Progetto NoSQL

Modelli e Architetture Avanzati di Basi di Dati

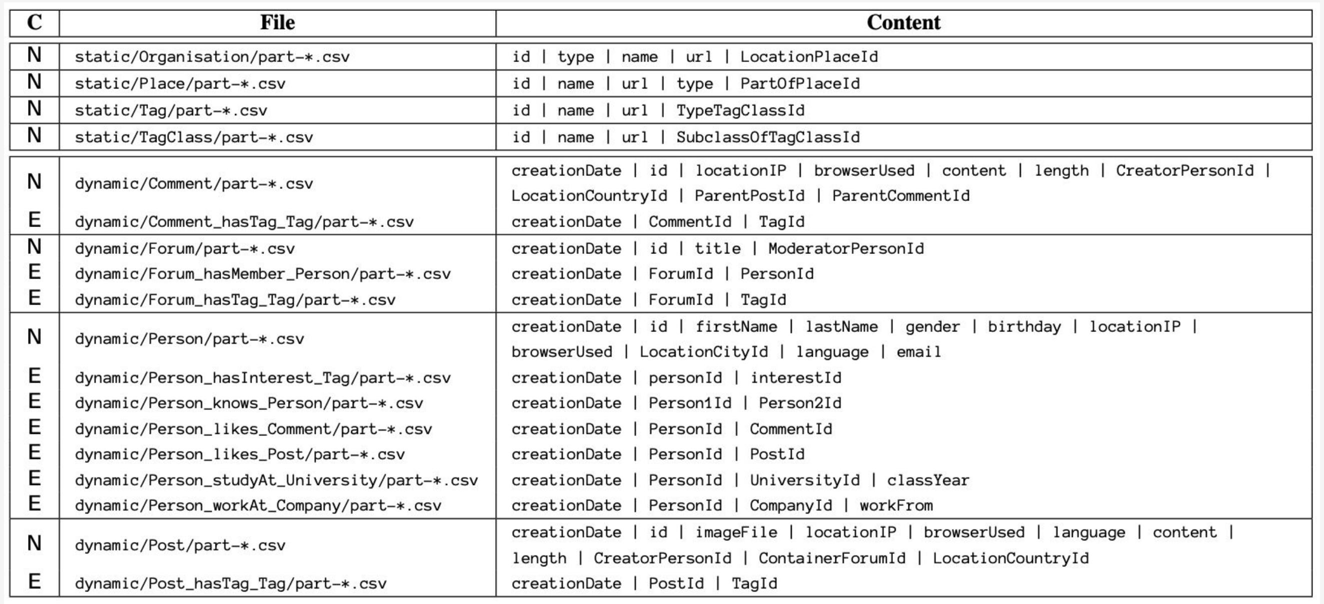
Camoia Andrea

UNIVERSITÀ DI TORINO

Introduzione

Data Generation

Il dataset è stato generato tramite l'**LDBC Social Network Benchmark** Datagen (Spark) nella sua versione normalizzata in **"Composite Merged FK"**



C	File	Content
N	static/Organisation/part-*.csv	id | type | name | url | LocationPlaceId
N	static/Place/part-*.csv	id | name | url | type | PartOfPlaceId
N	static/Tag/part-*.csv	id | name | url | TypeTagClassId
N	static/TagClass/part-*.csv	id | name | url | SubclassOfTagClassId
N	dynamic/Comment/part-*.csv	creationDate | id | locationIP | browserUsed | content | length | CreatorPersonId | LocationCountryId | ParentPostId | ParentCommentId
E	dynamic/Comment_hasTag_Tag/part-*.csv	creationDate | CommentId | TagId
N	dynamic/Forum/part-*.csv	creationDate | id | title | ModeratorPersonId
E	dynamic/Forum_hasMember_Person/part-*.csv	creationDate | ForumId | PersonId
E	dynamic/Forum_hasTag_Tag/part-*.csv	creationDate | ForumId | TagId
N	dynamic/Person/part-*.csv	creationDate | id | firstName | lastName | gender | birthday | locationIP | browserUsed | LocationCityId | language | email
E	dynamic/Person_hasInterest_Tag/part-*.csv	creationDate | personId | interestId
E	dynamic/Person_knows_Person/part-*.csv	creationDate | Person1Id | Person2Id
E	dynamic/Person_likes_Comment/part-*.csv	creationDate | PersonId | CommentId
E	dynamic/Person_likes_Post/part-*.csv	creationDate | PersonId | PostId
E	dynamic/Person_studyAt_University/part-*.csv	creationDate | PersonId | UniversityId | classYear
E	dynamic/Person_workAt_Company/part-*.csv	creationDate | PersonId | CompanyId | workFrom
N	dynamic/Post/part-*.csv	creationDate | id | imageFile | locationIP | browserUsed | language | content | length | CreatorPersonId | ContainerForumId | LocationCountryId
E	dynamic/Post_hasTag_Tag/part-*.csv	creationDate | PostId | TagId

Progettazione

Document-Based Database

Document Oriented Database implementato tramite MongoDB: al suo interno vengono gestite tutte le entità con molti attributi ma che partecipano a poche relazioni.

- Person
- Organization
- Place
- Person_workAt_Company
- Person_studyAt_University

Database a Grafo

Database a Grafo implementato tramite Neo4j, in cui vengono gestite tutte le relazioni tra le entità del dataset, principalmente quelle sociali.

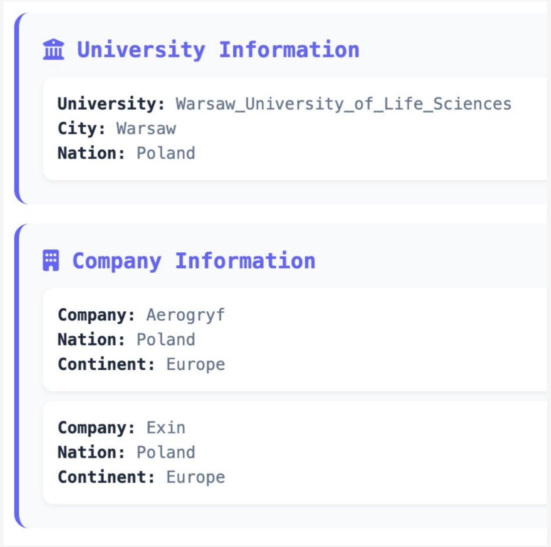
```
-   (Person)-[KNOWS]->(Person)
-   (Person)-[CREATED]->(Post)
-   (Person)-[LIKES]->(Post)
-   (Post)-[HASTAG]->(Tag)
```

Query

Query 1

Query 1: Location Finder

Individuare la posizione dell'università dove ha studiato una certa persona e quella dell'azienda in cui lavora.



University Information

University: Warsaw_University_of_Life_Sciences
City: Warsaw
Nation: Poland

Company Information

Company: Aerogryf
Nation: Poland
Continent: Europe

Company: Exin
Nation: Poland
Continent: Europe

- Implementata in MongoDB tramite singole micro-query in modo da ottimizzare la distribuzione.

- Tante sinquery ma con poco traffico.

Query

Query 2: Cross-Database

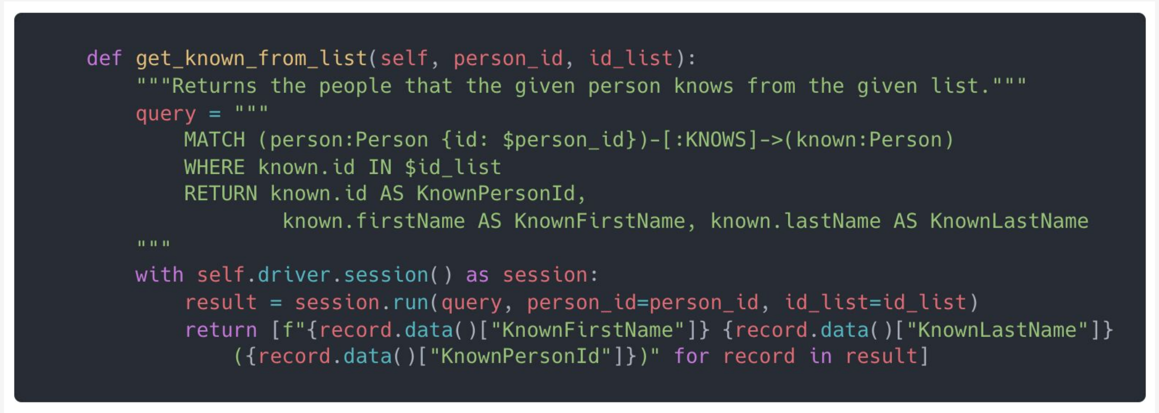
Query 2: Known Colleagues

Data una persona, individuare tutte le altre persone che conosce ("knows") all'interno dell'azienda in in cui lavora / dell'università in cui studia.

- In MongoDB:

```
university_colleagues = mongo_manager.get_university_colleagues(person_id)
work_colleagues = mongo_manager.get_work_colleagues(person_id)
```

- In Neo4j:



```
    def get_known_from_list(self, person_id, id_list):
        """Returns the people that the given person knows from the given list."""
        query = """
            MATCH (person:Person {id: $person_id})-[:KNOWS]->(known:Person)
            WHERE known.id IN $id_list
            RETURN known.id AS KnownPersonId,
                    known.firstName AS KnownFirstName, known.lastName AS KnownLastName
        """
        with self.driver.session() as session:
            result = session.run(query, person_id=person_id, id_list=id_list)
            return [f"{record.data()["KnownFirstName"]} {record.data()["KnownLastName"]}
                ({record.data()["KnownPersonId"]})" for record in result]
```

Query

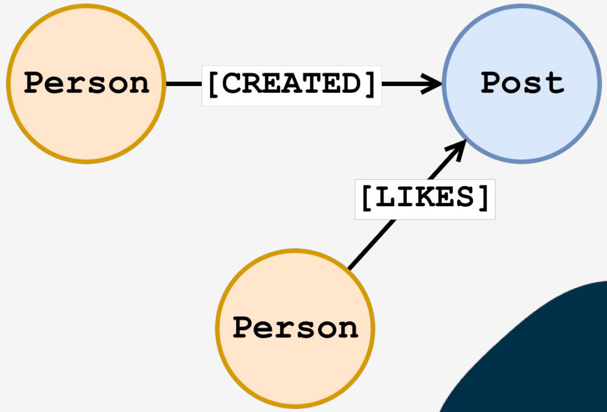
Query 3

Query 3: Most Likes

Individuare la persona più influente in termini di Likes totali ai suoi post.

- Si matchano e conteggiano le relazioni nel grafo.

```
def get_most_liked_person(self):
    """Returns the person with the most likes (across all posts)."""
    query = """
    MATCH (author:Person)-[:CREATED]->(post:Post)<-[like:LIKES]-(:Person)
    RETURN author.firstName AS Name,
           author.lastName AS Surname,
           count(like) AS TotalLikes
    ORDER BY TotalLikes DESC
    LIMIT 1
    """
    with self.driver.session() as session:
        result = session.run(query)
        record = result.single()
        if record:
            return record.data()

        return None
```

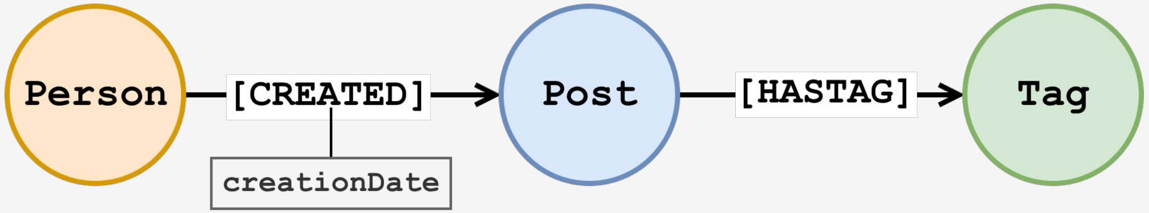


Query 4

Query 4: Top Tag

Individuare i 5 Tag più utilizzati in un dato range temporale

- Si sfrutta l’attributo della relazione CREATED



Query 5

Query 5: Most Influent Person

Individuare l'utente più popolare (in termini di persone che lo conoscono) all'interno di un'università.

- In MongoDB:

```
university_students = mongo_manager.get_university_students(university_id)
```

- In Neo4j:

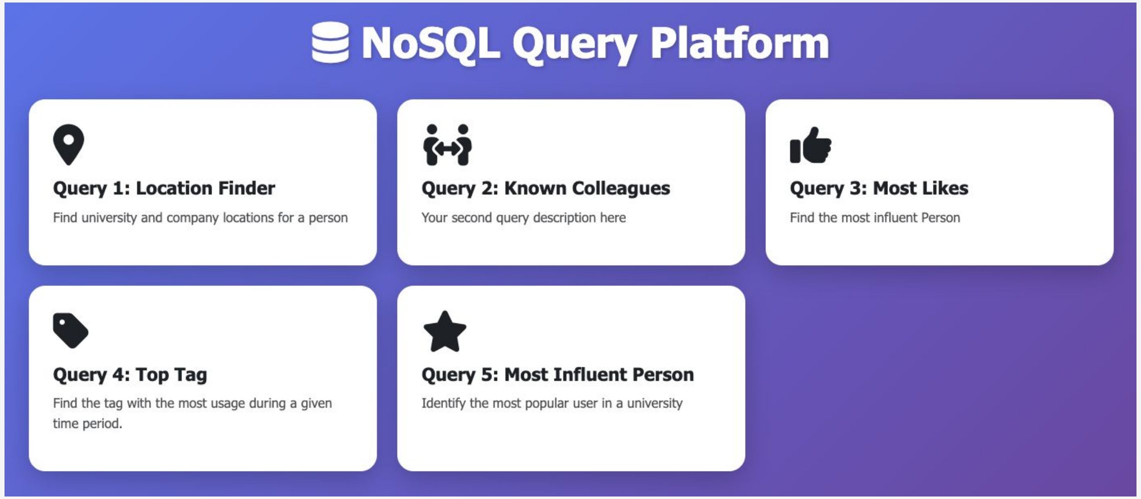
```
def get_most_popular_in_list(self, person_ids):
    """ get the most known person from a list of people """
    query = """
        MATCH (person:Person)-[:KNOWS]->(known:Person)
        WHERE known.id IN $person_ids
        RETURN known.id AS KnownPersonId, count(person) AS KnownCount
        ORDER BY KnownCount DESC
        LIMIT 1
    """
    with self.driver.session() as session:
        result = session.run(query, person_ids=person_ids)
        record = result.single()
        if record:
            return record.data()

        return None
```

Implementazione

WebApp

WebApp dimostrativa che permette di eseguire e visualizzare i risultati delle query.

- Realizzata in Python tramite la libreria EEL



NoSQL Query Platform

Query 1: Location Finder
Find university and company locations for a person

Query 2: Known Colleagues
Your second query description here

Query 3: Most Likes
Find the most influent Person

Query 4: Top Tag
Find the tag with the most usage during a given time period.

Query 5: Most Influent Person
Identify the most popular user in a university