

Introduction to Neo4j

Defne Yıldırım
150230727

[Github Repo](#)

The Challenges of Traditional Databases (SQL)

- The SQL Approach: Data is stored in tables (rows & columns).
- The Challenge: Querying deep relationships (e.g., "friend-of-a-friend").
- Massive, complex "JOIN Monster" queries
- Disastrous performance drops
- Rigid Schema

The Solution: Neo4j

- Native Graph Database
- Core Philosophy: "Relationships are First-Class Citizens"



This means:

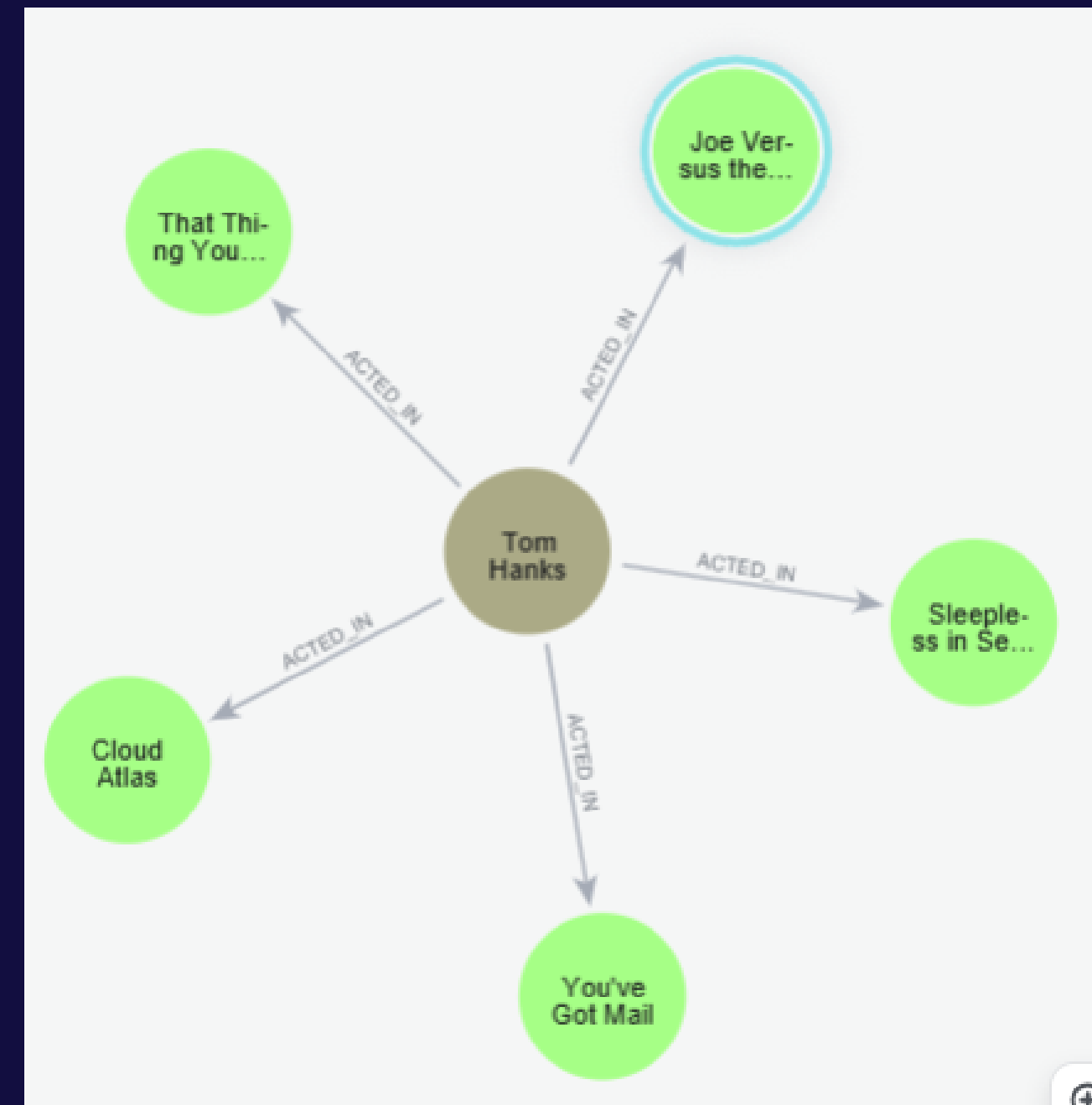
Data is stored as a graph (not in tables).

Connections are physical pointers, not computed at query time.

Optimized for high-speed, JOIN-free queries.

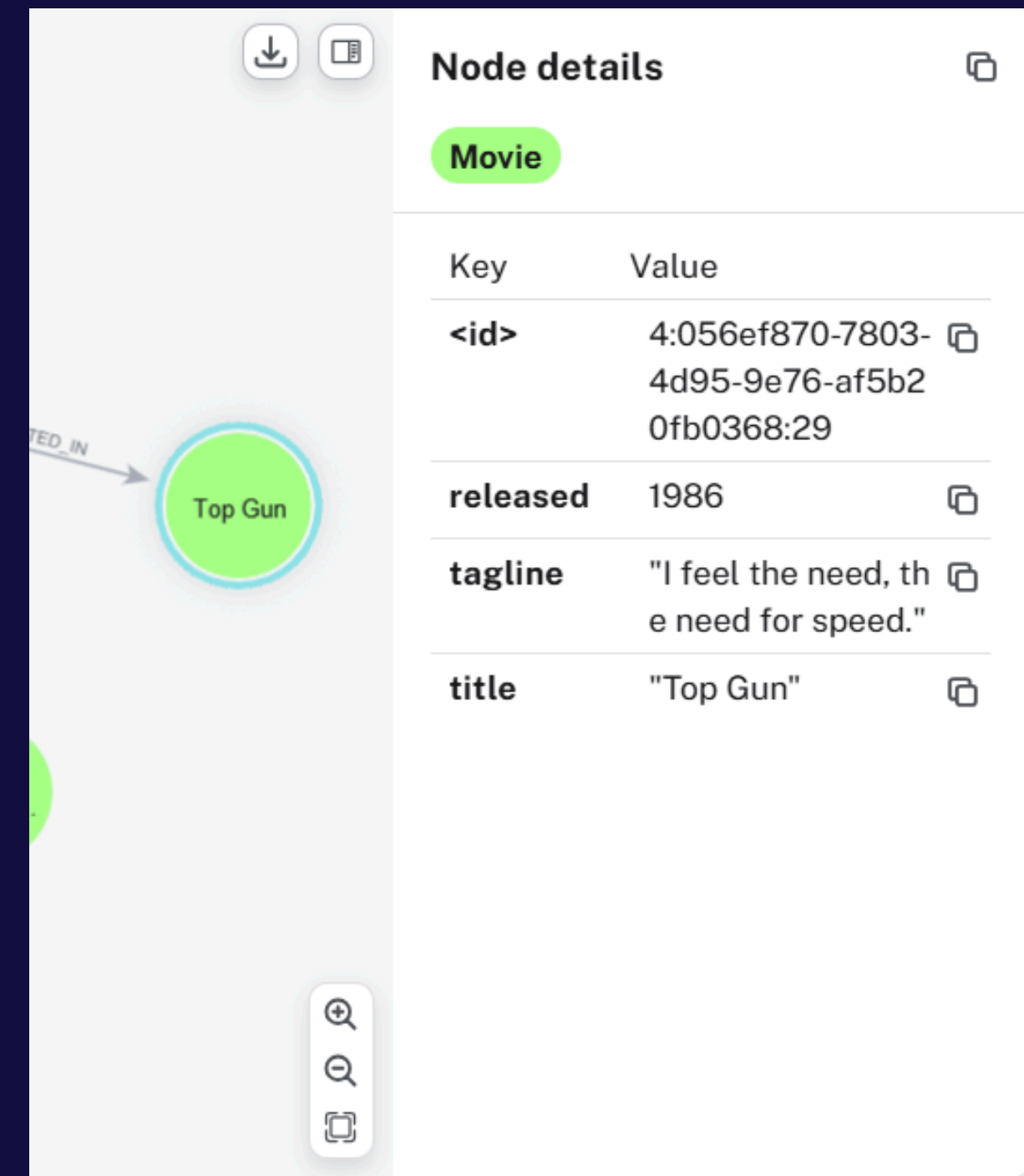
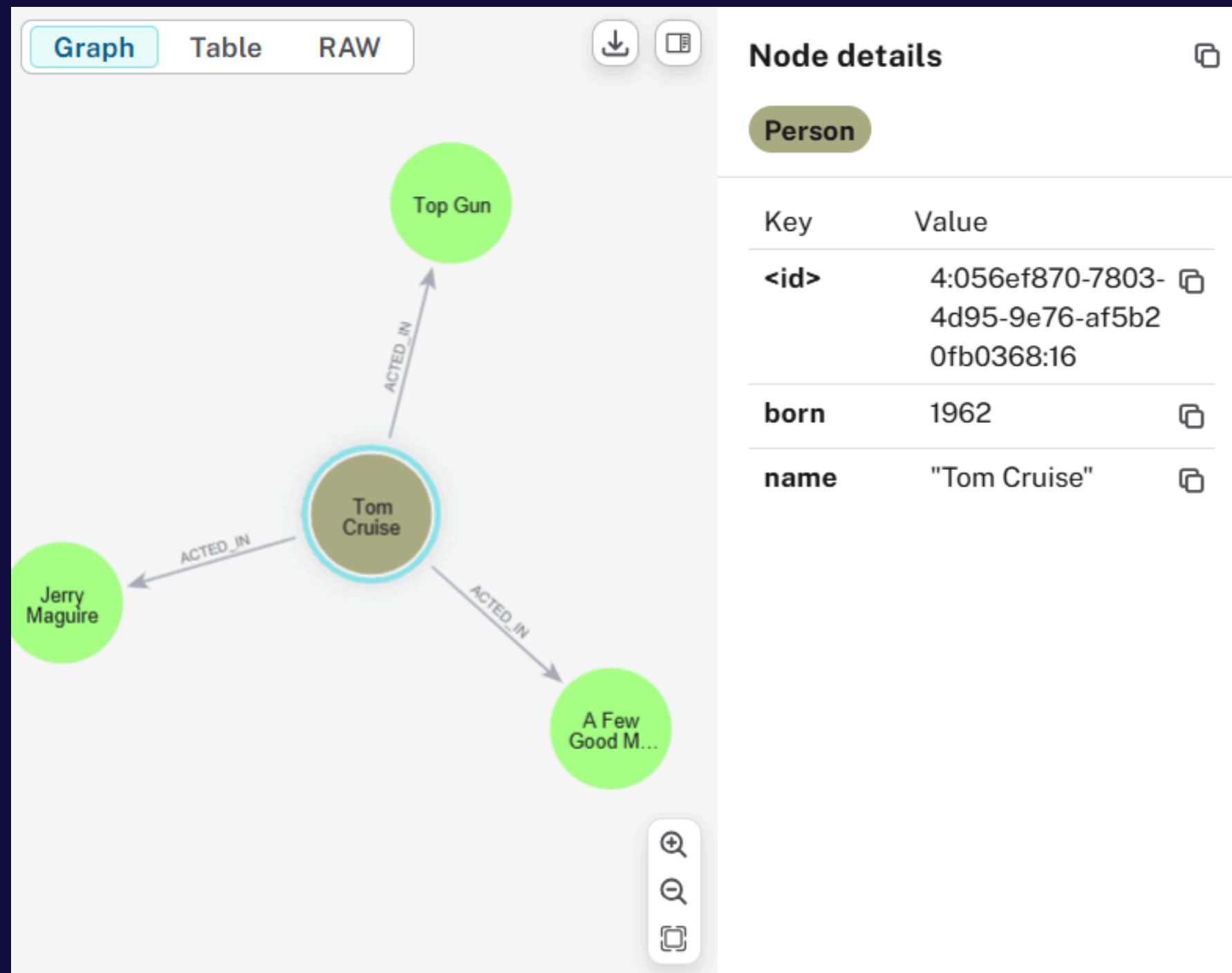
The Building Blocks: Property Graph Model

- Nodes: The entities or "dots".
- Relationships: Connect nodes. They have direction and type.
- Properties: Key-value data on both nodes and relationships.
{name: 'Ali'}, {date: '2025-11-11'}



The Building Blocks: Property

Graph Model



The Query Language: Cypher

The "SQL for Graphs"

- (and) = Nodes (The "dots")
 - -[]-> = Relationships (The "arrows")
- Declarative
 - Visual
 - Intuitive

```
j$ MATCH (p:Person {name: "Tom Hanks"}) RETURN p
```



```
1 MATCH (p:Person {name: 'Tom Hanks'})-[r:ACTED_IN]->(m:Movie)
2 RETURN p, r, m
```



```
1 MATCH (m:Movie) WHERE m.released >= 1990 AND m.released < 2000
2 RETURN m.title, m.released
```



neo4j\$ MATCH path=shortestPath((:Person {name:"Kevin Bac

GraphTableRAW

```
graph LR; TC((Tom Cruise)) -- ACTED_IN --> AGM((A Few Good M...)); AGM -- ACTED_IN --> KB((Kevin Bacon)); TC -- ACTED_IN --> TG((Top Gun)); TG -- ACTED_IN --> MR((Meg Ryan));
```

Results overview

Nodes (5)
* (5) Movie (2) Person (3)

Relationships (4)
* (4) ACTED_IN (4)

neo4j\$ MATCH path = (p:Person {name: 'Tom Cruise'})-[*1..

GraphTableRAW

```
graph LR; TC((Tom Cruise)) -- ACTED_IN --> AGM((A Few Good M...)); AGM -- ACTED_IN --> KB((Kevin Bacon)); TC -- ACTED_IN --> TG((Top Gun)); TG -- ACTED_IN --> MR((Meg Ryan));
```

Node details

Person

Key	Value
<id>	4:056ef870-7803-4d95-9e76-af5b20fb0368:16
born	1962
name	"Tom Cruise"

Real-World Use Cases

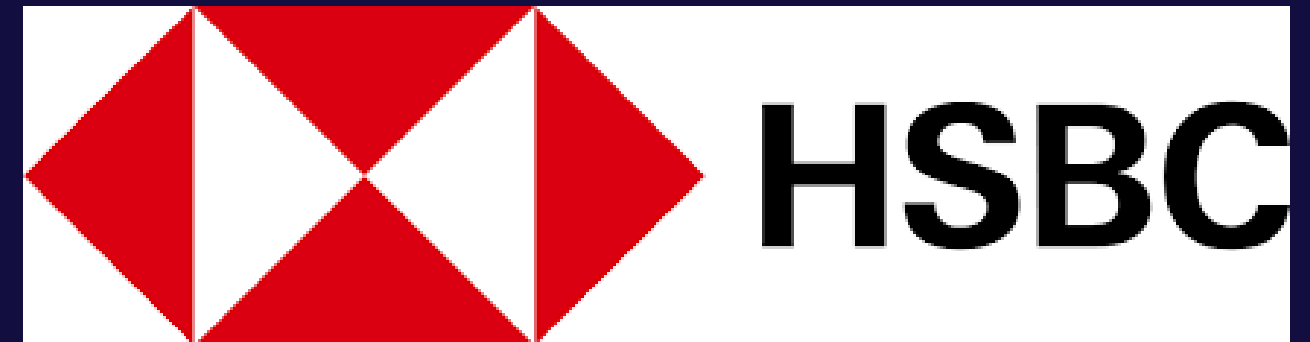
1. Recommendation Engines

Query: "People who bought this, also bought..."



2. Fraud Detection

Query: "Does this transaction look like a known fraud ring?"



3. Social Networks

Query: "Who are my 'mutual friends'?"



**THANK
YOU!**

