



Mahidol University
Wisdom of the Land

BIG DATA PROCESSING

Graph Network

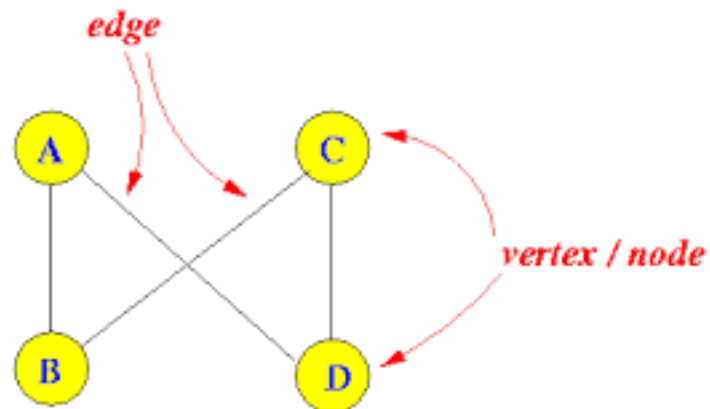
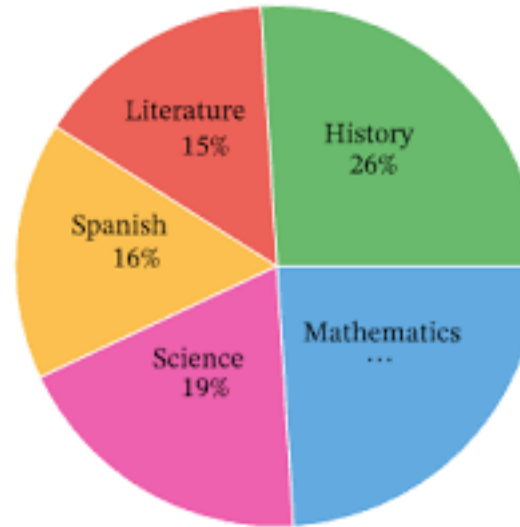


What is Graph

- Graph is not a chart!!
- Graph
 - Vertices
 - Edges

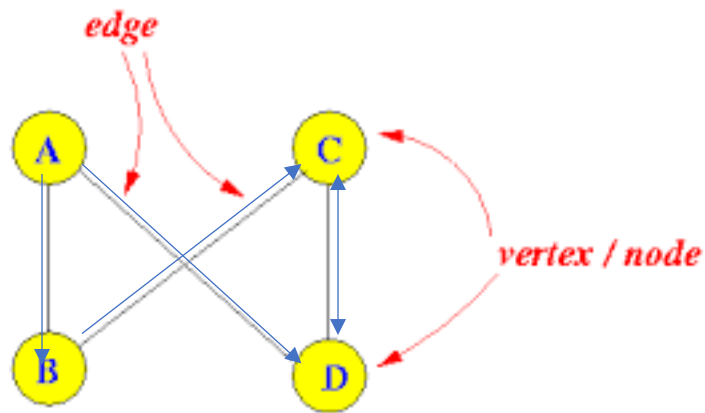
$V = \{A, B, C, D\}$

$E = \{(A, D), (A, B), (C, D), (C, B)\}$



Data structure operation

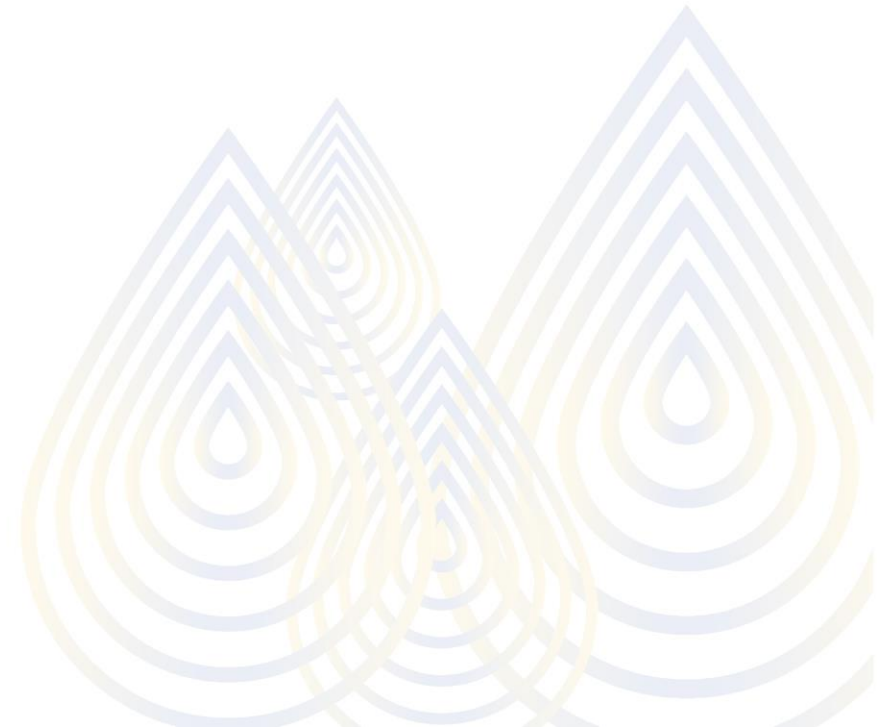
- Add edge
- Add vertex
- Get neighbor (and etc)



	A	B	C	D
A	0	1	0	1
B	0	0	1	0
C	0	0	0	1
D	0	0	1	0

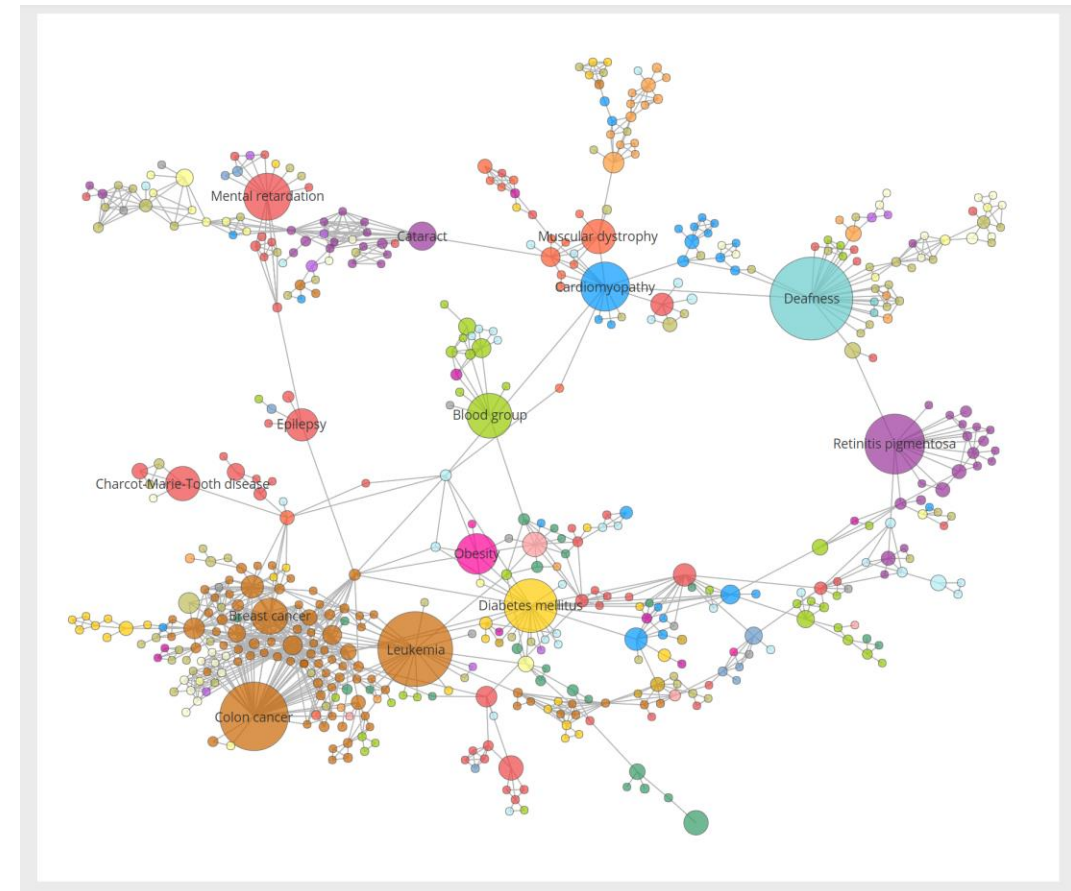
Real World graph problem

- Social Media
 - Vertex/ Node : User
 - Edges: Tweet/post
 - Benefits
 - What are the interactions?
 - Who is the influencer?
 - Does it show violent behaviour?
 - Is this person addicted to online social media?



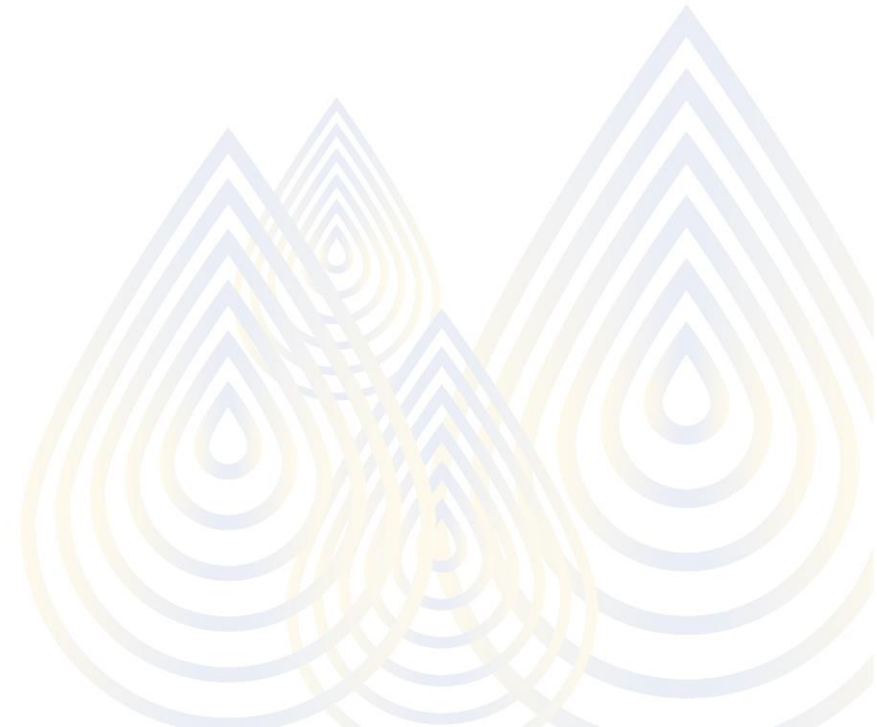
Real World graph problem 2

- Biology Network
 - Gene-Protein relationship
 - Gene-gene interaction
 - cell-cell signaling
 - Genes-disease relationship
 - Human knowledges
- Discovery unknown connection
 - Gene \rightarrow 1 disease



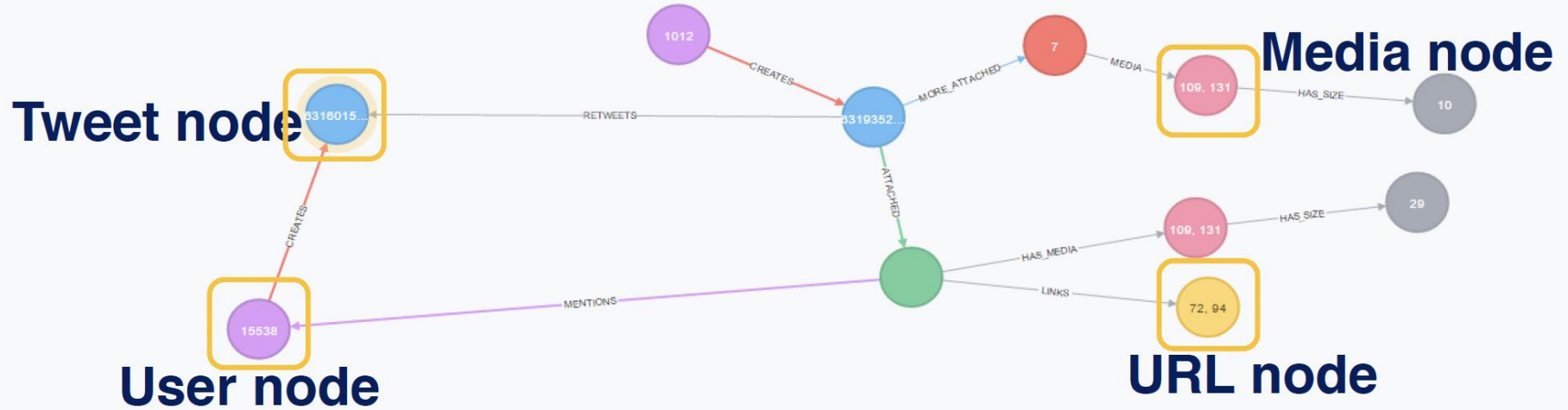
Real World graph problem3

- Human Connection
 - Connection connectivity
 - Example of information can be added
 - LinkIn
 - Telephone network
 - using Calendar Event
 - Business transaction
 - GPS signals
 - Benefits
 - Influencer analysis
 - Match making (job,interest, so on)
 - Thread detection



Graph Analytics

Node types



Tweet <id>: 15 idStr: 631601575551602688 createdAt: 1439420524000 lang: en retweeted: false source: TweetDeck filterLevel: low truncated: false
text: We've just posted a sneak preview of some upcoming WoW pets and mounts! http://t.co/2CcECmio4b http://t.co/xTpnlbvsh3 possiblySensitive: false tweetId: 631601575551602700 retweetCount: 489 favorited: false favoriteCount: 786

V: a set of vertices

E: a set of edges

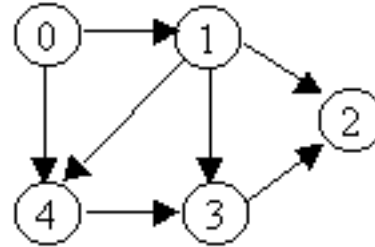
TN: a set of node types

f (TN-> V): type assignment to nodes

Weight

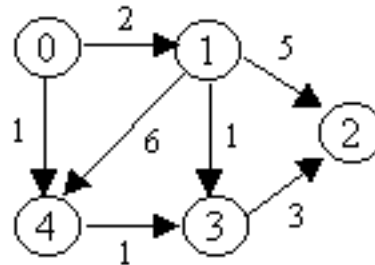
- Edge Property

- Distance



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

- Strenght of connection



$$A = \begin{bmatrix} \infty & 2 & \infty & \infty & 1 \\ \infty & \infty & 5 & 1 & 6 \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 3 & \infty & \infty \\ \infty & \infty & \infty & 1 & \infty \end{bmatrix}$$

- Likelihood of interaction (Biology)

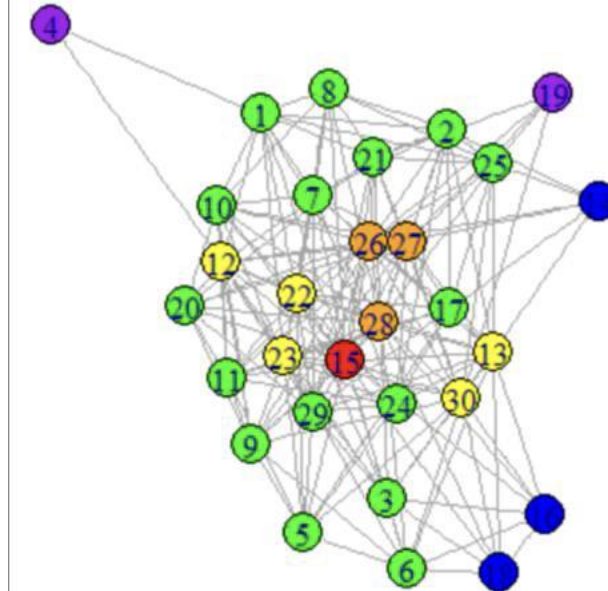
- Certainty of information (knowledge network)

Degree Centrality

- Count of the number of edges **incident**, normalized by the possible number of edges

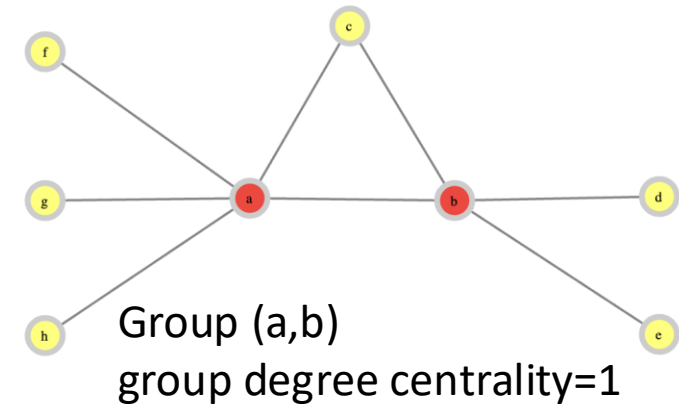
$$(\# \text{ of edges}) / (N-1)$$

- “hub” (maximally-connected) nodes



Group Degree Centrality

- Consider a group as a single entity



- Count of the number of edges incident within group, normalized by non group

$$\frac{\#edges\ in\ the\ group}{\#non - group}$$

Closeness Centrality

Average of shortest-path distances from all other nodes

- Low raw closeness means node has short distance from other nodes

For an information flow network

- Low closeness nodes receive information sooner than other nodes
- Same for other flows *if the flow happens through shortest paths*
- A low closeness can influence many others, directly and indirectly

Node	Shortest Path from A
B	1
C	2
D	3
E	4
F	5
G	5
H	4

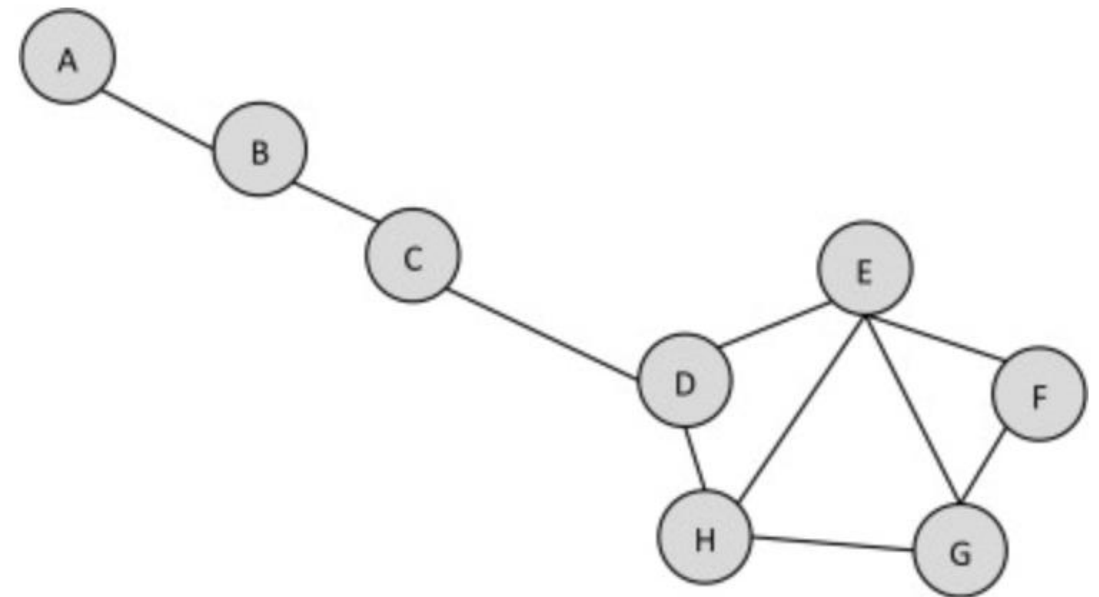
Here, the average shortest path length is:

$$(1 + 2 + 3 + 4 + 5 + 5 + 4) \div 7 = 24 \div 7 = \mathbf{3.43}.$$

Node	Shortest Path from D
A	3 (D-C-B-A)
B	2
C	1
E	1
F	2
G	2
H	1

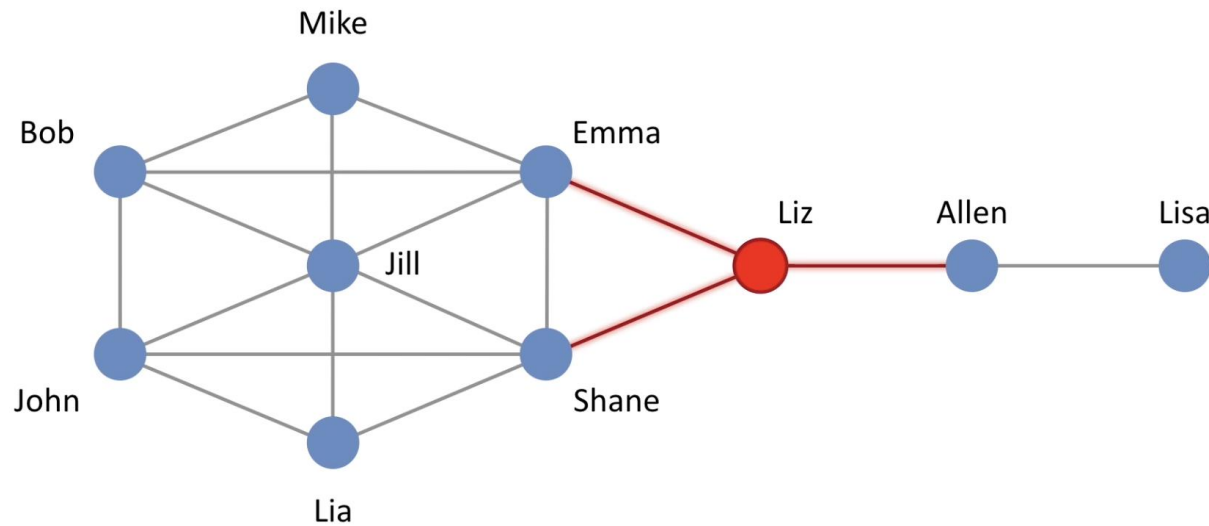
The average of those shortest path lengths is:

$$(3 + 2 + 1 + 1 + 2 + 2 + 1) \div 7 = 12 \div 7 = \mathbf{1.71}.$$



To inject a new piece of information into the network with the idea that it should read every other node quickly

Betweenness Centrality



Not suitable for

- Any quantity that does not flow in shortest path channels like **infection** or **rumor** on the internet doesn't work well with betweenness centrality.

- Ratio of pairwise shortest path that pass through that node
- Measures fraction of shortest-path commodity flow passing through a node
- Junction path in the network
- high betweenness centrality is often **controllers** of power or information

Community Analytics

- Static Analyses
 - What are the communities at time T ?
 - Who belong to a community?
 - How closely knit is this community?
- Temporal/Evolution Analyses
 - How did this community form?
 - Which communities are stable?
 - Find strong *transient* communities – why did they form or dissolve?
- Predictive Analyses
 - Is this community likely to grow?
 - Will these nodes continue as a community in future?
 - Are dominant roles emerging in this community?



Neo4J

- <https://console.neo4j.io/#databases>

Let's create your first database

Step 1 of 2

Database type less detail ^

AuraDB Free

For learning, prototyping and exploring Neo4j in a forever free instance with basic requirements.

- 1 forever free database
- Limits on graph size (50k nodes, 175k relationships)
- Standard procedure library (apoc-core)
- Auto-pause after 3 days of inactivity

You have reached the Free database limit

AuraDB Professional

For production applications with scalability, disaster recovery and advanced development needs.

- Unlimited databases
- Starting at 1 GB of RAM
- Scalable on-demand
- Available in all global regions
- Daily backups with 7-day retention
- On-demand snapshots

Starting at
\$65/month

Instance details

Instance Name

TEST

GCP Region

Singapore (asia-southeast1)

Starting dataset



Learn about graphs
with a movie dataset

Beginner



Load or create your
own data in a blank
instance

× Cancel

✓ Create Instance

Database connection

\$:server connect

Connect to Neo4j

Database access might require an authenticated connection

Connect URL

neo4j+s:// ▼ 6e80c064.databases.neo4j.io:7687

Authentication type

Username / Password ▼

Username

neo4j

Password

.....

Connect

Creating Nodes: Cypher

:guide movie-graph

- CREATE (TheMatrix:Movie {title:'The Matrix', released:1999, tagline:'Welcome to the Real World'})
- CREATE (Keanu:Person {name:'Keanu Reeves', born:1964})
- CREATE (Carrie:Person {name:'Carrie-Anne Moss', born:1967})
- CREATE (Keanu)-[:ACTED_IN {roles:['Neo']}]>(TheMatrixReloaded),
- (Carrie)-[:ACTED_IN {roles:['Trinity']}]>(TheMatrixReloaded),
- (Laurence)-[:ACTED_IN {roles:['Morpheus']}]>(TheMatrixReloaded),
- (Hugo)-[:ACTED_IN {roles:['Agent Smith']}]>(TheMatrixReloaded),
- (LillyW)-[:DIRECTED]>(TheMatrixReloaded),
- (LanaW)-[:DIRECTED]>(TheMatrixReloaded),

← Neo4j Browser Guides

Movie Graph Guide

The Movie Graph is a mini graph application, containing actors and directors that are related through the movies they have collaborated on.

This guide shows how to:

Load: Insert movie data into the graph.

Constrain: Create unique node property constraints.

Index: Index nodes based on their labels.

Find: Retrieve individual movies and actors.

Query: Discover related actors and directors.

Solve: The Bacon Path.

Query some data

- // Get some data
- MATCH (n1)-[r]->(n2) RETURN r, n1, n2 LIMIT 25
- MATCH (tom {name: "Tom Hanks"}) RETURN tom

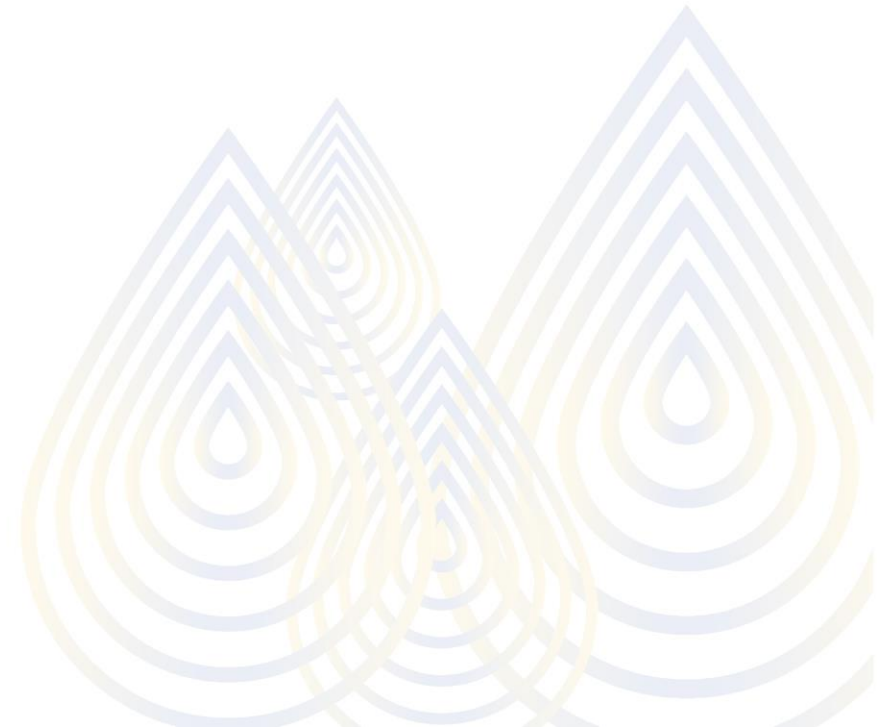
```
neo4j$ MATCH (people:Person) RETURN people.name LIMIT 10
```

	people.name
1	"Keanu Reeves"
2	"Carrie-Anne Moss"
3	"Laurence Fishburne"
4	"Hugo Weaving"
5	"Lilly Wachowski"
6	"Lana Wachowski"
7	

Started streaming 10 records after 1 ms and completed after 3 ms.

Deleting

- **Delete all nodes and edges**
 - match (n)-[r]-() delete n, r
- **Delete all nodes which have no edges**
 - match (n) delete n
- **Delete all edges**
 - match (n)-[r]-() delete r
- **Delete only ToyRelation edges**
 - match (n)-[r:ToyRelation]-() delete r



Simple Query

Counting the number of nodes

```
match (n:MyNode)
return count(n)
```

Counting the number of edges

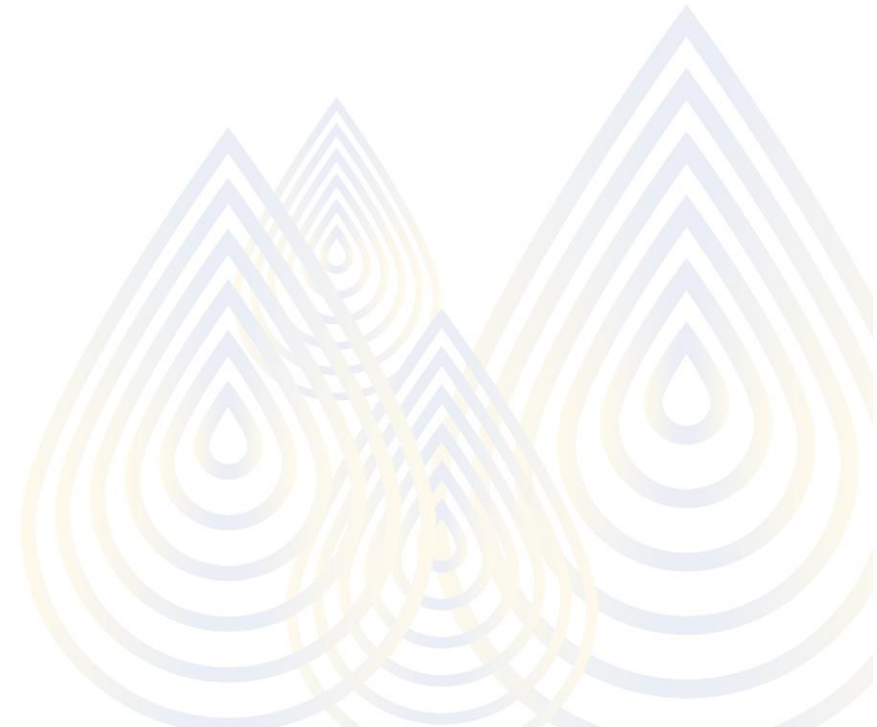
```
match (n:MyNode) - [r] -> ()
return count(r)
```

Finding the types of a node:

```
match (n)
```

Finding root nodes:

```
match (m) - [r:TO] -> (n:MyNode)
where not ( () --> (m) )
return m
```



Simple Query

Finding leaf nodes:

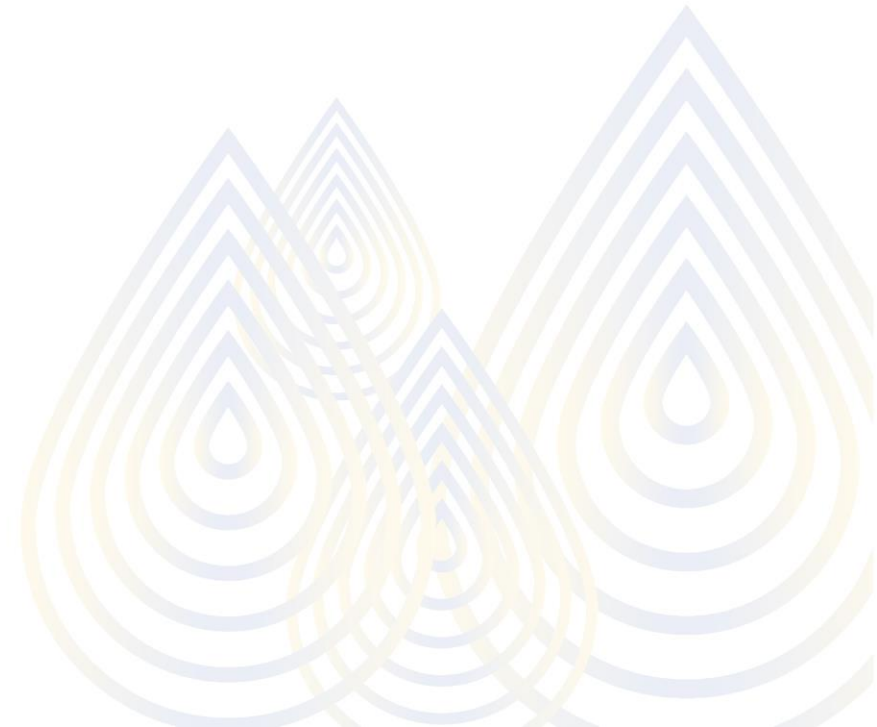
```
match (n:MyNode) -[r:TO] -> (m)
where not ((m) --> ())
return m
```

Finding triangles:

```
match (a) -[:TO] -> (b) -[:TO] -> (c) -[:TO] -> (a)
return distinct a, b, c
```

Finding 2nd neighbors of D:

```
match (a) -[:TO*..2] - (b)
where a.Name='D'
return distinct a, b
where n.Name = 'Afghanistan'
return labels(n)
```



Analytics

- **Viewing the graph**

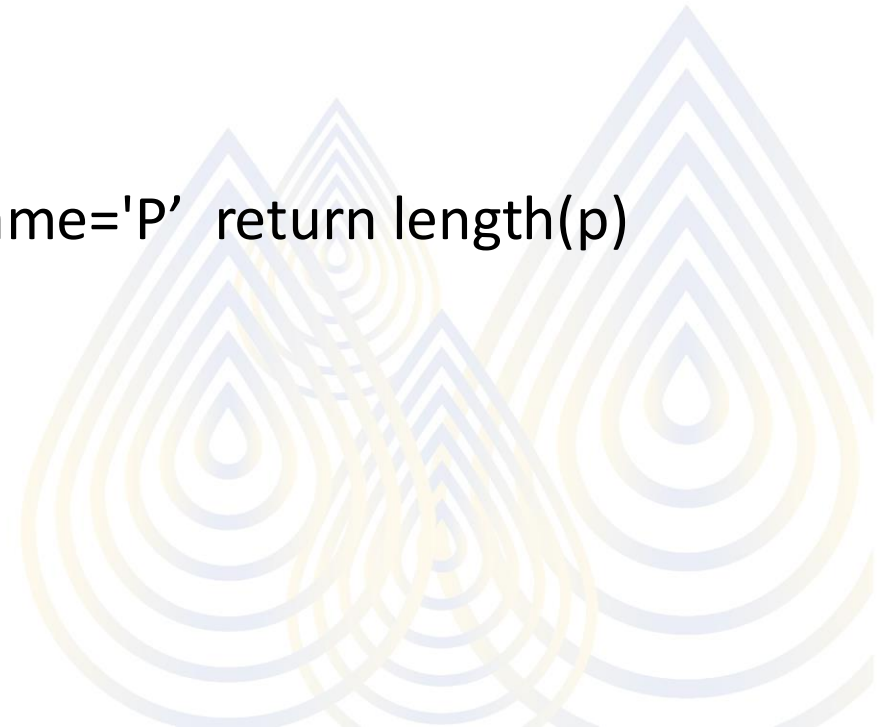
- `match (n:MyNode)-[r]->(m) return n, r, m`

- **Finding paths between specific nodes*:**

- `match p=(a)-[:TO*]-(c) where a.Name='H' and c.Name='P' return p limit 1`
- `match p=(a)-[:TO*]-(c) where a.Name='H' and c.Name='P' return p order by length(p) asc limit 1`

- **Finding the length between specific nodes:**

- `match p=(a)-[:TO*]-(c) where a.Name='H' and c.Name='P' return length(p) limit 1`



Analytics

- **Finding a shortest path between specific nodes:**

- `match p=shortestPath((a)-[:TO*]-(c)) where a.Name='A' and c.Name='P'`
`return p, length(p) limit 1`

- **All Shortest Paths:**

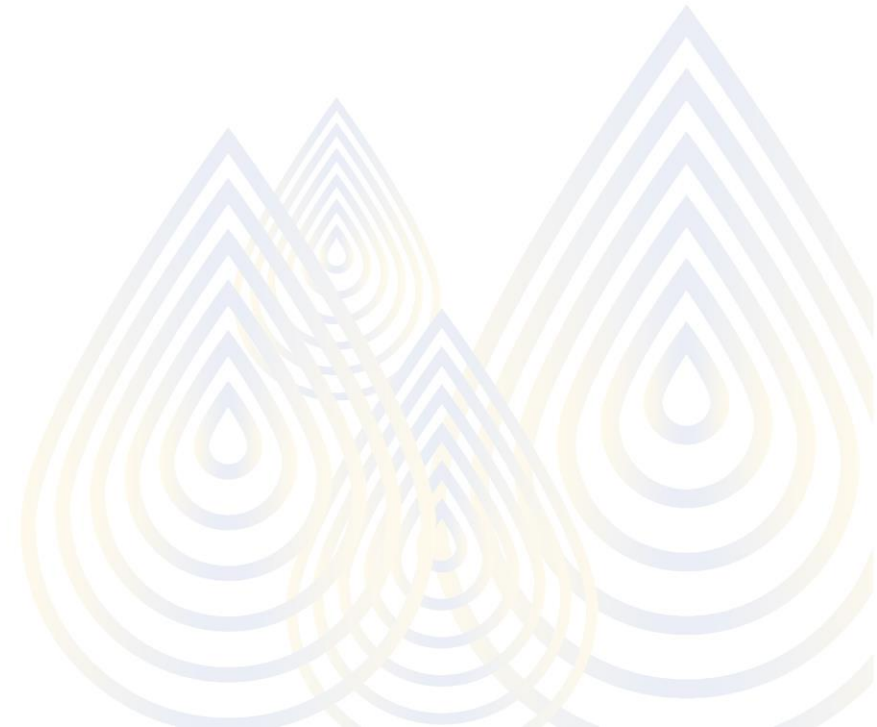
- `MATCH p = allShortestPaths((source)-[r:TO*]-(destination)) WHERE`
`source.Name='A' AND destination.Name = 'P' RETURN EXTRACT(n IN`
`NODES(p) | n.Name) AS Paths`

- **All Shortest Paths with Path Conditions:**

- `MATCH p = allShortestPaths((source)-[r:TO*]->(destination)) WHERE`
`source.Name='A' AND destination.Name = 'P' AND LENGTH(NODES(p)) > 5`
`RETURN EXTRACT(n IN NODES(p) | n.Name) AS Paths,length(p)`

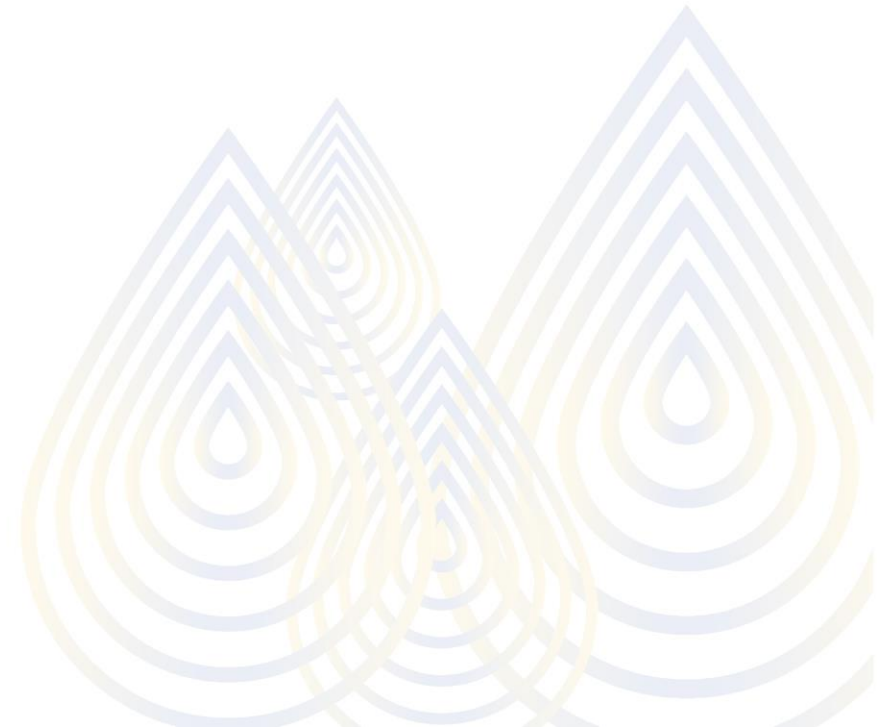
Example

- `match p=shortestPath((a:Person)-[:ACTED_IN*]-(c:Person)) where a.name='Hugo Weaving' and c.name='Keanu Reeves' return p, length(p)`



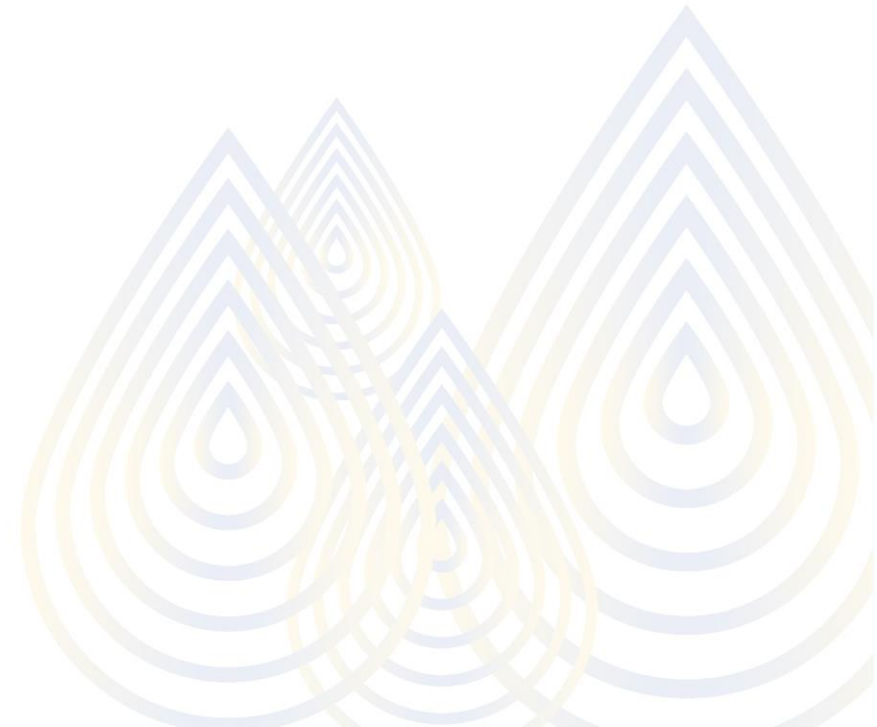
Small data

- Experiment
- Import
- Check nodes/edges
- Shortest Path



Deleting networks

- MATCH (n) DETACH DELETE n
 - Deleted 178 nodes, deleted 259 relationships, completed after 70 ms.



Experiment

Create 5 Nodes & relationship

- `create (N1:ToyNode {name: 'Tom'}) - [:ToyRelation {relationship: 'knows'}] -> (N2:ToyNode {name: 'Harry'}),`
- `(N2) - [:ToyRelation {relationship: 'co-worker'}] -> (N3:ToyNode {name: 'Julian', job: 'plumber'}),`
- `(N2) - [:ToyRelation {relationship: 'wife'}] -> (N4:ToyNode {name: 'Michele', job: 'accountant'}),`
- `(N1) - [:ToyRelation {relationship: 'wife'}] -> (N5:ToyNode {name: 'Josephine', job: 'manager'}),`
- `(N4) - [:ToyRelation {relationship: 'friend'}] -> (N5);`

Querying

- **Five Nodes**

- N1 = Tom, N2 = Harry, N3 = Julian, N4 = Michele, N5 = Josephine

- **Five Edges**

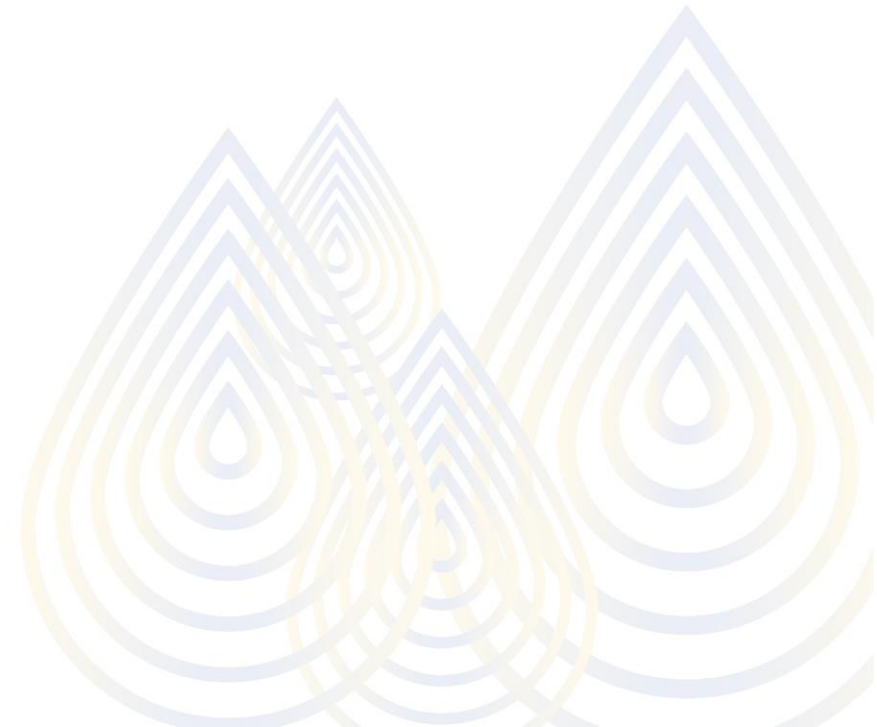
- e1 = Harry 'is known by' Tom
- e2 = Julian 'is co-worker of' Harry
- e3 = Michele 'is wife of' Harry
- e4 = Josephine 'is wife of' Tom
- e5 = Josephine 'is friend of' Michele

- **View the resulting graph**

- `match (n:ToyNode)-[r]-(m) return n, r, m`

- **Selecting an existing single ToyNode node**

- `match (n:ToyNode {name:'Julian'}) return n`



Modifying nodes

Adding a Node Correctly

- `match (n:ToyNode {name:'Julian'})`
- `merge (n)-[:ToyRelation {relationship: 'fiancee'}]->(m:ToyNode {name:'Joyce', job:'store clerk'})`

Adding a Node Incorrectly

- `create (n:ToyNode {name:'Julian'})-[:ToyRelation {relationship: 'fiancee'}]->(m:ToyNode {name:'Joyce', job:'store clerk'})`

Modifying nodes

Correct your mistake by deleting the bad nodes and edge

- `match (n:ToyNode {name:'Joyce'})-[r]-() delete n, r`
- `MATCH (n) WHERE size((n)--())=0 DELETE (n)`

Modify a Node's Information

- `match (n:ToyNode) where n.name = 'Harry' set n.job = 'drummer'`
- `match (n:ToyNode) where n.name = 'Harry' set n.job = n.job + ['lead guitarist']`

Connecting to spark

- `import org.neo4j.spark._`
- `import org.apache.spark.graphx._`
- `import org.apache.spark.graphx.lib._`

To connect to the Neo4j database, we need to create Neo4j instance:

Spark-shell:

- `val neo = Neo4j(sc)`

