Large Scale Data Management Project

A.Y. 2024-2025

Davide Fortunato 1936575

Index

- What the project is about
- Data gathering
- Technologies used in the project
- Source Schema
- Global Schema
- Mappings
- Result of the integration and queries
- Mapping CSV data to MongoDB and Redis stores
- Redis introduction
- My hands on experience with MongoDB and Redis

What the project is about

This project intends to cover both the two parts of the Large Scale Data Management course, namely Information Integration and Big Data Management. All the following work is centered about the football domain, where I intended to gather data from different sources about players, clubs, matches and some relevant events that happen in a football match and integrate everything in a unified database where we can run insightful queries.

Data gathering

The data has been collected from Kaggle CSV datasets and one dataset from GitHub. It includes information on matches, players, clubs, and player transfers, as well as club and international competitions, such as the World Cup. Additionally, there is a dataset focused on in-game events, including yellow cards, red cards, substitutions, and more resulting in a total of 10 CSV files that will be the base for the sources of our Information Integration system. Here are the links:

- https://www.kaggle.com/datasets/abecklas/fifa-world-cup
- https://www.kaggle.com/datasets/davidcariboo/player-scores
- https://github.com/schochastics/football-data/tree/master/data/goals_time
- https://www.kaggle.com/datasets/jahaidulislam/fifa-world-cup-all-goals-1930-2022-dataset
- https://www.kaggle.com/datasets/davidantonioteixeira/ballon-dor

Technologies used in the project

- **Pentaho Data Integration**: the integration pipeline has been carried out by using this tool, more details about it in the demo.
- **PostgreSQL**: the result of the integration was materialized in a PostgreSQL database.
- **Python**: used for some scripts that allowed me to manipulate the sources and create the DBs for Redis and MongoDB.
- **Redis and MongoDB**: five CSV files have been used in order to simulate document-based sources (MongoDB) and key-value sources (Redis). More details about them in the Big Data Management demo.

Source Schema

Here and in the next slides I provide the detailed description of all the sources.

Table 1: Attributes of the Ballon d'Or Nominees CSV File

Attribute Name	Description
Year	The year in which the Ballon d'Or was awarded.
Order	The ranking of the player for that year's Ballon d'Or (e.g., "1" for the winner).
Player	The name of the nominated player.
Nationality	The nationality of the player.
Club	The club(s) the player represented during the award period (e.g., "Blackpool").
Points	The total points the player received in the voting.

Table 2: Attributes of the Clubs CSV File (Transfermarkt Dataset)

Attribute Name	Description
club_id	The unique identifier for the club (e.g., "105").
club_code	The unique code for the club (e.g., "sv-darmstadt-98").
name	The name of the club (e.g., "SV Darmstadt 98").
domestic_competition_id	The identifier for the club's domestic competition (e.g., "L1" for Ligue 1).
total_market_value	The total market value of the club (e.g., "€3.05m").
squad_size	The total number of players in the club's squad (e.g., "27").
average_age	The average age of players in the squad (e.g., "25.6").
foreigners_number	The number of foreign players in the squad (e.g., "13").
foreigners_percentage	The percentage of foreign players in the squad (e.g., "48.1").
national_team_players	The number of players in the squad who are also part of their national teams (e.g., "1").
stadium_name	The name of the club's stadium (e.g., "Merck-Stadion am Böllenfalltor").
stadium_seats	The seating capacity of the stadium (e.g., "17810").
net_transfer_record	The net transfer record for the club (e.g., "+€3.05m").
coach_name	The name of the club's coach.
last_season	The last season for which the club data is available (e.g., "2023").

Source Schema

Table 3: Attributes of the Players CSV File (Transfermarkt Dataset)

Attribute Name	Description
player_id	The unique identifier for the player (e.g., "10").
first_name	The first name of the player (e.g., "Miroslav").
last_name	The last name of the player (e.g., "Klose").
name	The full name of the player (e.g., "Miroslav Klose").
last_season	The last season the player played (e.g., "2015").
current_club_id	The unique identifier for the player's current club (e.g., "398").
player_code	The unique code for the player (e.g., "miroslav-klose").
country_of_birth	The country where the player was born (e.g., "Poland").
city_of_birth	The city where the player was born (e.g., "Opole").
country_of_citizenship	The country of citizenship of the player (e.g., "Germany").
date_of_birth	The player's date of birth (e.g., "1978-06-09 00:00:00").
sub_position	The player's sub-position on the field (e.g., "Centre-Forward").
position	The player's primary position (e.g., "Attack").
foot	The preferred foot of the player (e.g., "right").
height_in_cm	The height of the player in centimeters (e.g., "184").
contract_expiration_date	The expiration date of the player's contract.
highest_market_value_eur	The highest market value of the player in euros (e.g., "30000000").

Table 4: Attributes of the Game Information CSV File

Attribute Name	Description
id_odsp	Unique identifier for the match.
link_odsp	URL linking to detailed match statistics.
adv_stats	Boolean indicating whether advanced statistics are available for the match (TRUE or FALSE).
date	The date when the match was played, in YYYY-MM-DD format.
league	Code representing the league (e.g., D1 for Bundesliga, F1 for Ligue 1).
season	Season year for the league (e.g., 2012).
country	The country in which the league is based (e.g., Germany, France).
ht	The name of the home team.
at	The name of the away team.
fthg	Full-time goals scored by the home team.
ftag	Full-time goals scored by the away team.
odd_h	Odds for a home team win.
odd_d	Odds for a draw.
odd_a	Odds for an away team win.
odd_over	Odds for the total goals being over a specified threshold (not available in this dataset).
odd_under	Odds for the total goals being under a specified threshold (not available in this dataset).
odd_bts	Odds for both teams scoring (not available in this dataset).
odd_bts_n	Odds for neither team scoring (not available in this dataset).

Source Schema

Table 5: Attributes of the Game Events CSV File

Attribute Name	Description
id_odsp	Unique identifier for the match, linking it to the main match dataset.
id_event	Unique identifier for the event within a match.
sort_order	The sequence number of the event in chronological order.
time	The minute in which the event occurred during the match.
text	A textual description of the event.
event_type	The main type of the event (e.g., 1 for attempts, 2 for goals).
event_type2	Additional classification for the event type (e.g., 12 for assist).
side	Indicates the side of the field where the event occurred (e.g., 1 for left, 2 for right).
event_team	The team associated with the event.
opponent	The opposing team during the event.
player	Name of the primary player involved in the event.
player2	Name of the secondary player involved in the event (e.g., assist provider).
player_in	Name of the player substituted in during the event (if applicable).
player_out	Name of the player substituted out during the event (if applicable).
shot_place	Location where the shot was taken (e.g., 6 for left side of the box).
shot_outcome	Outcome of the shot (e.g., 2 for missed, 1 for scored).
is_goal	Boolean indicating whether the event resulted in a goal (1 for yes, 0 for no).
location	General location of the event on the field (e.g., 9 for left side).
bodypart	The body part used in the event (e.g., 2 for left foot).
assist_method	Method of the assist (e.g., 1 for through ball).
situation	Type of situation during the event (e.g., 1 for open play).
fast_break	Boolean indicating whether the event was part of a fast break (1 for yes, 0 for no).

Table 6: Attributes of the Player Transfers CSV File

Attribute Name	Description
player_id	Unique identifier for the player involved in the transfer.
transfer_date	Date when the transfer occurred.
transfer_season	Season during which the transfer took place (e.g., 25/26 for 2025-2026).
from_club_id	Unique identifier for the club from which the player was transferred.
to_club_id	Unique identifier for the club to which the player was transferred.
from_club_name	Name of the club from which the player was transferred.
to_club_name	Name of the club to which the player was transferred.
transfer_fee	The fee paid for the transfer of the player (0.000 indicates a free transfer).
market_value_in_eur	Market value of the player at the time of the transfer, in euros.
player_name	Full name of the player involved in the transfer.

Source Schema

Table 7: Attributes of the Match Results CSV File

Attribute Name	Description
home_team	Name of the team that played the match as the home team.
away_team	Name of the team that played the match as the away team.
home_score	Number of goals scored by the home team in the match.
away_score	Number of goals scored by the away team in the match.
date	Date when the match was played, formatted as YYYY-MM-DD.
competition	Name of the competition in which the match took place.

Table 8: Attributes of the International Football Match CSV File

Attribute Name	Description
date	Date when the match was played, formatted as YYYY-MM-DD.
home_team	Name of the team that played as the home team in the match.
away_team	Name of the team that played as the away team in the match.
home_score	Number of goals scored by the home team.
away_score	Number of goals scored by the away team.
tournament	Name of the tournament in which the match was played (e.g., Friendly).
city	City where the match was held.
country	Country where the match took place.
neutral	Boolean value indicating whether the match was played on neutral ground (TRUE or FALSE).

Source Schema

Table 9: Attributes of the International Football Goals CSV File

Attribute Name	Description
date	Date of the match in which the goal was scored, formatted as YYYY-MM-DD.
home_team	Name of the team that played as the home team.
away_team	Name of the team that played as the away team.
team	Name of the team that scored the goal.
scorer	Name of the player who scored the goal.
minute	Minute of the match in which the goal was scored.
own_goal	Boolean value indicating if the goal was an own goal (TRUE or FALSE).
penalty	Boolean value indicating if the goal was scored from a penalty kick (TRUE or FALSE).

Table 10: Attributes of the Competitions CSV File

Attribute Name	Description
competition_id	Unique identifier for the competition.
competition_code	Code representing the competition (e.g., "italy-cup").
name	Name of the competition.
sub_type	Sub-type of the competition, such as "domestic_cup".
type	Overall type of the competition (e.g., "domestic_cup").
country_id	Numerical identifier for the country hosting the competition.
country_name	Name of the country hosting the competition (e.g., "Italy").
domestic_league_code	Code representing the associated domestic league.
confederation	Confederation to which the competition belongs (e.g., "europa").
url	URL providing detailed information about the competition.

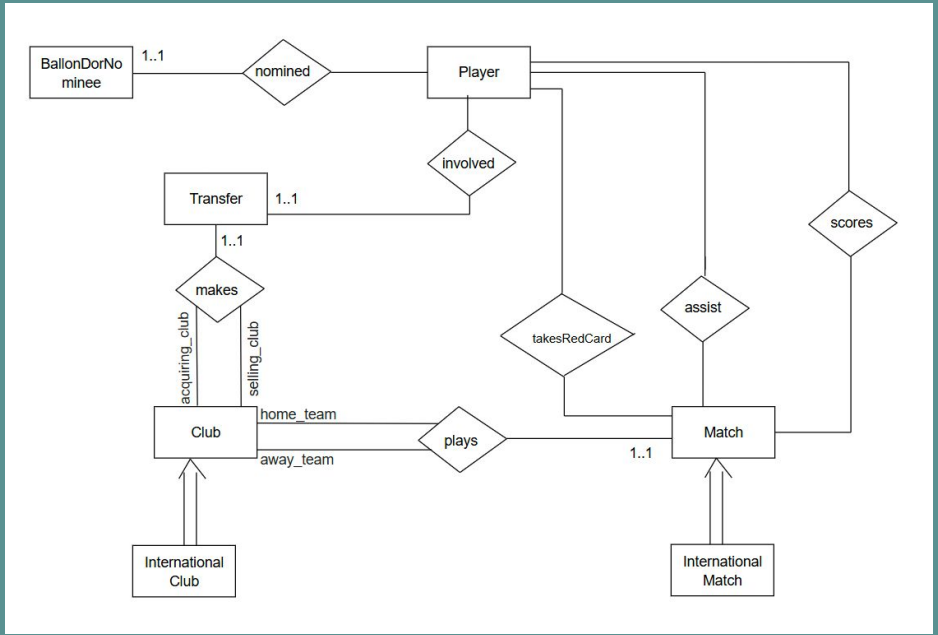
Source Schema

Obviously, not all attributes have been used and this will be seen in the demo. Given the high number of attributes per table, from now on, referring to the table numbers indicated above the table itself, I will refer to the sources table in the following way:

- $T1(\vec{a}) \leftarrow BallonDor(\vec{a})$
- $T2(\vec{a}) \leftarrow ClubsTFK(\vec{a})$
- $T3(\vec{a}) \leftarrow PlayersTFK(\vec{a})$
- $T4(\vec{a}) \leftarrow GameInfo(\vec{a})$
- $T5(\vec{a}) \leftarrow GameEvents(\vec{a})$
- $T6(\vec{a}) \leftarrow Transfers(\vec{a})$
- $T7(\vec{a}) \leftarrow MatchResults(\vec{a})$
- $T8(\vec{a}) \leftarrow InternationalMatches(\vec{a})$
- $T9(\vec{a}) \leftarrow InternationalGoals(\vec{a})$
- $T10(\vec{a}) \leftarrow CompetitionsTFK(\vec{a})$

Global Schema

Below is an ER schema (simplified without attributes) representing the conceptual Global Schema, which helps illustrate the key entities and their relationships, providing a clear overview of the reconciled view's structure.



Global Schema

Now, I present the logical schema of the reconciled view, which will be translated into the final result as PostgreSQL tables.

- **Club**(id, name, stadium_name, stadium_seats, country)
- **Player**(id, name, country, birth, position, foot, height, highest_market_value)
- **PlayerTakesRedCard**(player_id, match_id, date)
- **Match**(id, home_club_id, away_club_id, date, international_match, home_score, away_score)
- **PlayerScores**(player_id, match_id, minute, penalty)
- **PlayerNominedBD**(player_id, year, position, club_id)
- **PlayerAssist**(player_id, match_id, minute)
- **Transfer**(id, from_club_id, acquiring_club_id, player_id, cost, date)

Mappings dependencies

Let's dive into the mappings assertions, whose aim is defining the correspondence between the global schema and the local data sources. In my project, they consist of LAV (Local-as-View) and GLAV (hybrid approach among LAV and Global-as-View) mappings, so we will have at least one mapping assertion for each source. Let's break them down:

- Mapping on Ballon D'Or source

$$\forall \vec{t}.\, T1(\vec{t}) \rightarrow \exists c_1, c_2, c_3, c_4, c_5, c_6, b_1, b_2, b_3, b_4. \quad (1)$$
$$Player(c_1, t_3, t_4, c_2, c_3, c_4, c_5, c_6)$$
$$\wedge\, PlayerNominedBD(c_1, t_1, t_2, b_1)$$
$$\wedge\, Club(b_1, t_5, b_2, b_3, b_4)$$

Notice that source attributes are represented using vector notation, where each specific attribute (e.g. t_4) corresponds to its position in the order defined by the tables in the source schema.

Mappings

- Mapping on ClubsTFK and CompetitionsTFK sources

$$\forall \vec{t}, \vec{a}.\, (T2(\vec{t}) \wedge T10(\vec{a}) \wedge t_4 = a_1) \rightarrow \exists c_1 Club(c_1, t_3, t_{11}, t_{12}, a_7) \qquad (2)$$

- Mapping on PlayersTFK source

$$\forall \vec{t}.\, T3(\vec{t}) \rightarrow \exists c_1 .\, Player(c_1, t_4, t_{10}, t_{11}, t_{13}, t_{14}, t_{15}, t_{17}) \qquad (3)$$

Mappings

- Mapping on GameInfo source

$$\forall \vec{t}.\, T4(\vec{t}) \rightarrow \exists c_1, c_2, c_3, c_4, b_1, b_2, b_3, b_4, m_1. \quad (4)$$
$$Club(c_1, t_8, c_2, c_3, c_4) \wedge$$
$$Club(b_1, t_9, b_2, b_3, b_4) \wedge$$
$$Match(m_1, c_1, b_1, t_4, \text{"0"}, t_{10}, t_{11})$$

- Mappings on GameEvents source

$$\forall \vec{t}, \vec{a}.\, (T5(\vec{a}) \wedge t_{17} = \text{"1"} \wedge T4(\vec{t}) \wedge a_1 = t_1) \rightarrow \exists c_1, c_2, c_3, c_4, c_5, c_6, c_7, \quad (5)$$
$$l_1, l_2, l_3, l_4, l_5, l_6, l_7$$
$$Player(c_1, a_{12}, c_2, c_3, c_4, c_5, c_6, c_7) \wedge$$
$$Player(l_1, a_{11}, l_2, l_3, l_4, l_5, l_6, l_7) \wedge$$
$$\exists f_1, f_2, f_3, f_4, b_1, b_2, b_3, b_4, m_1.$$
$$Club(f_1, t_5, f_2, f_3, f_4) \wedge$$
$$Club(b_1, t_6, b_2, b_3, b_4) \wedge$$
$$Match(m_1, f_1, b_1, t_4, \text{"0"}, t_{10}, t_{11}) \wedge$$
$$PlayerAssist(c_1, m_1, a_4) \wedge$$
$$PlayerScores(l_1, m_1, a_4, \text{"0"})$$

$$\forall \vec{t}, \vec{a}.\, (T5(\vec{a}) \wedge t_6 = \text{"6"} \wedge T4(\vec{t}) \wedge a_1 = t_1) \rightarrow \exists c_1, c_2, c_3, c_4, c_5, c_6, c_7. \quad (6)$$
$$Player(c_1, a_{12}, c_2, c_3, c_4, c_5, c_6, c_7) \wedge$$
$$\exists f_1, f_2, f_3, f_4, b_1, b_2, b_3, b_4, m_1.$$
$$Club(f_1, t_5, f_2, f_3, f_4) \wedge$$
$$Club(b_1, t_6, b_2, b_3, b_4) \wedge$$
$$Match(m_1, f_1, b_1, t_4, \text{"0"}, t_{10}, t_{11}) \wedge$$
$$PlayerTakesRedCard(c_1, m_1, t_4)$$

Mappings

- Mapping on TransfersTFK source

$$\forall \vec{t}, \vec{a}.\ (T3(\vec{a}) \wedge T6(\vec{t}) \wedge a_1 = t_1) \rightarrow \exists c_1, c_2, c_3, c_4, c_5, c_6, c_7. \quad (7)$$
$$Player(c_1, a_4, a_{10}, a_{11}, a_{13}, a_{14}, a_{15}, a_{17}) \wedge$$
$$\exists f_1, f_2, f_3, f_4, b_1, b_2, b_3, b_4, m_1$$
$$Club(f_1, t_6, f_2, f_3, f_4) \wedge$$
$$Club(b_1, t_7, b_2, b_3, b_4) \wedge$$
$$Transfer(m_1, f_1, b_1, c_1, t_8, t_2)$$

- Mapping on MatchResults source

$$\forall \vec{t}.\ T7(\vec{t}) \rightarrow \exists c_1, c_2, c_3, c_4, b_1, b_2, b_3, b_4, m_1. \quad (8)$$
$$Club(c_1, t_1, c_2, c_3, c_4) \wedge$$
$$Club(b_1, t_2, b_2, b_3, b_4) \wedge$$
$$Match(m_1, c_1, b_1, t_5, \text{"0"}, t_3, t_4)$$

Mappings

- Mapping on InternationalMatches source

$$\forall \vec{t}.\, T8(\vec{t}) \rightarrow \exists c_1, c_2, c_3, b_1, b_2, b_3, m_1. \quad (9)$$
$$Club(c_1, t_2, c_2, c_3, t_2) \wedge$$
$$Club(b_1, t_3, b_2, b_3, t_3) \wedge$$
$$Match(m_1, c_1, b_1, t_1, \text{"1"}, t_4, t_5)$$

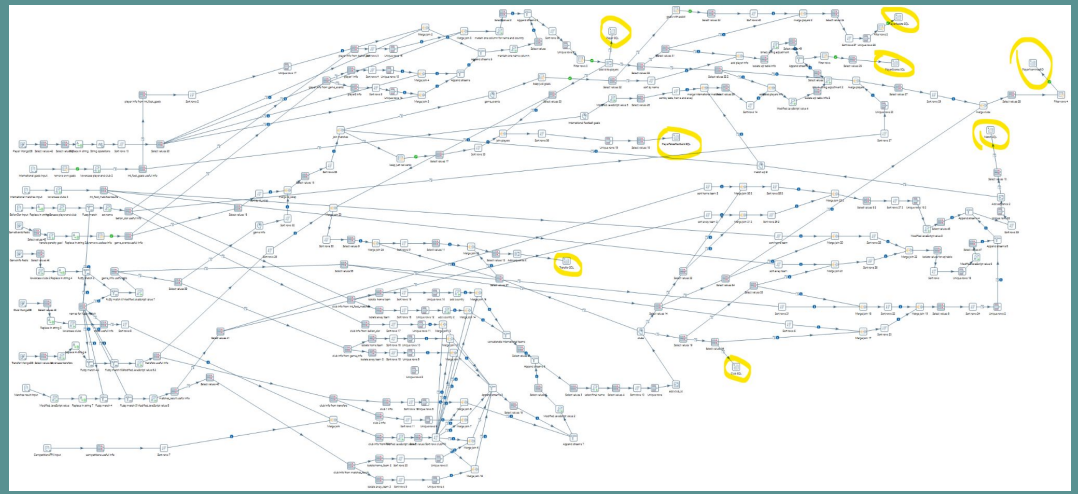
- Mapping on InternationalGoals source

$$\forall \vec{t}.\, T9(\vec{t}) \rightarrow \exists c_1, c_2, c_3, c_4, c_5, c_6, b_1, b_2, b_3, m_1. \quad (10)$$
$$Player(c_1, t_5, t_4, c_2, c_3, c_4, c_5, c_6) \wedge$$
$$Club(d_1, t_2, c_2, c_3, t_2) \wedge$$
$$Club(b_1, t_3, b_2, b_3, t_3) \wedge$$
$$Match(m_1, c_1, b_1, t_1, \text{"1"}, t_4, t_5) \wedge$$
$$PlayerScores(c_1, m_1, t_6, t_8)$$

Result of the integration and queries

As already anticipated, I materialized the result of the Information Integration System in a PostgreSQL database, which, in the context of our course, represents a **Solution** of the data-exchange setting composed by the Source Schema, Target Schema and Mappings previously defined . However, what I got is not a **Universal Solution** since for missing information I used SQL nulls, which are different from the **marked nulls** seen in the course.

We can now proceed to demonstrate through a demo of the application some examples of interesting queries, where, just to remember, our goal is to compute the certain answers to the queries.



The image shows the Pentaho pipeline that led to the final population of the PostgreSQL tables, circled in yellow.

Mapping CSV data to MongoDB and Redis

To incorporate the Big Data Management aspect of the course, I developed a Python script to convert CSV sources from the Transfermarkt dataset into JSON files, making them compatible with MongoDB for document-based storage.

For Redis, I populated the key-value database directly from a Python script. Then, I used another script to extract data from Redis and transform it into JSON files.

Obviously, in the integration pipeline defined in Pentaho I used the JSON version of the sources.

Redis introduction

Since we have already covered an introduction to MongoDB in the course, I decided to skip it and focus on Redis. Redis is primarily a key-value store, but it offers much more functionality than a simple key-value database. Unlike traditional key-value stores that only support plain string values (as shown in the slides), Redis allows structured data types such as lists, sets, sorted sets, and hashes. Additionally, Redis can function as a message broker, enabling pub/sub messaging.

A fundamental characteristic of Redis is its exceptional speed, which sets it apart from traditional databases. This speed is primarily due to its nature as an in-memory data store, meaning that all data is stored directly in RAM rather than on disk. Unlike disk-based databases, which require frequent read and write operations to persistent storage, Redis can access and manipulate data almost instantaneously. This results in extremely fast key lookups and low-latency operations, making it highly efficient for applications requiring real-time performance.

Because of its design, Redis is particularly well-suited for scenarios where data retrieval must be as fast as possible, and complex querying or relational joins are not necessary. Instead of executing expensive SQL queries with multiple table joins, Redis operates using simple key-based lookups, hash-based indexing, and lightweight operations on structured data types. This makes it an ideal choice for use cases that involve frequent read and write operations, minimal computational overhead, and data that fits within available memory.

However, it is important to note that since Redis primarily relies on RAM, it does not inherently provide persistent storage in the same way as traditional databases. While Redis does offer persistence options (such as RDB snapshots and AOF logging), its core strength lies in its ability to serve high-speed, transient data, making it an excellent choice for caching, real-time analytics, and session management.

Redis introduction

Redis Data Type	Contains	Read/write ability
String	Binary-safe strings (up to 512 MB), Integers or Floating point values, Bitmaps.	Operate on the whole string, parts, increment/decrement the integers and floats, get/set bits by position.
Hash	Unordered hash table of keys to string values	Add, fetch, or remove individual items by key, fetch the whole hash.
List	Doubly linked list of strings	Push or pop items from both ends, trim based on offsets, read individual or multiple items, find or remove items by value.
Set	Unordered collection of unique strings	Add, fetch, or remove individual items, check membership, intersect, union, difference, fetch random items.
Sorted Set	Ordered mapping of string members to floating-point scores, ordered by score	Add, fetch, or remove individual items, fetch items based on score ranges or member value.
Geospatial index	Sorted set implementation using geospatial information as the score	Add, fetch or remove individual items, search by coordinates and radius, calculate distance.
HyperLogLog	Probabilistic data structure to count unique things using 12Kb of memory	Add individual or multiple items, get the cardinality.(http://antirez.com/news/75)

My hands on experience with MongoDB and Redis

I explored the two data models using their respective graphical interfaces: MongoDB Compass for MongoDB and Redis Insight for Redis. For Redis, I performed simple queries and navigated the data sources I structured as a key-value store, highlighting the strengths and limitations of this model. In contrast, MongoDB Compass provided a more intuitive and feature-rich interface, allowing for seamless data exploration and execution of complex queries. As part of the demonstration, I designed an advanced aggregation pipeline to showcase a fundamental concept in the NoSQL paradigm: aggregation.

Thanks for the attention!