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

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

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

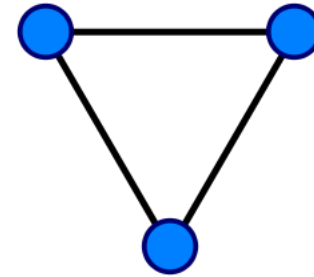▶ Edge is also called Arc/Link

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) Add WeChat edu_assist_pro

  ▶ (a, b) is not the same as (b, a)

Representing Graph Data

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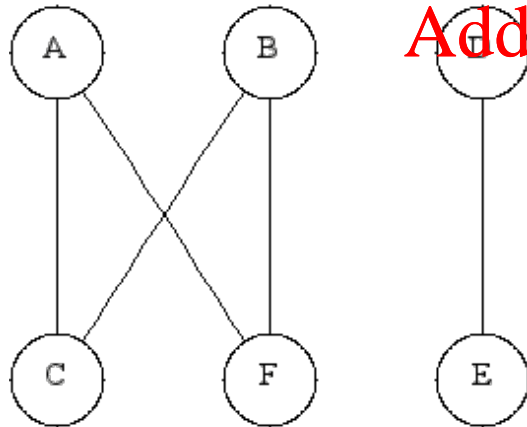 - Adjacency matrix

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Adjacency List

A	→	(A)
B	→	C F
C	→	A C
D	→	E
E	→	D
F	→	A

A	→	C F
B	→	C F
C	→	A B
D	→	E
E	→	D
F	→	A B

Undirected Graph

Adjacency matrix – Directed Graph

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro
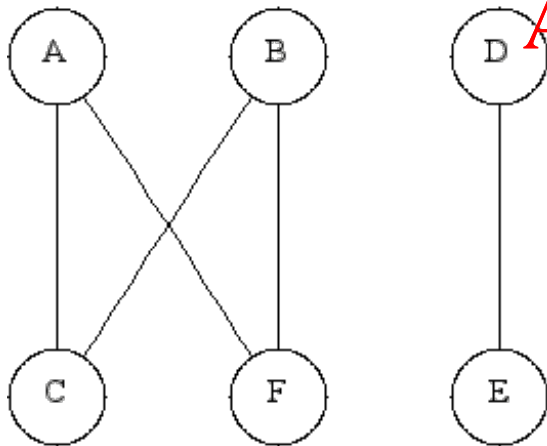
Undirected Graph

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/

Add WeChat edu_assist_pro

Nodes do not have to be of same type

A small social graph

Edges have different meanings

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

Page 19 of the graph database book

Transaction information
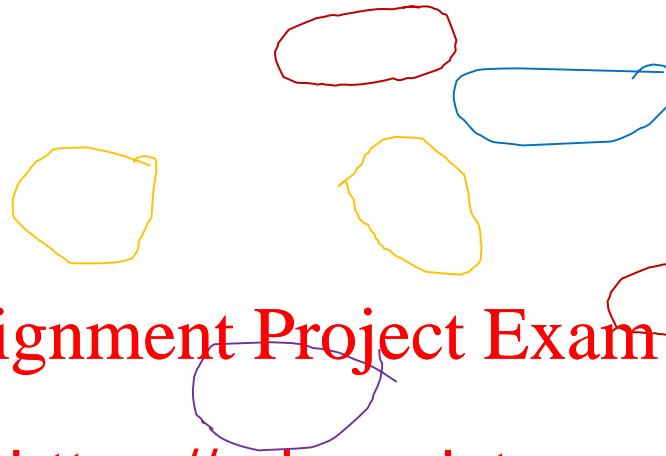
This graph shows three types of entities

And various types of relationships among those entities

Assignment Project Exam Help

https://eduassistpro.github.io/

User PLACED order

Add WeChat edu_assist_pro

User MOST_RECENT order

Order CONTAINS Item

Order PREVIOUS Order

Page 23 of the graph database book

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/
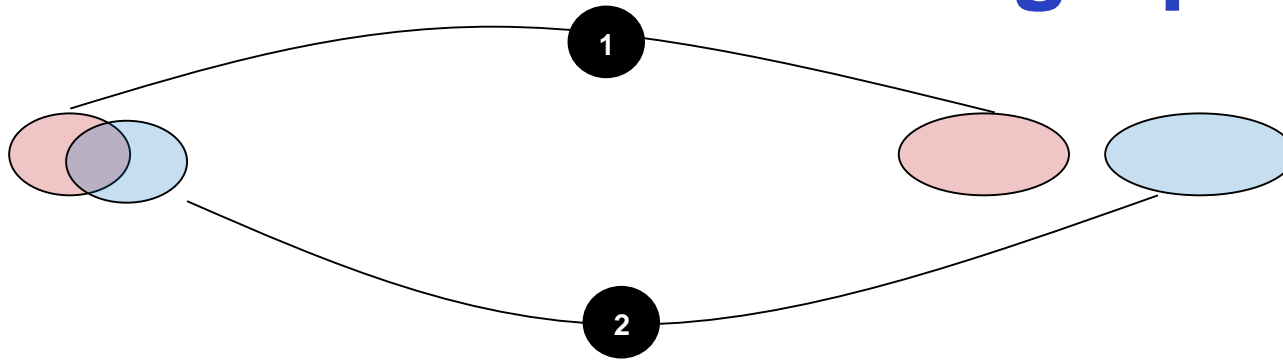
- Who are Bob's friends?

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?

Assignment Project Exam Help

https://eduassistpro.github.io/

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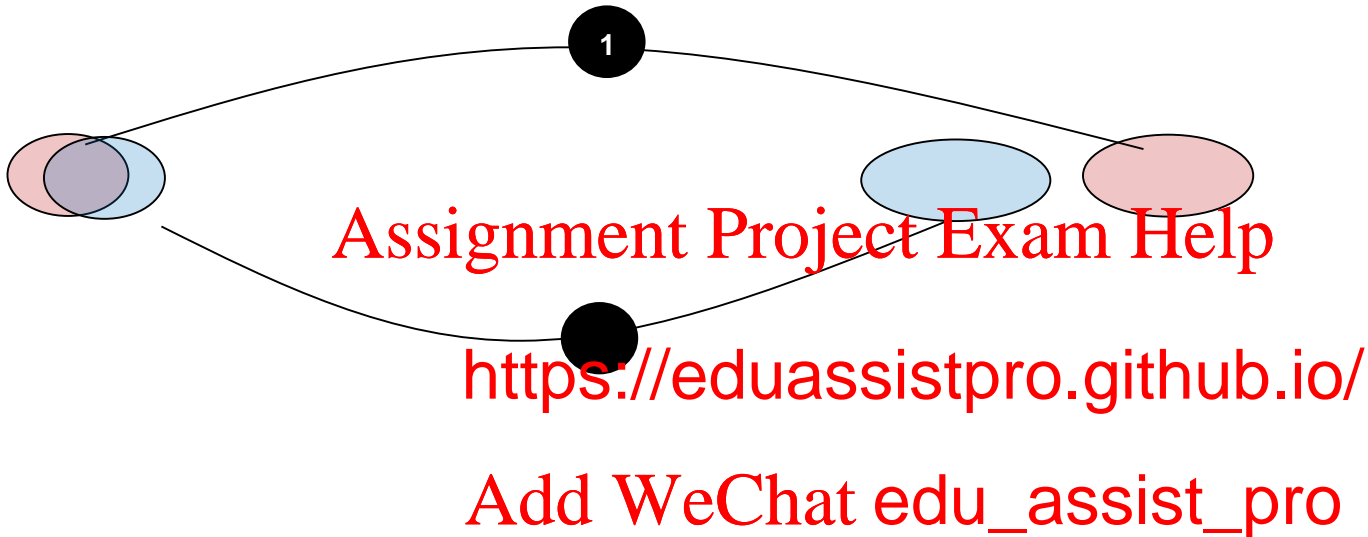
```
SELECT    p1.Person
FROM      Person p1 JOIN PersonFriend pf ON pf.PersonID = p1.ID
          JOIN Person p2 ON pf.FriendID = p2.ID
WHERE
          p2.Person = "Bob"
```

RDBMS to store Graphs

- Who are Alice's friends-of-friends?

Assignment Project Exam Help

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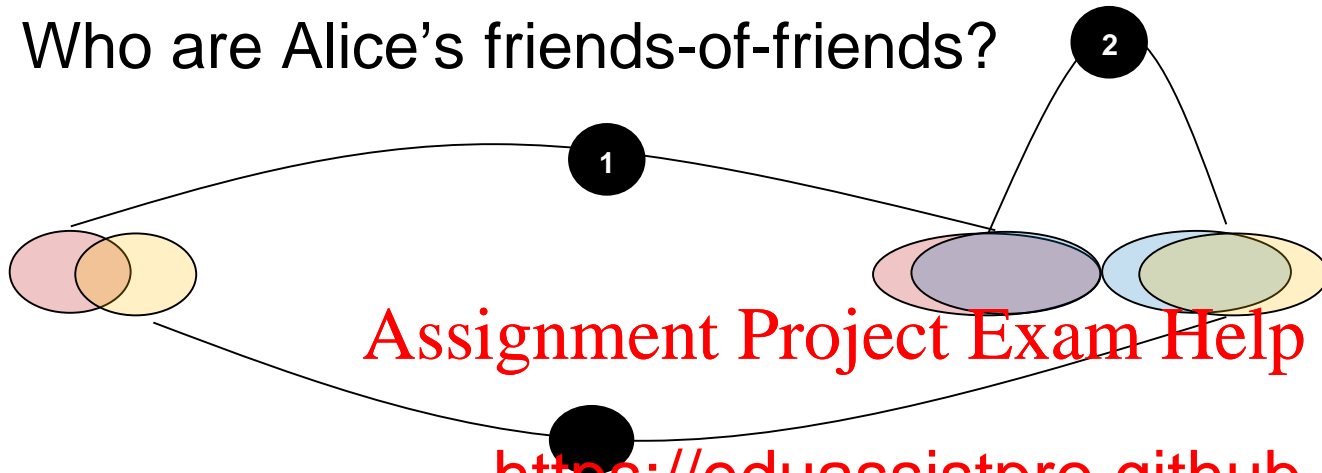
Add WeChat edu_assist_pro

```
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})

- Who are friends with Bob?

Assignment Project Exam Help

https://eduassistpro.github.io/

  - ▶ Find out the perso ... with Bob
    - db.persons.fi ... son: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 MongDB 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

    connectToFirld:"pid",

    maxDepth: 1,

    as: "friendsnetwork"}}

  ])

{"_id" : 1,
  "person" : "Alice",
  "friends" : [2],
  "friendsnetwork" : [
    {"_id" : 99.0,
    "name" : "Zach",
    "friends" : [1, 3],

    "n
    "fri
    "depth" : 1},
    {"_id" : 2,
    "name" : "Bob",
    "friends" : [1, 99],
    "depth" : 0}
  ] }

In Summary

- It is possible to store graph data in various storage systems
  - ▶ Shallow traversal
    - Relatively easy to implement
    - Performance OK
  - ▶ Deep traversal or traversal in other direction
    - Complicated t
      - Multiple joi
    - 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,
  - ▶ support high-lev
  - ▶ native storage of graph data
  - ▶ OLTP – OnLine Transaction Processing possible
  - ▶ OLAP – also possible

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Graph Data Models

- **RDF (Resource Description Framework) Model**
  - ▶ Express node-edge relation as "subject, predicate, object" triple (RDF statement)
  - ▶ SPARQL query language
  - ▶ Examples: AllegroGraph, Apache Jena
- **Property Graph**
  - ▶ Express node a ~~~~~~~~~~~~~~~~~~~~~~~s, 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
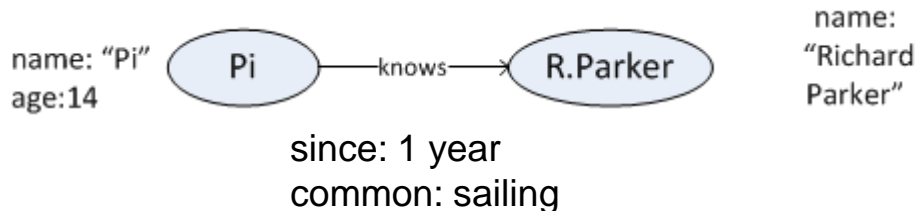
Property Graph Model

- 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 stora https://eduassistpro.github.io/
- The query language varies

Assignment Project Exam Help

Add WeChat edu_assist_pro

name: "Pi"
age:14

Pi —knows→ R.Parker

name: "Richard Parker"

since: 1 year
common: sailing

Neo4j

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

Property Graph Model as in Neo4j

- Property graph has the following characteristics
  - It contains <u>nodes</u> and <u>relationships</u>
  - Nodes contain <u>properties</u>
    - Properties are stored in the form of key-value pairs
    - A node can have labels (classes)
  - Relationships co
    - Has a *directio*~~de and a~~ *target node*
    - No dangling relationships (can't d   a relationship)
  - <u>Properties</u>
    - 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

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 used 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 dyna
  - ▶ It can be added
  - ▶ It can be used to tag node tempor
    - ▪ E.g. :Suspend, :OnSale, etc

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro
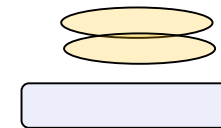
properties

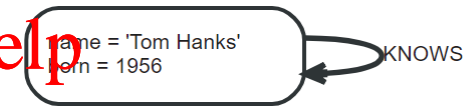A node with two labels and two properties

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 t
- It can have properties

Assignment Project Exam Help

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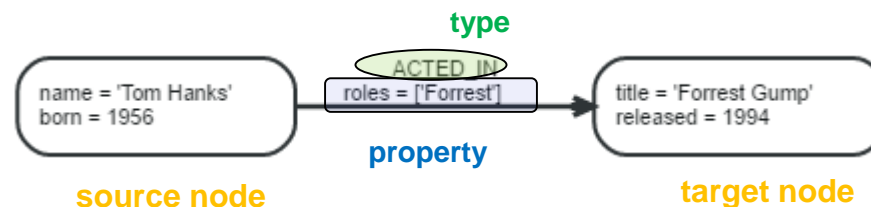
Add WeChat edu_assist_pro

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

Node

A path of length zero

Node 1

Relationship 1

Node 2
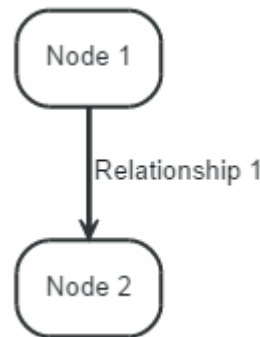
A path of length one

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

Assignment Project Exam Help

  - ▶ E.g. a pattern m   de is having a
    transaction with https://eduassistpro.github.io/
  - ▶ There are basic   relationships and path

Add WeChat edu_assist_pro

- It uses **clauses** to build querie   lauses and keywords are inspired by SQL
  - ▶ A query may contain multiple clauses
- **Functions** can be used to perform aggregation and other types of analysis

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 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   )-[*..5]->(b)
  - ▶ Both bounds ca   b)

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Pattern Examples
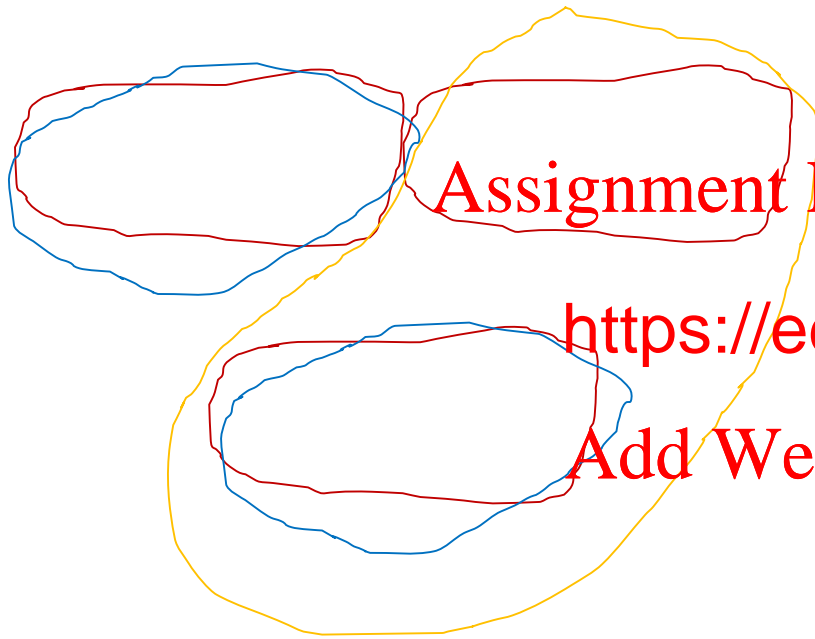
- Pattern: **(n)**
- Matches all nodes in the graph

- Pattern: **(m:Movie)**
- Matches the movie node in the

Assignment Project Exam Help

https://eduassistpro.github.io/

{name: 'Tom Hanks'})

Add WeChat edu_assist_pro person node with

Hanks' in the graph

- Pattern: **(p1)-[r:DIRECTED]->(m1)**
- Matches the path from person Robert Zemeckis to movie "Forrest Grum'

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,
                                            eal World'})
```

We give the nod   later in the same query

- Create a node **keanu** with the l

```
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

Cypher – <u>R</u>ead

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

- Return all nodes:

  Assignment Project Exam Help

  ```
  MATCH (n) RET
  ```
  https://eduassistpro.github.io/

- Return all nodes with a given Add WeChat edu_assist_pro 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{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,

Cypher - Update

■ **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 prope

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

■ Remove a label

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

Cypher - <u>D</u>elete

- **MATCH ... DELETE**

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

- Delete a node a   ip
  ```
  MATCH (m{title:"The Matrix"})
  DELETE m,r
  ```

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>=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/

AddWeChat 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** is a variable ... nd it is the grouping key
  - ▶ **MATCH(n:PERSON) RETURN n** ... **UNT(*)**
    - Count the number of nodes repre ... nd 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)

  The grouping key is type(r) which is a sc   turns the type of relationship in the matching results

Aggregation Examples: DISTINCT

Assignment Project Exam Help

https://eduassistpro.github.io/

ariable **friend_of_friend** refers to
e same node in both matches

Add WeChat edu_assist_pro

MATCH **(me:Person)-->(friend:Pers** **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:Per https://eduassistpro.github.io/ **)**

    WITH p, count(*) as movies

    WHERE movies >= 3 Add WeChat edu_assist_pro

    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 r   perty of array type
  - ▶ Example: create   operty

    create (Keanu)-[:ACTED_IN{roles:['Ne   ix)
  - ▶ Example: update an existing node   operty

    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: turn an array into multiple rows

■ Example: find all the movies released in 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

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:Perso

    WHERE phone in m.phone and n<>m

    RETURN m.name

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 <u>properties</u> of <u>labelled node</u>
- Index has similar behaviour as those in relational systems
- Create Index
  - ▶ **CREATE INDEX ON :Person(name)**
    Assignment Project Exam Help
- Drop Index
  - ▶ **DROP INDEX ON :** https://eduassistpro.github.io/
- Storage and query execution w 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 Comp   ls*, ICDE Workshops 2013 (DOI-10.1109/ICDEW.20
- Renzo Angeles and Clau   *abase Models*, ACM Computing Surveys, Vol. 40, N0. 1, Article 1, F   OI-10.1145/1322432.1322433)

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro