BUILD: Project III / Design & Implement a Key-Value In-Memory Database

1. Business Requirement

Our client is a sport event statistics committee who would like to have a database with information of Olympic Games during the past 120 years. Information they are interested in includes how many athletes participated in a game, how many and teams attended in a game, the number of medals for each team in a game, the athletes' information like their age, weight, and height, the number of athletes in an event, how many athletes have participated in multiple Olympic Games, which athlete won the most medals in the history of Olympic Games in a certain event, the general trend of athlete's body information along the years, as well as types of sports in different Olympic Games.

Nouns (In Red):

athletes
teams
game
medals
age
weight
height
event
types
sports
body information

Verbs (In Blue):

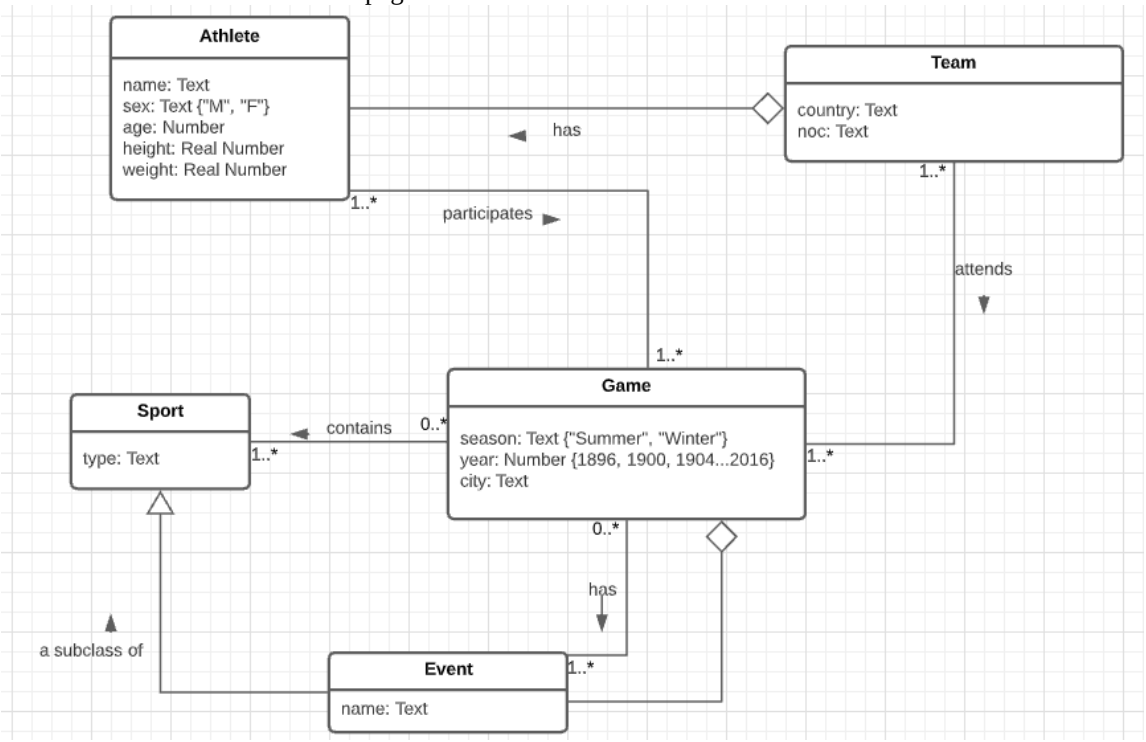
participated
attended
won

Business Rules

- A team could participate in multiple Olympic Games
- A team must participate in 1 Game to be recorded in the database.
- A team attending a Game must have at least 1 athlete.
- An athlete must participate in 1 Olympic Game to be recorded in the database.
- An athlete could participate in multiple events in an Olympic Games.
- An athlete could win several medals in 1 Game.
- An athlete could participate in Olympic Games for different teams, for instance Mary could participated in 1992 Summer with Team US and with Team Japan in 1996 Summer.
- A Sport type could have multiple events
- An event must belong to a Sport type.
- A Game must have multiple teams participating.

2. Conceptual Model in UML

https://lucid.app/lucidchart/47f13deb-0c7c-49cc-9430-6288f6ab24e9/edit?invitationId=inv_6602d40b-2285-42b7-b5b8-753d12cb2c87&page=sAIlboIWdedJ#



3. Added Functionalities Using Redis

Based on Project 2, where Mongodb was used as the main database, I will support a new function that allows users to get all the Olympic games an athlete has participated, and support CRUD the participation information in the database using Redis. After Redis makes the changes, it will update Mongodb database correspondingly.

Here, Redis serves as a cache to store only the athlete's participation information to make the query and update super-fast. The main intention behind the design aims to help data analytics or specialists who need to work on participation data frequently to do their job faster than working with Mongodb, where a lot of information is nested and requires aggregations to get participation information.

4. Redis Data Structures Used

I used 3 data structures in Redis: Sorted Set, Hash, and String.

- Sorted Set: "athleteName" as the key, "gameYear" as score and "gameYear gameSeason" as value. This way, particpation data could be represented in a chronological order.

- Hash: “gameId” as the key, “gameSeason” and “gameYear” as fields in the hash. This hash could help quickly get the information to populate the above sorted set.

- String, “gameYear gameSeason” as key and “gameId” as value for a fast fetch of gameId when deleting a participation in *Mongodb*.