14A – GRAPH DATABASES

CS 1656

Introduction to Data Science

Alexandros Labrinidis – http://labrinidis.cs.pitt.edu
University of Pittsburgh

The landscape

- Relational Databases
  - Tabular, multi-dimension data
- NoSQL Databases
  - Document stores
  - Column stores
  - Key-value stores
- Graph Databases
  - Graph data

- Relational Databases
  - ACID properties
  - SQL (bad for graphs)
- NoSQL Databases
  - No ACID support
  - Limited query functionality
- Graph Databases
  - New query languages

Queries on Graphs

Graph Databases

- Online database management system
- Support Create, Read, Update, Delete methods
- Use a **graph data model**

- Underlying storage:
  - Native graph storage
  - Non-native

Graph Compute Engines

- Enable **global** graph computation algorithms

- Example:
  - How many relationships on average does everyone in a social network have?

Queries on Graphs

Graph Databases

- Neo4j
  - http://neo4j.com
  - http://console.neo4j.org/
  - April 5, 2016: Neo4j Powers the Biggest Financial Leaks in History – the Tax Haven Scandals Exposed in 'The Panama Papers

Graph Compute Engines

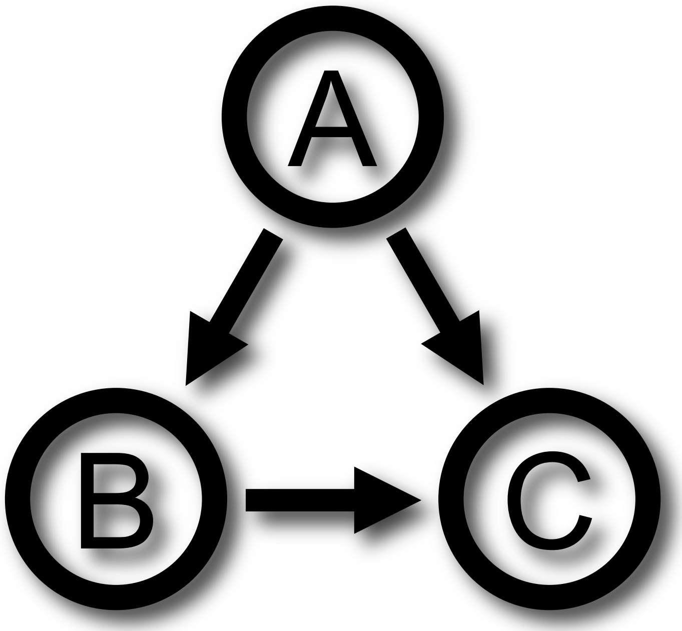
- Apache Giraph
  - http://giraph.apache.org/
- Project Pegasus
  - Graph Mining System
  - http://www.cs.cmu.edu/~pegasus/
- GraphLab Create
  - Machine Learning Framework
  - https://dato.com/products/create/

Neo4j's Cypher

[Slides adapted from Michael Hunger's – Intro to Cypher]

http://www.slideshare.net/jexp/intro-to-cypher?qid=c64ad247-4ee7-43ea-a5e1-d1eb1654429d

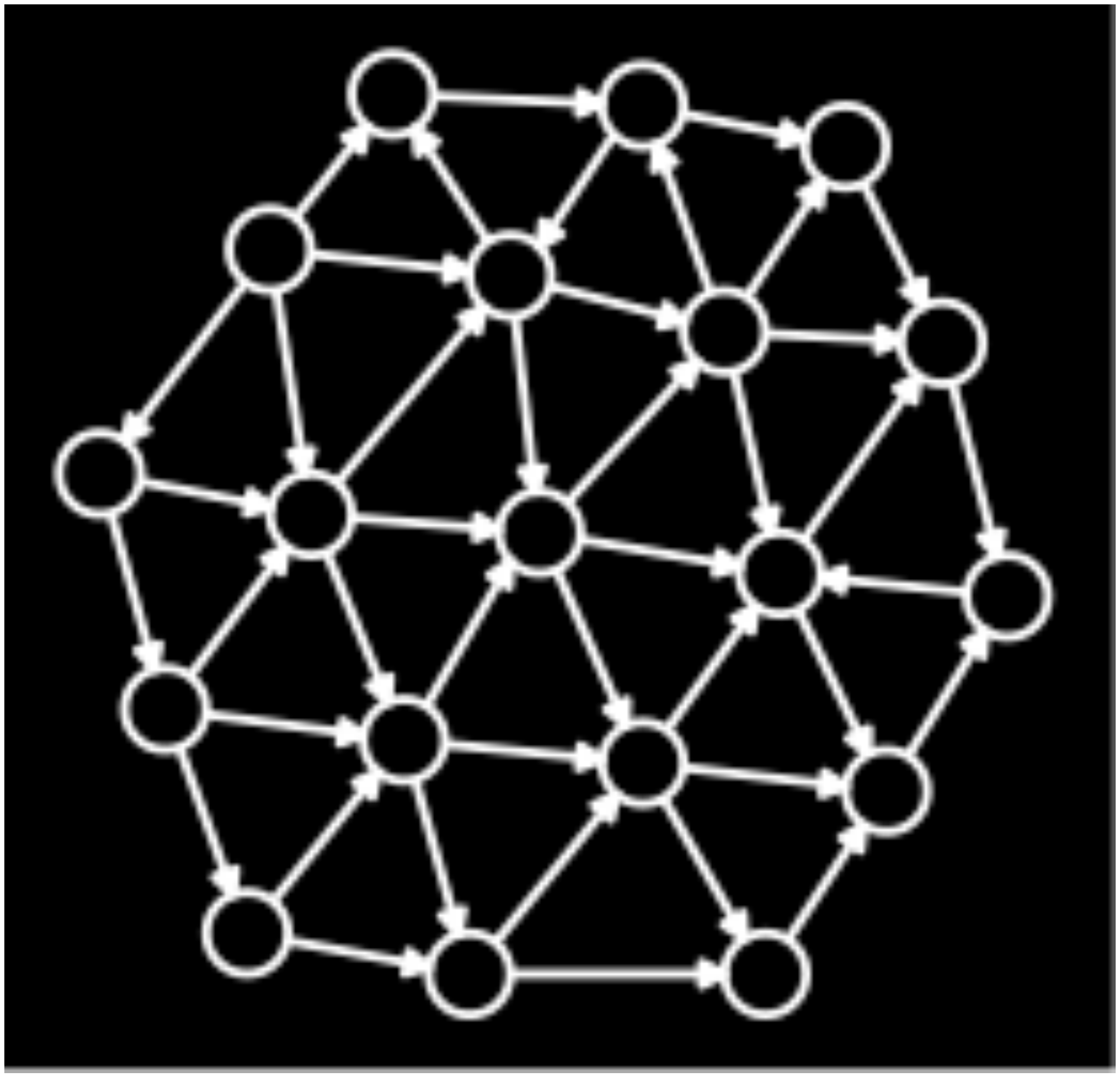
It‘s all about Patterns

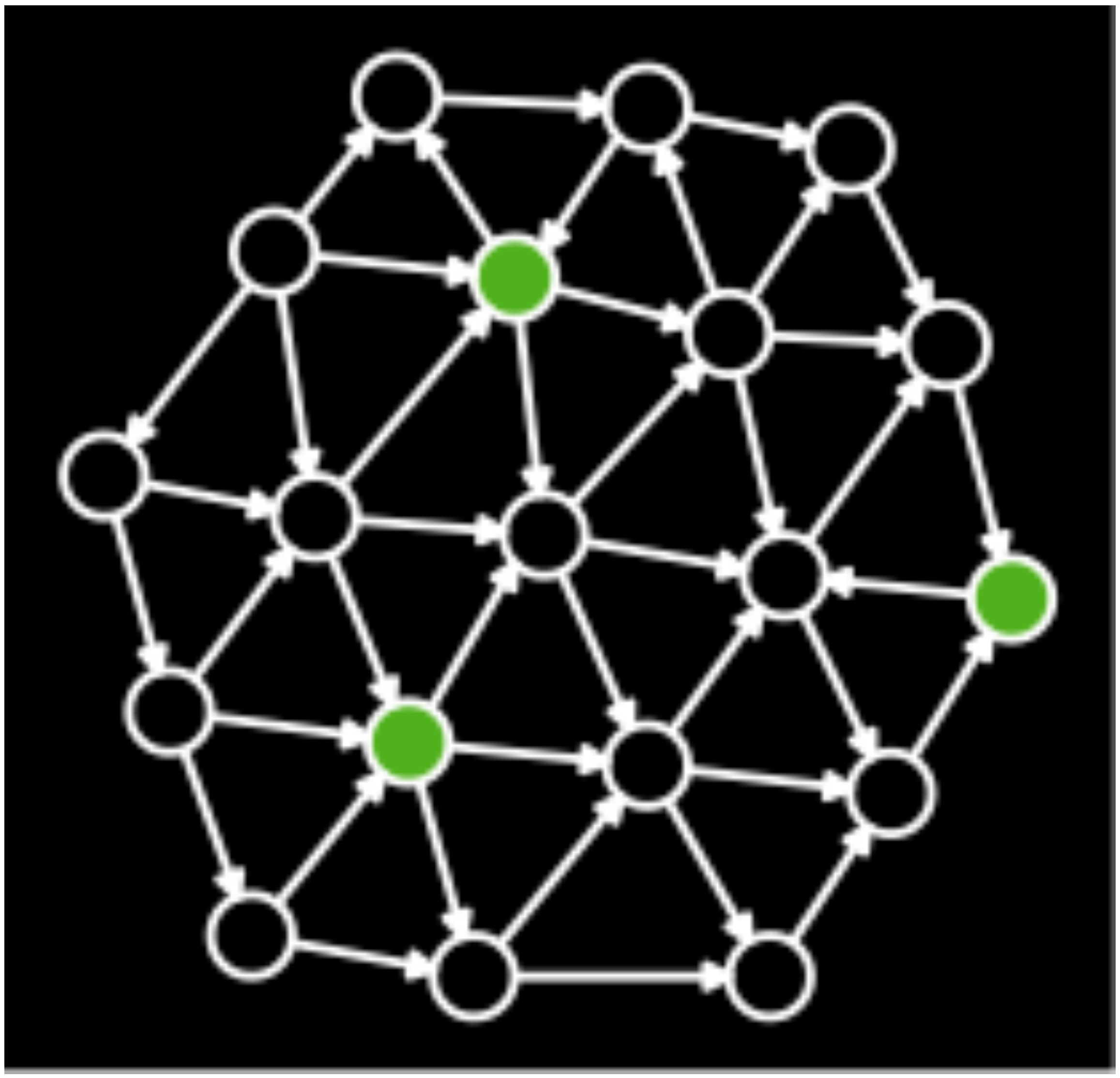


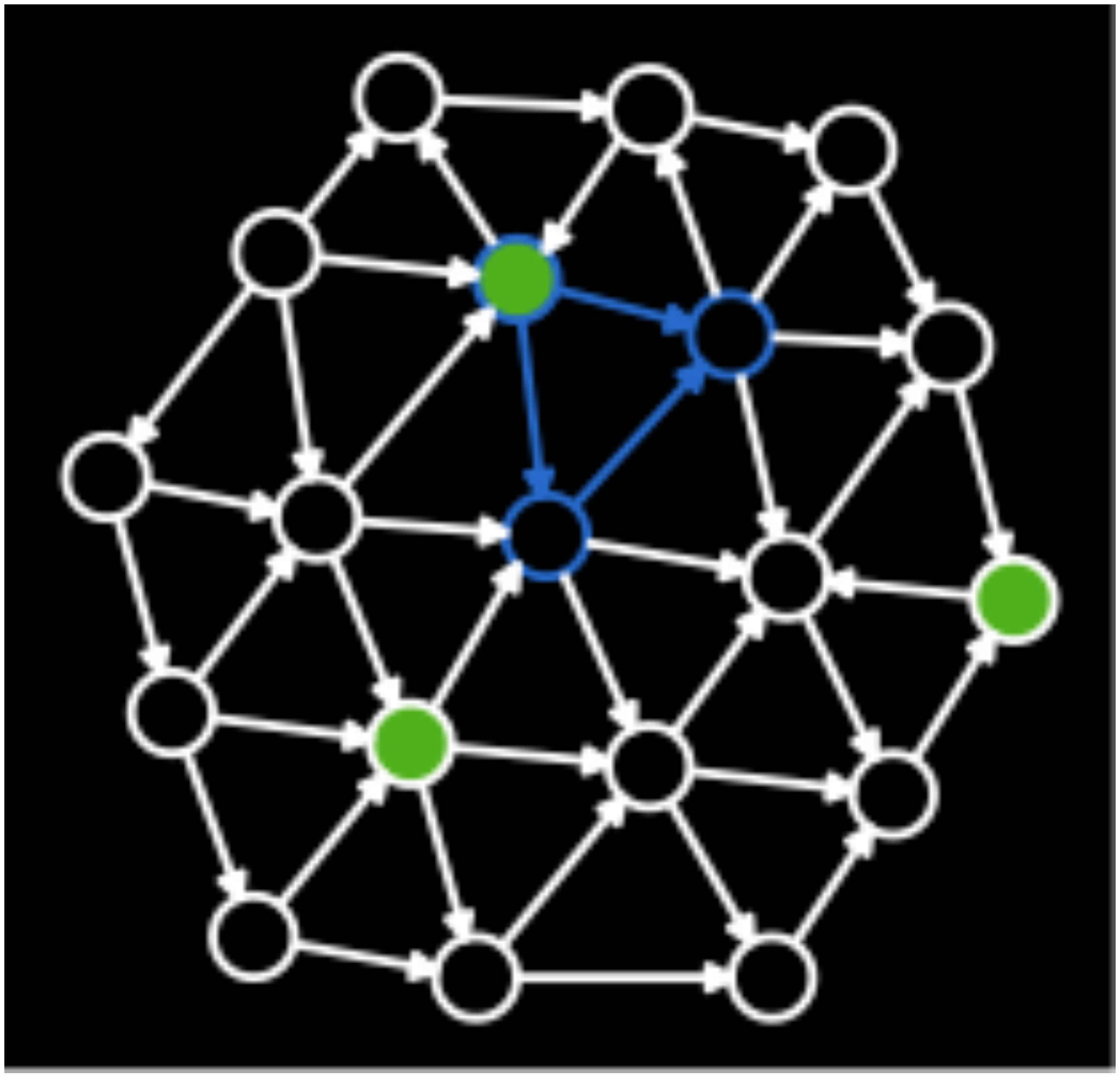
$$\begin{tikzcd} & A \arrow[dl] \arrow[dr] & \\ B \arrow[rr] & & C \end{tikzcd}$$

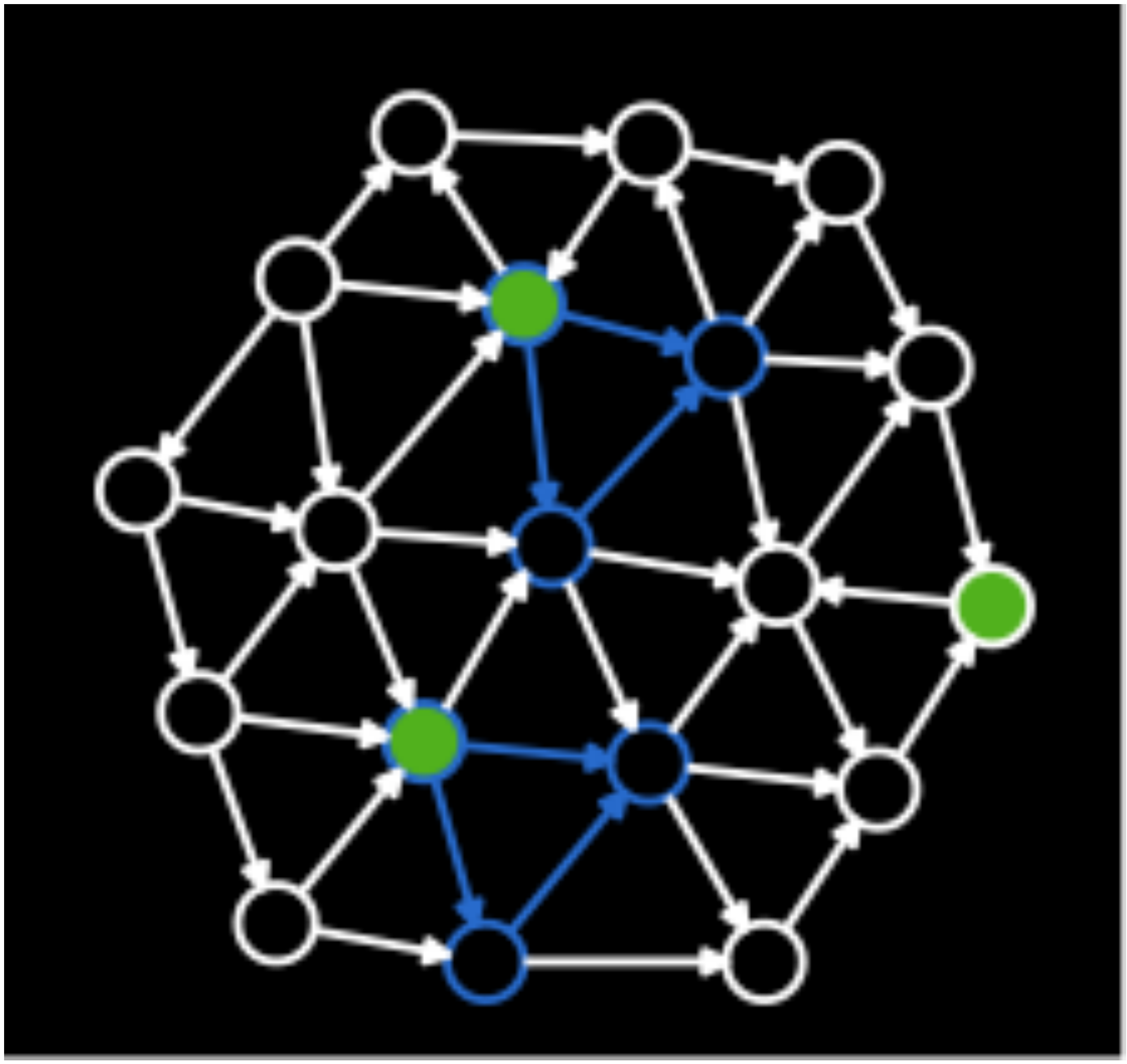
We want to find this Pattern!

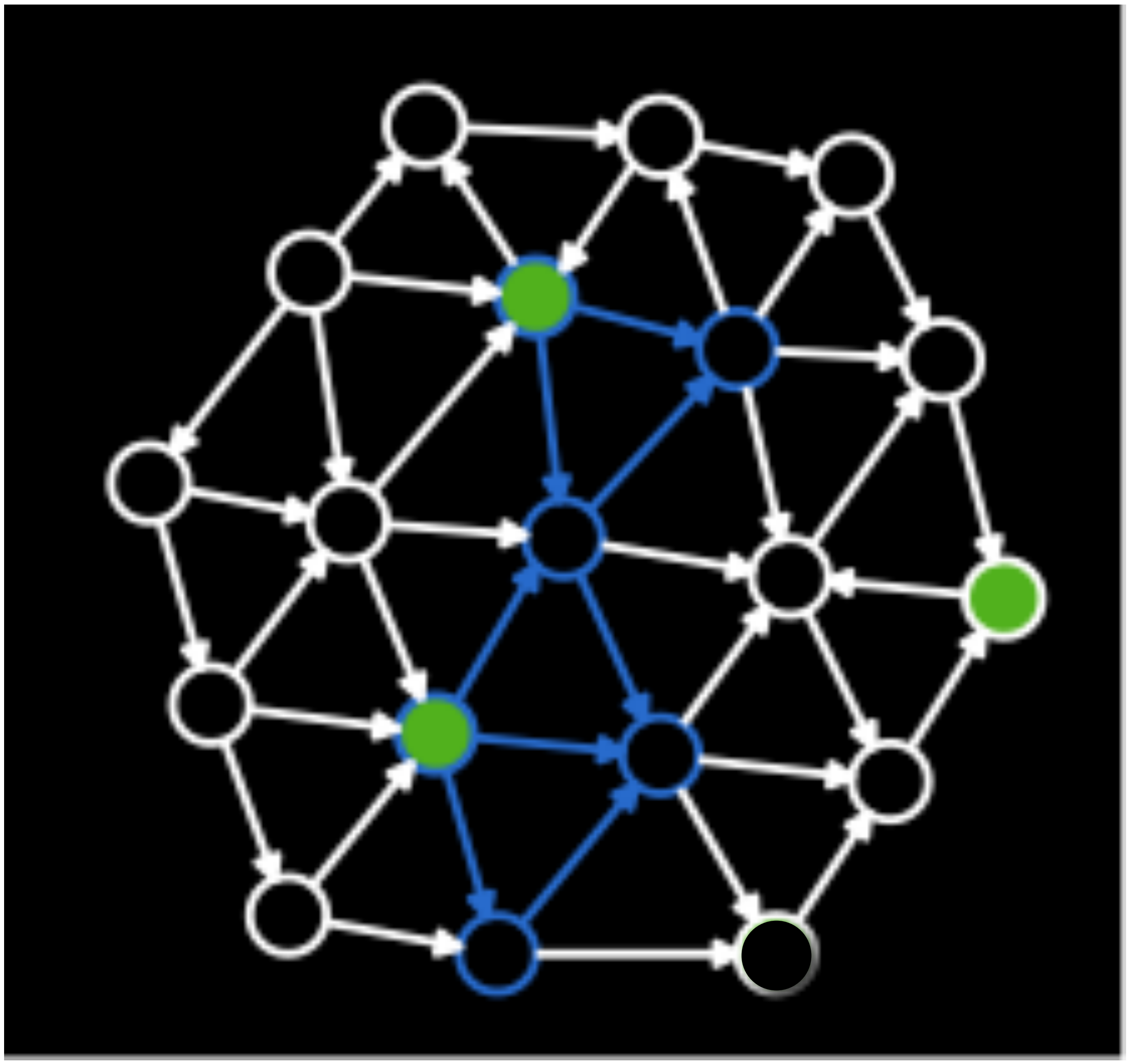
Patterns in a Graph



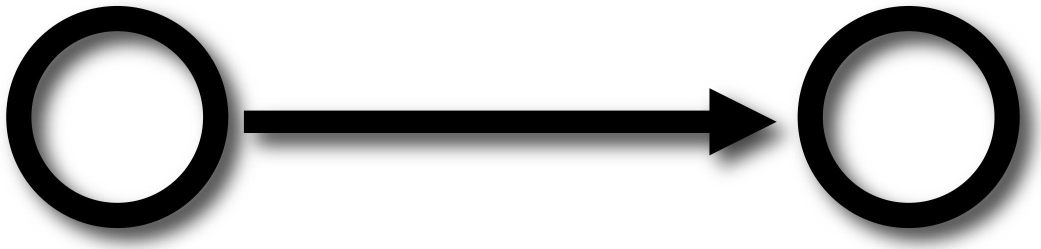






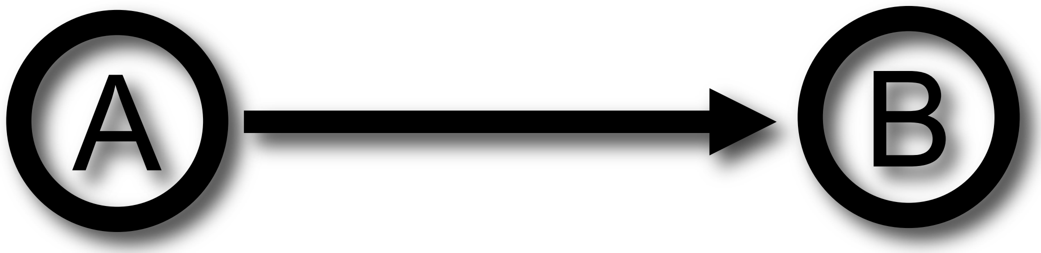


Patterns as ASCII-art



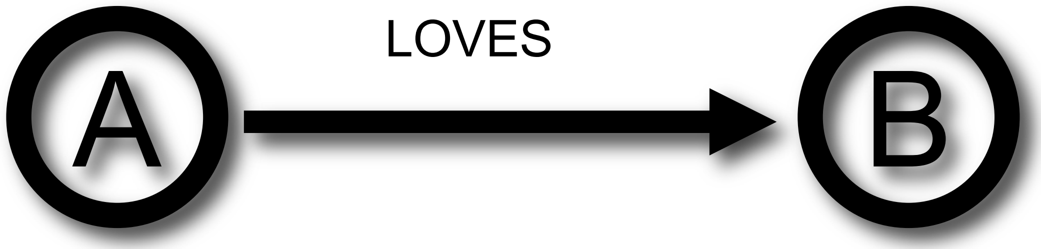
```
()  -->  ()
```

Named Nodes



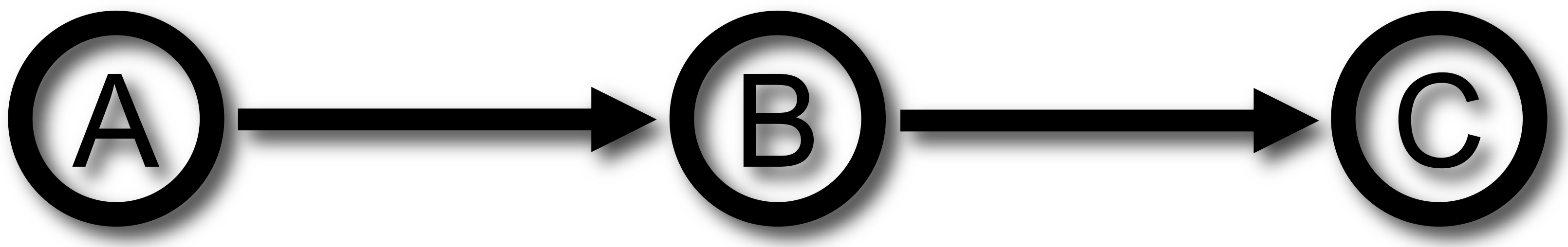
```
(A) --> (B)
```

Named Directed Rels



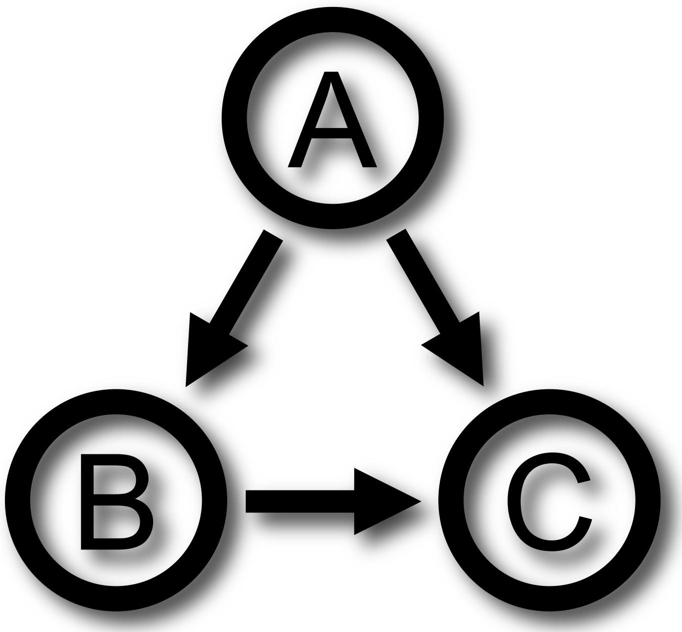
```
A -[:LOVES]-> B
```

Paths



A --> B --> C

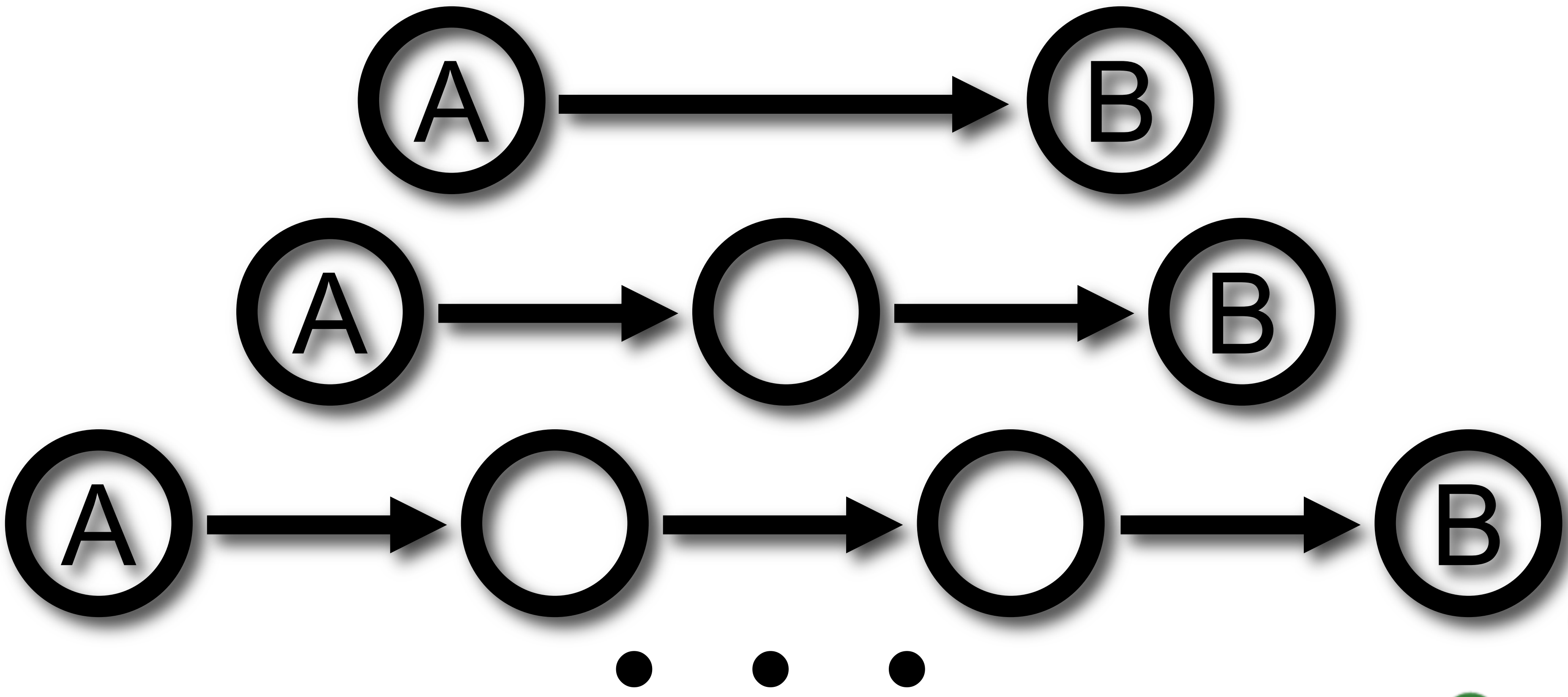
Cyclic-Path-Patterns



A --> B --> C, A --> C

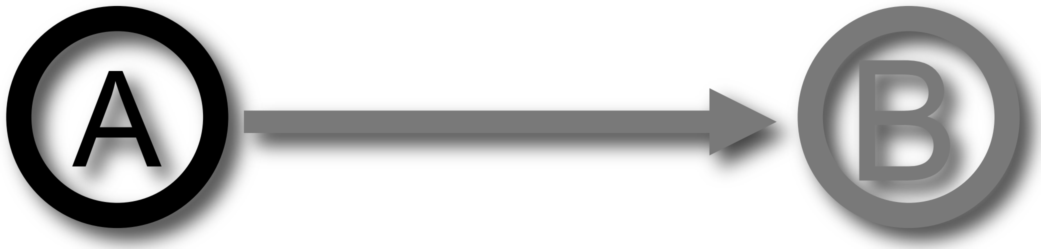
A --> B --> C <-- A

Variable Length Paths



```
A -[*]-> B
```

Optional Relationships



```
A -[?]-> B
```

Cypher Demo

Cypher Demo / SETUP

- Reference: http://neo4j.com/docs/stable/cypher-refcard
- Sandbox: https://neo4j.com/sandbox

- Database:

```
create (Neo:Crew {name:'Neo'}), (Morpheus:Crew {name: 'Morpheus'}),
(Trinity:Crew {name: 'Trinity'}), (Cypher:Crew:Matrix {name: 'Cypher'}),
(Smith:Matrix {name: 'Agent Smith'}), (Architect:Matrix {name:'The Architect'}),
(Neo)-[:KNOWS]->(Morpheus),
(Neo)-[:LOVES]->(Trinity),
(Morpheus)-[:KNOWS]->(Trinity),
(Morpheus)-[:KNOWS]->(Cypher),
(Cypher)-[:KNOWS]->(Smith),
(Smith)-[:CODED_BY]->(Architect),
(Keanu:Actor {name:'Keanu Reeves'}), (Keanu)-[:PLAYS]->(Neo),
(Lara:Actor {name:'Lara Flynn Boyle'}), (Lara)-[:PLAYS]->(Trinity)
```

Cypher Demo / Queries

- MATCH (n:Crew)-[r:KNOWS*]-(m)
  WHERE n.name='Neo’
  RETURN n AS Neo,r,m

- MATCH (n1:Actor)-[:PLAYS]->(c1:Crew)-[:LOVES]->(c2:Crew),
  (n2:Actor)-[:PLAYS]->(c2:Crew)
  RETURN n1, c1, c2, n2

- MATCH (n:Crew)-->(c)
  WHERE n.name='Neo’
  RETURN n,c

- MATCH (n:Crew)-[*]->(c)
  WHERE n.name='Neo’
  RETURN n,c