

Isaac Araya Solano | Carnet: 2018151703

## Resumen #5

# Graph Databases for Beginners

---

## Chapter 1: Graphs are the Future

Graph databases offer significant benefits over traditional databases due to their superior performance, flexibility, and agility. Graph databases maintain consistent performance even as data grows. The structure and schema of graph data models are flexible and can evolve alongside business requirements. Graph databases are intuitive to understand, as they are composed of nodes representing entities and relationships representing how nodes are associated. Twitter is an example of a graph database connecting millions of users. Understanding the basics of nodes and relationships is key to understanding graph databases.

Graph databases are a type of database management system that prioritize relationships between data points over other factors. This makes them particularly useful for applications that involve large amounts of data with complex relationships. Graph databases are more flexible and agile than traditional relational databases, and can handle growing data volumes without sacrificing performance.

A graph is composed of nodes, which represent entities, and relationships, which represent how two nodes are associated. Graph databases use graph storage and a graph processing engine to manage data. Native graph processing is the most efficient means of processing data, as connected nodes physically point to each other in the database. Neo4j is a leader in the industry when it comes to both graph storage and processing.

## Chapter 2: Why Data Relationships Matter

Relational databases are good at organizing tabular data, but not great at handling data relationships that change frequently. Their inflexible schema makes it difficult to adapt to evolving business needs, and attempts to compensate for this can lead to performance issues and more complex code. If you have a database schema, at a certain point, your business needs will entirely outgrow your current database schema. The problem, however, is that migrating your data to a new schema becomes incredibly effort-intensive.

### Why Other NoSQL Databases Don't Fix the Problem Either

While NoSQL databases may offer a performance advantage over relational databases due to their disconnected construction, they also face challenges in properly handling data relationships. Some developers try to add data relationships to NoSQL databases by embedding aggregate identifying information within the field of another aggregate using foreign keys. However, this approach can become just as prohibitively expensive as in a relational database when joining aggregates at the application level later. Additionally, foreign keys only "point" in one direction, which can make reciprocal queries too time-consuming to run.

In contrast, graph databases store data relationships as relationships, making them a better solution for cohesive big data analysis, including the connections between elements. Graph databases' explicit storage of relationship data reduces disconnects between evolving schema and the actual database, allowing for more flexibility and simpler data migration. With data relationships at their center, graph databases are highly efficient for query speeds, even for deep and complex queries. As a result, graph databases are becoming

increasingly popular for a wide range of applications, including logistics route optimization, retail suggestion engines, fraud detection, and social network monitoring.

## **Chapter 3: Data Modeling Basics**

### **What is Modeling Exactly?**

Data modeling is the process of mapping business and user needs to a structure for storing and organizing data. With traditional relational databases, this process can be complex and often results in a final database that looks nothing like the original plan. In contrast, modeling for a graph database is simpler, as the initial whiteboard structure of circles and boxes connected by arrows is already a graph. Creating a graph database from this model only requires a few lines of code.

### **A Relational Vs. Graph Data Modeling Match-Up**

In the data center management domain, creating a data model with a relational database involves convening with business leaders, subject-matter experts, and system architects to create a data model that maps to tables and relations. However, this approach adds complexity and often results in JOIN tables, which slow down queries. In contrast, a graph data modeling approach starts with a whiteboard sketch of the data model and enriches it according to business and user needs. This approach results in a graph database that includes labels, attributes, and relationships, without the added complexity of JOIN tables.

### **Why Data Modeling Isn't a One-Off Activity**

The article discusses the differences between relational and graph data modeling approaches. While relational databases require a complex modeling process that often leads to added complexity and slower query speed, graph databases offer a more agile approach that allows for quick adaptation to changing business and user needs. This makes graph databases a better choice for applications that require constant evolution and flexibility.

## **Chapter 4: Data Modeling Pitfalls to Avoid**

The text explains how a weak data model for fraud detection in email communications can be improved by adding nodes to represent each email and new relationships to track their attributes and interactions. This results in a more comprehensive graph model that allows us to track the entire communication chain for a given email thread, including any replies or forwards. By building a strong data model, we can gain deep insights into the data and identify patterns or trends that might be missed with a weaker model.