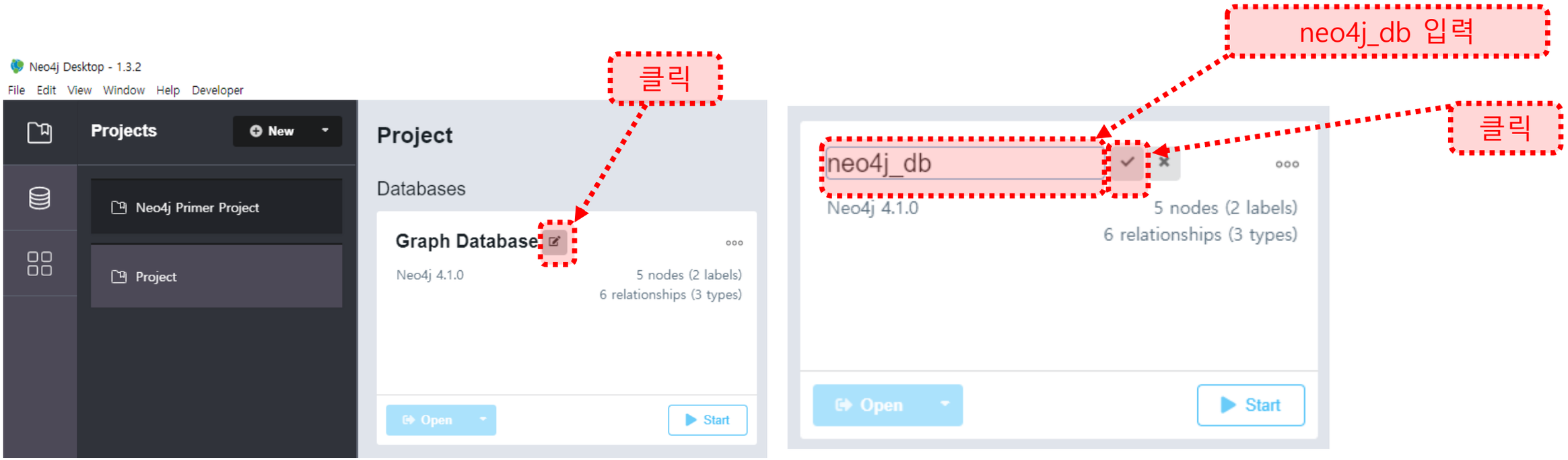
- neo4j Graph Database명 변경

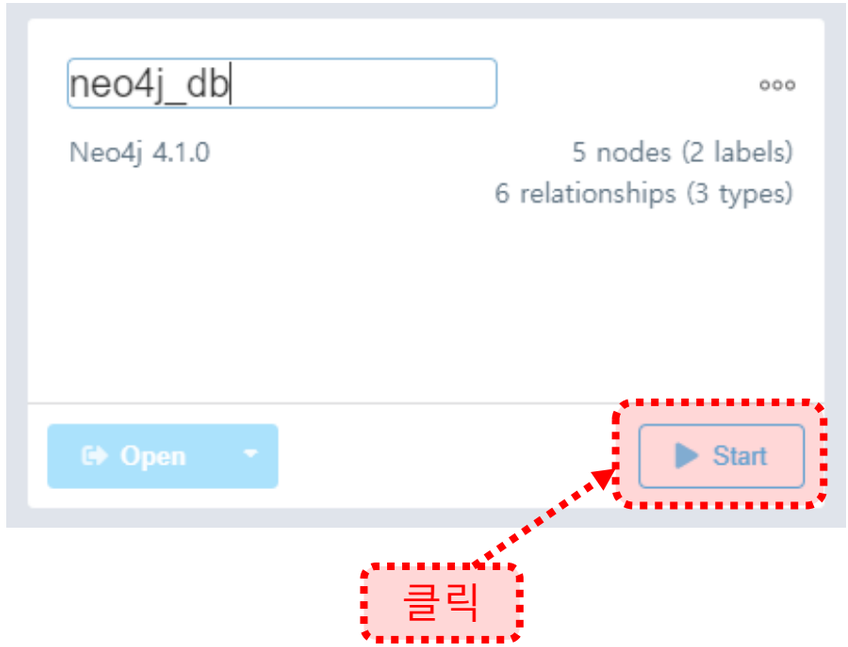
① neo4j Graph DB 실행

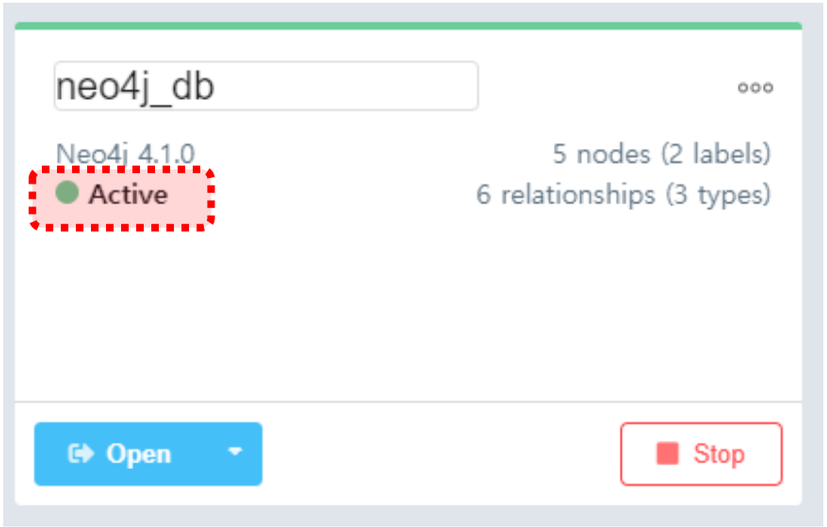
② Database명 변경



- neo4j Graph Database 기동

  start 버튼을 클릭하여 neo4j Graph Database 기동



- Graph DB 접속 정보

  DB접속 URI, user_id, password 등의 정보 필요

  URI는 neo4j://localhost:7687 로 사용

```
from neo4j import GraphDatabase
```

```
uri = 'neo4j://localhost:7687'
user = 'neo4j'
password = '1234'
```

- Graph DB 접속 Driver 로딩

  DB접속 정보를 입력하여 드라이버 로딩

```
driver = GraphDatabase.driver(uri, auth=(user, password))
```

- Graph DB 쿼리 및 결과 처리 함수 쿼리  함수 정의

  Graph DB 접속 후 쿼리 수행 및 수행 결과 처리를 위한 함수 정의

```
def get_students(tx):
    query = 'MATCH (n:STUDENT) RETURN n.성명 AS 성명, n.학년 AS 학년, n.학과 AS 학과'
    result = tx.run(query)
    for record in result:
        print(record['성명'], record['학년'], record['학과'])
```

- Graph DB 접속, 트랜잭션 처리 함수 정의 및 실행

```
def run_query(func):
    with driver.session() as session:
        session.read_transaction(func)
```

```
run_query(get_students)
```

```
이몽룡 4학년 컴퓨터공학과
성춘향 3학년 컴퓨터공학과
변사또 2학년 컴퓨터공학과
```

- 학년이 '3학년' 이상인 학생의 성명 출력

```
neo4j$ MATCH (a:STUDENT) WHERE a.학년 >= '3학년' RETURN a.성명 AS 성명
```

Table / Text / Code

| | 성명 |
|---|---|
| 1 | "이몽룡" |
| 2 | "성춘향" |

```python
def get_name(tx):
    query = 'MATCH (a:STUDENT) WHERE a.학년 >= $grade RETURN a.성명 AS 성명'
    result = tx.run(query, grade = '3학년')
    for record in result:
        print(record['성명'])

run_query(get_name)
```

```
이몽룡
성춘향
```

- 딥러닝을 수강하는 학생의 성명 출력

```
neo4j$ MATCH (a:STUDENT)-[:수강하다]→(b:SUBJECT{과목명:'딥러닝'}) RETURN a.성명 AS 성명
```

Table

| | 성명 |
|---|---|
| 1 | "변사또" |

Text

```
def get_name2(tx):
    query = """
                MATCH (a:STUDENT)-[:수강하다]->(b:SUBJECT{과목명:$subject})
                RETURN a.성명 AS 성명
            """
    result = tx.run(query, subject = '딥러닝')
    for record in result:
        print(record['성명'])

run_query(get_name2)
```

변사또

- 성춘향과 교제하는 학생의 수강하는 과목의 과목명 출력

```
neo4j$ MATCH (a:STUDENT)-[:교제하다]→(b:STUDENT{성명:'성춘향'}) MATCH (a) - [:수강하다] → (d:SUBJECT) RETURN d.과목명 AS 과목명
```

Table

| | 과목명 |
|---|---|
| 1 | "인공지능기초" |

Text

```
def get_subject(tx):
    query = """
                MATCH (a:STUDENT)-[:교제하다]->(b:STUDENT{성명:$name})
                MATCH (a) - [:수강하다] -> (d:SUBJECT)
                RETURN d.과목명 AS 과목명
            """
    result = tx.run(query, name = '성춘향')
    for record in result:
        print(record['과목명'])

run_query(get_subject)
```

인공지능기초

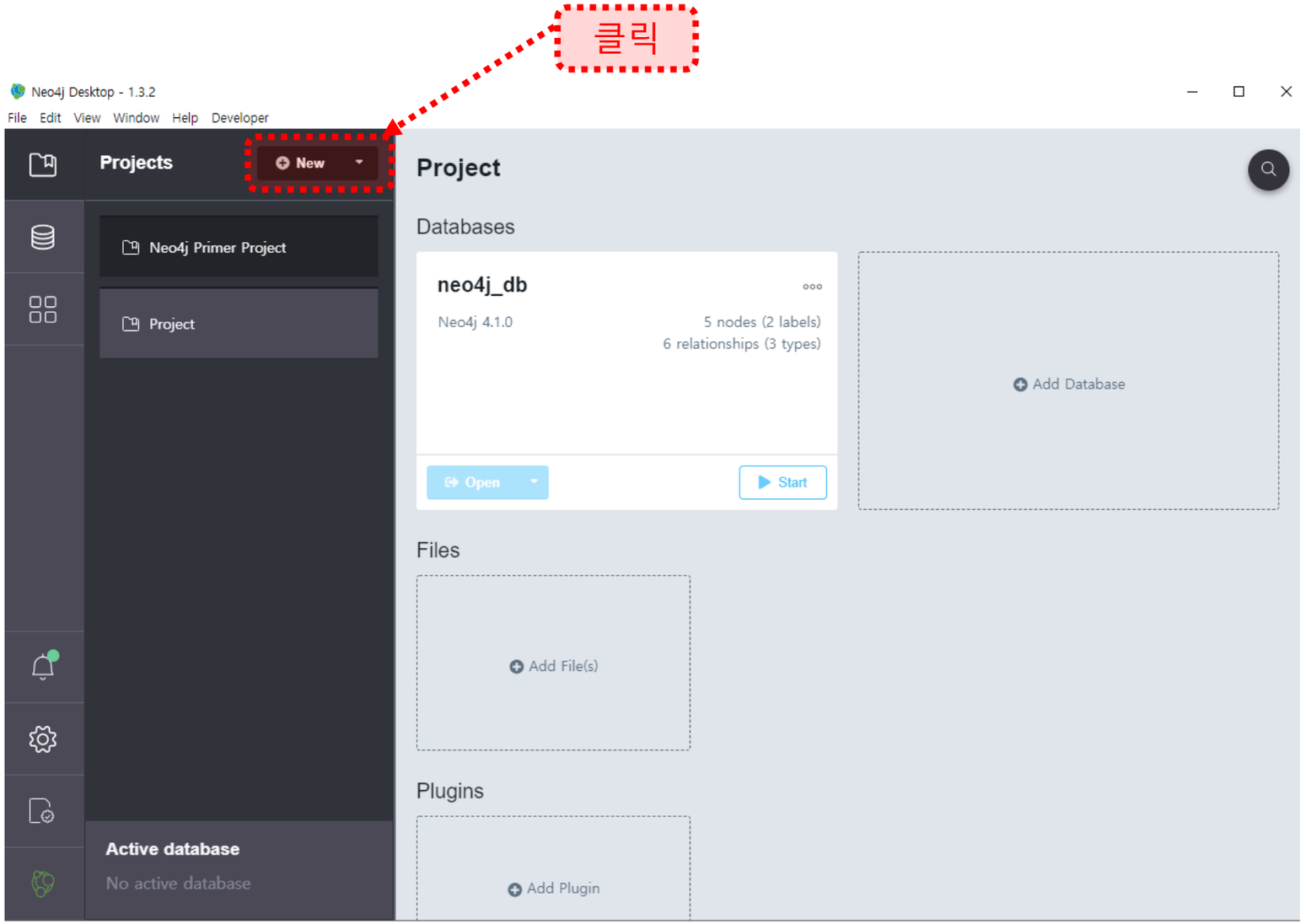
- Graph DB의 리턴 type인 Node 클래스로 부터 정보 출력

```
def get_all_subject(tx):
    query = """
                MATCH (a:SUBJECT) RETURN a
            """
    result = tx.run(query)
    for record in result:
        for node in record:
            print(node)
            for key, val in node.items():
                print(key, ':', val)
run_query(get_all_subject)
```

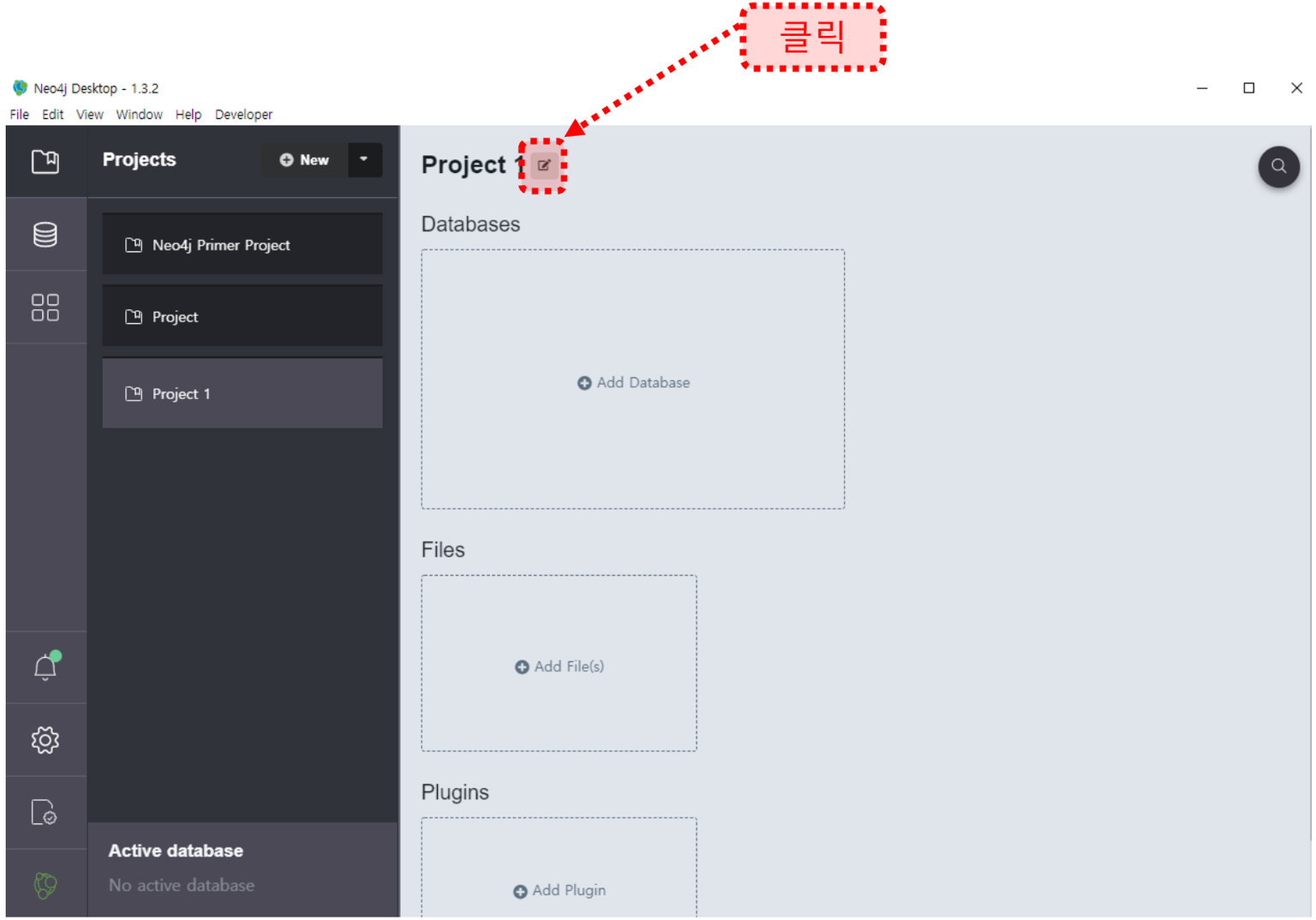
```
<Node id=3 labels=frozenset({'SUBJECT'}) properties={'과목명': '인공지능기초', '담당교수': '김교수', '학점': 2}>
과목명 : 인공지능기초
담당교수 : 김교수
학점 : 2
<Node id=4 labels=frozenset({'SUBJECT'}) properties={'과목명': '딥러닝', '담당교수': '박교수', '학점': 3}>
과목명 : 딥러닝
담당교수 : 박교수
학점 : 3
```

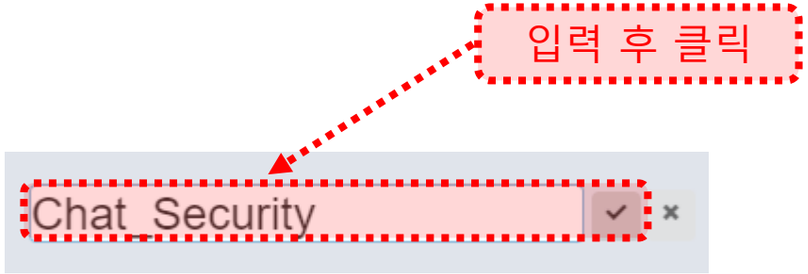
# Neo4j.ipynb 실습

- neo4j 프로젝트 생성

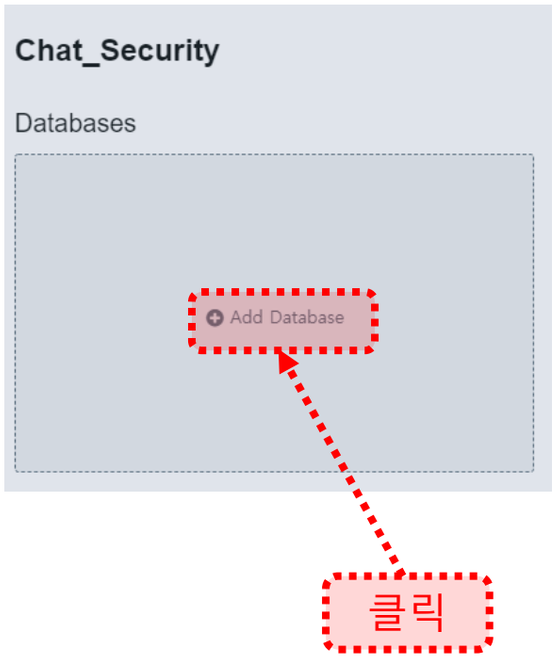


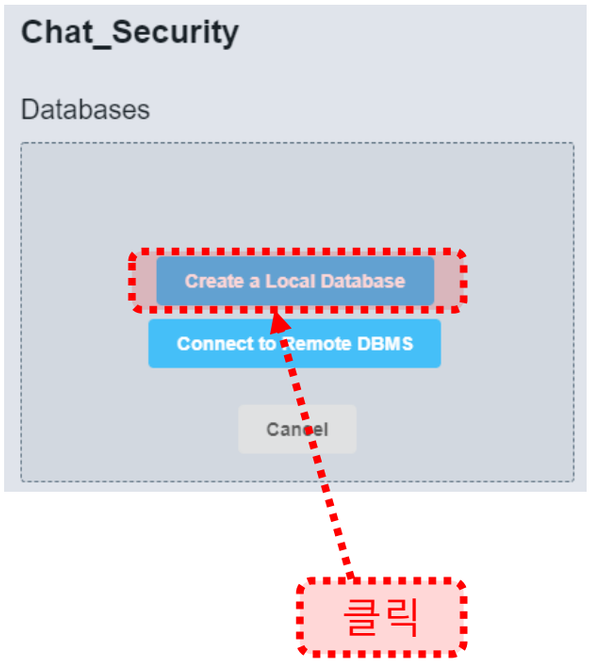
- neo4j 프로젝트명 변경

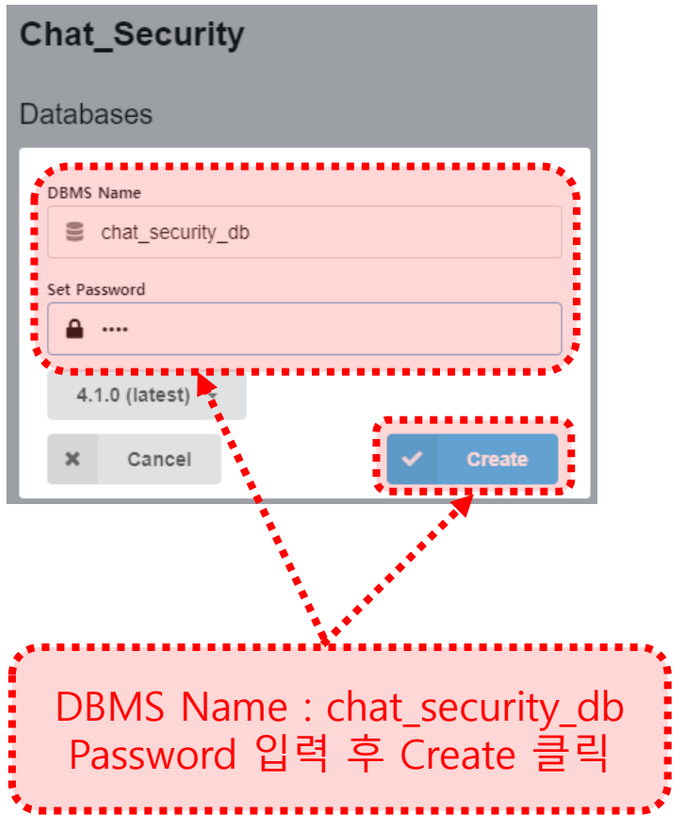


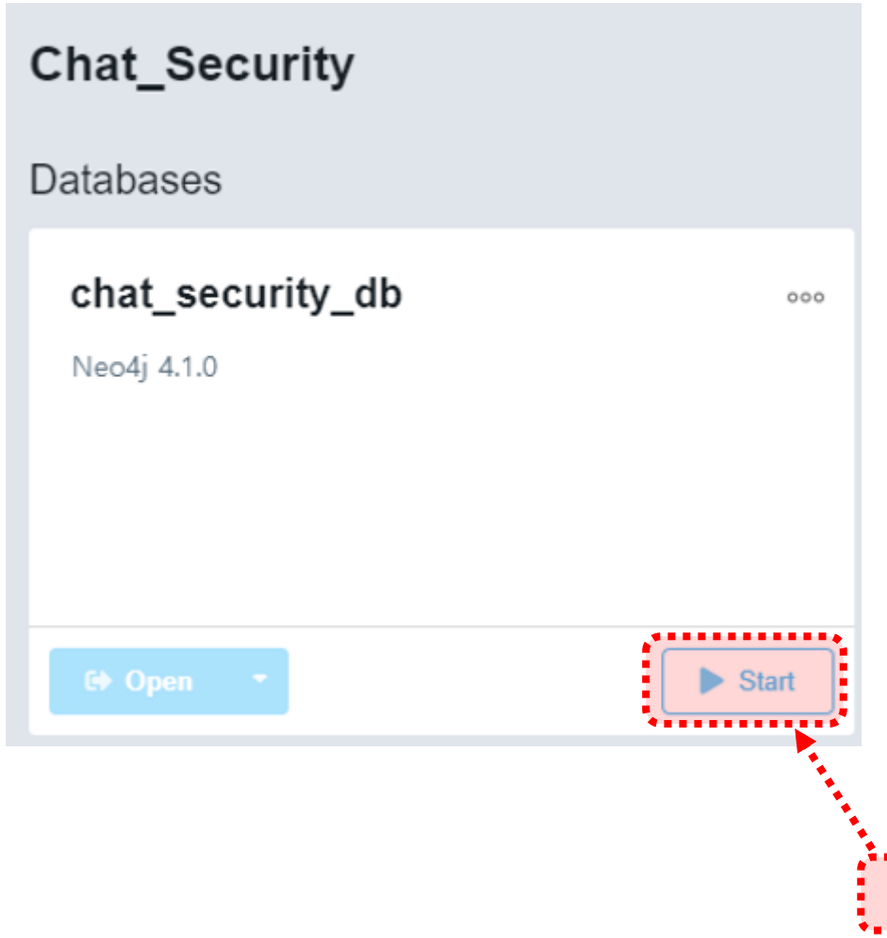
- neo4j Database 생성

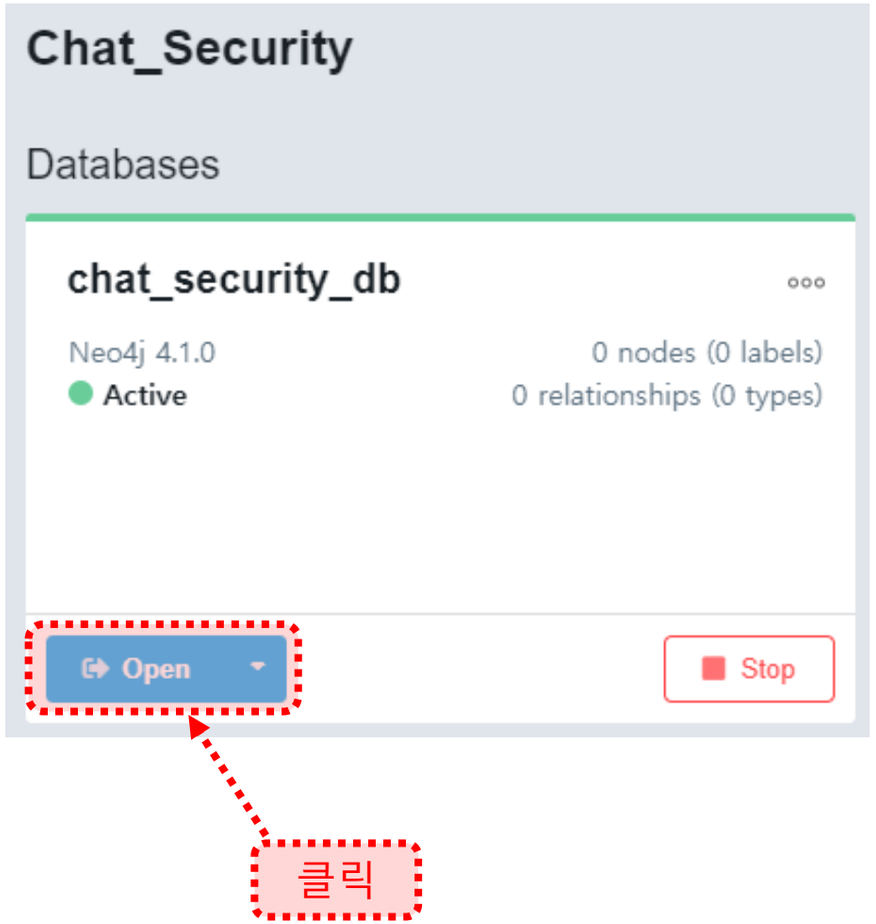


- neo4j Database start 및 Neo4j 브라우저 기동



## ▪ 암호화 방식 저장

CREATE (a:암호화 { 명칭 : '대칭키암호',
설명 : '암호화에 사용되는 키(암호화 키)와 복호화에 사용되는 키(복호화 키)가 서로 동일한 암호화 알고리즘' });

CREATE (a:암호화 { 명칭 : '해시함수',
설명 : '임의의 길이의 문자열을 고정된 길이의 이진 문자열로 매핑하여 주는 함수로 해시 함수는 데이터의 무결성, 인증, 부인 방지 등에서 사용' });

CREATE (a:암호화 { 명칭 : '비대칭키암호', 설명 : '암호화에 쓰이는 키와 복호화에 쓰이는 키가 상이한 암호화 알고리즘,
공개키(Public Key)와 개인키(Private Key)의 두 쌍으로 구성되어 있으며, 이들은 서로 역관계의 쌍(Pair)으로 동작하는 메커니즘' });

CREATE (a:암호화 { 명칭 : '비밀키암호' });

CREATE (a:암호화 { 명칭 : '해시알고리즘' });

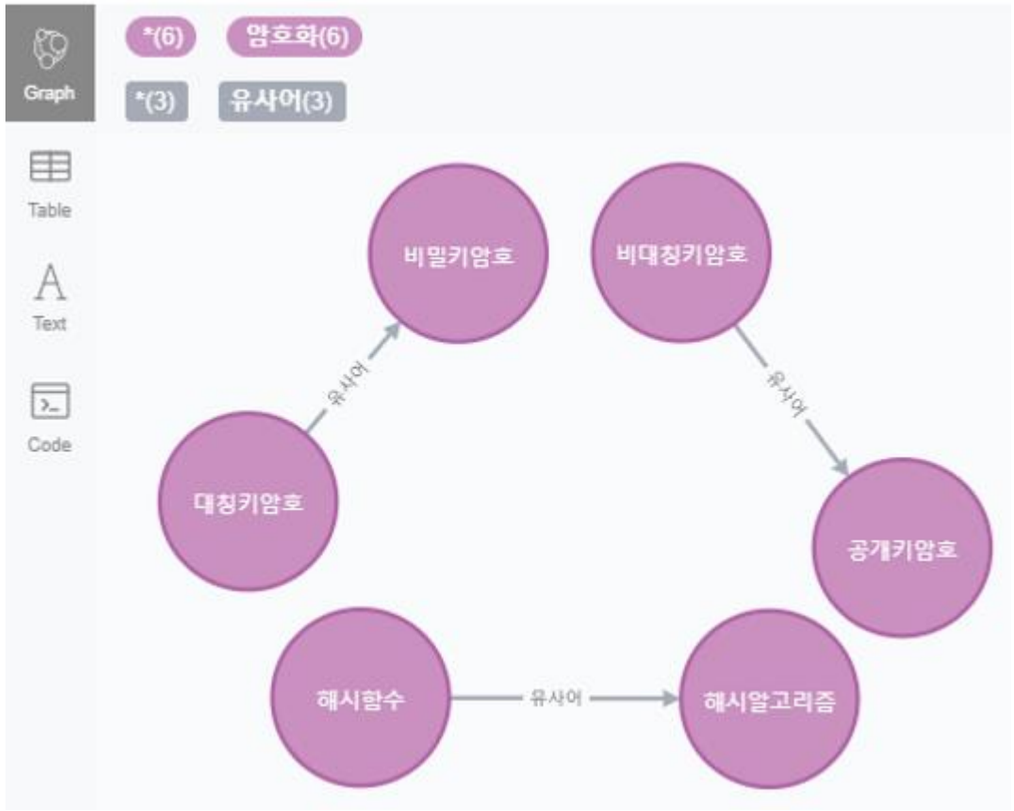
CREATE (a:암호화 { 명칭 : '공개키암호' });

## ▪ 암호화 방식 유사어 관계 저장

MATCH (a{명칭:'대칭키암호'}) MATCH (b{명칭:'비밀키암호'}) CREATE (a) - [r:유사어] -> (b);

MATCH (a{명칭:'해시함수'}) MATCH (b{명칭:'해시알고리즘'}) CREATE (a) - [r:유사어] -> (b);

MATCH (a{명칭:'비대칭키암호'}) MATCH (b{명칭:'공개키암호'}) CREATE (a) - [r:유사어] -> (b);

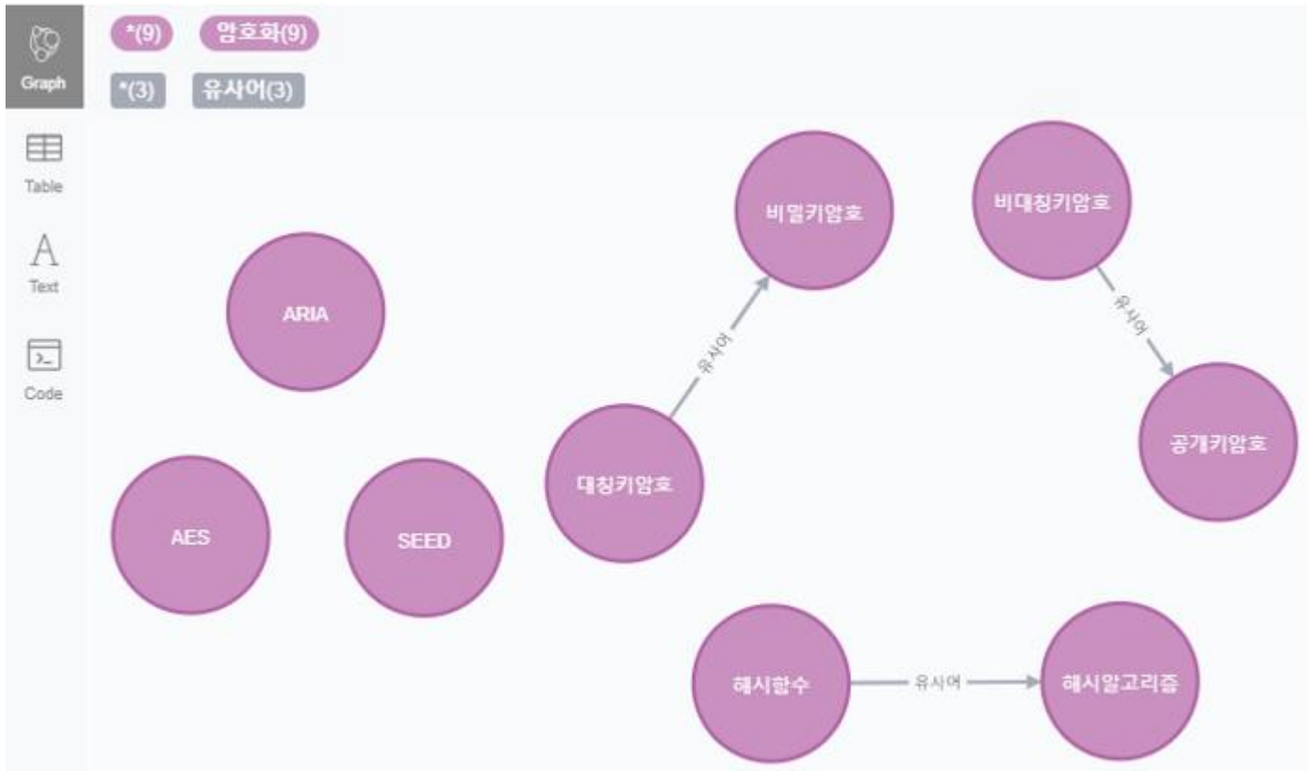


## ▪ 대칭키 암호 알고리즘 저장

CREATE (a:암호화 { 명칭 : 'SEED', 키길이 : '128, 256',
설명 : '1999년 2월 한국정보보호진흥원(한국인터넷진흥원의 전신)의 기술진이 개발한 대칭 키 블록 암호 알고리즘' });

CREATE (a:암호화 { 명칭 : 'ARIA', 키길이 : '128, 192, 256',
설명 : '국내 Academy(학계), Research Institute(연구소), Agency(정부 기관) 공동으로 개발한 대칭 키 블록 암호 알고리즘' });

CREATE (a:암호화 { 명칭 : 'AES', 키길이 :  '128, 192, 256',
설명 : '2001년 미국 표준 기술 연구소(NIST)에 의해 지정된 대칭 키 블록 암호 알고리즘' });

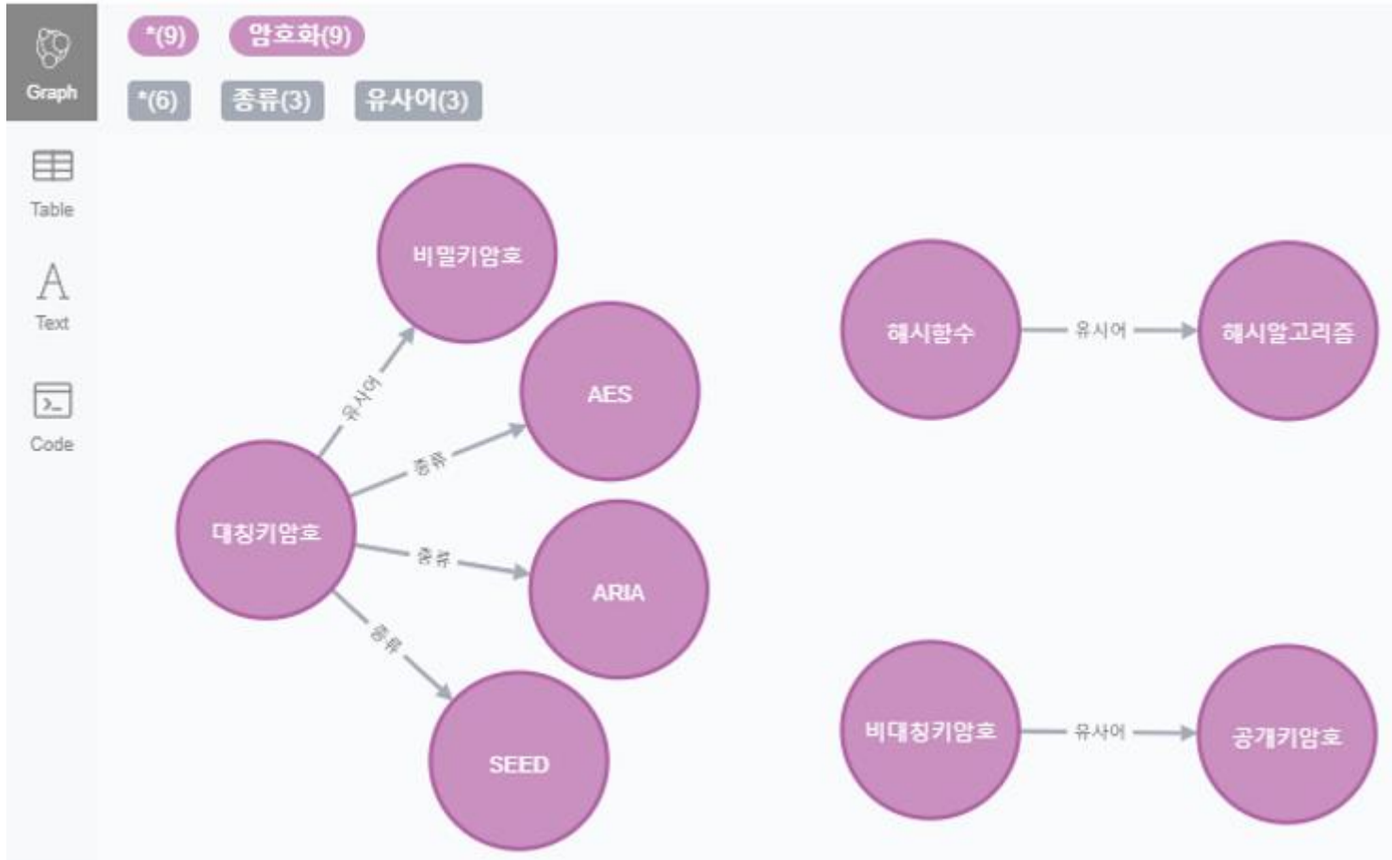


- 대칭키 암호방식과 암호 알고리즘 관계 저장

MATCH (a{명칭:'대칭키암호'}) MATCH (b{명칭:'SEED'}) CREATE (a) - [r:종류] -> (b);

MATCH (a{명칭:'대칭키암호'}) MATCH (b{명칭:'ARIA'}) CREATE (a) - [r:종류] -> (b);

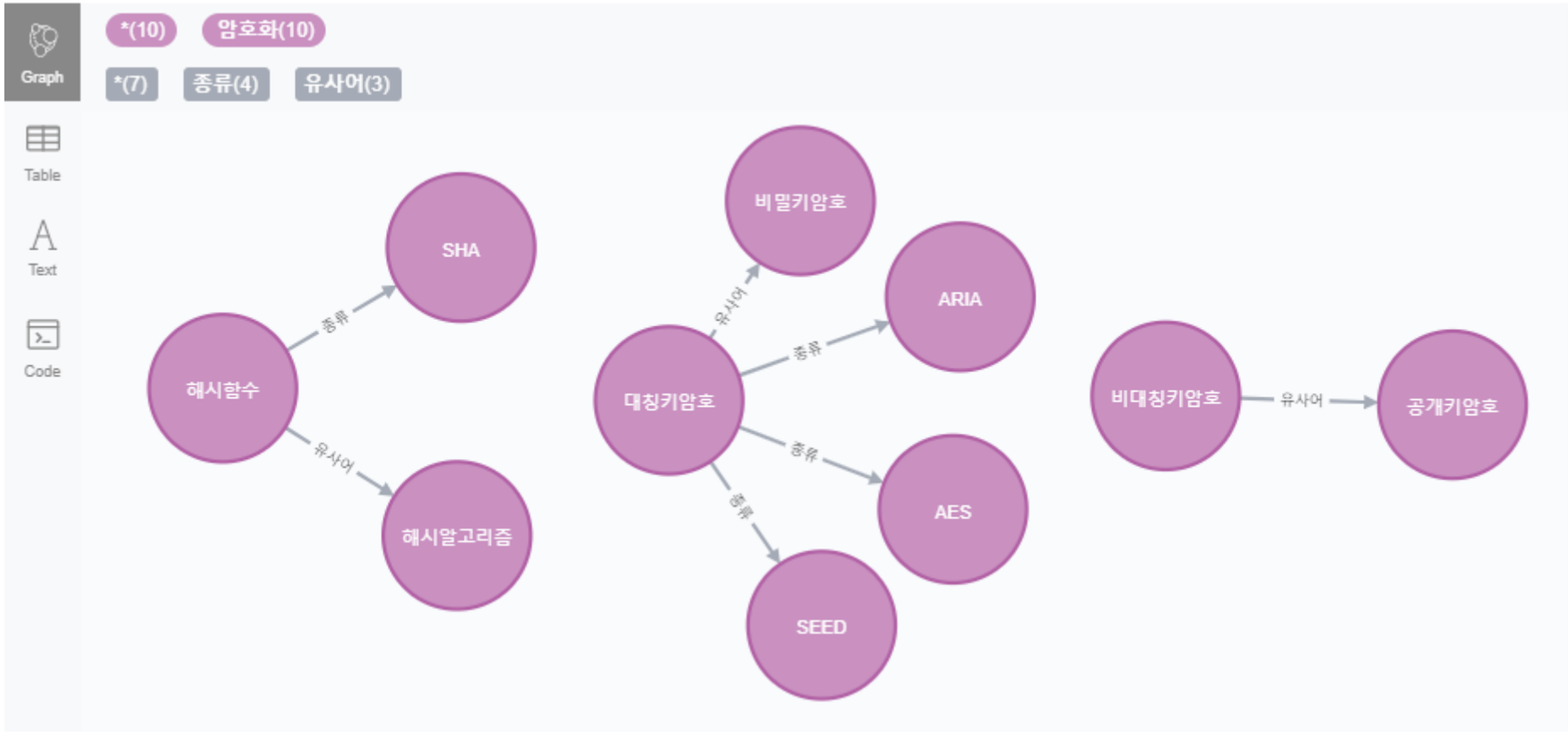
MATCH (a{명칭:'해시암호'}) MATCH (b{명칭:'SHA'}) CREATE (a) - [r:종류] -> (b);



- 해쉬 함수 종류 저장 및 해쉬함수와 관계 저장

CREATE (a:암호화 { 명칭 : 'SHA', 출력길이 : '224, 256, 384, 512',
설명 : '암호학적 해시 함수들로 미국 국가안보국(NSA)이 1993년에 처음으로 설계했으며 미국 국가 표준으로 지정' });

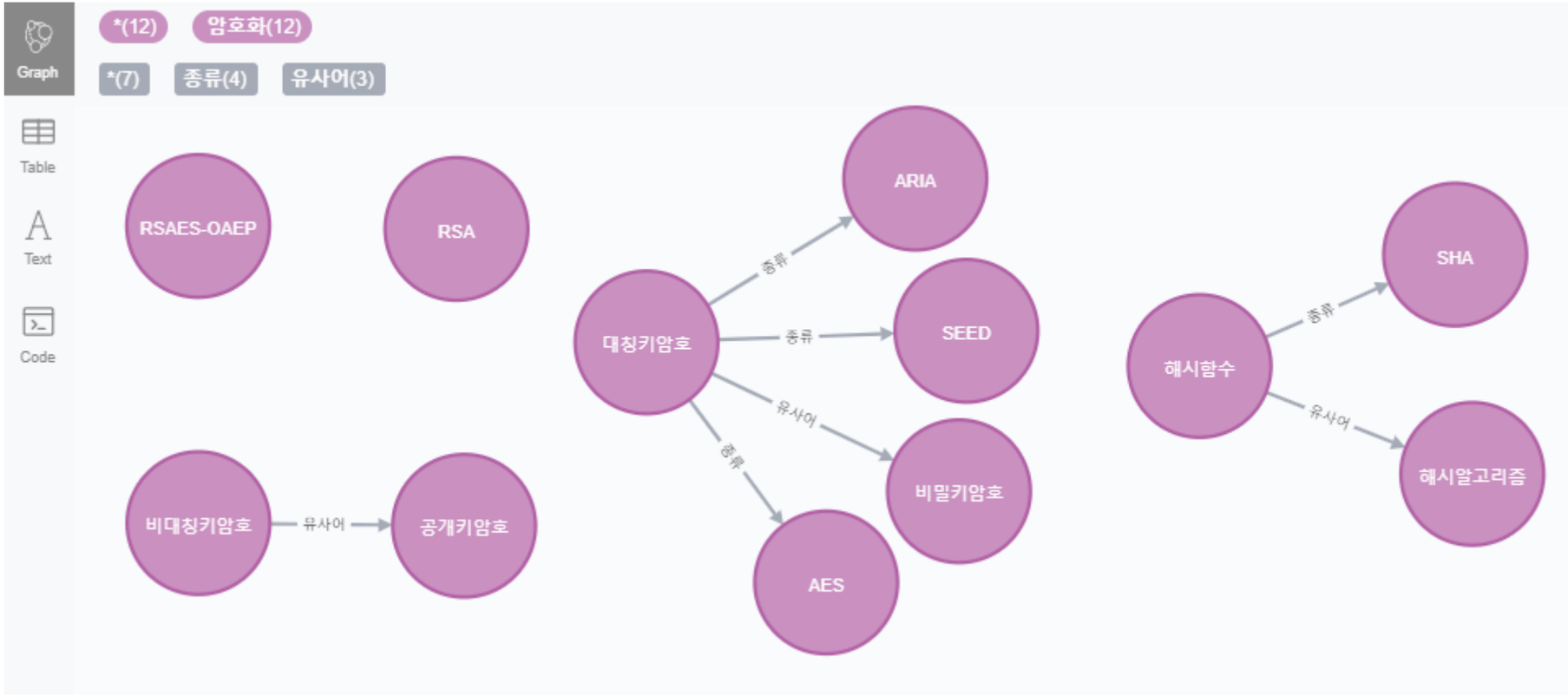
MATCH (a{명칭:'해시함수'}) MATCH (b{명칭:'SHA'}) CREATE (a) - [r:종류] -> (b);



## ▪ 비대칭키 암호 알고리즘 정보 저장

CREATE (a:암호화 { 명칭 : 'RSA', 키길이 : '2048, 3072',
설명 : '매우 큰 수의 소인수분해가 어렵다는 수학적 원리를 기반으로 만들어진 공개키 암호 알고리즘' });

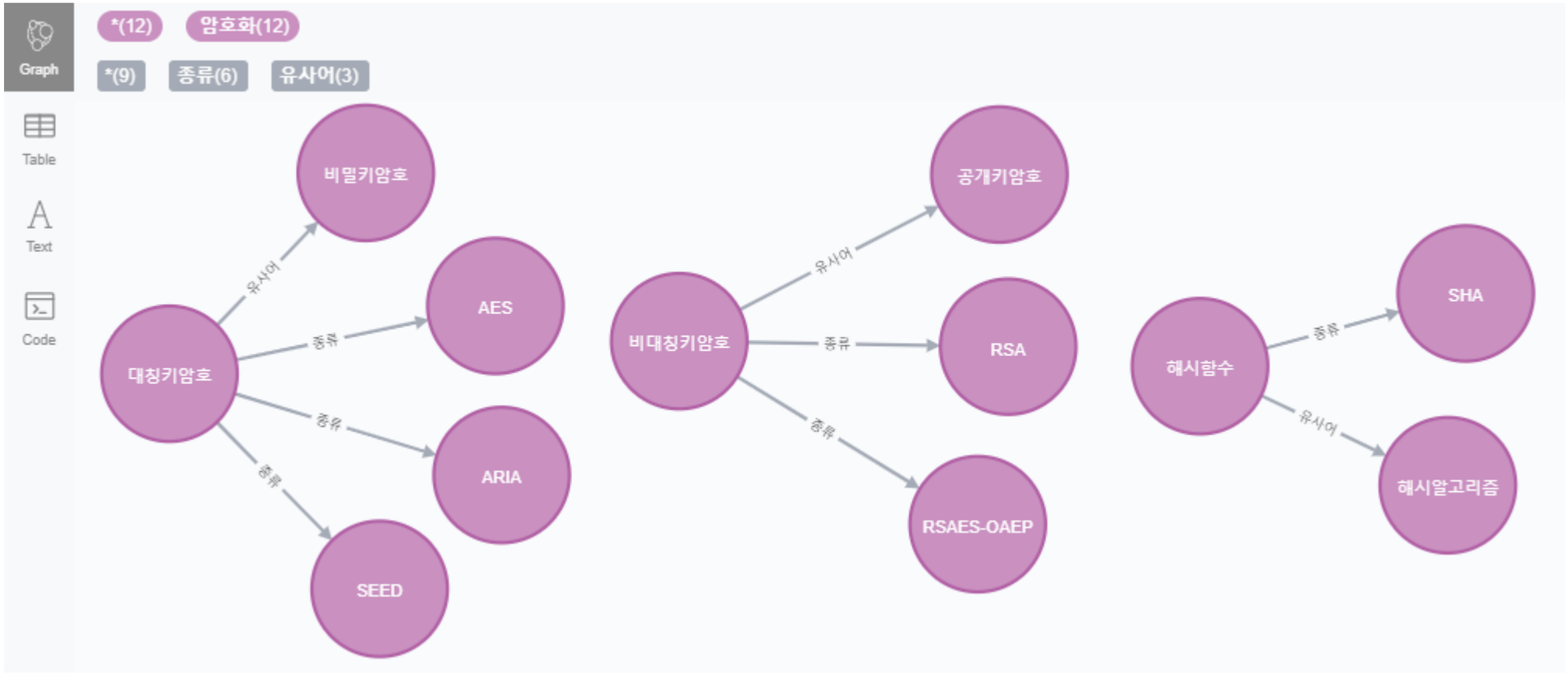
CREATE (a:암호화 { 명칭 : 'RSAES-OAEP', 키길이 : '128, 192, 256',
설명 : 'RSA를 개량한 알고리즘으로 난수를 이용해서 암호문이 매번 다른 패턴이 되도록 하여 보다 안전성 제고' });



- 비대칭키 암호 방식과 암호 알고리즘 관계 정보 저장

MATCH (a{명칭:'비대칭키암호'}) MATCH (b{명칭:'RSA'}) CREATE (a) - [r:종류] -> (b);

MATCH (a{명칭:'비대칭키암호'}) MATCH (b{명칭:'RSAES-OAEP'}) CREATE (a) - [r:종류] -> (b);



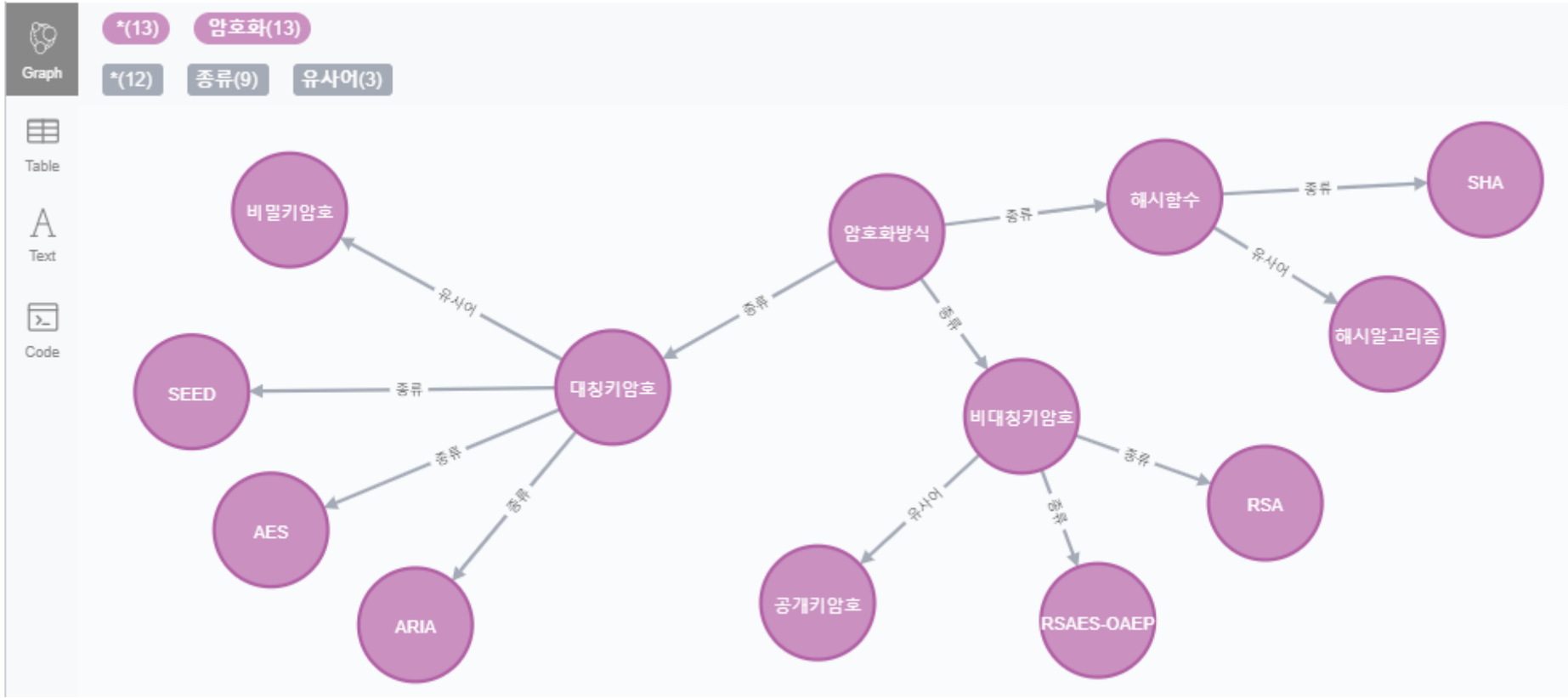
## ▪ 암호화방식 노드 생성 및 암호화방식 관계 정보 저장

CREATE (a:암호화 { 명칭 : '암호화방식' });

MATCH (a{명칭:'암호화방식'}) MATCH (b{명칭:'대칭키암호'}) CREATE (a) - [r:종류] -> (b);

MATCH (a{명칭:'암호화방식'}) MATCH (b{명칭:'해시함수'}) CREATE (a) - [r:종류] -> (b);

MATCH (a{명칭:'암호화방식'}) MATCH (b{명칭:'비대칭키암호'}) CREATE (a) - [r:종류] -> (b);



# 정보보안 지식베이스 저장 실습

- 지식베이스에서 암호화 방식의 종류 검색
  - 질의 : 암호화방식의 종류는?

```
def get_cypher_method(tx):
    lst = []
    query = """
                MATCH (a{명칭:'암호화방식'}) - [r:종류] -> (b)
                RETURN b.명칭 AS 암호화방식종류
            """
    result = tx.run(query)

    for record in result:
        lst.append(record['암호화방식종류'])

    print('암호화방식의 종류에는', ', '.join(lst), '등이 있습니다.')

run_query(get_cypher_method)
```

```
암호화방식의 종류에는 비대칭키암호, 해시함수, 대칭키암호 등이 있습니다.
```

- 대칭키 암호 알고리즘 종류 및 키길이 검색
  - 질의 : 대칭키 암호 알고리즘의 종류와 키길이는?

```
def get_symetric_key(tx):
    lst = []
    query = """
        MATCH (a{명칭:'대칭키암호'}) - [r:종류] -> (b)
        RETURN b.명칭 AS 명칭, b.키길이 AS 키길이
        """
    result = tx.run(query)
    for record in result:
        lst.append('{}(키길이:{})'.format(record['명칭'], record['키길이']))
    print('대칭키 암호 알고리즘은', ''.join(lst),'등이 있습니다.')
run_query(get_symetric_key)
```

```
대칭키 암호 알고리즘은 AES(키길이:128, 192, 256)ARIA(키길이:128, 192, 256)SEED(키길이:128, 256) 등이 있습니다.
```

- 대칭키 암호 알고리즘 개념 및 종류 검색
  - 질의 : 비밀키 암호 알고리즘은 무엇이며,
    비밀키 암호 알고리즘의 종류에는 어떤 것들이 있는가?

```
def get_private_key(tx):
    query = """
              MATCH (a) - [r1:유사어] -> (b{명칭:'비밀키암호'})
              RETURN a.설명 AS 설명
            """
    result = tx.run(query)
    for record in result:
        print('비밀키 암호 알고리즘은', record['설명'], '입니다.')

    lst = []
    query = """
              MATCH (a) - [r1:유사어] -> (b{명칭:'비밀키암호'})
              MATCH (a) - [r2:종류] -> (c)
              RETURN c.명칭 AS 명칭
            """
    result = tx.run(query)
    for record in result:
        lst.append(record['명칭'])
    print('비밀키 알고리즘의 종류에는', ', '.join(lst), '등이 있습니다.')

run_query(get_private_key)
```

```
비밀키 암호 알고리즘은 암호화에 사용되는 키(암호화 키)와 복호화에 사용되는 키(복호화 키)가 서로 동일한 암호화 알고리즘 입니다.
비밀키 알고리즘의 종류에는 AES, ARIA, SEED 등이 있습니다.
```

- 해시함수의 종류
  - 질의 : 해시함수의 종류에는 어떤 것들이 있는가?

```
def get_Hash(tx):
    lst = []
    query = """
                MATCH (a{명칭:'해시함수'}) - [r:종류] -> (b)
                RETURN b.명칭 AS 명칭
                ORDER BY 명칭
            """
    result = tx.run(query)
    for record in result:
        lst.append(record['명칭'])
    print('해시함수 종류에는', ', '.join(lst), '등이 있습니다.')

run_query(get_Hash)
```

```
해시함수 종류에는 SHA 등이 있습니다.
```

- 해시함수의 종류 추가
  - 질의 : 해시함수의 종류 중 SHA3 추가

    CREATE 쿼리는 session.write_transaction( ) 함수 사용

```
def run_exec_query(func):
    with driver.session() as session:
        session.write_transaction(func)
```

```
def set_Hash(tx):

    get_Hash(tx)

    query = """
              CREATE (a:암호화 { 명칭 : 'SHA3', 출력길이 : '224, 256, 384, 512', \
              설명 : 'SHA-1과 SHA-2를 대체하기 위해 미국 국립표준기술연구소(NIST)가 2015년 8월 5일에 발표한 스펀지 구조의 암호화 해시함수' });
            """
    result = tx.run(query)

    query = """
              MATCH (a{명칭:'해시함수'})
              MATCH (b{명칭:'SHA3'})
              CREATE (a) - [r:종류] -> (b);
            """
    result = tx.run(query)

    get_Hash(tx)

run_exec_query(set_Hash)
```

```
해시함수 종류에는 SHA 등이 있습니다.
해시함수 종류에는 SHA, SHA3 등이 있습니다.
```

# neo4j_knowledge_base.ipynb