CITS5504 Project2

Bohan Cai (23719417), Sally Chen (23687599)

Design and Implementation Process

Firstly, the dataset consists of players who participated in the 2014 FIFA World Cup, with attributes for each player including player ID, player name, position, number, club, club (country), date of birth (D.O.B.), age, height, country, number of caps, international goals, and whether or not they played in their home country. So, the next step is to define nodes for our property graph. The main entities identified include Player, Club,

Country, Position, and Year.

Relationships between these nodes are then defined to create links between them. A player can play a specific position, play for a specific club, belong to a national team country, and be born in a specific year. Thus, we establish relationships such as plays as (→ Position), plays for (→ Club), belongs to (→ Country), and born in (→ Year) for Player nodes. Similarly, the Club node is connected to the Country node through the based in (→ Country) relationship indicating the country where the club is located.

The implementation phase starts with preprocessing the dataset. This includes extracting the relevant columns and creating separate CSV files for Player, Club, Country, Position, and Year. The process involves ensuring that there are no duplicates in the Club, Country, Position, and Year datasets.

The final step is to import the data into Neo4j using Cypher queries to create nodes and relationships as defined.

Design A Property Graph

1. Caption: **player**

   Properties:

   - playerId
   - playerName
   - dateOfBirth
   - age
   - height
   - caps
   - internationalGoals
   - playInHomeCountry

   Relationships:

   - plays as (→ position)
   - plays for (→ club)
   - belongs to (→ country)
   - born in (→ year)
2. Caption: **club**

   Properties:

   - clubName

   Relationships:

   - based in (→ country)
3. Caption: **country**

   Properties:

   - countryName
4. Caption: **position**

Properties:

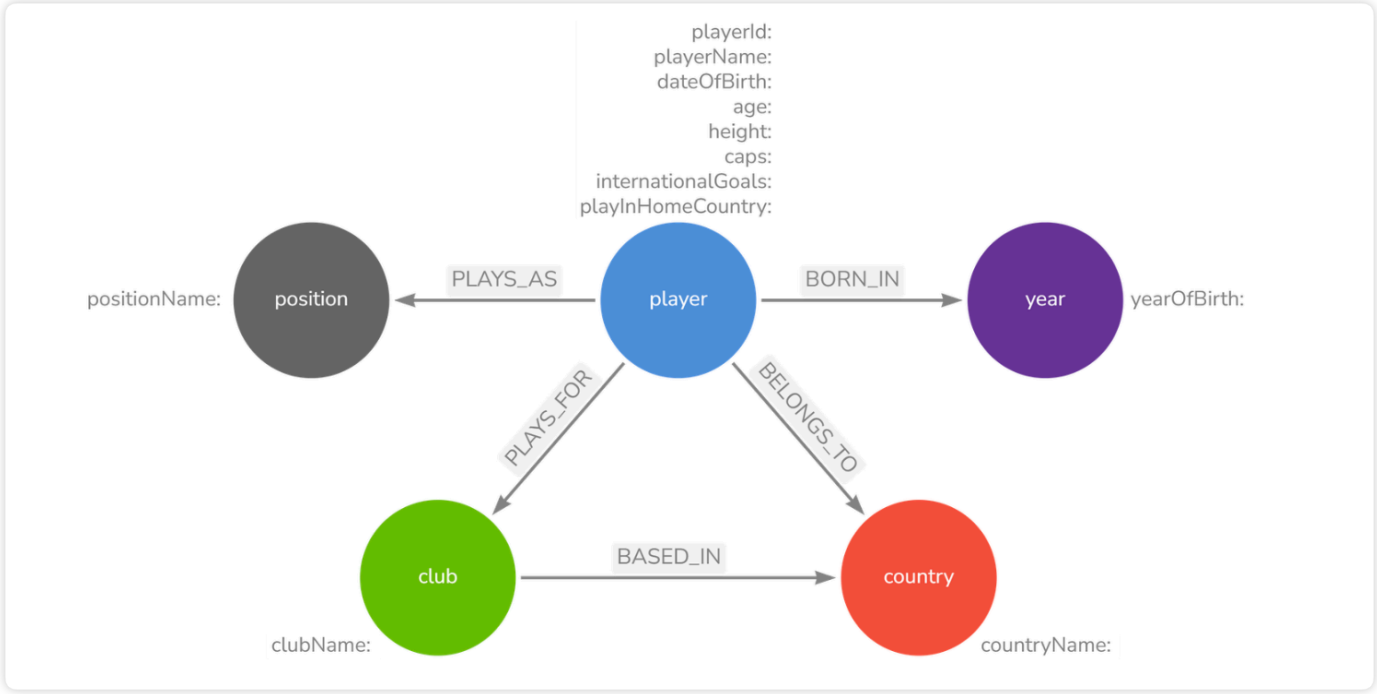
- positionName

5. Caption: **year**

   Properties:

   - yearOfBirth

A Screenshot of Our Design

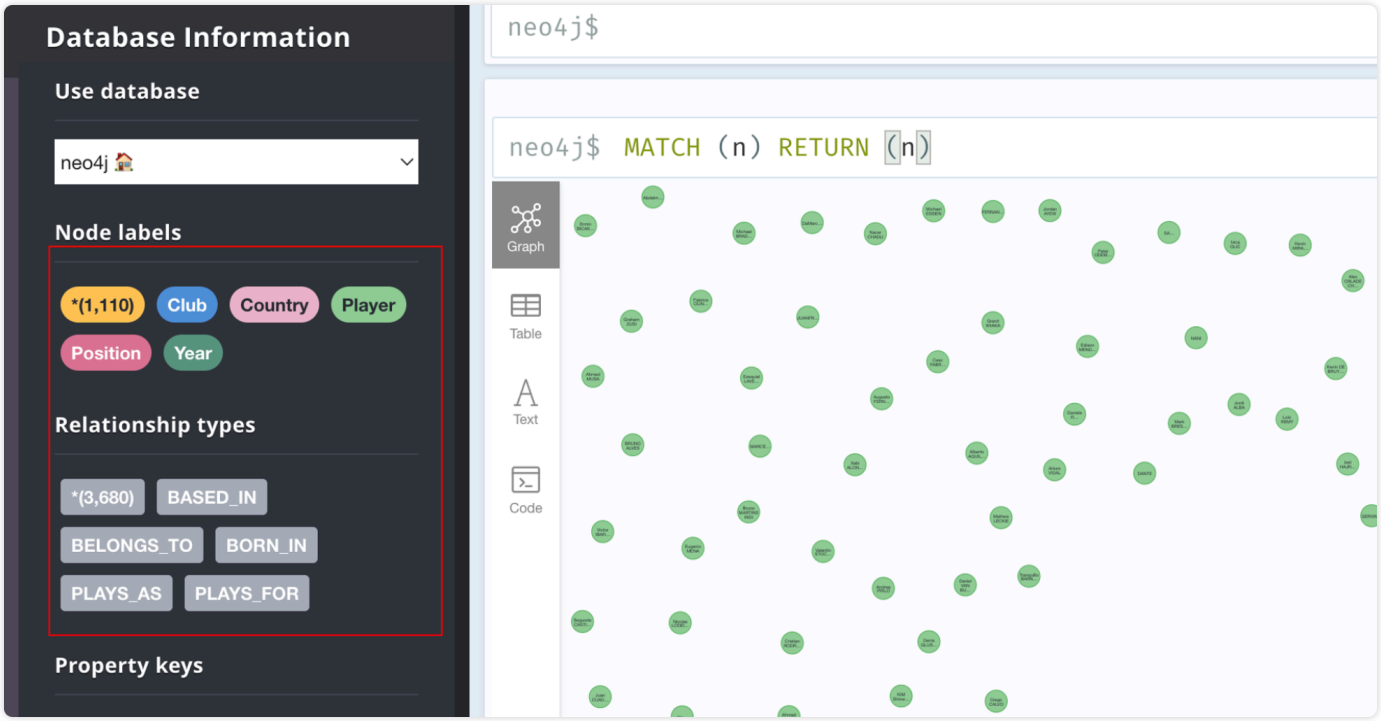


Scripts and Steps for ETL

```
import pandas as pd
# load CSV files into pandas DataFrames.
fifa2014_table = pd.read_csv('./FIFA2014 - all players.csv')
fifa2014_table.head()

# Create node csvs
# Node: Player
# Keeping only specified columns using loc method
player_table = fifa2014_table.loc[:, ['Player id', 'Player', 'Number', 'D.O.B', 'Age',
'Height (cm)', 'Caps', 'International goals', 'Plays in home country?']]
# export to csv
player_table.to_csv('./player.csv', index=False)
print(player_table.head())

# Node: Club
# Keeping only specified columns using loc method
club_table = fifa2014_table.loc[:, ['Club']]
# drop duplicates
club_table = club_table.drop_duplicates()
# export to csv
```

```
club_table.to_csv('./club.csv', index=False)
print(club_table.head())

# Node: Country
# Keeping only specified columns using loc method
country_a_table = fifa2014_table.loc[:, ['Club (country)']]
country_b_table = fifa2014_table.loc[:, ['Country']]
# Concatenating two DataFrames and drop duplicates
all_countries = pd.concat([country_a_table['Club (country)'],
country_b_table['Country']]).drop_duplicates()
all_countries_df = pd.DataFrame(all_countries, columns=['all countries'])
# export to csv
all_countries_df.to_csv('./country.csv', index=False)
print(all_countries_df.head())

# Node: Position
# Keeping only specified columns using loc method
position_table = fifa2014_table.loc[:, ['Position']]
# drop duplicates
position_table = position_table.drop_duplicates()
# export to csv
position_table.to_csv('./position.csv', index=False)
print(position_table.head())

# Node: Year of Birth
# Keeping only specified columns using loc method
year_of_birth_table = fifa2014_table.loc[:, ['D.O.B']]
# Extracting year from D.O.B. column
year_of_birth_table['D.O.B'] = year_of_birth_table['D.O.B'].str[-4:]
# drop duplicates
year_of_birth_table = year_of_birth_table.drop_duplicates()
# Rename column name
year_of_birth_table = year_of_birth_table.rename(columns={'D.O.B': 'Year of Birth'})
# export to csv
year_of_birth_table.to_csv('./year_of_birth.csv', index=False)
print(year_of_birth_table.head())

# Create relationship csvs
# The relationship between "player" and "country"
playerbelongsto_table = fifa2014_table.loc[:, ['Player id', 'Country']]
playerbelongsto_table.to_csv('./rel_playerbelongsto.csv', index=False)
print(playerbelongsto_table.head())

# The relationship between "player" and "club"
playsfor_table = fifa2014_table.loc[:, ['Player id', 'Club']]
playsfor_table.to_csv('./rel_playsfor.csv', index=False)
print(playsfor_table.head())

# The relationship between "club" and "Club (country)"
clubisfrom_table = fifa2014_table.loc[:, ['Club', 'Club (country)']]
clubisfrom_table.to_csv('./rel_clubisbasedin.csv', index=False)
print(clubisfrom_table.head())
```

```
# The relationship between "player" and "position"
playsas_table = fifa2014_table.loc[:, ['Player id', 'Position']]
playsas_table.to_csv('./rel_playsas.csv', index=False)
print(playsas_table.head())

# The relationship between "player" and "year of birth"
bornin_table = fifa2014_table.loc[:, ['Player id', 'D.O.B']]
# Extracting year from D.O.B. column
bornin_table['D.O.B'] = bornin_table['D.O.B'].str[-4:]
# Rename column to "Year of Birth"
bornin_table = bornin_table.rename(columns={'D.O.B': 'Year of Birth'})
# export to csv
bornin_table.to_csv('./rel_bornin.csv', index=False)
print(bornin_table.head())
```



Screenshots of Queries and Corresponding Results

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

```
MATCH (p:Player {playerId: '<specific player id>'})
RETURN p.number AS jerseyNumber;
```

> example: 336722

```
MATCH (p:Player {playerId: '336722'})
RETURN p.number AS jerseyNumber;
```

Table | Text | Code

	jerseyNumber
1	"11"

2. Which clubs are based in ?

```
MATCH (c:Club)-[:BASED_IN]->(co:Country {countryName: '<specific country>'})
RETURN c.clubName AS clubName;
```

example: Brazil

```
MATCH (c:Club)-[:BASED_IN]→(co:Country {countryName: 'Brazil'})
RETURN c.clubName AS clubName;

```

Table | Text | Code

	clubName
1	"Palmeiras"
2	"Fluminense FC"
3	"Santos FC"
4	"Atletico Mineiro"
5	"CR Flamengo"
6	"Botafogo FR"

3. Which club does play for?

```
MATCH (p:Player {playerName: '<specific player>'})-[:PLAYS_FOR]->(c:Club)
RETURN c.clubName AS clubName;
```

example: Lionel MESSI

```
MATCH (p:Player {playerName: 'Lionel MESSI'})-[:PLAYS_FOR]→
(c:Club)
RETURN c.clubName AS clubName;
```

clubName
"FC Barcelona"

4. How old is ?

```
MATCH (p:Player {playerName: '<specific player>'})
RETURN p.age AS age;
```

example: Lionel MESSI

```
MATCH (p:Player {playerName: 'Lionel MESSI'})
RETURN p.age AS age;
```

age
26

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

```
MATCH (p:Player {playerName: '<specific player>'})-[:PLAYS_FOR]->(c:Club)-[:BASED_IN]->
(co:Country)
RETURN DISTINCT co.countryName AS countryName;
```

example: Lionel MESSI

```
MATCH (p:Player {playerName: 'Lionel MESSI'})-[:PLAYS_FOR]→(c:Club)-[:BASED_IN]→(co:Country)
RETURN DISTINCT co.countryName AS countryName;
```

Table | Text | Code

	countryName
1	"Spain"

6. Find a club that has players from .

```
MATCH (p:Player)-[:BELONGS_TO]->(co:Country {countryName: '<specific country>'})
MATCH (p)-[:PLAYS_FOR]->(c:Club)
RETURN DISTINCT c.clubName AS clubName;
```

example: Spain

```
MATCH (p:Player)-[:BELONGS_TO]→(co:Country {countryName: 'Spain'})
MATCH (p)-[:PLAYS_FOR]→(c:Club)
RETURN DISTINCT c.clubName AS clubName;
```

Table | Text | Code

	clubName
1	"Atletico Madrid"
2	"SSC Napoli"
3	"Chelsea FC"
4	"Manchester City FC"
5	"FC Barcelona"
6	"Arsenal FC"
7	

Started streaming 9 records after 6 ms and completed after 7 ms.

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

```
MATCH (p:Player)-[:PLAYS_FOR]->(c:Club {clubName: '<specific club>'})
RETURN p.playerName AS playerName, p.age AS age
ORDER BY p.age ASC;
```

example: FC Barcelona

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

Table | Text | Code

	playerName	age
1	"NEYMAR"	22
2	"Jordi ALBA"	25
3	"Alexis SANCHEZ"	25
4	"Sergio BUSQUETS"	25
5	"Alexandre SONG"	26
6	"Lionel MESSI"	26
7		

Started streaming 13 records after 6 ms and completed after 8 ms.

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

```
MATCH (p:Player)-[:PLAYS_AS]->(pos:Position {positionName: '<specific position>'})
MATCH (p)-[:BELONGS_TO]->(co:Country {countryName: '<specific country>'})
RETURN p.playerName AS playerName, p.caps AS caps
ORDER BY p.caps DESC;
```

example: Forward - Argentina

```
MATCH (p:Player)-[:PLAYS_AS]→(pos:Position {positionName:
'Forward'})
MATCH (p)-[:BELONGS_TO]→(co:Country {countryName: 'Argentina'})
RETURN p.playerName AS playerName, p.caps AS caps
ORDER BY p.caps DESC;
```

Table | Text | Code

	playerName	caps
1	"Lionel MESSI"	84
2	"Sergio AGUERO"	50
3	"Gonzalo HIGUAIN"	36
4	"Ezequiel LAVEZZI"	29
5	"Rodrigo PALACIO"	21

Started streaming 5 records after 9 ms and completed after 12 ms.

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

```
MATCH (p:Player)-[:BORN_IN]->(y:Year {yearOfBirth: '<specific year>'})
MATCH (p)-[:BELONGS_TO]->(co:Country {countryName: '<specific country>'})
RETURN p.playerName AS playerName, p.caps AS caps
ORDER BY p.caps DESC;
```

example: 1987 - Argentina

```
MATCH (p:Player)-[:BORN_IN]→(y:Year {yearOfBirth: '1987'})
MATCH (p)-[:BELONGS_TO]→(co:Country {countryName: 'Argentina'})
RETURN p.playerName AS playerName, p.caps AS caps
ORDER BY p.caps DESC;
```

Table | Text | Code

	playerName	caps
1	"Lionel MESSI"	84
2	"Sergio ROMERO"	45
3	"Gonzalo HIGUAIN"	36

Started streaming 3 records after 9 ms and completed after 12 ms.

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

```
MATCH (p:Player)-[:BELONGS_TO]->(co:Country {countryName: '<specific country>'})
MATCH (p)-[:PLAYS_FOR]->(c:Club)
WITH c, p
ORDER BY p.internationalGoals DESC
WITH c.clubName AS clubName, collect(p.playerName) AS players,
collect(p.internationalGoals) AS goals
RETURN clubName, players, goals;
```

example: Argentina

```
MATCH (p:Player)-[:BELONGS_TO]->(co:Country {countryName: 'Argentina'})
MATCH (p)-[:PLAYS_FOR]->(c:Club)
WITH c, p
ORDER BY p.internationalGoals DESC
WITH c.clubName AS clubName, collect(p.playerName) AS players, collect(p.internationalGoals)
AS goals
RETURN clubName, players, goals;
```

Table | Text | Code

	clubName	players	goals
1	"FC Barcelona"	["Lionel MESSI", "Javier MASCHERANO"]	[37, 2]
2	"Manchester City FC"	["Sergio AGUERO", "Martin DEMICHELIS", "Pablo ZABALETA"]	[21, 2, 0]
3	"SSC Napoli"	["Gonzalo HIGUAIN", "Federico FERNANDEZ"]	[20, 2]
4	"CA Newells Old Boys"	["Maxi RODRIGUEZ"]	[15]
5	"Real Madrid CF"	["Angel DI MARIA"]	[9]
6	"Paris Saint-Germain FC"	["Ezequiel LAVEZZI"]	[4]
7			

Started streaming 15 records in less than 1 ms and completed after 3 ms.

11. Count how many players are born in .

```
MATCH (p:Player)-[:BORN_IN]->(y:Year {yearOfBirth: '<specific year>'})
RETURN COUNT(p) AS numberOfPlayers;
```

example: 1985

```
MATCH (p:Player)-[:BORN_IN]->(y:Year {yearOfBirth: '1985'})
RETURN COUNT(p) AS numberOfPlayers;
```

Table | Text | Code

	numberOfPlayers
1	66

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

```
MATCH (p:Player)
RETURN p.age AS age, COUNT(*) AS participation
ORDER BY participation DESC
LIMIT 1;
```

```
MATCH (p:Player)
RETURN p.age AS age, COUNT(*) AS participation
ORDER BY participation DESC
LIMIT 1;
```

Table | Text | Code

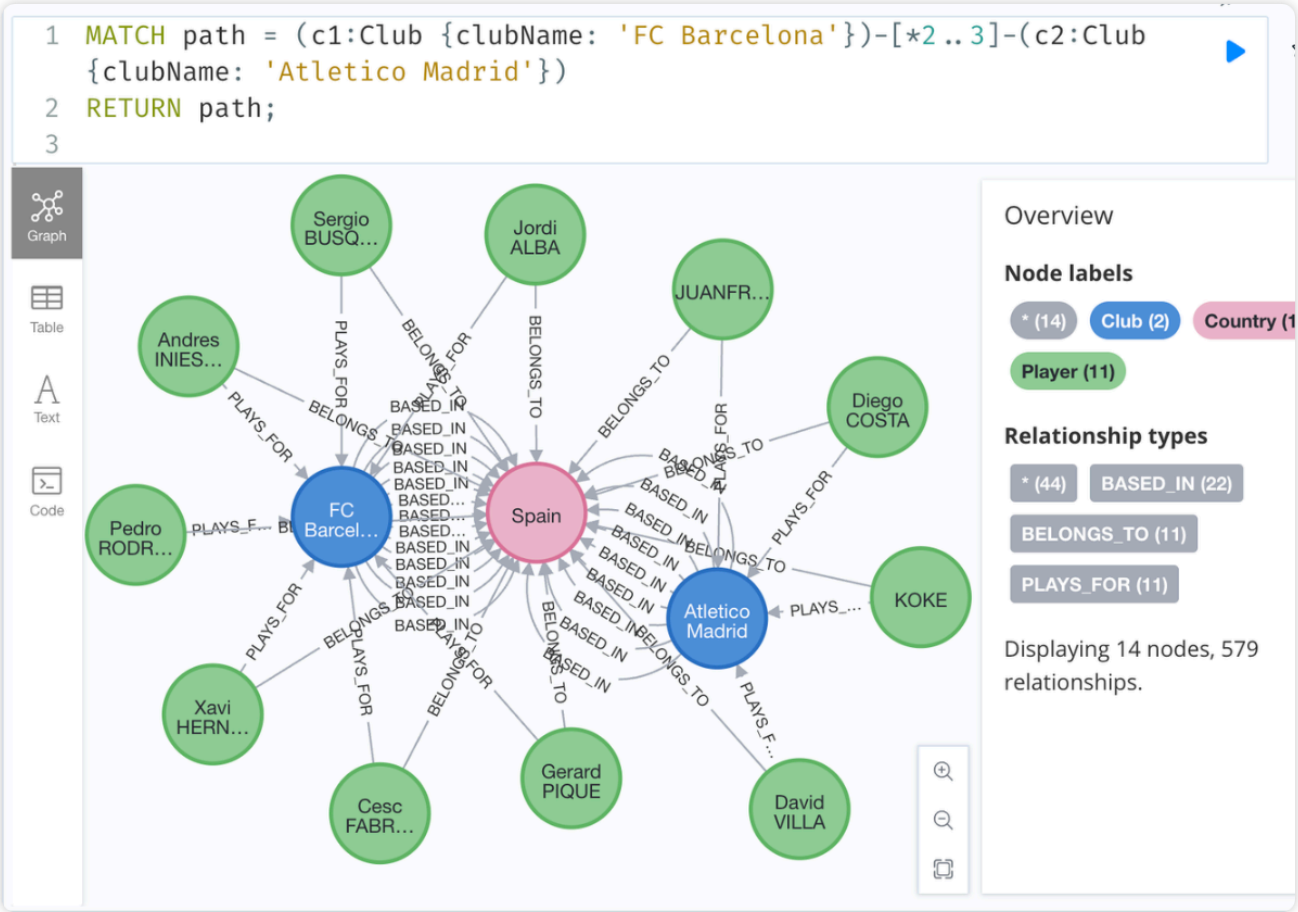
	age	participation
1	27	77

13. Find the path with a length of 2 or 3 between .

```
MATCH path = (c1:Club {clubName: '<club1>'})-[*2..3]-(c2:Club {clubName: '<club2>'})
RETURN path;
```

example: FC Barcelona - Atletico Madrid

```
MATCH path = (c1:Club {clubName: 'FC Barcelona'})-[*2..3]-(c2:Club
{clubName: 'Atletico Madrid'})
RETURN path;
```



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.

```
MATCH (p:Player)-[:BELONGS_TO]→(c:Country)
WITH c.countryName AS country, AVG(p.internationalGoals) AS
avgGoals
RETURN country, avgGoals
ORDER BY avgGoals DESC
LIMIT 5;
```

Table | Text | Code

	country	avgGoals
1	"Germany"	9.521739130434781
2	"Spain"	9.434782608695652
3	"Netherlands"	7.0
4	"Uruguay"	6.217391304347826
5	"Portugal"	6.130434782608695

Started streaming 5 records in less than 1 ms and completed after 4 ms.

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.

```
MATCH (p1:Player)-[:BELONGS_TO]->(c:Country)<-[:BELONGS_TO]-(p2:Player),
      (p1)-[:PLAYS_AS]->(pos1:Position),
      (p2)-[:PLAYS_AS]->(pos2:Position)
WHERE p1.playerId < p2.playerId AND pos1.positionName <> pos2.positionName
WITH p1, p2, pos1, pos2, abs(p1.caps - p2.caps) AS capsDifference
ORDER BY capsDifference ASC
RETURN p1.playerName AS Player1, pos1.positionName AS Position1, p2.playerName AS Player2,
pos2.positionName AS Position2, capsDifference
LIMIT 10;
```

```
MATCH (p1:Player)-[:BELONGS_TO]→(c:Country)←[:BELONGS_TO]-(p2:Player),
      (p1)-[:PLAYS_AS]→(pos1:Position),
      (p2)-[:PLAYS_AS]→(pos2:Position)
WHERE p1.playerId < p2.playerId AND pos1.positionName <> pos2.positionName
WITH p1, p2, pos1, pos2, abs(p1.caps - p2.caps) AS capsDifference
ORDER BY capsDifference ASC
RETURN p1.playerName AS Player1, pos1.positionName AS Position1, p2.playerName AS Player2,
pos2.positionName AS Position2, capsDifference
LIMIT 10;
```

Table | Text | Code

	Player1	Position1	Player2	Position2	capsDifference
1	"Diego REYES"	"Defender"	"Marco FABIAN"	"Midfielder"	0
2	"Alfredo TALAVERA"	"Goalkeeper"	"Marco FABIAN"	"Midfielder"	0
3	"Hector HERRERA"	"Midfielder"	"Miguel LAYUN"	"Defender"	0
4	"VIEIRINHA"	"Forward"	"BETO"	"Goalkeeper"	0
5	"ANDRE ALMEIDA"	"Defender"	"WILLIAM"	"Midfielder"	0
6	"RAUL MEIRELES"	"Midfielder"	"NANI"	"Forward"	0
7					

Started streaming 10 records after 22 ms and completed after 77 ms.

16.1 Identify the top 10 players (position ≠ 'Goalkeeper') with the most caps. List their names, position, caps, international goals and their club names, and a new column to calculate the goals per cap. Sort the results by the descending order of caps.

```
MATCH (p:Player)-[:PLAYS_AS]->(pos:Position),
      (p)-[:PLAYS_FOR]->(c:Club)
WHERE pos.positionName <> 'Goalkeeper'
RETURN p.playerName AS Player,
       pos.positionName AS Position,
       p.caps AS Caps,
       p.internationalGoals AS InternationalGoals,
       c.clubName AS Club,
       p.internationalGoals * 1.0 / p.caps AS GoalsPerCap
ORDER BY p.caps DESC
LIMIT 10;
```

```
MATCH (p:Player)-[:PLAYS_AS]→(pos:Position),
      (p)-[:PLAYS_FOR]→(c:Club)
WHERE pos.positionName <> 'Goalkeeper'
RETURN p.playerName AS Player,
       pos.positionName AS Position,
       p.caps AS Caps,
       p.internationalGoals AS InternationalGoals,
       c.clubName AS Club,
       p.internationalGoals * 1.0 / p.caps AS GoalsPerCap
ORDER BY p.caps DESC
LIMIT 10;
```

Table | Text | Code

	Player	Position	Caps	InternationalGoals	Club	GoalsPerCap
1	"Yasuhito ENDO"	"Midfielder"	143	12	"Gamba Osaka"	0.08391608391608392
2	"Javad NEKOUNAM"	"Midfielder"	137	37	"Kuwait SC"	0.27007299270072993
3	"Georgios KARAGOUNIS"	"Midfielder"	134	10	"Fulham FC"	0.07462686567164178
4	"Miroslav KLOSE"	"Forward"	131	68	"SS Lazio"	0.5190839694656488
5	"Xavi HERNANDEZ"	"Midfielder"	131	12	"FC Barcelona"	0.0916030534351145
6	"Carlos SALCIDO"	"Defender"	120	10	"Tigres UANL"	0.08333333333333333
7						

Started streaming 10 records after 18 ms and completed after 21 ms.

16.2 Identify the top 10 players (position: “Forward”) with the most goals and their club names and sort the results based on the descending order of their goals.

```
MATCH (p:Player)-[:PLAYS_AS]->(pos:Position {positionName: "Forward"}),
      (p)-[:PLAYS_FOR]->(c:Club)
RETURN p.playerName AS Player,
       p.internationalGoals AS InternationalGoals,
       c.clubName AS Club
ORDER BY p.internationalGoals DESC
LIMIT 10;
```

```
MATCH (p:Player)-[:PLAYS_AS]→(pos:Position {positionName: "Forward"}),
      (p)-[:PLAYS_FOR]→(c:Club)
RETURN p.playerName AS Player,
       p.internationalGoals AS InternationalGoals,
       c.clubName AS Club
ORDER BY p.internationalGoals DESC
LIMIT 10;
```

Table | Text | Code

	Player	InternationalGoals	Club
1	"Miroslav KLOSE"	68	"SS Lazio"
2	"Didier DROGBA"	61	"Galatasaray SK"
3	"David VILLA"	56	"Atletico Madrid"
4	"Samuel ETOO"	54	"Chelsea FC"
5	"CRISTIANO RONALDO"	49	"Real Madrid CF"
6	"Lukas PODOLSKI"	46	"Arsenal FC"
7			

Started streaming 10 records after 12 ms and completed after 14 ms.

16.3 Identify the top 5 clubs that has the highest number of young players (age ≤ 25). Provide such clubs along with their respective country names, sorting the results based on the ascending order of the number of young players.

```
MATCH (p:Player)-[:PLAYS_FOR]->(c:Club),
      (c)-[:BASED_IN]->(co:Country)
WHERE p.age <= 25
WITH c, co, COUNT(p) AS NumberOfYoungPlayers
ORDER BY NumberOfYoungPlayers DESC
LIMIT 5
RETURN c.clubName AS Club, co.countryName AS Country, NumberOfYoungPlayers
```

```
MATCH (p:Player)-[:PLAYS_FOR]→(c:Club),
      (c)-[:BASED_IN]→(co:Country)
WHERE p.age ⩽ 25
WITH c, co, COUNT(p) AS NumberOfYoungPlayers
ORDER BY NumberOfYoungPlayers DESC
LIMIT 5
RETURN c.clubName AS Club, co.countryName AS Country, NumberOfYoungPlayers
```

Table | Text | Code

	Club	Country	NumberOfYoungPlayers
1	"FC Bayern Muenchen"	"Germany"	105
2	"Manchester United FC"	"England"	84
3	"Chelsea FC"	"England"	60
4	"FC Barcelona"	"Spain"	52
5	"Liverpool FC"	"England"	50

Started streaming 5 records in less than 1 ms and completed after 7 ms.

Discussion 1

Discuss the capabilities of graph databases compared to relational databases. Highlight what can be achieved in graph databases that is challenging or not feasible in relational databases.

Graph Databases and Relational Databases differ significantly in the way data is stored, queried, and manipulated. Relational databases use tables to store data, typically in two dimensions. Queries are performed using the SQL language, with table-to-table relationships implemented through foreign keys, and complex queries requiring multiple table join operations (Vicknair et al., 2010). Graph databases use nodes, edges, and properties to represent and store data. The data model of graph databases is more flexible, allowing for the dynamic addition of nodes and edges. Commonly used query languages include Cypher, which supports graph traversal and pattern matching (Jouili and Vansteenberghe, 2013; Vicknair et al., 2010).

Graph databases excel at handling complex, interconnected data and can efficiently manage intricate relationships and hierarchical structures, especially for multi-level, multi-hop relational queries. For instance, in applications such as social network analysis, recommender systems, and network topology analysis, graph databases can perform rapid relational traversals and queries directly through nodes and edges. In contrast, relational databases require complex multi-table joins and nested queries for similar operations (Vicknair et al., 2010).

Additionally, graph databases support a dynamic schema, allowing new nodes and edges to be added at runtime without altering the existing data model. This feature is advantageous for applications requiring frequent adjustments and extensions to the data structure. In relational databases, modifying the schema typically necessitates database migrations and complex schema update operations, which can impact system availability and performance (Jouili and Vansteenberghe, 2013).

Graph databases also facilitate the graphical visualization of data, enabling users to intuitively understand the data by displaying the relationships between nodes and edges through a graphical interface. Conversely, relational databases usually present data in tabular form, making it challenging to directly represent complex relationships and structures (Vicknair et al., 2010).

Reference

Jouili, S. and Vansteenberghe, V., 2013. An empirical comparison of graph databases. Eura Nova R&D, Universite Catholique de Louvain.

Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y. and Wilkins, D., 2010. A comparison of a graph database and a relational database: A data provenance perspective. Department of Computer and Information Science, University of Mississippi.

Discussion 2

Discuss how Graph Data Science can be applied to at least one practical application.

A prominent application area of graph data science is social network analytics, especially on social media platforms like Facebook. Facebook generates a large amount of data every day that can be represented as a graph with users as nodes and user interactions as edges. This graph-based representation allows graph data science to dig deeper into complex relationships and provide important insights.

Community detection is a key application of graph data science in Facebook. Community detection algorithms, such as the Louvain algorithm and the Girvan-Newman algorithm, are effective in identifying tightly knit groups of users. The Louvain algorithm discovers communities by optimizing the degree of modularity, while the Girvan-Newman algorithm separates communities by removing edges with high betweenness centrality. By applying these algorithms, Facebook can identify a user's circle of friends to push content and advertisements more accurately, improving user experience and advertising effectiveness (Anandhan et al., 2016).

Influence analysis is another important application aimed at identifying users who have significant influence on information dissemination. PageRank algorithms and various centrality measures (e.g., degree centrality, closeness centrality, and betweenness centrality) are commonly used. PageRank algorithms measure the influence of a user by evaluating the importance of the node, which can help brands and businesses to increase their visibility and market influence (Vicknair et al., 2010).

The recommendation system is also an important feature of Facebook that benefits from graph data science. Personalized recommendations can be generated by analyzing relationship and behavioral data between users. Graph embedding algorithms such as Node2Vec and DeepWalk transform the graph structure into a low-dimensional vector space that preserves the relationship information, thus enabling similarity computation between users and items, recommending friends, groups, and content that may be of interest to users, and improving their social experience and platform activity (Jouili and Vansteenberghe, 2013).

In addition, graph data science plays an important role in Facebook's privacy and security assurances. The Graph Model Inversion (GraphMI) attack demonstrates the potential risks of Graph Neural Networks (GNNs) in terms of privacy breaches. GraphMI is able to reconstruct the graph structure in the training data, revealing sensitive relationships between users. Understanding these risks can help Facebook implement techniques such as differential privacy to protect the privacy of user data while maintaining service quality (Zhang et al., 2021).

Reference

Anandhan, P., Uma, K., & Anuradha, J. (2016). An overview of application of graph theory. International Journal of ChemTech Research, 9(2), 242-248.

Jouili, S. and Vansteenberghe, V. (2013). An empirical comparison of graph databases. Eura Nova R&D, Universite Catholique de Louvain.

Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y. and Wilkins, D. (2010). A comparison of a graph database and a relational database: A data provenance perspective. Department of Computer and Information Science, University of Mississippi.

Zhang, Z., Liu, Q., Huang, Z., Wang, H., Lu, C., Liu, C., & Chen, E. (2021). GraphMI: Extracting private graph data from graph neural networks. arXiv preprint arXiv:2106.02820.