AT82.02

DATA MODELING AND MANAGEMENT

UNIT 2-4: GRAPH MODEL

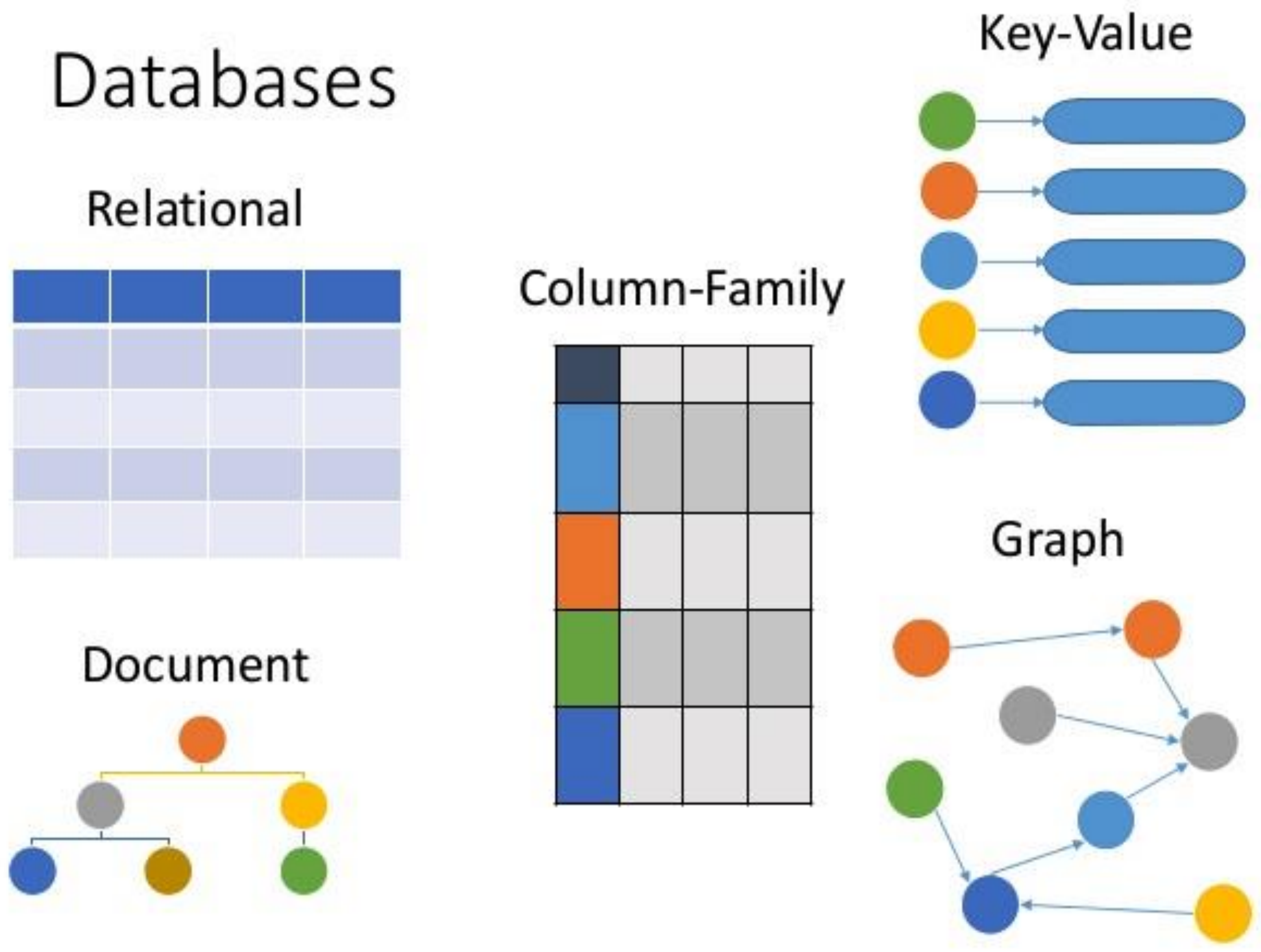
CHUTIPORN ANUTARIYA (CHUTI AT AIT DOT AC DOT TH)

Co-funded by the Erasmus+ Programme of the European Union

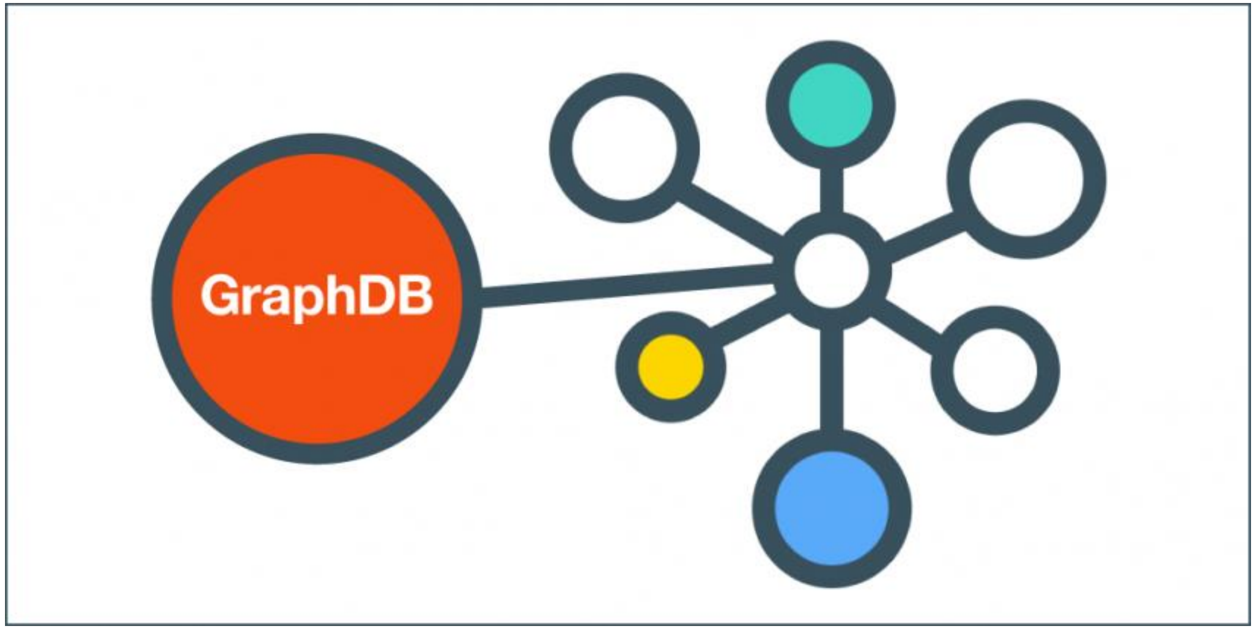
DS&AI

Database Family

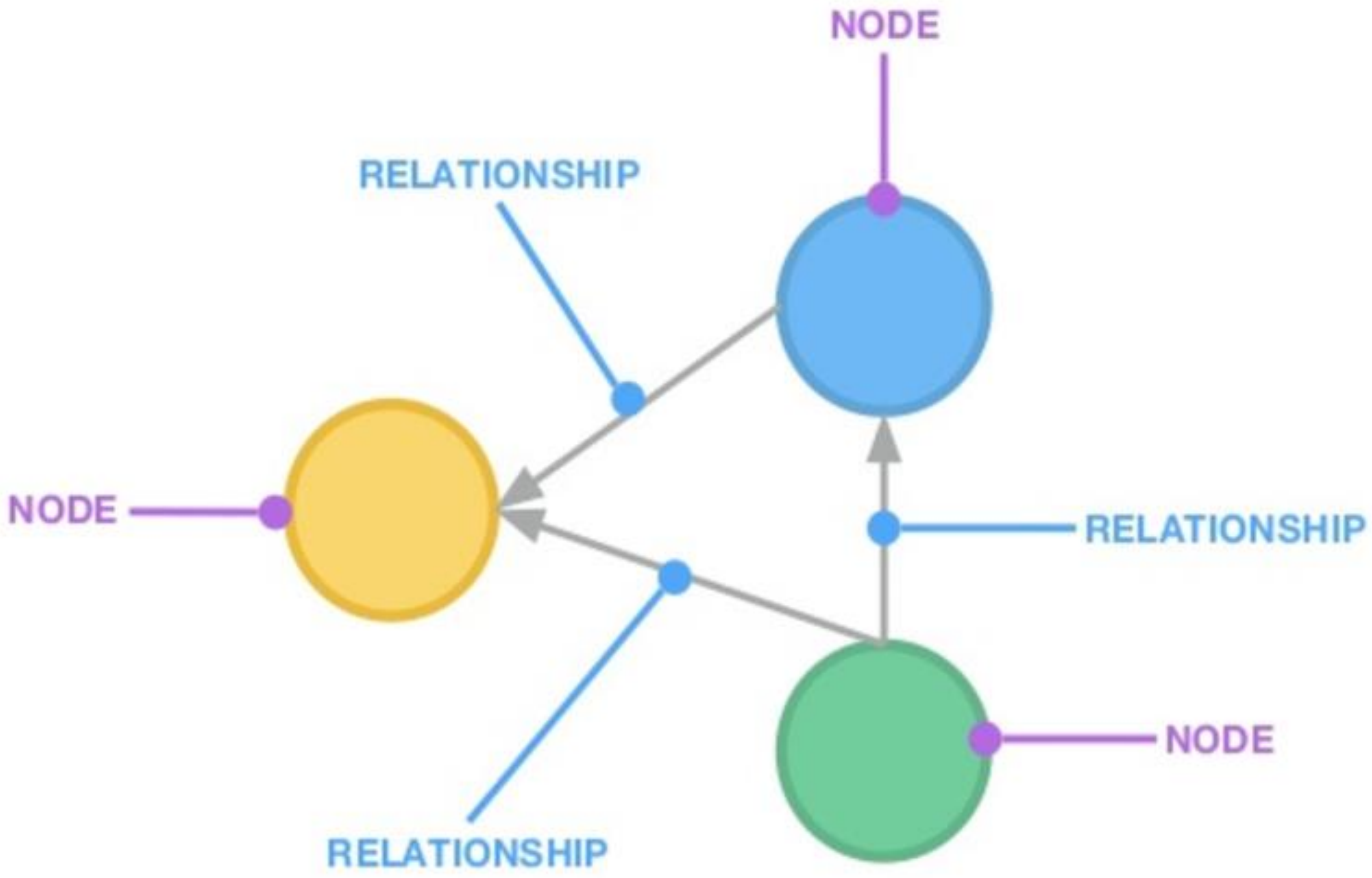
RECAP



Graph Model



A Graph



Graph Model

Graph store uses graph structures for semantic queries with nodes, edges and properties to represent and store data.

The relationships allow data in the store to be linked together directly, and in many cases retrieved with one operation.

A query on a graph is known as traversing the graph.

The biggest advantage of the graph store is that joins are not necessary.

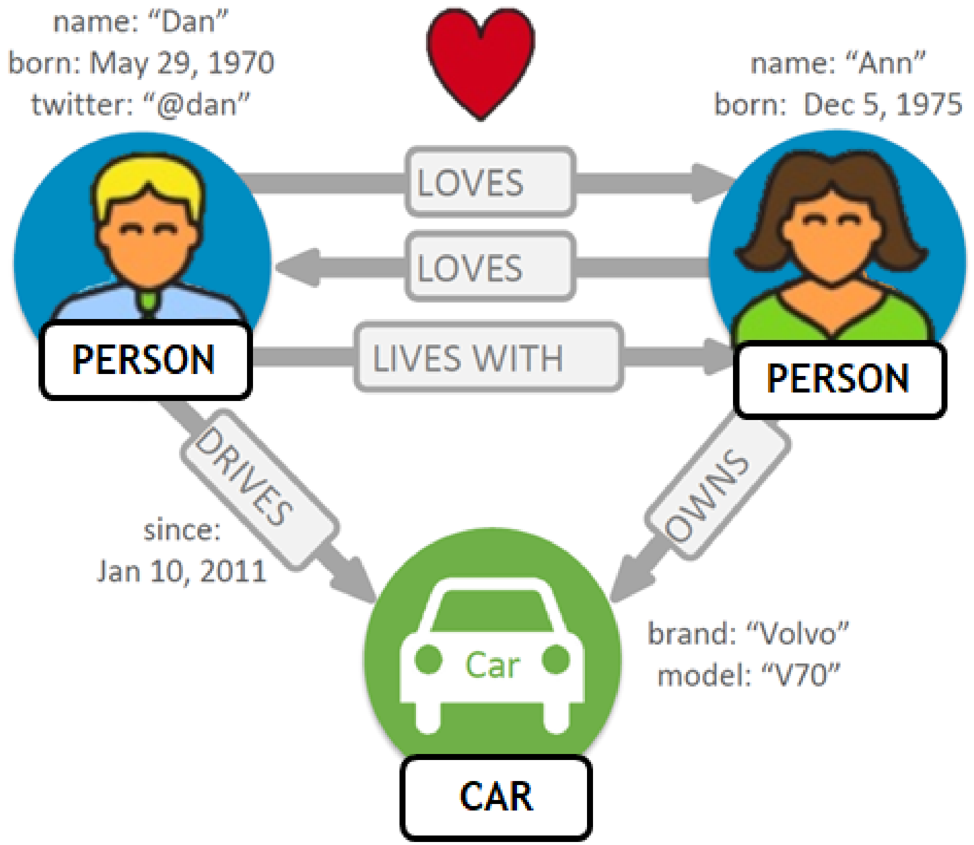
Graph Model: Property

Nodes
- Represent the objects in the graph
- Can be *labeled*

Relationships
- Relate nodes by *type* and *direction*

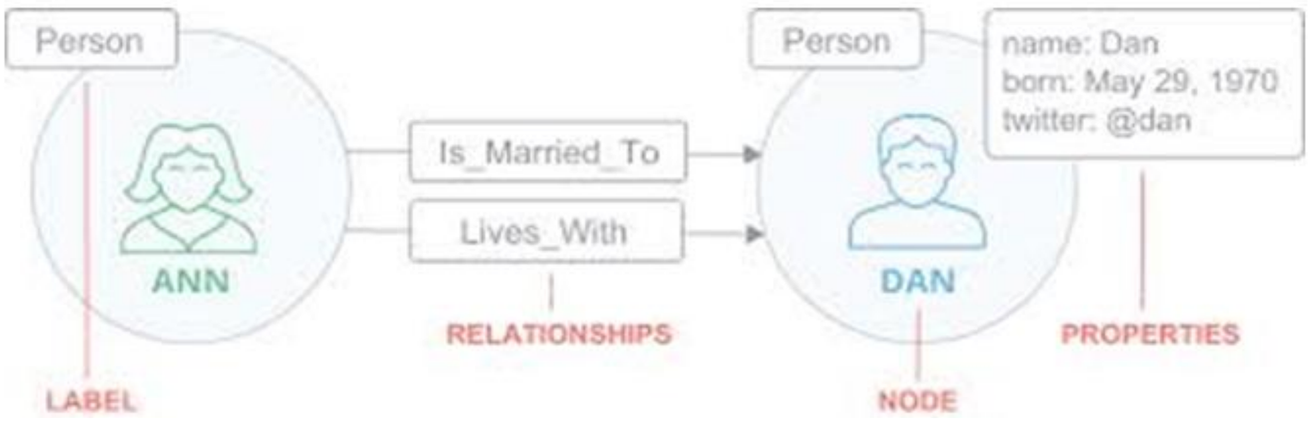
Properties
- Name-value pairs that can go on nodes and relationships.

Label
- Associate a set of nodes.
- A node can have zero or more labels
- Labels do not have any properties



Summary: Graph Model Property

- **Nodes** - Entities and complex value types
- **Relationships** - Connect entities and structure domain
- **Properties** - Entity attributes, relationship qualities, metadata
- **Labels** - Group nodes by role



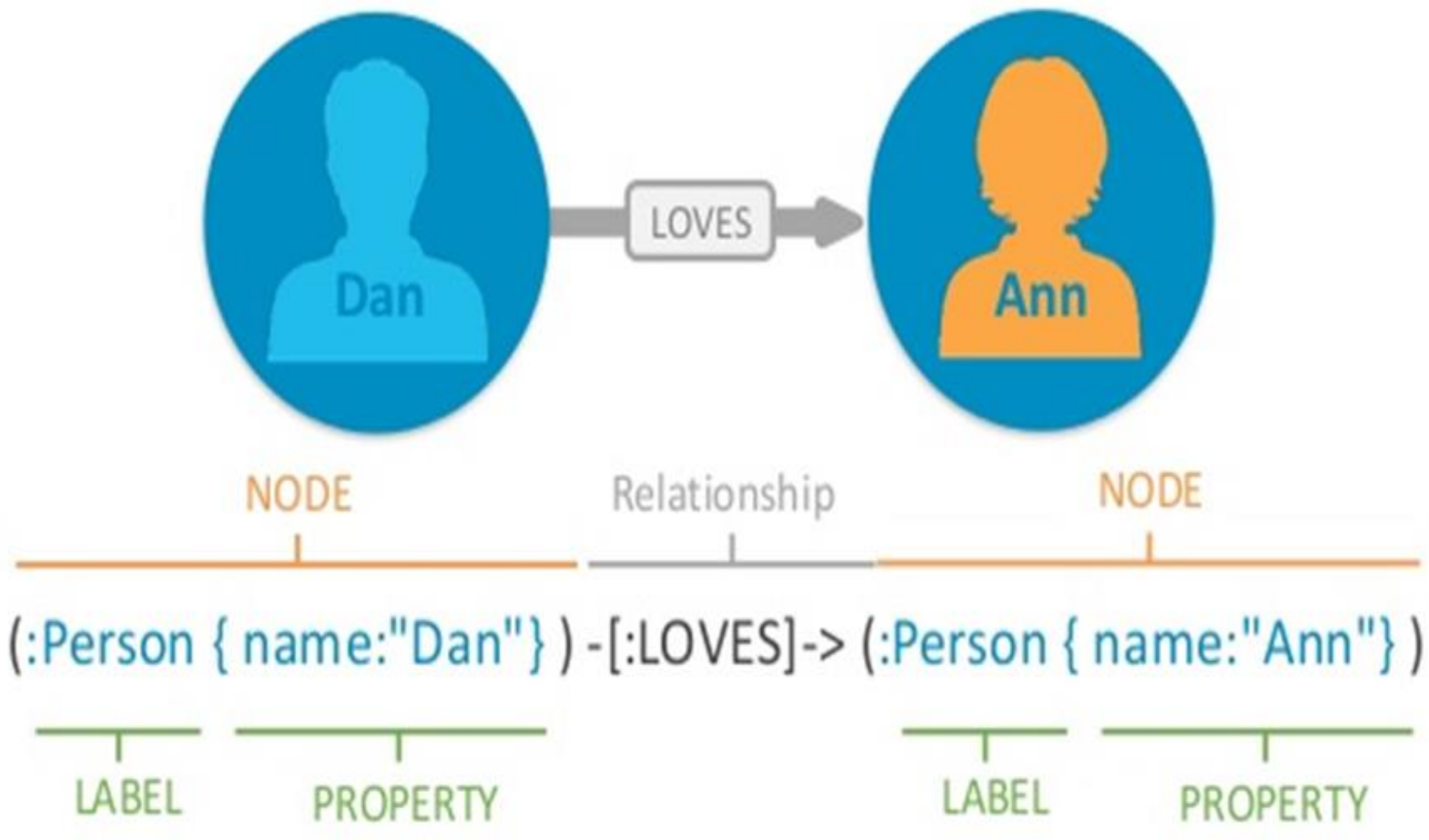
Cypher: (Neo4j) graph query language

uses ***patterns*** to describe graph data

familiar SQL-like clauses

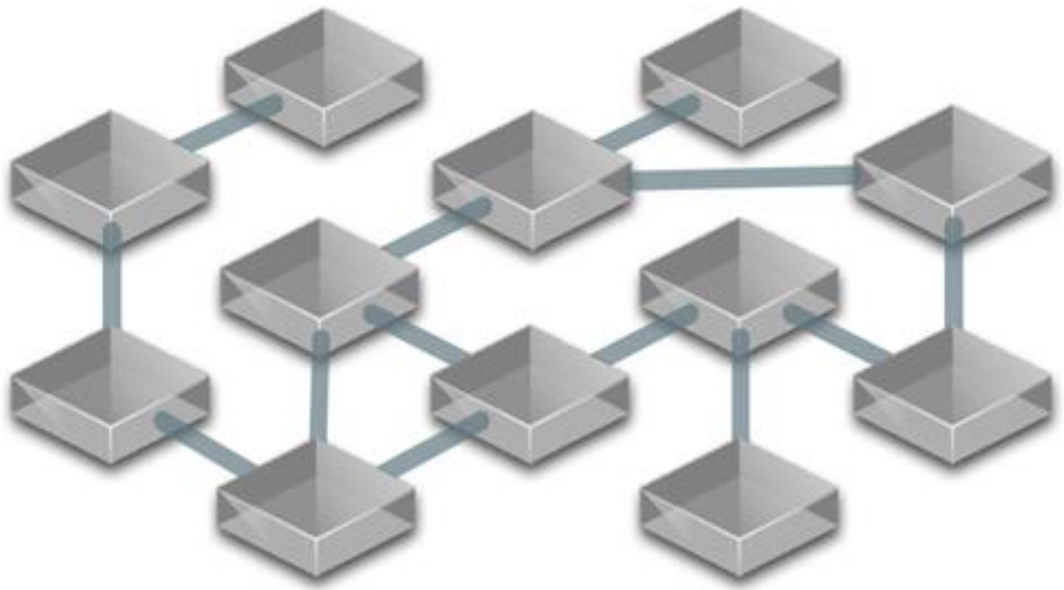
declarative, describing what to find, not how to find it

Cypher: Express Graph Patterns



Native Graph Processing

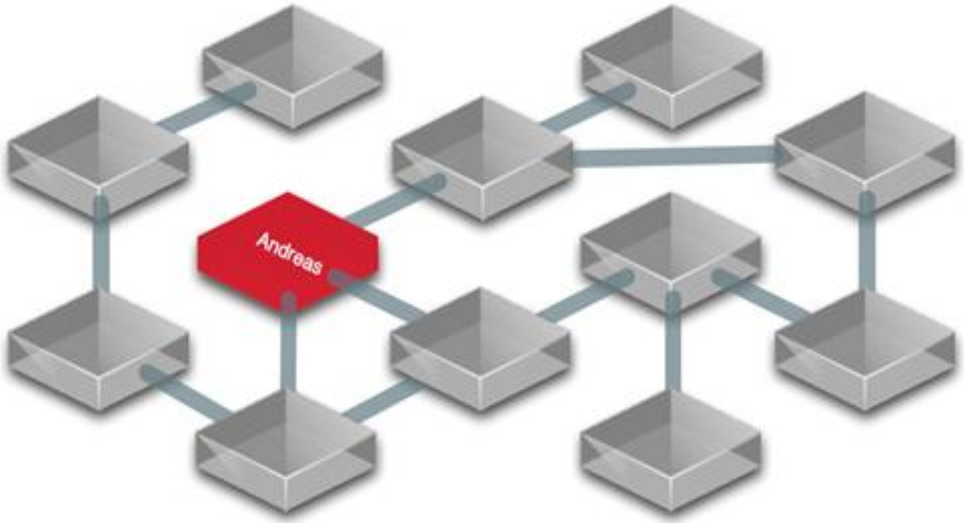
You traverse the graph



Native Graph Processing

You traverse the graph

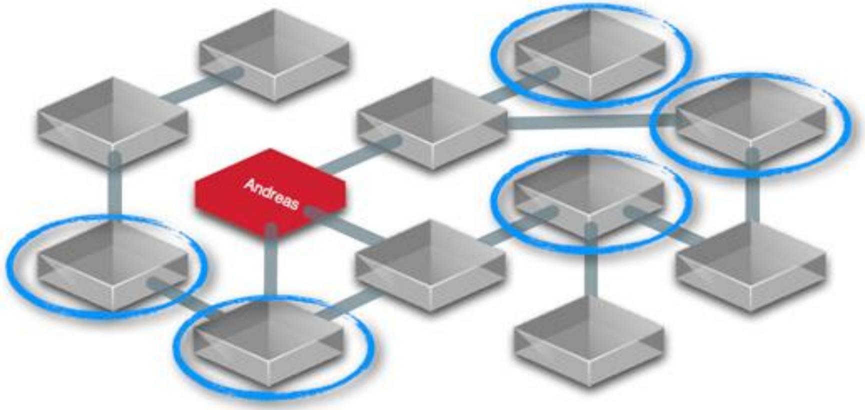
```
// find starting nodes
MATCH (me:Person {name:'Andreas'})
```



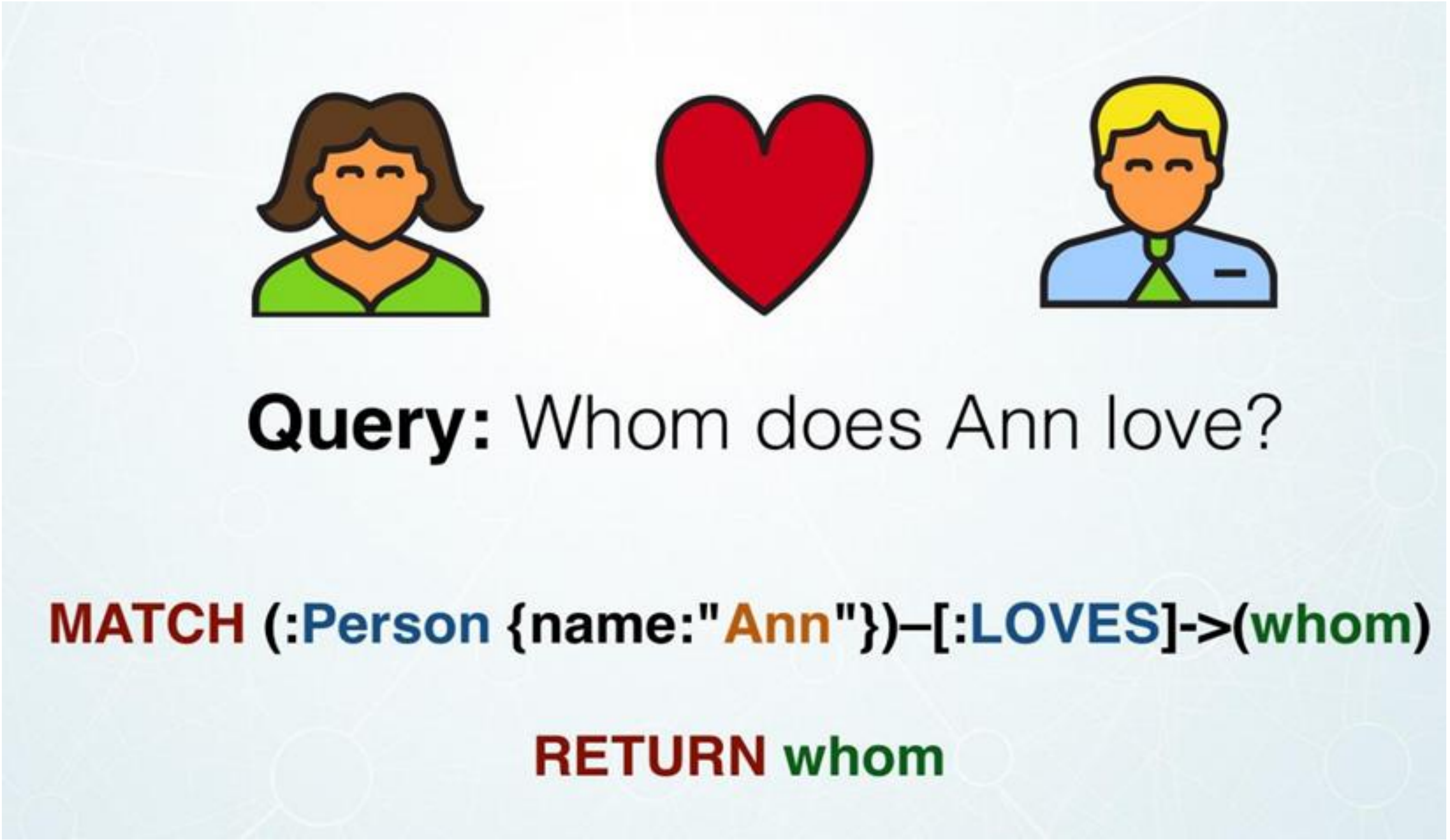
Query : friends of friend of Andreas

You traverse the graph

```
// then traverse the relationships
MATCH (me:Person {name:'Andreas'})-[:FRIEND]-(friend)
                                   -[:FRIEND]-(friend2)
RETURN friend2
```

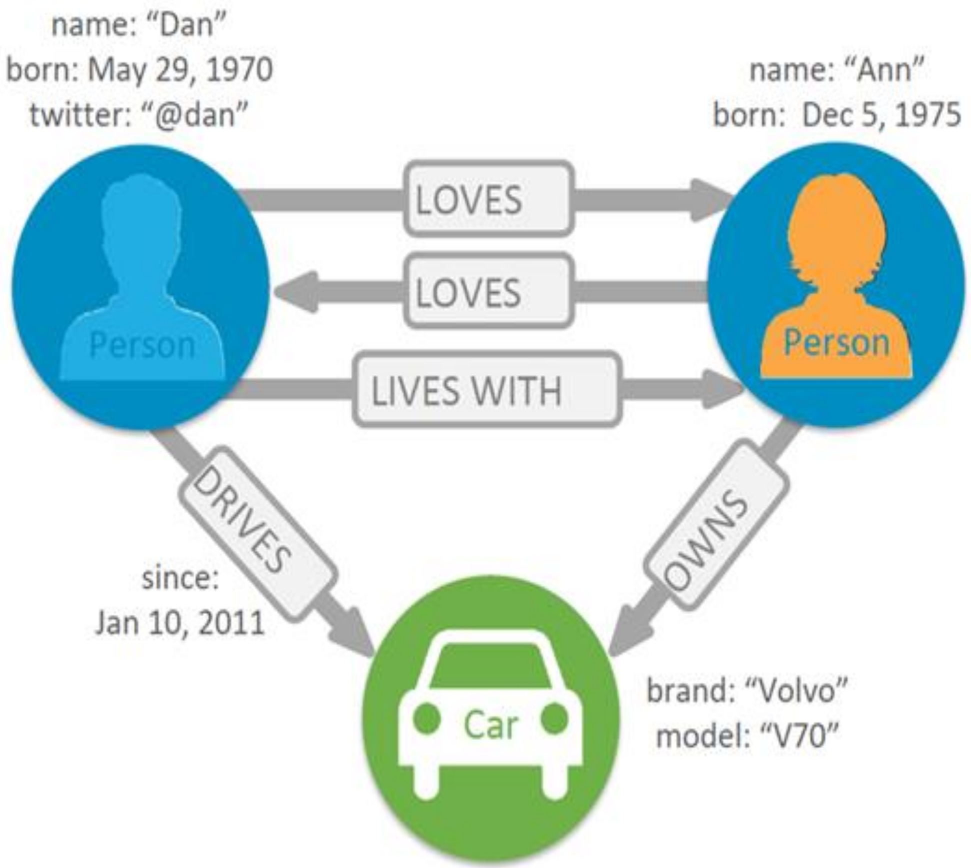


Example



Query: Whom does Ann love?
MATCH (:Person {name:"Ann"})–[:LOVES]->(whom)
RETURN whom

Native Graph Processing

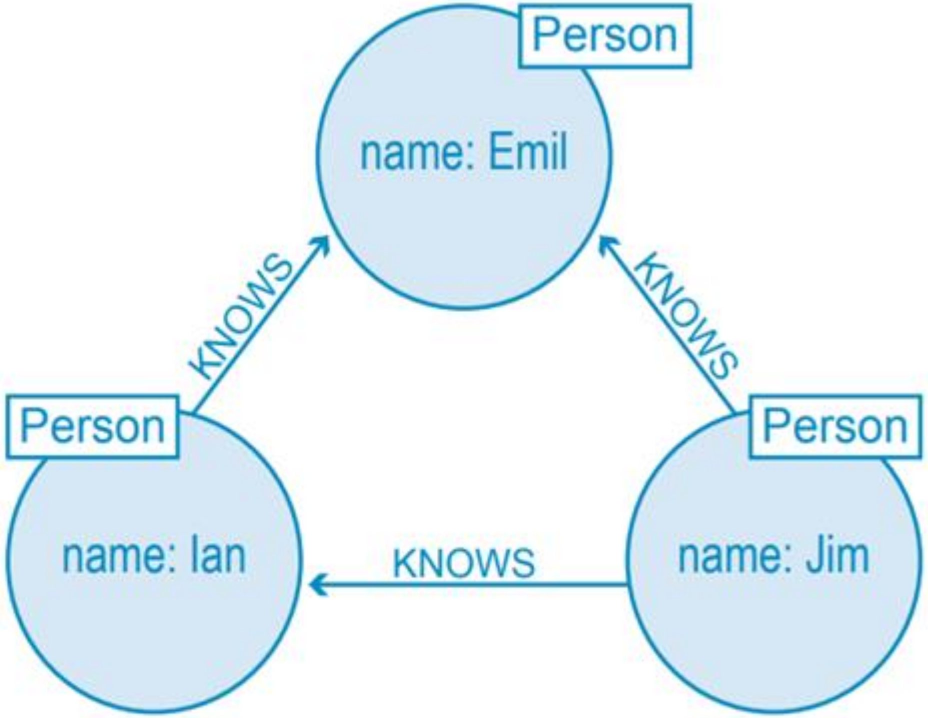



```
MATCH (:Person {name:"Ann"})–[:LOVES]->(whom)

RETURN whom
```

Example:

```
MATCH (a:Person {name:'Jim'})-[:KNOWS]->(b)-[:KNOWS]- >(c),
          (a)-[:KNOWS]->(c)
RETURN b, c
```



WHY
Graph Model?

Relational DB Pains

- Complex to model and Store relationships.
- Performance degrades with increases in data.
- Queries get long and complex.
- Maintenance is painful.

Example

"What items did Alice buy?"

User

UserID	User	Address	Phone	Email	Alternate
1	Alice	123 Foo St.	12345678	alice@example.org	alice@neo4j.org
2	Bob	456 Bar Ave.		bob@example.org	
...	...	...	...	...	...
99	Zach	99 South St.		zach@example.org	

Order

OrderID	UserID
1234	1
5678	1
...	...
5588	99

LineItem

OrderID	ProductID	Quantity
1234	765	2
1234	987	1
...	...	...
5588	765	1

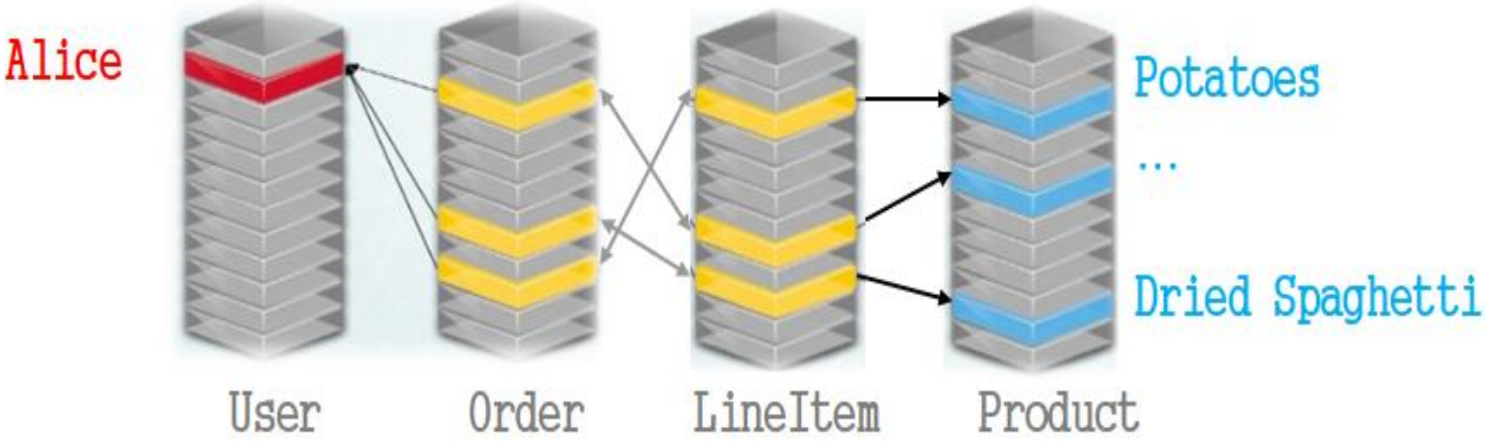
Product

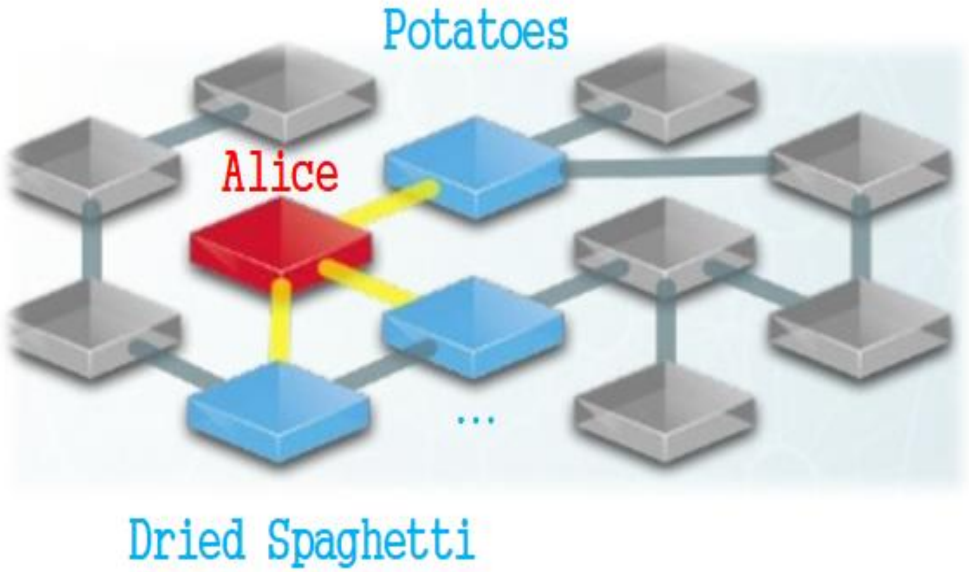
ProductID	Description	Handling
321	strawberry ice cream	freezer
765	potatoes	
...	...	
987	dried spaghetti	

"Which customers bought this product?"

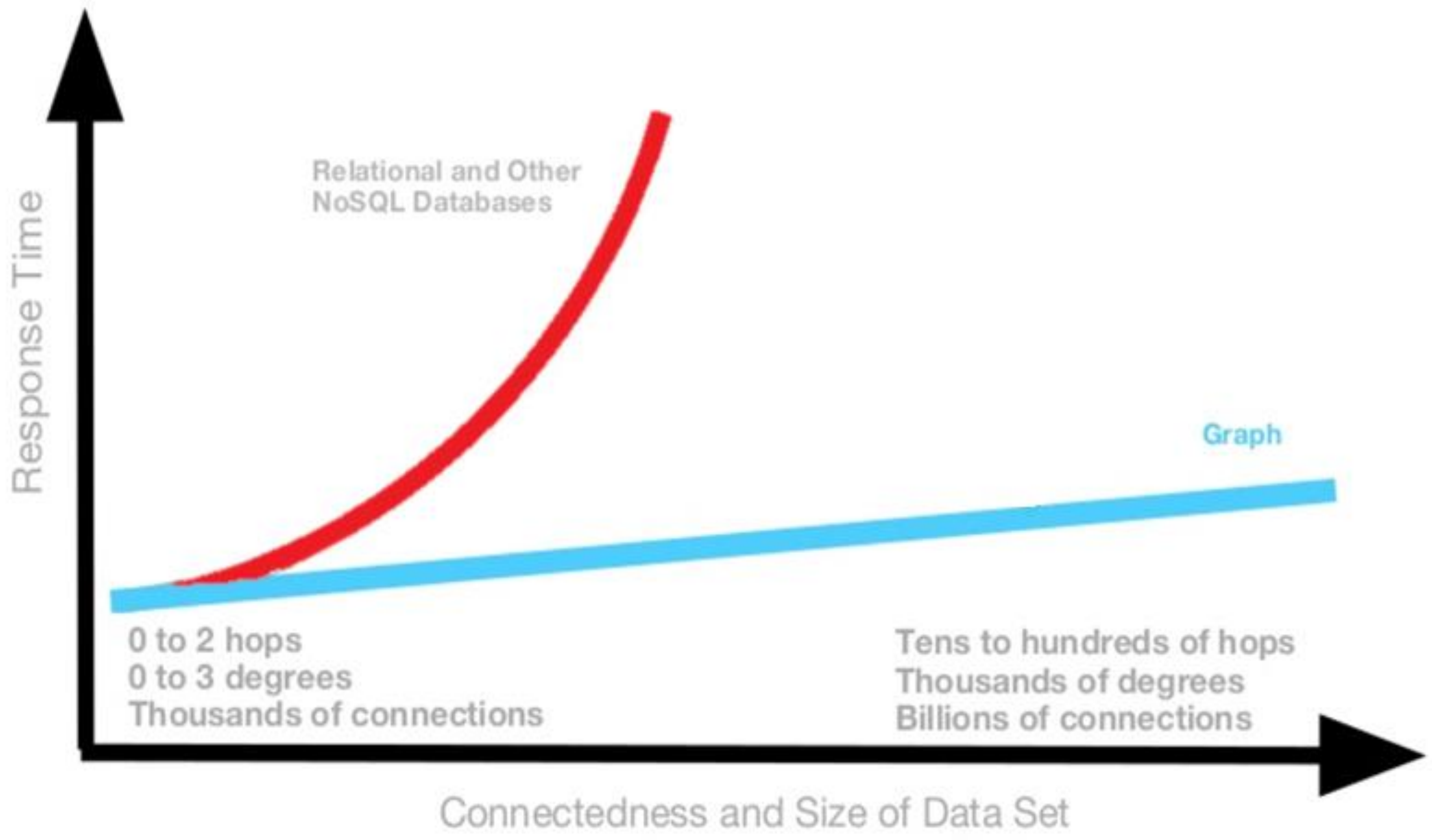
"Which customers buying this product also bought that product?"

Relational vs Graph Model

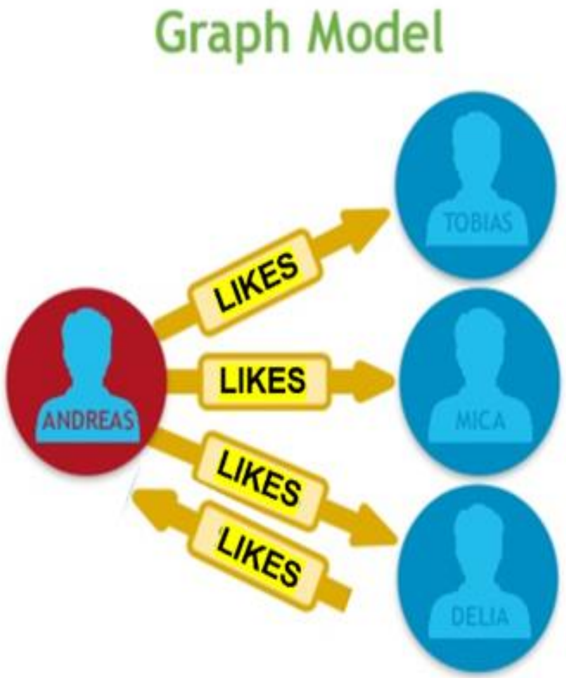




Performance degrades with increases in data.



Complex to model and Store relationships.



What if direction of relationships are concerned !!!

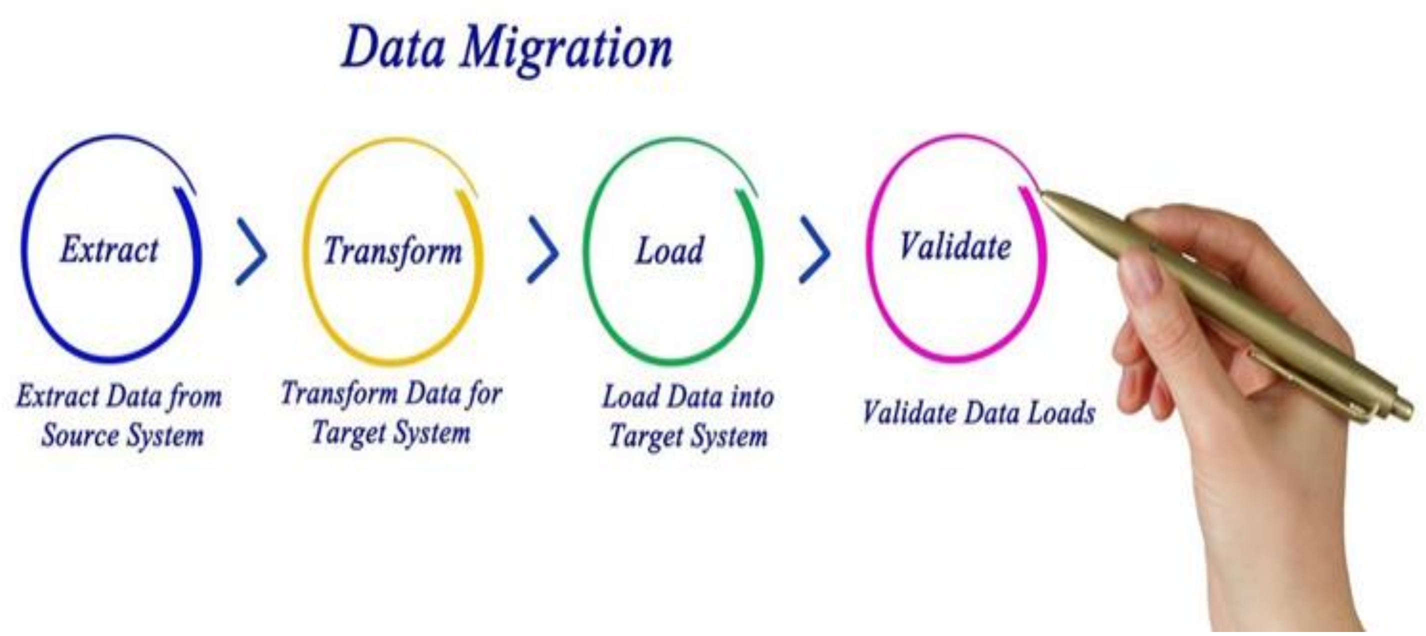
Person	
ID	Person
1	Alice
2	Bob
...	...
99	Zach

PersonFriend	
PersonID	FriendID
1	2
2	1
2	99
...	...
99	1

Queries get long and complex.

Maintenance is painful.

- Business requirement changes such as Adding new properties or relationships→ MIGRATION.



Graph Gains

Intuitiveness - Easy to model and store relationships

Speed - Performance of relationship traversal remains constant with growth in data size

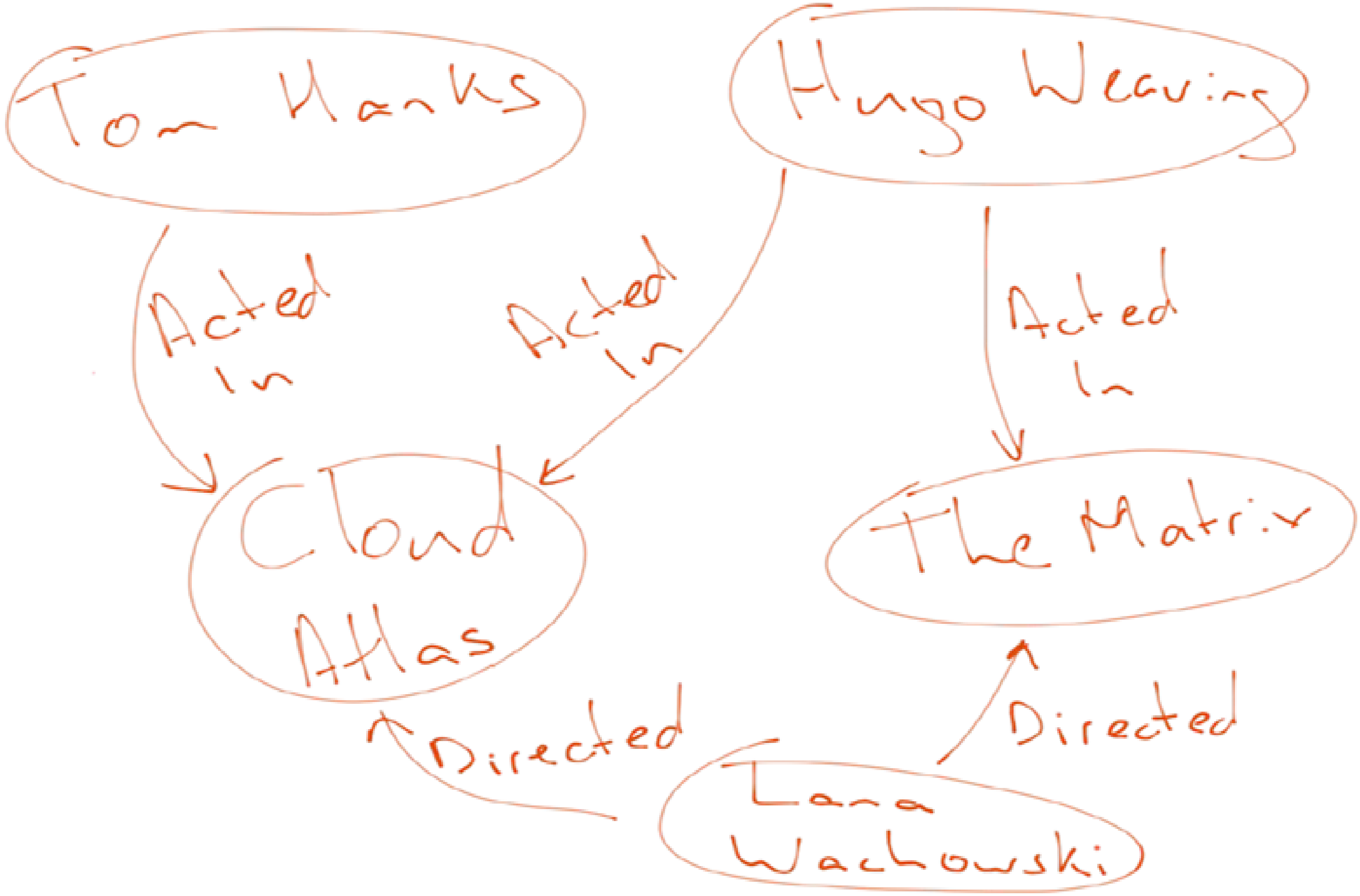
Agility

Queries are shortened and more readable

Adding addition properties and relationships can be done on the fly – no migrations

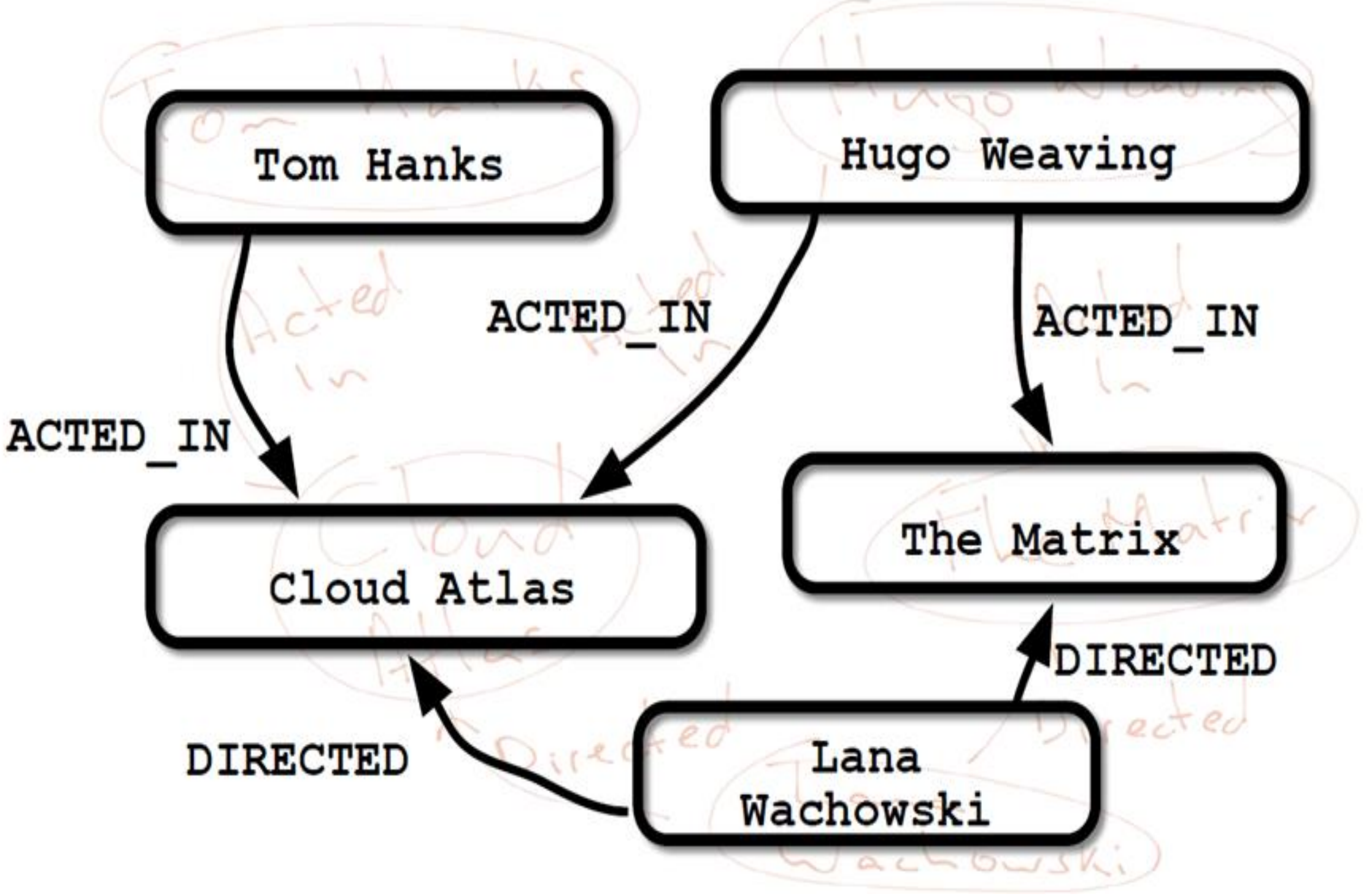
Intuitiveness

◎ Easy to model and store relationships



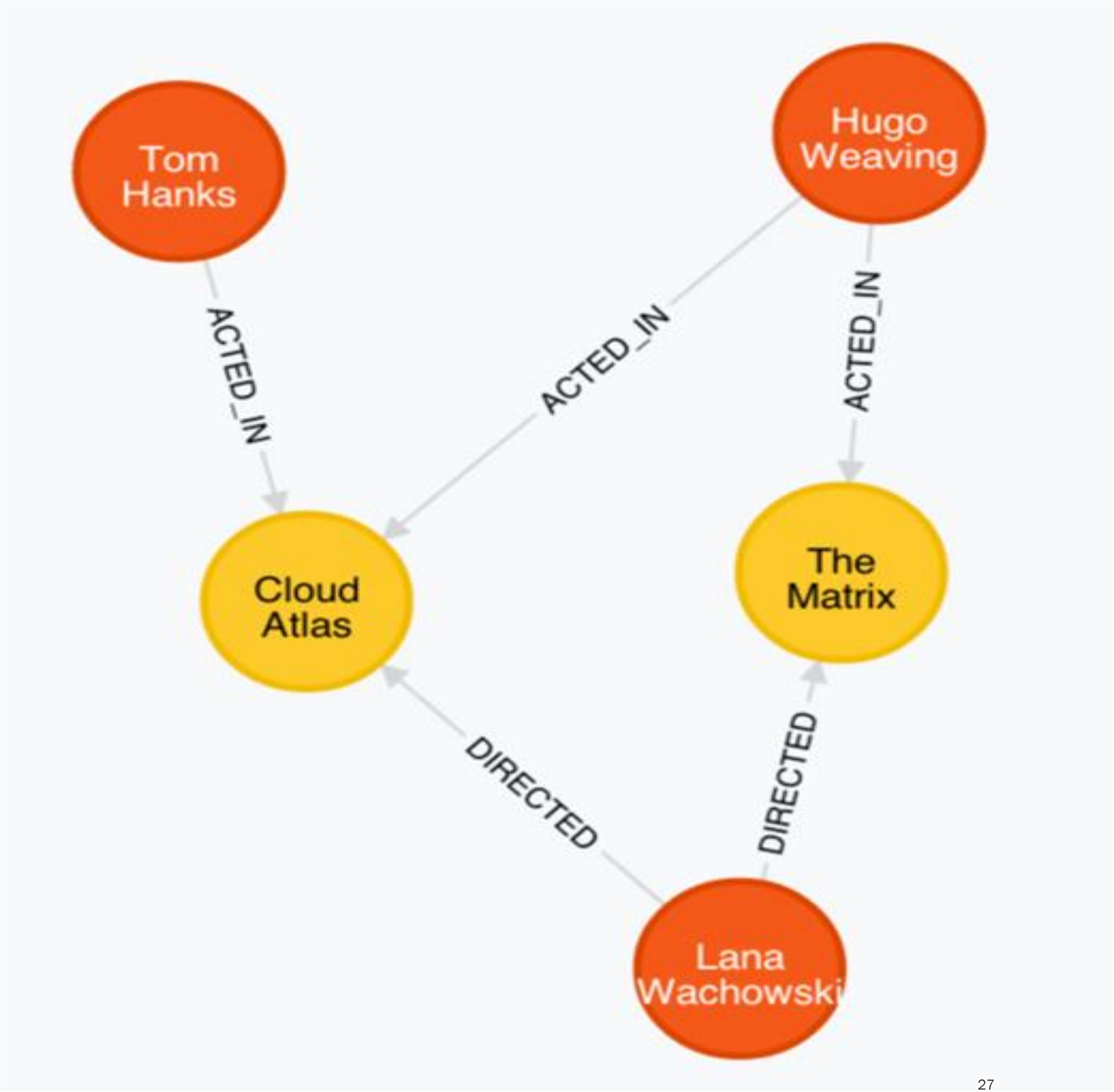
Intuitiveness

◎ Easy to model and store relationships

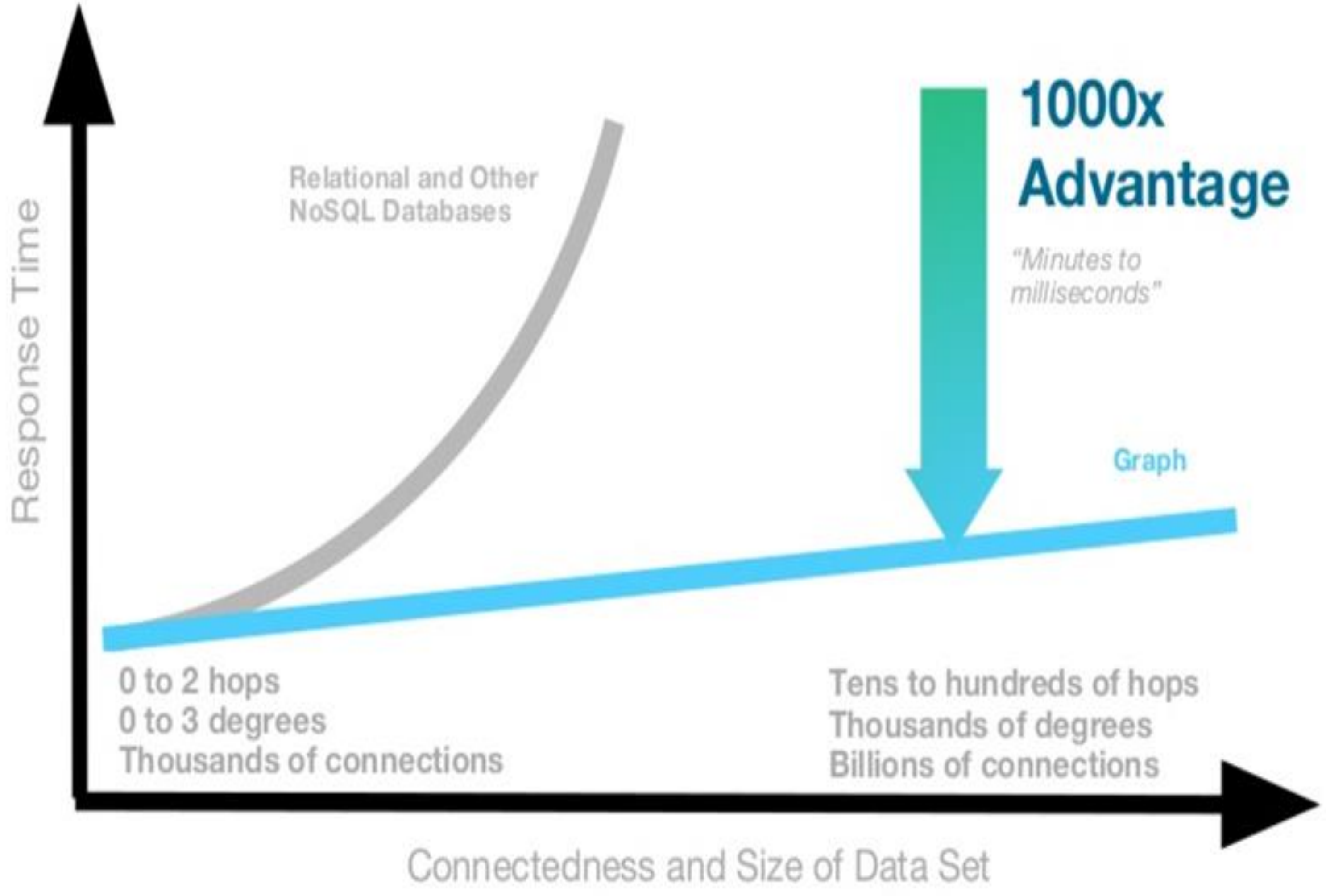


Intuitiveness

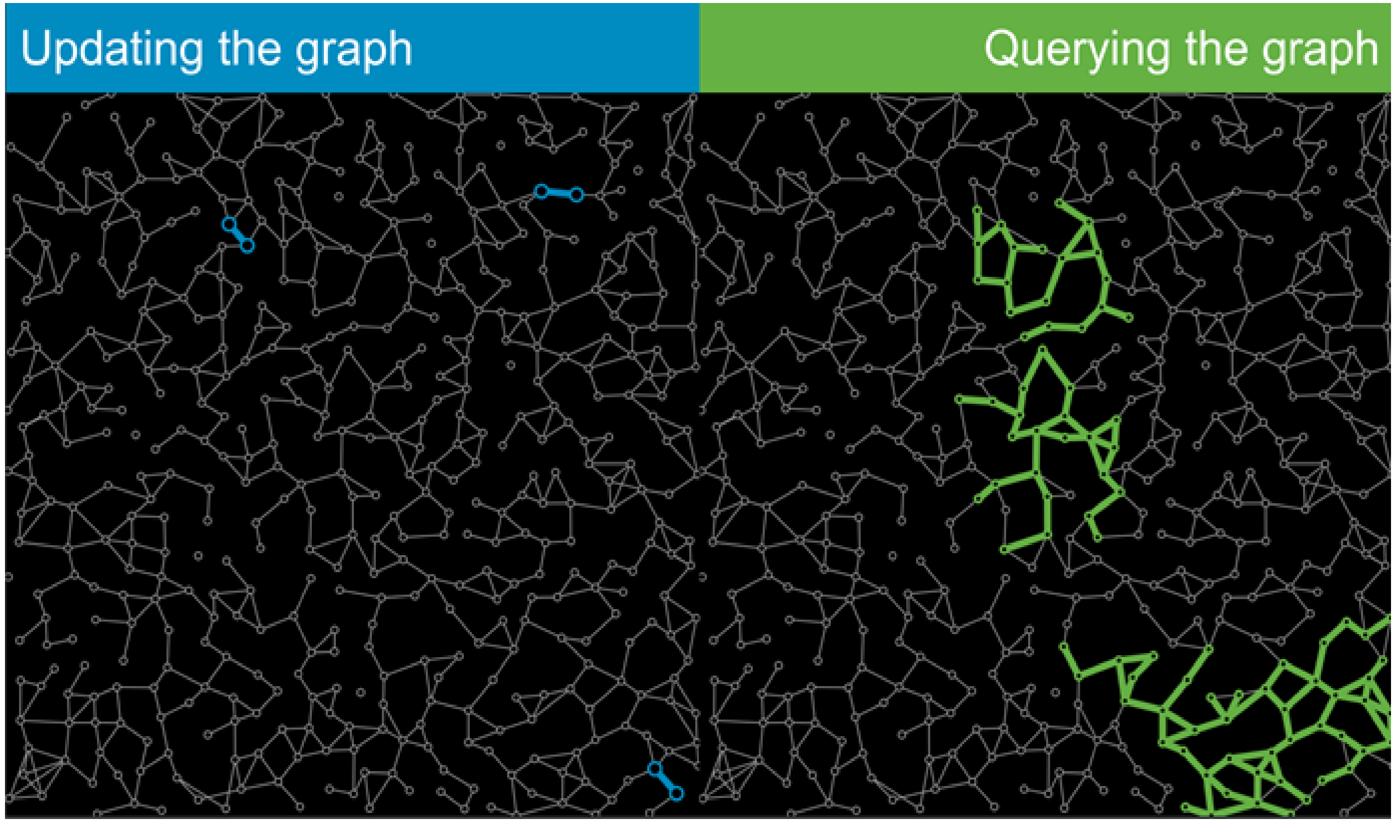
- Easy to model and store relationships



Performance of relationship traversal remains constant with growth in data size



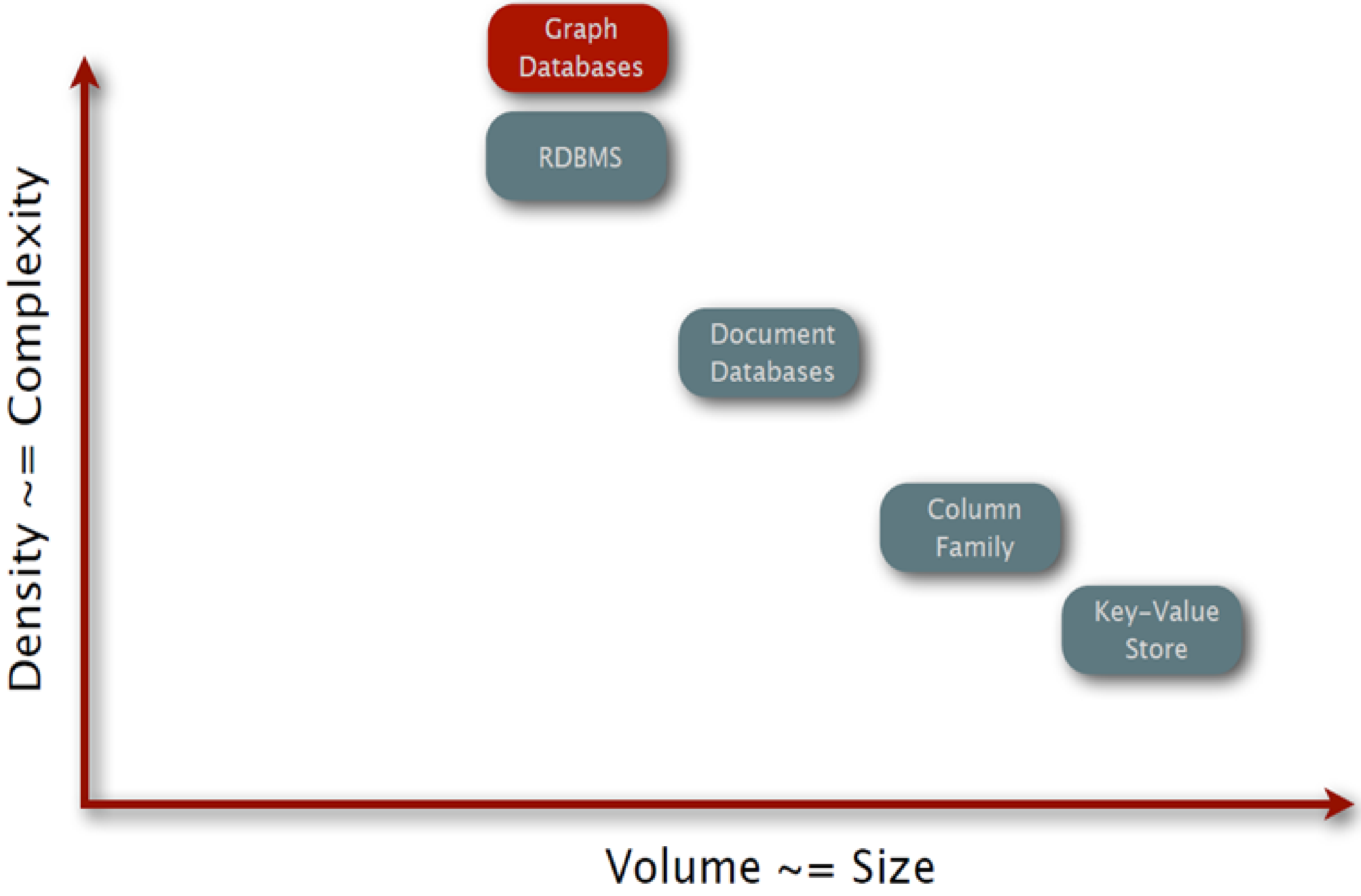
Agile



NoSQL DB allow to alter the schema without migration.

- add new nodes and relationships without the process of migrating.

Use the Right Database for the Right Job



the REFERENCE

- P. Sadalage and M. Fowler: NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence, Addison-Wesley Professional, 2013
- Jan L. Harrington: Relational Database Design and Implementation, 4th edition, Morgan Kaufmann, 2016
- A. Makris, K. Tserpesa, V. Andronikou Dimosthenis Anagnostopoulos: A Classification of NoSQL Data Stores Based on Key Design Characteristics, Procedia Computer Science, Vol. 97, 2016, pp. 94-103.