TECHNISCHE UNIVERSITÄT DRESDEN

DATABASE THEORY

Lecture 16: Graph Databases and Path Queries

Markus Krötzsch

Knowledge-Based Systems

TU Dresden, 26th June 2018

Review: Datalog

Datalog is a powerful recursive query language

Advantages:

- Natural extension of (U)CQs with recursion
- Can be extended with (EDB) negation
- Polynomial data complexity of query answering

Disadvantages:

- High query and combined complexity (ExpTime)
- Perfect optimisation is undecidable
- Somewhat complicated to write queries

Graph Databases

Our original motivation for going from FO queries to Datalog:
Reachability of nodes in a (directed) graph $\rightsquigarrow$ let's focus on graphs

Graph database: a DBMS that supports "graphs" as its datamodel

There are many kinds of graphs:

- Directed or undirected?
- Labelled or unlabelled edges/nodes?
- What kinds of labels? Datatypes?
- Parallel edges (multi-graphs)? With same label?
- One graph or several graphs per database?

Two types of graph database models dominate the market today: Resource Description Framework (RDF) and Property Graph

Resource Description Framework (RDF)

RDF is a W3C standard for representing linked data on the Web

- Directed labelled graph; nodes are identified by their labels
- Labels are URIs or datatype literals
- Multiple parallel edges only when using different edge labels
- Supports multiple graphs in one database
- W3C standard; implementations for many programming languages
- Datatype support based on W3C XML Schema datatypes
- Graphs can be exchanged in many standard syntax formats

Property Graph

Property Graph is a popular data model of many graph databases

- Directed labelled multi-graph; labels do not identify nodes
- “Labels” can be lists of attribute-value pairs
- Multiple parallel edges with the exact same labels are possible
- No native multi-graph support (could be simulated with additional attributes)
- No standard definition of technical details; most common implementation: Tinkerpop/Blueprints API (Java)
- Datatype support varies by implementation
- No standard syntax for exchanging data

Representing Graphs

Graphs (of any type) are usually viewed as sets of edges

- RDF: triples of form subject-predicate-object
  - When managing multiple graphs, each triple is extended with a fourth component (graph ID) $\rightsquigarrow$ quads
  - RDF databases are sometimes still called “triple stores”, although most modern systems effectively store quads
- Property Graph: edge objects with attribute lists
  - represented by Java objects in Blueprints

Graphs can be stored in relational databases

- RDF: table Triple[Subject,Predicate,Object]
- Property Graph: tables Edge[SourceId,EdgeId,TargetId] and Attributes[Id,Attribute,Value]

Representing Data in Graphs

Property Graphs can represent RDF:

- use attributes to store RDF node and edge labels (URIs)
- use key constraints to ensure that no two distinct nodes can have same label

Representing Data in Graphs

Property Graphs can represent RDF:

- use attributes to store RDF node and edge labels (URIs)
- use key constraints to ensure that no two distinct nodes can have same label

RDF can represent Property Graphs:

- use additional nodes to represent Property Graph edges
- use RDF triples with special predicates to represent attributes

Either model can also represent hypergraphs/RDBs (exercise)

$\rightsquigarrow$ all models can represent all data in principle
$\rightsquigarrow$ supported query features and performance will vary

Querying Graphs

Preferred query language depends on graph model

- RDF: W3C SPARQL query language
- Property Graph: no uniform approach to data access
  - many tools prefer API access over a query language
  - proprietary query languages, e.g., “Cypher” for Neo4j

However, there are some common basics in almost all cases:[1]

- Conjunctive queries
- Regular path queries

[1] Might not be true for Cypher, which – in contrast to most other database query languages – is based on a variant of graph isomorphism rather than homomorphism; and which supports only specific path expressions

Conjunctive Queries over Graphs

Basic descriptions of local patterns in a graph

Formally, it suffices to say:

> "CQs over RDF correspond to CQs over relational databases with a single table Triple[Subject,Predicate,Object]"

(and analogously for Property Graphs)

- All complexity results for query answering and optimisation carry over from RDBs (in particular, restricting to graphs does not make anything simpler)
- Details of representation of data in tables do not matter
- CQs are restricted to local patterns (no reachability . . .)

Summary and Outlook

Graph databases as an important class of “noSQL” databases

Two main data models

- Resource Description Framework (RDF)
- Property Graph

Conjunctive queries can be used on graphs, but can hardly take the connected nature of graphs into account

Next topics:

- Regular path queries
- Logical dependencies
- Query answering under constraints