Track Project

Avery Mariol | Huntington University | CS 415 Database Management – Fall 2025 - Final Project

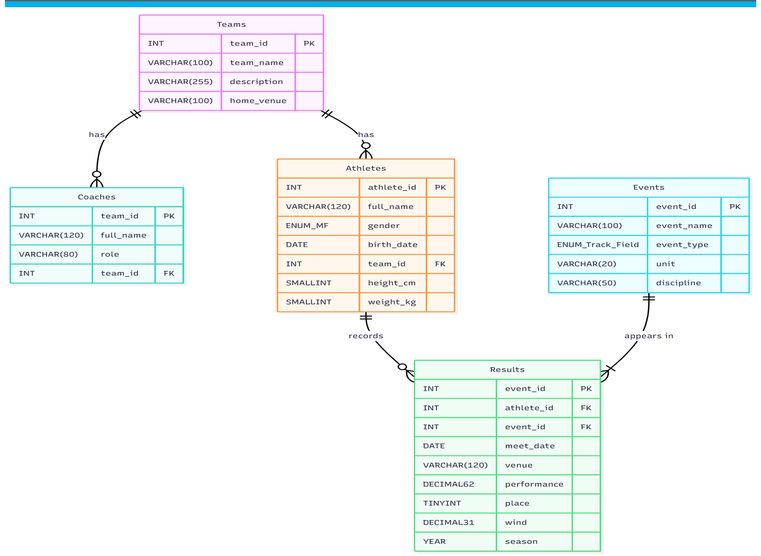
Overview

- This project designs a relational database for a Track and field coaching staff to manage athletes, competitive events and performance results. The database provides a structured way to track athlete profiles, record performances, organize teams and analyze meet results.
- From a user's perspective, coaches are able to look up an athlete, view their season best times or distance, compare performance across events, and generate statistical reports to assist with training and competition decisions.

Design

- The database includes five core tables with normalized relations, Teams, Coaches, Athletes, Events and results. Each table stores a category of information to avoid redundancy and have consistent data. Athlete and event information are separated from performance data which prevents repeated storage of athlete names or event properties in the results table.
- Normalization led to multiple design decisions, A single results table functions as a join table connecting athletes and events while including meet related information such as date, venue, wind and place. Although meet data could be separated into its own table, keeping this data inside the results make queries simpler and aligns with the idea of the project, the team relationships are one to many, meaning each team can have multiple athlete and coaches.

ER Model



Data

- 6 teams representing high school, club, and college programs
- 6 coaches with multiple disciplines
- 30 athletes with gender, Birthdate, height, weight and assigned team
- 10 events including sprints, distances runs, jumps and throws
- 120 performance results of data across many meets

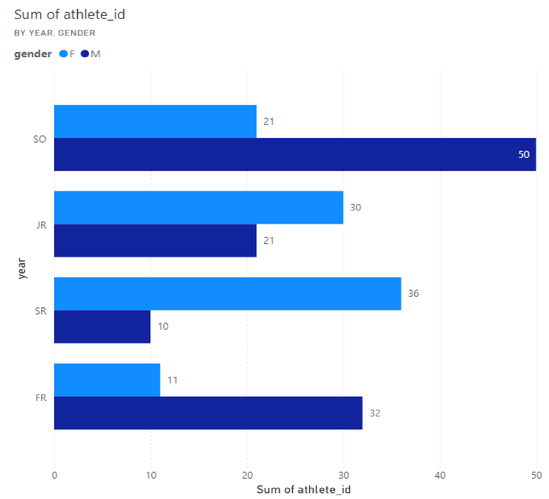
Queries

- SELECT athlete_id, first_name, last_name, gender, year, team_id
  -> FROM athlete
  -> ORDER BY year, last_name;
- SELECT athlete_id, CONCAT(first_name, ' ', last_name)AS full_name FROM athlete;
- SELECT
  -> first_name,
  -> last_name,
  -> LEFT(first_name, 1) AS first_initial
  -> FROM athlete;
- SELECT team_id, COUNT(*) AS athlete_count FROM athlete GROUP BY team_id HAVING COUNT(*) > 10;
- SELECT a.first_name, e.event_name, r.performance FROM result r
  -> JOIN athlete a ON r.athlete_id = a.athlete_id
  -> JOIN event e ON r.event_id = e.event_id
  -> JOIN team t ON a.team_id = t.team_id;
- SELECT t.team_name, a.first_name AS athlete FROM team t LEFT JOIN athlete a ON t.team_id = a.team_id;
- UPDATE athlete SET year = 'SR' WHERE athlete_id = 10;
- DELETE FROM result WHERE result_id = 200;
- CREATE VIEW upperclassman AS SELECT athlete_id, first_name, last_name, year FROM athlete WHERE year IN ('JR', 'SR');
- START TRANSACTION;

UPDATE athlete SET year = 'SR' WHERE athlete_id = 10;

ROLLBACK;

Reports



This chart report was created using power BI which imported directly from the MariaDB tables used in my project. Power BI automatically recognized the categories and allowed me to build this chart comparing the number of male and female athletes for each academic year. The chart aggregates athlete records using a “Count of athlete_id” grouped by both year and gender. This report is valuable for coaches because it provide a visual breakdown of roster composition for class years as well as gender distribution on a team to help athletic directors maintain balanced teams and future planning. This report is a good way to show a roster analysis.

Future Work

- Add a dedicated meets table for normalization
- Add relay team compositions
- Store injury history or training logs
- Implement a wed front end for coaches
- Add automatic import from meet timing systems

Works Cited

- MariaDB
- Python
- Excel
- GitHub