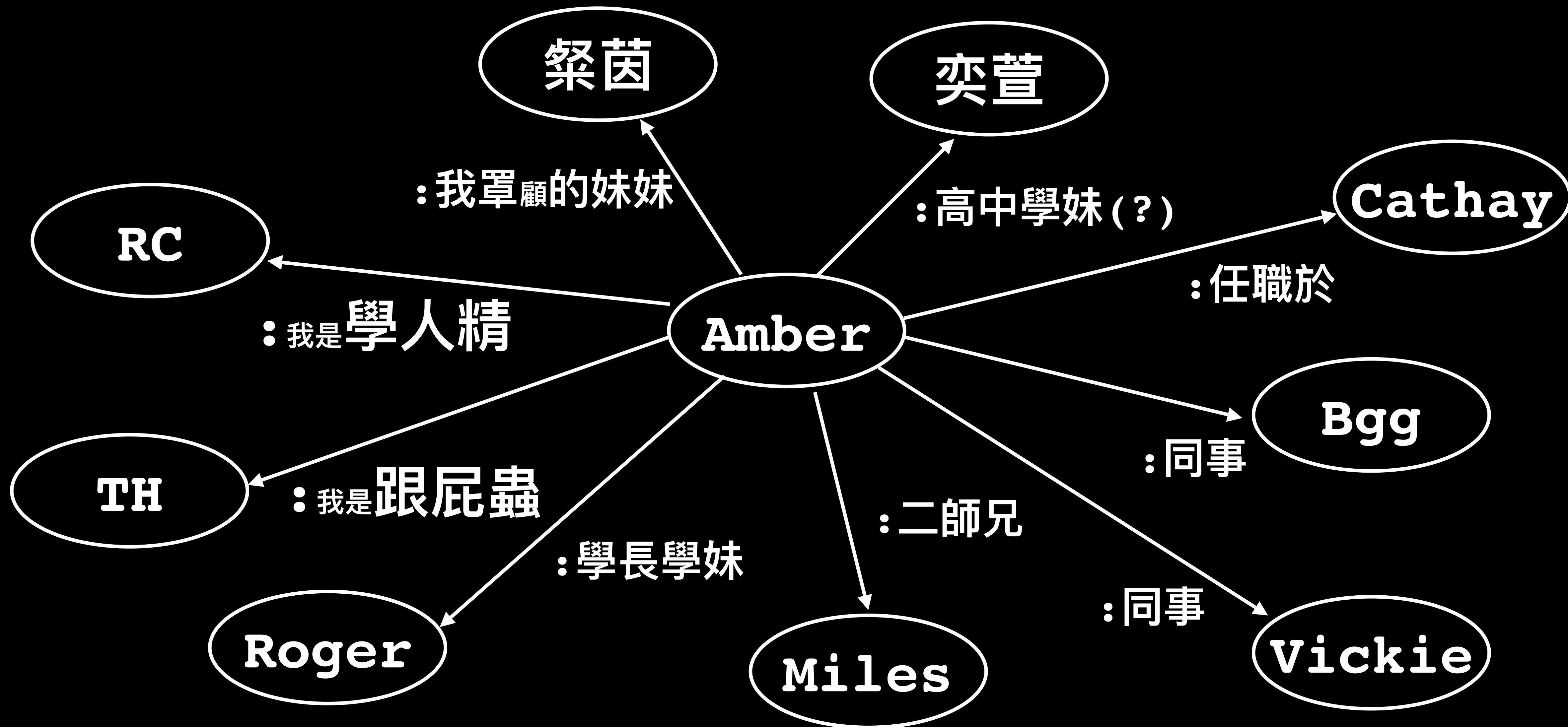


Neo4j

About Amber



Agenda

1. NoSQL Database: Graph DB
2. Neo4j + Cypher
3. Neo4j + Python
4. Graph Analysis: SNA

NoSQL = Not Only SQL

1. 資料用表格以外的形式儲存
2. 不同的查詢語言
3. 主流的NoSQL系統
 - 文件儲存：HBase
 - 鍵值儲存：MongoDB
 - 圖資料庫：Neo4j

Graph Database

1. 沒有Schema，根據圖學的架構設計
2. 只有2種資料型態：
 - 節點 (node)
 - 關係 (relationship)
3. 特別的查詢語言
 - 依據產品而有所差異
4. 被使用在社群網絡服務

Neo4j

1. 基於Java架構的資料庫系統

2. 資料類型：

- Node
- Label
- Property
- Relationship

3. 使用Cypher query language (CQL)

4. 已有商業與科學應用的實際使用案例

5. 提供企業版與社群版

Neo4j Example

*員編
姓名
性別
生日
地址
電話

員工

屬於

部門

*名稱
業務內容
成立年
編制人數

擁有

技能

*名稱
類別

*表示不可重複

Install Neo4j

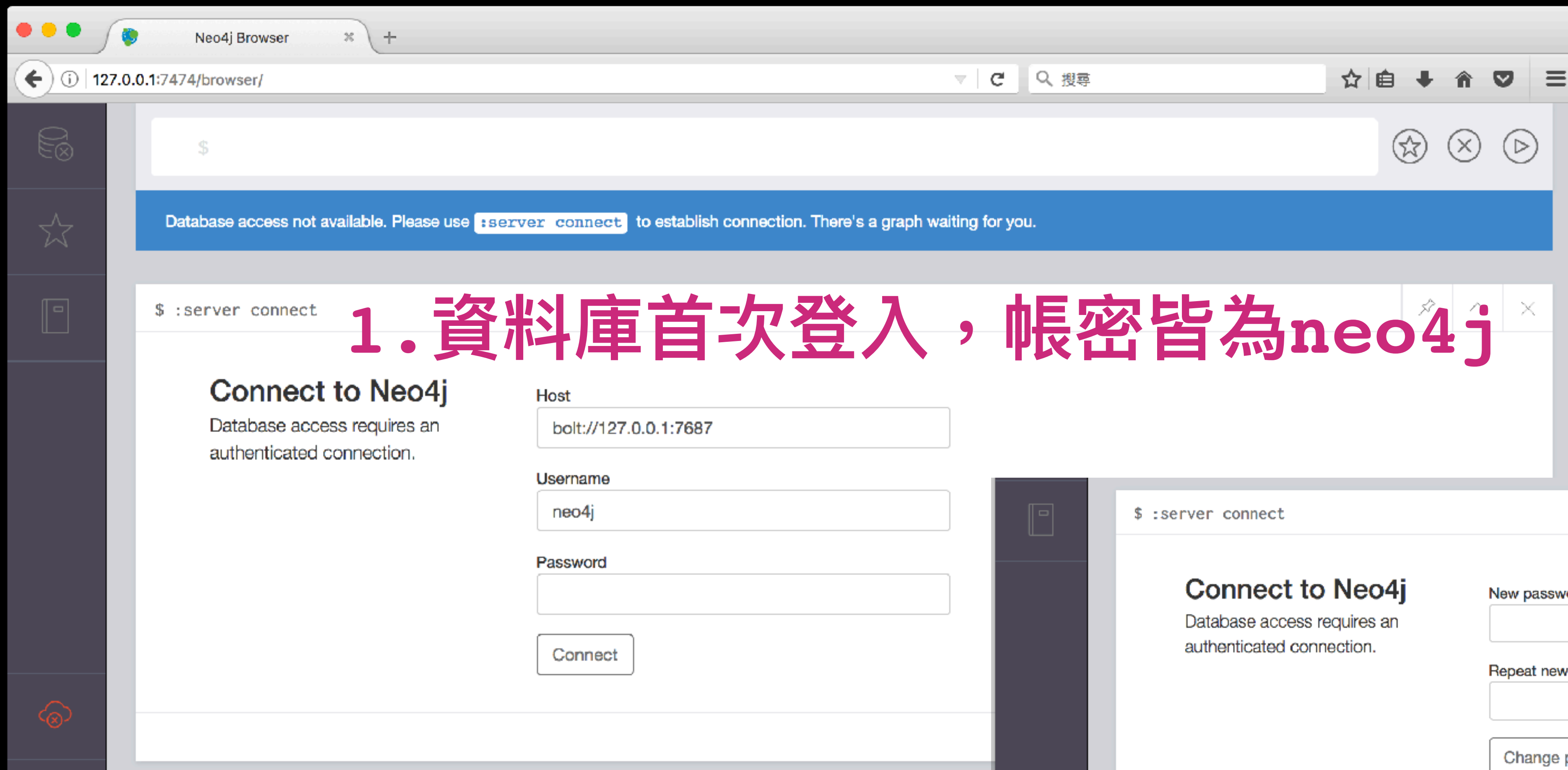
1. Download Neo4j Community Edition

<https://neo4j.com/download/>

2. Install Python to Neo4j Package:

```
$ pip install py2neo
```


Install Neo4j



1. 資料庫首次登入，帳密皆為neo4j



2. 設定新密碼

.neo4j.conf 設定檔

Neo4j 支援3種網路協定 (Protocol)

Bolt connector

dbms.connector.bolt.enabled=true

#dbms.connector.bolt.tls_level=OPTIONAL

#dbms.connector.bolt.listen_address=:7687

HTTP默認的port是7474，Bolt默認的port是7687，必須在防火牆中允許遠端主機訪問這些port

HTTP Connector #dbms.connector.http.listen_address=0.0.0.0:7474

dbms.connector.http.enabled=true

#dbms.connector.http.listen_address=:{default.http.port}

HTTPS Connector

dbms.connector.https.enabled=true

#dbms.connector.https.listen_address=:{default.https.port}

網頁操作介面

127.0.0.1:7474/browser/ 搜尋

Database Information

Node Labels

* Customer

Relationship Types

* CREDIT_CARD GUARANTEE
INSURE TRANSFER

Property Keys

cust_json_prop cust_property
customer_id deg level_cnt
level_name

Connected as

Username: neo4j
Roles: admin
Admin: :server user add

Database

Version: 3.2.2
Edition: Community
Name: test3
Size: 4.86 MiB
Information: :sysinfo

Cypher查詢輸入區

```
1 match (n:Customer)
2 where n.deg > 10 return n
```

\$ match (n:Customer) where n.deg > 10 return n

*(18) Customer(18)

Cypher查詢結果顯示區

Displaying 18 nodes, 19 relationships.

\$:play start

檢視系統狀態資訊

\$:sysinfo

Store Sizes	
Array Store	8.00 KiB
Logical Log	877.57 KiB
Node Store	8.00 KiB
Property Store	39.84 KiB
Relationship Store	23.91 KiB
String Store	8.00 KiB
Total Store Size	1.16 MiB

ID Allocation	
Node ID	369
Property ID	880
Relationship ID	511
Relationship Type ID	4

Page Cache	
Faults	-
Evictions	-
File Mappings	-
Bytes Read	-
Flushes	-
Eviction Exceptions	-
File Unmappings	-
Bytes Written	-

Transactions	
Last Tx Id	-
Current	-
Peak	-
Opened	-
Committed	-

Cypher

其實很像 SQL

Cypher

基本查詢：MATCH, CREATE, WHERE, RETURN

範例：

```
MATCH (user)-[:Birth_Place]->(country)  
RETURN user, country
```



```
(user)-[:Birth_Place]->(country)
```

CREATE 建立新節點

變數 標籤 屬性

↑ ↑ ↑

```
CREATE (n:Employee {Name:"Eric",  
Gender:"M"});
```

建立1個標籤名稱是Employee的節點，並寫入屬性

MATCH 與 RETURN

```
MATCH (n:Employee) RETURN n LIMIT 100;
```

查詢所有標籤是Employee的節點，並回傳其中100個

```
MATCH (n:Employee) RETURN n.Gender;
```

查詢所有標籤是Employee的節點，列出該節點的Gender欄位值

RETURN 格式化查詢結果

1. 排序：

- 加上 **ORDER BY**
- 預設是升冪排序，可用 **DESC** 降冪排序

2. 列出不重複的結果：**DISTINCT**

3. 計算數目：**COUNT()**

WHERE 附加條件

```
MATCH (n:Employee) WHERE n.Name="Eric" RETURN n;
```

查詢name屬性符合特定字串的Employee標籤節點

```
MATCH (n:Employee) WHERE n.Name=~"E.*" RETURN n;
```

查詢節點，配合正規表達式搜尋屬性內容

```
MATCH (n:Employee) WHERE n.Name STARTS WITH "Er"  
RETURN n;
```

查詢節點，使用資料庫引擎內建的查詢模組

更多 WHERE 語法

- 運算子: `n.Age >= 25`
- 範圍: `n.Age > 25 AND n.Age < 40`
- 篩選 NULL 的項目: `IS NULL`
- 否定: `NOT`
- 或: `OR`

CREATE 建立新關係

```
CREATE (e:Employee {Name:"Eric"})-[ :WORK_IN ]->
(c:Company {Name:"Cathay Bank"});
```

建立節點間的一對一單向關係

```
CREATE (s:Skill {Name:"Neo4j"})<- [ :KNOWS ] -
(e:Employee {Name:"Eric"})-[ :WORK_IN ]-> (c:Company
{Name:"Cathay Bank"});
```

建立三個節點，其中一個節點向外指向另外兩個節點的關係

CREATE 建立新關係

```
MATCH (e:Employee {Name:"Eric"}), (s:Skill) WHERE  
s.Name="Neo4j" CREATE UNIQUE (e)-[:KNOWS]->(s);
```

搜尋現有符合條件的節點，將他們建立關係

加入關係的查詢

```
MATCH (e:Employee)-[:KNOWS]->(s:Skill) RETURN  
e.Name, s.Name;
```

查詢符合此關係的兩節點，僅顯示兩個節點的Name屬性

```
MATCH (e:Employee)-[:KNOWS]->(s:Skill) WHERE  
s.Name="Neo4j" RETURN e.Name;
```

查詢符合此關係的兩節點，其中skill標籤的節點有條件限制，
查詢結果僅顯示Employee標籤節點的Name屬性

```
MATCH (e:Employee)-[r]->(s:Skill) RETURN r;
```

查詢兩個節點之間存在哪一種關係

其他

可以將csv檔案
透過Cypher匯入至Neo4j

<http://jude90.github.io/2016/01/14/csv-to-cypher.html>

先來一點參與感

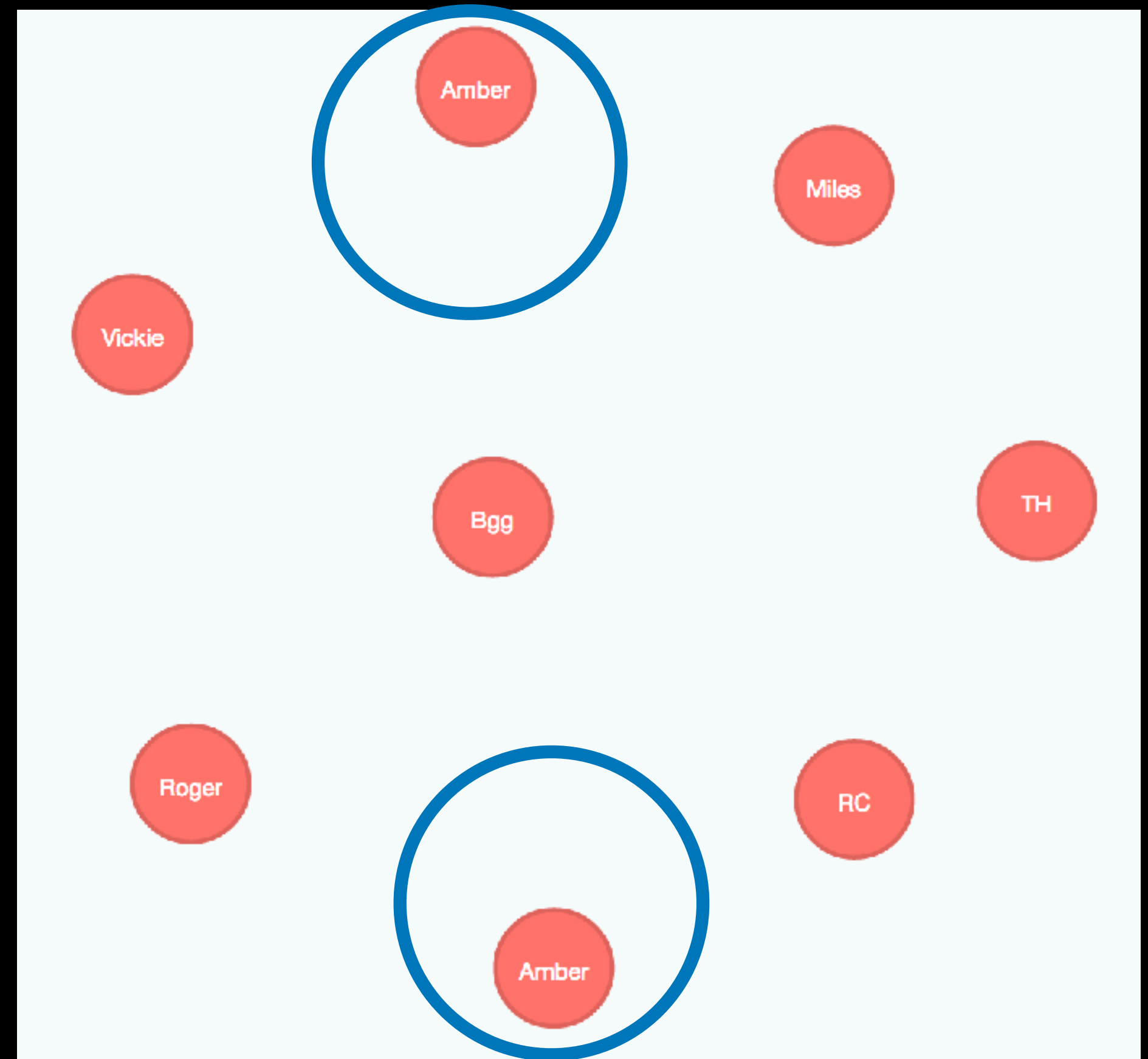
避免重複的節點

```
CREATE CONSTRAINT ON (n:Person) ASSERT  
n.name IS UNIQUE;
```

預先設定條件限制，規定特定屬性不得重複
加入限制條件之後，會自動建立相關索引
要查詢所有定義的限制，語法為：`schema`

ERROR Neo.ClientError.Schema.ConstraintValidationFailed

Node(0) already exists with label `Person` and property `name` = 'Amber'



建立索引

```
CREATE INDEX ON :Person(name);
```

針對特定標籤的特定屬性建立索引

ERROR Neo.ClientError.Schema.ConstraintAlreadyExists

Label 'Person' and property 'name' have a unique constraint defined on them, so an index is already created that matches this.

使用 Neo4j 網頁介面

```
CREATE (n:Person {name:"Amber", gender:"F"});  
CREATE (n:Person {name:"TH", gender:"F"});  
CREATE (n:Person {name:"Vickie", gender:"F"});  
CREATE (n:Person {name:"RC", gender:"M"});  
CREATE (n:Person {name:"Miles", gender:"M"});  
CREATE (n:Person {name:"Roger", gender:"M"});  
CREATE (n:Person {name:"Bgg", gender:"M"});
```

使用 Neo4j 網頁介面

```
MATCH (n:Person {name:"Amber"})  
CREATE (n)-[:WORK_IN]-> (c:Company {name:"Cathay Bank"});
```

針對已存在的節點，建立新的關係

```
CREATE (n:Database {name:"Neo4j"})-[:language]->  
(m:Programming {name:"Cypher"});
```

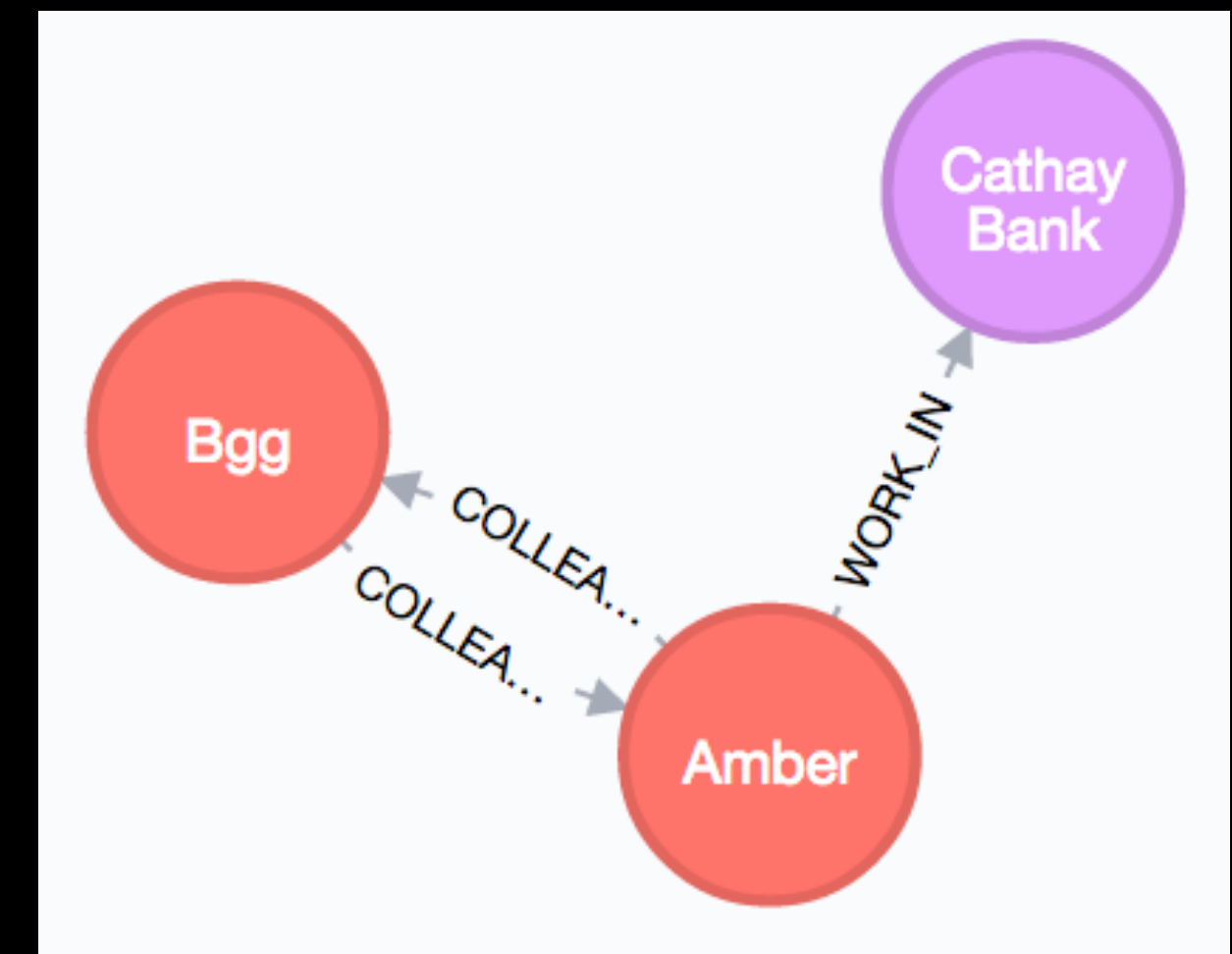
建立新節點，以及新節點之間的關係

使用 Neo4j 網頁介面

```
MATCH (n1:Person {name:"Amber"}), (n2:Person {name:"Bgg"})  
CREATE (n1)-[:COLLEAGUE]->(n2)
```

```
MATCH (n1:Person {name:"Amber"}), (n2:Person {name:"Bgg"})  
CREATE (n1)<-[:COLLEAGUE]-(n2)
```



```
MATCH (n:Person{name:"Amber"})-[r]-(m)  
RETURN n,r,m
```



創建關係時，必須有方向性；但在做查詢時，可以不限定方向

查詢所有使用的標籤

MATCH (n) RETURN DISTINCT labels(n);
查詢所有的標籤，然後去重複後顯示

\$ MATCH (n) RETURN DISTINCT labels(n);	
 Table	labels(n)
	["Person"]
	["Company"]
 Text	

刪除資料

MATCH (n) - [r] - () DELETE r;

刪除所有關係

MATCH (n) DELETE n;

刪除節點 (欲刪除節點時，必須先刪除該節點已經存在的關係)

MATCH (n) - [r] - (q) DELETE n, r, q;

刪除節點、節點之間的關係

Neo4j API

API 簡介

1. 官方負責維護的驅動程式：

- Python
- JavaScript
- .Net
- Java

2. 第三方開發者製作的 Neo4j API：

- Perl 有 REST::Neo4p
- Python 有 **Py2neo**
- R 有 RNeo4j
- Ruby 有 neography, neo4j-core 或 neo4j.rb
- PHP

Hands On

下集待續



Agenda

1. Advanced Cypher
2. Neo4j + Gephi
3. Graph Analysis: SNA