to a property value



Benefits

Facilitates traversal of the database from one node to the next (for example for content on the web), providing:

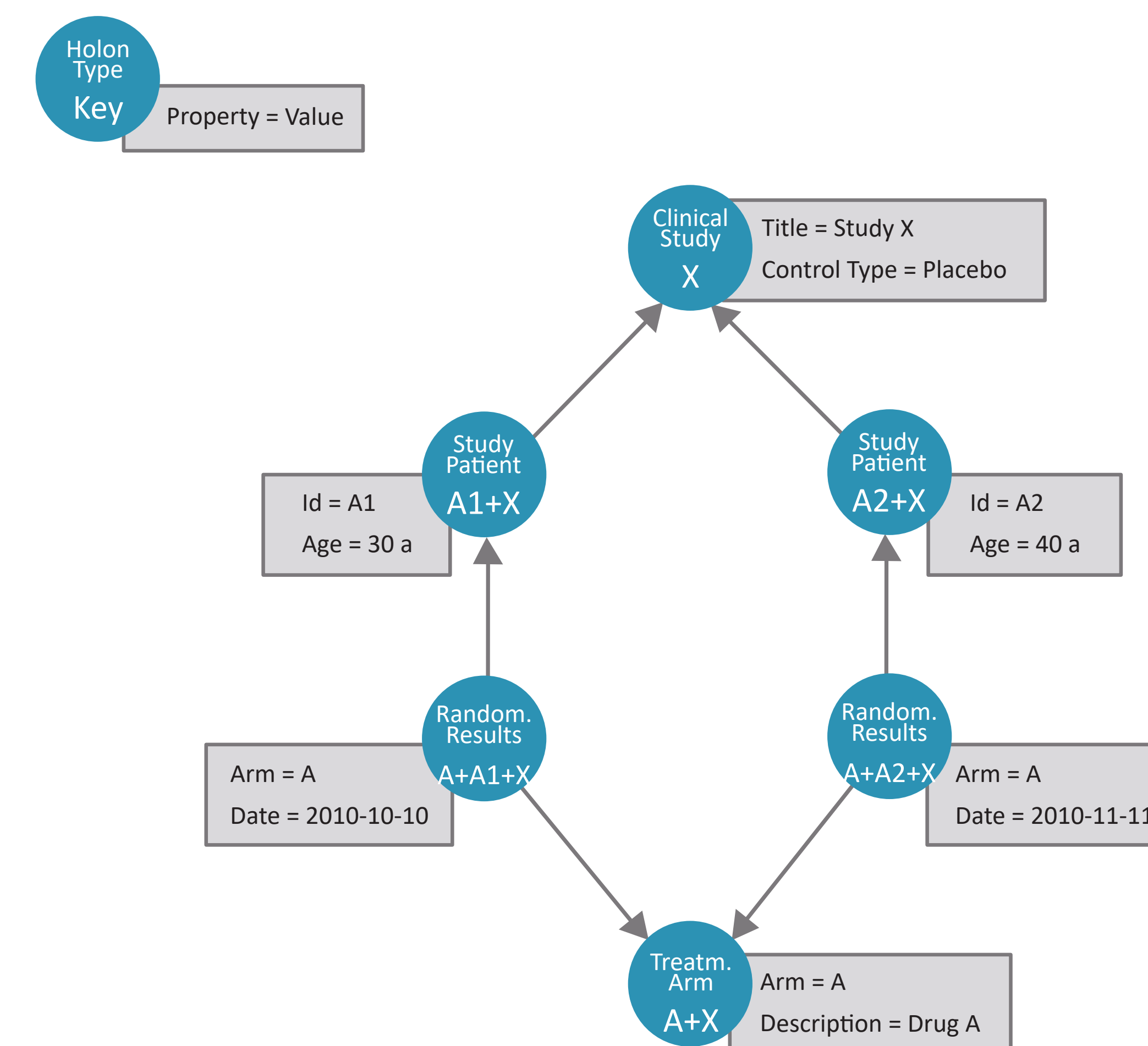
- A possibility to do "friend of a friend" traversals
- A possibility to do inference (e.g. the conclusion that John knows Mary in the triple store example above)
- Facilitated data sharing between many parties, due to an easy handling of many different vocabularies and namespaces

The data used in this comparison is clinical trial data from the demography (DM) and trial summary (TS) domains in the CDISC SDTM standard.

STUDYID	DOMAIN	SUBJID	RFSTDTC	AGE	AGEU	ARMCD	ARM
X	DM	A1	2010-10-10	30	YEARS	A	Drug A
X	DM	A2	2010-11-11	40	YEARS	A	Drug A

STUDYID	DOMAIN	TSPARMCD	TSPARM	TSVAL
X	TS	TITLE	Trial Title	Study X
X	TS	TCNTRL	Control Type	Placebo

CDISC SDTM Tables



Benefits

Facilitates analytical processing by avoiding costly node traversals and by describing the data with an ontology, providing:

- A more intuitive data querying process, since the user does not need to know the chain of relations
- A possibility to add a reflection point, which enables users to:
 - Analyze the database from different perspectives
 - Search for a specific criterion and retrieve a broader result set
- Enhanced data quality, since the data is strictly curated and modelled according to an ontology

Capish Holon Graph

Property graph databases store the data as a graph internally. Capish has advanced the classic property graph database by using an ontology as a data model and by implementing several indices.

In the classic property graph database approach, there is a gray area whether a specific data point should end up as a node or as a relation. In the Capish ontology-based information model, all data points end up in the nodes (termed "Holons") and information needed to interpret the relations is given in so-called "Relation Holons".

DISCUSSION

The differences between RDF and property graph databases boil down to how the graphs are stored and modelled. Which one to use depends on the actual use case.

For a data sharing arrangement between many parties, the URIs of an RDF database are beneficial. Also in cases of graph traversals, RDF is advantageous.

However, the traversal from one node to the next is not ideal for analytical processing, as the traversal process results in performance issues. Property graph databases reduce the number of nodes, which therefore could help to increase the performance.

To further allow for an efficient analysis of patterns and correlations, several indices can be calculated. By this, query performance is greatly improved, as related nodes can be accessed in a single step. Compared to classic analysis approaches, this is avoiding the costly join operations needed for relational database solutions, or node traversal as in other graph database implementations.

By matching the graph with an ontology, time will be saved also in the long run, since data is properly modelled directly and curation does not need to be repeated for each analysis purpose.

CONCLUSION

While much emphasis in the clinical world has been on RDF databases, they are not considered to be ideal for analytical processing. By using a property graph together with an ontology and an indexing process, many of the performance issues associated with the classic graph database approach could be eliminated.

Convert graph databases into powerful analytical systems