

COMP5338 – Advanced Data Models

Week 7: Graph Data and Neo4j Introduction

Assignment Project Exam Help

Ying Zhou
School of Information Technologies

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



THE UNIVERSITY OF
SYDNEY

Administrative

- The group project instruction and data set will be published this week (week 7)
- Group can have up to 2 students
 - ▶ Self-enrolled groups will be set up on Canvas
 - ▶ Group name indicates the lab you will demo
 - ▶ Ok to form groups
- Project overview
 - ▶ You are given a d get queries
 - ▶ Design schema for two storage options
 - MongoDB based
 - Neo4j based
 - ▶ Load data, set up index, run target queries and observe performance
 - ▶ Describe your schema, query and performance in a report
 - ▶ Demo your solution to the tutor in week 10 lab
- Individual contribution will be assessed and members may get different marks

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat [edu_assist_pro](#)



Outline

- Brief Review of Graphs
- Examples of Graph Data
- Modelling Graph Data
- Property Graph Model
- Cypher Query

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Graphs

- A graph is just a collection of *vertices* and *edges*
 - ▶ Vertex is also called Node
 - ▶ Edge is also called Arc/Link

Assignment Project Exam Help

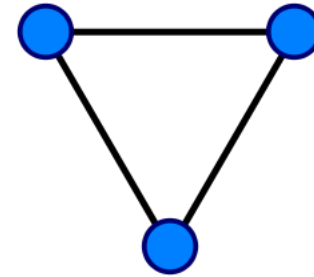
<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Type of Graphs

■ Undirected graphs

- ▶ Edges have no orientation (direction)
- ▶ (a, b) is the same as (b, a)



Assignment Project Exam Help

<https://eduassistpro.github.io/>

■ Directed graphs

- ▶ Edges have orientation (direction)
- ▶ (a, b) is not the same as (b, a)

Add WeChat: edu_assist_pro

Representing Graph Data

- Data structures used to store graphs in programs
 - ▶ Adjacency list
 - ▶ Adjacency matrix

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

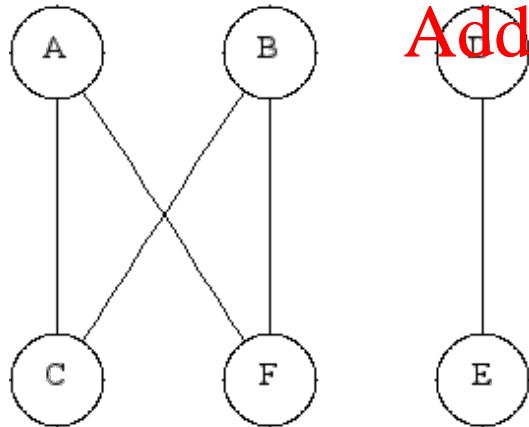


Adjacency List

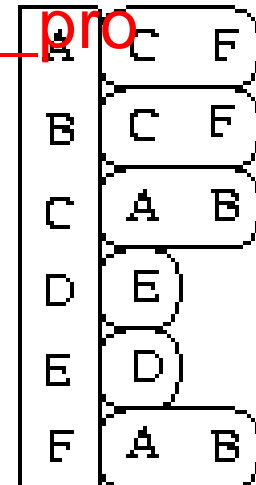
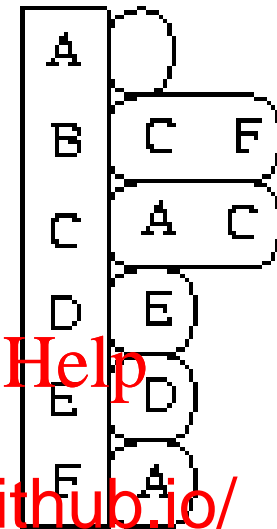
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Undirected Graph

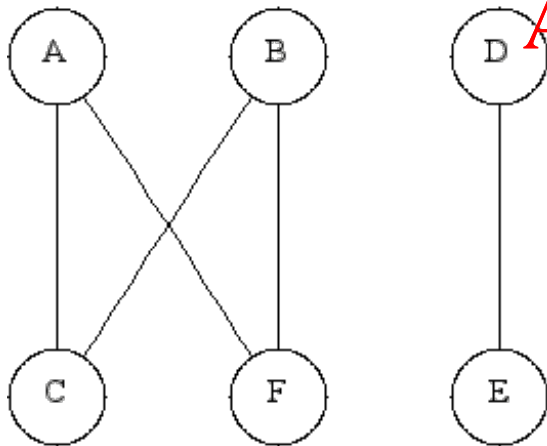


Adjacency matrix – Directed Graph

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Undirected Graph

Outline

- Brief Review of Graphs
- Examples of Graph Data
- Modelling Graph Data
- Introduction to Neo4j
- Cypher Query

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Examples of graphs

■ Social graphs

- ▶ Organization structure
- ▶ Facebook, LinkedIn, etc.

■ Computer Network topologies

- ▶ Data centre layout
- ▶ Network routing

■ Road, Rail and

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Social Graphs and extension



Assignment Project Exam Help

<https://eduassistpro.github.io/>

Nodes do not have to be of same type

Add WeChat edu_assist_pro

Edges have different meanings

A small social graph

Related information can be captured in the graph

Page 2 of the graph database book

Page 3 of the graph database book



Social Graph with Various Relationships

Multiple edges between the same pair of nodes

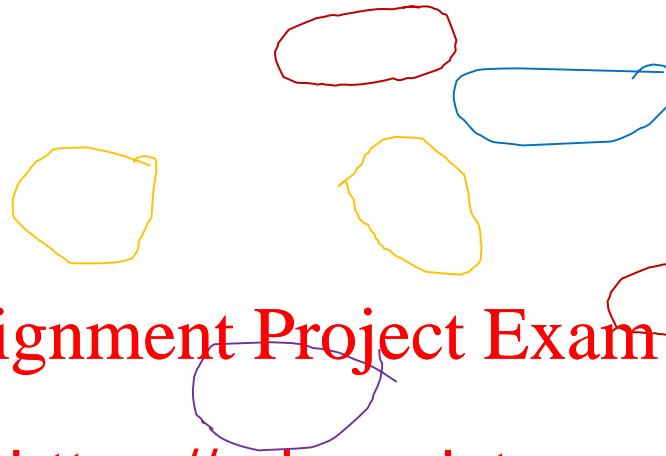
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Transaction information



This graph shows three types of entities

And various types of relationships among those entities

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

User PLACED order

User MOST_RECENT order

Order CONTAINS Item

Order PREVIOUS Order

Outline

- Brief Review of Graphs
- Examples of Graph Data
- Modelling Graph Data
- Property Graph Model
- Cypher Query

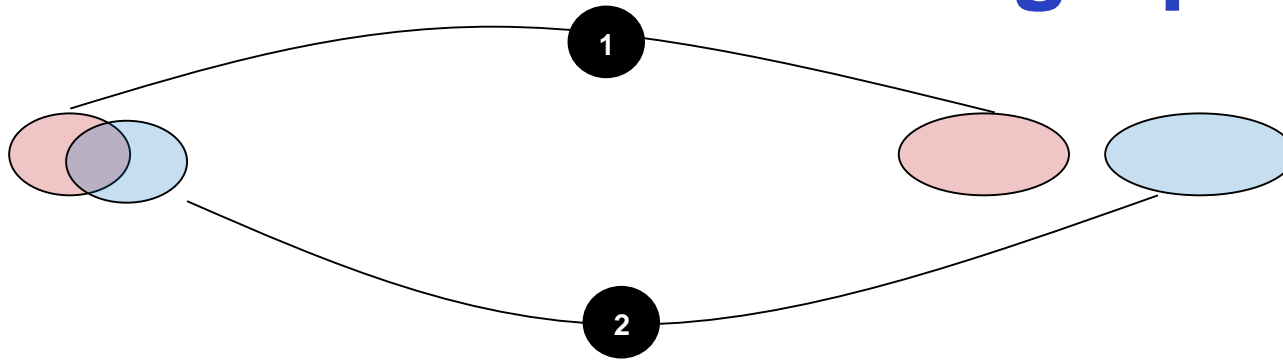
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



RDBMS to store graph



Assignment Project Exam Help

<https://eduassistpro.github.io/>

■ Who are Bob's friends?

Add WeChat edu_assist_pro

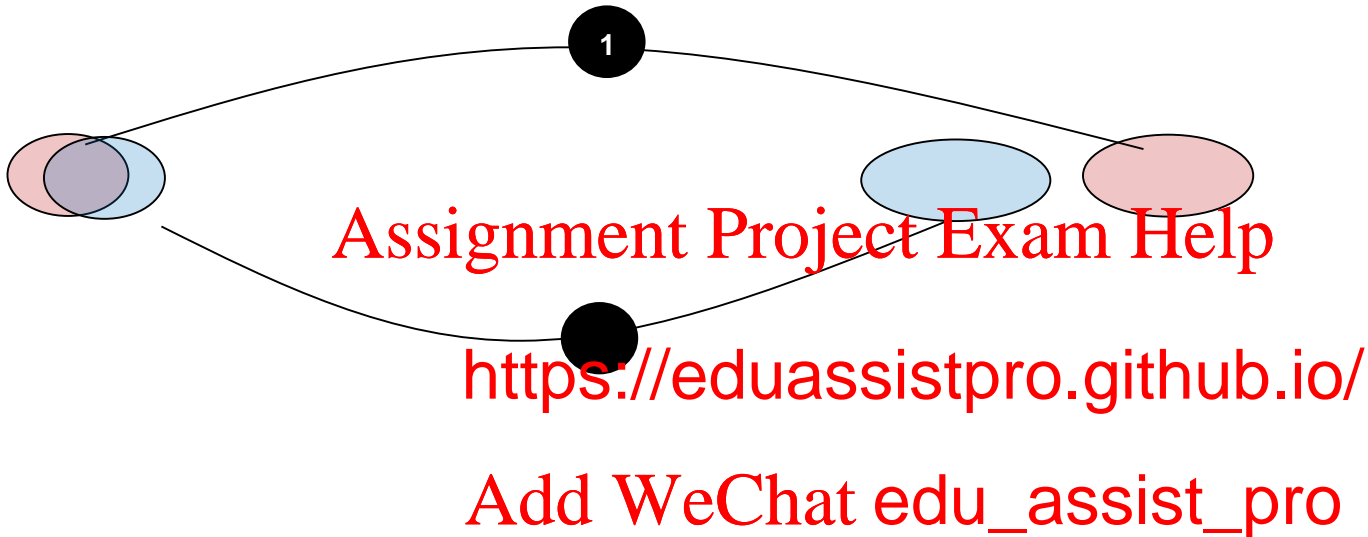
```
SELECT p1.Person
FROM   Person p1 JOIN PersonFriend pf ON pf.FriendID = p1.ID
      JOIN Person p2 ON pf.PersonID = p2.ID
WHERE  p2.Person = "Bob"
```

Page 13 of the graph database book



RDBMS to store Graphs

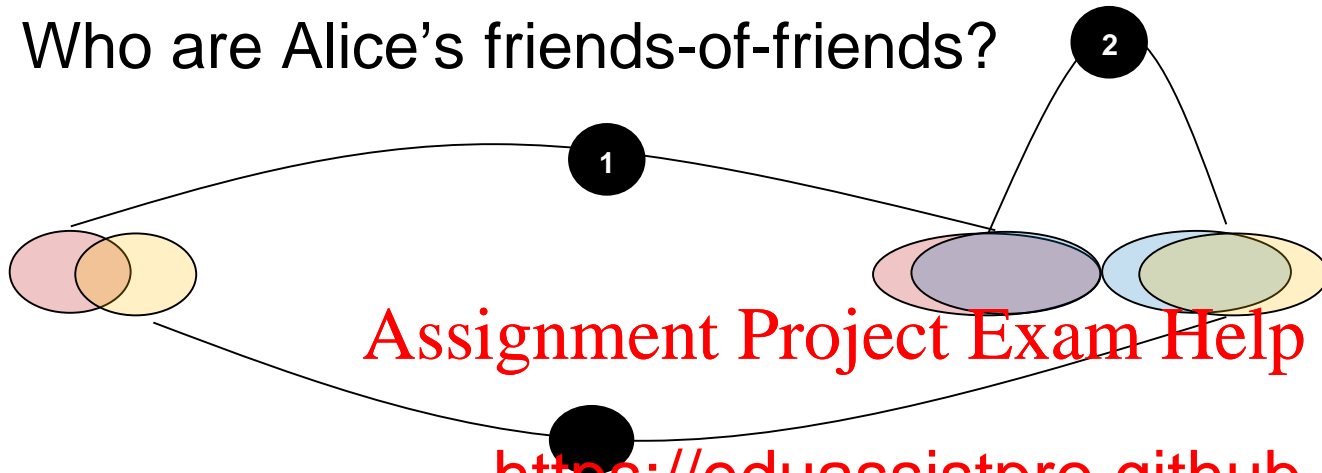
- Who are friends with Bob?



```
SELECT  p1.Person
FROM    Person p1 JOIN PersonFriend pf ON pf.PersonID = p1.ID
        JOIN Person p2 ON pf.FriendID = p2.ID
WHERE   p2.Person = "Bob"
```


RDBMS to store Graphs

■ Who are Alice's friends-of-friends?



Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

```
SELECT  p1.Person AS PERSON, p2.Person AS FRIEND_OF_FRIEND
FROM    PersonFriend pf1 JOIN Person p1 ON pf1.PersonID = p1.ID
        JOIN PersonFriend pf2 ON pf2.PersonID = pf1.FriendID
        JOIN Person p2 ON pf2.FriendID = p2.ID
WHERE   p1.Person = "Alice" AND pf2.FriendID <> p1.ID
```

MongoDB to store Graph

persons collection

```
{ _id: 1,  
  person: "Alice",  
  friends:[2]  
}  
  
{ _id: 2,  
  person: "Bob",  
  friends:[1,99]  
}  
  
{ _id: 99,  
  person: "Zach",  
  friends:[1]  
}
```

■ Who are Bob's friends?

▶ Find out Bob's friends' ID

- `db.persons.find({person:"Bob"},{friends:1})`

▶ For each id, find out the actual person

- `db.persons.find({_id: 1},{person:1}),`
`db.persons.find({_id: 99},{person:1}),`

- `db.persons.find({_id:{$in:[1,99]}}, {person:1})`

Assignment Project Exam Help

■ Who are friends with Bob?

<https://eduassistpro.github.io/>

▶ Find out the person with Bob

- `db.persons.find({person:"Bob"},{friends:1})`

Add WeChat edu_assist_pro

■ Who are Alice's friend

▶ Find out Alice's friends ID

- `db.persons.find({person:"Alice"},{friends:1})`

▶ For each id, find out the friends ID again

- `db.persons.find({_id:{$in:[2]}}, {friends:1})`

▶ For each id, find out the actual person

- `db.persons.find({_id:{$in:[1,99]}}, {person:1})`

■ The MongoDB 3.4 and later has a new aggregation stage called \$graphLookup



\$graphLookup

```
db.persons.aggregate([
  {$match:{person:"Alice"}},
  {$graphLookup:{
    from: "persons",
    startWith: "$friends",
    connectFromField:"f
    connectToField:"pid",
    maxDepth: 1,
    as: "friendsnetwork"}}
])
```

```
{ "_id" : 1,
  "person" : "Alice",
  "friends" : [ 2],
  "friendsnetwork" : [
    { "_id" : 99.0,
      "name" : "Zach",
      "friends" : [ 1, 3],
      "depth" : 1},
    { "_id" : 2,
      "name" : "Bob",
      "friends" : [1, 99],
      "depth" : 0}
  ] }
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



In Summary

- It is possible to store graph data in various storage systems
 - ▶ Shallow traversal
 - Relatively easy to implement
 - Performance OK
 - ▶ Deep traversal of traversal in other direction
 - Complicated t
 - Multiple join
 - Less efficient
 - Error prone

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Outline

- Brief Review of Graphs
- Examples of Graph Data
- Modelling Graph Data
- Property Graph Model
- Cypher Query

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Graph Technologies

■ Graph Processing

- ▶ take data in any input format and perform graph related operations
- ▶ OLAP – OnLine Analysis Processing of graph data
- ▶ Google Pregel, Apache Giraph

■ Graph Databases

- ▶ manage, query, <https://eduassistpro.github.io/>
- ▶ support high-lev
- ▶ native storage of graph data [Add WeChat edu_assist_pro](#)
- ▶ OLTP – OnLine Transaction Processing possible
- ▶ OLAP – also possible



Graph Data Models

■ RDF (Resource Description Framework) Model

- ▶ Express node-edge relation as “subject, predicate, object” triple (RDF statement)
- ▶ SPARQL query language
- ▶ Examples: Allegro Graph, Apache Jena

■ Property Graph

- ▶ Express node and edges, both can have properties
- ▶ Various query language
- ▶ Examples
 - Apache Titan
 - Support various NoSQL storage engine: BerkeleyDB, Cassandra, HBase
 - Structural query language: Gremlin
 - Neo4j
 - Native storage manager for graph data (Index-free Adjacency)
 - Declarative query language: Cypher query language

Assignment Project Exam Help

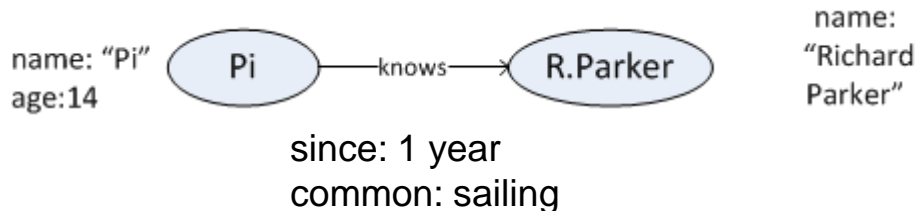
<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Property Graph Model

- Proposed by **Neo** technology
- No standard definition or specification
- Both Node and Edges can have property
 - ▶ RDF model cannot express edge property in a natural and easy to understand way
- The actual storage <https://eduassistpro.github.io/>
- The query language varies



Neo4j

- Native graph storage using property graph model
- Index-free Adjacency
 - ▶ Nodes and Relationships are stored
- Supports indexes
- Replication
 - ▶ Single master-master replication
- Neo4j cluster is limited to master-slave replication configuration
 - ▶ Database engine is not distributed
- Cypher – query language

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Property Graph Model as in Neo4j

■ Property graph has the following characteristics

- ▶ It contains nodes and relationships
- ▶ Nodes contain properties
 - Properties are stored in the form of key-value pairs
 - A node can have labels (classes)
- ▶ Relationships co
 - Has a *direction* from a *source node* and a *target node*
 - No dangling relationships (can't do a relationship)
- ▶ Properties
 - Both nodes and relationships have properties
 - Useful in modeling and querying based on properties of relationships

<http://docs.neo4j.org/chunked/milestone/graphdb-neo4j.html>



Property Graph Model Example

Assignment Project Exam Help

<https://eduassistpro.github.io/>

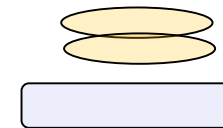
Add WeChat edu_assist_pro

It models a graph with three entities: 2 person and one movie, each with a set of properties;

It also models the relationship among them: one person acted in the movie with a role, another person directed the movie

Property Graph Model: Nodes

- Nodes are often used to represent entities, e.g. objects
 - ▶ It has properties
 - ▶ It can have labels
- A label is a way to group similar nodes
 - ▶ It acts like the 'class' concept in programming world
- Label is a dynamic property
 - ▶ It can be added
 - ▶ It can be used to tag nodes temporarily
 - E.g. :Suspend, :OnSale, etc



properties

A node with two labels and two properties

Property Graph Model: Relationships

- A relationship connects two nodes: source node and target node

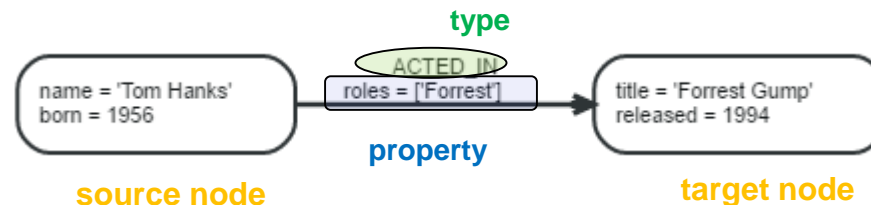
- ▶ The source and the target node can be the same one

- It always has a direction

- ▶ But traversal can happen in either direction

- It can have a type

- It can have properties



<https://eduassistpro.github.io/>
Add WeChat edu_assist_pro

Property Graph Model: Properties

- A property is a pair of property key and property value
- The property value can have the following type:
 - ▶ Number: Integer and Float
 - ▶ String
 - ▶ Boolean
 - ▶ Spatial Type: P
 - ▶ Temporal Type
 - Date
 - Time
 - ...

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Property Graph Model: Paths

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

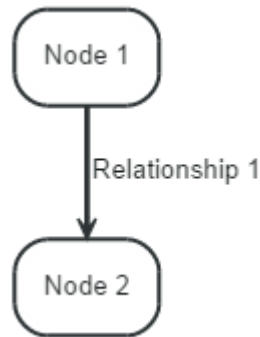
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



A path of length zero



A path of length one

Outline

- Brief Review of Graphs
- Examples of Graph Data
- Modelling Graph Data
- Property Graph Model
- Cypher Query

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Cypher

- Cypher is a query language specific to Neo4j
- Easy to read and understand
- It uses **patterns** to represents core concepts in the property graph model
 - ▶ E.g. a pattern m de is having a transaction with
 - ▶ There are basic relationships and path
- It uses **clauses** to build queries and keywords are inspired by SQL
 - ▶ A query may contain multiple clauses
- **Functions** can be used to perform aggregation and other types of analysis

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Cypher patterns: node

■ A single node

- ▶ A node is described using a pair of parentheses, and is typically given an identifier (variable)
- ▶ E.g.: **(n)** means a node n
- ▶ The variable's scope is restricted in a single query statement

■ Labels

- ▶ Label(s) can be
- ▶ E.g.: **(a:User)** or **(a:User:Admin)**

■ Specifying properties

- ▶ Properties are a list of name value pairs enclosed in a curly brackets
- ▶ E.g.: **(a { name: "Andres", sport: "Brazilian Ju-Jitsu" })**

<https://neo4j.com/docs/developer-manual/current/cypher/syntax/patterns/>



Cypher patterns: relationships

- Relationship is expressed as a pair of dashes (--)
 - ▶ Arrowhead can be added to indicate direction
 - ▶ Relationship always need a source and target node.
- Basic Relationships
 - ▶ Directions are not important: `(a)--(b)`
 - ▶ Named relations
 - ▶ Named and type `E ->(b)`
 - ▶ Specifying Relationship that may be of a set of types: `(a)-[r:TYPE1|TYPE2]->(b)`
 - ▶ Typed but not named relationship: `(a)-[:REL_TYPE]->(b)`

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Relationship patterns (cont'd)

■ Relationship of variable lengths

- ▶ $(a)-[*2]->(b)$ describes a path of length 2 between node a and node b
 - This is equivalent to $(a)-->()->(b)$
- ▶ $(a)-[*3..5]->(b)$ describes a path of minimum length of 3 and maximum length of 5 between node a and node b
- ▶ Either bound ca $(a)-[*..5]->(b)$
- ▶ Both bounds ca $(a)-[3..5]->(b)$

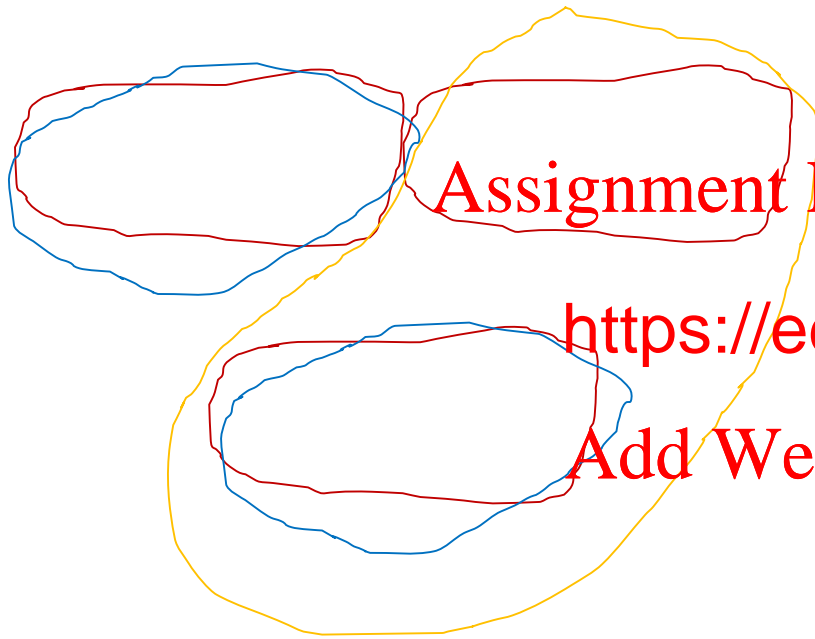
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Pattern Examples



- Pattern: **(n)**
- Matches all nodes in the graph
- Pattern: **(m:Movie)**
- Matches the movie node in the graph
- Pattern: **(p1)-[r:DIRECTED]->(m1)**
- Matches the path from person Robert Zemeckis to movie "Forrest Grum"

Assignment Project Exam Help

<https://eduassistpro.github.io/>

{name: 'Tom Hanks'}

Add WeChat edu_assist_pro the person node with Hanks' in the graph

Cypher Clauses - Create

■ CREATE

- ▶ Create nodes or relationships with properties

■ Create a node **matrix1** with the label **Movie**

```
CREATE (matrix1:Movie {title:'The Matrix', released:1999,  
                        real World'})
```

We give the nod

later in the same query

■ Create a node **keanu** with the label **Actor**

```
CREATE (keanu:Actor {name:'Keanu Reeves', born:1964})
```

■ Create a relationship **ACTS_IN**

```
CREATE (keanu)-[:ACTS_IN {roles:'Neo'}]->(matrix1)
```

The identifier “Keanu” and “matrix1” are used in the this create clause.

We did not give the relationship a name/identifier.

We need to write the three clauses in a single query statement to be able to use those variables

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Cypher – Read

■ MATCH ... RETURN

- ▶ MATCH is the main reading clause
- ▶ RETURN is a projecting clause
- ▶ They are chained to make a query

■ Return all nodes:

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

<https://eduassistpro.github.io/>

■ Return all nodes with a given id * from movie

```
MATCH (movie:Movie) RETURN movie
```

■ Return all actors' name in the movie "The Matrix"

We give the Actor node an identifier "a" so we can use refer to in the RETURN sub-clause

```
MATCH (a:Actor)-[:ACTS_IN]->(movie:Movie{title:"The Matrix"})  
RETURN a.name
```

We do not need to return the relationship so we did not give an identifier to it

We do not need to give an identifier to the Movie node too,

Assignment Project Exam Help

Add WeChat edu_assist_pro



Cypher - Uppdate

■ MATCH ... SET/REMOVE ... RETURN

■ Set the age property for all actor nodes

```
MATCH (n:Actor)
SET n.age = 2014 - n.born
RETURN n
```

■ Remove a property

```
MATCH (n:Actor)
REMOVE n.age
RETURN n
```

■ Remove a label

```
MATCH (n:Actor{name:"Keanu Reeves"})
REMOVE n:Actor
RETURN n
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Cypher - Delete

■ MATCH ... DELETE

■ Delete relationship

```
MATCH (n{name:"Keanu Reeves"})-[r:ACTS_IN]->()  
DELETE r
```

<https://eduassistpro.github.io/>
ip

■ Delete a node a

```
MATCH (m{title:'The Matrix'})  
DELETE m,r
```

Add WeChat edu_assist_pro



More on READ: WHERE

- The **WHERE** sub clause can be used to specify various query conditions

MATCH (n)

WHERE $n.age < 30$ and $n.employ \geq 3$

RETURN n.na

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Functions

- Functions may appear in various clauses

- ▶ Build-in and user-defined functions

- Build-in functions

- ▶ Prediction functions

- ▶ Scalar functions

- ▶ Aggregation fun

- ▶ List functions

- ▶ Mathematical functions

- ▶ String functions

- ▶ Temporal functions

- ▶ Spatial Functions

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Aggregating Functions

- GROUP BY feature in Neo4j is achieved using aggregating functions
 - ▶ E.g. `count()`, `sum()`, `avg()`, `max()`, `min()` and so on
- The grouping key is implied in the return clause
 - ▶ None aggregate expression in the return clause is the grouping key
 - ▶ **RETURN n, count(n)**
 - n is a variable and it is the grouping key
 - ▶ **MATCH(n:PERSON) RETURN n, count(n)**
 - Count the number of nodes representing n and female in the graph
 - A person's gender is the grouping key
- A grouping key is not always necessary, the aggregation function can apply to all results returned
 - ▶ **MATCH (n:PERSON) RETURN COUNT(*)**
 - To count the number of Person nodes in the graph

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Aggregation Examples

- To find out the earliest year a Person was born in the data set

```
MATCH (n:PERSON) RETURN min (n.born)
```

- To find out the distribution of relationship types belonging to nodes with certain feature

```
MATCH (n { na  
RETURN type(r)
```

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

The grouping key is type(r) which is a scalar
relationship in the matching results

turns the type of

Aggregation Examples: DISTINCT

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

variable friend_of_friend refers to the same node in both matches

```
MATCH (me:Person)-->(friend:Person)
WHERE me.name = 'A'
RETURN count(DISTINCT friend_of_friend), count(friend_of_friend)
```

count(DISTINCT friend_of_friend)	count(friend_of_friend)
1	2
1 row	



More on READ: subqueries

- The **WITH** clause can chain different query parts together in a pipeline style
 - ▶ Used to apply conditions on aggregation result
 - ▶ Used to modify (order, limiting, etc) the results before collecting them as a list

- Examples **Assignment Project Exam Help**

- ▶ Find the person ies
MATCH (p:Person)
WITH p, count(*) as movies
WHERE movies >= 3
RETURN p.name, movies

- ▶ Return the oldest 3 person as a list

```
MATCH (n:Person)  
WITH n  
ORDER by n.age DESC LIMIT 3  
RETURN collect(n.name)
```

```
MATCH (n:Person)  
RETURN n.name  
ORDER by n.age DESC LIMIT 3
```



Dealing with Array type

- Array literal is written in a similar way as it is in most programming languages

- ▶ examples

- An array of integer: [1,2,3]
- An array of string: ["Sydney", "University"]

- Both node and rproperty of array type

- ▶ Example: create

```
create (Keanu)-[:ACTED_IN {roles:['Ne
```

- ▶ Example: update an existing node

```
MATCH (n:Person{name: "Tom Hanks"})
```

```
set n.phone=["0123456789","93511234"]
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Dealing with Array type (cont'd)

■ Querying array property

- ▶ The **IN** operator: check if a value is in an array

- Example: find out who has played 'Neo' in which movie

```
MATCH (a:Person) -[r:ACTED_IN]->(m:Movie)
```

```
WHERE 'Neo' IN r.roles
```

```
RETURN a , m
```

- ▶ The **UNWIND** operator splits a single row into multiple rows

- Example: find all the movies released 1999 or in 2003

```
UNWIND [1999,2003] as year
```

```
MATCH (m: Movie)
```

```
WHERE m.released = year
```

```
RETURN m.title, m.released
```

This is equivalent to

```
MATCH(m: Movie)
```

```
WHERE m.released IN [1999,2003]
```

```
RETURN m.title, m.released
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Dealing with Array Type (cont'd)

■ A relatively complex query

- ▶ Update another node

```
MATCH (n:Person{name: "Meg Ryan"}) set n.phone=["0123456789"]
```

- ▶ Run a query to see who shares any phone number with Tom Hanks

```
MATCH (n:Person{name: "Tom Hanks"})
```

```
WITH n.phone as phones, n
```

```
UNWIND phones
```

```
MATCH (m:Person
```

```
WHERE phone in m.phone and n<>m
```

```
RETURN m.name
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Where to find more about cypher query:

Developer's guide: <http://neo4j.com/docs/developer-manual/current/cypher/>

Reference card: <https://neo4j.com/docs/cypher-refcard/current/>



Indexing

- Neo4j supports index on properties of labelled node
- Index has similar behaviour as those in relational systems
- Create Index
 - ▶ `CREATE INDEX ON :Person(name)`
- Drop Index
 - ▶ `DROP INDEX ON :`
- Storage and query execution w

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

ed in week 8



References

- Ian Robinson, Jim Webber and Emil Eifrem, *Graph Databases*, Second Edition, O'Reilly Media Inc., June 2015
 - ▶ You can download this book from the Neo4j site, <http://www.neo4j.org/learn> will redirect you to <http://graphdatabases.com/>
- The Neo4j Document
 - ▶ The Neo4j Graph Database Concept (<http://neo4j.com/docs/stable/graphdb-neo4j.html>)
- Noel Yuhanna, *Market Overview: Graph Databases*, Forrester White Paper, May, 2015
- Renzo Angeles, *A Comparison of Graph Database Models*, ICDE Workshops 2013 (DOI-10.1109/ICDEW.2013.6628582)
- Renzo Angeles and Claudio A. T. V. *Graph Database Models*, ACM Computing Surveys, Vol. 40, No. 1, Article 1, February 2008 (DOI-10.1145/1322432.1322433)

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

