MariaDB vs Neo4j

Comparative Performance Analysis of Relational and NoSQL Database Systems

Cristian Santaroni 1800659

Francesco Fortunato 1848527

Data Management Course

Master in Engineering in Computer Science

A.y. 2022/2023

OVERVIEW

Topic

Pick up a relational database system, a noSQL database system and a dataset, to compare their performances w.r.t. the execution of some relevant queries.

Goal

Find queries where relational database systems perform better and queries where noSQL database system perform better.

TOOLS

- Relational Database System: MariaDB
- NoSQL Database System: Neo4j
- Python
- Dbeaver

DATASET DESCRIPTION

Dataset Details:

- Clean, Structured, and Updated Football Data
- Sourced from Transfermarkt (Kaggle link)

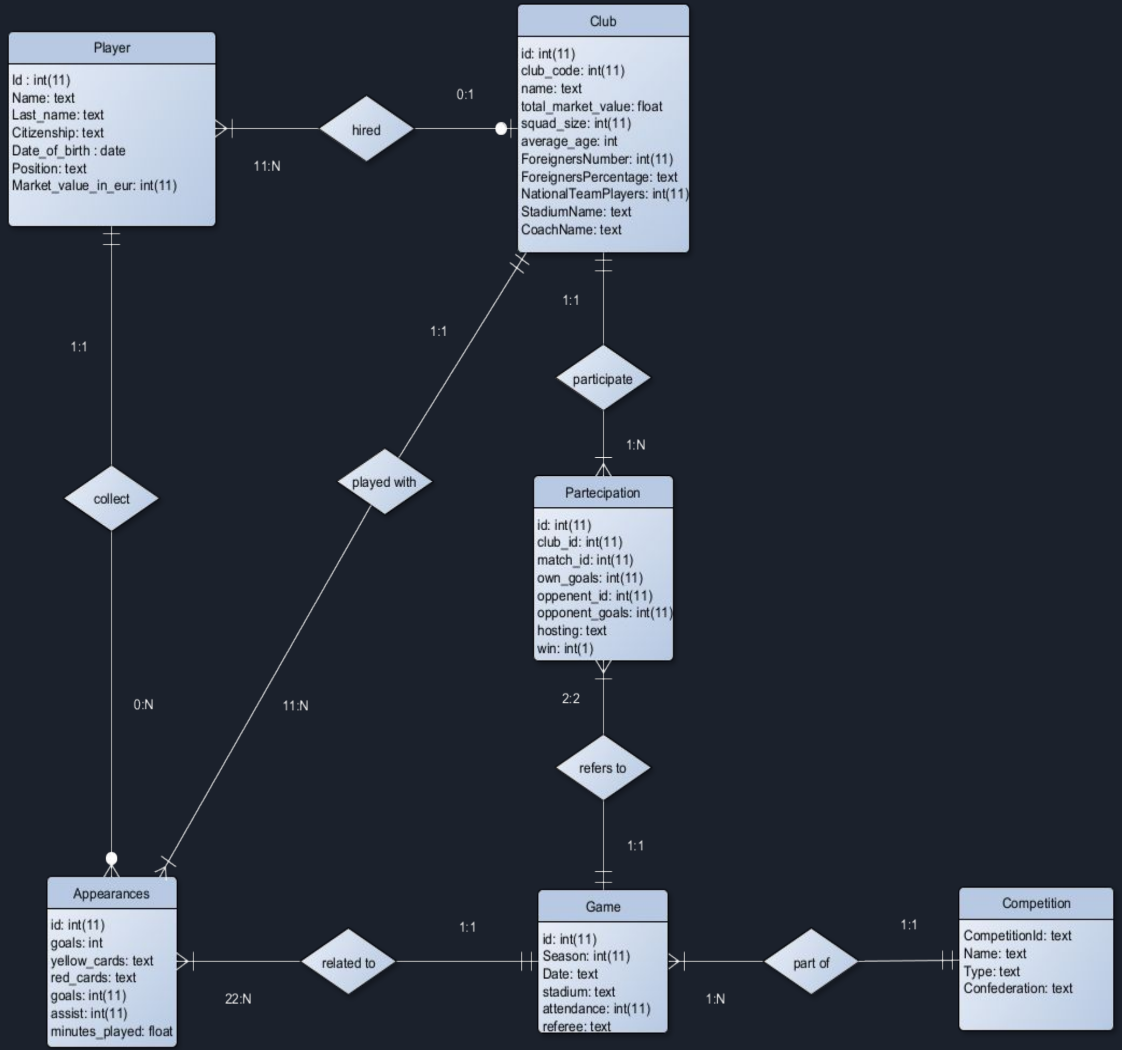
Dataset Contents:

- 63.281 Games Spanning Multiple Seasons
- 106.074 Participations Related to Games
- Representing All 43 Major Competitions
- Data on 426 Football Clubs
- Information on 29.455 Football Players
- Extensive Data on 1.179.060 Player Appearances
- Total numbers of records: 1.378.339

ER SCHEMA

Tables:

- **Player**: This table stores information about football players.
- **Appearances**: This table keeps records of player appearances in various games.
- **Game**: This table contains details about football games.
- **Club**: This table holds information about football clubs.
- **Participation**: This table links players and clubs to the games in which they participated.
- **Competition**: This table contains data about football competitions.



Player
Id : int(11)
Name: text
Last_name: text
Citizenship: text
Date_of_birth : date
Position: text
Market_value_in_eur: int(11)

Club
id: int(11)
club_code: int(11)
name: text
total_market_value: float
squad_size: int(11)
average_age: int
ForeignersNumber: int(11)
ForeignersPercentage: text
NationalTeamPlayers: int(11)
StadiumName: text
CoachName: text

Partecipation
id: int(11)
club_id: int(11)
match_id: int(11)
own_goals: int(11)
oppenent_id: int(11)
opponent_goals: int(11)
hosting: text
win: int(1)

Appearances
id: int(11)
goals: int
yellow_cards: text
red_cards: text
goals: int(11)
assist: int(11)
minutes_played: float

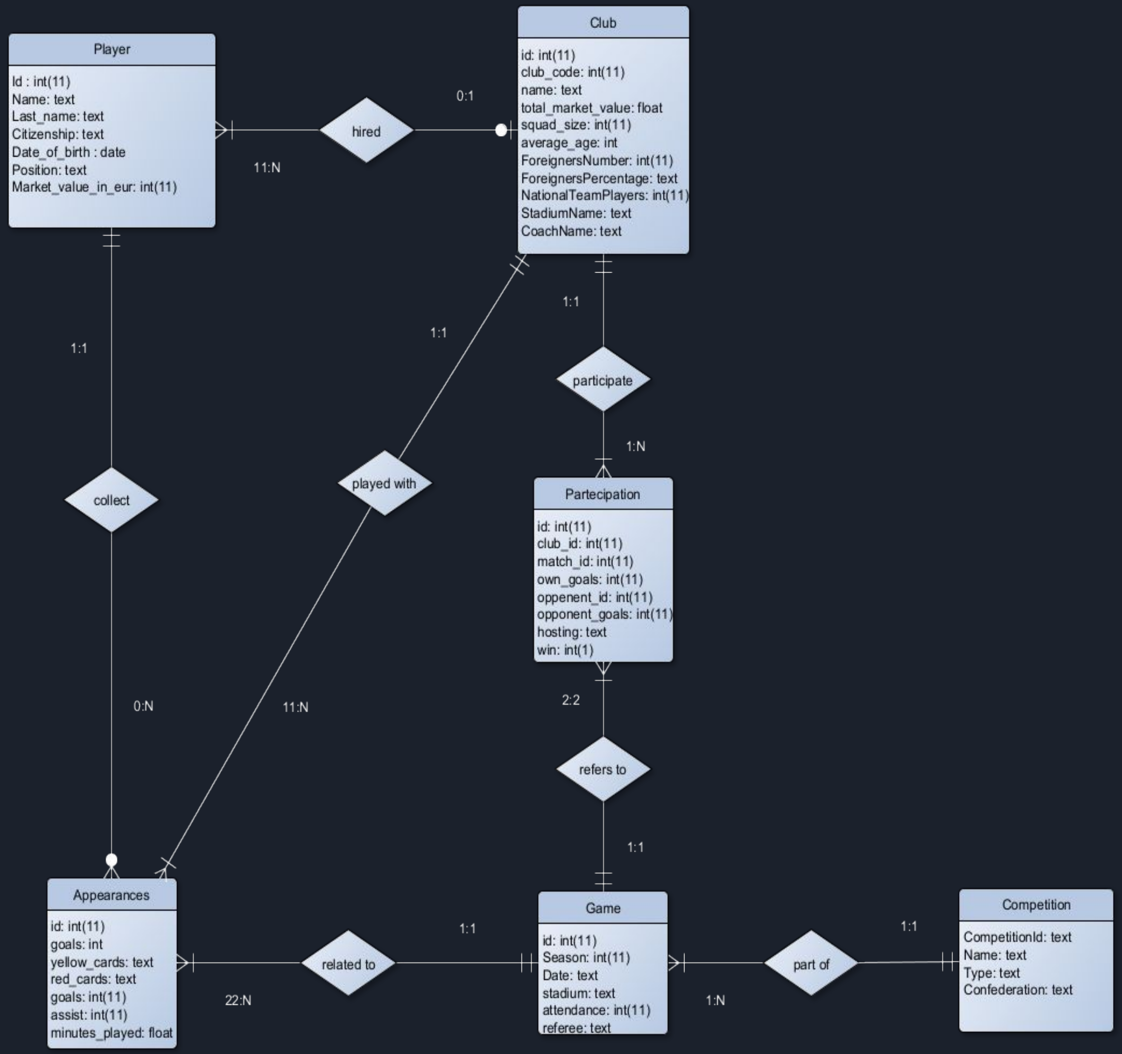
Game
id: int(11)
Season: int(11)
Date: text
stadium: text
attendance: int(11)
referee: text

Competition
CompetitionId: text
Name: text
Type: text
Confederation: text

ER SCHEMA

Relations:

- **Collect**: player collects an appearance
- **Hired**: a player can be hired by a club
- **Refers to**: a game must have 2 clubs
- **Part of**: a game belongs to a competition
- **Played with**: an appearance is always played with a club
- **Participate**: a participation is always related to a club



Player

Id : int(11)
Name: text
Last_name: text
Citizenship: text
Date_of_birth : date
Position: text
Market_value_in_eur: int(11)

Club

id: int(11)
club_code: int(11)
name: text
total_market_value: float
squad_size: int(11)
average_age: int
ForeignersNumber: int(11)
ForeignersPercentage: text
NationalTeamPlayers: int(11)
StadiumName: text
CoachName: text

Partecipation

id: int(11)
club_id: int(11)
match_id: int(11)
own_goals: int(11)
oppenent_id: int(11)
opponent_goals: int(11)
hosting: text
win: int(1)

Appearances

id: int(11)
goals: int
yellow_cards: text
red_cards: text
goals: int(11)
assist: int(11)
minutes_played: float

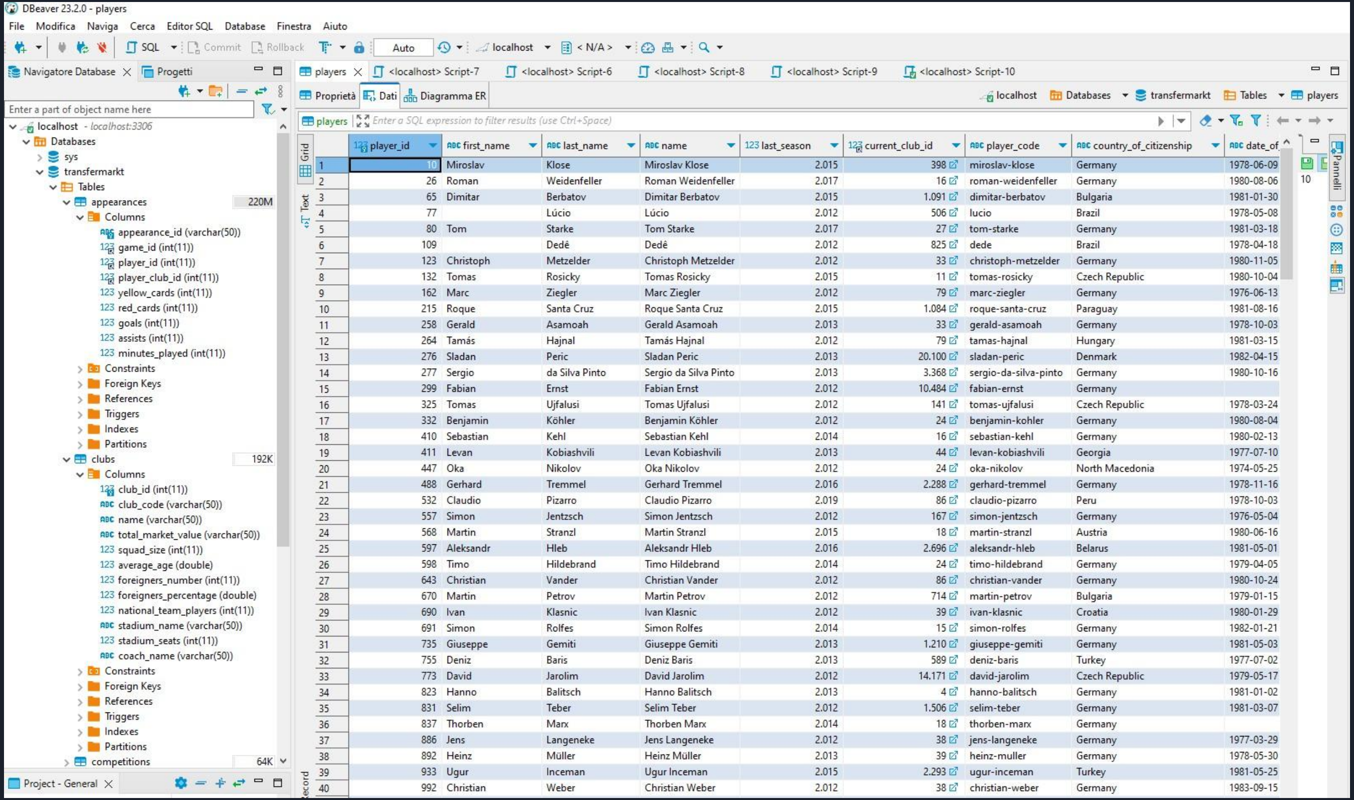
Game

id: int(11)
Season: int(11)
Date: text
stadium: text
attendance: int(11)
referee: text

Competition

CompetitionId: text
Name: text
Type: text
Confederation: text

DBeaver: EXCERPT



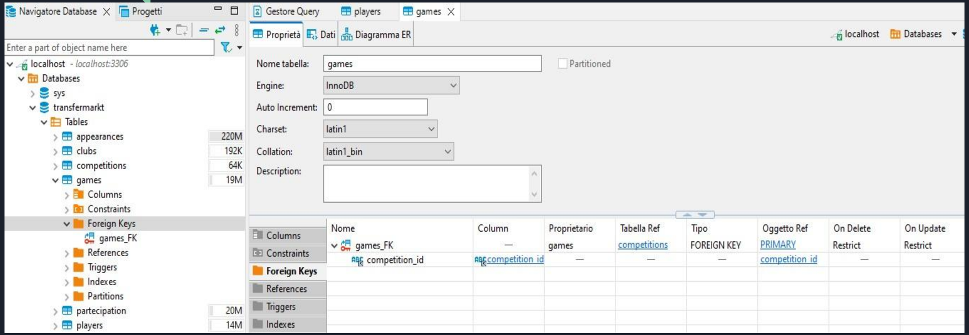
- localhost - localhost:3306
  - Databases
    - sys
    - transfermarkt
      - Tables
        - appearances 220M
          - Columns
            - appearance_id (varchar(50))
            - game_id (int(11))
            - player_id (int(11))
            - player_club_id (int(11))
            - yellow_cards (int(11))
            - red_cards (int(11))
            - goals (int(11))
            - assists (int(11))
            - minutes_played (int(11))
          - Constraints
          - Foreign Keys
          - References
          - Triggers
          - Indexes
          - Partitions
        - clubs 192K
          - Columns
            - club_id (int(11))
            - club_code (varchar(50))
            - name (varchar(50))
            - total_market_value (varchar(50))
            - squad_size (int(11))
            - average_age (double)
            - foreigners_number (int(11))
            - foreigners_percentage (double)
            - national_team_players (int(11))
            - stadium_name (varchar(50))
            - stadium_seats (int(11))
            - coach_name (varchar(50))
          - Constraints
          - Foreign Keys
          - References
          - Triggers
          - Indexes
          - Partitions
        - competitions 64K

players | Enter a SQL expression to filter results (use Ctrl+Space)

	player_id	first_name	last_name	name	last_season	current_club_id	player_code	country_of_citizenship	date_of
1	10	Miroslav	Klose	Miroslav Klose	2.015	398	miroslav-klose	Germany	1978-06-09
2	26	Roman	Weidenfeller	Roman Weidenfeller	2.017	16	roman-weidenfeller	Germany	1980-08-06
3	65	Dimitar	Berbatov	Dimitar Berbatov	2.015	1.091	dimitar-berbatov	Bulgaria	1981-01-30
4	77		Lúcio	Lúcio	2.012	506	lucio	Brazil	1978-05-08
5	80	Tom	Starke	Tom Starke	2.017	27	tom-starke	Germany	1981-03-18
6	109		Dedê	Dedê	2.012	825	dede	Brazil	1978-04-18
7	123	Christoph	Metzelder	Christoph Metzelder	2.012	33	christoph-metzelder	Germany	1980-11-05
8	132	Tomas	Rosicky	Tomas Rosicky	2.015	11	tomas-rosicky	Czech Republic	1980-10-04
9	162	Marc	Ziegler	Marc Ziegler	2.012	79	marc-ziegler	Germany	1976-06-13
10	215	Roque	Santa Cruz	Roque Santa Cruz	2.015	1.084	roque-santa-cruz	Paraguay	1981-08-16
11	258	Gerald	Asamoah	Gerald Asamoah	2.013	33	gerald-asamoah	Germany	1978-10-03
12	264	Tamás	Hajnal	Tamás Hajnal	2.012	79	tamas-hajnal	Hungary	1981-03-15
13	276	Sladan	Peric	Sladan Peric	2.013	20.100	sladan-peric	Denmark	1982-04-15
14	277	Sergio	da Silva Pinto	Sergio da Silva Pinto	2.013	3.368	sergio-da-silva-pinto	Germany	1980-10-16
15	299	Fabian	Ernst	Fabian Ernst	2.012	10.484	fabian-ernst	Germany	
16	325	Tomas	Ujfalusi	Tomas Ujfalusi	2.012	141	tomas-ujfalusi	Czech Republic	1978-03-24
17	332	Benjamin	Köhler	Benjamin Köhler	2.012	24	benjamin-kohler	Germany	1980-08-04
18	410	Sebastian	Kehl	Sebastian Kehl	2.014	16	sebastian-kehl	Germany	1980-02-13
19	411	Levan	Kobiashvili	Levan Kobiashvili	2.013	44	levan-kobiashvili	Georgia	1977-07-10
20	447	Oka	Nikolov	Oka Nikolov	2.012	24	oka-nikolov	North Macedonia	1974-05-25
21	488	Gerhard	Tremmel	Gerhard Tremmel	2.016	2.288	gerhard-tremmel	Germany	1978-11-16
22	532	Claudio	Pizarro	Claudio Pizarro	2.019	86	claudio-pizarro	Peru	1978-10-03
23	557	Simon	Jentzsch	Simon Jentzsch	2.012	167	simon-jentzsch	Germany	1976-05-04
24	568	Martin	Stranzl	Martin Stranzl	2.015	18	martin-stranzl	Austria	1980-06-16
25	597	Aleksandr	Hleb	Aleksandr Hleb	2.016	2.696	aleksandr-hleb	Belarus	1981-05-01
26	598	Timo	Hildebrand	Timo Hildebrand	2.014	24	timo-hildebrand	Germany	1979-04-05
27	643	Christian	Vander	Christian Vander	2.012	86	christian-vander	Germany	1980-10-24
28	670	Martin	Petrov	Martin Petrov	2.012	714	martin-petrov	Bulgaria	1979-01-15
29	690	Ivan	Klasnic	Ivan Klasnic	2.012	39	ivan-klasnic	Croatia	1980-01-29
30	691	Simon	Rolfes	Simon Rolfes	2.014	15	simon-rolfes	Germany	1982-01-21
31	735	Giuseppe	Gemiti	Giuseppe Gemiti	2.013	1.210	giuseppe-gemiti	Germany	1981-05-03
32	755	Deniz	Baris	Deniz Baris	2.013	589	deniz-baris	Turkey	1977-07-02
33	773	David	Jarolim	David Jarolim	2.012	14.171	david-jarolim	Czech Republic	1979-05-17
34	823	Hanno	Balitsch	Hanno Balitsch	2.013	4	hanno-balitsch	Germany	1981-01-02
35	831	Selim	Teber	Selim Teber	2.012	1.506	selim-teber	Germany	1981-03-07
36	837	Thorben	Marx	Thorben Marx	2.014	18	thorben-marx	Germany	
37	886	Jens	Langeneke	Jens Langeneke	2.012	38	jens-langeneke	Germany	1977-03-29
38	892	Heinz	Müller	Heinz Müller	2.013	39	heinz-muller	Germany	1978-05-30
39	933	Ugur	Inceman	Ugur Inceman	2.015	2.293	ugur-inceman	Turkey	1981-05-25
40	992	Christian	Weber	Christian Weber	2.012	38	christian-weber	Germany	1983-09-15

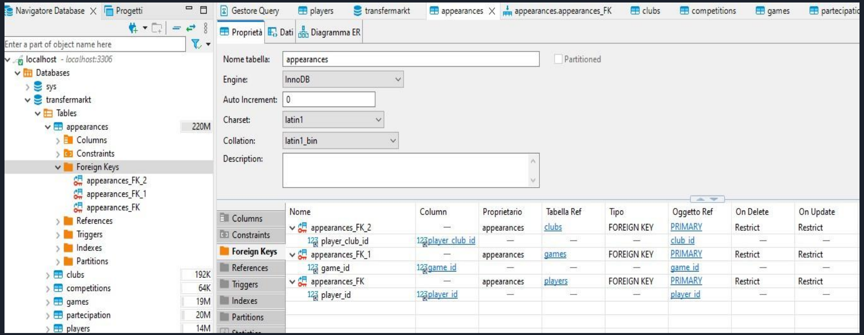
10

DBeaver: FOREIGN KEYS



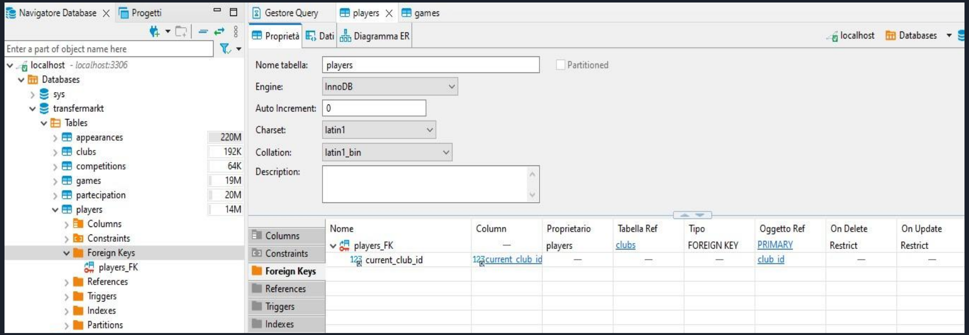
Nome tabella: games
□ Partitioned
Engine: InnoDB
Auto Increment: 0
Charset: latin1
Collation: latin1_bin
Description:

Nome	Column	Proprietario	Tabella Ref	Tipo	Oggetto Ref	On Delete	On Update
games_FK	—	games	competitions	FOREIGN KEY	PRIMARY	Restrict	Restrict
competition_id	competition_id	—	—	—	competition_id	—	—



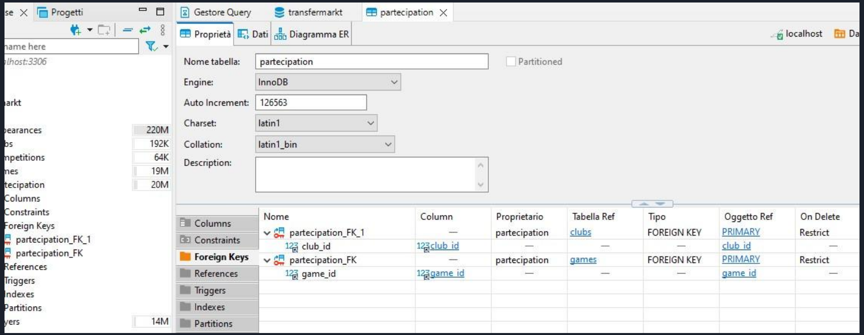
Nome tabella: appearances
□ Partitioned
Engine: InnoDB
Auto Increment: 0
Charset: latin1
Collation: latin1_bin
Description:

Nome	Column	Proprietario	Tabella Ref	Tipo	Oggetto Ref	On Delete	On Update
appearances_FK_2	—	appearances	clubs	FOREIGN KEY	PRIMARY	Restrict	Restrict
player_club_id	player_club_id	—	—	—	club_id	—	—
appearances_FK_1	—	appearances	games	FOREIGN KEY	PRIMARY	Restrict	Restrict
game_id	game_id	—	—	—	game_id	—	—
appearances_FK	—	appearances	players	FOREIGN KEY	PRIMARY	Restrict	Restrict
player_id	player_id	—	—	—	player_id	—	—



Nome tabella: players
□ Partitioned
Engine: InnoDB
Auto Increment: 0
Charset: latin1
Collation: latin1_bin
Description:

Nome	Column	Proprietario	Tabella Ref	Tipo	Oggetto Ref	On Delete	On Update
players_FK	—	players	clubs	FOREIGN KEY	PRIMARY	Restrict	Restrict
current_club_id	current_club_id	—	—	—	club_id	—	—



Nome tabella: partecipation
□ Partitioned
Engine: InnoDB
Auto Increment: 126563
Charset: latin1
Collation: latin1_bin
Description:

Nome	Column	Proprietario	Tabella Ref	Tipo	Oggetto Ref	On Delete
partecipation_FK_1	—	partecipation	clubs	FOREIGN KEY	PRIMARY	Restrict
club_id	club_id	—	—	—	club_id	—
partecipation_FK	—	partecipation	games	FOREIGN KEY	PRIMARY	Restrict
game_id	game_id	—	—	—	game_id	—

Python scripts: importing data

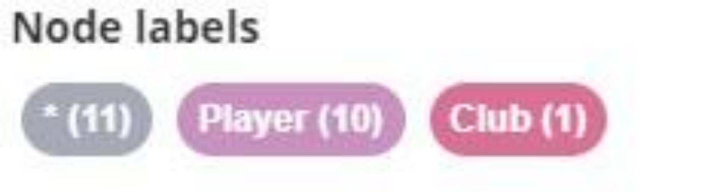
In order to import all the data in our graph db, to create nodes and edges, we designed a python script. To improve the creation of the nodes and the respective edges we used a “**Uniqueness Constraint** “ which ensure the uniqueness of the nodes based on the ID. This is used by Neo4j which automatically define an index to speed up the loading process and queries.

The python code can be divided in 3 parts:

- **Data Extraction** : opening all the csv files taken from kaggle and organizing them into dictionaries.
- **Node Creation**: using cypher queries, nodes are created for each data type.
- **Edges Creation**: using cypher, the relationships are created between the nodes

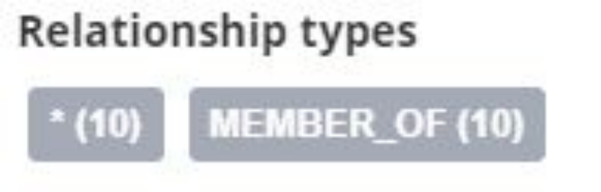
After this phase, data consistency was achieved with a total number of nodes of 1.378.339 and 3.842.064 relationships

Neo4j



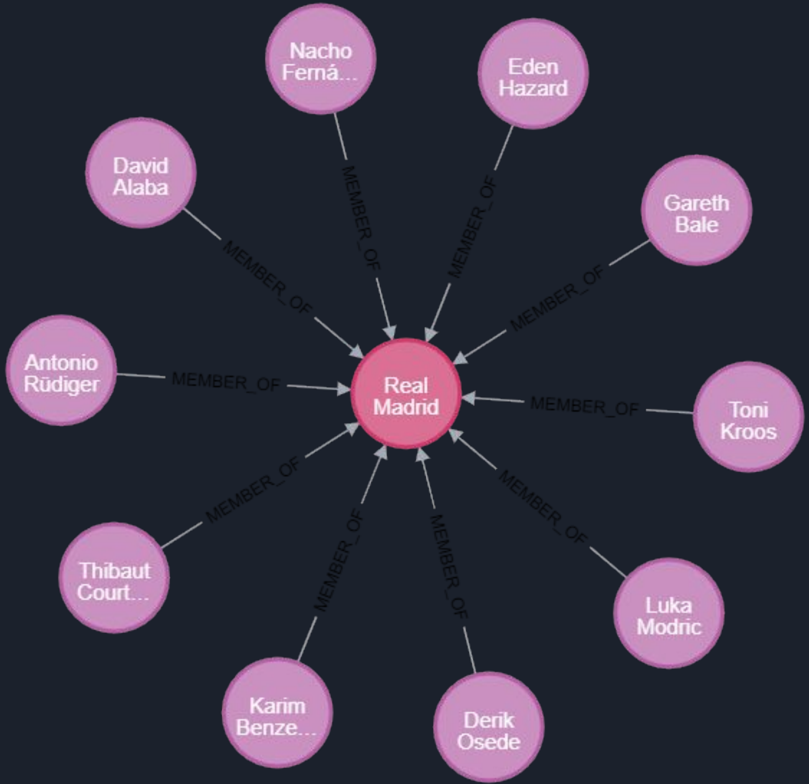
Node labels

* (11) Player (10) Club (1)

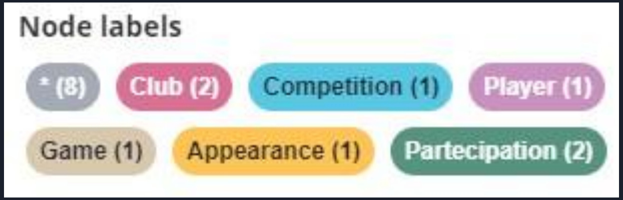


Relationship types

* (10) MEMBER_OF (10)



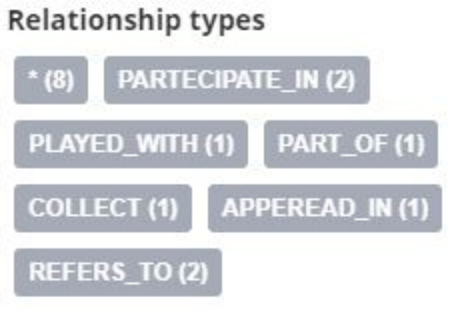
Neo4j



Node labels

* (8) Club (2) Competition (1) Player (1)

Game (1) Appearance (1) Partecipation (2)



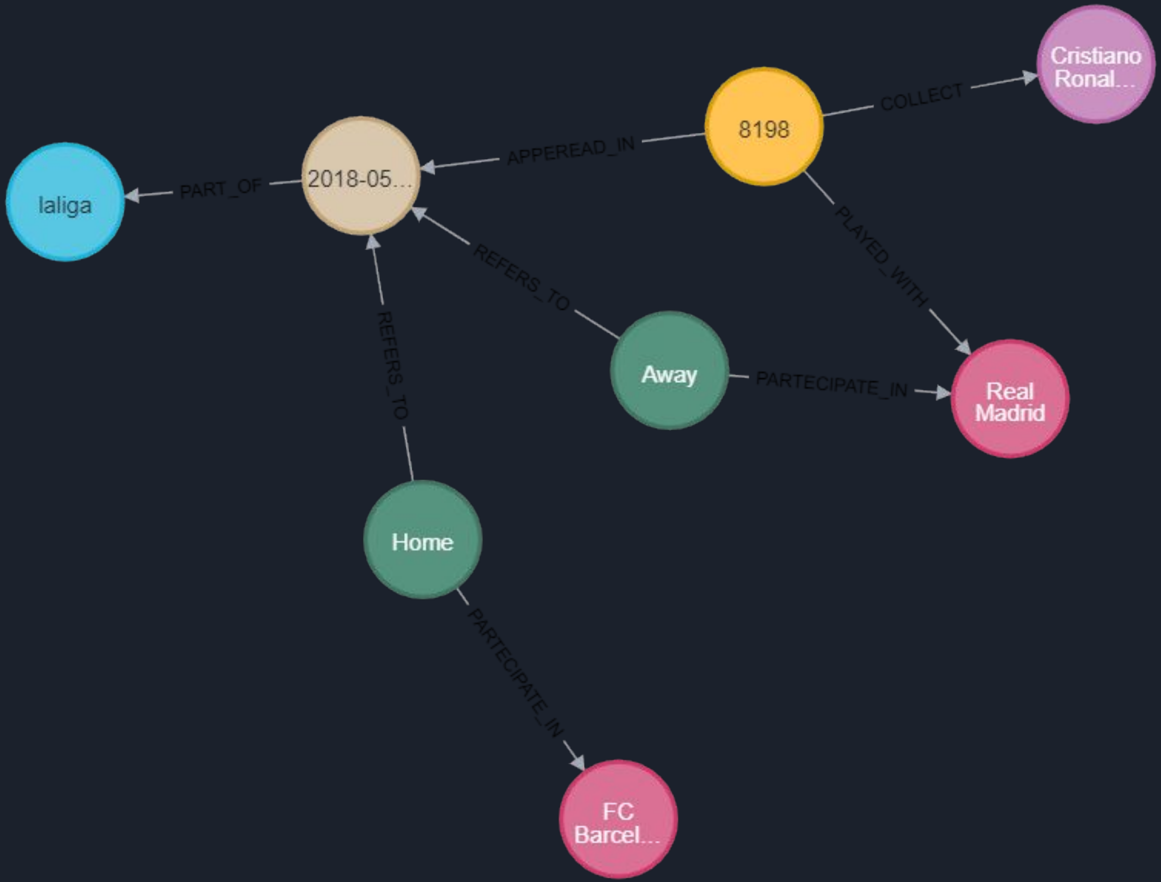
Relationship types

* (8) PARTECIPATE_IN (2)

PLAYED_WITH (1) PART_OF (1)

COLLECT (1) APPEREAD_IN (1)

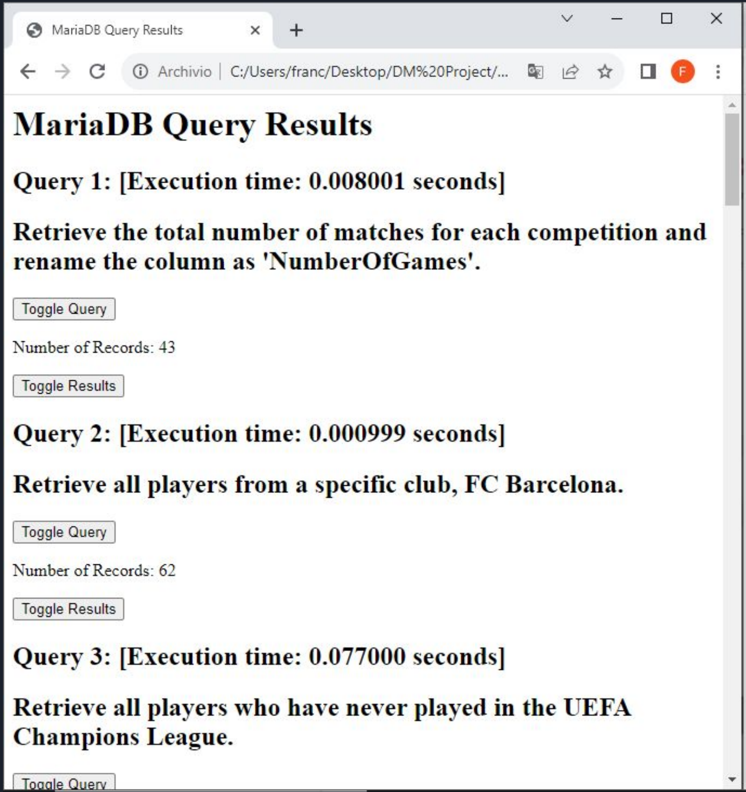
REFERS_TO (2)



Python scripts: average execution time and demo

- For the live demo, we designed a Python script which runs 20 queries and stores the results inside two different html files, one for MariaDB and one for Neo4j. The code can be reachable here.
- Regarding the performance evaluation, to measure and compare the average execution time between Mariadb and Neo4j across various queries, we've developed a dedicated Python script. The script runs each query 1000 times, and then computes the average execution time of each query. In the subsequent slides, we will present the results of this performance evaluation, providing the average execution times for each query comparing MariaDB and Neo4j.

Python scripts: Live Demo



MariaDB Query Results

Query 1: [Execution time: 0.008001 seconds]

Retrieve the total number of matches for each competition and rename the column as 'NumberOfGames'.

Number of Records: 43

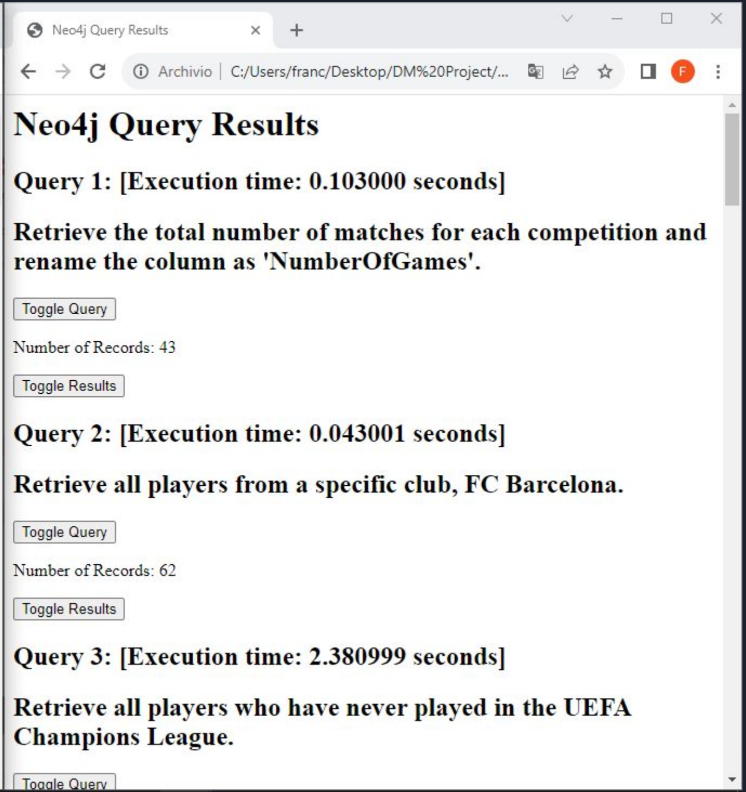
Query 2: [Execution time: 0.000999 seconds]

Retrieve all players from a specific club, FC Barcelona.

Number of Records: 62

Query 3: [Execution time: 0.077000 seconds]

Retrieve all players who have never played in the UEFA Champions League.



Neo4j Query Results

Query 1: [Execution time: 0.103000 seconds]

Retrieve the total number of matches for each competition and rename the column as 'NumberOfGames'.

Number of Records: 43

Query 2: [Execution time: 0.043001 seconds]

Retrieve all players from a specific club, FC Barcelona.

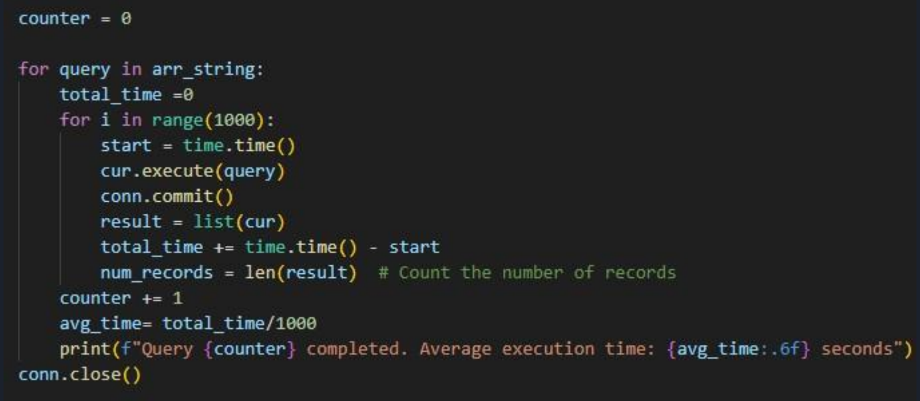
Number of Records: 62

Query 3: [Execution time: 2.380999 seconds]

Retrieve all players who have never played in the UEFA Champions League.

Python scripts: Average execution time

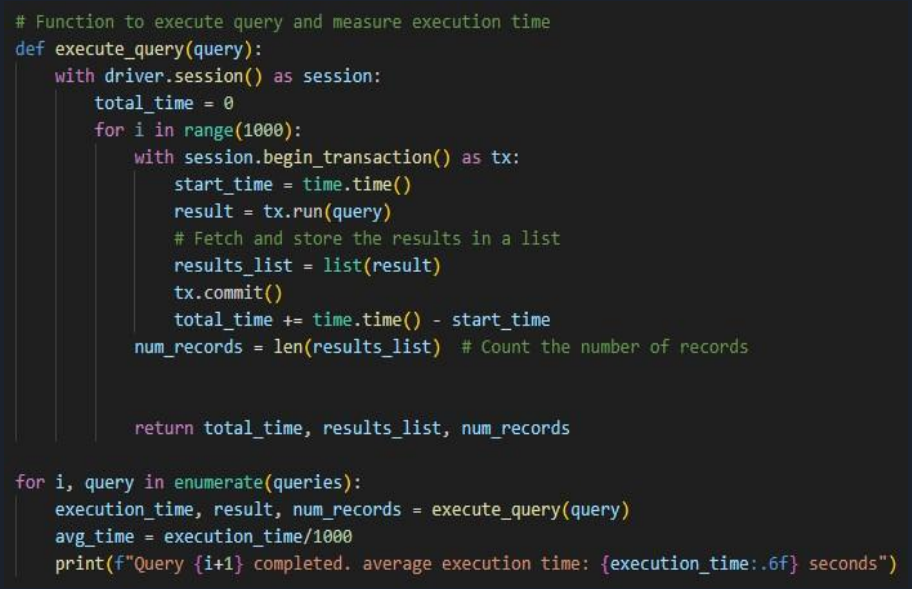
MariaDB



```python
counter = 0

for query in arr_string:
    total_time =0
    for i in range(1000):
        start = time.time()
        cur.execute(query)
        conn.commit()
        result = list(cur)
        total_time += time.time() - start
        num_records = len(result)  # Count the number of records
    counter += 1
    avg_time= total_time/1000
    print(f"Query {counter} completed. Average execution time: {avg_time:.6f} seconds")
conn.close()
```

Neo4j



```python
# Function to execute query and measure execution time
def execute_query(query):
    with driver.session() as session:
        total_time = 0
        for i in range(1000):
            with session.begin_transaction() as tx:
                start_time = time.time()
                result = tx.run(query)
                # Fetch and store the results in a list
                results_list = list(result)
                tx.commit()
                total_time += time.time() - start_time
            num_records = len(results_list)  # Count the number of records


            return total_time, results_list, num_records

for i, query in enumerate(queries):
    execution_time, result, num_records = execute_query(query)
    avg_time = execution_time/1000
    print(f"Query {i+1} completed. average execution time: {execution_time:.6f} seconds")
```

Average execution time (1/2)

Specs:

- AMD Ryzen 5800x3d
- Ram 16 GB
- GTX 1080

# Query	MariaDB	Neo4j
Query 1	8 ms	14.8 ms
Query 2	0.3 ms	18 ms
Query 3	80.9 ms	1.912253 s
Query 4	184.1 ms	637.721 ms
Query 5	1.604314 s	1.101904 s
Query 6	13.5 ms	299.801 ms
Query 7	1.365251 s	362.8 ms
Query 8	5.790 ms	9.4 ms
Query 9	49.865 ms	18.3 ms
Query 10	2.340601 s	116.9 ms

Average execution time (2/2)

Specs:

- AMD Ryzen 5800x3d
- Ram 16 GB
- GTX 1080

# Query	MariaDB	Neo4j
Query 11	1.754104 s	2.798699 s
Query 12	23.5 ms	287.551 ms
Query 13	998.044 ms	313.567 ms
Query 14	2.5874 s	1.466901 s
Query 15	69.4 ms	67.4 ms
Query 16	3.0753 s	2.754757 s
Query 17	172.201 ms	96.407 ms
Query 18	61.505777 s	8.48902 s
Query 19	48.671087 s	12.008614 s
Query 20	9.5 ms	28.7 ms

Conclusions: Neo4j wins

Mariadb

Pros:

- Fast Data Loading
- Excelled in simpler, structured queries
- Effective filtering

Cons:

- Sometimes queries are more complex to write
- Performance tapered off with more complex query
- Query Optimization Challenges

Neo4j

Pros:

- User-Friendly Queries
- Competitive Performance for Complex Graph-related Queries
- Most of the time is faster than RDBMS
- Consistent Performance

Cons:

- Filtering Challenges (Not great at filtering out useless nodes)
- Initial Loading Complexity.

Thank you