



Cơ sở dữ liệu đồ thị

Biên soạn: Bộ môn HTTT

NỘI DUNG



1. Cơ sở dữ liệu đồ thị



2. Giới thiệu Neo4J



3. Tạo cơ sở dữ liệu đồ thị

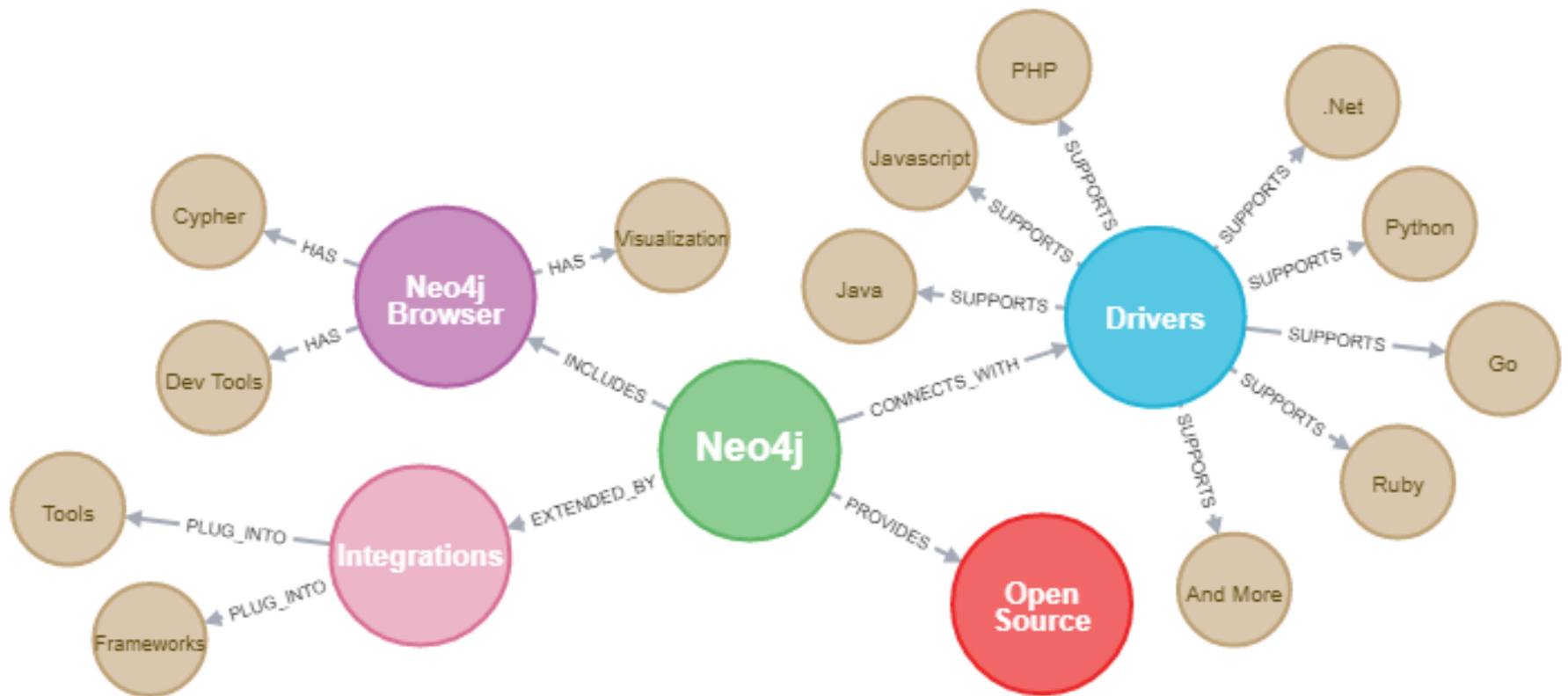


4. Truy vấn dữ liệu

Cơ sở dữ liệu đồ thị

- ❖ Cơ sở dữ liệu đồ thị là cấu trúc được thiết kế để lưu trữ và xử lý dữ liệu theo mô hình đồ thị.
- ❖ Các hệ quản trị cơ sở dữ liệu đồ thị có cơ chế xử lý dữ liệu hiệu quả với nhiều mối liên kết phức tạp trên đồ thị mà các mô hình dữ liệu khác không thể đáp ứng.

Giới thiệu Neo4J

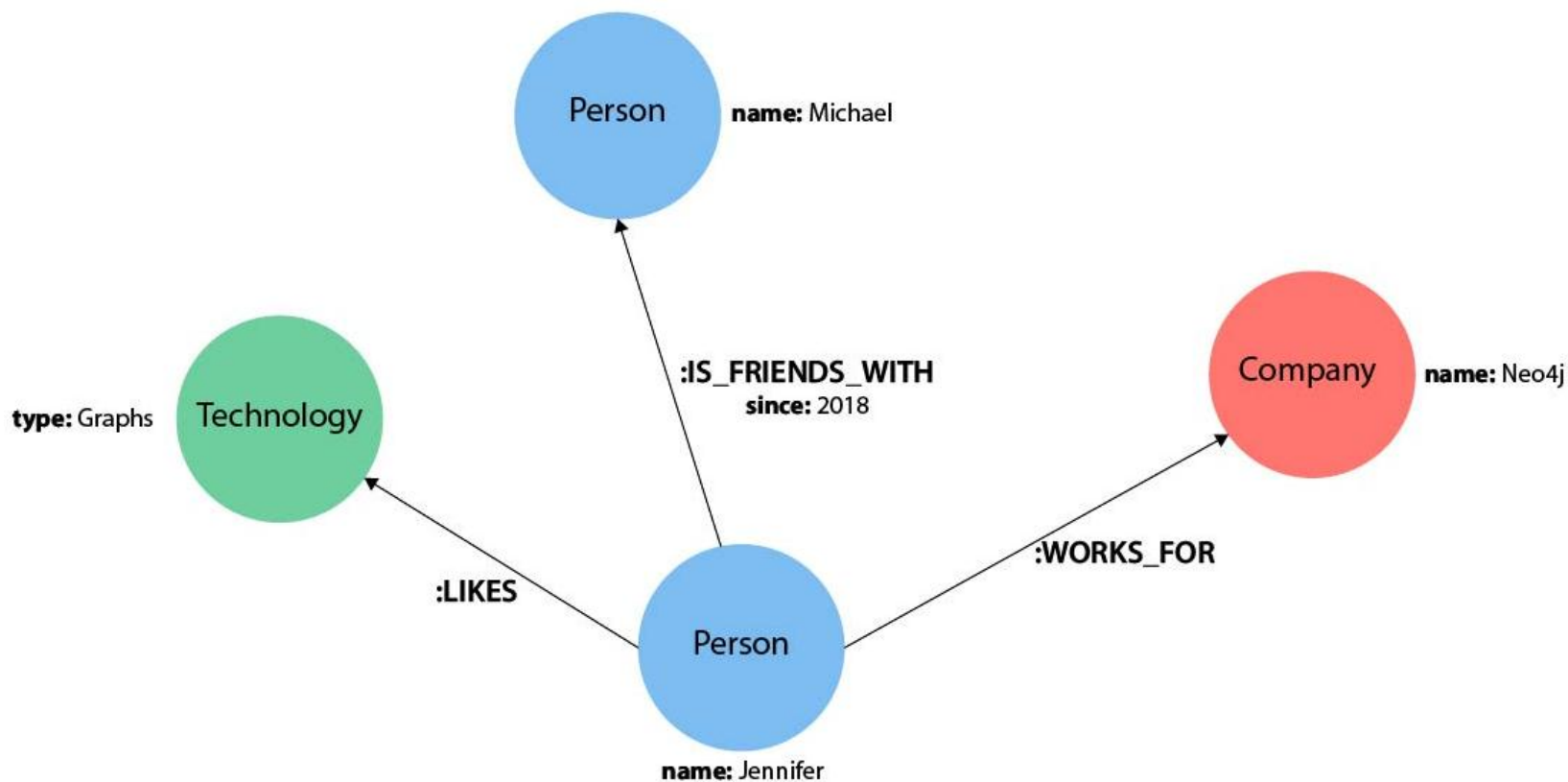


Giới thiệu Neo4J

- ❖ Neo4J là một hệ quản trị CSDL đồ thị được phát triển bởi công ty Neo4J
- ❖ Sử dụng ngôn ngữ Cypher để tạo đồ thị, cập nhật dữ liệu và truy vấn.
- ❖ Cung cấp các driver để kết nối với các ứng dụng lập trình trên nhiều ngôn ngữ khác nhau: Java, .Net, Python,...
- ❖ Neo4J có 2 phiên bản:
 - Neo4J Community (mã nguồn mở)
 - Neo4J Enterprise (thương mại)

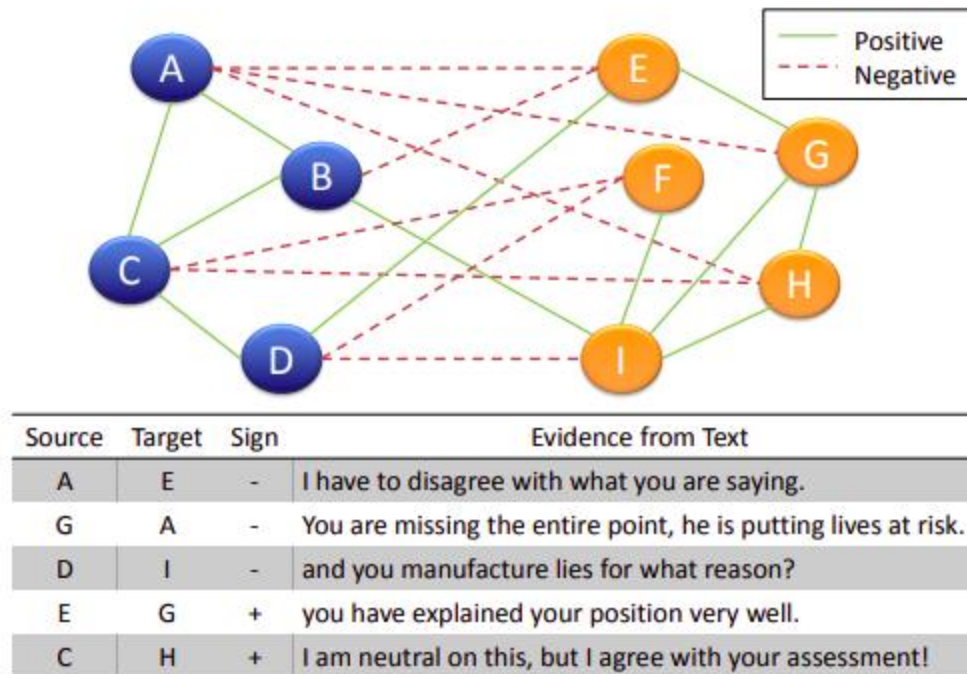
Cơ sở dữ liệu đồ thị trong Neo4J

- ❖ Ví dụ cơ sở dữ liệu đồ thị được mô tả sau:
*Jennifer thích Graphs. Jennifer là bạn của Michael.
Jennifer làm việc cho Neo4j.*



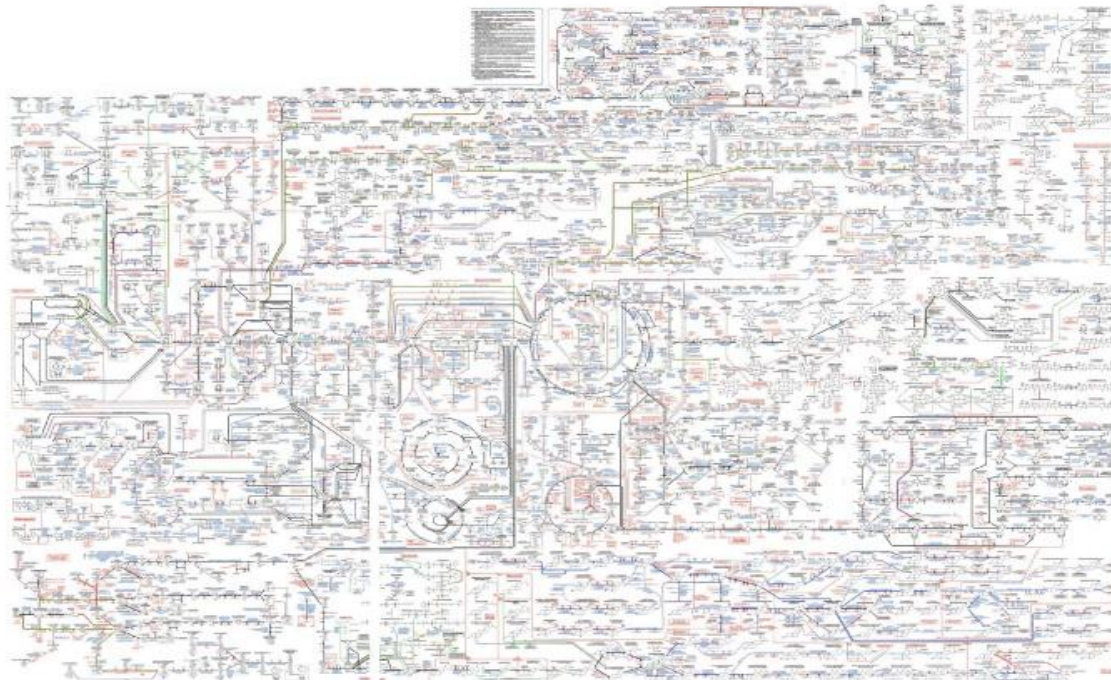
Ứng dụng của đồ thị

- ❖ Khoa học xã hội: Để tìm những người Nổi tiếng X trong mạng xã hội, chúng ta có thể thống kê số lượng Follow của X và đưa ra những người có số lượng Follow cao nhất. Nói theo cách toán học thì ta tìm bậc của đỉnh hay số lượng cạnh nối với đỉnh đó (X) và tìm ra những đỉnh có bậc cao nhất.



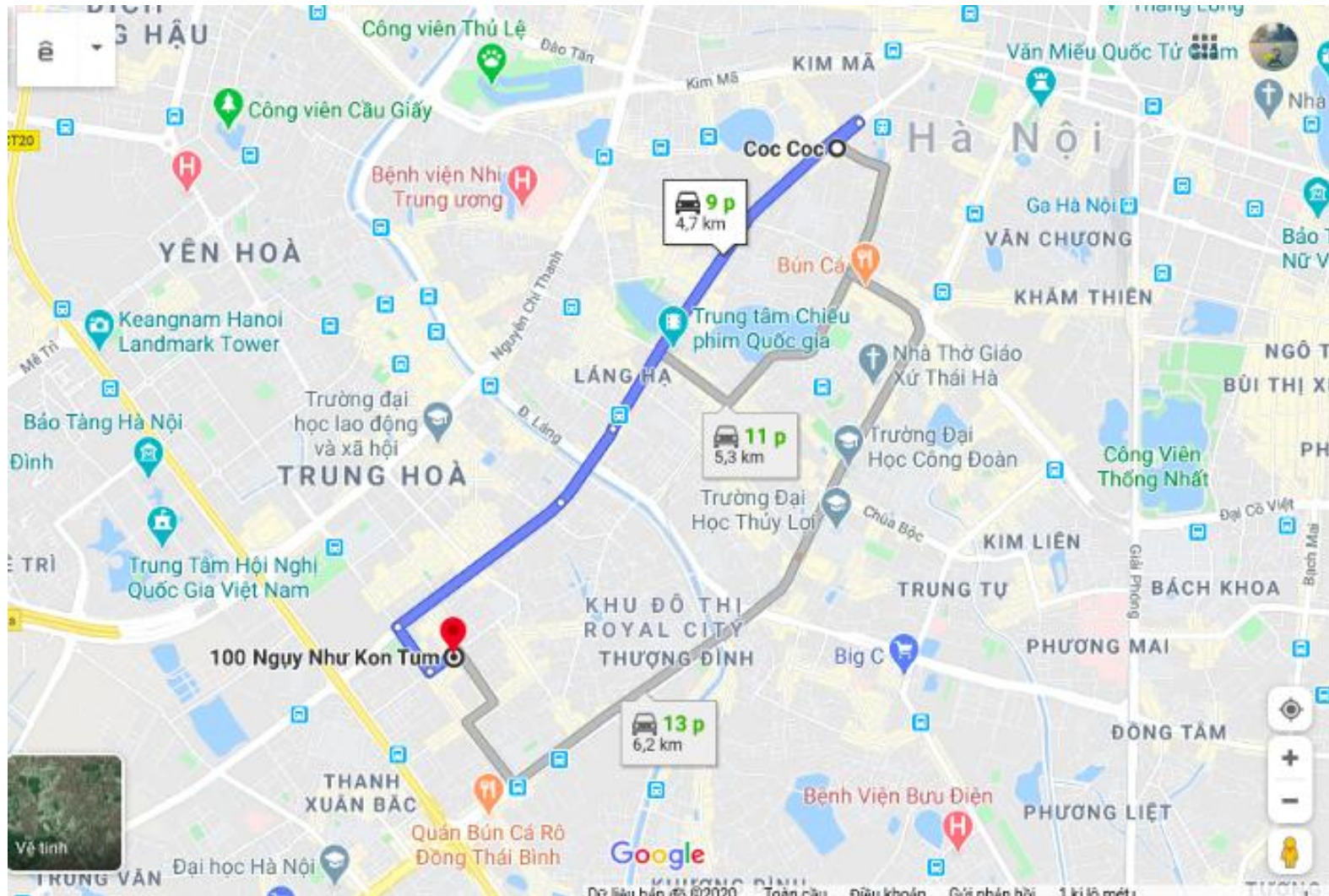
Ứng dụng của đồ thị

- ❖ Nghiên cứu sinh học: Các thành phần sinh học (protein, phân tử, gen) và các tương tác của chúng cũng tạo nên một đồ thị sinh học. Dựa vào đó người ta có thể tìm hiểu được quá trình trao đổi chất trong cơ thể, sự tương tác giữa các bộ phận khác nhau trên cơ thể



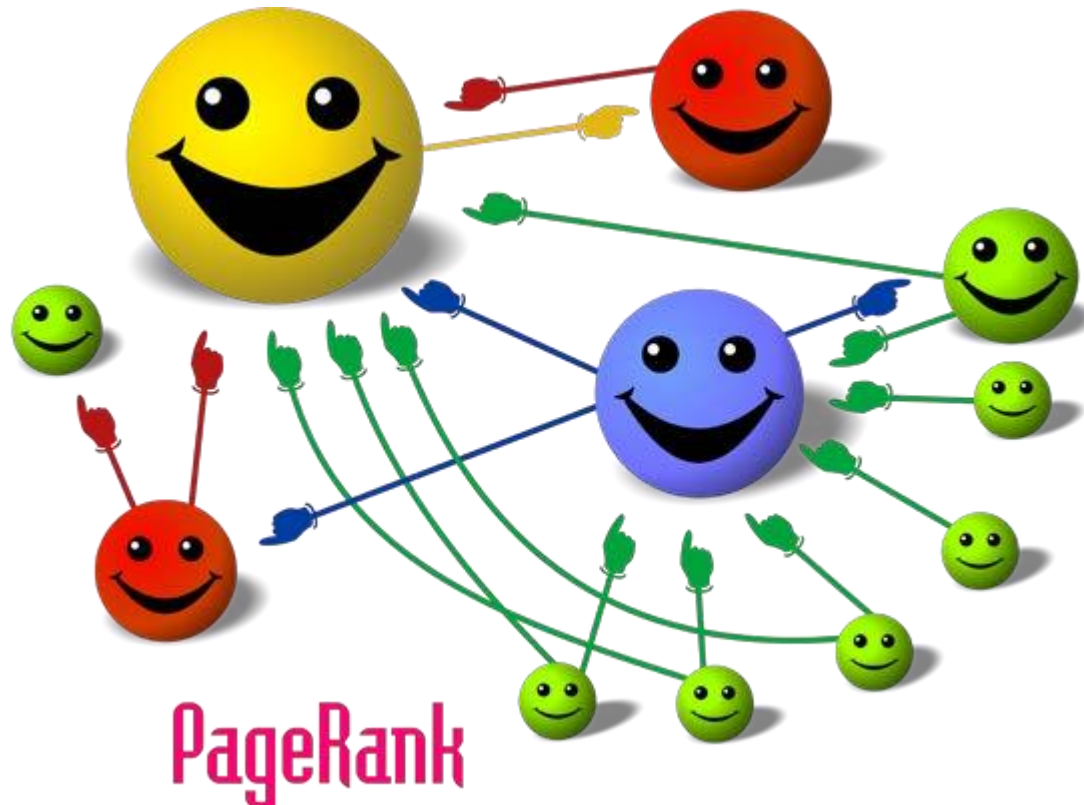
Ứng dụng của đồ thị

❖ Bài toán tìm đường đi



Ứng dụng của đồ thị

- ❖ WebSearch: Pagerank là thuật toán phân tích các liên kết được dùng trong Google Search để xếp hạng các trang web.



Ngôn ngữ Cypher

- ❖ Kiểu dữ liệu trong Cypher
- ❖ Quy cách đặt tên
- ❖ Sử dụng ghi chú
- ❖ Biểu diễn nút, biến nút
- ❖ Biểu diễn mối quan hệ, biến mối quan hệ
- ❖ Biểu diễn một mẫu (Pattern)
- ❖ Tạo cơ sở dữ liệu mẫu
- ❖ Truy vấn dữ liệu

Kiểu dữ liệu trong Cypher

Property Type	Structural types	Composite types
<ul style="list-style-type: none">• Number: Integer, Float• String• Boolean	Node: <ul style="list-style-type: none">• Id• Label(s)• Map (of properties)	List , a heterogeneous, ordered collection of values, each of which has any property, structural or composite type.
Temporal: <ul style="list-style-type: none">• Date• Time• LocalTime• DateTime• LocalDateTime• Duration	Relationship: <ul style="list-style-type: none">• Id• Type• Map (of properties)• Id of the start node• Id of the end node	Map , a heterogeneous, unordered collection of (<i>Key</i> , <i>Value</i>) pairs. <i>Key</i> is a String <i>Value</i> has any property, structural or composite type
Point	Path , an alternating sequence of nodes and relationships	

Quy cách đặt tên

- Tên đối tượng (nút, quan hệ, biến,...) bắt đầu bằng ký tự alphabet.
- Được chứa dấu gạch dưới “_”
- Không được bắt đầu bằng số.
- Không được chứa các ký tự đặt biệt như \$, @,...Ngoại trừ ký tự \$ đứng đầu là tham số.
- Có phân biệt chữ hoa, chữ thường

Sử dụng ghi chú (comment)

❖ Sử dụng hai dấu gạch chéo: //

❖ Ví dụ:

```
//data stored with this direction
```

```
CREATE (p:Person)-[:LIKES]->(t:Technology)
```

```
//query relationship backwards will not return results
```

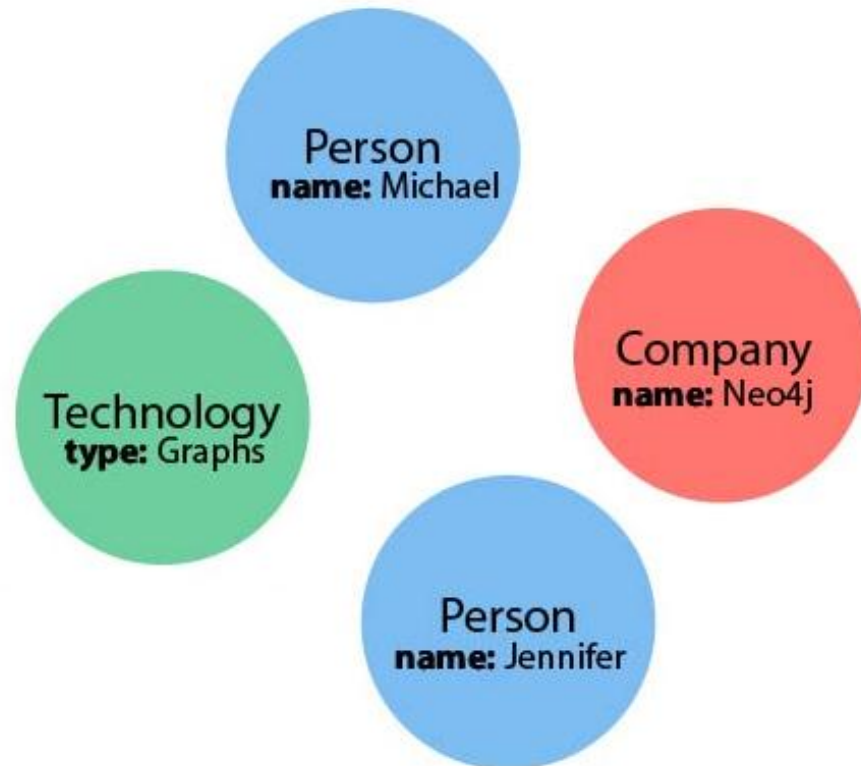
```
MATCH (p:Person)<-[:LIKES]-(t:Technology)
```

```
//better to query with undirected relationship unless sure of direction
```

```
MATCH (p:Person)-[:LIKES]-(t:Technology)
```


Biểu diễn các nút của đồ thị

- Mỗi nút được biểu diễn đồ họa như hình dưới. Ký hiệu (node).
- Trong hình có 4 nút có tên là: Jennifer, Michael, Graphs và Neo4j

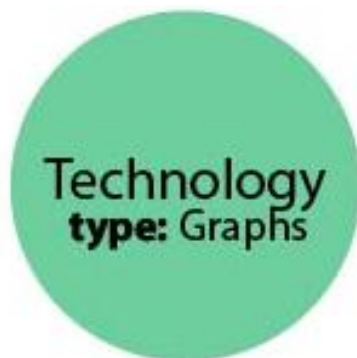


Biến nút (Node Variables)

- ❖ Sử dụng biến nút để tham chiếu đến một nút. Biểu diễn: (tên_biến). Nút không có tên biến là nút ẩn danh.
- ❖ Ví dụ:
 - (): nút ẩn danh
 - (p): biến có tên là p
 - (person): biến có tên là person
 - (t): biến có tên là t
 - (thing): biến có tên là thing

Nhãn nút (Node Labels)

- Những nút có cùng loại được gán cùng một nhãn
- Trong hình, các nhãn nút là: Person, Technology và Company
- Tương quan với SQL, mỗi nút tương tự như một dòng, mỗi nhãn nút tương tự như một bảng.
- Nếu không chỉ định nhãn thì truy vấn sẽ duyệt qua hết tất cả các nút trong đồ thị



Biểu diễn mối quan hệ

- ❖ Mỗi quan hệ biểu diễn sự quan hệ giữa 2 nút trên đồ thị.
- ❖ Tên mỗi quan hệ thường là động từ.
- ❖ Có 2 loại quan hệ:
 - Có hướng:
–[:RELATIONSHIP]-> hoặc <-[: RELATIONSHIP]-
 - Vô hướng:
-[: RELATIONSHIP]-
 - Trong đó: RELATIONSHIP là tên mỗi quan hệ.

Ví dụ quan hệ giữa 2 nút

$(p:\text{Person})-[:\text{IS_FRIEND_WITH}]\rightarrow(q:\text{Person})$

$(p:\text{Person})\leftarrow[:\text{IS_FRIEND_WITH}](q:\text{Person})$

$(p:\text{Person})-[:\text{IS_FRIEND_WITH}](q:\text{Person})$

Biến mối quan hệ

- ❖ Được dùng để tham chiếu đến một mối quan hệ.
 - Ký hiệu `-[variable]->`, `<-[variable]-` hoặc `-[variable]-`
 - Có thể chỉ định kiểu quan hệ như:
`-[variable:TYPE]->`
- ❖ Ví dụ:
 - `-[r]->` hoặc `-[r:LIKES]->`

Thuộc tính nút hoặc mối quan hệ

- ❖ Thuộc tính nút hoặc thuộc tính quan hệ được biểu diễn: {name:value}
- ❖ Ví dụ:
 - Thuộc tính nút:
(p:Person {name: 'Jennifer'})
 - Thuộc tính mối quan hệ:
-[rel:IS_FRIENDS_WITH {since: 2018}]->

Patterns trong Cypher

- ❖ Một mẫu (pattern) được tạo thành từ các nút và các mối quan hệ.
- ❖ Một mẫu phức tạp có thể được tách thành các mẫu nhỏ đơn giản hơn.

❖ Ví dụ:

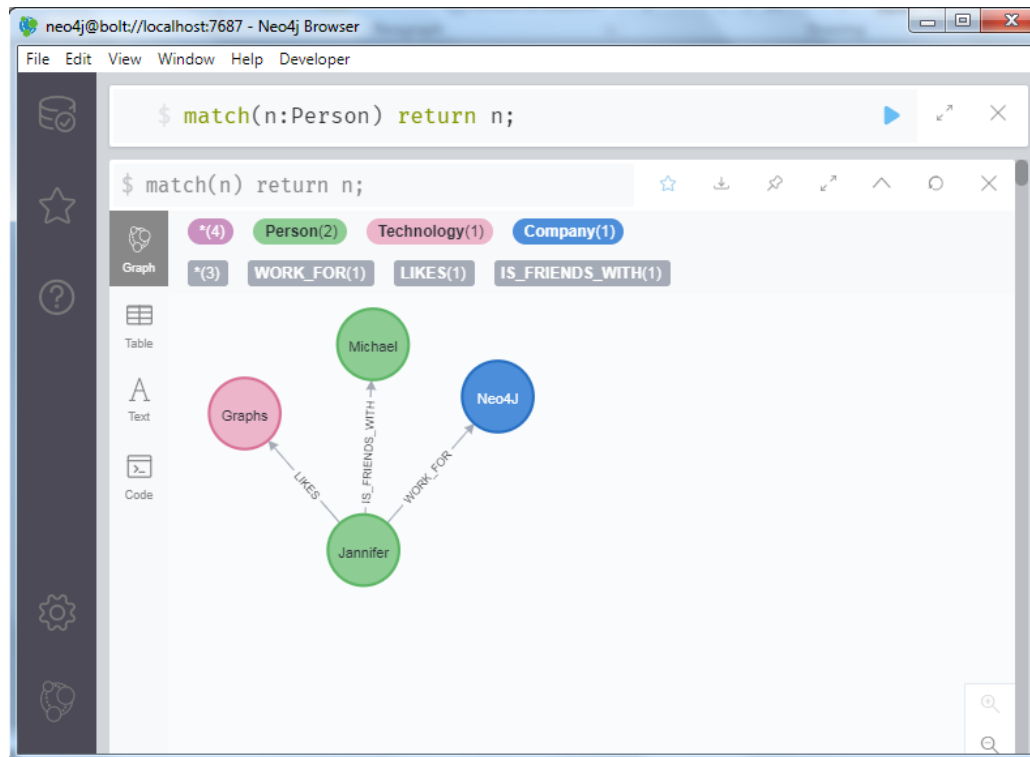
```
(p:Person {name: "Jennifer"})-[rel:LIKES]->(g:Technology  
{type: "Graphs"})
```

Các lệnh thao tác cơ bản

- ❖ Thao tác trên cửa sổ lệnh
- ❖ Thao tác trên nút
 - Thêm/xóa/sửa nút
- ❖ Thao tác trên mối liên kết
 - Thêm/xóa/sửa liên kết

Thao tác trên cửa sổ lệnh

- Start cơ sở dữ liệu cần thao tác trên Neo4J Desktop, sau đó mở công cụ Neo4J Browser từ Neo4J Desktop.
- Gõ lệnh thao tác ở khung nhập phía trên. Xuống dòng lệnh bằng phím Shift+Enter. Thực thi bằng phím Ctrl+Enter hoặc click vào nút thực thi màu xanh bên phải.



Thao tác trên nút

- Thêm nút mới:

\$ Create (a) //Thêm một nút mới ko thuộc tính, ko nhãn

\$ Create (b),(c) //Thêm 2 nút

\$ create (p:Person) //Thêm nút với nhãn Person

\$ Create(:Person) //Thêm nút với nhãn Person không chỉ định biến

\$ create (p:Person:VietNam) return p //Thêm nút với 2 nhãn là Person và VietNam, hiển thị nút vừa tạo bằng lệnh return

\$ create (p:Person {name: 'Andy', title: 'Developer'}) return p
//Thêm nút với nhãn Person và 2 thuộc tính name và title, hiển thị nút p vừa tạo

Thao tác trên nút (tt)

- Xóa nút:

\$ MATCH (p:Person{name:'Andy'}) DELETE p; //xóa nút thuộc nhãn Person có tên là Andy

\$ MATCH (p:Person {name: 'Andy'}) DETACH DELETE n
//xóa nút thuộc nhãn Person có tên là Andy và bất kỳ liên kết nối với nút.

\$ MATCH (n) DETACH DELETE n //Xóa tất cả các nút và mối liên kết

\$ MATCH (p) WHERE NOT (EXISTS(p.name)) DETACH
DELETE p; //Xóa các nút không có tên

\$ MATCH(p) WHERE ID(p)=4 DELETE p //Xóa nút có thuộc tính ID
= 4

Thao tác trên nút (tt)

- Thêm/sửa thuộc tính nút:

```
$ MATCH (n {name: 'Andy'}) //Thêm thuộc tính surname = 'Taylor'
```

```
SET n.surname = 'Taylor' //cho nút có tên là Andy
```

```
RETURN n.name, n.surname;
```

```
$ MATCH (p:Person {name: 'Jennifer'})
```

```
SET p.birthdate = date('1980-01-01'), p.gender='female'
```

```
RETURN p
```

- Xóa thuộc tính nút

```
$ MATCH (n {name: 'Andy'})
```

```
REMOVE n.surname, n.age //Xóa 2 thuộc tính là surname và age
```

```
RETURN n
```

Thao tác trên nút (tt)

- Thêm ràng buộc duy nhất:

```
$ CREATE CONSTRAINT ON (p:Person)  
  ASSERT p.name is UNIQUE
```

- Xóa ràng buộc:

```
$ DROP CONSTRAINT ON (p:Person)  
  ASSERT p.name IS UNIQUE
```

Thao tác trên mối liên kết

- Thêm liên kết:

```
$ MATCH (a:Person), (b:Person)  
  WHERE a.name = 'Andy' AND b.name = 'Michael'  
  CREATE (a)-[r:KNOWS{since:2018}]->(b)  
  RETURN a,b
```

- Xóa liên kết

```
$ MATCH (n {name: 'Andy'})-[r:KNOWS]->()  
  DELETE r
```

Thao tác trên mối liên kết (tt)

- Thêm/sửa thuộc tính của liên kết:

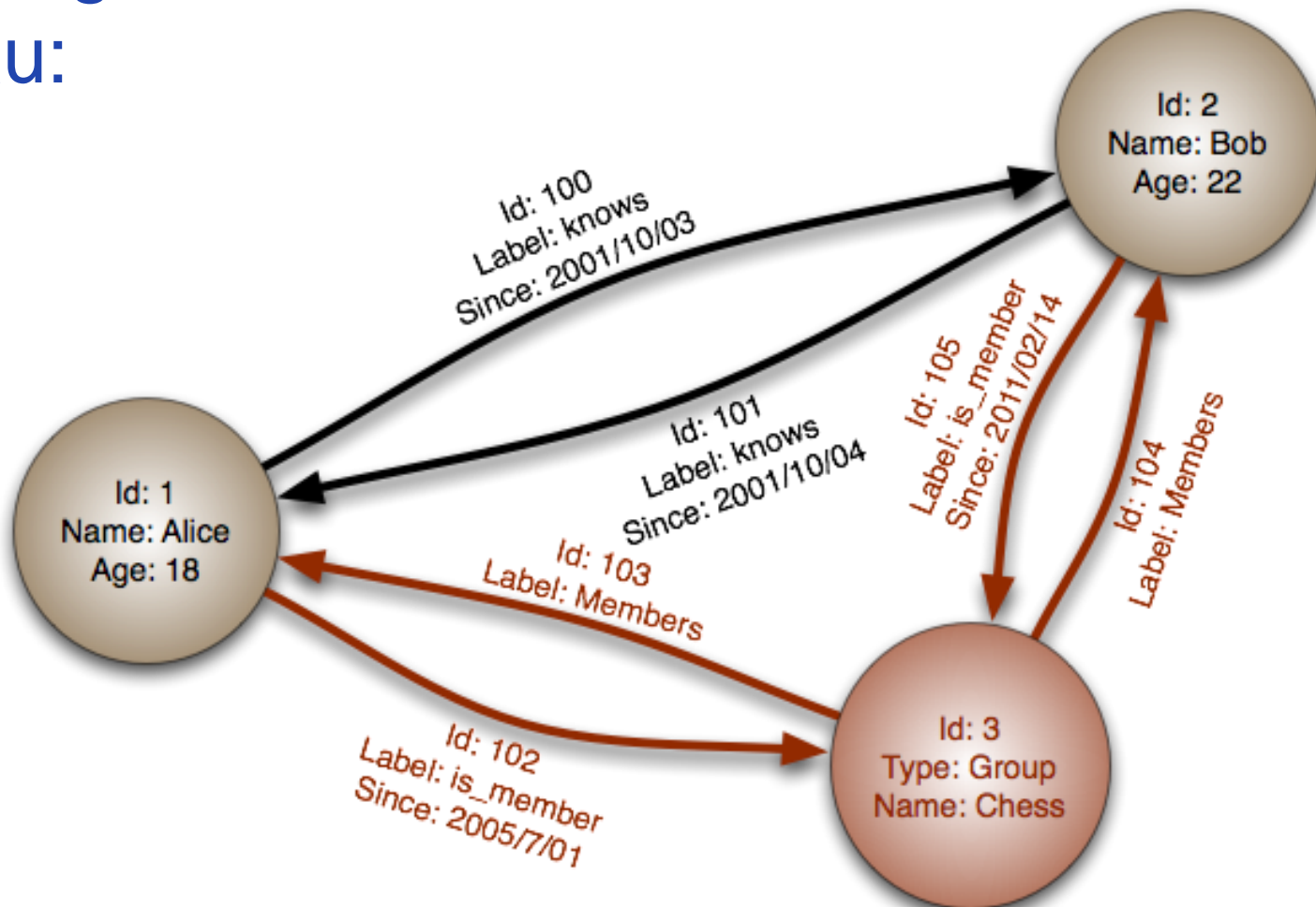
```
$ MATCH (n{name:'Andy'})-[r:KNOWS]->(m{name:'Michael'})  
  set r.since=2018  
  return n,m
```

- Xóa thuộc tính

```
$ MATCH (n{name:'Andy'})-[r:KNOWS]->(m{name:'Michael'})  
  REMOVE r.since  
  return n,m
```

Ví dụ tạo cơ sở dữ liệu đồ thị

❖ Dùng lệnh tạo cơ sở dữ liệu đồ thị như hình sau:



Tạo cơ sở dữ liệu đồ thị

❖ Tạo cơ sở dữ liệu đồ thị như ví dụ trên

- Tạo node tên là Jennifer có nhãn Person

```
$ Create(p:Person{name: 'Jennifer'})  
return p;
```

- Thiết lập id là khóa:

```
$ create constraint on (p:Person) assert p.id is unique
```

- Tạo node tên là Michael có nhãn Person

```
$ Create(p:Person{name: 'Michael'})  
return p;
```


Tạo cơ sở dữ liệu đồ thị (tt)

❖ Tạo cơ sở dữ liệu đồ thị như ví dụ trên

- Tạo node type là Graphs, có nhãn Technology

```
$ Create(p:Technology{type: 'Graphs'}) return p;
```

- Thiết lập id là khóa cho nhãn Technology:

```
$ create constraint on (p:Technology) assert p.id is  
unique
```

- Tạo node tên là Neo4J có nhãn Company

```
$ Create(p:Company{name: 'Neo4J'}) return p;
```

- Thiết lập id là khóa cho nhãn Company

```
$ create constraint on (p:Company) assert p.id is unique
```

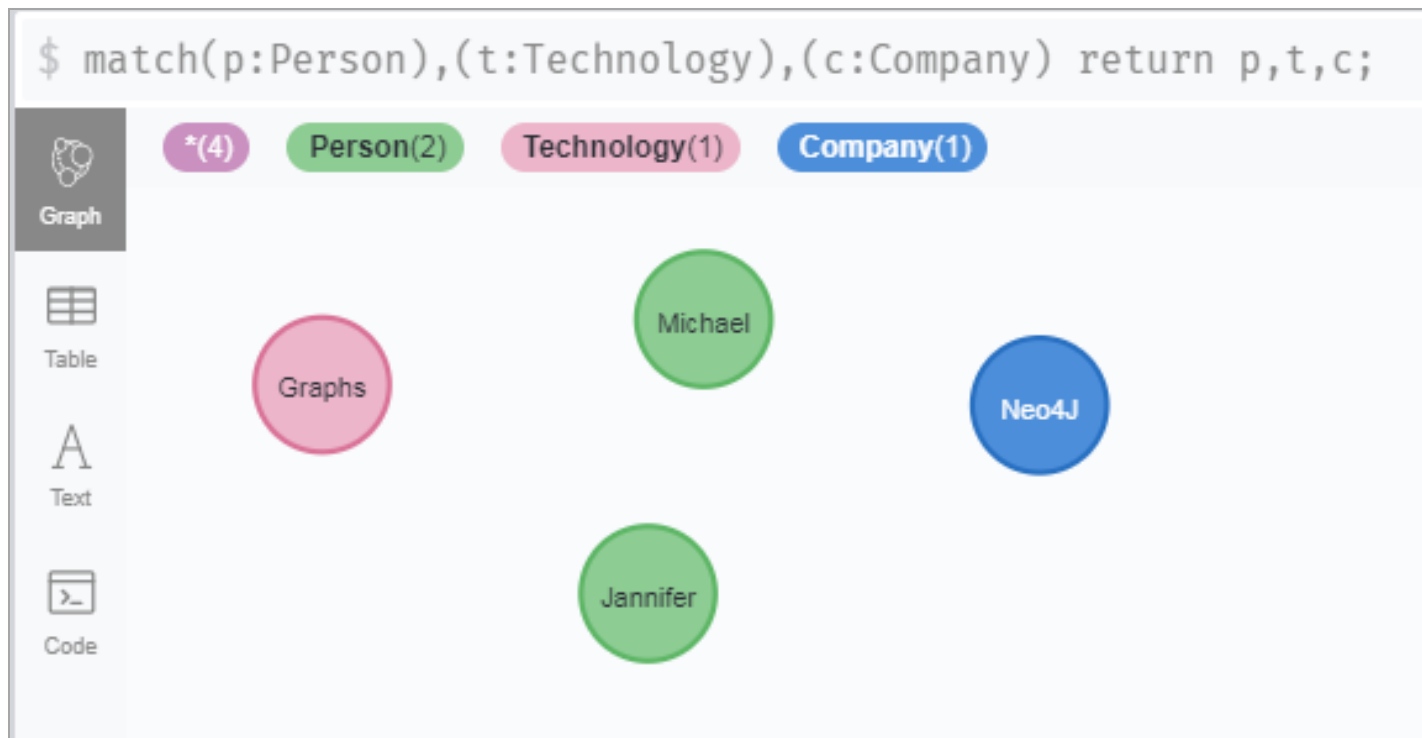
Tạo cơ sở dữ liệu đồ thị (tt)

❖ Hiển thị các nút đã tạo

```
$ match(p:Person),(t:Technology),(c:Company) return p,t,c;
```

■ Hoặc

```
$ match(n) return n;
```



Tạo các liên kết

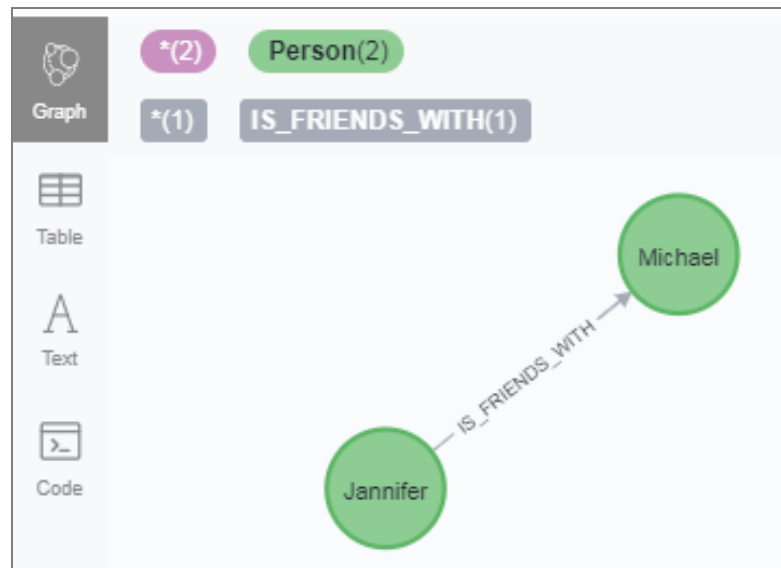
❖ Tạo liên kết giữa 2 nút Jennifer và Michael có thuộc tính *since:2018*

```
MATCH (a:Person),(b:Person)
```

```
WHERE a.name='Jennifer' and b.name='Michael'
```

```
CREATE (a)-[r:IS_FRIENDS_WITH{since:2018}]->(b)
```

```
RETURN a,b;
```



Tạo các liên kết (tt)

- Tạo liên kết giữa 2 nút Jennifer và Graphs không có thuộc tính.

```
match(a:Person),(b:Technology)
where a.name='Jennifer' and b.type='Graphs'
create (a)-[r:LIKES]->(b)
return a,b;
```

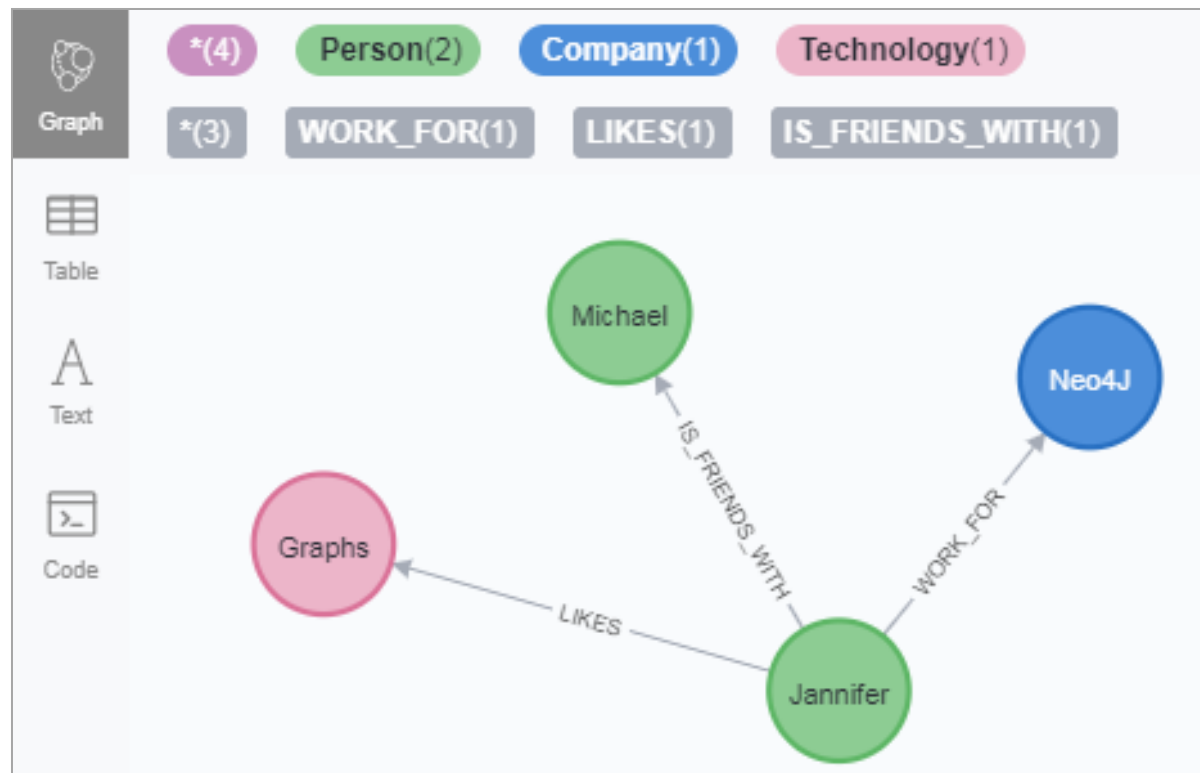
- Tạo liên kết giữa 2 nút Jennifer và Neo4J không thuộc tính

```
match(a:Person),(b:Company)
where a.name='Jennifer' and b.name='Neo4J'
create (a)-[r:WORK_FOR]->(b)
return a,b;
```

Cơ sở dữ liệu kết quả

- Xem các nút và liên kết

`$ match(a:Person),(b:Company),(c:Technology) return a,b,c;`



Truy vấn dữ liệu

//Hiển thị tất cả các nút

```
$ MATCH(p) RETURN p;
```

//Hiển thị các nút thuộc nhãn Person

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

//Hiển thị nút thuộc nhãn Person có tên là Jennifer

```
$ MATCH(j:Person{name:'Jennifer'}) RETURN j
```

//Hiển thị tên công ty mà người có tên Jennifer làm việc.

```
$ MATCH(:Person{name:'Jennifer'})-[:WORK_FOR]->  
  (c:Company)  
  RETURN c;
```

~~✗~~ Có thể dùng RETURN c.name để chỉ trả về tên công ty

Truy vấn label nút và type relationship

//Hiển thị label của nút

```
$MATCH (p{name:"Jennifer"}) return labels(p);
```

//Hiển thị tất cả các label có trong CSDL

```
$CALL db.labels();
```

//Hiển thị type của mỗi liên kết

```
$MATCH (p)-[r]->(q) return type(r);
```

Truy vấn với mệnh đề WHERE

//Hiển thị nút thuộc nhân Person có tên là Jennifer

```
$ MATCH(j:Person{name:'Jennifer'}) RETURN j
```

Tương đương khi sử dụng WHERE

```
$ MATCH(p:Person)  
  WHERE p.name='Jennifer'  
  RETURN p;
```

//Hiển thị tên công ty mà người có tên Jennifer làm việc.

```
$ MATCH(p:Person),(c:Company)  
  WHERE (p)-[:WORK_FOR]->(c)  
  RETURN c;
```


Truy vấn với mệnh đề WHERE (tt)

//Hiển thị những người không có tên là Jennifer

```
$ MATCH (j:Person)  
  WHERE NOT j.name = 'Jennifer'  
  RETURN j;
```

//Hiển thị những người có tuổi từ 20 đến 30

```
$ MATCH(p:Person)  
  WHERE 20<=p.age<=30  
  RETURN p;
```

//Hiển thị tên những người có thuộc tính birthdate

```
$ MATCH (p:Person)  
  WHERE exists(p.birthdate)  
  RETURN p.name
```

Truy vấn với mệnh đề WHERE (tt)

❖ Xử lý chuỗi:

//Điều kiện thuộc tính bắt đầu bằng ký tự 'M'

```
$ MATCH (p:Person)
```

```
WHERE p.name STARTS WITH 'M'
```

```
RETURN p.name;
```

//Thuộc tính có chứa ký tự 'a'

```
$ MATCH (p:Person)
```

```
WHERE p.name CONTAINS 'a'
```

```
RETURN p.name;
```

//Thuộc tính kết thúc với ký tự 'n'

```
$ MATCH (p:Person)
```

```
WHERE p.name ENDS WITH 'n'
```

```
RETURN p.name;
```

Truy vấn với mệnh đề WHERE (tt)

❖ Xử lý chuỗi:

//Thuộc tính có 2 ký tự đầu là Je

```
$ MATCH (p:Person)  
  WHERE p.name =~ 'Je.*'  
  RETURN p.name;
```

//Sử dụng từ khóa IN

```
$ MATCH (p:Person)  
  WHERE p.yearsExp IN [1, 5, 6]  
  RETURN p.name, p.yearsExp;
```

Exists và Not Exists với pattern

//Tìm những người là bạn của những người làm việc cho công ty có tên là Neo4J

```
$ MATCH (p)-[:IS_FRIENDS_WITH]-(q)
  WHERE EXISTS(
    (p)-[:WORK_FOR]->(:Company{name:'Neo4J'}))
  return q;
```

//Tìm những người bạn của Jennifer mà không làm việc cho công ty nào.

```
$ MATCH (p:Person)-[r:IS_FRIENDS_WITH]->(friend:Person)
  WHERE p.name = 'Jennifer'
  AND NOT exists((friend)-[:WORKS_FOR]->(:Company))
  RETURN friend.name
```

Một số truy vấn phức tạp

//Tìm những người thích Graphs ngoài Jennifer

```
$ MATCH (j:Person {name: 'Jennifer'})-[:LIKES]-  
      (graph:Technology {type: 'Graphs'})-[:LIKES]-(p:Person)  
      RETURN p.name;
```

// Tìm những người thích Graphs ngoài Jennifer và Jennifer cũng là bạn với những người đó.

```
$ MATCH (j:Person {name: 'Jennifer'})-[:LIKES]-  
      >(:Technology {type: 'Graphs'})<-[:LIKES]-(p:Person),  
      (j)-[:IS_FRIENDS_WITH]-(p)  
      RETURN p.name;
```

Aggregation trong Cypher

- Các hàm count, sum,

//Đếm số lượng người có email

```
$ MATCH(p:Person)
```

```
$ RETURN count(p.email);
```

Nếu dùng count(*) sẽ đếm hết các đối tượng kể cả có thuộc tính email bằng null

//Cho biết độ tuổi trung bình

```
$ MATCH(p:Person)
```

```
RETURN avg(p.age);
```

Sử dụng tương tự cho các hàm sum, min, max




Kết hợp giá trị với collect

//Hiển thị tên của mỗi người và danh sách bạn bè của người đó

```
$ MATCH (p:Person)-[:IS_FRIENDS_WITH]-  
      >(friend:Person)
```

```
RETURN p.name, collect(friend.name) AS friend
```

Hàm collect sẽ đưa các kết quả vào danh sách

\$ MATCH (p:Person)-[:IS_FRIENDS_WITH]-(friend:Person) RET...			⌵
 Table	p.name	friend	
	"Sally"	["John", "Jennifer"]	
	"Dan"	["Ann"]	
	"John"	["Sally", "Jennifer"]	
 Text	"Diana"	["Joe"]	
	"Jennifer"	["Sally", "John", "Ann", "Mark", "Melissa"]	
 Code	"Ann"	["Dan", "Jennifer"]	
	"Mark"	["Joe", "Jennifer"]	
	"Joe"	["Diana", "Mark"]	

Kết hợp giá trị với collect(tt)

//Hiển thị tên của mỗi người và số lượng bạn bè của người đó.

```
$ MATCH (p:Person)-[:IS_FRIENDS_WITH]->(friend:Person)
RETURN p.name, size(collect(friend.name)) AS numberOfFriends
```

\$ MATCH (p:Person)-[:IS_FRIENDS_WITH]->(friend:Person) RETURN p.name,		
Table	p.name	numberOfFriends
1	"Jennifer"	2

//Hiển thị tên của mỗi người, danh sách bạn bè tương ứng sao cho mỗi người trong danh sách này có số lượng bạn bè >1

```
MATCH (p:Person)-[:IS_FRIENDS_WITH]->(friend:Person)
WHERE size((friend)-[:IS_FRIENDS_WITH]-(:Person)) > 1
RETURN p.name, collect(friend.name) AS friends, size((friend)-[:IS_FRIENDS_WITH]-(:Person)) AS numberOfFoFs
```

\$ MATCH (p:Person)-[:IS_FRIENDS_WITH]->(friend:Person) WH...			
Table	p.name	friends	numberOfFoFs
	"Joe"	["Mark"]	2
	"Jennifer"	["Mark", "John", "Sally", "Ann"]	2
	"John"	["Sally"]	2

Sử dụng WITH

- WITH được dùng để chuyển giá trị từ phần này đến phần khác trong câu truy vấn.

```
$ MATCH (a:Person)-[r:LIKES]-(t:Technology)
```

```
  WITH a.name AS name, collect(t.type) AS technologies
```

```
  RETURN name, technologies
```

//Sử dụng WITH viết lại truy vấn:Hiển thị tên của mỗi người, danh sách bạn bè tương ứng sao cho mỗi người trong danh sách này có số lượng bạn bè >1

```
$ MATCH (p:Person)-[:IS_FRIENDS_WITH]->(friend:Person)
```

```
  WITH p, collect(friend.name) AS friendsList, size((friend)-  
  [:IS_FRIENDS_WITH]-(:Person)) AS numberOfFoFs
```

```
  WHERE numberOfFoFs > 1 RETURN p.name, friendsList,  
  numberOfFoFs
```

Sử dụng WITH (tt)

- WITH còn được sử dụng để thiết lập giá trị tham số trước câu truy vấn.

```
$ WITH 2 AS experienceMin, 6 AS experienceMax  
  MATCH (p:Person)  
  WHERE experienceMin <= p.yrsExperience <=  
  experienceMax  
  RETURN p
```

Sử dụng UNWIND

- UNWIND được dùng để tách một mảng thành các giá trị riêng lẻ để xử lý

//Với danh sách kỹ năng cho trước, cần tìm những người có kỹ năng tương ứng trong danh sách.


```
$ WITH ['Graphs','Query Languages'] AS techRequirements
UNWIND techRequirements AS technology
MATCH (p:Person)-[r:LIKES]-(t:Technology {type: technology})
RETURN t.type, collect(p.name) AS potentialCandidates
```

\$ WITH ["Graphs","Query Languages"] as techRequirements U...				
 Table	t.type	potentialCandidates		
	"Graphs"	["Diana", "Mark", "Melissa", "Jennifer"]		
 Text	"Query Languages"	["Diana", "Melissa", "Joe"]		

Sử dụng UNWIND (tt)

//Với danh sách số năm kinh nghiệm, tìm các ứng viên có số năm kinh nghiệm thuộc danh sách.

```
$ WITH [4, 5, 6, 7] AS experienceRange  
  UNWIND experienceRange AS number  
  MATCH (p:Person) WHERE p.yearsExp = number  
  RETURN p.name, p.yearsExp
```

\$ WITH [4, 5, 6, 7] as experienceRange UNWIND experienceR...		↓
 Table	p.name	p.yearsExp
	"Sally"	4
	"Jennifer"	5
	"John"	5
	"Dan"	6

Sắp xếp kết quả

- ORDER BY được dùng để sắp xếp kết quả truy vấn. Mặc định sắp tăng dần, Sử dụng DESC để sắp giảm dần.

//Sử dụng truy vấn trước: Với danh sách số năm kinh nghiệm, tìm các ứng viên có số năm kinh nghiệm thuộc danh sách.

```
$ WITH [4, 5, 6, 7] AS experienceRange
  UNWIND experienceRange AS number
  MATCH (p:Person)
  WHERE p.yearsExp = number
  RETURN p.name, p.yearsExp
  ORDER BY p.yearsExp DESC
```

Loại bỏ dòng trùng với Distinct

- Distinct được dùng để loại bỏ kết quả trùng

//Tìm những người có tài khoản email hoặc thích graphs hoặc thích query languages

```
$ MATCH (user:Person)
```

```
WHERE user.email IS NOT null
```

```
WITH user
```

```
MATCH (user)-[:LIKES]-(t:Technology)
```

```
WHERE t.type IN ['Graphs','Query Languages']
```

```
RETURN DISTINCT user.name
```

Giới hạn kết quả trả về với LIMIT

- Ví dụ: *Tìm 3 người đầu tiên có nhiều bạn nhất*

```
$ MATCH (p:Person)-[r:IS_FRIENDS_WITH]-(other:Person)  
  RETURN p.name, count(other.name) AS numberOfFriends  
  ORDER BY numberOfFriends DESC  
  LIMIT 3
```



Hết