CIS 450 Final Project Writeup

Team: Lawrence Choi, Kristen Gee, Ryan O'Gorman
December 16th, 2016

Abstract

Our application uses the Olympic dataset to present a comprehensive view of Olympic achievement focused primarily at the individual level: we want users to be able to easily find data on any given individual athlete's Olympic achievements and background. Our application also places a secondary emphasis on nations, and provides users with data on aggregate national Olympic performance, as well as country facts.

Modules and Architecture

- /index - Home page that includes Olympic logo and brief abstract.
- /athletesearch - Page with a search bar that allows the user to type in a name or portion of a name of an athlete and search for it in the database.
- /athleteresults - Results table listing all athletes that match the search query.
- /athlete/:id - Athlete information page that displays athlete data based on the given id.
- /countrysearch - Page with a search bar that allows the user to type in a name or portion of a name of a country and search for it in the database.
- /countryresults - Results table listing all countries that match the search query.
- /country/:code - Country information page that displays country data based on the given country code.
- app.js - Handles all HTTP requests and routing. Also handles setting up connections to our databases on AWS, creating and sending queries to the databases, and responding to the requests with the appropriate data.

SQL Schema

Athlete(<u>id</u>, name, gender, country)
Sport(<u>id</u>, sport, discipline, event, eventGender)
Medal(<u>athleteID</u>, <u>sportID</u>, <u>event</u>, city, <u>year</u>, <u>type</u>)
Country(<u>countryCode</u>, name, numAthletes, numMedals)

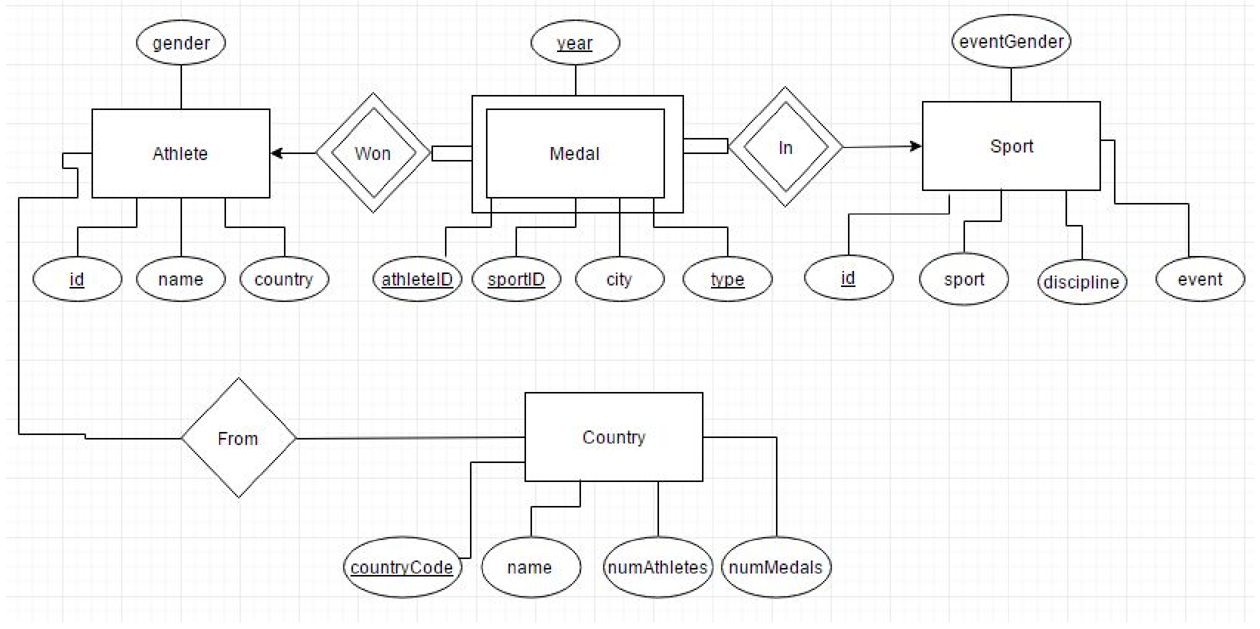
```
CREATE TABLE Athlete (
        id INTEGER PRIMARY KEY,
        name VARCHAR(255),
        gender VARCHAR(6),
        country VARCHAR(255)
);

CREATE TABLE Sport (
	id INTEGER PRIMARY KEY,
	sport VARCHAR(255),
	discipline VARCHAR(255),
	event VARCHAR(255),
	eventGender VARCHAR(1)
);

CREATE TABLE Medal (
	athleteID INTEGER,
	sportID INTEGER,
	city VARCHAR(255),
	year INTEGER,
	type VARCHAR(6),
	PRIMARY KEY (athleteID, sportID, year, type),
	FOREIGN KEY (athleteID) REFERENCES Athlete(id) ON DELETE CASCADE,
	FOREIGN KEY (sportID) REFERENCES Sport(id) ON DELETE CASCADE
);

CREATE TABLE Country (
	countryCode VARCHAR(3) PRIMARY KEY,
	name VARCHAR(255),
	numAthletes INTEGER,
	numMedals INTEGER
);
```

ER Diagram



Complementary Sources

- ➢ Supplemental country data like population, continent, GDP, and images of the national flag and the country's location on the globe was extracted from Wikidata through the MediaWiki API. This information was stored in our MongoDB instance.
- ➢ Supplemental athlete data, namely the athlete's photo and date of birth, was parsed from Wikipedia (if the athlete had a Wikipedia page). A Node module called wiki-infobox was used to retrieve this data from the infobox on an athlete's Wikipedia page, given the athlete's name as a search key.

Data Instance Use

- ➢ /athletesearch - Gets the user input and passes this to /athleteresults.
- ➢ /athleteresults - Breaks the search query into a list of words and, for each word, creates a LIKE condition that matches any string containing that word. Then it ANDs all of the conditions together. With this as the WHERE clause in the query, this route queries the Athlete table in our Oracle database to get the id and name of every athlete whose name matches the condition.
- ➢ /athlete/:id - Queries data based on an athlete's id from the Athlete and Medal tables in our Oracle database.
- ➢ /countrysearch - Gets the user input and passes the query to /countryresults.
- ➢ /countryresults - Builds a query in the same way as /athleteresults, except it queries the Country table using the name column for the condition, and returns the country code and name of each country to be rendered on the page.
- ➢ /country/:code - Queries data based on a country's code and gets data from the Country table that is stored in Mongodb. It also queries the Oracle table Country to get the number of athletes and medals for the country.

Data Cleaning and Import Mechanism

- ➢ Data Cleaning: We cleaned the data in our csv files using Excel worksheet functions. To give a few examples, hyphenated last names did not have an uppercase letter following the hyphen, which affected our ability to query Wikipedia pages. Excel's PROPER() function helped us resolve this issue. Furthermore, last names were made sentence case for aesthetic purposes.
- ➢ Import Mechanism: We utilized a range of different Excel functions to create and assign values to various attributes like athleteID, sportID, numAthletes, numMedals, etc. We then used SQL Developer to import the custom data, stored in csv files, into our tables.

Algorithms and Communication Protocols

One specific algorithm we had to employ in our application was the MD5 hashing algorithm. When parsing athlete image data from the infoboxes on Wikipedia, we were able to extract the actual name of the image file but did not have access to the Wikimedia upload URL. We found that we were able to hardcode the initial part of the URL, https://upload.wikimedia.org/wikipedia/commons/, but there were some subdirectories that followed that varied with every image. After doing some research, we found that these subdirectories corresponded to the first and first two characters of the MD5 hash of the image filename. So, we found a node module to efficiently compute MD5 hashes, and after some string manipulations we were able to display athlete photos on their pages.

Use Cases

- ➢ The 'Search for Athlete' feature can be used to search for any Olympic athlete. Inputting any part of an athlete's name will return a table of possible athletes the user may have been searching for. Once a given athlete is selected, our application takes the user to a page that queries the athlete's medals and country from the database, and extracts the athlete's birth date and photo from Wikipedia. From an athlete's page, a user can learn more about the nation the athlete is from by clicking on the link to the country page.
- ➢ The 'Search for Country' feature can be used to search for any Olympic country. A country page contains information about a given country such as its population, continent, area, GDP, and Gini coefficient. The country page also contains images of the national flag and the country's location on the globe. Country data was extracted from Wikidata.

Optimization Techniques Employed

We did not employ any optimization techniques other than the default indices on our primary keys, as we perceived our application to run at a satisfactory speed in all cases and felt that further optimization was not necessary.

Technical Specifications

- ➢ Node.js - to write the server for our web application
- ➢ Amazon Web Services - to host our databases
- ➢ Oracle - to store and access athlete data from the Summer Olympics dataset
- ➢ MongoDB - to store and access country data extracted from Wikidata
- ➢ Bootstrap - to design web pages

Special Features

The search bar does not require the user to search the full name of an athlete or a country and also does not require the words to be in any sort of order.

Potential Future Extensions

One idea for additional features involves drawing some basic insights between national Olympic performance and some of the statistics in our supplemental country data. For example, we could include graphs to better visualize the data and make comparisons with other countries, and perform correlation calculations or regression analyses to find if there are any variables that relate to Olympic performance with strong statistical significance. We could also aggregate data based on other factors such as sports or the year of the specific Olympic games. Furthermore, another possible extension could be extracting more information from athletes' infoboxes like hometown, height, and weight in order to open up the door to even more statistical tests.

Division of Work Between Group Members

Ryan worked on querying from our databases and presenting the data from a user's search. Lawrence's responsibility was setting up the backend infrastructure for the server, SQL, and noSQL databases. Kristen focused on the frontend and presenting the data on the athlete and country pages. We all agree that the work was fairly and evenly divided between the three of us. We enjoyed working on this project and having the opportunity to learn and use new technologies!