Module 5: Graph Database

Text chapters 12, 13, & 14

University of Illinois
at Springfield

What is Graph Database

- Graphs are mathematical objects that consist of two parts: vertices and edges.
- Vertices represent things or entities such as cities, employees, proteins, or electrical circuits.
- Vertices have relationships with other vertices.
  - For example, employees manage other employees and cities are linked to other cities by highways.
- The links or connections between entities are represented by edges.

What is Graph Database

- Graphs are well suited to model networks and other objects that have relationships between objects.

- Graphs are used to model:
  - cities and highways,
  - the spread of infectious disease,
  - abstract organizational structures such as governments, and social media.

Graph Database Advantages

The following are advantages of graph databases:

- **Faster querying by avoiding joins**
- **Simplified modeling**
- **Support for multiple relations between entities**

Graph Database Advantages

- Graph databases are unique in the family of NoSQL databases.
- Like relational databases, a formal mathematical model underlies graph databases.
  - This mathematical basis is graph theory.
  - Mathematicians have developed many algorithms for analyzing graphs that might be of use to graph database applications.

Graph Database Terminology

- **Vertex: An entity marked with a unique identifier analogous to a row key in a column family database or a primary key in a relational database.**

- **Edge: Represents a relationship. Also known as a *link* or *arc*, an edge defines relationships between vertices or objects connecting vertices.**

Graph Database Terminology

- **Path: A set of vertices, along with the edges between those vertices.**
  - **The vertices in a graph are all different from each other.**
  - **If edges are directed, the path is a directed path.**
  - **If the graph is undirected, the paths in it are undirected paths.**
- **Loop: An edge that connects a vertex to itself.**
- **Union: The combined set of vertices and edges in a graph.**

Graph Database Terminology

- **Intersection: The set of vertices and edges that are common to both graphs.**
- **Graph traversal: The process of visiting all vertices in a graph in a particular way.**
- **Isomorphism: Two graphs are isomorphic if for each vertex in the first graph there is a corresponding vertex in the other graph.**

Graph Database Terminology

- **Order: The number of vertices in a graph**
- **Size: The number of edges in a graph.**
- **Degree: The number of edges linked to a vertex.**
- **Closeness: A property of a vertex that indicates how far the vertex is from all others in the graph.**

Graph Database Terminology

- **Betweenness: A measure of how much of a bottleneck a given vertex is.**
- **Undirected graph: A graph in which the edges are not directed.**
- **Directed graph: A graph in which the edges have a direction.**

Graph Database Terminology

- **Flow network: A directed graph in which each edge has a capacity and each vertex has a set of incoming and outgoing edges.**
- **Bipartite graph, or bigraph: A graph with two distinct sets of vertices where each vertex in one set is only connected to vertices in the other set.**
- **Multigraph: A graph with multiple edges between vertices.**

Graph Database Terminology

- **Weighted graph: A graph in which each edge has a number assigned to it.**
- **Dijkstra's algorithm: An algorithm used to find the shortest path in a network.**

Graph Database Design

Graph database applications frequently include queries and analysis that involve:

- **Identifying relations between two entities**
- **Identifying common properties of edges from a node**
- **Calculating aggregate properties of edges from a node**

Graph Database Design

The following are some example questions Graph Database applications try to answer:

- **How many hops (that is, edges) does it take to get from vertex A to vertex B?**
- **How many edges between vertex A and vertex B have a cost that is less than 100?**
- **How many edges are linked to vertex A?**
- **What is the centrality measure of vertex B?**
- **Is vertex C a bottleneck; that is, if vertex C is removed, will parts of the graph become disconnected?**

Graph Database Design

- **Queries drive graph data modeling.**
  - **Vertices should be the things one is interested in modeling as objects.**
  - **Edges should be relationships one is interested in modeling.**

Graph Database Design

Part of the process of designing graph databases and queries is mapping from graph-centric queries to domain-centric queries.

- **For example, a graph-centric query is: How many hops (that is, edges) does it take to get from vertex A to vertex B?**
- **The same query in domain-centric form is: How many follows relations are between Developer A and Developer B?**

Graph Database Design

The following are basic steps to getting started with graph database design:

- **Identify the queries you would like to perform.**
- **Identify entities in the graph.**
- **Identify relations between entities.**
- **Map domain-specific queries to more abstract queries so you can implement graph queries and use graph algorithms to compute additional properties of nodes.**

Graph Database Design

There are two commonly used graph query methods:

- The Cypher query language provides a declarative, SQL-like language for building queries.
  - Cypher is used with the Neo4j graph database.
- Gremlin is a graph traversal language that works with several different graph databases.
  - Graphs can be traversed in depth first or breadth first order.
- Graph databases support indexing and should be used to improve query response time.

Graph Database Design

- **Edges may be directed or undirected. The correct choice is determined by the domain and queries.**
  - **In directed graphs, care should be taken to account for cycles.**

- **Graph databases have some unusual characteristics. Algorithms that run in reasonable time on small graphs can take a surprisingly long time with moderate or large graphs.**

Graph Database Design

- When designing graphs, try to think of all entities (nouns) used in the domain.
  - It helps to consider all possible combinations of entities having relations when the number of entity types is small.
  - Some of the combinations might not be relevant to the types of queries you will pose; they can be eliminated from consideration.
- This process helps reduce the chance that you miss a relevant relation early in the design phase.
  - As the number of entities grows, you might want to focus on combinations that are reasonably likely to support your queries.