A Comparision of Current Graph Database Models

R. Angles

Department of Computer Science, Universidad de Talca

Presenter: Bill Zhang
https://qdata.github.io/deep2Read

Outline

- Graph databases are a type of NoSQL database
- First used in early 90's, but regaining popularity due to data which relies more on relations than entities
- Complex graph-like data becoming more common
- This paper outlines advantages and disadvantages of different graph database models

- Graph Database Model: Data structures for the schema and instances are modeled as graphs (or generalizations of graphs)
- Data manipulation expressed by graph-oriented operations and type constructors
- Introduce level of abstraction which more naturally model graph data
- Allow for direct querying on the graph structure

Current Graph Databases

Databases vs. Stores

- Graph databases must provide most of the major components of database management systems
  - External interfaces (UI or API)
  - Database languages (for data definition, querying, manipulating)
  - Query optimizer
  - Database, storage, and transaction engines
  - Management and operation features (tuning, backup, etc.)
- Graph stores just provide basic facilities for storing and querying graphs

- **AllegroGraph**: Precursor to others; now oriented for Semantic Web standards
- **DEX**: Java library for persistent and temporary graphs; based on bitmaps, good for large graphs
- **HyperGraphDB**: Hypergraph model where edge connects ≥ 2 nodes; good for higher-order relations

Current Graph Databases

Graph Databases

- **InfiniteGraph**: Supports large-scale graphs in distributed environment; efficient traversal of relations
- **Neo4j**: Network oriented model; relations are first class objects
- **Sones**: Inherent support for high-level abstraction (i.e. walks)

- **Filament**: Graph storage library with support for SQL
- **G-Store**: Storage manager for large vertex-labeled graphs
- **redis_graph**: Basic Python implementation for storing graphs
- **VertexDB**: Graph store on top of TokyoCabinet (B-tree Key/Value disk store)
- Prototypes: CloudGraph, Horton, Trinity
- Other standard databases use graph structures and algorithms as well

Comparison

Data Storing

- Look at data storage in main memory, external memory, and back-end storage
- Also implementation of indexes: basis to improve data retrieval ops
- External memory required for large amounts of data

DATA STORING FEATURES

Graph Database	Main memory	External memory	Backend Storage	Indexes
AllegroGraph	•	•		•
DEX	•	•		•
Filament	•		•	
G-Store		•		
HyperGraphDB	•	•	•	•
InfiniteGraph		•		•
Neo4j	•	•		•
Sones	•			•
vertexDB		•	•	

- Implementation of API/GUI
- Divide database language into three parts
  - Definition: Add, change, delete objects in schema
  - Manipulation: Insert, delete, update data
  - Query: Retrieval of data through queries

OPERATION AND MANIPULATION FEATURES

Graph Database	Data Definition Language	Data Manipulat. Language	Query Language	API	GUI
AllegroGraph	•	•	•	•	•
DEX				•	
Filament				•	
G-Store	•		•	•	
HyperGraphDB				•	
InfiniteGraph				•	
Neo4j				•	
Sones	•	•	•	•	•
vertexDB				•	

Comparison

Data Structures

- **Simple Graphs**: Simple flat graph with nodes connected by edges
- **Hypergraphs**: Edge can relate an arbitrary set of nodes
- **Nested Graphs**: Nodes can be graphs (hypernodes)
- **Attributed Graphs**: Nodes and edges have attributes describing their properties

GRAPH DATA STRUCTURES

	Graphs				Nodes		Edges		
Graph Database	Simple graphs	Hypergraphs	Nested graphs	Attributed graphs	Node labeled	Node attribution	Directed	Edge labeled	Edge attribution
AllegroGraph	•				•		•	•	
DEX				•	•	•	•	•	•
Filament	•				•		•	•	
G-Store	•				•		•	•	
HyperGraphDB		•			•		•	•	
InfiniteGraph				•	•	•	•	•	•
Neo4j				•	•	•	•	•	•
Sones		•		•	•	•	•	•	•
vertexDB	•				•		•	•	

- Look at ability to represent data
- Object nodes and relations have an ID for nodes and relations
- Complex nodes can represent complex entities like tuples or sets
- Complex relations include grouping, derivation, inheritance, etc.

Comparison

Representation of Entities and Relations

REPRESENTATION OF ENTITIES AND RELATIONS

	Schema			Instance					
Graph Database	Node types	Property types	Relation types	Object nodes	Value nodes	Complex nodes	Object relations	Simple relations	Complex relations
AllegroGraph					•			•	
DEX	•		•	•	•		•	•	
Filament					•			•	
G-Store					•			•	
HyperGraphDB	•		•		•			•	•
InfiniteGraph	•		•	•	•		•	•	
Neo4j				•	•		•	•	
Sones					•			•	•
vertexDB					•			•	

Comparison
Query Languages

- Most databases focus on retrieval

COMPARISON OF QUERY FACILITIES (● INDICATES SUPPORT, AND ○ PARTIAL SUPPORT)

	Type			Use		
Graph Database	Query Lang.	API	Graphical Q. L.	Retrieval	Reasoning	Analysis
AllegroGraph	○	●	●	●	●	●
DEX		●		●		●
Filament		●		●		
G-Store	●			●		
HyperGraphDB		●		●		
InfiniteGraph		●		●		
Neo4j	○	●		●		
Sones	●		●	●		●
vertexDB		●		●		

- **Type Checking**: Consistency of instance with schema definition
- **Node/edge Identity**: All entities/relations can be identified by a value or by its neighborhood
- **Referential Integrity**: Only existing entities are referenced
- **Cardinality Checking**: Verify uniqueness of properties/relations
- **Functional Dependency**: Test if one element is dependent on another
- **Graph Pattern Constraints**: Structural constraints (i.e. path constraints)

Comparison
Integrity Support

- Very little support because flexibility desired

COMPARISON OF INTEGRITY CONSTRAINTS

Graph Database	Types checking	Node/edge identity	Referential integrity	Cardinality checking	Functional dependency	Graph pattern constrains
DEX	•	•	•			
HyperGraphDB	•	•				
InfiniteGraph	•	•				
Sones		•		•		

Support for Querying Graphs

- **Adjacency Queries**: See if two nodes are adjacent or find neighborhood of nodes
- **Reachability Queries**: Find if there is a path between two nodes (fixed-length paths, simple paths, shortest path)
- **Pattern Matching Queries**: Find all sub-graphs that are isomorphic to pattern graph; NP-complete
- **Summarization Queries**: Summarize query results (i.e. aggregate operations)

Support for Querying Graphs

CURRENT GRAPH DATABASES AND THEIR SUPPORT FOR ESSENTIAL GRAPH QUERIES

	Adjacency		Reachability				
Graph Database	Node/edge adjacency	k-neighborhood	Fixed-length paths	Regular simple paths	Shortest path	Pattern matching	Summarization
Allegro	•		•			•	
DEX	•		•	•	•	•	
Filament	•		•			•	
G-Store	•		•	•	•	•	
HyperGraph	•					•	
Infinite	•		•	•	•	•	
Neo4j	•		•	•	•	•	
Sones	•					•	
vertexDB	•		•	•		•	

Conclusion

- Surveyed graph databases and compared them according to data modeling features
- Showed that most graph databases provide innate support for different graph structures, query facilities (APIs), query languages, and integrity constraints
- Defined set of essential graph queries and evaluated query facilities of these databases