# Introduction to the Graph Data Model

## 1. What is a Graph Database?

A **Graph Database** is a type of NoSQL database that uses a **graph structure** to represent and store data.

### Key Components:

- **Nodes (Vertices):** Represent entities (e.g., people, locations, objects).
- **Edges (Relationships):** Connect nodes and define relationships between them.
- **Properties:** Attributes associated with nodes and edges (e.g., name, age, weight).

### Graph Database Capabilities

- Supports **graph-oriented queries** such as:
    - **Traversals** (finding paths between nodes).
    - **Shortest path calculations**.
    - **Community detection and centrality analysis**.

## 2. Where Do Graphs Show Up?

Graphs are widely used across various domains:

| Domain | Example Use Case |
|---|---|
| **Social Networks** | Modeling connections on platforms like Instagram and LinkedIn. |
| **The Web** | The internet itself is a giant graph of **webpages (nodes)** linked via **hyperlinks (edges)**. |
| **Biological & Chemical Systems** | Used in genetics and chemistry to model molecule interactions. |
| **Supply Chains & Logistics** | Optimizing transportation and delivery routes. |

# 3. Basics of Graph Theory

## Labeled Property Graph

A **Labeled Property Graph** consists of:

- **Nodes** grouped by labels (e.g., "Person," "Car").
- **Edges (Relationships)** that connect nodes.
- **Properties** (key-value pairs) that store additional information.

**Key Rules:**

- Nodes **can exist without relationships**.
- **Edges must always connect two nodes**.

## Example Graph Structure

Labels: **Person, Car**
Relationships: **Drives, Owns, Lives_with, Married_to**
Properties: **Name, Age, Car Model, Relationship Status**

# 4. Paths in Graphs

- A **path** is an ordered sequence of nodes connected by edges.
- A valid path **does not repeat nodes or edges**.
- Example of a **valid path**:
  $1 \rightarrow 2 \rightarrow 6 \rightarrow 5$
- **Invalid path (cycle detected):**
  $1 \rightarrow 2 \rightarrow 6 \rightarrow 2 \rightarrow 3$

# 5. Types of Graphs

| Graph Type | Description |
|---|---|
| **Connected vs. Disconnected** | A graph is **connected** if every node can be reached from any other node. If not, it is **disconnected**. |
| **Weighted vs. Unweighted** | A **weighted graph** has edges with numerical weights (e.g., distances in a map). An **unweighted graph** treats all edges equally. |

| | |
|---|---|
| **Directed vs. Undirected** | A **directed graph** has edges with a **defined direction** (e.g., Twitter follows). An **undirected graph** has **bidirectional** edges (e.g., Facebook friendships). |
| **Cyclic vs. Acyclic** | A **cyclic graph** contains loops (e.g., a city road system). An **acyclic graph** has no cycles (e.g., a family tree). |
| **Sparse vs. Dense** | A **sparse graph** has relatively few edges compared to nodes. A **dense graph** has many edges relative to nodes. |

# 6. Graph Algorithms

## Pathfinding Algorithms

- Used to **find the shortest path** between nodes.
- Example: **Google Maps uses pathfinding to optimize routes.**

| Algorithm | Purpose |
|---|---|
| **Breadth-First Search (BFS)** | Finds the shortest path in **unweighted** graphs. |
| **Depth-First Search (DFS)** | Explores deeper paths before backtracking. |
| **Dijkstra's Algorithm** | Finds the shortest path in **positively weighted** graphs. |
| *A Algorithm** | Similar to Dijkstra's but uses **heuristics** to optimize traversal. |
| **Minimum Spanning Tree** | Finds the lowest-cost way to connect all nodes. |

## Centrality & Community Detection

Used to analyze **importance and influence** in networks.

- **Centrality Analysis**:
  - Determines **which nodes are most important** (e.g., social media influencers).
- **Community Detection**:
  - Identifies **clusters** of tightly connected nodes.

**Real-World Example:**
**Google's PageRank Algorithm** evaluates webpage importance based on incoming links.

# 7. Graph Databases

## Neo4j - A Popular Graph Database

- A **NoSQL database** optimized for graph-based data.
- Supports **both transactional and analytical** graph queries.
- **Schema-optional** (allows flexibility in data structure).
- **ACID-compliant** (ensures data consistency).
- **Distributed computing support** for handling large datasets.

**Similar Graph Databases:**

- **Microsoft CosmosDB**
- **Amazon Neptune**