

Cypher란?

- Cypher는 Neo4j에서 노드, 관계, 속성 등을 생성하고 탐색할 수 있도록 설계된 선언적 (declarative) 그래프 쿼리 언어입니다.
- 사람이 읽고 쓰기 쉬운 문법으로 되어 있어서, 복잡한 그래프 쿼리도 직관적으로 표현할 수 있는 것이 큰 장점입니다.

기본 개념 요약

개념	설명	예시
노드 (Node)	개체 (사람, 기술 등)	<code>(p:Person {name: "경원"})</code>
관계 (Relationship)	노드 간의 연결	<code>[:KNOWS] , [:WORKS_AT]</code>
속성 (Property)	노드/관계의 정보	<code>{name: "윤서", age: 30}</code>
패턴 (Pattern)	노드와 관계의 구조 표현	<code>(a) - [:MARRIED_T0] -> (b)</code>

자주 쓰는 Cypher 키워드

키워드	기능
CREATE	노드 또는 관계 생성
MATCH	특정 패턴과 일치하는 그래프 요소 찾기
RETURN	결과 반환
WHERE	조건 필터링
MERGE	있으면 유지, 없으면 생성 (UPSERT)
SET	속성 추가/수정
DELETE	노드 또는 관계 삭제

Cypher vs SQL 비교

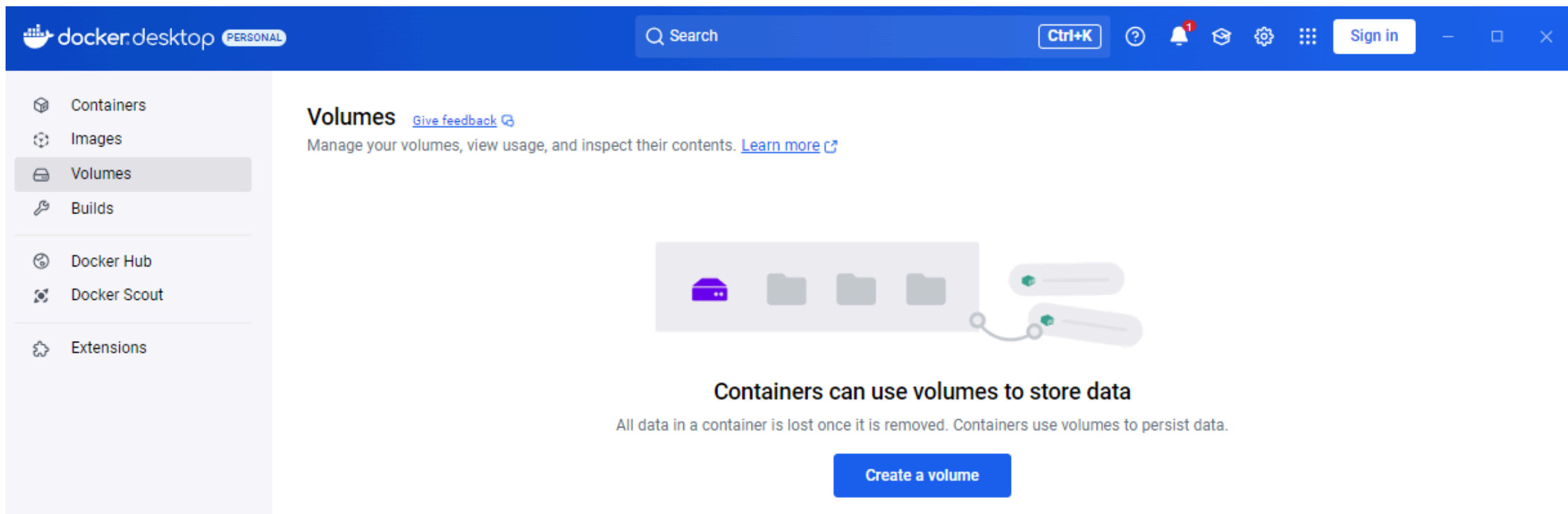
기능	SQL	Cypher
데이터 모델	테이블 기반	그래프 기반 (노드, 관계)
JOIN 방식	명시적으로 JOIN	관계 패턴으로 자연스럽게 표현
예시	<code>SELECT * FROM people</code>	<code>MATCH (p:Person) RETURN p</code>

정리

- Cypher는 그래프 탐색을 직관적이고 시각적으로 표현할 수 있는 쿼리 언어
- SQL보다 관계 표현이 훨씬 자연스럽고 간결함
- Neo4j를 활용하는 GraphRAG, 지식그래프, 추천 시스템, 네트워크 분석 등에서 핵심 도구로 사용됨

Neo4j 생성

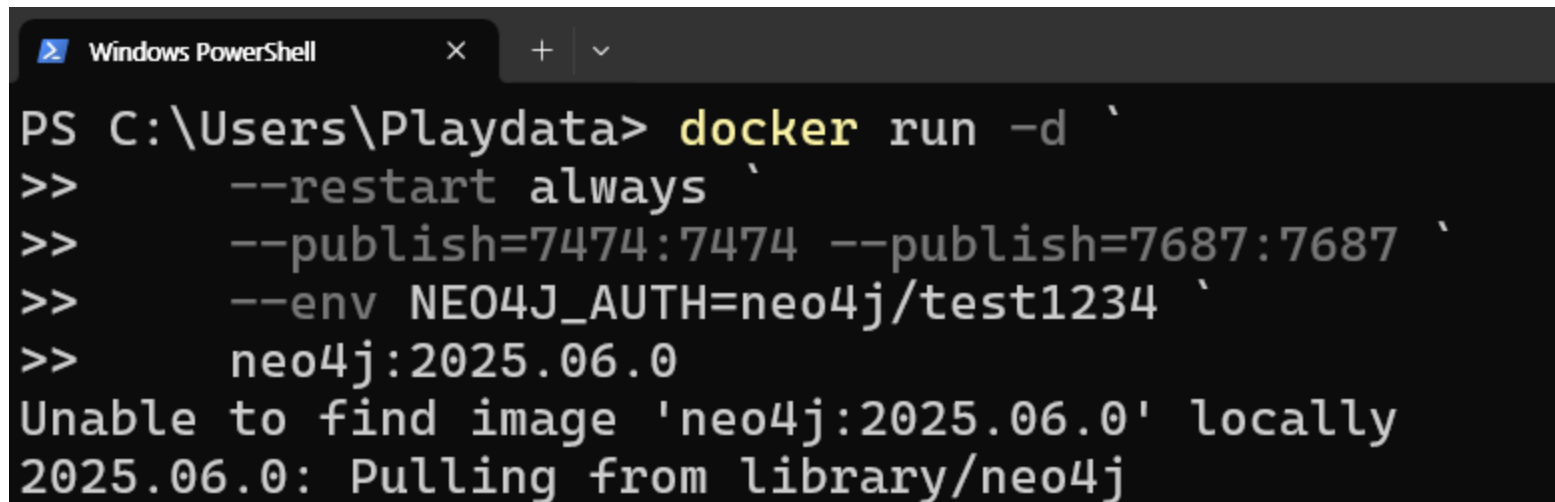
1. 단계: Docker desktop 실행



2. 단계: Neo4j Docker 생성 및 실행

윈도우용

```
docker run -d `
  --restart always `
  --publish=7474:7474 --publish=7687:7687 `
  --env NEO4J_AUTH=neo4j/test1234 `
  neo4j:2025.06.0
```



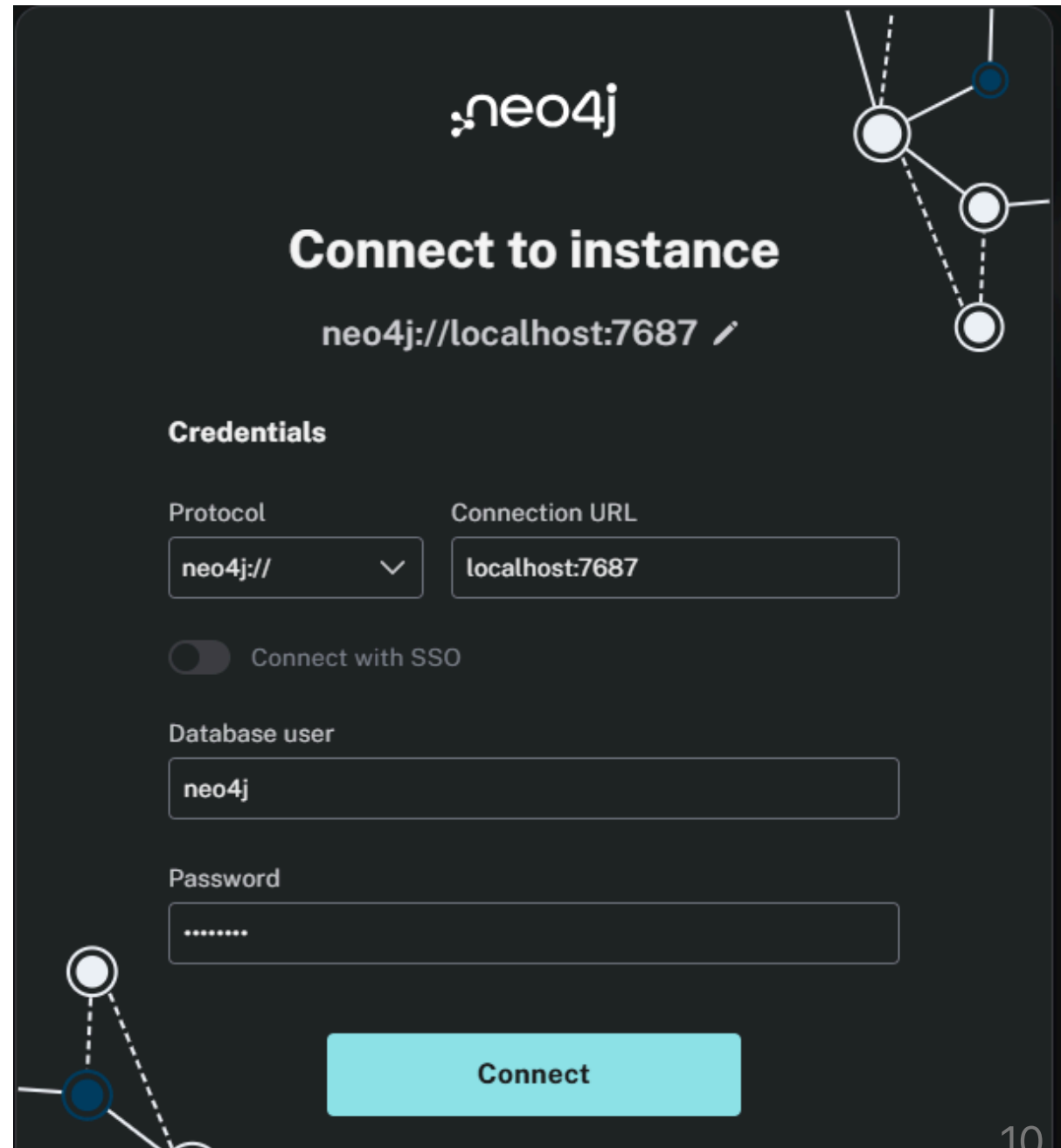
```
Windows PowerShell
PS C:\Users\Playdata> docker run -d `
>>   --restart always `
>>   --publish=7474:7474 --publish=7687:7687 `
>>   --env NEO4J_AUTH=neo4j/test1234 `
>>   neo4j:2025.06.0
Unable to find image 'neo4j:2025.06.0' locally
2025.06.0: Pulling from library/neo4j
```


3. 단계: Neo4j 접속

The screenshot shows the Docker Desktop interface. The top bar is blue with the Docker logo, 'docker.desktop PERSONAL', a search bar, 'Ctrl+K', and various icons including a help icon, a notification bell with a red '1', a Docker Scout icon, a settings gear, a grid icon, and a 'Sign in' button. The left sidebar contains navigation links: Containers (selected), Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is titled 'Containers' with a 'Give feedback' link. Below the title is the text 'View all your running containers and applications. [Learn more](#)'. There are two summary cards: 'Container CPU usage' showing '1.00% / 1600% (16 CPUs available)' and 'Container memory usage' showing '769.1MB / 15.08GB', both with information icons and a 'Show charts' link. Below these is a search bar and a toggle switch for 'Only show running containers'. A table lists the running containers. The first container, 'reverent_bassi', is highlighted with a green dot. Its 'Port(s)' column shows '7474:7474' with a red box and a red arrow pointing to it, and '7687:7687' with a red box and a red arrow pointing to it. The table has columns for Name, Container ID, Image, Port(s), CPU (%), Last started, and Actions.

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	reverent_bassi	8e2712fa2ea0	neo4j:2025.06.0	7474:7474 7687:7687 Show less	1.03%	3 minutes ago	

- 접속 url: <http://localhost:7474>
- database user: neo4j
- password: test1234



The image shows the Neo4j web interface for connecting to an instance. At the top, the Neo4j logo is displayed. Below it, the heading "Connect to instance" is followed by the connection string "neo4j://localhost:7687" with an edit icon. The "Credentials" section contains a "Protocol" dropdown menu set to "neo4j://", a "Connection URL" text box containing "localhost:7687", a toggle switch for "Connect with SSO" which is currently off, a "Database user" text box containing "neo4j", and a "Password" text box with masked characters. A large blue "Connect" button is positioned at the bottom right of the form area. The interface is decorated with a dark background and stylized network diagrams in the corners.

neo4j

Connect to instance

neo4j://localhost:7687 /

Credentials

Protocol Connection URL

neo4j:// localhost:7687

☐ Connect with SSO

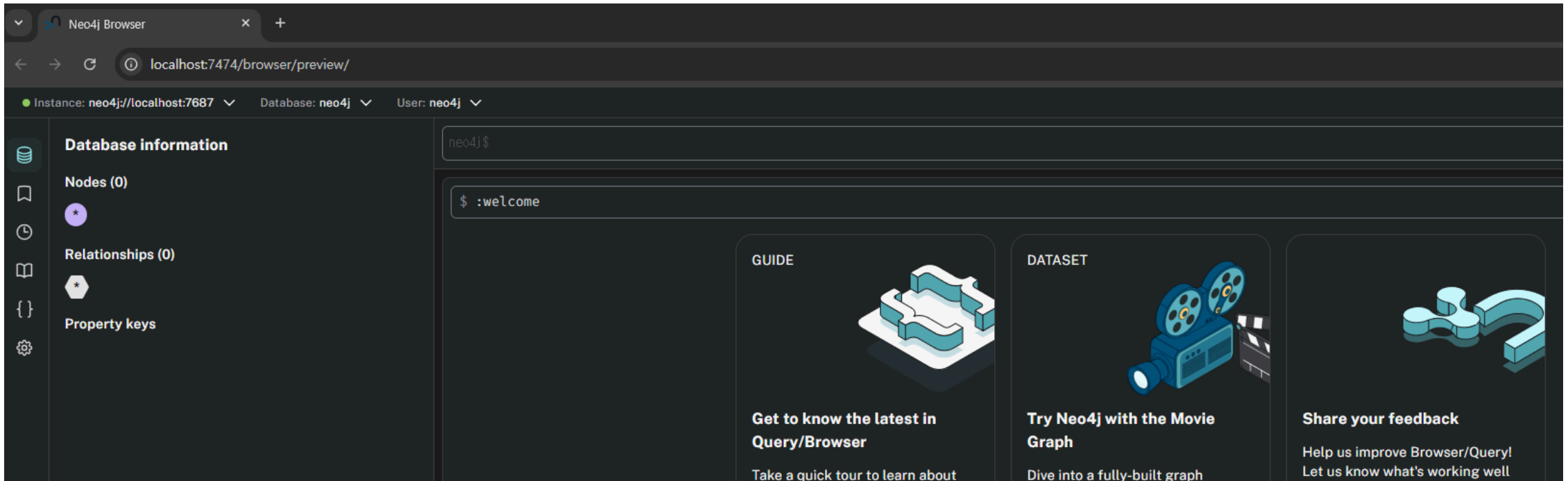
Database user

neo4j

Password

.....

Connect



Cypher 쿼리 예제

노드 생성

```
// Person 레이블을 가진 노드를 생성하고, name과 job 속성을 설정  
CREATE (:Person {name: '홍길동', job: 'AI강사'})
```



Instance: neo4j://localhost:7687 Database: neo4j User: neo4j

Go back to old Browser

Database information

Nodes (1)

* Person

Relationships (0)

*

Property keys

job

name

neo4j\$

neo4j\$ MATCH (n:Person) RETURN n LIMIT 25;

Graph

Table

RAW

Download

Copy

Results overview

Nodes (1)

* (1) Person (1)

honggil Dong

관계 생성

```
// 신사임당 노드 생성  
CREATE (:Person {name: '신사임당', job: '웹개발강사'})
```

```
// 홍길동과 신사임당 노드를 각각 찾아서 변수 a, b에 할당  
MATCH (a:Person {name: '홍길동'}), (b:Person {name: '신사임당'})  
// 홍길동에서 신사임당으로 향하는 MARRIED_TO 관계 생성  
CREATE (a)-[:MARRIED_TO]->(b)
```

The image shows the Neo4j Browser interface. On the left, the 'Database information' sidebar displays 'Nodes (2)' with a 'Person' node, 'Relationships (1)' with a 'MARRIED_TO' relationship, and 'Property keys' for 'job' and 'name'. The main panel shows two queries executed in the 'neo4j' database. The first query, 'MATCH (a:Person {name: '홍길동'}), (b:Person {name: '신사임당'}) CREATE (a)-[:MARRIED_TO]->(b)', successfully created 1 relationship and includes a warning: 'This query builds a cartesian product between disconnected patterns.' The second query, 'CREATE (:Person {name: '신사임당', job: '웹개발강사'})', successfully created 1 node, set 2 properties, and added 1 label.

Instance: neo4j://localhost:7687 Database: neo4j User: neo4j [Go back to old Browser](#)

Database information

Nodes (2)

- Person

Relationships (1)

- MARRIED_TO

Property keys

- job
- name

neo4j\$

neo4j\$ MATCH (a:Person {name: '홍길동'}), (b:Person {name: '신사임당'}) CREATE (a)-[:MARRIED_TO]->(b)

Created 1 relationship

Completed after 73 ms

> ⓘ This query builds a cartesian product between disconnected patterns.

neo4j\$ CREATE (:Person {name: '신사임당', job: '웹개발강사'})

Created 1 node, set 2 properties, added 1 label

Completed after 14 ms

Database information

Nodes (2)

- Person

Relationships (1)

- MARRIED_TO

Property keys

- job
- name

neo4j\$

neo4j\$ MATCH p=()-[:MARRIED_TO]->() RETURN p LIMIT 25;

Graph

Table

RAW

```
graph BT; A((신사임당)) -- MARRIED_TO --> B((홍길동));
```

Results overview

Nodes (2)

- Person (2)

Relationships (1)

- MARRIED_TO (1)

관계 기반 조회

```
// MARRIED_TO 관계로 연결된 Person 노드들을 찾아서  
MATCH (a:Person)-[:MARRIED_TO]->(b:Person)  
// 각 노드의 name 속성을 반환  
RETURN a.name, b.name
```

The screenshot shows the Neo4j Browser interface. At the top, it displays the instance details: neo4j://localhost:7687, Database: neo4j, and User: neo4j. On the left sidebar, under 'Database information', there are sections for 'Nodes (2)' showing a 'Person' node, 'Relationships (1)' showing a 'MARRIED_TO' relationship, and 'Property keys' listing 'job' and 'name'. The main area shows the Cypher query: `neo4j$ MATCH (a:Person)-[:MARRIED_TO]->(b:Person) RETURN a.name, b.name`. Below the query, there are tabs for 'Table' and 'RAW', with 'Table' selected. The results are displayed in a table with two columns: 'a.name' and 'b.name'. The first row shows the values '"홍길동"' and '"신사임당"'. At the bottom right, a status message reads: 'Started streaming 1 record after 49 ms and completed after 52 ms.'

a.name	b.name
"홍길동"	"신사임당"

조건 검색

```
// 모든 Person 노드를 찾아서  
MATCH (p:Person)  
// name 속성이 '신사임당'인 노드만 필터링  
WHERE p.name = '신사임당'  
// 해당 노드 전체 반환  
RETURN p
```

The image shows the Neo4j Desktop interface. On the left, the 'Database information' sidebar displays the database structure: 2 Nodes (Person), 1 Relationship (MARRIED_TO), and property keys (job, name). The main workspace shows a Cypher query: `neo4j$ MATCH (p:Person) WHERE p.name = '신사임당' RETURN p`. The query is executed, and the results are displayed in the 'Graph' view as a single green circular node labeled '신사임당'. The 'Results overview' panel on the right shows 'Nodes (1)' with a count of 1 for the 'Person' type.

연결된 경로 검색

```
// 1단계부터 3단계까지 연결된 Person 노드들의 경로를 찾아서
MATCH path = (p1:Person)-[*1..3]-(p2:Person)
// 시작점이 '홍길동'인 경로만 필터링
WHERE p1.name = '홍길동'
// 전체 경로 반환
RETURN path
```

The image shows the Neo4j web interface. On the left, the 'Database information' sidebar displays the schema: 2 nodes of type 'Person' and 1 relationship of type 'MARRIED_TO'. The main area shows a Cypher query: `neo4j$ MATCH path = (p1:Person)-[*1..3]-(p2:Person) WHERE p1.name = '홍길동' RETURN path`. The query is executed, and the results are shown in 'Graph' view. The graph displays a path from '홍길동' (Hong Gildong) to '신사임당' (Shinsaimdang) via the 'MARRIED_TO' relationship. On the right, the 'Results overview' panel shows the query results: 2 nodes of type 'Person' and 1 relationship of type 'MARRIED_TO'.

Database information

Nodes (2)

- * Person

Relationships (1)

- * MARRIED_TO

Property keys

- job
- name

neo4j\$

neo4j\$ MATCH path = (p1:Person)-[*1..3]-(p2:Person) WHERE p1.name = '홍길동' RETURN path

Graph Table RAW

Results overview

Nodes (2)

- * (2) Person (2)

Relationships (1)

- * (1) MARRIED_TO (1)

데이터 삭제

```
// name이 '홍길동'인 Person 노드를 찾아서  
MATCH (p:Person {name: '홍길동'})  
// 관계까지 포함하여 완전히 삭제 (DETACH DELETE)  
DETACH DELETE p
```

