Mining Big Datasets

Group Assignment

Team Members

Meropi Maria Argyropoulou (BAFT 1722)

Lila Tsopelakou (BAFT 1716)

Lisa Schramm (BAFT 1701)

Contents

Tables

Images

1. Neo4J Assignment

1.1. Data Model

The file that we were given in the context of this assignment contains the data from 684 chess games played in world chess tournaments. More specifically, the file includes the players' details, the tournaments, the date, the result of the game, as well as all the moves of each game and the positions that occur in the chessboard with each move. To model the data as a property graph and answer queries based on it, we decided to create four basic nodes Player, Game, Event and Position and for relationships among them as depicted in the Image 1 (between Player-Game, Game-Event, Game-Position for the 1st move and transition from a Position to Position for the rest of the moves). In addition, we put the information of the Gamenumber as a property of the move in order to know the game number that this specific move was played, which will later help us run the queries.

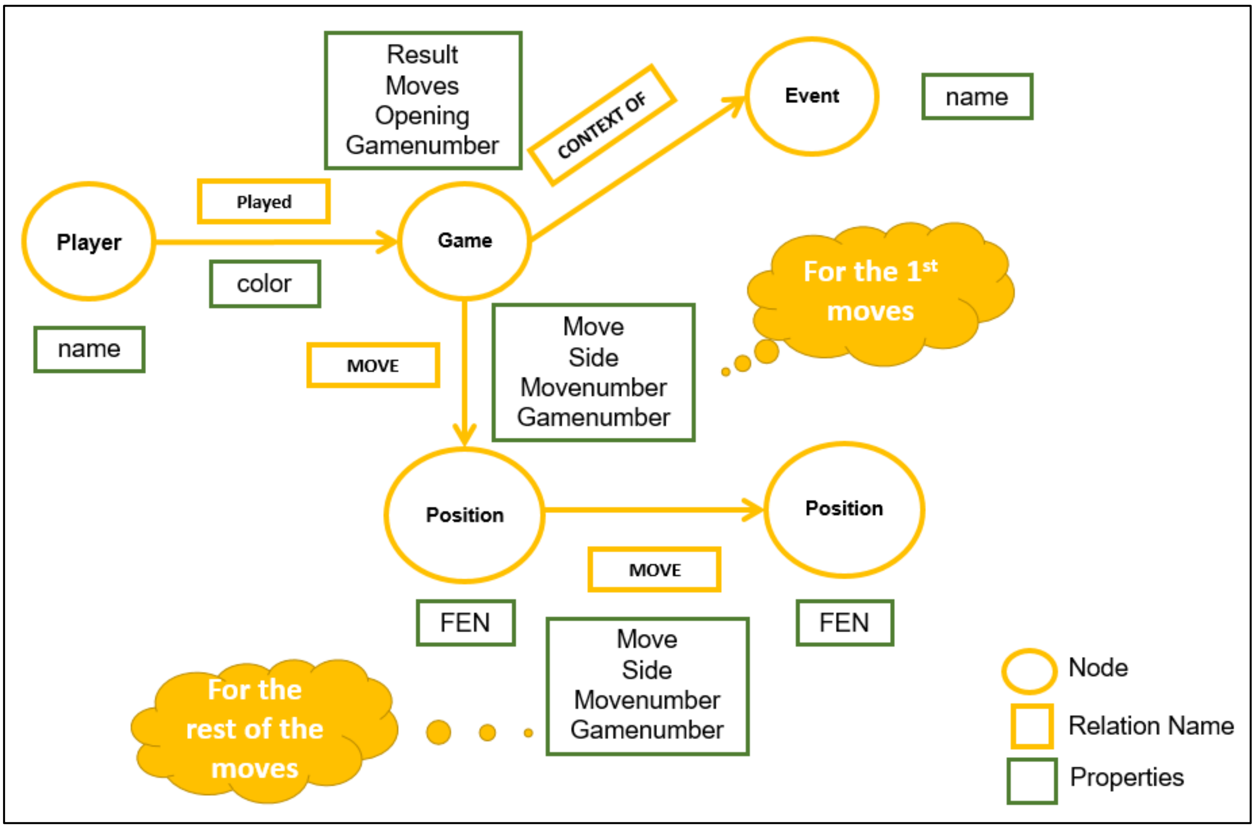


Image 1: High Level Data Model

Additionally, based on the aforementioned data model, we created an example with data illustrated on Image 2.

In the next sections, we explain how we parsed the data from the initial file provided *chessData.txt*, imported them in the neo4j graph database and wrote and executed queries in cypher language on them.

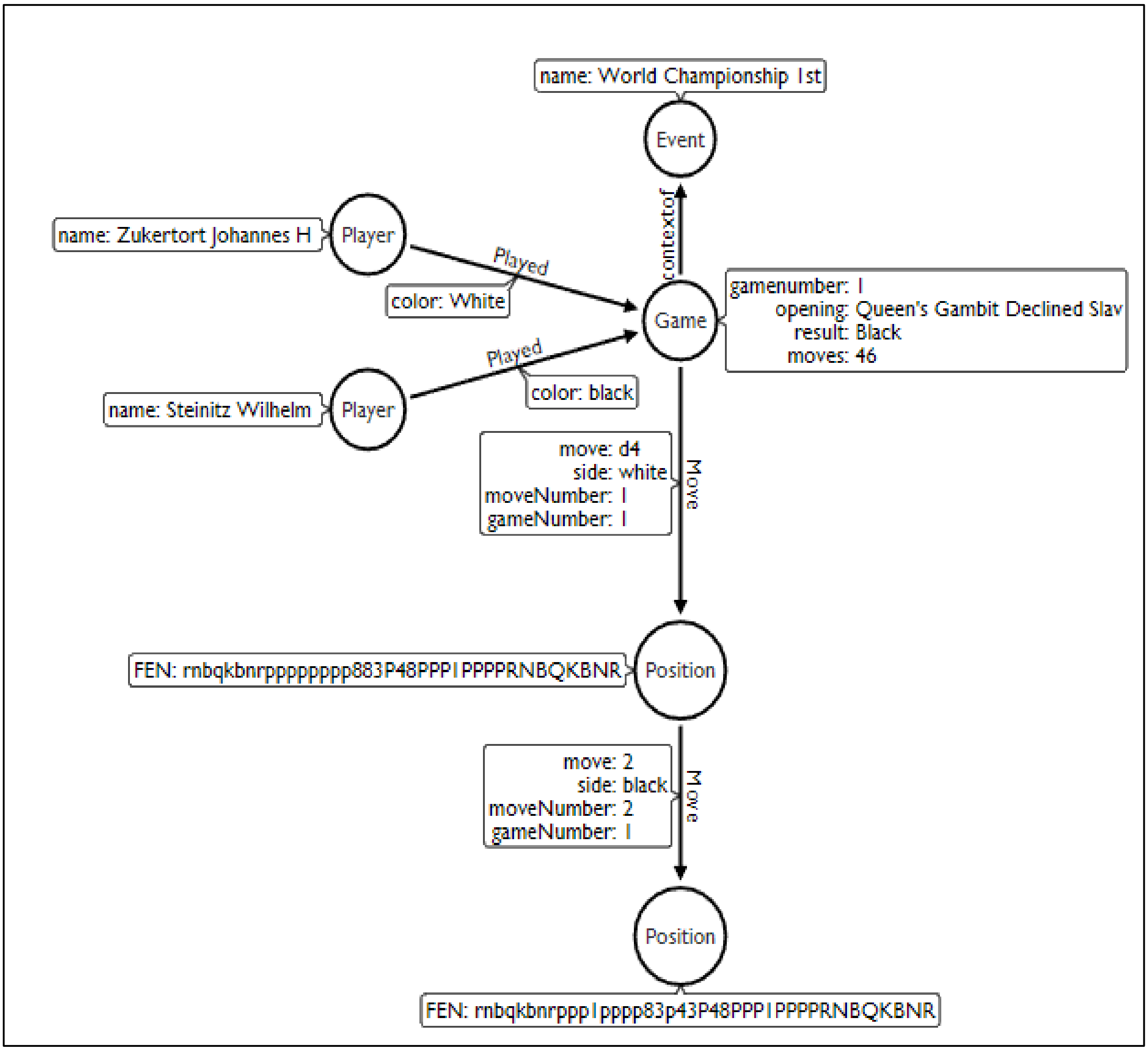


Image 2: Example based on the Data Model

1.2. Parsing the files

We created a program in R programming language which reads line by line the chessData.txt file and based on the split lines:

- "============================Game===",
- "-- Game Moves ---" and
- "==",

creates two new files, one containing the game details (*output.txt*) and one the moves information per game (*outputmoves.txt*).

Afterwards we wanted to transform our data to dataframes. In consequence we performed the following modifications:

- Dataframe games:
  For the game details we read the lines of the *output.txt* document and split by ":" the line in two columns. Because we wanted to make column names the Black, BlackElo, Date, ECO, Event, EventDate, GameNumber, HalfMoves, Moves, Opening, Result, Round, Site, White, WhiteElo we used *tidyr* and *purrr* packages to map the values of column 1 tp the values of column 2 and transform them then into a dataframe.
- Dataframe moves:
  For the move details we read the lines and trimmed the names of each line (*sub() function*). Then we put the names as columns creating that way a dataframe.

Based on the data model explained above, we created 8 csv files, depicted in Table 1 before importing them to Neo4J.

Table 1:csv files

Names	Description	csv file	Details
Player	Node	player.csv	Distinct player names
Game	Node	game.csv	Distinct games
Position	Node	position.csv	Distinct positions
Event	Node	event.csv	Distinct events
Played	Relationship between Player-Game	rpg.csv	Data of the relationship
Move	Relationship between Game-Position	rgpos.csv	Data of the relationship
Move	Relationship between Position-Position	rpospos.csv	Data of the relationship
ContextOf	Relationship between Game-Event	reg.csv	Data of the relationship

1.3. Importing the files into Neo4J

The code that follows includes all the statements that were executed to import the data into the Neo4J and create the respective graph data model. We downloaded the neo4j-community-3.4.1 version and making our machine a server, we connected to http://localhost:7474/browser/ and executed the commands.

Table 2: Import Commands

```
// Create node game
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///game.csv" AS row
CREATE (:Game {gameNumber: TOINT(row.GameNumber), moves: row.Moves, opening:
row.Opening, result: row.Result});

// Create node event
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///event.csv" AS row
CREATE (:Event {event: row.Event});

// Create node player
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///player.csv" AS row
CREATE (:Player {name: row.Player});

//Creating indexes
CREATE INDEX ON:Game(gameNumber);
CREATE INDEX ON:Game(result);
CREATE INDEX ON:Player(name);
CREATE INDEX ON:Event(event);

// Create relationship between game and event
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///reg.csv" AS row
MATCH (game:Game {gameNumber: TOINT(row.GameNumber)})
MATCH (event:Event {event:row.Event})
MERGE(game)-[:CONTEXTOF]->(event)

// Create relationship between game and player
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///rpg.csv" AS row
MATCH (game:Game {gameNumber: TOINT(row.GameNumber)})
MATCH (player:Player {name:row.Player})
MERGE(player)-[:PLAYED{color: row.Color}]->(game)

// Create position
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///position.csv" AS row
CREATE (:Position {FEN: row.Position});

//Creating indexes
CREATE INDEX ON:Position(FEN);

// Create relationship between game and position
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///rgpos.csv" AS row
MATCH (game:Game {gameNumber: TOINT(row.GameNumber)})
MATCH (position:Position {FEN:row.FEN})
MERGE(game)-[:MOVE{gameNumber: TOINT(row.GameNumber), movenumber:row.MoveNumber,
move:row.Move, side:row.Side}]->(position)
```

```
// Create relationship between position and position
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///rpospos.csv" AS row
MATCH (game:Game {gameNumber: TOINT(row.GameNumberBefore)})
MATCH (positionBefore:Position {FEN:row.FENBefore})
MATCH (positionAfter:Position {FEN:row.FENAfter})
MERGE(positionBefore)-[:MOVE{gameNumber: TOINT(row.GameNumberBefore),
movenumber:row.MoveNumberAfter, move:row.MoveAfter, side:row.SideAfter}]->(positionAfter)
```

1.4. Cypher Queries

- **Query 1:**

  In how many games (count) the position with FEN: r1bqkbnrpppp1ppp2n51B2p34P35N2PPPP1PPPRNBQK2R has appeared and what was the percentage that white wins.

  - Cypher Query:

Table 3: Query 1

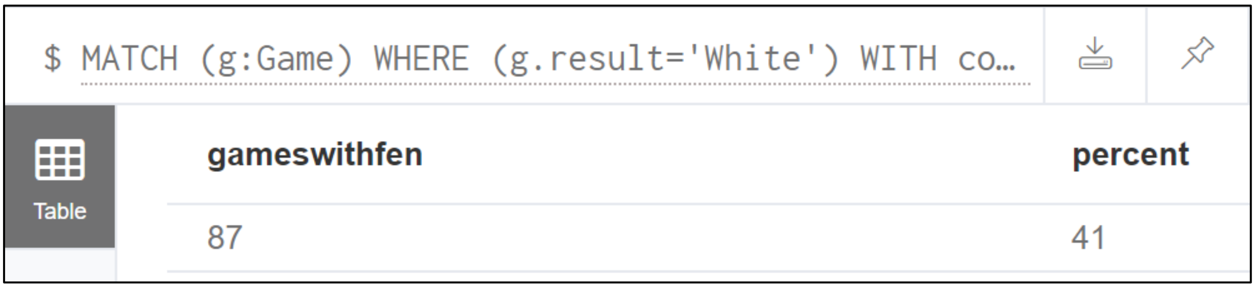
```
MATCH (g:Game)
WHERE (g.result='White')
WITH collect(g.gameNumber) as whitewinners
MATCH (n1:Position)-[r:MOVE]->(n2:Position)
WHERE (n1.FEN = 'r1bqkbnrpppp1ppp2n51B2p34P35N2PPPP1PPPRNBQK2R' OR n2.FEN
='r1bqkbnrpppp1ppp2n51B2p34P35N2PPPP1PPPRNBQK2R') AND (r.gameNumber) in whitewinners
WITH count(DISTINCT (r.gameNumber)) AS WHITEWITHFEN
MATCH (n1:Position)-[r1:MOVE]->(n2:Position)
WHERE (n1.FEN = 'r1bqkbnrpppp1ppp2n51B2p34P35N2PPPP1PPPRNBQK2R' OR n2.FEN
='r1bqkbnrpppp1ppp2n51B2p34P35N2PPPP1PPPRNBQK2R')
WITH 100*WHITEWITHFEN/count(DISTINCT(r1.gameNumber)) AS percent, count(DISTINCT(r1.gameNumber))
as gameswithfen
RETURN  gameswithfen, percent
```

  - Results:

Table 4: Results Query 1

gameswithfen	percent
87	41

  - Screenshot from Neo4J:



$ MATCH (g:Game) WHERE (g.result='White') WITH co…

gameswithfen	percent
87	41

Image 3: Screenshot Query 1

- **Query 2:**

For all games containing position FEN: r1bqkbnrpppp1ppp2n51B2p34P35N2PPPP1PPPRNBQK2R, how many times (count) won the white, won the black or the game was draw.

- Cypher Query:

Table 5:Query 2

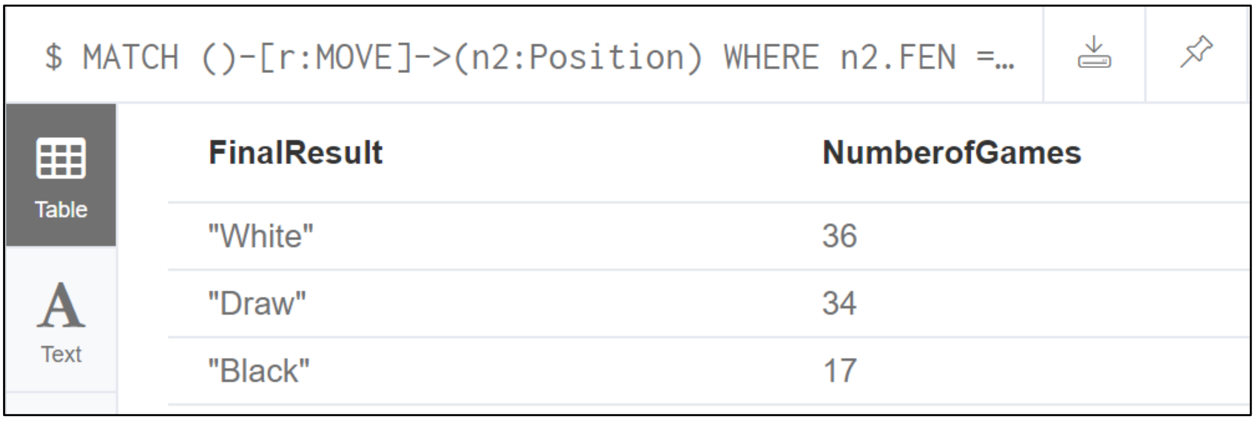
```
MATCH ()-[r:MOVE]->(n2:Position)
WHERE n2.FEN ='r1bqkbnrpppp1ppp2n51B2p34P35N2PPPP1PPPRNBQK2R'
WITH collect(DISTINCT(r.gameNumber)) AS AVECFEN
MATCH (g:Game)
WHERE g.gameNumber IN AVECFEN
UNWIND g.result AS FinalResult
RETURN FinalResult,COUNT(g) AS NumberofGames
```

- Results:

Table 6: Results Query 2

FinalResult	NumberofGames
White	36
Draw	34
Black	17

- Screenshot from Neo4J:



```
$ MATCH ()-[r:MOVE]->(n2:Position) WHERE n2.FEN =…
```

FinalResult	NumberofGames
"White"	36
"Draw"	34
"Black"	17

Image 4: Screenshot Query 2

- **Query 3:**

  Which was the event that had the most games, and in how many of these games had played "Karpov Anatoly" with white or black.

  - Cypher Query:

Table 7: Query 3

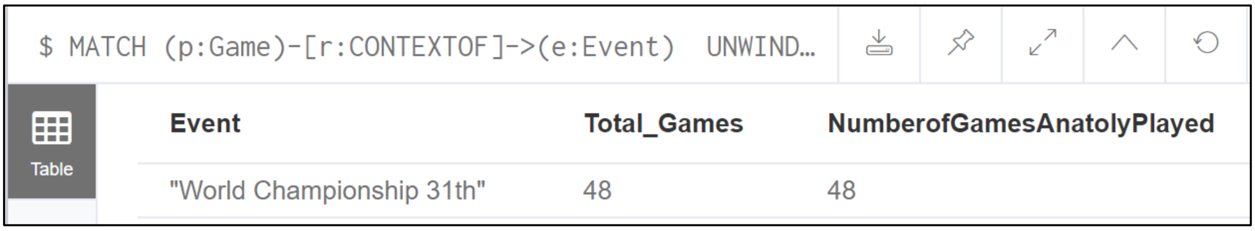
```
MATCH (p:Game)-[r:CONTEXTOF]->(e:Event)
UNWIND e.event AS Ev
WITH Ev, COUNT(r) AS cnt
ORDER BY cnt DESC
WITH COLLECT([Ev, cnt]) AS events
UNWIND events AS EV
WITH EV
WHERE EV[1]=events[0][1]
WITH COLLECT(EV[0]) AS even
MATCH (p1:Game)-[r2:CONTEXTOF]->(e2:Event)
WHERE e2.event IN even
WITH COLLECT([p1.gameNumber, e2.event]) AS games
UNWIND games AS GM
MATCH (pl:Player)-[r3:PLAYED]->(g:Game)
WHERE  pl.name='Karpov  Anatoly' AND g.gameNumber IN GM[0]
WITH count(g.gameNumber) AS NumberofGamesAnatolyPlayed, GM[1] AS Event, games AS games
UNWIND games AS GM2
WITH GM2, NumberofGamesAnatolyPlayed,Event
WHERE GM2[1]=Event
RETURN  Event, count(GM2[1]) AS Total_Games, NumberofGamesAnatolyPlayed
```

  - Results:

Table 8: Results Query 3

Event	Total_Games	NumberofGamesAnatolyPlayed
World Championship 31th	48	48

  - Screenshot from Neo4J:



```
$ MATCH (p:Game)-[r:CONTEXTOF]->(e:Event)  UNWIND...
```

Event	Total_Games	NumberofGamesAnatolyPlayed
"World Championship 31th"	48	48

Image 5: Screenshot Query 3

- **Query 4:**

  Which player had played most games with "Ruy Lopez" opening.

  - Cypher Query:

Table 9: Query 4

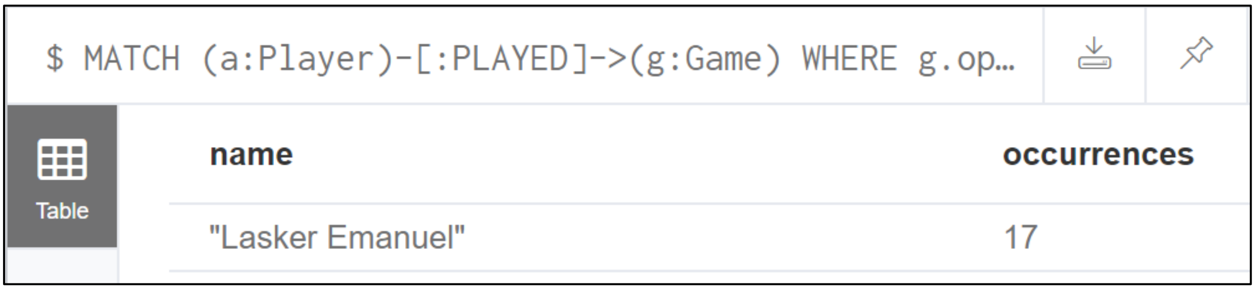
```
MATCH (a:Player)-[:PLAYED]->(g:Game)
WHERE g.opening='Ruy Lopez'
RETURN a.name AS name, count(a.name) as occurrences
ORDER BY occurrences DESC
LIMIT 1
```

  - Results:

Table 10: Results Query 4

name	occurrences
Lasker Emanuel	17

  - Screenshot from Neo4J:



```
$ MATCH (a:Player)-[:PLAYED]->(g:Game) WHERE g.op...
```

name	occurrences
"Lasker Emanuel"	17

Image 6: Screenshot Query 4

- **Query 5:**

  How many games had the sequence of moves (in the exact order) “Nc6”, “Bb5”, “a6”, and which was the players of these games.

  - Cypher Query:

Table 11: Query 5

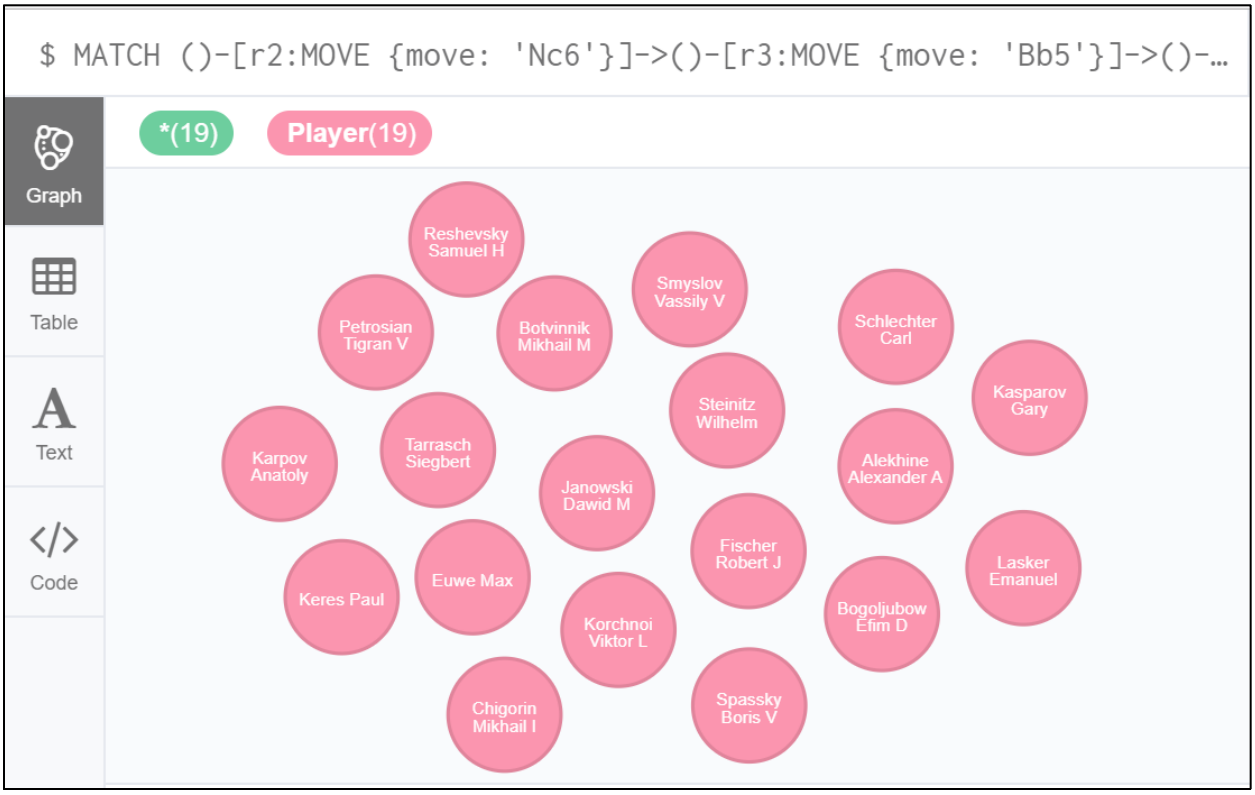
```
MATCH  ()-[r2:MOVE {move: 'Nc6'}]->()-[r3:MOVE {move: 'Bb5'}]->()-[r4:MOVE {move: 'a6'}]->()

WITH collect(DISTINCT r4.gameNumber) as moveordering

MATCH (p:Player)-[pl:PLAYED]->(g:Game)

WHERE g.gameNumber in moveordering

RETURN count(distinct(g.gameNumber)) as total_games_with_move_sequence, collect(DISTINCT(p)) as players
```

  - Results:

Table 12: Results Query 5

total_games_with_move_sequence	players
52	[{name:Korchnoi Viktor L}, {name:Karpov Anatoly}, {name:Botvinnik Mikhail M}, {name:Smyslov Vassily V}, {name:Bogoljubow Efim D}, {name:Alekhine Alexander A}, {name:Spassky Boris V}, {name:Fischer Robert J}, {name:Schlechter Carl}, {name:Lasker Emanuel}, {name:Petrosian Tigran V}, {name:Reshevsky Samuel H}, {name:Keres Paul}, {name:Tarrasch Siegbert}, {name:Euwe Max}, {name:Janowski Dawid M}, {name:Chigorin Mikhail I}, {name:Steinitz Wilhelm}, {name:Kasparov Gary}]

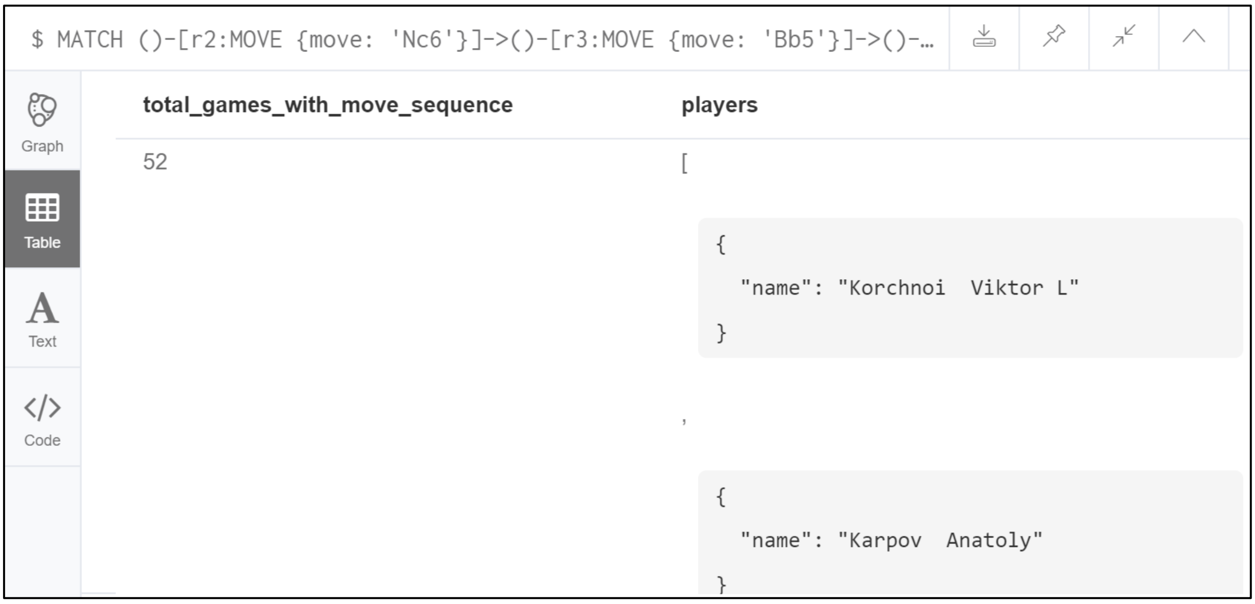
- Screenshot from Neo4J (1):



```
$ MATCH ()-[r2:MOVE {move: 'Nc6'}]->()-[r3:MOVE {move: 'Bb5'}]->()-…
```

Image 7: Names of players

- Screenshot from Neo4J (2):



```
$ MATCH ()-[r2:MOVE {move: 'Nc6'}]->()-[r3:MOVE {move: 'Bb5'}]->()-…
```

total_games_with_move_sequence	players

52	[
	{
	"name": "Korchnoi Viktor L"
	}
	,
	{
	"name": "Karpov Anatoly"
	}

Image 8: Screenshot Query 5

- **Query 6:**

  Display all game details, event, players and moves of the game with GameNumber: 636.

  - Cypher Query:

Table 13: Query 6

```
MATCH ()-[r1:MOVE]->(p1:Position)
MATCH (n:Game)-[c:CONTEXTOF]->(e:Event)
MATCH (p:Player)-[pl:PLAYED]->(n:Game)
WHERE r1.gameNumber = 636 and n.gameNumber = 636
RETURN n as game, r1 as move, p1 as positionaftermove, e as event, pl as color, p as player, n.opening as
opening,n.result as result,n.moves as moves
```

  - Results:

We don’t display the table here as the requested information is expanded in 162 (81 moves by 2 players) rows. Alternatively, the graph of the game is depicted below in Image 9 and zoomed in Image 10**Error! Reference source not found.** from the Neo4J platform.

  - Screenshot from Neo4J (1):

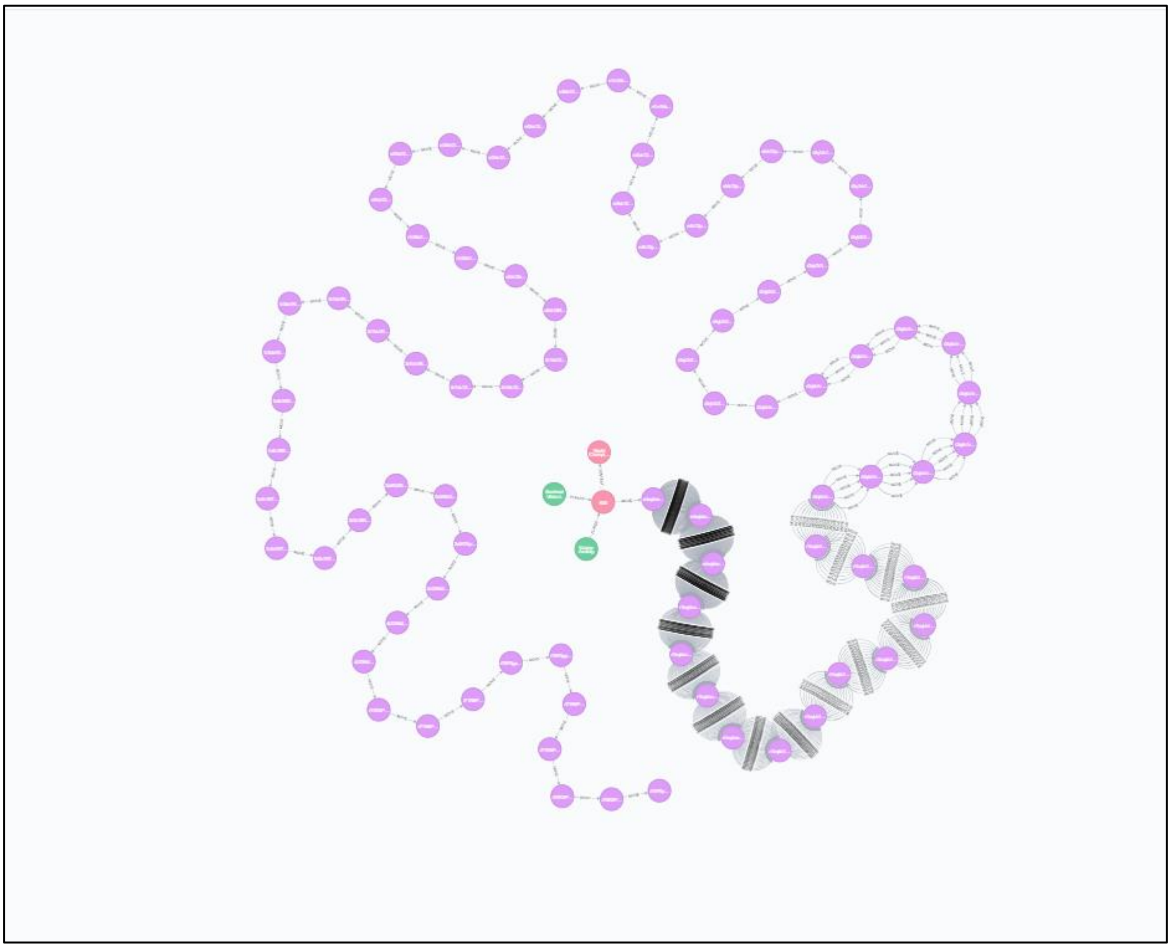
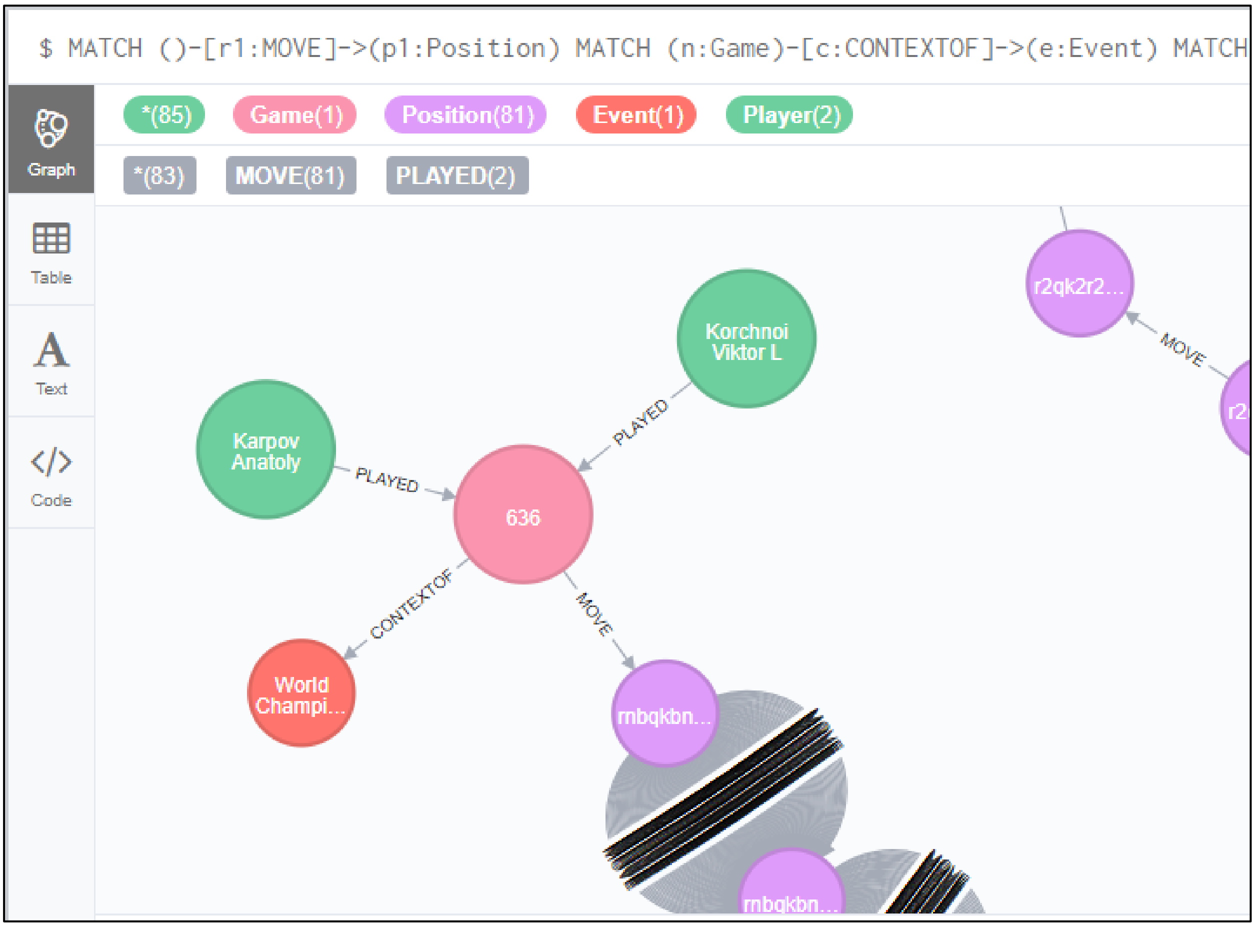


Image 9: Graph of Game 636

- Screenshot from Neo4J (2):



```
$ MATCH ()-[r1:MOVE]->(p1:Position) MATCH (n:Game)-[c:CONTEXTOF]->(e:Event) MATCH
```

Image 10: Game 636 zoomed

- **Query 7:**

  Display all chess games that include the position with FEN: r1bqkbnrpppp1ppp2n51B2p34P35N2PPPP1PPPRNBQK2R and after this position the move "a6" was not played. Also display the alternative moves and the game result.

  - Cypher Query:

Table 14: Query 7

```
MATCH (n1:Position{FEN:'r1bqkbnrpppp1ppp2n51B2p34P35N2PPPP1PPPRNBQK2R'})-[r2:MOVE]->()
WHERE r2.move<>'a6'
WITH collect([r2.gameNumber, r2.move]) AS results, r2.gameNumber AS gamenumbers
UNWIND results as games_alternative_moves
MATCH (g:Game)
where g.gameNumber IN games_alternative_moves
RETURN g.gameNumber AS GameNumber, g.result AS FinalResult, games_alternative_moves[1] as
Alternative_Move
ORDER BY g.gameNumber ASC
```

  - Results:

    There were 35 chess games that included the position with FEN *'r1bqkbnrpppp1ppp2n51B2p34P35N2PPPP1PPPRNBQK2R'* and after this the move 'a6' was not played. The respective chess games, the final results and the alternative moves are appearing in Table 15 below:

Table 15: Results Query 7

GameNumber	FinalResult	Alternative_Move
4	Black	Nf6
6	White	Nf6
8	Draw	Nf6
10	Draw	Nf6
12	White	Nf6
14	Draw	Nf6
16	White	Nf6
18	White	Nf6
23	White	d6
39	Black	d6
58	Draw	Nf6
60	White	Nf6
67	Black	d6
70	White	Nf6
80	White	d6
81	White	Nf6
82	White	d6
84	Draw	d6

GameNumber	FinalResult	Alternative_Move
86	White	d6
88	White	d6
99	Black	Nf6
115	Black	Nf6
117	Black	Nf6
121	Draw	Nf6
123	White	Nf6
125	White	Nf6
127	Draw	Nf6
129	Black	Nf6
151	Draw	Nf6
153	Draw	Nf6
155	White	Nf6
166	Draw	Nf6
172	Draw	Nf6
174	Black	Nf6
620	White	Nf6