#NoSQL: The Graph-like Data Model

Saptarshi Pyne
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology Jodhpur, Rajasthan, India 342030

CSL4030 Data Engineering Lecture 10
August 23st, 2023

What we discussed in the last class

- NoSQL databases
  - The document data model: Suitable when
    - we have documents having a one-to-many tree relationship
    - and we need to load the whole tree every time
  - Document oriented DBMS
    - MongoDB by MongoDB Inc.
    - CouchDB by Apache (open source)

Today we will discuss

- NoSQL databases
  - The graph-like data model

The graph-like data model

- Suitable when we have documents having many-to-many relationships

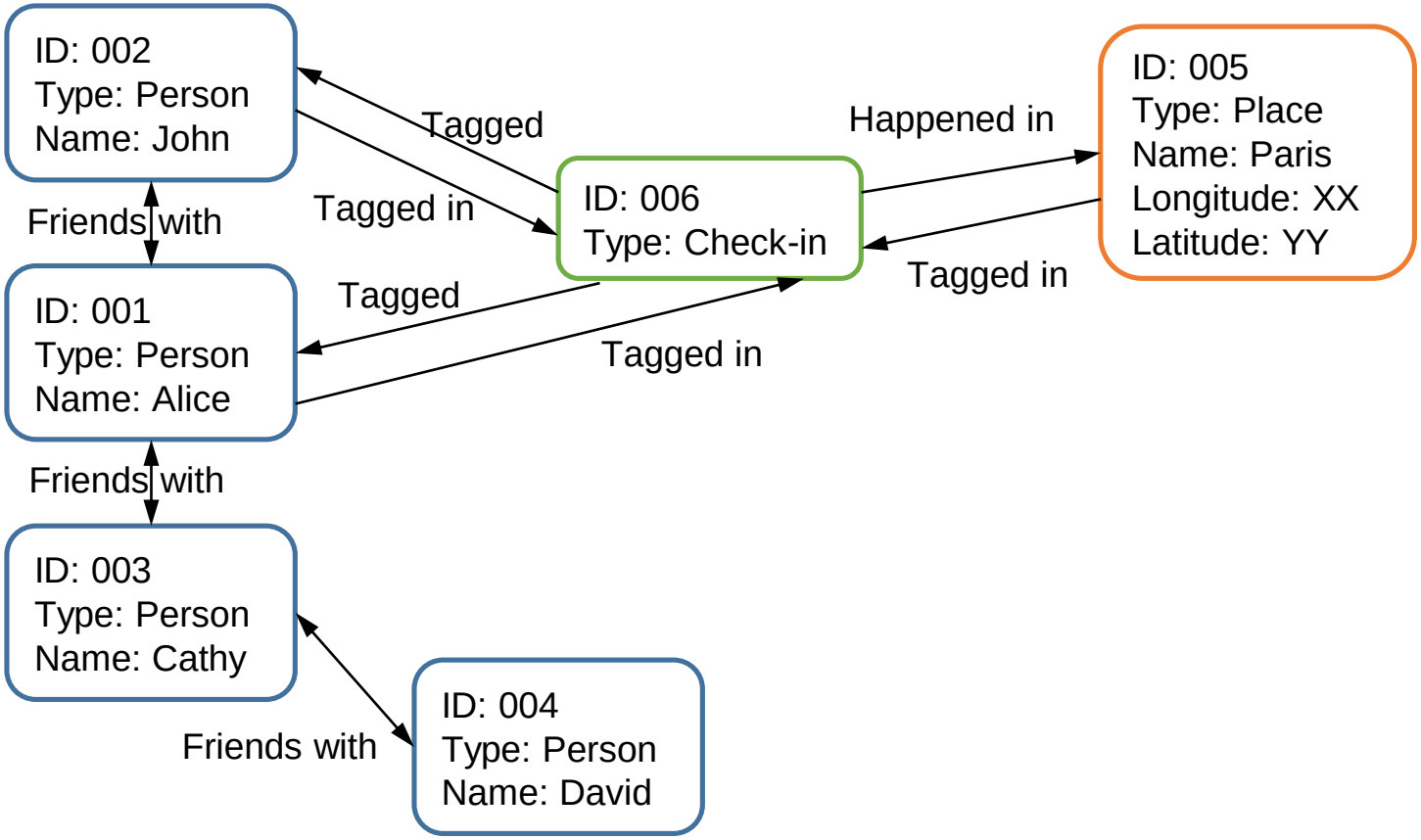
- Organize documents as a graph
  - Social media apps
    - Nodes = People
    - Edges = Friends
  - Internet
    - Nodes = Webpages
    - Edges = Hyperlinks

An example of a graph-like database for social media apps

@Alice checked in @Paris with @John.
Cathy: Wish @David and I were there!
David: Liked Cathy's comment.

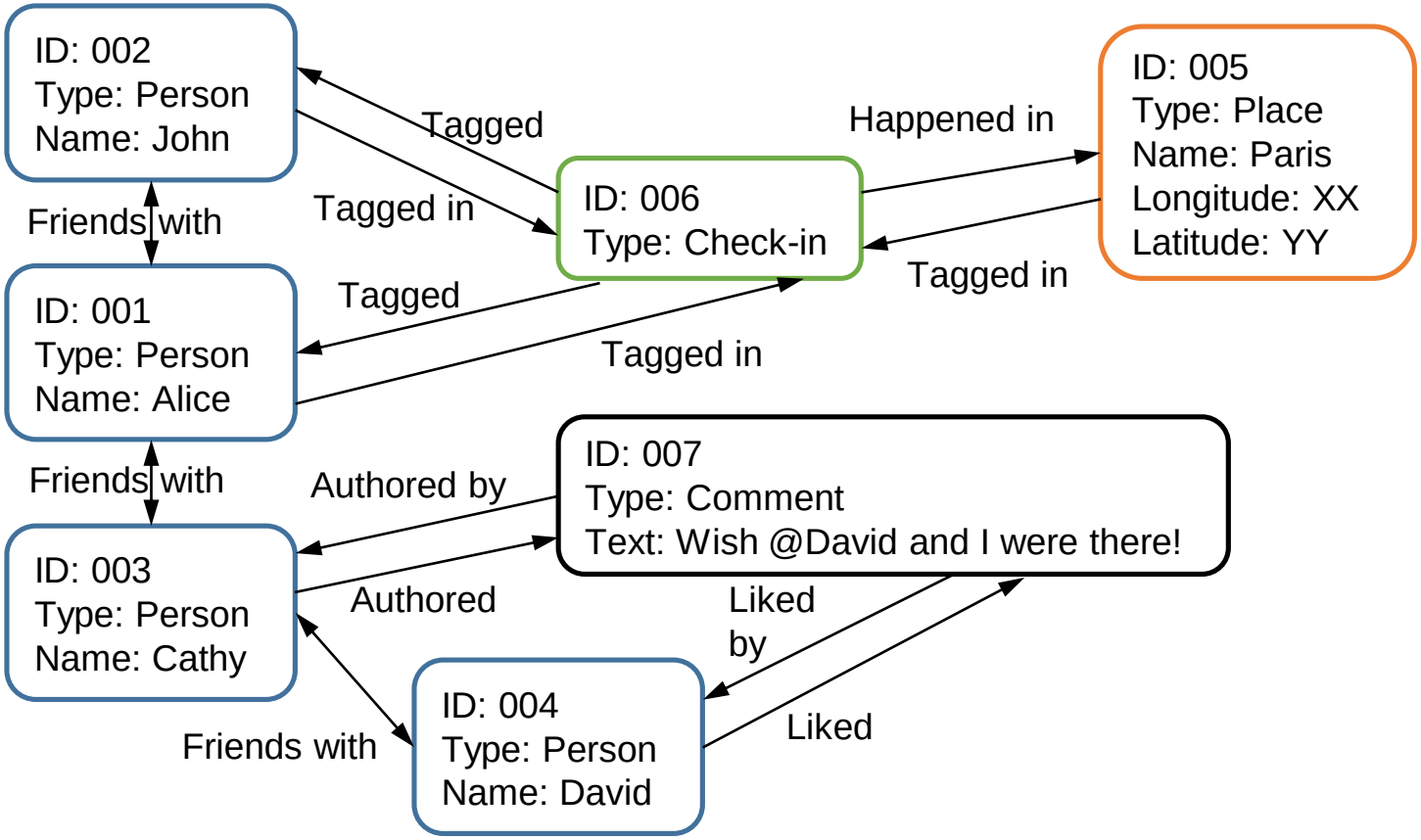
An example of a graph-like database for social media apps

@Alice checked in @Paris with @John.
Cathy: Wish @David and I were there!
David: Liked Cathy's comment.



An example of a graph-like database for social media apps

@Alice checked in @Paris with @John.
Cathy: Wish @David and I were there!
David: Liked Cathy's comment.



Ref: https://www.usenix.org/conference/atc13/technical-sessions/presentation/bronson

Relational data model vs. Graph-like data model

Relational data model	Graph-like data model
Relations/Tables	(Sub)graphs
Rows/Tuples	Nodes
Columns/Attributes	Properties
Constraints	Edges
Joins	Graph traversals

Ref: https://www.tutorialspoint.com/neo4j/neo4j_overview.htm

Sub-types of graph-like data model

- Property graph model
  - E.g., Neo4j by Neo4j Inc., Titan
- Triple-store model
  - E.g., RDF4J by Eclipse Foundation, AllegroGraph

Neo4j: The Cypher query language

```
CREATE
(NAmerica:Location {name:'North America', type:'continent'}),
(USA:Location      {name:'United States', type:'country'  }),
(Idaho:Location    {name:'Idaho',         type:'state'    }),
(Lucy:Person       {name:'Lucy' }),
(Idaho) -[:WITHIN]->  (USA)  -[:WITHIN]-> (NAmerica),
(Lucy)  -[:BORN_IN]-> (Idaho)
```

Vertex

Edge

Vertex: ID, properties i.e. (key, value) pairs, incoming and/or outgoing edges.
Edge: ID, head and tail vertices, a relationship label, properties i.e. (key, value) pairs.

Can edges have properties?

Neo4j: The Cypher query language

Vertex

Edge

```
CREATE
  (NAmerica:Location {name:'North America', type:'continent'}),
  (USA:Location      {name:'United States', type:'country'  }),
  (Idaho:Location    {name:'Idaho',         type:'state'    }),
  (Lucy:Person       {name:'Lucy' }),
  (Idaho) -[:WITHIN]->  (USA)  -[:WITHIN]-> (NAmerica),
  (Lucy)  -[:BORN_IN]-> (Idaho)
```

How does Cypher execute the following query?

```
MATCH
  (person) -[:BORN_IN]->  () -[:WITHIN*0..]-> (us:Location {name:'United States'}),
  (person) -[:LIVES_IN]-> () -[:WITHIN*0..]-> (eu:Location {name:'Europe'})
RETURN person.name
```

Neo4j: The Cypher query language

- A declarative query language (QL)
  - What is the difference between a ‘declarative’ QL and an ‘imperative’ QL?

- A cheat sheet for the Cypher query language: https://www.tutorialspoint.com/neo4j/neo4j_cql_introduction.htm

The triple-store model

- (subject, predicate, object)
- Subject = a vertex
- Object = another vertex or a value
- Predicate = an edge.

The Turtle data creation format

```
@prefix : <urn:example:>.
_:lucy a :Person.
_:lucy :name "Lucy".
_:lucy :bornIn _:idaho.
_:idaho a :Location.
_:idaho :name "Idaho".
_:idaho :type "state".
_:idaho :within _:usa.
_:usa a :Location.
_:usa :name "United States".
_:usa :type "country".
_:usa :within _:namerica.
_:namerica a :Location.
_:namerica :name "North America".
_:namerica :type "continent".
```

Querying with SPARQL (pronun. sparkle) in RDF4J

```
PREFIX : <urn:example:>

SELECT ?personName WHERE {
  ?person :name ?personName.
  ?person :bornIn  / :within* / :name "United States".
  ?person :livesIn / :within* / :name "Europe".
}
```

References

- M. KLEPPMANN (2017), Designing Data-Intensive Applications The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, O'Reilly.
  - Chapter 2. Data Models and Query Languages

- Paper: Bronson et al., "TAO: Facebook's Distributed Data Store for the Social Graph", 2013 USENIX Annual Technical Conference (USENIX ATC '13).
  - Video: https://www.usenix.org/conference/atc13/technical-sessions/presentation/bronson

Thank you