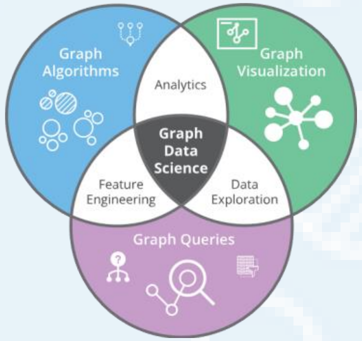


Habib University
shaping futures

CS343 Graph Data Science

Spring 2024

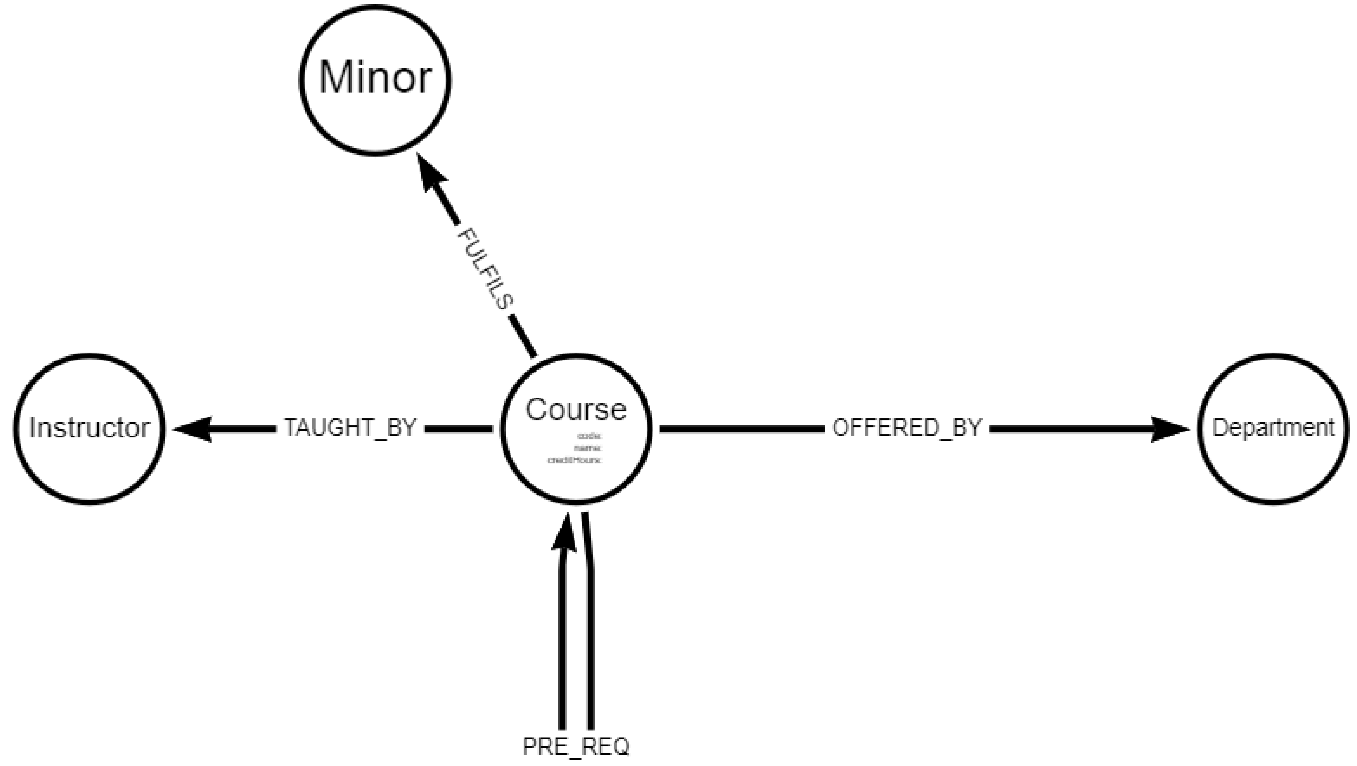
Querying Graph – Part 1

Chapter 3, Ian Robinson
Chapter 3, Tomaz

Muhammad Qasim Pasta
qasim.pasta@sse.habib.edu.pk

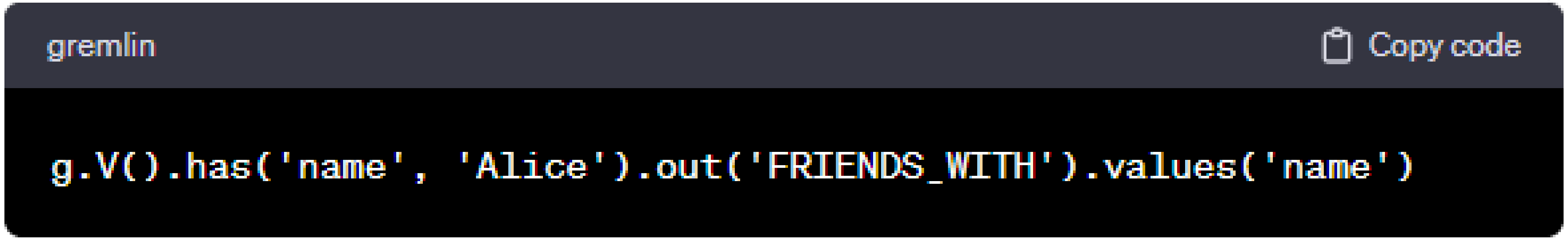
Slides are intended to be filled during the lectures. Certain details are intentionally omitted for in-class discussions. These slides are not meant to be used as reading material.

Courses: Graph Model



Graph Query Languages: Gremlin vs Cypher

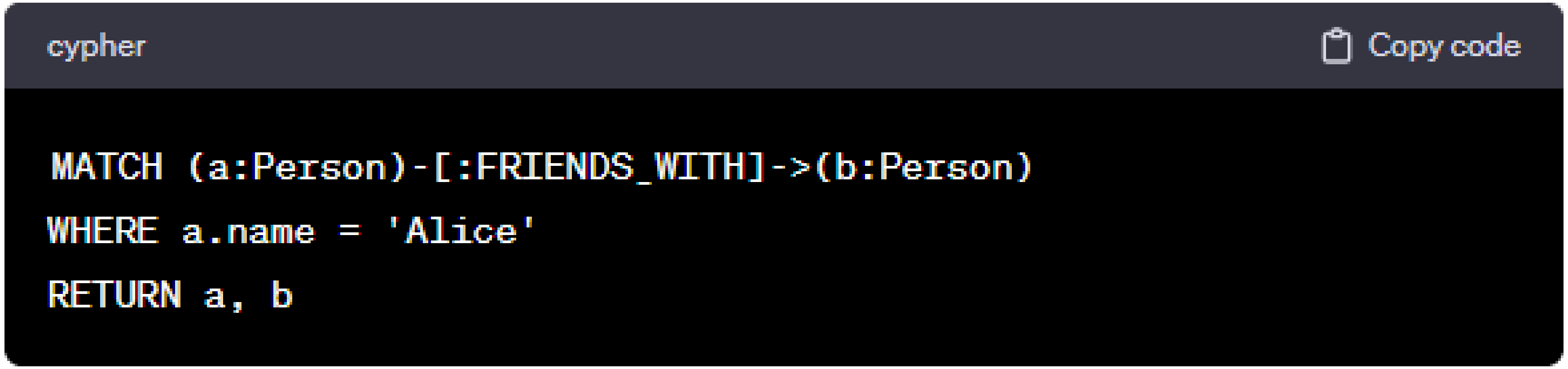
- **Gremlin:**
  - – Part of the Apache TinkerPop graph computing framework
  - – Compatible with various graph databases.
  - – A graph traversal language with an imperative style.
  - – User specifies the sequence of steps to traverse the graph and retrieve information.



```gremlin
g.V().has('name', 'Alice').out('FRIENDS_WITH').values('name')
```

Graph Query Languages: Gremlin vs Cypher

- **Cypher:**
  - Associated with Neo4j
  - Declarative Language
  - User define patterns to find in the graph
  - focuses on expressing relationships and patterns between nodes



```
MATCH (a:Person)-[:FRIENDS_WITH]->(b:Person)
WHERE a.name = 'Alice'
RETURN a, b
```

Patterns as ASCII Art

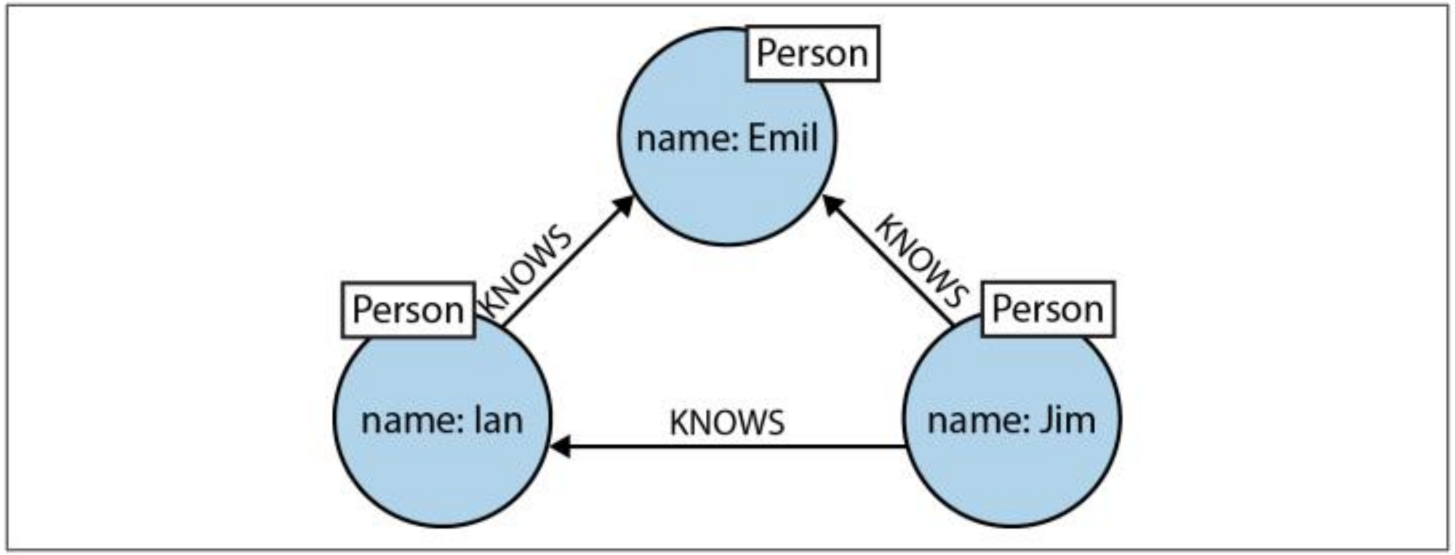


Figure 3-1. A simple graph pattern, expressed using a diagram

(emil)<-[:KNOWS]-(jim)-[:KNOWS]->(ian)-[:KNOWS]->(emil)

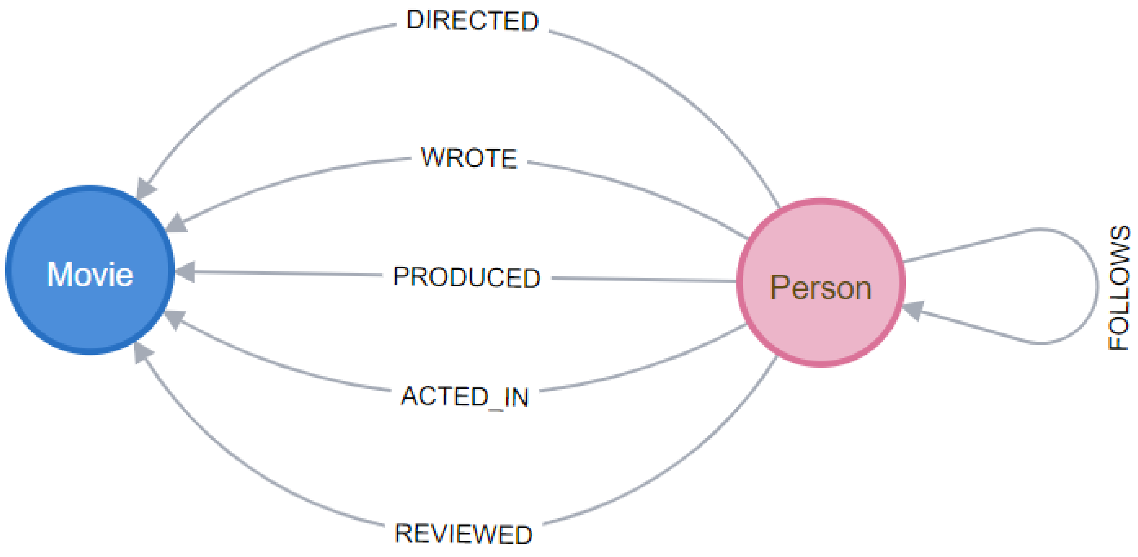
Patterns as ASCII Art

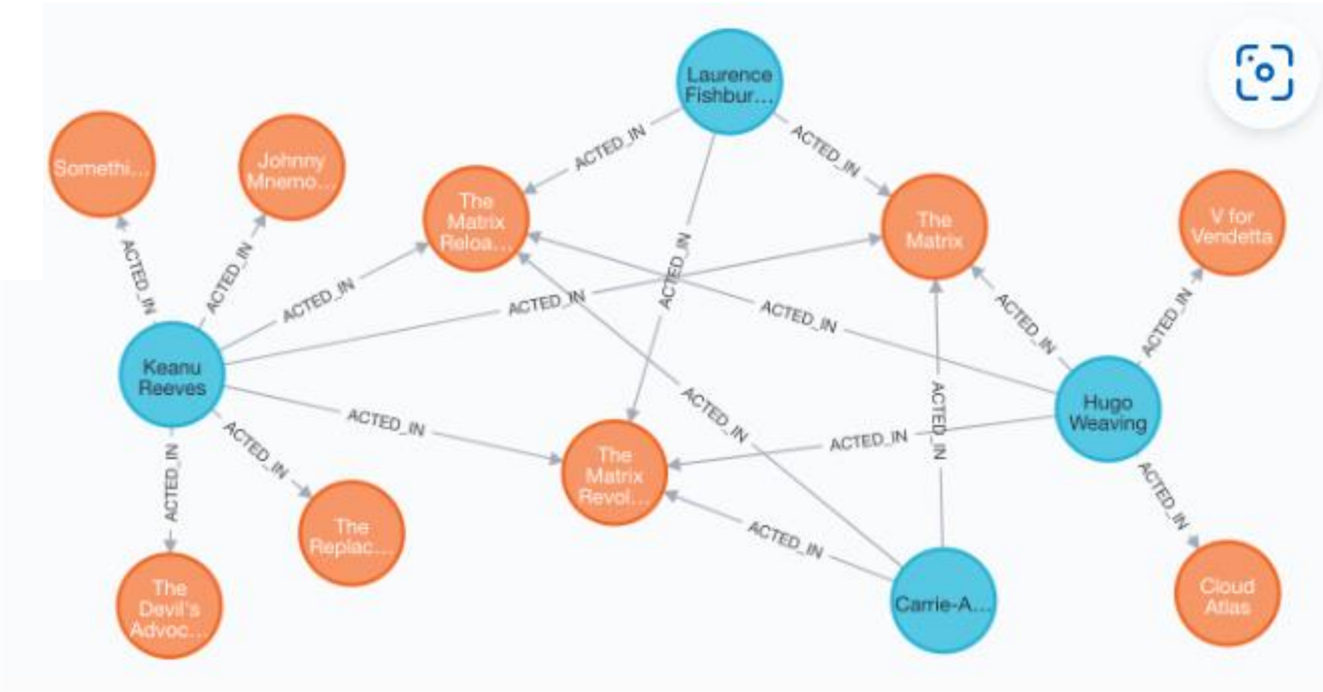
- Generate graph for the following patterns:
- (ahmed) –[:KNOWS]->(basit)<-[:KNOWS]-(bilal)
- (ahmed) –[:WORKS]->(PNS)<-[:LOCATED]-(KARACHI)
- (:Person{name:”Karim”}) -[:DRIVES]->(:Car{number:”ABC123”})

Cypher Clauses:

- **MATCH:** Specifies the patterns to match in the graph. It is used to find nodes, relationships, and paths that meet certain criteria.
- **OPTIONAL MATCH:** Similar to MATCH, but it allows for patterns that may not exist, and it does not affect the overall query result if the pattern is not matched.
- **RETURN:** Specifies what data to include in the result set. It is used to define the structure of the output, including nodes, relationships, properties, etc.
- **WHERE:** Filters the results based on specified conditions. It is used to include only the data that satisfies the specified criteria.
- **WITH:** Breaks the query into multiple parts. It is used to pass results from one part of the query to another, allowing for more complex queries.
- **ORDER BY:** Sorts the result set based on specified criteria. It is used to control the order of the output.
- **LIMIT:** Limits the number of results returned by the query. It is used to reduce the size of the result set.
- **SKIP:** Skips a specified number of results in the result set. It is often used in conjunction with LIMIT for pagination.

Movie Database





RETURN

```
WITH 10 as x
RETURN x
```

```
WITH 10 as x, 20 as y
RETURN x,y
```

```
WITH 10 as x, 20 as y
RETURN *
```

MATCH

Specifies the patterns to match in the graph. It is used to find nodes, relationships, and paths that meet certain criteria.

```
MATCH (n)
RETURN n
```

```
MATCH (p:Person)
RETURN p
```

```
MATCH (p:Person {name: 'Tom Hanks'})
RETURN p
```

```
MATCH (p:Person {name: 'Tom Hanks’})
RETURN p.born
```

MATCH

```
MATCH (p:Person {name: 'Tom Hanks'})-->()
RETURN *
```

```
MATCH (p:Person {name: 'Tom Hanks'})-->(m)
RETURN *
```

```
MATCH (p:Person {name: 'Tom Hanks'})-[:ACTED_IN]->(m)
RETURN *
```