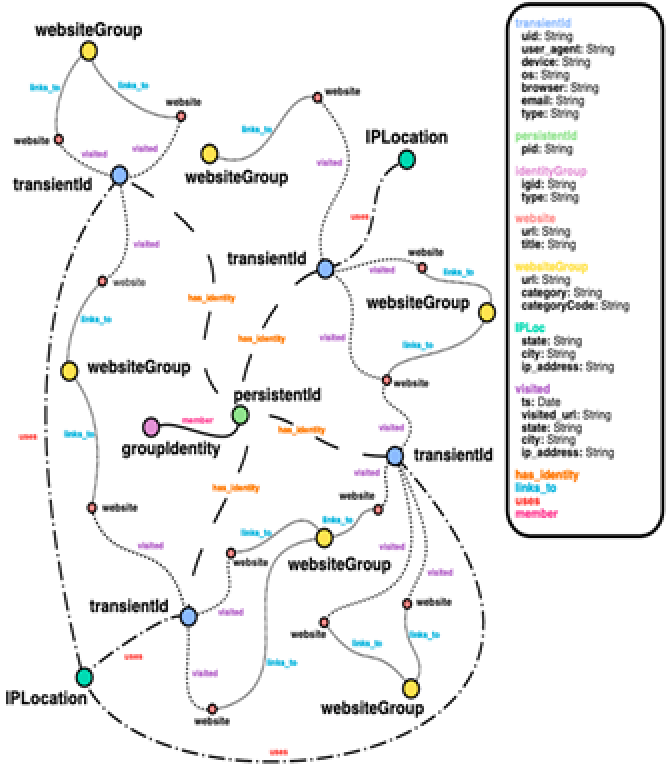
Graph Databases

Group A6 - Sarah Al Bastaki, Jonathan Hopkins, Tia Kungwani, Oruaroghene Mukoro, Aneesh Seemakurthy

Introduction - Graph Databases

- **What are graph databases?**
  - Graph databases are specially designed to **store and formalize the relationships** between data points using nodes, edges, and properties
    - Nodes: the **specific entities** in the graph (i.e. people or accounts)
    - Edges: **relationships** between nodes (represented as links)
    - Properties: **attributes** that are attached to nodes/edges
  - We explore graph databases for real-world use through a process called **traversal**, which is fast and efficient
- **How are they used in a business context?**
  - They are used to manage **highly connected data and capture complex relationships**. They help us:
    - Query vast relationship networks efficiently **in real-time** (useful fo fraud detection)
    - Understand **subtle patterns and connections** in traditional databases
    - Retrieve results from any relationship queries **incredibly fast**
    - **Generate recommendations** from data (think Netflix or Tiktok)
- **Why use them?**
  - Graph databases **excel with relationship-heavy data** unlike traditional, relational databases
  - There is an **increasing focus on analyzing connected data** as it is useful for financial services, supply chains, social networks, etc.



(https://aws.amazon.com/neptune/)

(Information obtained from aws.amazon.com & neo4j.com)

	Supported Models	Supported Query Languages	Hosting	Scalability	Cost
Neo4j	Property graphs	Cypher	self-hosted or managed w/ AuraDB	vertical, horizontal via sharding (*composite databases*)	free tier or monthly subscription
Amazon Neptune	Property & RDF graphs	Gremlin or openCypher (property graphs), or SPARQL (RDF graphs)	managed by AWS	vertical for writes, replication for reads, automatic scaling	on demand, by the hour
ArangoDB	Property graphs, documents, key-value	AQL	self-hosted or managed w/ ArangoGraph	horizontal (sharding + SmartGraphs)	on demand, by the hour

Technical Details - Amazon Neptune

- Fully managed, cloud-native graph database service by AWS, designed for optimally storing and querying large-scale, highly connected datasets
- Supports both property graphs (Gremlin & OpenCypher) and RDF graphs (SPARQL)
- Architecture:
  - Superior scalability and availability
  - Multi-AZ architecture for fault tolerance and data replication, supports up to 15 read replicas
  - ACID-compliant to ensure strong data integrity
  - Log-structured storage with in-point recovery
- Typical Use Cases:
  - Knowledge Graphs
  - Identity and Access Management
  - Enterprise Data Integration
- Key Differentiators:
  - High scalability + cloud-native design
  - Supports auto-scaling horizontally
  - Multi-model support
  - Seamless integration with other AWS services

(Information obtained from https://aws.amazon.com/neptune/, https://docs.aws.amazon.com/neptune/latest/userguide/migration-architectural-differences.html)

Technical Details - Neo4j

- Leading native graph database management system designed to efficiently store and process large-scale connected data
- Represents data as a network of nodes and edges representing relationships between them
- Supports Cypher query language for graph traversal + library of graph algorithms
- Architecture:
  - High performance and reliability
  - Supports schema-based indexes and in-memory caching to significantly speed up query execution
  - ACID-compliant to ensure strong data integrity
  - Multiple deployment options
- Typical Use Cases:
  - Social Networks
  - Recommendation Systems
  - Fraud Detection
- Key Differentiators:
  - Highly optimized query execution
  - Powerful graph algorithms
  - Supports a dynamic schema, allowing model to represent evolving relationships

(Information obtained from https://aws.amazon.com/neptune/, https://www.analyticsvidhya.com/blog/2024/08/neo4j-vs-amazon-neptune/)

Sample Application 1: PayPal Fraud Detection

Graph databases provide an efficient way to detect fraudulent activities by analyzing relationships and identifying suspicious patterns. PayPal uses Neo4j to analyze relationships between different entities (users, transactions, IP addresses, devices, credit cards, etc.).

1. **Quickly identify fraud rings**
   a. Fraudsters often create multiple fake accounts linked by common elements. A graph-based model connects all these elements and reveals fraud rings in real-time.

2. **Analyze historical transaction patterns**
   a. By applying graph algorithms, it identifies users with unusually high levels of interconnected transactions, signaling potential fraud.

3. **Perform real-time fraud prevention**
   a. With graph traversal algorithms, PayPal can block suspicious transactions instantly before they are completed.

(logos-world.net)

(Information obtained from medium.com & developer.paypal.com)

Sample Application 2: LinkedIn Recommendation Engine

LinkedIn's platform thrives on meaningful connections between professionals, offering recommendations for people, jobs, and companies. With billions of relationships, LinkedIn needed a solution that could quickly traverse networks of professional connections.

1. **People You May Know**
   a. Instead of performing costly table joins in a relational database, a graph database efficiently traverses relationships to identify the shortest paths between people.

2. **Job Recommendations**
   a. Using graph-based similarity algorithms, LinkedIn identifies users with similar career paths and recommends jobs based on patterns from successful candidates.

3. **Content and Learning Recommendations**
   a. Instead of relying solely on direct user preferences, graph algorithms infer interests based on connections, industry trends, and behavioral patterns.

(vecteezy.com)

(Information obtained from linkedin.com & infoq.com)

Popularity Rankings

Why Neo4j?

- **First Mover Advantage**
  - Was one of the first graph databases to emerge, giving it a significant head start in establishing itself as a leader in the graph database market
- **ACID Compliant**
  - Atomicity, Consistency, Isolation, Durability
- **Graph-Native Database**
  - Built from the ground up graph based, resulting in faster queries
- **Cache Sharding**
  - Bouncing requests to other instances based on a hashing algorithm to increase the chances a value is hit in a cache. This improves read speeds
  - Actual sharding is quite hard for graph databases, NP Problem, it is generally done using Vertex Cut or Edge Cut. Vertices/Edges are broken and distributed
  - Some alternatives like Dgraph shard by RDF predicates
- **Large feature set**
  - Many graph algorithms are implemented for users.
  - Graph visualization
- **Open source**

2025	DBMS	Model Type(s)	Popularity Score
1.	**Neo4j**	Graph	46.15
2.	**Microsoft Azure Cosmos DB**	Multi-model	22.27
3.	**Aerospike**	Multi-model	5.22
4.	**Virtuoso**	Multi-model	3.20
5.	**ArangoDB**	Multi-model	2.87
6.	**OrientDB**	Multi-model	2.72
7.	**GraphDB**	Multi-model	2.68
8.	**Memgraph**	Graph	2.61
9.	**Amazon Neptune**	Multi-model	2.07
10.	**JanusGraph**	Graph	1.73

https://www.packtpub.com/en-ar/learning/tech-guides/neo4j-most-popular-graph-database?srsltid=AfmBOoonUv3C-pgvsGFTQmsv1luDF6D1-jdq6167XvvIzw1IEUdima1p
https://dgraph.io/blog/post/db-sharding/

Revenues & Projections

- **Graph Database Market is currently estimated to be worth $3.01 billion (2023)**
  - 40% from the US Market (Google, Datastax, Marriott, Verizon etc.)
- **Quite small when compared to the $70.76 billion valuation that relational databases have. Why is that?**
  - May break company wide paradigms
  - Lack of Visualization
  - Require new query language (Gremlin, Cypher etc.) which is expensive to train
  - Lack of standardization across graph database services
  - Speed gains are most times marginal. The strength of these databases is in recursive queries which are often avoided by well designed systems and/or are infrequent
  - Little benefit from parallelism
  - Takes up lots of RAM
  - More expensive than alternatives
  - Some consumers opt to create their own databases (Facebook TAO).
- **Projected to be worth $10.9 billion by 2032 an approximate 2x increase**
  - Growing need for connected data analysis
  - Creating and storing Knowledge graphs (especially in this AI boom)
  - Increased adoption by HealthCare providers for patient history over varying systems
  - Cloud based graph database services help abstract some of its pitfalls

https://www.marketresearchfuture.com/reports/graph-database-market-21397
https://engineering.fb.com/2013/06/25/core-infra/tao-the-power-of-the-graph/
https://www.marketsandmarkets.com/Market-Reports/graph-database-market-126230231.html
https://www.youtube.com/watch?v=aDoorU4X6Jk

Marketing Strategies

- Unique value gained through handling complex relationships in large datasets, highlighting use cases like
  - social media,
  - recommendation engines
  - fraud detection,
- Efficient relationship retrieval and augmentation
- Can be used in conjunction with other DBMS, e.g DataLakes as a way to manage data insights, which helps with Knowledge graph creation, expansion and maintenance

Future Trends

- **Emerging Developments:**
  - Integrating machine learning systems into graph analytics to **improve the ability to predict** based on relationships
  - Connecting massive graph databases together to create **federated graph systems**; conducive to creating an interconnected world and analyzing more larger scale relationships
  - Growing adoption of a standardized **Graph Query Language (GQL)** as a means of accessing relationships in these graph databases
- **Industry Evolution:**
  - Cloud-native graph databases are gaining traction as they **manage infrastructure complexity** without organizations needing to
  - Modern business are requiring **real-time graph processing** to take in streaming data and immediately update their graph relationships, all the while maintaining query performance
  - Traditional databases providers like Oracle and PostgreSQL are creating **graph extensions** to make their capabilities available to companies without having to change their infrastructure
- **Research Directions and Challenges:**
  - Currently tackling how to efficiently handle **trillion-edge graphs**, which requires better distributed storage systems, partitioning algorithms, and memory optimization
  - Attempting to make **graph traversal more efficient for complex queries** by using an adaptive query processing technique that incorporates learned patterns and caching strategies
  - Determining how to allow visualization tools to **better support complex graphs**; primarily exploring hierarchical aggregation or context-aware visualization
  - Researching temporal graphs and how to capture how **properties change over time**

(Information obtained from neo4j.com & dbta.com)