BD07

Students: Peng simin 2023020207

Chen yinuo2023020101

Xu hongye2023020204

1.

(a) Neo4j's ACID Compliance: Neo4j is an ACID (Atomicity, Consistency, Isolation, Durability) compliant graph database, which means that during data transaction processing, it can ensure the following:

Atomicity: Transactions are either fully executed or not executed at all, and there is no possibility of partial completion.

Consistency: Data remains consistent before and after transactions. Any rules or constraints during transaction execution, such as uniqueness constraints, remain intact.

Isolation: Concurrent transactions are isolated from each other and do not interfere with each other, ensuring data integrity.

Durability: Once a transaction is committed, changes to the data will be persistent and can be recovered even if the system crashes.

(b) How to handle data relationships in relational databases, MongoDB, and Neo4j:

Relational database: Use foreign keys to establish relationships between tables, usually by querying related data through JOIN operations.

MongoDB is a NoSQL document database that typically uses nested documents or references to handle relationships.

Neo4j: Represent relationships through connections between nodes and edges, which are directly structured connections in the database, making queries more efficient.

(c) The difference between native and non-native storage and their impact on processing performance:

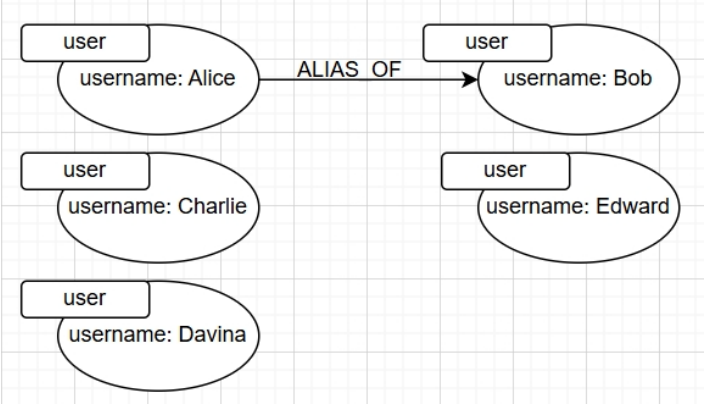
Native Storage: Neo4j uses a native graph structure to store data, which is designed specifically for graph data and therefore has high query performance, making it particularly suitable for complex graph relationships.

Non Native Storage: Non graph databases typically store data in key value pairs or document formats, requiring more computational and transformation steps for querying graph relationships.

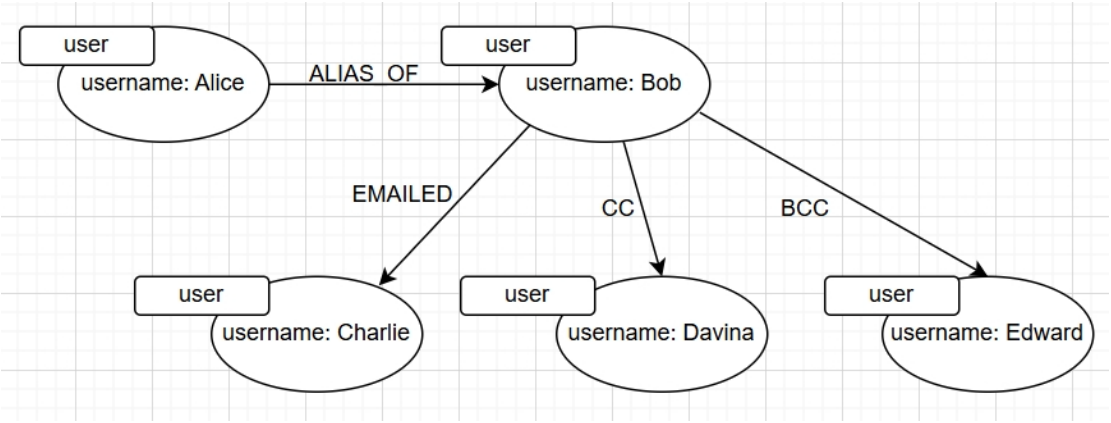
Performance impact: The structured query efficiency of native storage is higher, reducing the cost of data conversion, making it very suitable for data scenarios with a large number of relational connections.

2.

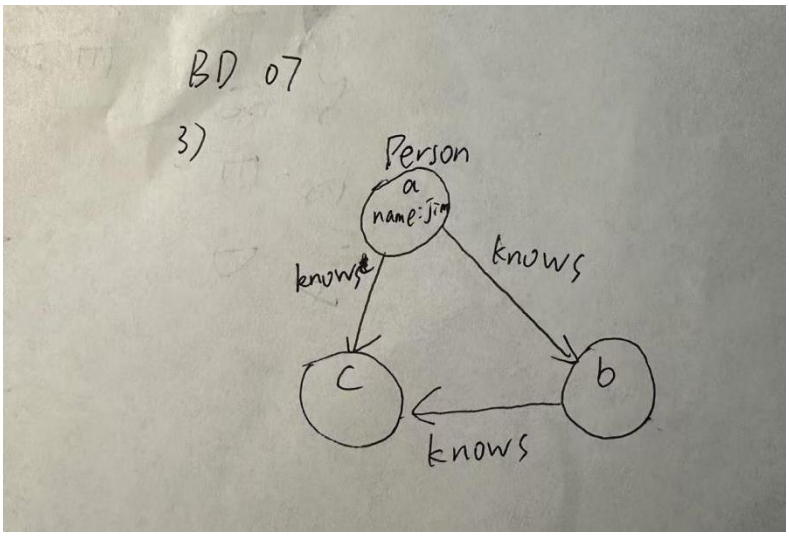
(a)The main purpose of this Cypher script is to create 5 nodes, all of which are labeled User, and each node has a username attribute, which is 'Alice', 'Bob', 'Charlie', 'Davina', and 'Edward'. Then, a relationship is created from node Alice to node Bob, and the relationship type is ALIAS_OF.



(b) Firstly, this script uses a MATCH clause to find 4 nodes that have a User label and a specific username value. Then, 3 relationships are created via the CREATE clause. A relationship of type EMAILED is created from the bob node to the charlie node. A relationship of type CC is created from the bob node to the davina node. A relationship of type BCC is created from the bob node to the edward node.



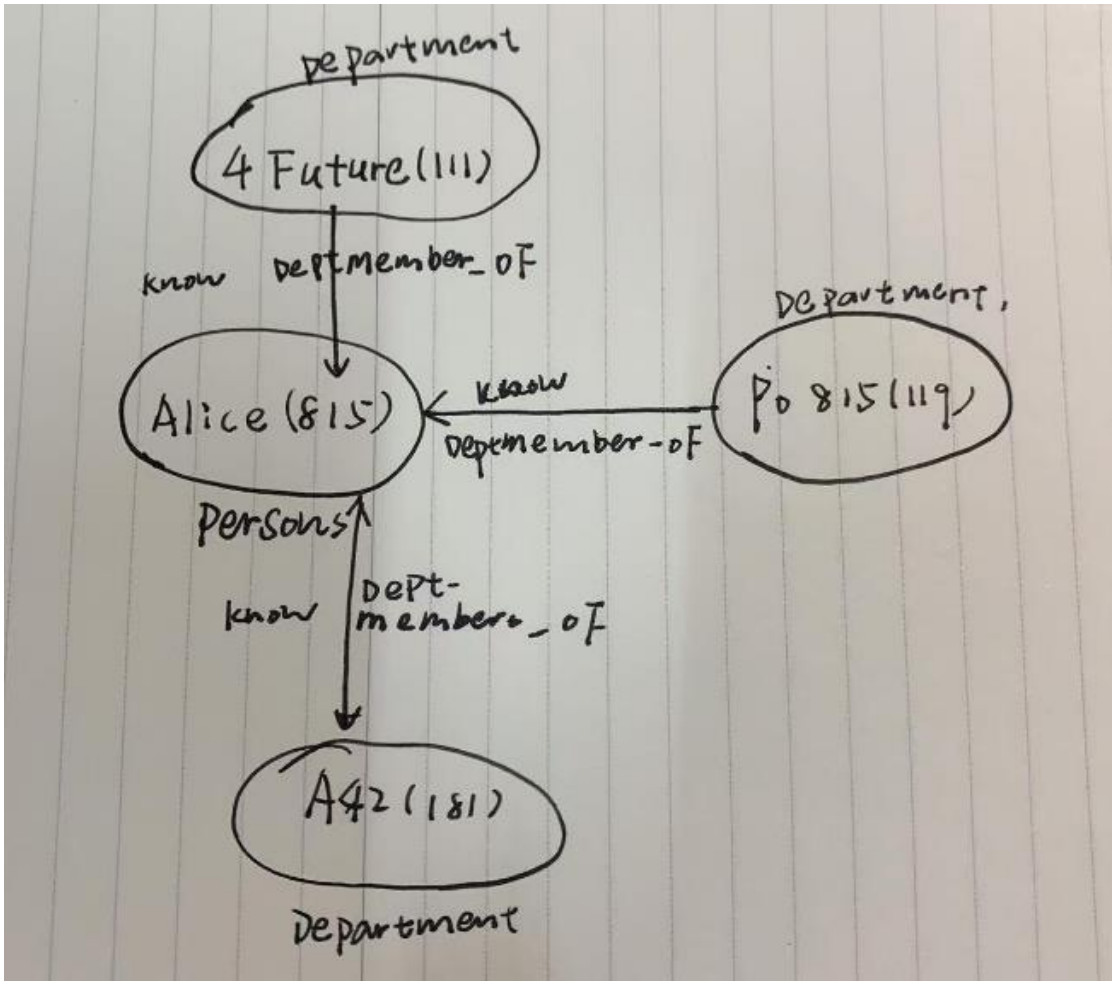
3.



.

4. CREATE (c: Company {name: 'NEO'}), (p: Person {name: 'Lan'})
CREATE (p)- [: EMPLOYMENT]->(c)
CREATE (j: Job {start: '2011-01-05'}), (p)- [: EMPLOYER]->(j)
CREATE (j)- [: ROLE]->(r), (r: Role {name: 'engineer'})

5.



6.

a)

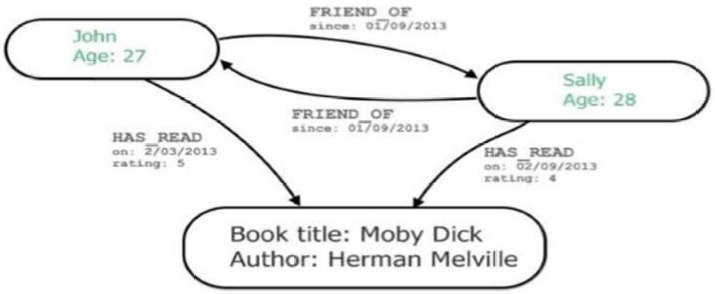
State the purpose of the following script:

```
MATCH (sally:Person { name: 'Sally' })
MATCH (john:Person { name: 'John' })
MATCH (sally)-[r:FRIEND_OF]-(john)
RETURN r.since AS friends_since
```

Answer:

The purpose of this script is to find two individuals named Sally and John in a graph database, and to check if there exists a relationship labeled **FRIEND_OF** between them. If such a relationship exists, the script returns the since property of this relationship, which indicates the date when Sally and John became friends. Here, r represents the **FRIEND_OF** relationship between Sally and John, and **r.since** is a property of this relationship that records the starting time of their friendship. Ultimately, this starting time is returned with the name **friends_since.**

b) Attempt to write the Cypher script for the following queries:



i) What is the average rating of Moby Dick?

```
MATCH (p:Person)-[:READ]->(b:Book {title: 'Moby Dick'})
RETURN AVG(p.rating) AS averageRating
```

ii) Who are the authors of Moby Dick?

```
MATCH (b:Book {title: 'Moby Dick'})
RETURN b.author AS authorName
```

iii) How old is Sally?

```
MATCH (p:Person {name: 'Sally'})
RETURN p.age AS sallyAge
```

iv) Who read Moby Dick first, Sally or John?

```
MATCH
    (b:Book {title: 'Moby Dick'})<-[:READ]-(s:Person {name: 'Sally'}),
    (b:Book {title: 'Moby Dick'})<-[:READ]-(j:Person {name: 'John'})
WITH s, j, s.readDate AS sallyDate, j.readDate AS johnDate
ORDER BY [sallyDate, johnDate] ASC
RETURN
    CASE
        WHEN sallyDate < johnDate THEN 'Sally'
```

```
    WHEN sallyDate > johnDate THEN 'John'
    ELSE 'Both read it on the same day'
END AS firstReader
```