CITS5504 Project 2: Graph Database Design and Cypher Query

Karen Lee Huan Yi (23715313) & Yiren Wang (23794201)

2024-04-30

Contents

Introduction

Uniting national teams worldwide in pursuit of the esteemed prize, the 2014 FIFA World Cup emerged as a significant global football competition. This project seeks to employ a graph database approach to analyze the participants of the 2014 FIFA World Cup. We aim to learn more about player connections, team performance, and individual player statistics through the design and implementation of a graph database.

Description of Dataset

This project utilizes a dataset that includes player profiles from the 2014 FIFA World Cup. It contains a variety of information, including player names, positions, clubs, nations, age, height, quantity of caps earned for the national team, and goals scored. The CSV-formatted dataset has 736 rows, each of which represents a distinct player.

The dataset's main columns are:

Player id: A distinct number assigned to every player.

Player: The player's name.

Position: The player's on-field position (such as forward, midfielder, defender, or goalkeeper).

Club: The club the player plays for.

Club (Country): The country of player's club located.

D.O.B: Player's date of birth in DD.MM.YYYY format.

Age: Player's age at 2014 FIFA World Cup.

Height (cm): Player's height in centimeters.

Country: Player's country.

Caps: Player's total number of games globally before the competition.

International goals: Player's total goal in national team games before the competition.

Plays in home country?: Specifies whether the player is associated with a club in their home country by TRUE / FALSE.

Design and Implementation Process

Graph Database Design

Creating the graph database schema is the first task in the project. In order to determine the key components and relationships that will serve as a structure for the graph database, the data being analyzed is thoroughly examined. Creating a schema that precisely captured the relationships and interactions between players, teams, and nations is one of the project's objective. Three primary entities have been identified as the nodes:

Player: A representation of a specific player that takes part in FIFA World Cup 2014. Player's name, position, age, height, amount of caps for the country, and international goals make up the player node.

Country: Represent the nations that the participating players are competing for. The country node contains the country id and their respective country names.

Club: Represent the clubs to which the players are associated. The club node contains the club id with its respective club name.

The following important relationships are determined in order to identify the relationships between those entities:

PLAYS_FOR: This link establishes between players and their respective clubs. It represents a player's connection with a specific club.

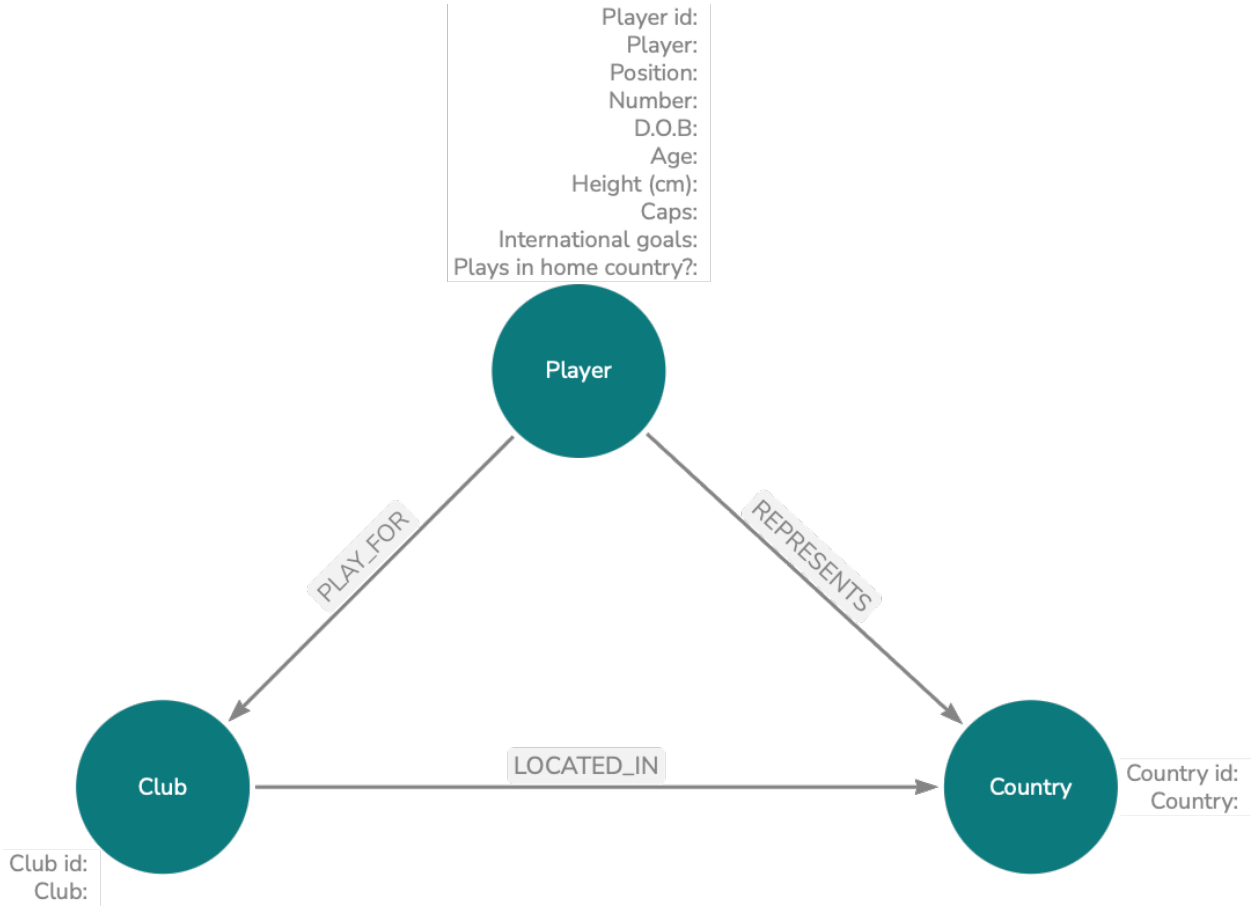
REPRESENTS: With this relationship, players represent their respective countries in the tournament. It represents the country that a player is associated with.

LOCATED_IN: This link establishes between clubs and the nations in which they are based. It determines the specific location for each club.

Graph Database Schema

A graphic representation of the graph database schema is generated using the Arrows app. The entities are depicted as nodes in the schema diagram, and the relationships between the nodes are shown as edges. The attributes of each entity as well as the type and direction of relationships between the entities are able been established by using the Arrows app.

The following attachment displays the generated graph database schema:



The complicated connections across players, clubs, and countries are seek to be captured by creating the schema in this process, which will allow us to run sophisticated queries and extract valuable insights from the data.

Data Preprocessing and Extract, Transform, Load (ETL)

After creating the graph database schema, the following step is preprocessing the dataset and carrying out ETL procedures to import the data into the graph database which is Neo4j.

Extract

Extracting the data out of the source is the first stage in the ETL process. This instance involves a CSV file named `FIFA2014 - all players.csv` that contains information on the players that take part in the FIFA World Cup 2014. The CSV file is read into a Data Frame named `FIFA` using Python's `pandas` package. `Pandas` is a powerful data manipulation package that offers user-friendly data structures and tools for data analysis.

```
import pandas as pd

FIFA = pd.read_csv('FIFA2014 - all players.csv')
```

Transform

The extracted data has to be transformed into a format that can be imported into the graph database. Different CSV files for nodes and relationships are created according to the designed graph database.

Create Nodes Tables

1. Create `Player` node CSV

```
# Create Player node CSV

# Convert the date format to 'yyyy-MM-dd'
FIFA['D.O.B'] = pd.to_datetime(FIFA['D.O.B'], format='%d.%m.%Y').dt.strftime('%Y-%m-%d')

player = FIFA.loc[:, ['Player id', 'Player', 'Position', 'Number', 'D.O.B', 'Age',
'Height (cm)', 'Caps', 'International goals', 'Plays in home country?']]
player.to_csv('player.csv', index=False)
```

- The `player` Data Frame, which will be utilized for creating the `Player` node in the graph database, is created by selecting the relevant columns from the `FIFA` Data frame.
- Using the `dt.strftime()` method and the `pd.to_datetime()` function, the `D.O.B` column is converted to the 'yyyy-MM-dd' format. Thisguarantees that the date format and the data type in Neo4j are compatible.
- Lastly, the `player` Data Frame is exported to a CSV file named `player.csv` by using `to_csv()` method.

2. Create `Country` node CSV

```
# Create Country node CSV
country = FIFA[['Country']].drop_duplicates()
country['Country id'] = range(1, len(country) + 1)
country = country[['Country id', 'Country']]
country.to_csv('country.csv', index=False)
```

- To create the `country` Data Frame, `drop_duplicates()` method is used to exract the unique country names from the `FIFA` Data Frame.
- `range()` function is used to provide each country a distinct `Country id` value as primary key.
- The `country` Data Frame is rearranged so that column `Country id` followed by `Country`.
- The `country` Data Frame is then exported to a CSV file `country.csv`.

3. Create `Club` node CSV

```
# Create Club node CSV
club = FIFA[['Club']].drop_duplicates()
club['Club id'] = range(1, len(club) + 1)
club = club[['Club id', 'Club']]
club.to_csv('club.csv', index=False)
```

- The method is the same as used to create the CSV for `Country` node.
- To create the `club` Data Frame, `drop_duplicates()` method is used to extract the unique club names from `FIFA` Data Frame.
- `range()` function is used to provide each club with a unique `Club id` value.
- `Club id` and `Club` are now the columns in the Data Frame.
- The `club` Data Frame is then exported to a CSV file called `club.csv`.

Create Relationships Tables The relationships between the nodes in the graph database have to be established by creating relationship tables after the node CSV files have been created.

1. Create `REPRESENT` relationship

```
# Create rel_represent CSV
rel_represent = pd.merge(FIFA[['Player id', 'Country']], country, on='Country')
rel_represent = rel_represent[['Player id', 'Country id']]
rel_represent.to_csv('rel_represent.csv', index=False)
```

- The `Player id` and `Country` columns from the `FIFA` Data Frame are joined with the `country` Data Frame using `merge()` function.
- The `rel_represent` Data Frame, which illustrates the `REPRESENT` relationship between `player` and `country`, is created by selecting the `Player id` and `Country id` columns.
- The `rel_represent` Data Frame is then exported as a CSV file named `rel_represent.csv`.

2. Create `LOCATED_IN` relationship

```
# Create rel_located_in CSV
rel_located_in = pd.merge(FIFA[['Club', 'Club (country)']], club, on='Club')
rel_located_in = pd.merge(rel_located_in, country, left_on='Club (country)', right_on='Country')
rel_located_in = rel_located_in[['Club id', 'Country id']]
rel_located_in.to_csv('rel_located_in.csv', index=False)
```

- The `Club` and `Club (country)` columns from the `FIFA` Data Frame are joined with the `club` Data Frame using `merge()` function.
- The `Country id` for every club is then obtained by merging the resulting Data frame with `country` Data Frame based on the `Club (country)` and `Country` columns.
- The `rel_located_in` Data Frame, which illustrates the `LOCATED_IN` relationship between `club` and `country`, is created by selecting `Club id` and `Country id` columns.
- The `rel_located_in` Data Frame is then exported as a CSV file named `rel_located_in.csv`.

3. Create `PLAY_FOR` relationship

```
# Create rel_play_for CSV
rel_play_for = pd.merge(FIFA[['Player id', 'Club']], club, on='Club')
rel_play_for = rel_play_for[['Player id', 'Club id']]
rel_play_for.to_csv('rel_play_for.csv', index=False)
```

- The `Club` and `Player id` columns from the `FIFA` Data Frame are joined with the `club` Data Frame based on the `Club` column using the `merge()` function.
- In order to generate the `rel_play_for` Data Frame, which represents the `PLAY_FOR` relationship between `player` and `club`, columns `Player id` and `Club id` are chosen as the columns in this Data Frame.
- The `rel_play_for` Data Frame is finally exported as a CSV file named `rel_play_for.csv`.

Load

Loading the data that is transformed into the Neo4j graph database is the final step. Based on the CSV files created, Cypher queries are utilized to establish the nodes and relationships.

1. Create `Player` node

```
// Load Player nodes
LOAD CSV WITH HEADERS FROM 'file:///player.csv' AS row
CREATE (:Player {
  playerId: toInteger(row['Player id']),
  name: row['Player'],
  position: row['Position'],
  number: toInteger(row['Number']),
  dob: date(row['D.O.B']),
  age: toInteger(row['Age']),
  height: toInteger(row['Height (cm)']),
  caps: toInteger(row['Caps']),
  internationalGoals: toInteger(row['International goals']),
  playsInHomeCountry: (row['Plays in home country?'] = 'TRUE')
});
```

- `LOAD CSV` command is used to read the data from `player.csv` file.
- Using the `CREATE` clause, `Player` node and its properties are created according to the data in the CSV files.

- Where necessary, the string values are converted to integers using the `toInteger()` function.
- In order to parse the `D.O.B` column as a date value, the `date()` function is utilized.

2. Create `Country` node

```
// Load Country nodes
LOAD CSV WITH HEADERS FROM 'file:///country.csv' AS row
CREATE (:Country {
  countryId: toInteger(row['Country id']),
  name: row['Country']
});
```

- `LOAD CSV` command is used to read the data from `country.csv` file.
- Using the `CREATE` clause, `Country` node and its properties are created according to the data in the CSV files.
- The `Country id` field is converted to integer using `toInteger()` function.

3. Create `Club` node

```
// Load Club nodes
LOAD CSV WITH HEADERS FROM 'file:///club.csv' AS row
CREATE (:Club {
  clubId: toInteger(row['Club id']),
  name: row['Club']
});
```

- `LOAD CSV` command is used to read the data from `club.csv` file.
- Using the `CREATE` clause, `Club` node and its properties are created according to the data in the CSV files.
- The `Club id` field is converted to integer using `toInteger()` function.

4. Create `REPRESENT` relationship

```
// Create rel_represent relationships
LOAD CSV WITH HEADERS FROM 'file:///rel_represent.csv' AS row
MATCH (p:Player {playerId: toInteger(row['Player id'])}),
      (c:Country {countryId: toInteger(row['Country id'])})
CREATE (p)-[:REPRESENTS]->(c);
```

- `LOAD CSV` command is used to extract the data from the `rel_represent.csv` file.
- Based on their respective IDs, `MATCH` clause is used to identify the `Player` and `Country` nodes.
- `CREATE` clause is used to establish the `REPRESENT` relationship between the matched `Player` and `Country` nodes.

5. Create `LOCATED_IN` relationship

```
// Create rel_located_in relationships
LOAD CSV WITH HEADERS FROM 'file:///rel_located_in.csv' AS row
MATCH (c:Club {clubId: toInteger(row['Club id'])}),
      (co:Country {countryId: toInteger(row['Country id'])})
CREATE (c)-[:LOCATED_IN]->(co);
```

- `LOAD CSV` command is used to extract the data from the `rel_located_in.csv` file.
- Based on their respective IDs, `MATCH` clause is used to identify the `Club` and `Country` nodes.
- `CREATE` clause is used to establish the `LOCATED_IN` relationship between the matched `Club` and `Country` nodes.

6. Create `PLAY_FOR` relationship

```
// Create rel_play_for relationships
LOAD CSV WITH HEADERS FROM 'file:///rel_play_for.csv' AS row
MATCH (p:Player {playerId: toInteger(row['Player id'])}),
      (c:Club {clubId: toInteger(row['Club id'])})
CREATE (p)-[:PLAYS_FOR]->(c);
```

- `LOAD CSV` command is used to extract the data from the `rel_play_for.csv` file.
- Based on their respective IDs, `MATCH` clause is used to identify the `Player` and `Club` nodes.
- `CREATE` clause is used to establish the `PLAY_FOR` relationship between the matched `Player` and `Club` nodes.

The original FIFA 2014 World Cup dataset is converted into a format that can be imported into Neo4j graph database through the implementation of ETL procedure. The data is extracted and transformed using the Python code, which also generates separate CSV files for nodes and relationships. the required nodes and relationships are then created in the graph database by the Cypher queries, which import the data from the CSV files according to the designed schema. Complex queries can be run and relationships between players, clubs, and countries can be investigated in the context of the 2014 FIFA World Cup as a result of this methodical and effective methodology of integrating the data into the graph database.

Cypher Queries and Results

1. What is the jersy number of the player with 229397?

```
1 //Question 1
2 MATCH (p:Player {playerId: 229397})
3 RETURN p.name AS PlayerName, p.number AS jerseyNumber;
```

Table | Text

	PlayerName	jerseyNumber
1	"Lionel MESSI"	10

The jersy number of `Lionel MESSI` with `playerId 229397` is `10`.

2. Which clubs are based in Nigeria?

```
1 //Question 2
2 MATCH (c:Club)-[:LOCATED_IN]→(co:Country {name: 'Nigeria'})
3 RETURN c
```

The clubs that based in `Nigeria` are:

- `Engu Rangers FC`
- `Warri Wolves FC`
- `Sunshine Stars FC`

3. Which club does Lionel Messi play for?

```
1 //Question 3
2 MATCH (p:Player {name: 'Lionel MESSI'})-[:PLAYS_FOR]→(c:Club)
3 RETURN p.name AS playerName, c.name AS clubName
```

Table

	playerName	clubName
1	"Lionel MESSI"	"FC Barcelona"

Text

`Lionel MESSI` plays for `FC Barcelona`.

4. How old is Lionel Messi?

```
1 //Question 4
2 MATCH (p:Player {name: 'Lionel MESSI'})
3 RETURN p.name AS playerName, p.age AS age
```

Table

	playerName	age
1	"Lionel MESSI"	26

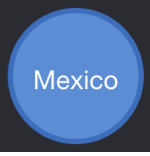
Text

`Lionel MESSI` was `26` years old in 2014.

5. In which country is the club that Alan PULIDO plays for?

```
1 //Question 5
2 MATCH (p:Player {name: 'Alan PULIDO'})-[:PLAYS_FOR]→(c:Club)-[:LOCATED_IN]→(co:Country)
3 RETURN DISTINCT co
```

Graph



Table

The country that the club `Alan PULIDO` plays for is `Mexico`.

6. Find a club that has players from Brazil.

```
1 //Question 6
2 MATCH (p:Player)-[:PLAYS_FOR]→(c:Club), (p)-[:REPRESENTS]→(co:Country {name: 'Brazil'})
3 RETURN DISTINCT c
4 LIMIT 1
```

Graph

Table

`Botafogo FR` is one of the club that has `Brazil` players.

7. Find all players play at FC Barcelona, returning in ascending orders of age.

```
//Question 7
MATCH (p:Player)-[:PLAYS_FOR]→(c:Club {name: 'FC Barcelona'})
RETURN p.name AS playerName, p.age AS age, c.name AS clubName
ORDER BY p.age ASC
```

Table | Text | Code

	playerName	age	clubName
1	"NEYMAR"	22	"FC Barcelona"
2	"Sergio BUSQUETS"	25	"FC Barcelona"
3	"Jordi ALBA"	25	"FC Barcelona"
4	"Alexis SANCHEZ"	25	"FC Barcelona"
5	"Pedro RODRIGUEZ"	26	"FC Barcelona"
6	"Alexandre SONG"	26	"FC Barcelona"

Above is the result that illustrates the players play for `FC Barcelona` in ascending order of age.

8. Find all Defender players in national team of Brazil, returning in descending order of caps.

```
//Question 8
MATCH (p:Player)-[:REPRESENTS]→(c:Country {name: 'Brazil'})
WHERE p.position = 'Defender'
RETURN p.name AS playerName, p.position AS position, c.name AS nationality, p.caps AS
caps
ORDER BY p.caps DESC
```

Table | Text | Code

	playerName	position	nationality	caps
1	"DANI ALVES"	"Defender"	"Brazil"	74
2	"MAICON"	"Defender"	"Brazil"	71
3	"THIAGO SILVA"	"Defender"	"Brazil"	45
4	"DAVID LUIZ"	"Defender"	"Brazil"	35
5	"MARCELO"	"Defender"	"Brazil"	30
6	"DANTE"	"Defender"	"Brazil"	12

The screenshot above shows the `Defender` players from `Brazil` with their number of caps in descending order.

9. Find all players born in 1987 and in national team of Argentina, returning in descending order of caps.

```
//Question 9
MATCH (p:Player)-[:REPRESENTS]→(c:Country {name: 'Argentina'})
WHERE p.dob.year = 1987
RETURN p.name AS playerName, "1987" AS dateOfBirth, c.name AS Country, p.caps AS caps
ORDER BY p.caps DESC
```

Table | Text | Code

	playerName	dateOfBirth	Country	caps
1	"Lionel MESSI"	"1987"	"Argentina"	84
2	"Sergio ROMERO"	"1987"	"Argentina"	45
3	"Gonzalo HIGUAIN"	"1987"	"Argentina"	36

There are three players born in `1987` in `Argentina`. The result is listed in descending order of their caps.

10. Find the players that belongs to the same club in national team of Argentina, returning in descending order of international goals.

```
//Question 10
MATCH (p1:Player)-[:REPRESENTS]→(c:Country {name: 'Argentina'}),
      (p1)-[:PLAYS_FOR]→(club:Club),
      (p2:Player)-[:REPRESENTS]→(c),
      (p2)-[:PLAYS_FOR]→(club)
WHERE p1 <> p2
RETURN p1.name AS player1, p2.name AS player2, club.name AS commonClub, c.name AS nationalTeam,
       p1.internationalGoals AS player1Goals, p2.internationalGoals AS player2Goals
ORDER BY p1.internationalGoals DESC, p2.internationalGoals DESC
```

Table | Text | Code

	player1	player2	commonClub	nationalTeam	player1Goals	player2Goals
1	"Lionel MESSI"	"Javier MASCHERANO"	"FC Barcelona"	"Argentina"	37	2
2	"Sergio AGUERO"	"Martin DEMICHELIS"	"Manchester City FC"	"Argentina"	21	2
3	"Sergio AGUERO"	"Pablo ZABALETA"	"Manchester City FC"	"Argentina"	21	0
4	"Gonzalo HIGUAIN"	"Federico FERNANDEZ"	"SSC Napoli"	"Argentina"	20	2
5	"Javier MASCHERANO"	"Lionel MESSI"	"FC Barcelona"	"Argentina"	2	37
6	"Martin DEMICHELIS"	"Sergio AGUERO"	"Manchester City FC"	"Argentina"	2	21

11. Count how many players are born in 1987.

```
//Question 11
MATCH (p:Player)
WHERE p.dob.year = 1987
RETURN COUNT(p) AS playersBornIn1987,  p.dob.year as Year
```

Table | Text

	playersBornIn1987	Year
1	140	1987

There are `140` players born in year `1987`.

12. Which age has the highest participation in the 2014 FIFA World Cup?

```
1 //Question 12
2 MATCH (p:Player)
3 RETURN p.age AS age, COUNT(p) AS playerCount
4 ORDER BY playerCount DESC
```

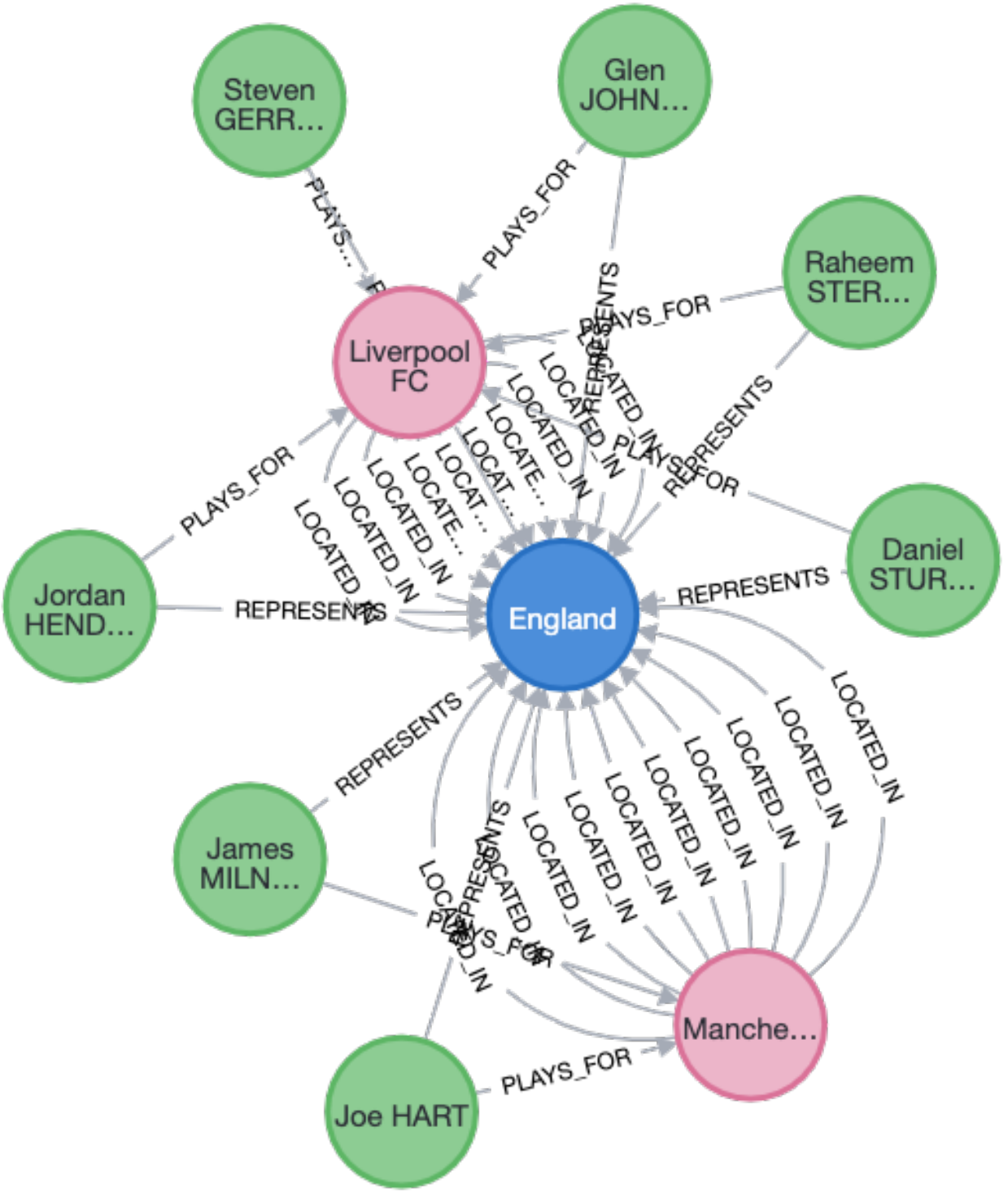
Table | Text | Code

	age	playerCount
1	27	154
2	28	148
3	29	130
4	26	128
5	25	124
6	24	124

Age 27 has the highest participation in 2014 FIFA World Cup.

13. Find the path with a length of 2 or 3 between Liverpool FC and Manchester City FC.

```
1 //Question 13
2 MATCH path = (l:Club {name: 'Liverpool FC'})-[*2..3]-(m:Club {name: 'Manchester City FC'})
3 RETURN path
```



The query investigates the relationships between the two clubs by examining the players who have played for both clubs. The following paths between Liverpool FC and Manchester City FC are shown in the graph

below:

- Relationship between the two clubs is established by the connection, which displyas James Milner from Manchester City FC and Glen Johnson from Liverpool FC are teammates to represent England in the tournament.
- The path also demonstrates that Joe Hart from Manchester City FC and Steven Gerrard from Liverpool FC are both players of the England team, highlighting the clibs' relationship through their players' national team representation.

As shown by the paths identified in the graph and obtained by the Cypher query, Liverpool FC and Manchester City FC are related due to their players' participation in the England national team. It demonstrates how players from competing clubs join together to serve their nation and how involvement in the national team fosters relationships between clubs unexpectedly.

14. Find the top 5 countries with players who have the highest average number of international goals. Return the countries and their average international goals in descending order.

```
1 //Question 14
2 MATCH (p:Player)-[:REPRESENTS]→(c:Country)
3 WITH c, AVG(p.internationalGoals) AS avgGoals
4 RETURN c.name AS country, avgGoals
5 ORDER BY avgGoals DESC
6 LIMIT 5
```

Table | Text | Code

	country	avgGoals
1	"Germany"	9.521739130434783
2	"Spain"	9.434782608695652
3	"Netherlands"	7.0
4	"Uruguay"	6.217391304347827
5	"Ivory Coast"	6.130434782608696

Top five countries with players have the highest average number of international goals are:

- `Germany`
- `Spain`
- `Netherlands`
- `Uruguay`
- `Ivory Coast`

15. Identify pairs of players from the same national team who play in different positions but have the closest number of caps. Return these pairs along with their positions and the difference in caps.

```
//Question 15
MATCH (p1:Player)-[:REPRESENTS]→(c:Country)←[:REPRESENTS]-(p2:Player)
WHERE p1.position <> p2.position
WITH p1, p2, ABS(p1.caps - p2.caps) AS capsDiff, c.name AS country
ORDER BY capsDiff
RETURN p1.name AS player1, p1.position AS player1Position, p1.caps AS player1Caps,
       p2.name AS player2, p2.position AS player2Position, p2.caps AS player2Caps,
       country, capsDiff
```

Table | Text | Code

	player1	player1Position	player1Caps	player2	player2Position	player2Caps	country	capsDiff
1	"Alfredo TALAVERA"	"Goalkeeper"	14	"Diego REYES"	"Defender"	14	"Mexico"	0
2	"Carlos PENA"	"Midfielder"	14	"Diego REYES"	"Defender"	14	"Mexico"	0
3	"Marco FABIAN"	"Midfielder"	14	"Diego REYES"	"Defender"	14	"Mexico"	0
4	"Miguel PONCE"	"Defender"	7	"Isaac BRIZUELA"	"Midfielder"	7	"Mexico"	0
5	"Miguel LAYUN"	"Defender"	12	"Hector HERRERA"	"Midfielder"	12	"Mexico"	0
6	"Isaac BRIZUELA"	"Midfielder"	7	"Miguel PONCE"	"Defender"	7	"Mexico"	0

16. Find the clubs with the most players participating in 2014 FIFA World Cup.

```
//Find the clubs with the most players participating in 2014 FIFA World Cup
MATCH (p:Player)-[:PLAYS_FOR]→(c:Club)
RETURN c.name AS club, COUNT(p) AS playerCount
ORDER BY playerCount DESC
```

Table | Text | Code

	club	playerCount
1	"FC Bayern Muenchen"	15
2	"Manchester United FC"	14
3	"FC Barcelona"	13
4	"SSC Napoli"	12
5	"Chelsea FC"	12
6	"Real Madrid CF"	12

Club `FC Bayern Muenchen` has the most players participating in 2014 FIFA World Cup.

17. Identify national teams with high diversity in club representation.

```
//Identify national teams with high diversity in club representation
MATCH (p:Player)-[:REPRESENTS]→(c:Country)
MATCH (p)-[:PLAYS_FOR]→(club:Club)-[:LOCATED_IN]→(clubCountry:Country)
WITH c.name AS country, COLLECT(DISTINCT clubCountry.name) AS clubCountries
RETURN country, SIZE(clubCountries) AS diversityScore, clubCountries
ORDER BY diversityScore DESC
```

Table | Text | Code

	country	diversityScore	clubCountries
1	"Columbia"	10	["Italy", "France", "England", "Netherlands", "Columbia", "Spain", "Argentina", "Portugal", "Germany", "Mexico"]
2	"Uruguay"	9	["Spain", "Italy", "England", "Brazil", "Portugal", "France", "Mexico", "Uruguay", "Japan"]
3	"Ghana"	9	["Netherlands", "France", "Belgium", "Germany", "Italy", "Russia", "England", "Greece", "Ghana"]
4	"Australia"	8	["England", "Australia", "South Korea", "Netherlands", "Germany", "Switzerland", "Belgium", "USA"]
5	"Nigeria"	8	["Nigeria", "France", "England", "Netherlands", "Italy", "Belgium", "Russia", "Spain"]
6	"Croatia"	8	["Italy", "France", "England", "Germany", "Spain", "Russia", "Croatia", "Greece"]

The top national teams with the most diverse club representation are displayed in the table.

Columbia

- With a diversity score of 10, Columbia leads all other countries in representation of national teams with players from 10 different clubs.
- Italy, France, England, Netherlands, Columbia, Spain, Argentina, Portugal, Germany, and Mexico are among the countries that make up the club.
- This suggests that players from Columbia are dispersed throughout the world's major football leagues.

The data indicates that players from national teams such as Nigeria, Ghana, Australia, Uruguay, Columbia, and Croatia are represented in a wide variety of clubs across different leagues and nations globally.

Graph Databases vs. Relational Databases: Capabilities and Advantages

Graph databases, which have particular advantages and capabilities in processing complex and interrelated data, have become an outstanding alternative to traditional relational databases. Graph databases are created specifically to effectively store, handle, and query extensively connected and semi-structured data, whereas relational databases are excellent at handling structured and tabular data (Robinson et al., 2015). This essay compares and contrasts the capabilities of relational and graph databases, emphasizing the capabilities that are achievable with graph databases and that are difficult or impractical with relational databases.

The capacity of graph databases to display and query intricate relationships between entities is one of their main advantages. Data is stored as nodes (entities) and edges (relationships) in a graph database, creating a structure similar to a network (Miller, 2013). Social networks, algorithms for recommendation, fraud detection, and other real-world scenarios where relationships are crucial can be intuitively modeled due to this natural representation. Deep insights into the relationships between entities can be obtained through the flexible and effective traversal and querying of these interactions made possible by graph databases (Angles, 2012).

Relational databases, on the other hand, have difficulty efficiently managing strongly related data. Tables, rows, and columns provide the foundation of relational databases, while join tables and foreign keys are commonly used to model relationships (Vicknair et al., 2010). In a relational database, querying and interpreting complex relationships sometimes involve high cost and complicated join procedures, which degrade performance as data volume and complexity rise. On the contrary, graph databases are designed with optimized relationship traversal, enabling quick and effective querying even for complex and deep connections (Buerli & Obispo, 2012).

The adaptability of graph databases in managing developing and semi-structured data is another benefit. Data schemas can evolve over time in many real-world contexts, and new forms of relationships or attributes can appear. Relational databases' rigid table-based structure makes it difficult to adjust to these kinds of changes; schema updates and data migrations are frequently needed (Holzschuher & Peinl, 2013). Graph databases, on the other hand, provide a more flexible and rapid method. The data model can evolve smoothly when business requirements change since nodes and relationships can be added or changed without affecting the schema in its entirety (Barmpis & Kolovos, 2014).

Graph databases are also excellent at handling complicated traversal and pattern matching queries. Finding certain connections and patterns in the data is essential in many fields, including fraud detection, recommendation engines, and social network analysis. Strong query languages like Gremlin and Cypher offered by graph databases allow for effective and expressive pattern matching (Robinson et al., 2015). According to Ciglan et al. (2012), these queries have the ability to traverse several levels of relationships, identify certain subgraphs, and identify hidden connections that would be difficult or prohibitive to express in SQL queries employed by relational databases.

Graph databases also provide advantages in performance when handling heavily interconnected data. Graph databases are significantly faster than relational databases at traversing relationships and retrieving related data because of their native graph indexing and storing methods (Jouili & Vansteenberghe, 2013). Graph databases work well for large-scale and real-time applications because of this performance benefit, which increases with data size and complexity (Buerli & Obispo, 2012).

In a nutshell when it comes to managing complicated and connected data, graph databases provide special features and benefits over relational databases. They offer performance benefits for densely connected data, a natural and straightforward approach to describe and query connections, flexibility in managing dynamic data schemas, and strong pattern matching and traversal queries. Relational databases are still useful for maintaining data integrity and organizing structured data, however graph databases are becoming a more attractive option for applications that need real-time performance and in-depth understanding of data relationships. Graph databases are expected to become more crucial in helping to realize the value of connected data as data volume and complexity increase.

Applications of Graph Data Science

Graph Data Science is an effective method that solves difficult problems by integrating data science, machine learning, and graph theory to derive insightful information. Three practical applications of graph data science are examined in this essay: social network analysis, recommendation systems, and fraud detection.

Application 1: Fraud Detection

By taking advantage of the interconnectedness of fraudulent behaviors, Graph Data Science can be used efficiently in fraud detection. Unusual patterns, hidden connections, and possible criminal networks can be found using strategies like community discovery, anomaly detection, and link prediction. Graph Data Science may create thorough and precise fraud detection models by combining graph structure with other data sources. This can assist businesses in minimizing financial losses and guaranteeing regulatory compliance (Akoglu et al., 2015; Sadgali et al., 2019).

Application 2: Recommendation Systems

Graph Data Science plays a vital role in building intelligent recommendation systems. For the purpose of creating intelligent recommendation systems, graph data science is essential. Personalized recommendation generation can be achieved through the use of graph embeddings and collaborative filtering approaches, which include describing user interactions and item relationships as a graph. According to Wang et al. (2019) and Ying et al. (2018), graph-based recommendation systems have the ability to comprehend intricate user preferences, recognize similar users or objects, and generate precise and varied recommendations that boost satisfaction and interaction.

Application 3: Social Network Analysis

In social network analysis, graph data science is widely employed to comprehend the dynamics, structure, and impact of social connections. The identification of influential people, the spread of information, and the formation of social groups are made possible by methods like community detection, centrality measurements, and information diffusion models. Applications for graph-based social network analysis include viral marketing, tracking public opinion, and comprehending the spread of misleading data (Bonchi et al., 2011; Li et al., 2018).

Graph Data Science provides an effective toolkit for resolving challenging issues across a range of fields. Its ability to derive insightful information from linked data is demonstrated by its applications in social network analysis, fraud detection, and recommendation systems. Graph Data Science empowers organizations to make data-driven decisions, increase operational effectiveness, and encourage innovation by utilizing the structure and relationships inside graphs. The importance of Graph Data Science will only rise along with the number and complexity of data, making it a vital strategy in the data science area.

References

Akoglu, L., Tong, H., & Koutra, D. (2015). Graph based anomaly detection and description: a survey. Data mining and knowledge discovery, 29, 626-688.

Angles, R. (2012, April). A comparison of current graph database models. In 2012 IEEE 28th International Conference on Data Engineering Workshops (pp. 171-177). IEEE.

Barmpis, K., & Kolovos, D. S. (2014). Evaluation of contemporary graph databases for efficient persistence of large-scale models. J. Object Technol., 13(3), 3-1.

Bonchi, F., Castillo, C., Gionis, A., & Jaimes, A. (2011). Social network analysis and mining for business applications. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3), 1-37.

Buerli, M., & Obispo, C. P. S. L. (2012). The current state of graph databases. Department of Computer Science, Cal Poly San Luis Obispo, mbuerli@ calpoly. edu, 32(3), 67-83.

Ciglan, M., Averbuch, A., & Hluchy, L. (2012, April). Benchmarking traversal operations over graph databases. In 2012 IEEE 28th International Conference on Data Engineering Workshops (pp. 186-189). IEEE.

Holzschuher, F., & Peinl, R. (2013, March). Performance of graph query languages: comparison of cypher, gremlin and native access in Neo4j. In Proceedings of the Joint EDBT/ICDT 2013 Workshops (pp. 195-204).

Jouili, S., & Vansteenberghe, V. (2013, September). An empirical comparison of graph databases. In 2013 International Conference on Social Computing (pp. 708-715). IEEE.

Miller, J. J. (2013, March). Graph database applications and concepts with Neo4j. In Proceedings of the southern association for information systems conference, Atlanta, GA, USA (Vol. 2324, No. 36, pp. 141-147).

Robinson, I., Webber, J., & Eifrem, E. (2015). Graph databases: new opportunities for connected data. " O'Reilly Media, Inc.".

Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., & Wilkins, D. (2010). A comparison of a graph database and a relational database: a data provenance perspective. In Proceedings of the 48th Annual Southeast Regional Conference (pp. 1-6).

Wang, X., He, X., Wang, M., Feng, F., & Chua, T. S. (2019, July). Neural graph collaborative filtering. In Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval (pp. 165-174).