Homework 3 (Neo4J)

Football Data from Transfermarkt

Original Dataset from Kaggle

The dataset is composed of multiple CSV files with information on competitions, games, clubs, players and appearances that is automatically updated once a week. It includes:

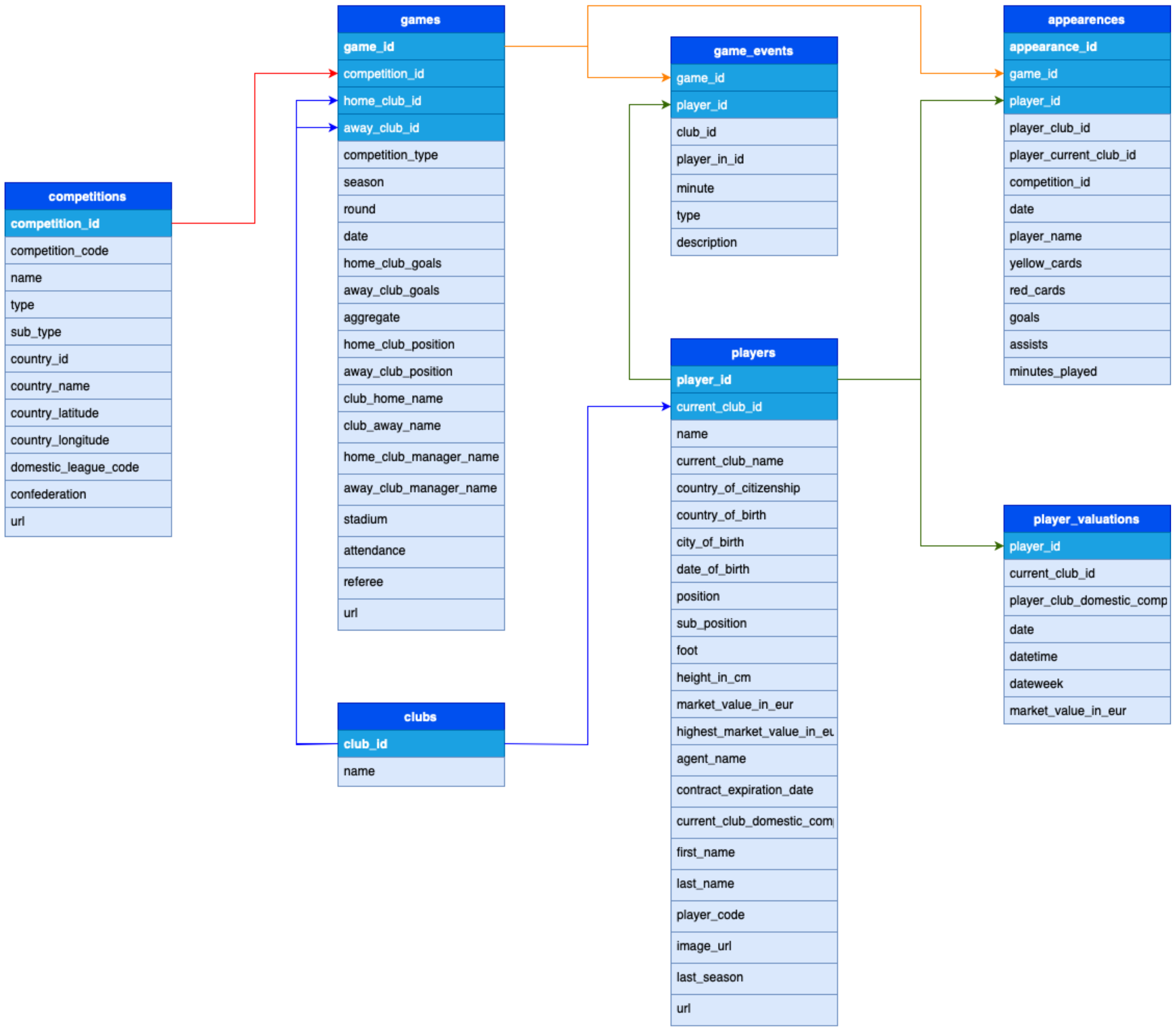
- 60.000+ games from many seasons on all major competitions
- 400+ clubs from those competitions
- 28.000+ players from those clubs
- 300.000+ player market valuations historical records
- 1.000.000+ player appearance records from all games

Football Data from Transfermarkt

Football (Soccer) data scraped from Transfermarkt website

https://www.kaggle.com/datasets/davidcariboo/player-scores

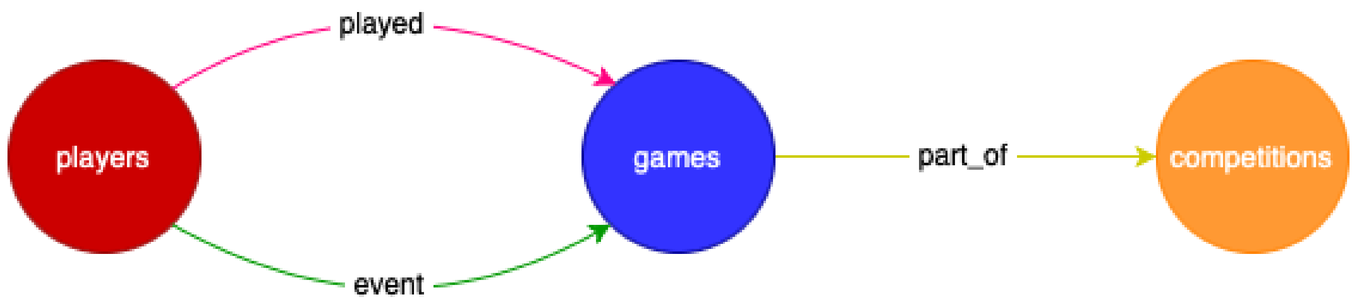
Previous Tables and Relationships (HW1)



competitions
competition_id
competition_code
name
type
sub_type
country_id
country_name
country_latitude
country_longitude
domestic_league_code
confederation
url
games
game_id
competition_id
home_club_id
away_club_id
competition_type
season
round
date
home_club_goals
away_club_goals
aggregate
home_club_position
away_club_position
club_home_name
club_away_name
home_club_manager_name
away_club_manager_name
stadium
attendance
referee
url
clubs
club_id
name
game_events
game_id
player_id
club_id
player_in_id
minute
type
description
players
player_id
current_club_id
name
current_club_name
country_of_citizenship
country_of_birth
city_of_birth
date_of_birth
position
sub_position
foot
height_in_cm
market_value_in_eur
highest_market_value_in_eu
agent_name
contract_expiration_date
current_club_domestic_com
first_name
last_name
player_code
image_url
last_season
url
appearences
appearance_id
game_id
player_id
player_club_id
player_current_club_id
competition_id
date
player_name
yellow_cards
red_cards
goals
assists
minutes_played
player_valuations
player_id
current_club_id
player_club_domestic_comp
date
datetime
dateweek
market_value_in_eur

Preprocessing

To fully exploit the potential of a graph dataset, we decided to preprocess our data by creating 3 types of nodes and 3 types of edges. Specifically, the structure will be as follows:



Nodes

competitions

- competition_id
- name

players

- player_id
- name

games

- game_id
- club_home_name

- type
- sub_type
- country_name
- country_latitude
- country_longitude
- domestic_league_code
- confederation

- current_club_name
- country_of_citizenship
- country_of_birth
- city_of_birth
- date_of_birth
- position
- sub_position
- foot
- height_in_cm
- market_value_in_eur
- highest_market_value_in_eur
- agent_name
- contract_expiration_date
- first_name
- last_name
- player_code
- last_season

- club_away_name
- home_club_goals
- away_club_goals
- aggregate
- home_club_position
- away_club_position
- home_club_manager_name
- away_club_manager_name
- stadium
- attendance
- referee

Edges

played (players —> games)

- date
- competition_id
- yellow_cards
- red_cards
- goals
- assists
- minutes_played
- club

event (players —> games)

- minute
- type
- description

part_of (games —> competitions)

- season
- round
- date

Python code to create the new csv files

- Read all the csv used for HW2

```
appearances = pd.read_csv('old_csv/appearances_cleaned.csv')
clubs = pd.read_csv('old_csv/clubs_v2.csv')
competitions = pd.read_csv('old_csv/competitions.csv')
game_events = pd.read_csv('old_csv/game_events.csv')
```

```
games = pd.read_csv('old_csv/games_cleaned.csv')
players = pd.read_csv('old_csv/players_v2.csv')
```

- Create a csv file to represent nodes of type **'players'**

```
players['name'].fillna(players['player_id'], inplace=True)
players_HW3 = players[['player_id', 'name', 'current_club_name', 'country_of_citizensh
                       'date_of_birth', 'position', 'sub_position', 'foot', 'height_in
                       'highest_market_value_in_eur', 'agent_name', 'contract_expirati
                       'last_name', 'player_code', 'last_season']]
display(players_HW3.head())
players_HW3.to_csv('new_csv/players.csv', encoding='utf-8', index=False)
```

- Create a csv file to represent nodes of type **'games'**

```
games['club_home_name'].fillna(games['home_club_id'], inplace=True)
games['club_away_name'].fillna(games['away_club_id'], inplace=True)
games_HW3 = games[['game_id', 'club_home_name', 'club_away_name', 'home_club_goals', '
                   'aggregate', 'home_club_position', 'away_club_position', 'home_club
                   'away_club_manager_name', 'stadium', 'attendance', 'referee']]
display(games_HW3.head())
games_HW3.to_csv('new_csv/games.csv', encoding='utf-8', index=False)
```

- Create a csv file to represent nodes of type **'competitions'**

```
competitions_HW3 = competitions[['competition_id', 'name', 'type', 'sub_type', 'countr
                                 'country_longitude', 'domestic_league_code', 'confede
display(competitions_HW3.head())
competitions_HW3.to_csv('new_csv/competitions.csv', encoding='utf-8', index=False)
```

- Create a csv file to represent edges of type **'played'**

```
clubs['name'].fillna(clubs['club_id'], inplace=True)
played = appearances[['game_id', 'player_id', 'player_club_id', 'date', 'competition_i
                      'yellow_cards', 'red_cards', 'goals', 'assists', 'minutes_playec
played = played.merge(clubs, left_on='player_club_id', right_on='club_id', how='left')
played = played.rename(columns={'name': 'club'})
played.drop(['player_club_id', 'club_id'], axis=1, inplace=True)
display(played.head())
print(played.shape)
played.to_csv('new_csv/played.csv', encoding='utf-8', index=False)
```

- Create a csv file to represent edges of type **'event'**

```
event = game_events[['game_id', 'player_id', 'minute', 'type', 'description']]
display(event.head())
print(event.shape)
event.to_csv('new_csv/event.csv', encoding='utf-8', index=False)
```

- Create a csv file to represent edges of type **'part_of'**

```
part_of = games[['game_id', 'competition_id', 'season', 'round', 'date']]
display(part_of.head())
print(part_of.shape)
part_of.to_csv('new_csv/part_of.csv', encoding='utf-8', index=False)
```

Cypher code to create the dataset

- Create **‘competitions’** nodes

```
LOAD CSV WITH HEADERS FROM 'file:///competitions.csv' AS f
CREATE (:competitions {competition_id:f.competition_id, name:f.name, type:f.type, sub_
 country_name:f.country_name, country_latitude:f.country_latitude, country_longitude:f
 domestic_league_code:f.domestic_league_code, confederation:f.confederation});
```

- Create **‘players’** nodes

```
LOAD CSV WITH HEADERS FROM 'file:///players.csv' AS f
CREATE (:players {player_id:f.player_id, name:f.name, current_club_name:f.current_club
 country_of_citizenship:f.country_of_citizenship, country_of_birth:f.country_of_birth,
 city_of_birth:f.city_of_birth, date_of_birth:f.date_of_birth, position:f.position, su
 foot:f.foot, height_in_cm:f.height_in_cm, market_value_in_eur:f.market_value_in_eur,
 highest_market_value_in_eur:f.highest_market_value_in_eur, agent_name:f.agent_name,
 contract_expiration_date:f.contract_expiration_date, first_name:f.first_name, last_na
 player_code:f.player_code, last_season:f.last_season});
```

- Create **‘games’** nodes

```
LOAD CSV WITH HEADERS FROM 'file:///games.csv' AS f
CREATE (:games {game_id:f.game_id, club_home_name:f.club_home_name, club_away_name:f.c
 home_club_goals:f.home_club_goals, away_club_goals:f.away_club_goals, aggregate:f.agg
 home_club_position:f.home_club_position, away_club_position:f.away_club_position,
 home_club_manager_name:f.home_club_manager_name, away_club_manager_name:f.away_club_m
 stadium:f.stadium, attendance:f.attendance, referee:f.referee});
```

- Create **‘played’** edges

```
CALL apoc.periodic.iterate(
    'LOAD CSV WITH HEADERS FROM "file:///played.csv" AS f
     RETURN f',
    'MATCH (p:players {player_id:f.player_id}), (g:games {game_id:f.game_id})
    CREATE (p)-[:played {date:f.date, competition_id:f.competition_id, yellow_cards:f.
     red_cards:f.red_cards, goals:f.goals, assists:f.assists, minutes_played:f.minutes
 {batchSize:1000, parallel:true});
```

- Create **‘event’** edges

```
CALL apoc.periodic.iterate(
    'LOAD CSV WITH HEADERS FROM "file:///event.csv" AS f
     RETURN f',
    'MATCH (p:players {player_id:f.player_id}), (g:games {game_id:f.game_id})
    CREATE (p)-[:event {minute:f.minute, type:f.type, description:f.description}]->(g
 {batchSize:1000, parallel:true});
```

- Create **'part_of'** edges

```
CALL apoc.periodic.iterate(
    'LOAD CSV WITH HEADERS FROM "file:///part_of.csv" AS f
     RETURN f',
    'MATCH (g:games {game_id:f.game_id}), (c:competitions {competition_id:f.competitio
    CREATE (g)-[:part_of {season:f.season, round:f.round, date:f.date}]->(c)',
 {batchSize:5000, parallel:true});
```

Query

1. Name, market value and number of goals scored with the head in 'Champions League' or 'Europa League' or 'Club World Cup' from defenders.

```
MATCH (p:players)-[played:played]->(g:games)
MATCH (p)-[e:event]->(g)
WHERE played.competition_id IN ['CL', 'EL', 'KLUB']
AND p.position = 'Defender'
AND toInteger(g.attendance) > 10000
AND e.description CONTAINS 'Header'
RETURN p.name AS Name,
toInteger(p.highest_market_value_in_eur) AS max_market_value,
COUNT(*) AS num_goals
ORDER BY num_goals DESC;
```

2. Players (with the number of goals) who have scored the most goals after the 80th minute in Serie A in the 2021 season.

```
MATCH (p:players)-[event:event]->(g:games)-[part_of:part_of]->(c:competitions)
WHERE c.name = 'Serie A' AND part_of.season = '2021'
AND toInteger(event.minute) >= 80
AND event.type = 'Goals'
RETURN p.name AS name, COUNT(DISTINCT event) AS num_goals
ORDER BY num_goals DESC
LIMIT 10;
```

3. Top 10 defenders by number of yellow + red cards and how often they get them.

```
MATCH (p:players)-[played:played]->(:games)-[:part_of]->(c:competitions)
WHERE p.position = 'Defender'
WITH p, SUM(toInteger(played.yellow_cards)) AS yellows, SUM(toInteger(played.red_cards
         SUM(toInteger(played.minutes_played)) AS minutes_played
RETURN p.name AS Player, yellows, reds, (yellows + reds) AS total_cards, minutes_playe
             minutes_played/(yellows + reds+ 1) AS time_interval
ORDER BY total_cards DESC
LIMIT 10;
```

3. Top 10 referees by appearances in European competitions (CL, EL, CL and EL qualifiers and Club World Cup).

```
MATCH (g:games)-[:part_of]->(c:competitions)
WHERE c.competition_id IN ['USC', 'CL', 'EL', 'KLUB', 'ECLQ', 'CLQ', 'ELQ']
WITH DISTINCT g.game_id AS game_id, g.referee AS referee, c.competition_id AS competit
RETURN referee,
       COUNT(DISTINCT game_id) AS Total_Appearances,
       COUNT(DISTINCT CASE WHEN competition = 'CL' THEN game_id END) AS Champions_Leag
       COUNT(DISTINCT CASE WHEN competition = 'EL' THEN game_id END) AS Europa_League,
       COUNT(DISTINCT CASE WHEN competition = 'USC' THEN game_id END) AS Super_Cup,
       COUNT(DISTINCT CASE WHEN competition = 'KLUB' THEN game_id END) AS Club_World_C
       COUNT(DISTINCT CASE WHEN competition = 'ECLQ' THEN game_id END) AS Conference_L
       COUNT(DISTINCT CASE WHEN competition = 'CLQ' THEN game_id END) AS Champions_Lea
       COUNT(DISTINCT CASE WHEN competition = 'ELQ' THEN game_id END) AS Europa_League
```

```
ORDER BY Total_Appearances DESC
LIMIT 10;
```

5. The 10 games with the most spectators in the history of the Allianz stadium with the match info.

```
MATCH (g:games)-[part_of:part_of]->(c:competitions)
WHERE g.stadium = 'Allianz Stadium'
OPTIONAL MATCH (g)<-[e:event]-(p:players)
WHERE e.type = 'Goals'
WITH g,
     part_of,
     c,
     p,
     e
ORDER BY e.minute ASC
WITH g,
     part_of,
     c,
     COLLECT(p.name + ' (' + e.minute + ')') AS scorers
RETURN DISTINCT toInteger(g.attendance) AS attendance,
       toString(date(part_of.date)) + ' | ' + g.club_home_name + ' - ' +
           g.club_away_name + ' (' + g.aggregate + ')' AS match_result,
       c.name + ', ' + part_of.round AS match_info,
       g.referee AS referee,
       REDUCE(scorer = '', r IN scorers | CASE WHEN scorer = '' THEN r ELSE scorer +
ORDER BY attendance DESC
LIMIT 10;
```

6. Teams that have won the most matches in the Champions League with a difference of at least 3 goals.

```
MATCH (c:competitions {competition_id: 'CL'})<-[:part_of]-(g:games)
WHERE abs(toInteger(g.home_club_goals) - toInteger(g.away_club_goals)) >= 3
WITH CASE
WHEN g.home_club_goals > g.away_club_goals THEN g.club_home_name
ELSE g.club_away_name
END AS winning_team,
COUNT(DISTINCT g.game_id) AS num_wins
RETURN winning_team,
             num_wins
ORDER BY num_wins DESC
LIMIT 10;
```

7. Players (excluding English players) who scored the most goals and assists in Premier League between 2015 and 2020 in January.

```
MATCH (p:players)-[played:played]->(g:games)-[part_of:part_of]->(c:competitions)
WHERE c.competition_id = 'GB1'
  AND date(part_of.date) >= date('2015-01-01')
  AND date(part_of.date) <= date('2020-01-31')
  AND p.country_of_citizenship <> 'England'
  AND datetime(part_of.date).month = 1
WITH DISTINCT games.game_id AS distinct_game_id, p.name AS player_name, toInteger(play
                            toInteger(played.assists) AS assists
RETURN player_name,
       SUM(goals) + SUM(assists) AS total_score,
       SUM(goals) AS num_goals,
       SUM(assists) AS num_assists
ORDER BY total_score DESC
LIMIT 10;
```

8. Top 5 coaches (nemesis) who have won the most games against Mourinho.

```
MATCH (g:games)
WHERE (g.home_club_manager_name = 'Jose Mourinho' AND g.home_club_goals < g.away_club_
     OR (g.away_club_manager_name = 'Jose Mourinho' AND g.away_club_goals < g.home_clu
WITH CASE WHEN g.home_club_manager_name = 'Jose Mourinho' THEN g.away_club_manager_nan
                    ELSE g.home_club_manager_name END AS manager_name, COUNT(*) AS nun
RETURN manager_name,
             num_wins
ORDER BY num_wins DESC
LIMIT 5;
```

9. Agents/agencies sorted by value of assisting players.

```
MATCH (p:players)
WHERE p.agent_name <> 'null'
    AND toInteger(p.market_value_in_eur) > 0
WITH p.agent_name AS agent_name, SUM(toInteger(p.market_value_in_eur)) AS total_market
         COUNT(DISTINCT p) AS num_players,
         SUM(toInteger(p.market_value_in_eur))/COUNT(distinct(p)) AS mean_player_value
RETURN agent_name,
             total_market_value,
             num_players,
             mean_player_value
ORDER BY total_market_value DESC;
```

10. Left-footed players with height < 175cm that scored with the right foot in top European countries (sorted by the number of goals).

```
MATCH (p:players)
MATCH (p)-[event:event]->(g:games)-[:part_of]->(c:competitions)
WHERE p.foot = 'Left' AND toInteger(p.height_in_cm) < 175
    AND event.type = 'Goals'
    AND event.description CONTAINS 'Right-footed'
    AND c.country_name IN ['Italy', 'Spain', 'Germany', 'England', 'France']
RETURN  p.name AS player,
                toInteger(p.height_in_cm) AS height,
                COUNT(DISTINCT event) AS goals
ORDER BY goals DESC;
```