

霍格沃兹测试学院 - 测试开发工程师的黄埔军校

---

# Neo4j 数据库

MrDong

---

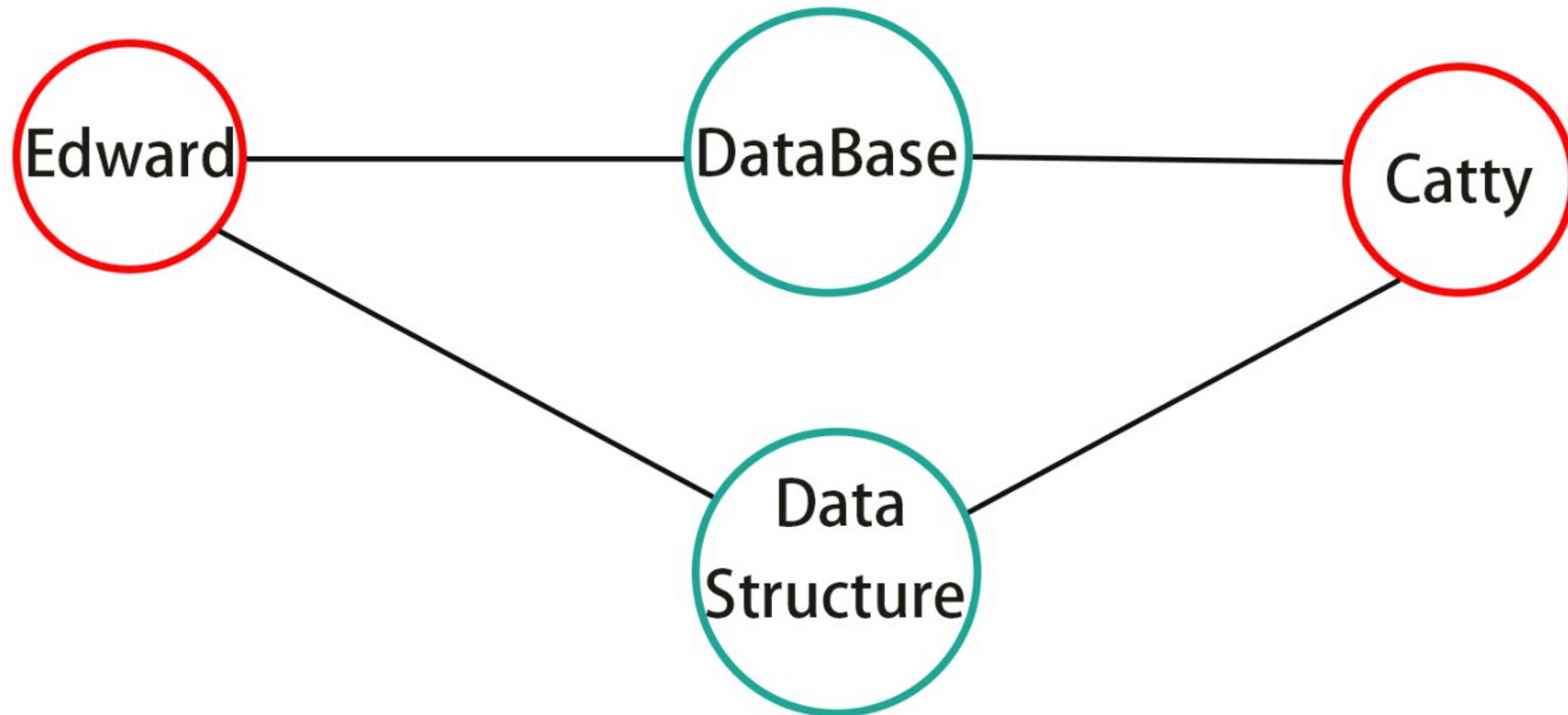


# 目录

- ❖ Neo4j 简介
- ❖ Neo4j 安装与使用
- ❖ 基础命令



# Neo4j







# Neo4j 定义

- ❖ Neo4j 是一个高性能的 ,NOSQL 图形数据库 , 它将结构化数据存储在网络上而不是表中



# Neo4j 特点

- ❖ 非关系型数据库
- ❖ NoSql
- ❖ CQL
- ❖ 遵循 SQL 语法，简单，且人性化



# 安装与使用



# 下载

- ❖ <https://neo4j.com/download-center/#community>





## Current Releases

[Enterprise Server](#)[Community Server](#)[Neo4j Desktop](#)

### Neo4j Community Edition 4.0.1

26 February 2020 [Release Notes](#) | [Read More](#)

OS	Download
Linux/Mac	<a href="#">Neo4j 4.0.1 (tar)</a> SHA-256
Windows	<a href="#">Neo4j 4.0.1 (zip)</a> SHA-256

### Neo4j Repositories

Debian/Ubuntu	<a href="#">Neo4j on Debian and Ubuntu</a> <a href="#">Cypher Shell</a>
Linux Yum	<a href="#">Neo4j Stable Yum Repo</a>



# 下载与使用

- ❖ 运行 Neo4j
  - ❖ neo4j console
- ❖ 输入默认账号密码
  - ❖ neo4j
- ❖ 修改账号密码



```
Microsoft Windows [版本 10.0.18363.657]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\yuruo>neo4j console
2020-03-03 23:19:40.854+0000 INFO  ===== Neo4j 4.0.1 =====
2020-03-03 23:19:40.858+0000 INFO  Starting...
2020-03-03 23:19:44.675+0000 INFO  Bolt enabled on localhost:7687.
2020-03-03 23:19:44.676+0000 INFO  Started.
2020-03-03 23:19:45.307+0000 INFO  Remote interface available at http://localhost:7474/
```



\$

Database access not available. Please use `:server connect` to establish connection. There's a graph waiting for you.

\$ :server connect

### Connect to Neo4j

Database access requires an authenticated connection.

Connect URL

Username

Password

Connect





```
$ :server connect
```

---


## Connect to Neo4j

Database access requires an authenticated connection.

New password

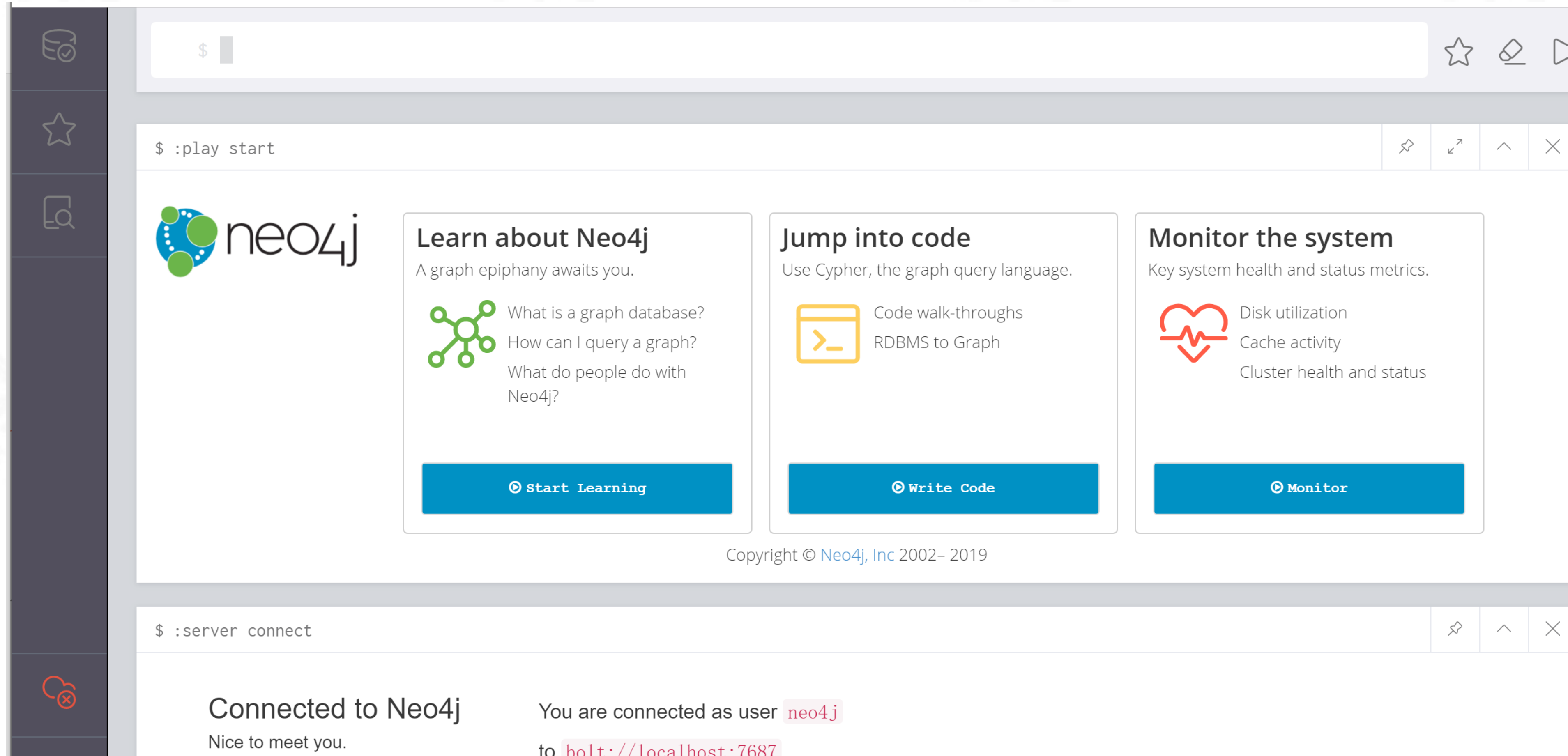
 

Repeat new password

Change password





The screenshot shows the Neo4j Desktop interface. On the left is a dark sidebar with icons for home, star, search, and a red connection indicator. The main area is divided into two terminal windows. The top window, titled '\$ :play start', displays the Neo4j logo and three main sections: 'Learn about Neo4j' with a green graph icon and questions like 'What is a graph database?', 'Jump into code' with a yellow code icon and 'Code walk-throughs RDBMS to Graph', and 'Monitor the system' with a red heart icon and 'Key system health and status metrics'. Each section has a blue button: 'Start Learning', 'Write Code', and 'Monitor'. The bottom window, titled '\$ :server connect', shows a successful connection message: 'Connected to Neo4j' and 'Nice to meet you.' followed by 'You are connected as user neo4j to bolt://localhost:7687'.

Copyright © Neo4j, Inc 2002–2019



# 创建节点



# 创建标签节点


- ❖ 打开浏览器
- ❖ 创建 emp 节点，标签为 Employee
  - ❖ `create(emp:Employee)`
- ❖ 创建 XiaoHong 节点，带有属性
  - ❖ `CREATE(XiaoHong:People{ no:1,age:10,gender:"M"})`





```
$ create (emp:Employee)
```

```
$ :play start
```



### Learn about Neo4j

A graph epiphany awaits you.

- What is a graph database?
- How can I query a graph?
- What do people do with Neo4j?

[Start Learning](#)

### Jump into code

Use Cypher, the graph query language.

- Code walk-throughs
- RDBMS to Graph

[Write Code](#)

### Monitor the system

Key system health and status metrics.

- Disk utilization
- Cache activity
- Cluster health and status

[Monitor](#)

Copyright © Neo4j, Inc 2002- 2019



```
$ CREATE (XiaoHong:People { no:1,age:10,gender:"M" })
```



```
$ CREATE (XiaoHong:People { no:1,age:10,gender:"M" })
```



Table

Added 1 label, created 1 node, set 3 properties, completed after 94 ms.



Code

Added 1 label, created 1 node, set 3 properties, completed after 94 ms.





### Database Information

**Node Labels**

(2) Employee People

**Relationship Types**

No relationships in database

**Property Keys**

age gender no

**Connected as**

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

**Database**

Version: 3.5.5  
Edition: Community  
Name: graph db

```
$ CREATE (XiaoHong:People { no:1,age:10,gender:"M" })
```

```
$ MATCH (n) RETURN n LIMIT 25
```

\*(2) Employee(1) People(1)

Graph

Table

Text

Code

3





\$ MATCH (n) RETURN n LIMIT 25



\*(2) Employee(1) People(1)

Graph

Table

Text

Code

People <id>: 20 age: 10 gender: M no: 1



# 添加属性



# 创建带属性的节点

❖ CREATE (book:Book {title:"book1",pages:340,price:250})



# 添加属性

- ❖ `match (book{title:"book1 "})`
- ❖ `set book.name = "hello"`
- ❖ `return book`





```
$ create(book:Book{title:"book1",pages:340,price:250})
```

\$ create(book:Book{title:"book1",pages:340,price:250})

Added 1 label, created 1 node, set 3 properties, completed after 149 ms.

Code

Added 1 label, created 1 node, set 3 properties, completed after 149 ms.



```
1 match (book{title:"book1"})  
2 set book.name="hello"  
3 return book
```



```
$ match (book{title:"book1"}) set book.name="hello" return book
```



**\*(1)** **Book(1)**

Graph



Table



Text



Code



**Book** <id>: 0 **name:** hello **pages:** 340 **title:** book1



# 查询



# 检索节点的 age 属性

- ❖ MATCH 需要与 RETURN 一起使用
  - ❖ MATCH (XiaoHong:People)
  - ❖ RETURN XiaoHong.age





# 检索节点的所有属性

- ❖ MATCH 与 RETURN 不能单独使用
  - ❖ MATCH (XiaoHong:People)
  - ❖ RETURN XiaoHong



```
1 MATCH (XiaoHong:People)
2 RETURN XiaoHong.age
```

\$ MATCH (XiaoHong:People) RETURN XiaoHong.age

XiaoHong.age	
	10

Started streaming 1 records after 1 ms and completed after 37 ms.



```
1 match (XiaoHong:People)
2 return XiaoHong
```

\$ match (XiaoHong:People) return XiaoHong


\*(1) People(1)

Graph

Table

Text

Code



Displaying 1 nodes, 0 relationships.



# 关系





# 什么是关系



# 创建关系

❖ CREATE (p1:Profile1)-[r1:LIKES]->(p2:Profile2)



```
$ CREATE (p1:Profile1{name:'XiaoHong'})-[r1:Friends]->
(p2:Profile2{name:'LiMei'})
```



```
$ CREATE (p1:Profile1{name:'XiaoHong'})-[r1:Friends]->(p2:Profile2{name:'LiMei'})
```



Table



Code

Added 2 labels, created 2 nodes, set 2 properties, created 1 relationship, completed after 11 ms.

Added 2 labels, created 2 nodes, set 2 properties, created 1 relationship, completed after 11 ms.



```
1 match(a:People),(b:People)
2 where a.name="xiaohong" And b.name="WangWu"
3 create (a)-[r1:Friends]->(b)
```



```
$ match(a:People),(b:People) where a.name="xiaohong" And b.name="WangWu" create (a)-[r1:...
```



Table



Code

Created 1 relationship, completed after 5 ms.

Created 1 relationship, completed after 5 ms.





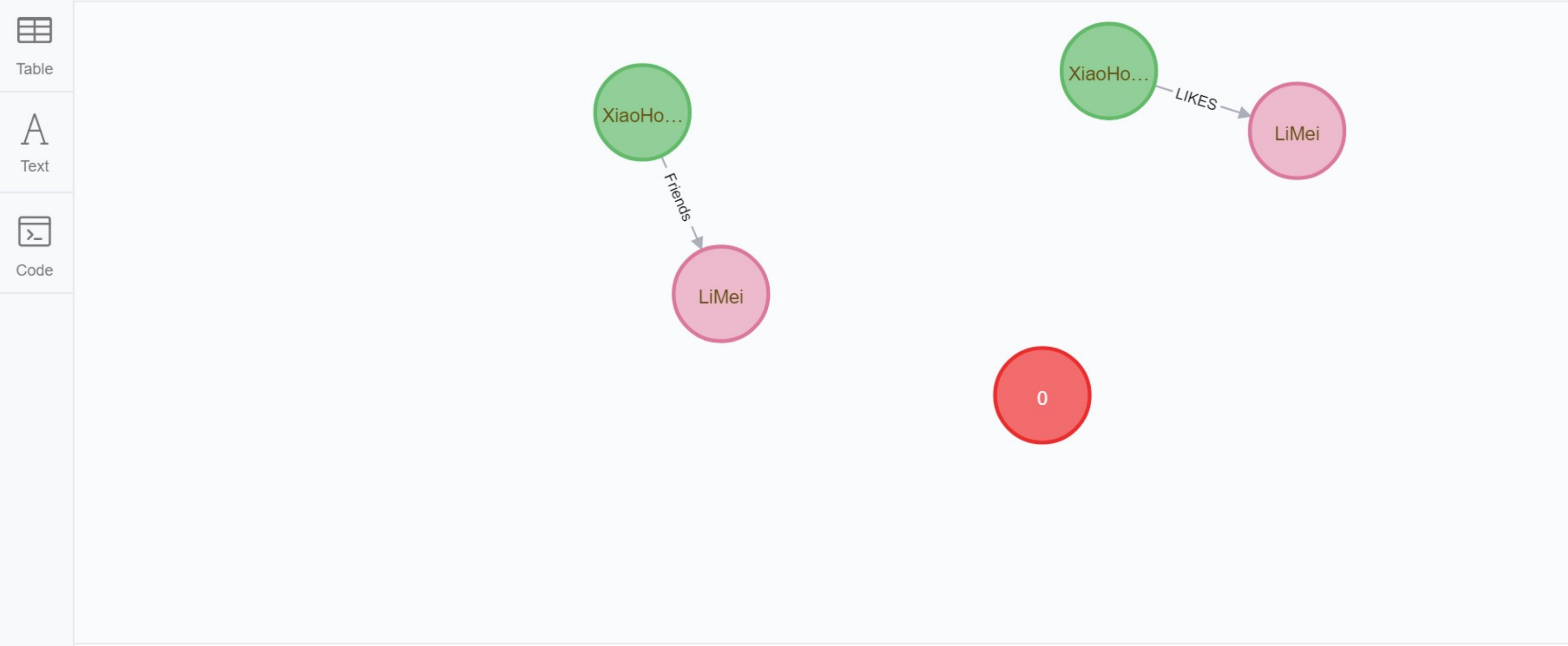
```
$ CREATE (p1:Profile1{name:'XiaoHong'})-[r1:Friends]->
(p2:Profile2{name:'LiMei'})
```

```
$ MATCH (n) RETURN n LIMIT 25
```

Navigation icons: download, share, zoom, scroll, refresh, close.

Entity filters: \*(8), Employee(1), Profile1(3), Profile2(3), People(1)

Relationship filters: \*(3), LIKES(2), Friends(1)



Displaying 8 nodes, 3 relationships.



# 删除节点和关系



# 删除相应属性下的节点

- ❖ MATCH (e: People) DELETE e



# 删除带有关系的节点

- ❖ MATCH (cc: Profile1)-[Friends]->(c:Profile2)
- ❖ DELETE cc,c,Friends





# 删除相应属性下的节点

- ❖ MATCH (e: People) DELETE e



```
$ MATCH (e: People) DELETE e
```

Deleted 1 node, completed after 1 ms.

Deleted 1 node, completed after 1 ms.

The image shows a screenshot of a database query interface. At the top, a search bar contains the Cypher query: `$ MATCH (e: People) DELETE e`. Below the search bar is a toolbar with icons for star, edit, and play. The main area displays the same query: `$ MATCH (e: People) DELETE e`. On the left side, there is a sidebar with two tabs: 'Table' (selected) and 'Code'. The 'Table' tab shows the result: 'Deleted 1 node, completed after 1 ms.'. The 'Code' tab is currently empty. At the bottom of the main area, the same result message is displayed: 'Deleted 1 node, completed after 1 ms.'.



```
1 MATCH (cc: Profile1)-[Friends]-(c:Profile2)
2 DELETE cc,c,Friends
```

\$ MATCH (cc: Profile1)-[Friends]-(c:Profile2) DELETE cc,c,Friends

Deleted 6 nodes, deleted 3 relationships, completed after 12 ms.

Deleted 6 nodes, deleted 3 relationships, completed after 12 ms.



# 删除属性





# 删除属性

- ❖ `match (book{title:"book1 "})`
- ❖ `Remove book.price`
- ❖ `Return book`



```
1 match (book{title:"book1"})  
2 remove book.price  
3 return book
```



```
$ match (book{title:"book1"}) remove book.price return book
```



Graph

\*(1)

Book(1)



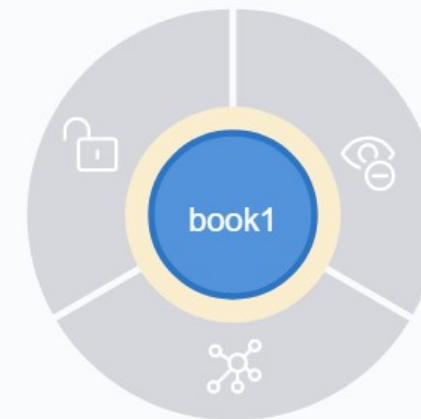
Table



Text



Code



Book <id>: 0 pages: 340 title: book1



# Delete 与 Remove

- ❖ DELETE 操作用于删除节点和关联关系
- ❖ REMOVE 操作用于删除标签和属性



# 额外补充 1

- ❖ 删除标签：
  - ❖ MATCH (m:Movie)
  - ❖ REMOVE m:Picture
- ❖ 排序：
  - ❖ MATCH (emp:Employee)
  - ❖ RETURN emp.empid,emp.name,emp.salary,emp.deptno
  - ❖ ORDER BY emp.name DESC





# 额外补充 2

- ❖ 更高级的查询：
  - ❖ MATCH (emp:Employee)
  - ❖ WHERE emp.name = 'Abc'
  - ❖ RETURN emp

