Transfermarkt: Neo4j graph and Cipher queries.

Project for Data Management course at Sapienza University of Rome.

Neo4j graphs application to a Transfermarkt dataset.

Candidate: Carmignani Federico 1845479

Neo4j installation in Python.

```
!pip3 install neo4j

    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/cola
    Requirement already satisfied: neo4j in /usr/local/lib/python3.7/dist-packa
    Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packag
```

Import of Graph databases driver and Pandas to manage data structures.

```
from neo4j import GraphDatabase
import pandas as pd
```

```
class Neo4jConnection:

    def __init__(self, uri, user, pwd):

        self.__uri = uri
        self.__user = user
        self.__pwd = pwd
        self.__driver = None

        try:
            self.__driver = GraphDatabase.driver(self.__uri, auth=(self._
        except Exception as e:
            print("Failed to create the driver:", e)

    def close(self):

        if self.__driver is not None:
            self.__driver.close()

    def query(self, query, parameters=None, db=None):

        assert self.__driver is not None, "Driver not initialized!"
        session = None
        response = None

        try:
            session = self.__driver.session(database=db) if db is not None
            response = list(session.run(query, parameters))
        except Exception as e:
            print("Query failed:", e)
        finally:
            if session is not None:
                session.close()
        return response
```

Connection creation with Neo4j.

```
uri = 'bolt://34.201.118.38:7687'
user = 'neo4j'
pwd = 'gasket-advertisement-fur'

conn = Neo4jConnection(uri=uri, user=user, pwd=pwd)
```

Let's see the first 10 rows of the CSV file.

```
query = """WITH 'https://s3-eu-west-1.amazonaws.com/football-transfers.neo4
LOAD CSV WITH HEADERS FROM url AS row
RETURN row
LIMIT 10"""

result = conn.query(query)
print(result)
#this is how to enter a record type returned by the queries
print("An example of a row: ",result[0][0]['playerUri'])
```

```
[<Record row={'playerUri': '/douglas-costa/profil/spieler/75615', 'playerIm
An example of a row:  /douglas-costa/profil/spieler/75615
```

Constraints: to avoid duplicates.

```
query = """CREATE CONSTRAINT ON (player:Player)
ASSERT player.id IS UNIQUE;"""
result = conn.query(query)
```

```
Query failed: {code: Neo.ClientError.Schema.EquivalentSchemaRuleAlreadyExis
```

```
query = """CREATE CONSTRAINT ON (club:Club)
ASSERT club.id IS UNIQUE;"""
result = conn.query(query)
```

```
Query failed: {code: Neo.ClientError.Schema.EquivalentSchemaRuleAlreadyExis
```

```
query = """CREATE CONSTRAINT ON (transfer:Transfer)
ASSERT transfer.id IS UNIQUE;"""
result = conn.query(query)
```

```
Query failed: {code: Neo.ClientError.Schema.EquivalentSchemaRuleAlreadyExis
```

```
query = """CREATE CONSTRAINT ON (country:Country)
ASSERT country.name IS UNIQUE;"""
result = conn.query(query)
```

```
Query failed: {code: Neo.ClientError.Schema.EquivalentSchemaRuleAlreadyExis
```

```
query = """CALL db.constraints()"""
result = conn.query(query)
print(len(result))
```

```
4
```

Import players.

```
query = """USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'https://s3-eu-west-1.amazonaws.com/football-tra
MERGE (player:Player {id: row.playerUri})
ON CREATE SET player.name =  row.playerName, player.position = row.playerPo
result = conn.query(query)
```

Take the players.

```
query = """MATCH (player:Player)
RETURN player
LIMIT 25"""

result = conn.query(query)
#print(result)
for i in range(0,25):
  print((result[i][0]).id, result[i][0]['name']) #for 'id'
   #and for 'name' in properties dictionary
```

```
0 Douglas Costa
1 Florent Sinama-Pongolle
2 Keisuke Honda
3 Alex Teixeira
4 Younès Kaboul
5 Alessandro Budel
6 Daniele Vantaggiato
7 Réver
8 Aleksandr Samedov
9 Éder Luís
10 Éverton
11 Christian Giménez
12 Mario Bolatti
13 Jonathan Pereira
14 Míchel
15 João Pereira
16 Felipe
17 Rúben Micael
18 Jackson Martínez
19 Airton
20 Dario Dainelli
21 Álex Geijo
22 Alexandru Epureanu
23 Alan Kardec
24 Sung-Yong Ki
```

Create countries.

```
query = """WITH 'https://s3-eu-west-1.amazonaws.com/football-transfers.neo4
LOAD CSV WITH HEADERS FROM url AS row
WITH row WHERE row.playerNationality <> ''
WITH DISTINCT row.playerNationality AS nationality
MERGE (country:Country {name: nationality })"""

result = conn.query(query)
```

Create relationships "FROM": a player comes from a specific country.

```
query = """USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'https://s3-eu-west-1.amazonaws.com/football-tra
WITH row WHERE row.playerNationality <> ''
MATCH (player:Player {id: row.playerUri})
MATCH (country:Country {name: row.playerNationality })
MERGE (player)-[:FROM]->(country)"""

result = conn.query(query)
```

Importing clubs: buying and selling clubs and also relationships "IN" with countries.

```
query = """WITH 'https://s3-eu-west-1.amazonaws.com/football-transfers.neo4
LOAD CSV WITH HEADERS FROM url AS row
UNWIND [
  {uri: row.sellerClubUri, name: row.sellerClubName, country: row.sellerClu
  {uri: row.buyerClubUri,  name: row.buyerClubName,  country: row.buyerClub
] AS club
WITH club WHERE club.uri <> ''
WITH DISTINCT club
MERGE (c:Club {id: club.uri})
ON CREATE SET c.name = club.name
MERGE (country:Country {name: club.country })
MERGE (c)-[:IN]->(country)"""

result = conn.query(query)
```

Importing transfers: each transfer relationship is linked to a player, a destination club and the former club.

```
query = """USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'https://s3-eu-west-1.amazonaws.com/football-tra
MATCH (player:Player {id: row.playerUri})
MATCH (source:Club {id: row.sellerClubUri})
MATCH (destination:Club {id: row.buyerClubUri})
MERGE (t:Transfer {id: row.transferUri})
ON CREATE SET t.season = row.season,
              t.fee = row.transferFee,
              t.timestamp = toInteger(row.timestamp)
MERGE (t)-[ofPlayer:OF_PLAYER]->(player) SET ofPlayer.age = row.playerAge
MERGE (t)-[:FROM_CLUB]->(source)
MERGE (t)-[:TO_CLUB]->(destination);"""

result = conn.query(query)
```

```
    Query failed: {code: Neo.ClientError.Transaction.TransactionTimedOut} {mess
```

Many strange values are inserted as fees so it is not possible to work with them, therefore now **data cleaning** is necessary.

```
query = """MATCH (transfer:Transfer)
RETURN transfer.fee, COUNT(*) AS occurrences
ORDER BY occurrences DESC
LIMIT 100"""

result = conn.query(query)
for i in range(0,100):
  print(result[i][0])
```

```
    Free transfer
    ?
    Loan
    -
    gratuito
    Swap deal
    £450k
    £90k
    £900k
    draft
    £270k
    £180k
    £1.35m
    £1.80m
    £225k
    £360k
    £45k
    £135k
```

```
£2.25m
£2.70m
£540k
None
£720k
£3.60m
£315k
£4.50m
Loan fee:£450k
£630k
£3.15m
£1.08m
Loan fee:£180k
£675k
Loan fee:£90k
£23k
Loan fee:£45k
Loan fee:£270k
£810k
£68k
£5.40m
£9k
£108k
£405k
£1.17m
ablosefrei
£1.62m
£54k
£7.20m
£27k
£4.05m
£990k
Loan fee:£900k
£18k
£72k
£6.30m
£1.44m
£113k
£1.98m
£36k
Loan fee:£225k
```

Now clean up (ETL data cleaning).

```
query = """MATCH (t:Transfer)
WHERE t.fee contains "?" or t.fee contains "-"
RETURN t.fee, count(*)"""

result = conn.query(query)
for i in result:
  print(i[0],i[1])
```

```
? 47798
- 26117
--- 1
?  4
 ? 2
-? 1
?ablösefrei 1
free? 1
```

Remove the ones containing '?' or "-" but keeping them as another type.

```
query = """MATCH (t:Transfer)
WHERE t.fee CONTAINS "?" or t.fee CONTAINS "-"
REMOVE t:Transfer
SET t:TransferWithoutFee"""
result = conn.query(query)
```

Add a one more type to the transfers: the loan.

```
query = """MATCH (t:Transfer)
WHERE t.fee STARTS WITH 'Loan'
SET t:Loan"""

result = conn.query(query)
```

Transfer fees as numeric values as a new property called "numericfee". The use of APOC functions https://neo4j.com/developer/neo4j-apoc/ for multiplication is because we are working with strings.

```
query = """MATCH (t:Transfer)
WITH t, replace(replace(replace(replace(t.fee, "k", ""), "m", ""), "Loan fe
WITH t,
CASE
 WHEN t.fee ENDS WITH "k" THEN toFloat(apoc.number.exact.mul(trim(rawNumeri
 WHEN t.fee ENDS WITH "m" THEN toFloat(apoc.number.exact.mul(trim(rawNumeri
 WHEN trim(t.fee) IN ["Free transfer", "ablösefrei ", "gratuito", "free", "
 WHEN NOT(exists(t.fee)) THEN 0
 WHEN rawNumeric = '' THEN 0
 ELSE toFloat(trim(rawNumeric))
END AS numericFee
SET t.numericFee = numericFee"""

result = conn.query(query)
```

Then it is about to eliminate the transfers having no numerical value.

```
query = """MATCH (t:Transfer)
WHERE not exists(t.numericFee)
REMOVE t:Transfer
SET t:TransferWithoutFee"""

result = conn.query(query)
```

QUERIES:

Top transfers.

```
query = """MATCH (transfer:Transfer)-[:OF_PLAYER]->(player),
      (from)<-[:FROM_CLUB]-(transfer)-[:TO_CLUB]->(to)
RETURN player.name, from.name, to.name, transfer.numericFee
ORDER BY transfer.numericFee DESC
LIMIT 3"""

result = conn.query(query)
for res in result:
  print(res[0],"from",res[1],"-->",res[2],"for",res[3])
```

```
Gareth Bale from Spurs --> Real Madrid for 90900000.0
Neymar from Santos FC --> FC Barcelona for 79380000.0
Luis Suárez from Liverpool --> FC Barcelona for 73550000.0
```

Transfers from team.

```
query = """MATCH (from:Club)<-[:FROM_CLUB]-(transfer:Transfer)-[:TO_CLUB]->
      (transfer)-[:OF_PLAYER]->(player)
WHERE from.name = "AS Roma"
RETURN player.name, to.name, transfer.numericFee, transfer.season
ORDER BY transfer.numericFee DESC"""

result = conn.query(query)
for res in result:
  print(res[0],"to",res[1],"for",res[2],"in",res[3])
```

```
Marquinhos to Paris SG for 28260000.0 in 2013/2014
Érik Lamela to Spurs for 27000000.0 in 2013/2014
Mirko Vucinic to Juventus for 13500000.0 in 2011/2012
Fabio Borini to Liverpool for 11970000.0 in 2012/2013
Bojan Krkic to FC Barcelona for 11700000.0 in 2013/2014
Jérémy Ménez to Paris SG for 7200000.0 in 2011/2012
Michael Bradley to Toronto FC for 6660000.0 in 2013/2014
Maarten Stekelenburg to Fulham for 5040000.0 in 2013/2014
Nicolás López to Udinese Calcio for 3600000.0 in 2013/2014
Alessio Cerci to Fiorentina for 3600000.0 in 2010/2011
Panagiotis Tachtsidis to Genoa for 2700000.0 in 2013/2014
Valerio Verre to Udinese Calcio for 2250000.0 in 2013/2014
Marco D'Alessandro to Atalanta for 1800000.0 in 2014/2015
Gianluca Caprari to Pescara for 1580000.0 in 2013/2014
Giammario Piscitella to Pescara for 1350000.0 in 2013/2014
Giammario Piscitella to Genoa for 1350000.0 in 2012/2013
Valerio Verre to Genoa for 1350000.0 in 2012/2013
Stefano Guberti to Sampdoria for 1350000.0 in 2010/2011
Marco Motta to Udinese Calcio for 1310000.0 in 2010/2011
Dodô to Inter for 1080000.0 in 2014/2015
Andrea Bertolacci to Genoa for 1080000.0 in 2012/2013
Gianluca Caprari to Pescara for 1080000.0 in 2012/2013
Tin Jedvaj to Bay. Leverkusen for 900000.0 in 2014/2015
Marquinho to Ittihad for 900000.0 in 2014/2015
Valerio Verre to Udinese Calcio for 810000.0 in 2014/2015
Stefano Sabelli to Bari for 810000.0 in 2013/2014
Leandro Greco to Olympiacos for 765000.0 in 2012/2013
Marco Andreolli to Chievo Verona for 720000.0 in 2010/2011
Marco Borriello to West Ham for 630000.0 in 2013/2014
Matteo Politano to Pescara for 450000.0 in 2013/2014
Adrian Stoian to Chievo Verona for 450000.0 in 2012/2013
Marco Borriello to Juventus for 450000.0 in 2011/2012
Federico Barba to FC Empoli for 270000.0 in 2013/2014
Adrian Stoian to Bari for 270000.0 in 2012/2013
Mattia Montini to Benevento for 234000.0 in 2012/2013
Marco Borriello to Genoa for 225000.0 in 2012/2013
Bojan Krkic to AC Milan for 225000.0 in 2012/2013
Alessandro Florenzi to Crotone for 225000.0 in 2011/2012
Stefano Pettinari to Crotone for 225000.0 in 2012/2013
```

```
    Federico Barba to Grosseto for 180000.0 in 2013/2014
    Alessandro Crescenzi to Pescara for 180000.0 in 2012/2013
    Stefano Okaka to Fulham for 158000.0 in 2009/2010
    Federico Ricci to Crotone for 0 in 2014/2015
    Amato Ciciretti to Pistoiese for 0 in 2014/2015
    Mattia Rosato to AS Gubbio for 0 in 2014/2015
    Junior Tallo to SC Bastia for 0 in 2014/2015
    Petar Golubović to Novara for 0 in 2013/2014
    Francis Obeng to Santarcangelo for 0 in 2013/2014
    Marquinho to Hellas Verona for 0 in 2013/2014
    Giammario Piscitella to Cittadella for 0 in 2013/2014
    Paolo Frascatore to Reggina for 0 in 2013/2014
    Nicolás Burdisso to Genoa for 0 in 2013/2014
    Federico Viviani to Latina Calcio for 0 in 2013/2014
    Tomas Svedkauskas to Pescara for 0 in 2013/2014
    Alessandro Crescenzi to Novara for 0 in 2013/2014
    Jonatan Lucca to Atlético-PR B for 0 in 2013/2014
    Wesley Yamnaine to RWDM Brussels for 0 in 2013/2014
    Junior Tallo to Ajaccio for 0 in 2013/2014
    Alessandro Crescenzi to Ajaccio for 0 in 2013/2014
```

Italian transfers.

```
query = """MATCH (to:Club)<-[:TO_CLUB]-(t:Transfer)-[:FROM_CLUB]-(from:Cl
      (t)-[:OF_PLAYER]->(player:Player)-[:FROM]->(country),
      (to)-[:IN]->(country:Country)<-[:IN]-(from)
WHERE country.name = "Italy"
RETURN player.name, from.name, to.name, t.numericFee, t.season
ORDER BY t.numericFee DESC
LIMIT 10"""

result = conn.query(query)
for res in result:
  print(res[0],"from",res[1],"to",res[2],"for",res[3],"in",res[4])
```

```
    Andrea Ranocchia from Genoa to Inter for 16650000.0 in 2010/2011
    Giampaolo Pazzini from Sampdoria to Inter for 16200000.0 in 2010/2011
    Alessandro Matri from Cagliari Calcio to Juventus for 13950000.0 in 2011/20
    Stephan El Shaarawy from Genoa to AC Milan for 13950000.0 in 2011/2012
    Leonardo Bonucci from Bari to Juventus for 13950000.0 in 2010/2011
    Angelo Ogbonna from Torino to Juventus for 13500000.0 in 2013/2014
    Giampaolo Pazzini from Inter to AC Milan for 11250000.0 in 2012/2013
    Manolo Gabbiadini from Juventus to Sampdoria for 10350000.0 in 2013/2014
    Mattia Destro from Genoa to AS Roma for 10350000.0 in 2012/2013
    Alessandro Matri from Juventus to AC Milan for 9900000.0 in 2013/2014
```

Aggregate query for most spent players.

```
query = """MATCH (t:Transfer)-[:OF_PLAYER]->(p:Player)
WITH p, sum(t.numericFee) as moneySum, COUNT(*) AS numberOfTransfers
RETURN p.name, moneySum, numberOfTransfers
ORDER BY moneySum desc
LIMIT 10"""

result = conn.query(query)
for res in result:
  print(res[0],"with",res[1],"in",res[2],"transfer(s)")
```

```
Luis Suárez with 97400000.0 in 2 transfer(s)
Gareth Bale with 90900000.0 in 1 transfer(s)
Neymar with 79380000.0 in 1 transfer(s)
Falcao with 74700000.0 in 2 transfer(s)
Edinson Cavani with 73350000.0 in 3 transfer(s)
David Luiz with 67050000.0 in 2 transfer(s)
Juan Mata with 64290000.0 in 2 transfer(s)
Willian with 63450000.0 in 2 transfer(s)
Alexis Sánchez with 61650000.0 in 2 transfer(s)
Cesc Fàbregas with 60300000.0 in 2 transfer(s)
```

Transfers linked in a temporal way through aggregation is done.

```
query = """MATCH (p:Player)<-[:OF_PLAYER]-(transfer)
WHERE transfer.numericFee > 0

WITH p, transfer
ORDER BY p.name, transfer.timestamp

WITH p, collect(transfer) AS transfers
WHERE size(transfers) > 1

UNWIND range(0, size(transfers)-2) AS idx
WITH transfers[idx] AS t1, transfers[idx+1] AS t2
MERGE (t1)-[:NEXT]->(t2)"""

result = conn.query(query)
```

A query on the max profits made by a club.

```
query = """MATCH (p:Player)<-[:OF_PLAYER]-(t1)-[:NEXT]->(t2),
      (initial)<-[:FROM_CLUB]-(t1)-[:TO_CLUB]->(club1)<-[:FROM_CLUB]-(t2)-[
WHERE none(t in [t1, t2] where t:Loan)
RETURN p.name as player, club1.name AS earningClub , initial.name as buysFr
ORDER BY profit DESC
"""

#pandas data frames
df = pd.DataFrame([dict(_) for _ in conn.query(query)])
df.head(10)
```

	player	earningClub	buysFrom	sellsTo	profit
0	Luis Suárez	Liverpool	AFC Ajax	FC Barcelona	49700000.0
1	Edinson Cavani	SSC Napoli	US Palermo	Paris SG	47250000.0
2	James Rodríguez	FC Porto	CA Banfield	Monaco	33880000.0
3	Diego Costa	Atlético Madrid	Real Valladolid	Chelsea	33300000.0
4	Axel Witsel	Benfica	Standard Liège	Zenit S-Pb	27900000.0
5	Mesut Özil	Real Madrid	Werder Bremen	Arsenal	26100000.0
6	Ander Herrera	Athletic Bilbao	Real Zaragoza	Man Utd	25650000.0
7	Oscar	Internacional	São Paulo	Chelsea	23400000.0
8	Marquinhos	AS Roma	Corinthians	Paris SG	23130000.0
9	David Luiz	Chelsea	Benfica	Paris SG	22050000.0

Specific player query.

```
query = """MATCH (p:Player {name:'Olivier Giroud'})<-[:OF_PLAYER]-(transfer
WHERE transfer.numericFee > 0
RETURN transfer.fee
LIMIT 10"""
print("Olivier Giroud transfers:")
df = pd.DataFrame([dict(_) for _ in conn.query(query)])
df.head(10)
```

```
Olivier Giroud transfers:
```

	transfer.fee
0	£10.80m
1	£1.80m

Loop transfers.

```
query = """MATCH (p:Player)<-[:OF_PLAYER]-(t:Transfer)
MATCH path = (t)-[:NEXT*]->(t2)
MATCH (t)-[:FROM_CLUB]->(club)<-[:TO_CLUB]-(t2)
WHERE none(t in [t, t2] where t:Loan)
WITH p, t.numericFee - t2.numericFee AS profit, [transfer in nodes(path) |
RETURN p.name, apoc.number.format(profit), transfers
ORDER BY profit DESC"""

df = pd.DataFrame([dict(_) for _ in conn.query(query)])
df.head(10)
```

	p.name	apoc.number.format(profit)	transfers
0	Víctor Montaño	5,130,000	[Montpellier->Stade Rennais, Stade Rennais->Mo...
1	Andreas Cornelius	4,680,000	[FC Copenhagen->Cardiff, Cardiff->FC Copenhagen]
2	Mehdi Carcela-González	4,230,000	[Standard Liège->Anzhi, Anzhi->Standard Liège]
3	Pedro León	3,600,000	[Getafe CF->Real Madrid, Real Madrid->Getafe CF]
4	Vágner Love	3,600,000	[CSKA Moscow->Flamengo, Flamengo->CSKA Moscow]
5	Igor de Camargo	3,510,000	[Standard Liège->Bor. M'gladbach, Bor. M'gladb...
6	Christopher	3,060,000	[Anzhi->QPR, QPR->Anzhi]

Even relationships between clubs can be created.