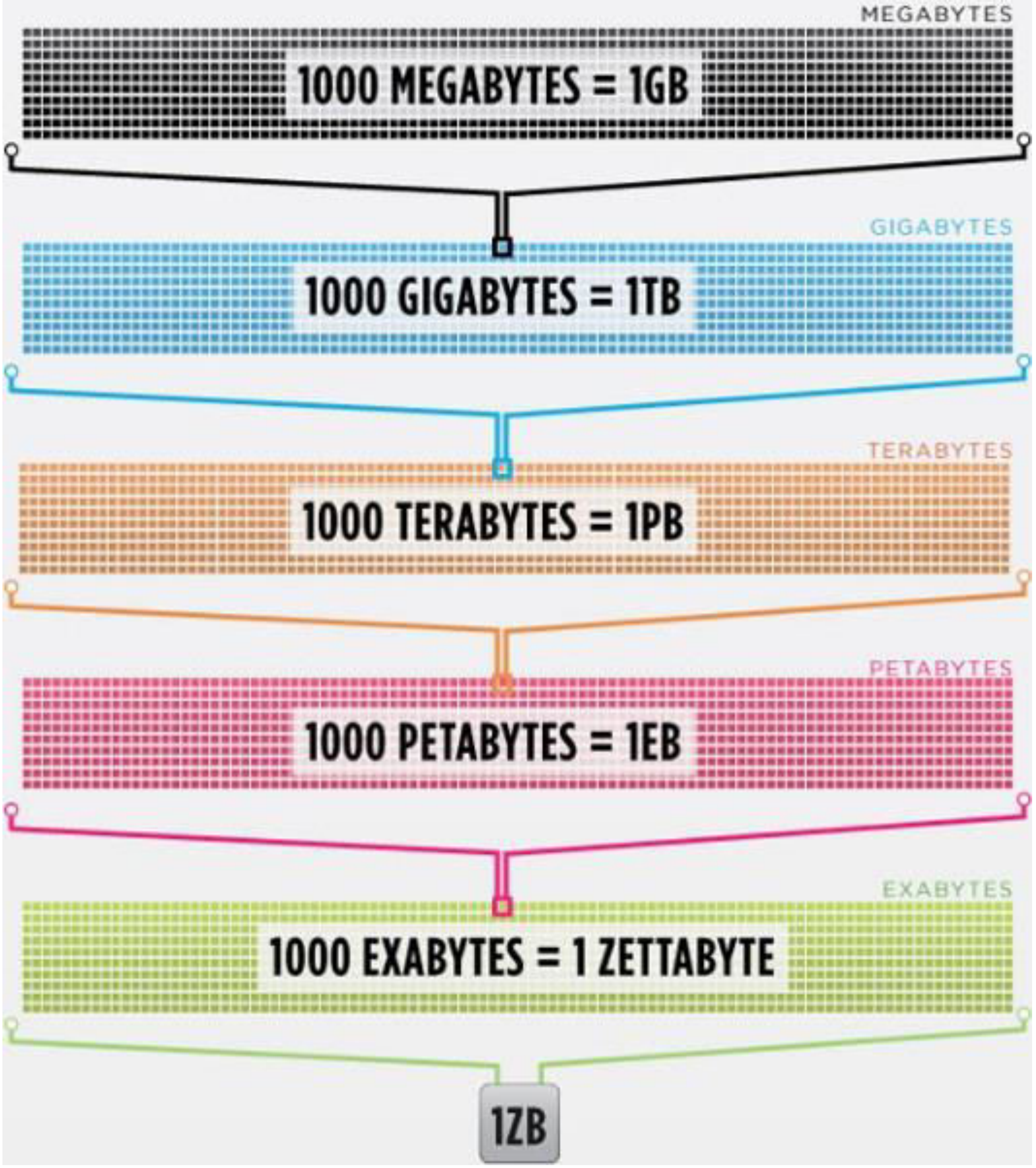
NoSQL Databases



Data is getting bigger:

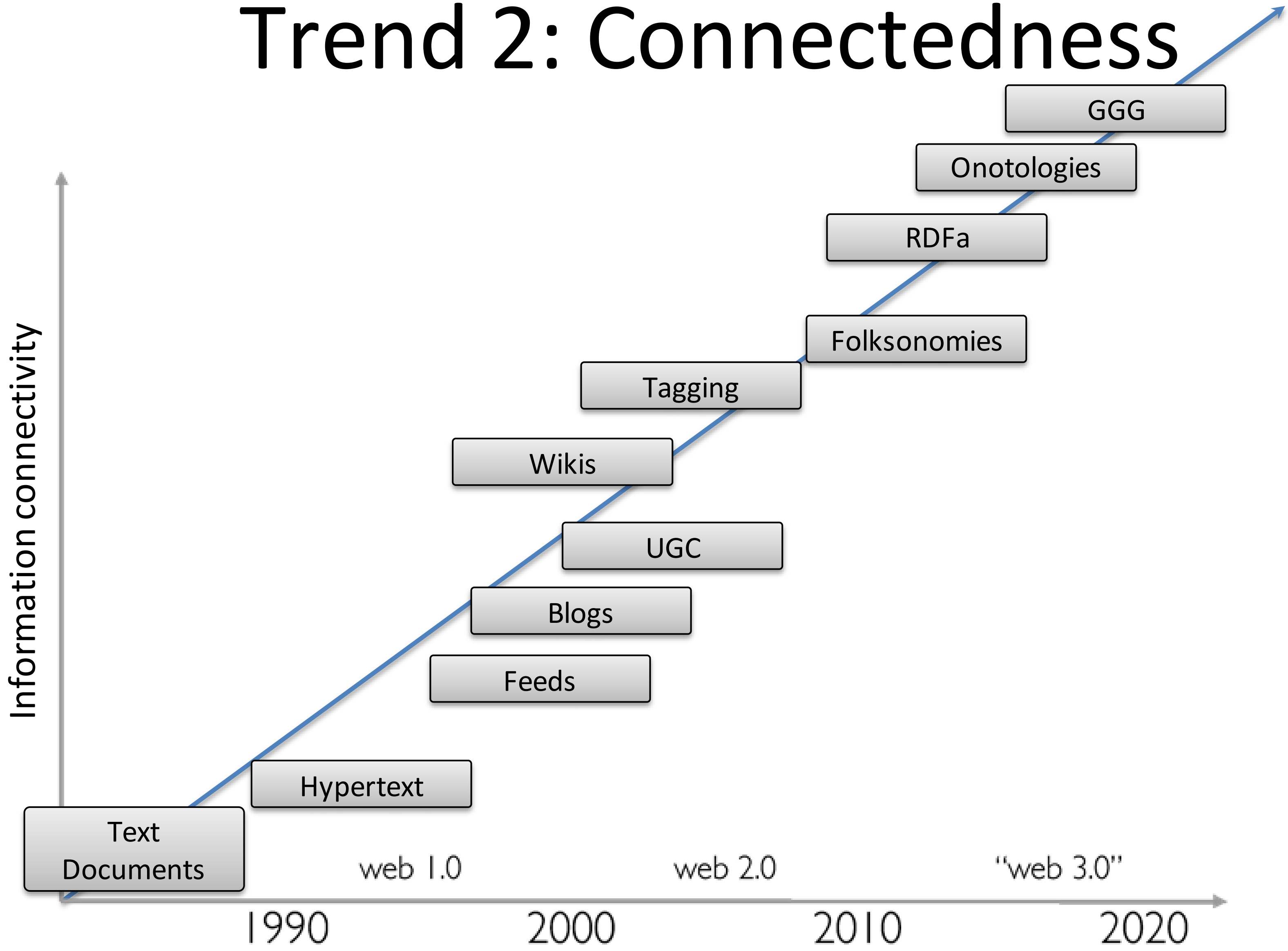
“Every 2 days we create as much information as we did up to 2003”

– Eric Schmidt, Google

Data is more connected:

- Text
- HyperText
- RSS
- Blogs
- Tagging
- RDF

Trend 2: Connectedness

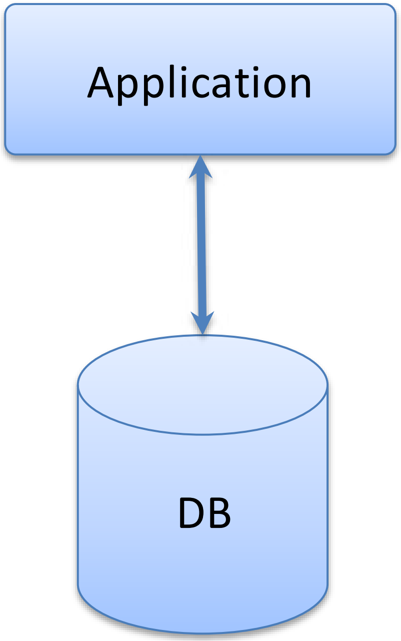


Data is more Semi-Structured:

- If you tried to collect all the data of every movie ever made, how would you model it?
- Actors, Characters, Locations, Dates, Costs, Ratings, Showings, Ticket Sales, etc.

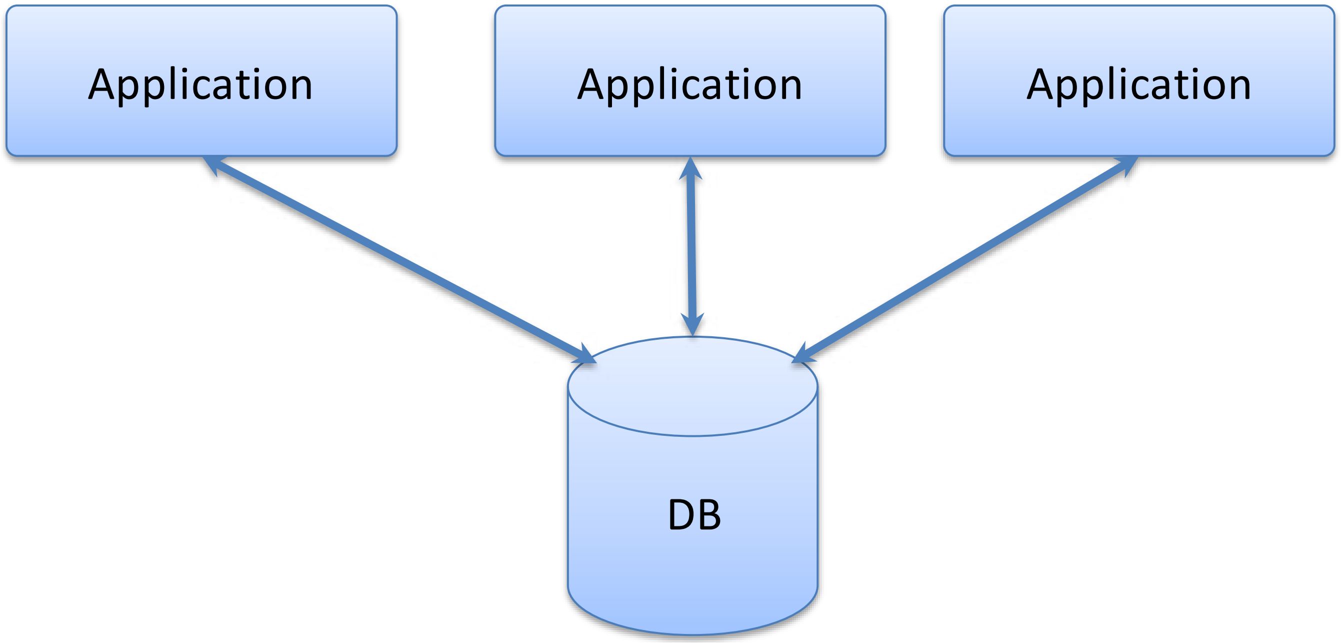
Architecture Changes Over Time

1980’s: Single Application



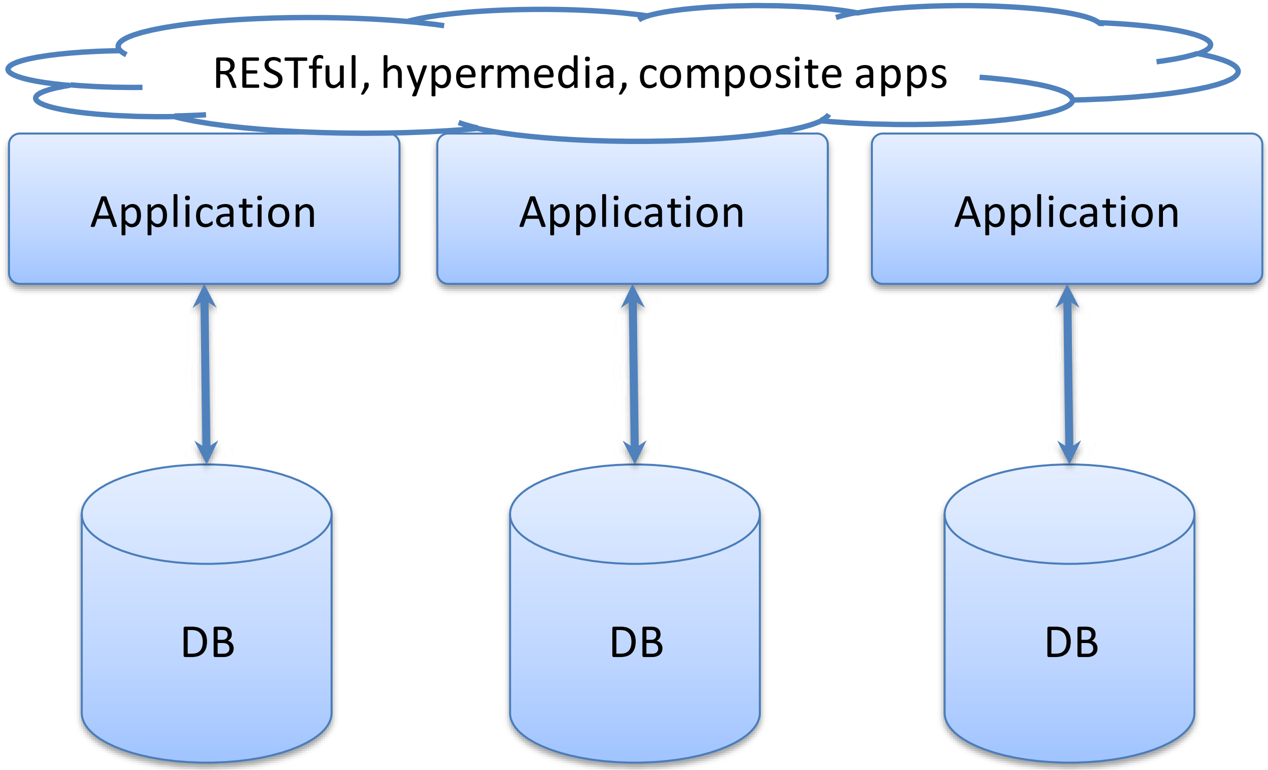
Architecture Changes Over Time

1990's: Integration
Database Antipattern

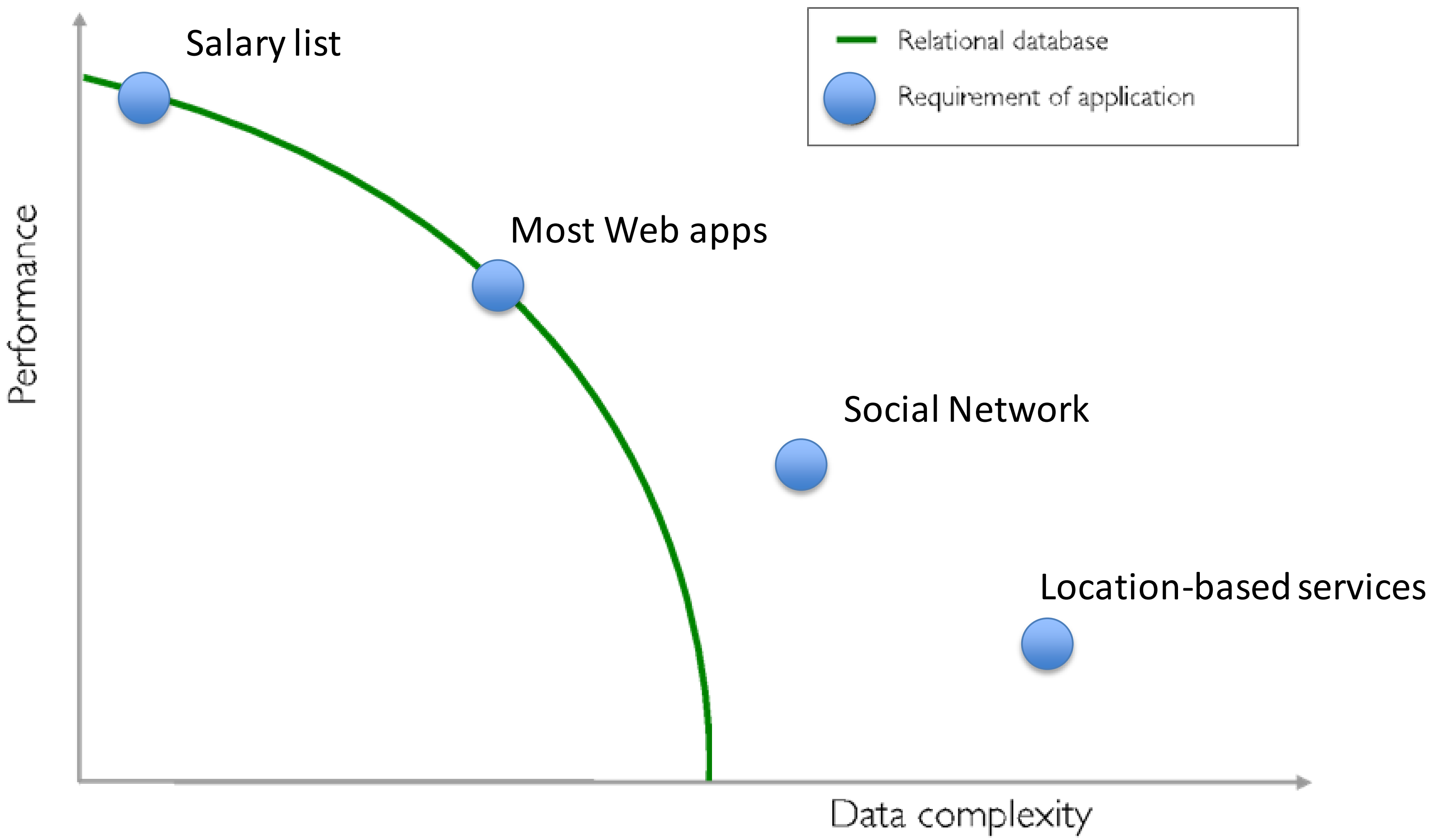


Architecture Changes Over Time

2000’s: SOA



Side note: RDBMS performance



NOSQL

Not Only SQL

Less than 10% of the NOSQL Vendors

Redis
HBASE
Cassandra
mongoDB
riak
CouchDB
relax
Tokyo
Cabinet
Scalaris
Project Voldemort
Neo4j
the graph database
membase

Key Value Stores

- Came from a research article written by Amazon (Dynamo)
  - Global Distributed Hash Table
    - Global collection of key value pairs

Key Value Stores: Pros and Cons

- Pros:
  - Simple data model
  - Scalable
- Cons
  - Poor for complex data

Column Family

- Most Based on **BigTable**: Google's Distributed Storage System for Structured Data
- Data Model:
  - A big table, with column families
    - Every row can have its own schema
    - Helps capture more "messy" data
  - Map Reduce for querying/processing
- Examples:
  - HBase, HyperTable, Cassandra

Column Family: Pros and Cons

- Pros:
  - Supports Semi-Structured Data
  - Naturally Indexed (columns)
  - Scalable
- Cons
  - Poor for interconnected data

Document Databases

- Inspired by Lotus Notes
  - Collection of Key value pair collections (called Documents)

Document Databases: Pros and Cons

- Pros:
  - Simple, powerful data model
  - Scalable
- Cons
  - Poor for interconnected data
  - Query model limited to keys and indexes
  - Map reduce for larger queries

Graph Databases

- Data Model:
  - Nodes and Relationships
- Examples:
  - Neo4j, OrientDB, InfiniteGraph, AllegroGraph

Graph Databases: Pros and Cons

- Pros:
  - Powerful data model, as general as RDBMS
  - Connected data locally indexed
  - Easy to query
- Cons
  - Sharding (lots of people working on this)
    - Scales UP reasonably well
  - Requires rewiring your brain

What are graphs good for?

- Recommendations
- Business intelligence
- Social computing
- Geospatial
- Systems management
- Web of things
- Genealogy
- Time series data
- Product catalogue
- Web analytics
- Scientific computing (especially bioinformatics)
- Indexing your *slow* RDBMS
- And much more!

What is a Graph?

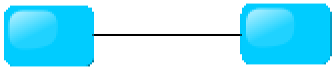
What is a Graph?

- An abstract representation of a set of objects where some pairs are connected by links.

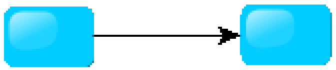
id	Object (Vertex, Node)
→	Link (Edge, Arc, Relationship)

Different Kinds of Graphs

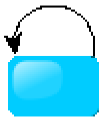
- Undirected Graph



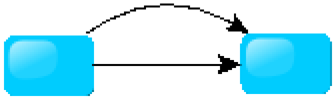
- Directed Graph



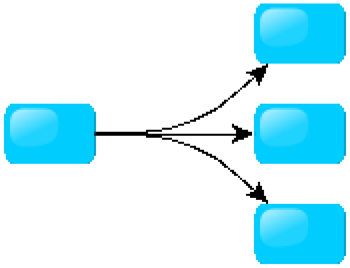
- Pseudo Graph



- Multi Graph

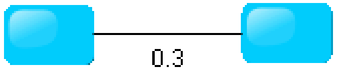


- Hyper Graph

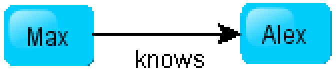


More Kinds of Graphs

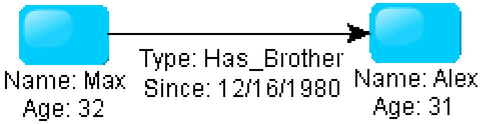
- Weighted Graph



- Labeled Graph



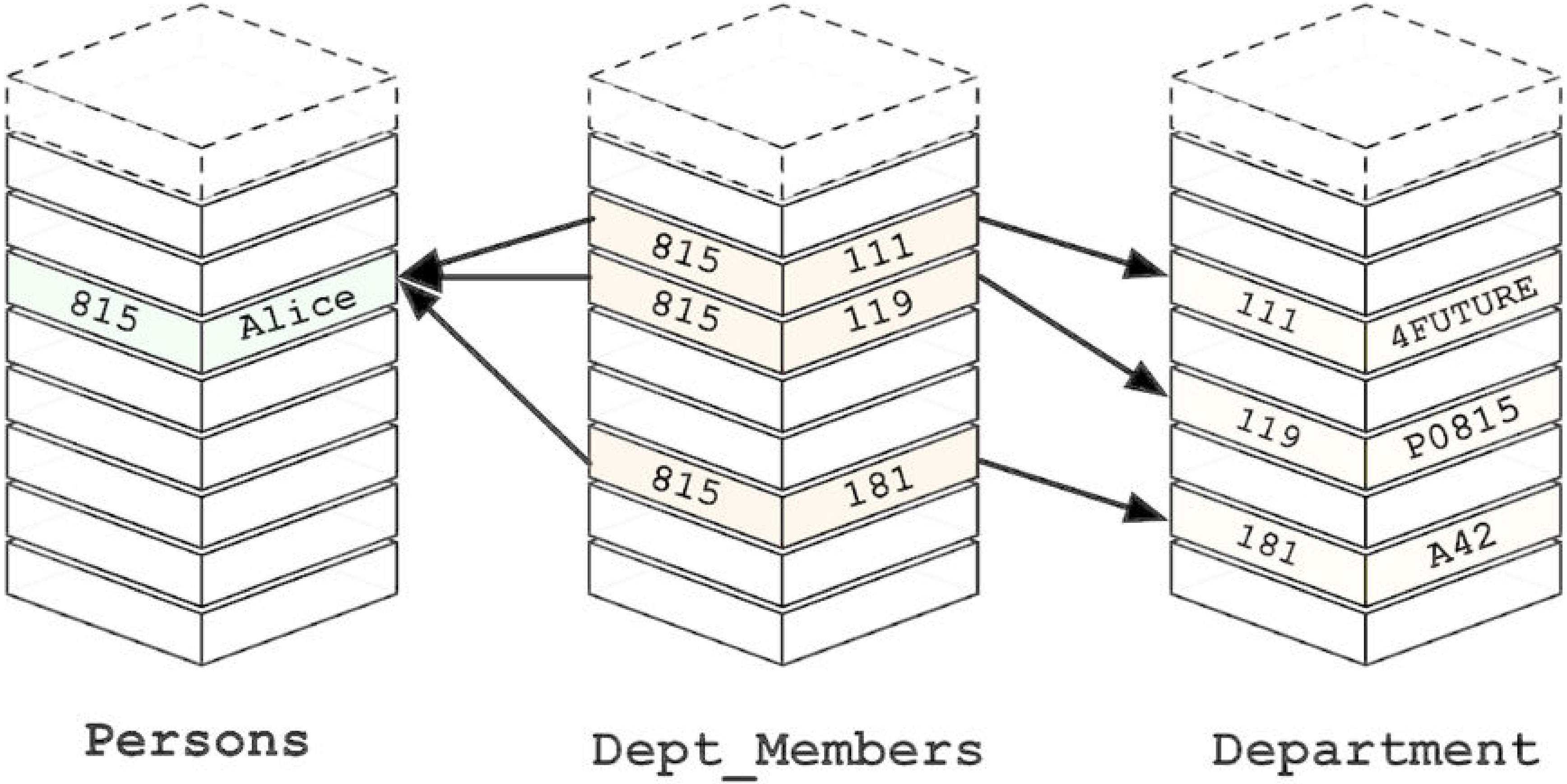
- Property Graph



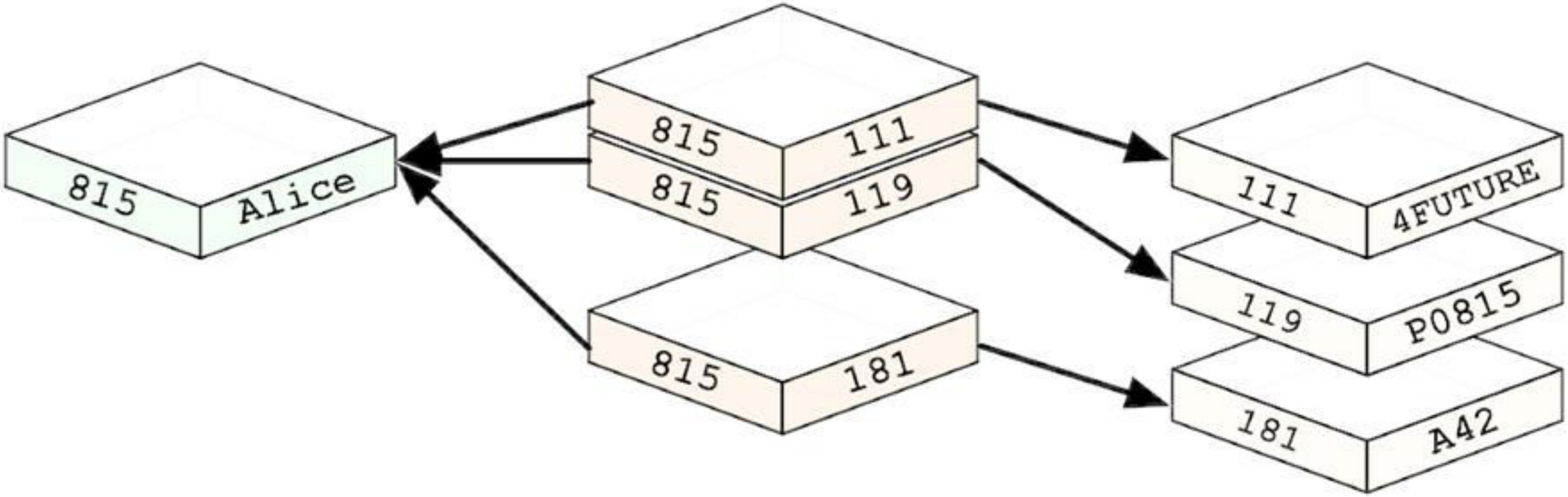
What is a Graph Database?

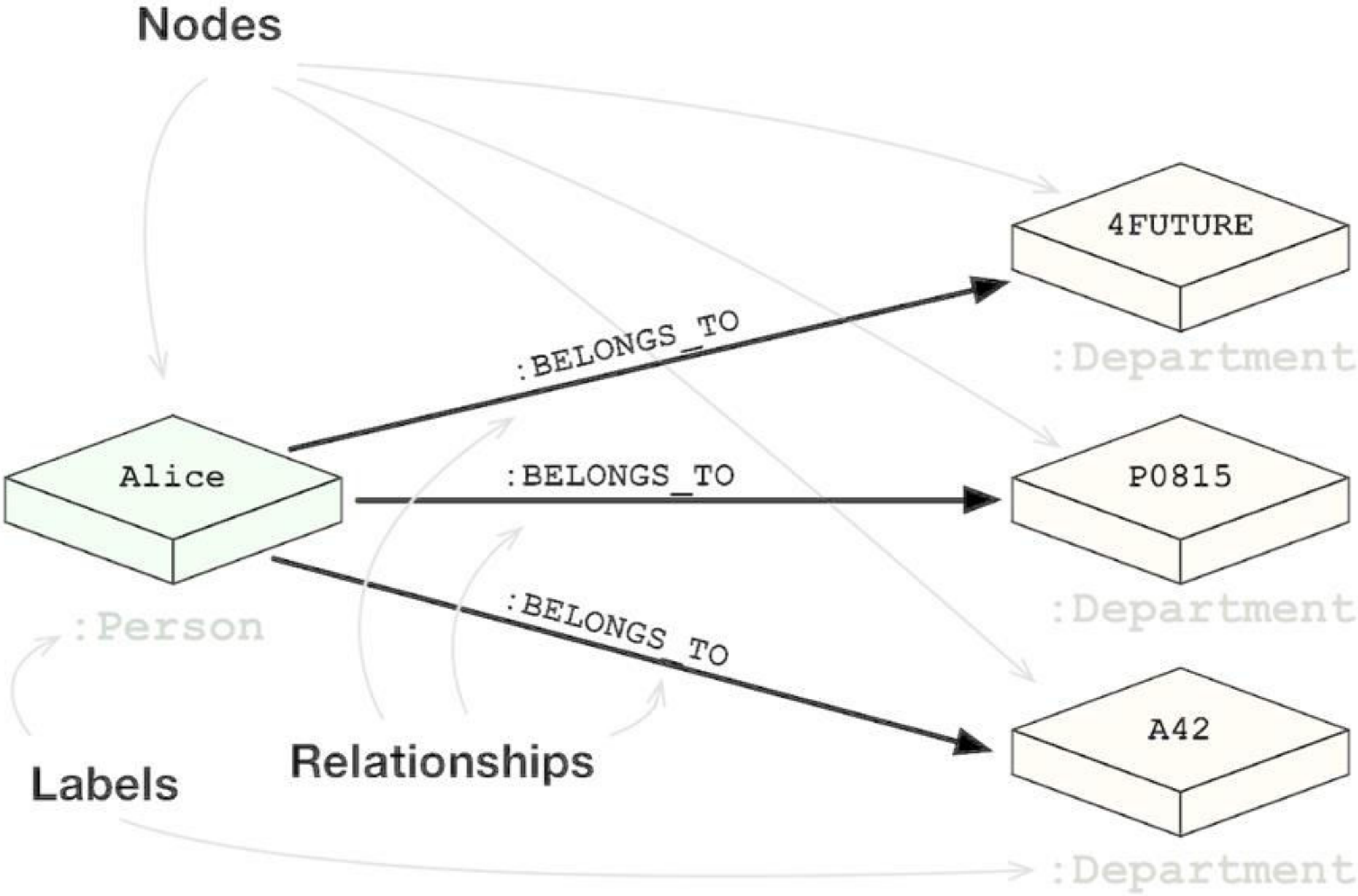
- A database with an explicit graph structure
- Each node knows its adjacent nodes
- As the number of nodes increases, the cost of a local step (or hop) remains the same
- Plus an Index for lookups

Relational Databases



Graph Databases

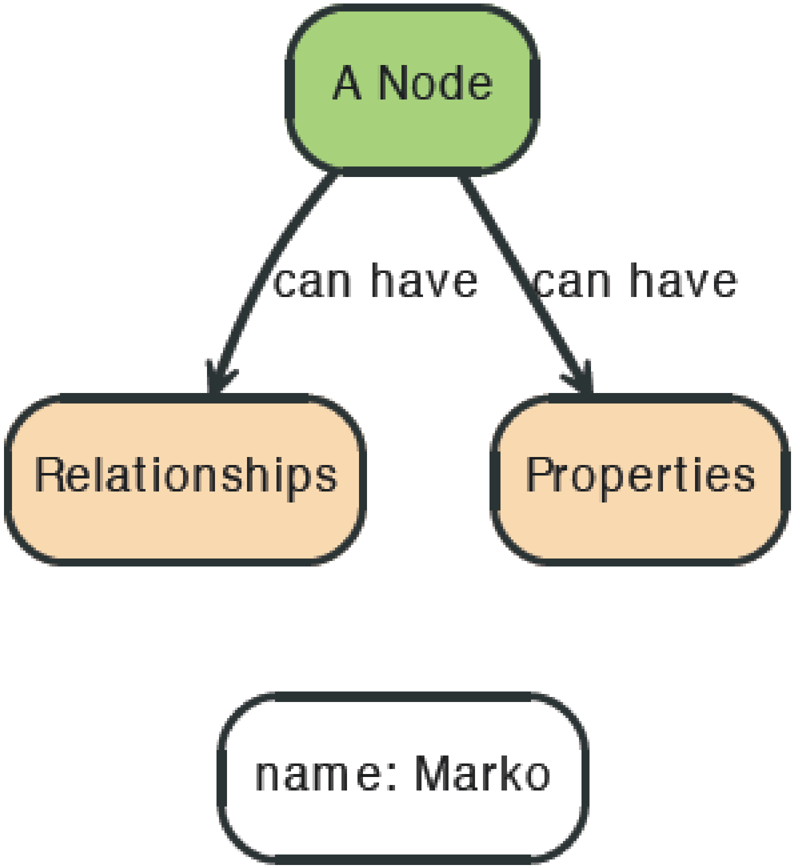



Nodes
4FUTURE
:Department
:BELONGS_TO
Alice
:BELONGS_TO
P0815
:Department
:Person
:BELONGS_TO
A42
Relationships
Labels
:Department

Neo4j Tips

- Each entity table is represented by a label on nodes
- Each row in a entity table is a node
- Columns on those tables become node properties.
- Join tables are transformed into relationships, columns on those tables become relationship properties

Node in Neo4j



A Node

can have

can have

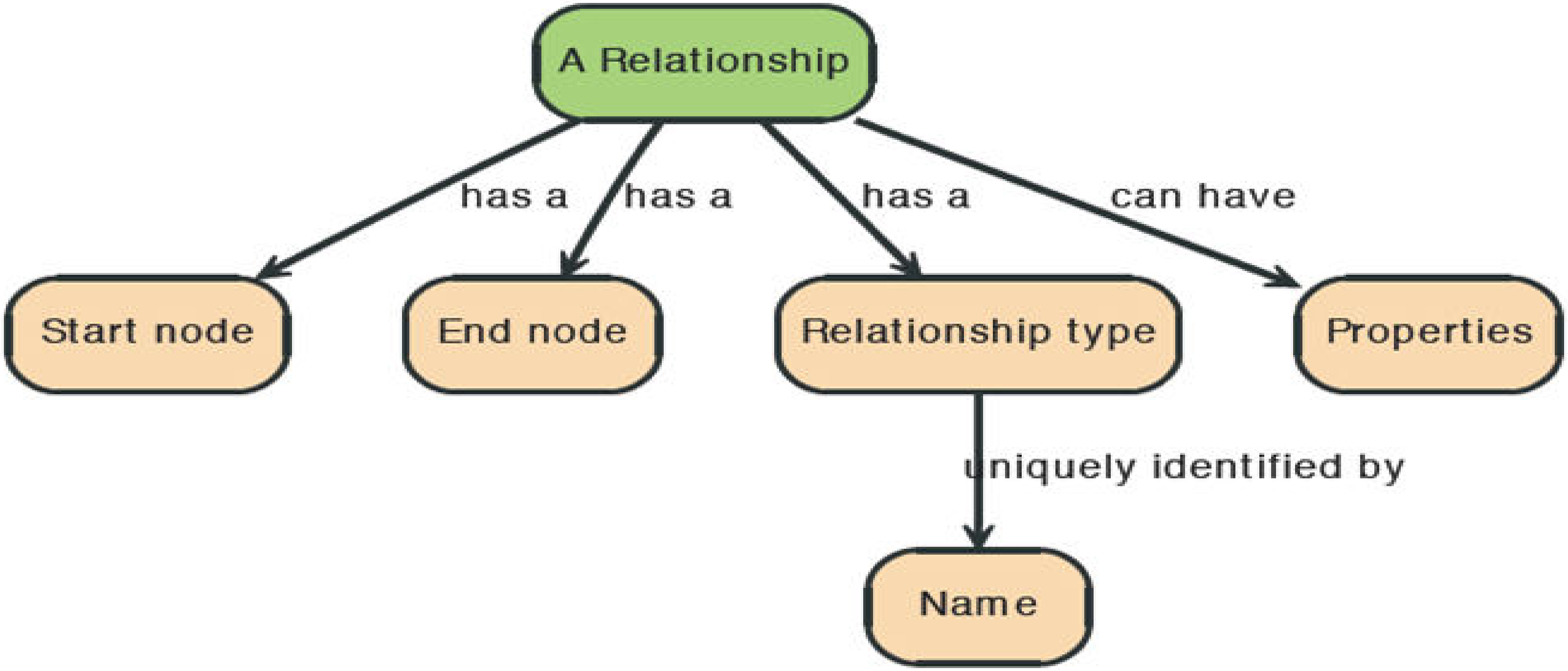
Relationships

Properties

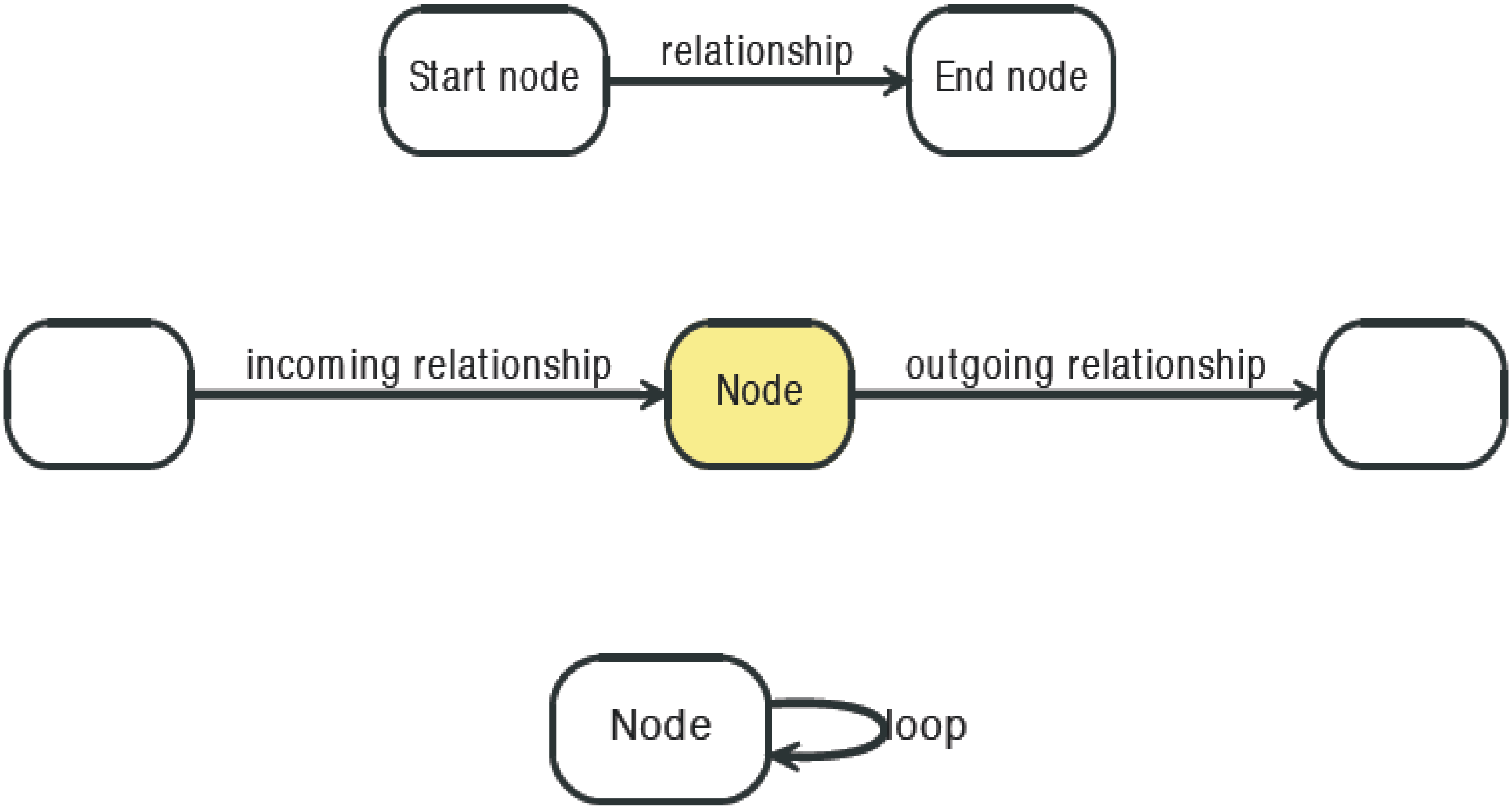
name: Marko

Relationships in Neo4j

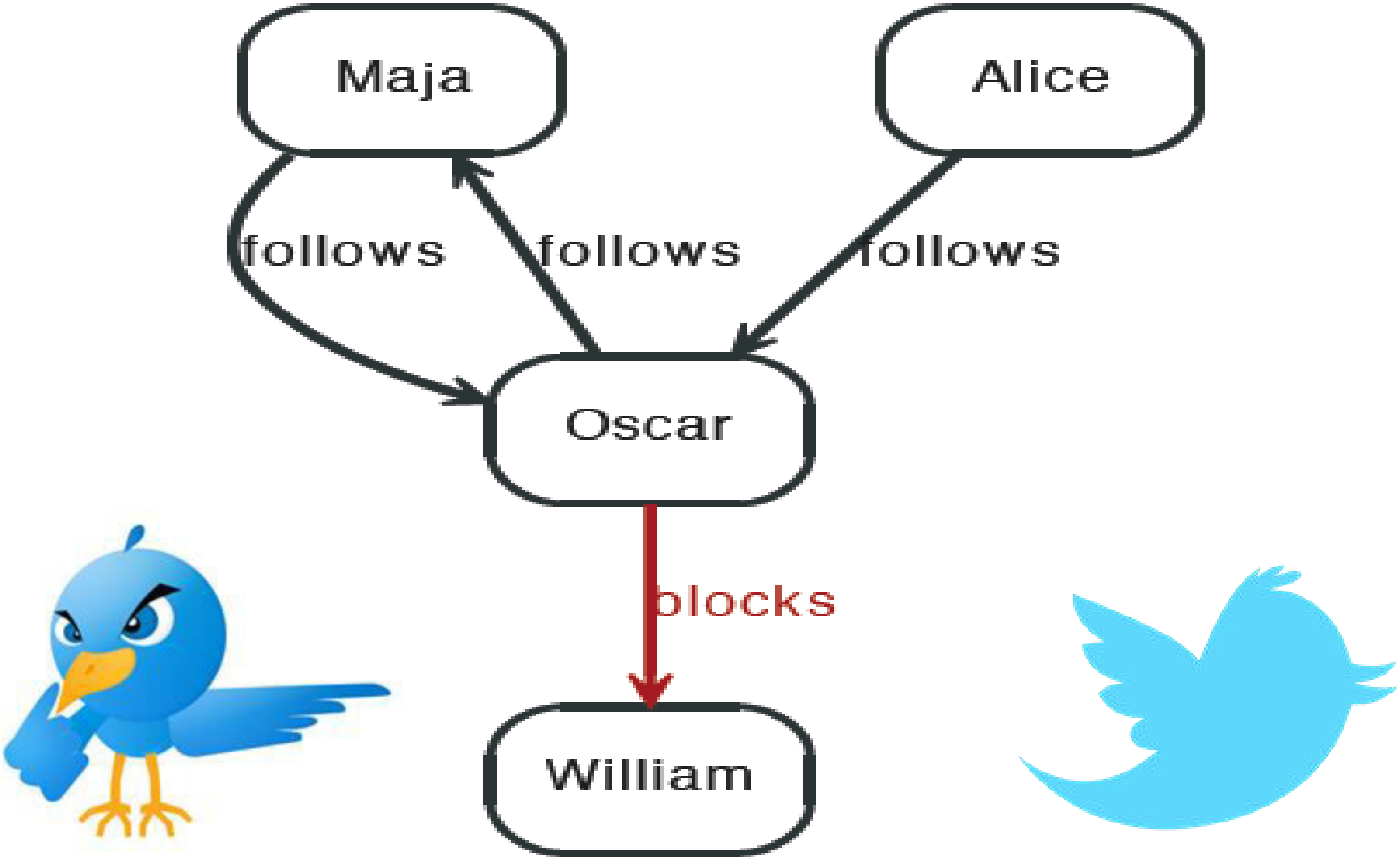
- Relationships between nodes are a key part of Neo4j.



Relationships in Neo4j



Twitter and relationships



Maja

Alice

follows

follows

follows

Oscar

blocks

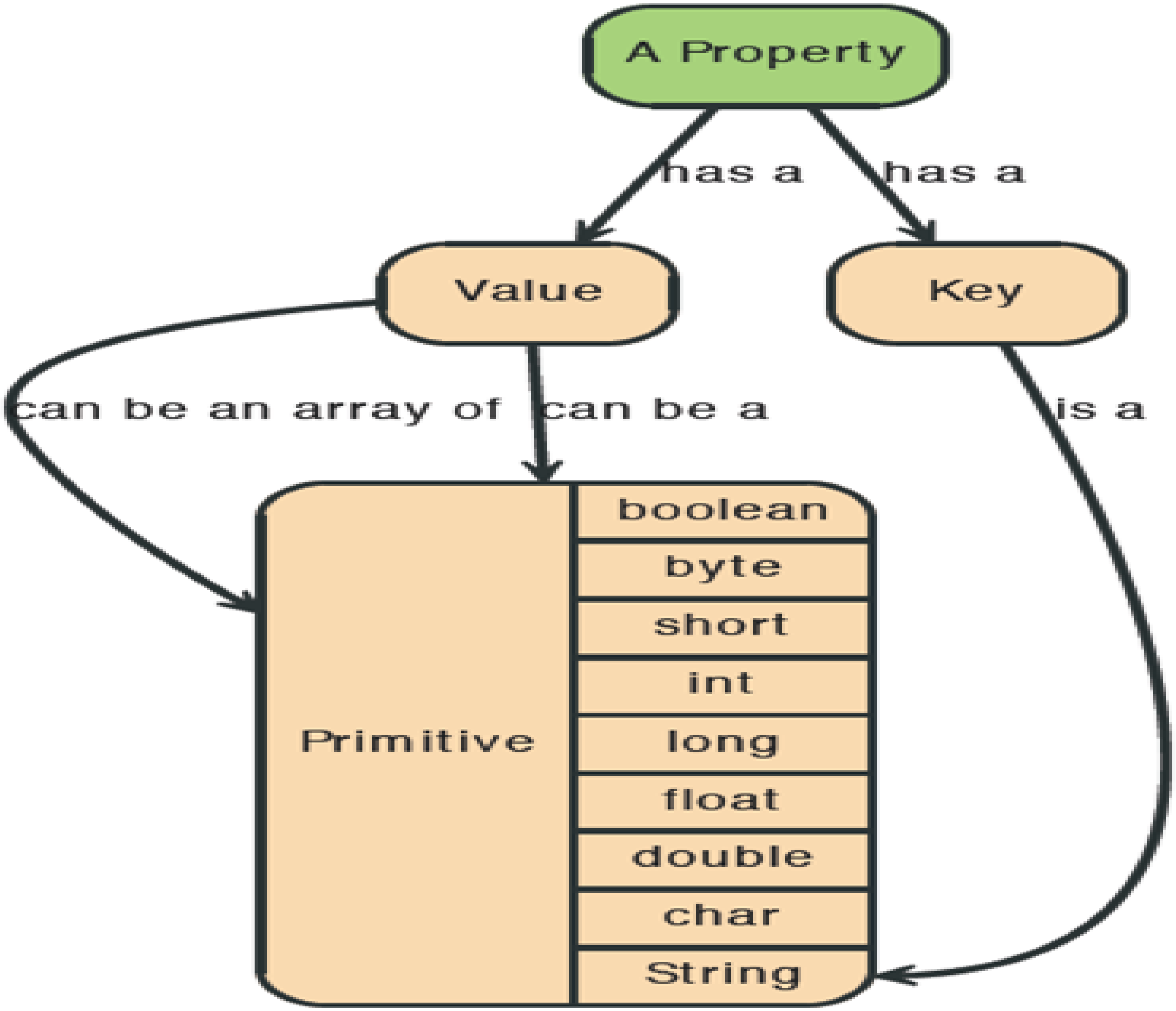
William

Properties

- Both nodes and relationships can have properties.
- Properties are key-value pairs where the key is a string.
- Property values can be either a primitive or an array of one primitive type.

  For example String, int and int[] values are valid for properties.

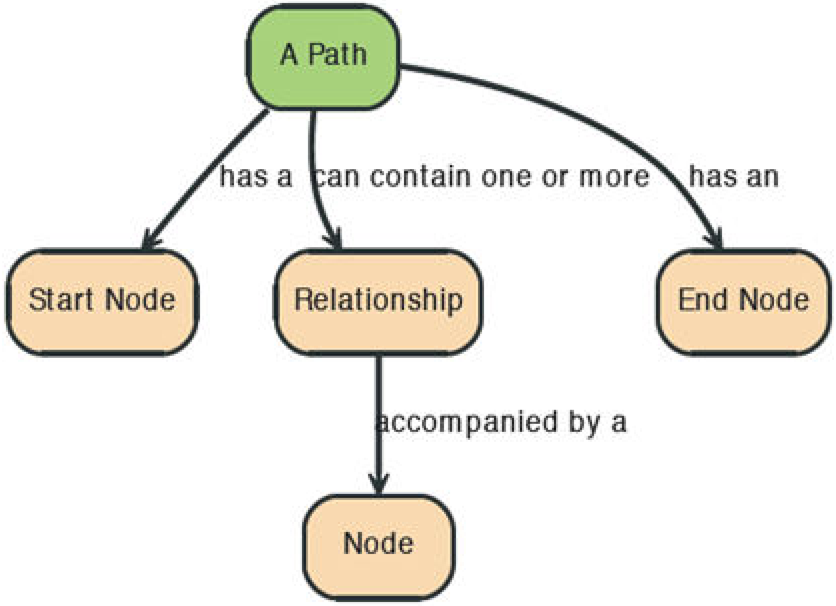
Properties

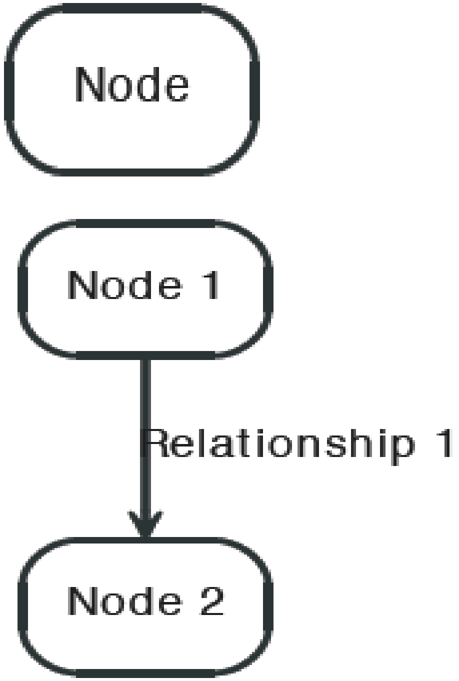


A Property
has a
has a
Value
Key
can be an array of
can be a
is a
Primitive
boolean
byte
short
int
long
float
double
char
String

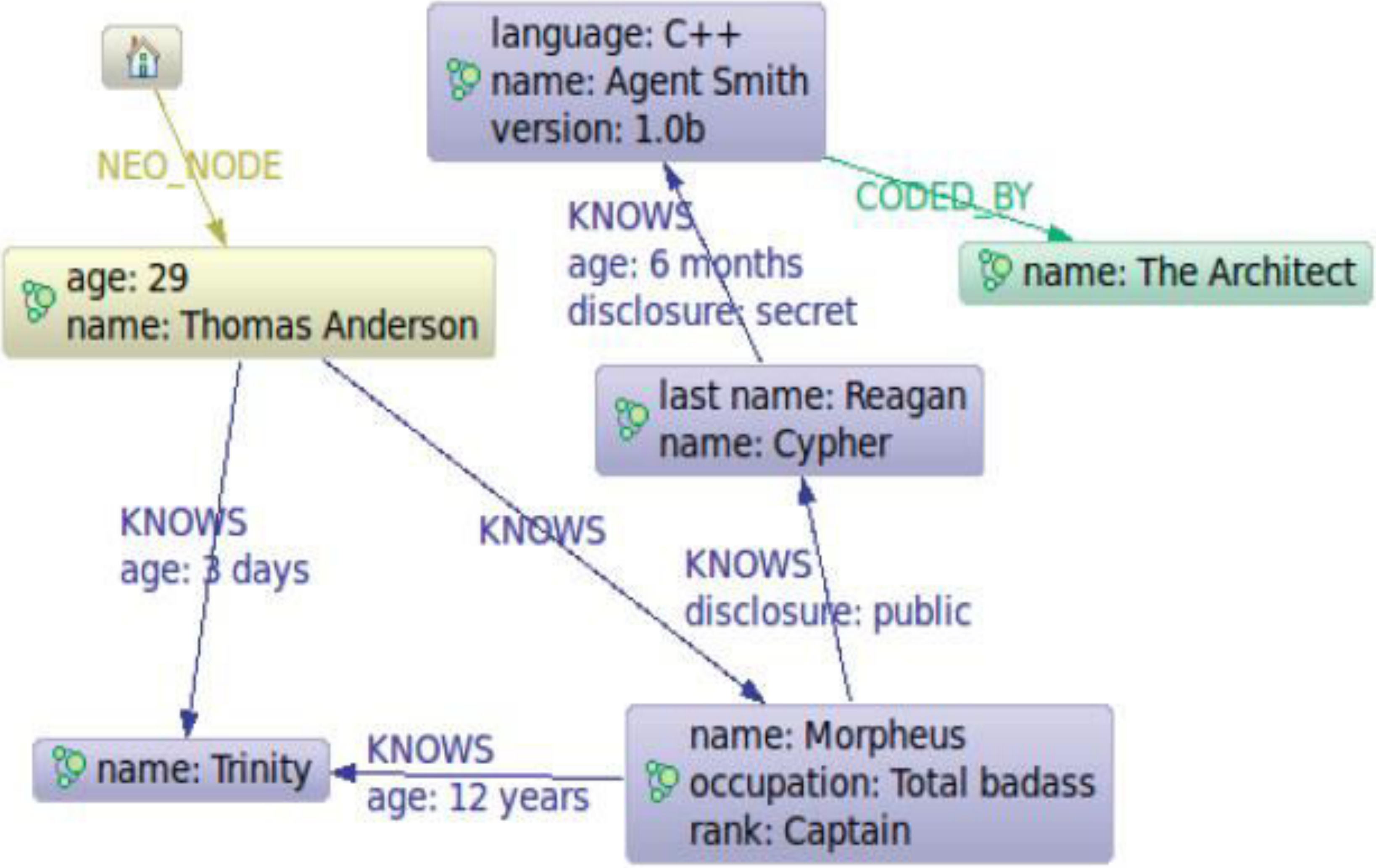
Paths in Neo4j

- A path is one or more nodes with connecting relationships, typically retrieved as a query or traversal result.





The Matrix Graph Database



Cassandra - A Decentralized Structured Storage System

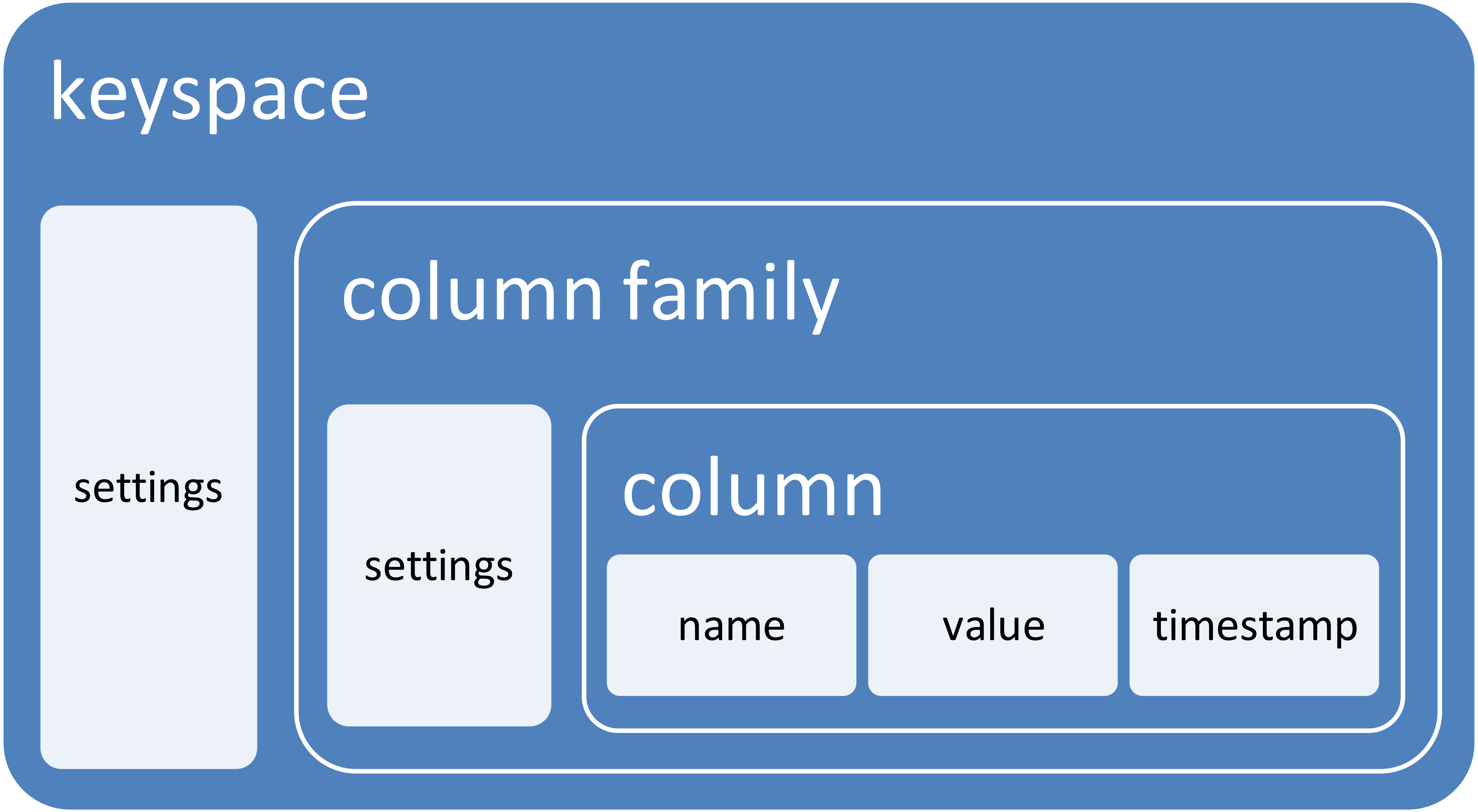
Cassandra

- Extension of Bigtable with aspects of Dynamo
- Motivations:
  - High Availability
  - High Write Throughput
  - Fail Tolerance

Data Model

- Table is a multi dimensional map indexed by key (row key).
- Columns are grouped into Column Families.
- 2 Types of Column Families
  - Simple
  - Super (nested Column Families)
- Each Column has
  - Name
  - Value
  - Timestamp

Data Model



* Figure taken from Eben Hewitt's (author of Oreilly's Cassandra book) slides.

System Architecture

- Partitioning

  How data is partitioned across nodes

- Replication

  How data is duplicated across nodes

- Cluster Membership

  How nodes are added, deleted to the cluster

Partitioning

- Nodes are logically structured in Ring Topology.
- Hashed value of key associated with data partition is used to assign it to a node in the ring.
- Hashing rounds off after certain value to support ring structure.

- Lightly loaded nodes moves position to alleviate highly loaded nodes.

Replication

- Each data item is replicated at N (replication factor) nodes.

- Different Replication Policies
  - **Rack Unaware** – replicate data at N-1 successive nodes after its coordinator
  - **Rack Aware** – uses ‘Zookeeper’ to choose a leader which tells nodes the range they are replicas for
  - **Datacenter Aware** – similar to Rack Aware but leader is chosen at Datacenter level instead of Rack level.

Gossip Protocols

- Network Communication protocols inspired for real life rumour spreading.
- Periodic, Pairwise, inter-node communication.
- Low frequency communication ensures low cost.
- Random selection of peers.
- Example – Node A wish to search for pattern in data
  - Round 1 – Node A searches locally and then gossips with node B.
  - Round 2 – Node A,B gossips with C and D.
  - Round 3 – Nodes A,B,C and D gossips with 4 other nodes ……
- Round by round doubling makes protocol very robust.

Gossip Protocols

- Variety of Gossip Protocols exists
  - **Dissemination protocol**
    - Event Dissemination: multicasts events via gossip. high latency might cause network strain.
    - Background data dissemination: continuous gossip about information regarding participating nodes
  - **Anti Entropy protocol**
    - Used to repair replicated data by comparing and reconciling differences. This type of protocol is used in Cassandra to repair data in replications.

Cluster Management

- Uses Scuttleback (a Gossip protocol) to manage nodes.
- Uses gossip for node membership and to transmit system control state.
- Node Fail state is given by variable ‘phi’ which tells how likely a node might fail (suspicion level) instead of simple binary value (up/down).
- This type of system is known as Accrual Failure Detector.

Accural Failure Detector

- If a node is faulty, the suspicion level monotonically increases with time.
- $\Phi(t)$ ⍰ k as t ⍰ k
- Where k is a threshold variable (depends on system load) which tells a node is dead.
- If node is correct, phi will be constant set by application. Generally
- $\Phi(t) = 0$

Bootstrapping and Scaling

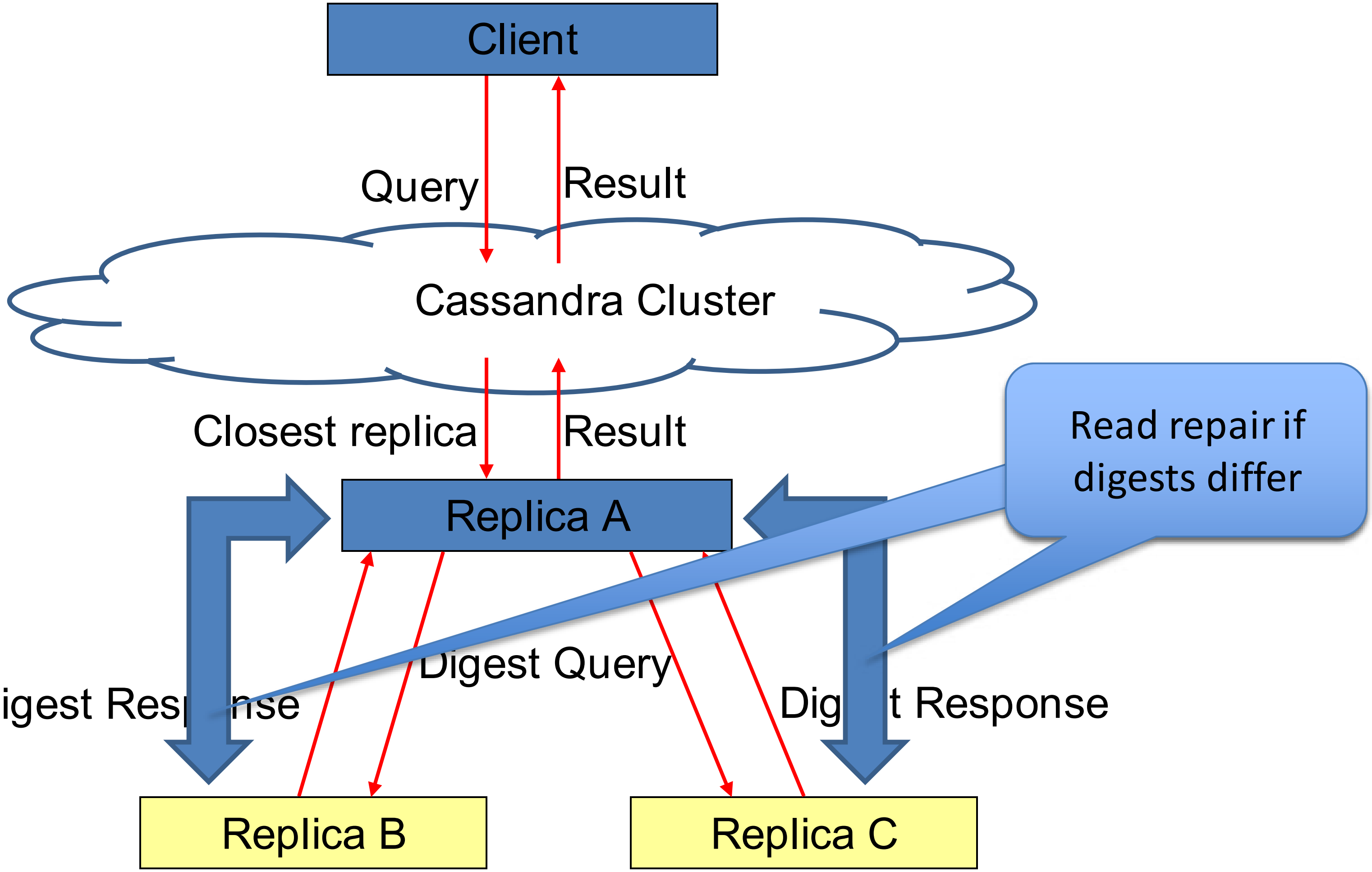
- Two ways to add new node
- New node gets assigned a random token which gives its position in the ring. It gossips its location to rest of the ring
- New node reads its config file to contact it initial contact points.
- New nodes are added manually by administrator via CLI or Web interface provided by Cassandra.

- Scaling in Cassandra is designed to be easy.
- Lightly loaded nodes can move in the ring to alleviate heavily loaded nodes.

Local Persistence

- Relies on local file system for data persistency.
- Write operations happens in 2 steps
- Write to commit log in local disk of the node
- Update in-memory data structure.
- Why 2 steps or any preference to order or execution?
- Read operation
- Looks up in-memory ds first before looking up files on disk.
- Uses Bloom Filter (summarization of keys in file store in memory) to avoid looking up files that do not contain the key.

Read Operation



Read repair if digests differ

* Figure taken from Avinash Lakshman and Prashant Malik (authors of the paper) slides.

Facebook Inbox Search

- Cassandra developed to address this problem.
- 50+TB of user messages data in 150 node cluster on which Cassandra is tested.
- Search user index of all messages in 2 ways.
- Term search : search by a key word
- Interactions search : search by a user id

Latency Stat	Search Interactions	Term Search
Min	7.69 ms	7.78 ms
Median	15.69 ms	18.27 ms
Max	26.13 ms	44.41 ms

Comparison with MySQL

- MySQL > 50 GB Data
  Writes Average : ~300 ms
  Reads Average : ~350 ms

- Cassandra > 50 GB Data
  Writes Average : 0.12 ms
  Reads Average : 15 ms

- Stats provided by Authors using facebook data.