Swimming Information Database Final Report

CSSE333 Databases Spring 2023

Blaise Swartwood

Brian Beasley

Ben Graham

X 

Blaise Swartwood

X 

Brian Beasley

X 

Ben Graham

Table Of Contents

Executive Summary

This final report document starts with a description of the problem that this project is solving. Then, the specific solution to address the problem, major challenges of the solution, and the design of the database will then follow. The implementation will then be analyzed for its strengths and weaknesses and possible improvements will be noted. The appendix near the end of the document will detail the final design of the database with the final entity relationship diagram and relational schema shown. Lastly, the glossary, external list, and index follow.

Introduction

The purpose of this document is to provide the final report for the swimming information database. The overall implementation and design of the solution to the problem will be detailed. Any changes compared to the initial problem statement and security analysis will be described and explained for changes. By the end of this document, a full description and understanding of the final project should be delivered.

Main Content

A. Problem Statement

A large part of competitive swimming is achieving certain time levels and cuts to compete at national swim meets [1]. While typical time standards are posted for swimmers to achieve, there is no easy way to compare one's best times to these time standards in a simple manner. Furthermore, many new swimmers entering the competitive scene need to buy a racing suit, also known as a tech suit [3]. However, there are many different tech suits available to buy and it is not easy to pick out which suit will be optimal for a swimmer. Since these suits cost around $200-500, this is not a cheap decision to make [3]. More data needs to be tracked to allow newer swimmers to determine which suit they should probably buy.

Ultimately, this project seeks to solve these two issues by allowing easy comparison of times to different time standards and tracking the suits that swimmers wear during their best time races so that new swimmers can see which suits are commonly used for their events.

B. Solution Description

The solution to this problem is to create a swimming information database that will track basic information about swimmers and allow for the two main features described above. The functionality for this database taken from the final problem statement follows.

Functionality:

1. Ability to track swimmers, swimmer's best times, standardized times, and equipment used in races
2. Ability to view time difference between a specific swimmer's best times and closest standardized times
3. Ability to view the tech suit worn during a best time swim
4. Ability to search for one specific event of a swimmer
5. Ability for coaches and admins to search up information about and the best times of every swimmer.
6. Ability to keep track of which teams swimmers are on and their information for that team
7. Ability to simply view all time standards for every event
8. Ability for users to login
9. Ability for users to update their best times or add new best times
10. Ability for maintainers to update time standards and add new pieces of equipment
11. Ability for swimmers and coaches to change their personal information as well as information about their team membership

Front-End Discussion

The front-end of the database is implemented using Java Swing graphics. The graphical user interface (GUI) is designed to be as simple and intuitive as possible. There is an initial login page when the program is started. New users have the option to register themselves to the database by giving some personal information, their status as a swimmer or coach, their current team, and their designed login information. From there, users can then login to the database. If accurate login information is given, the user will then be directed to the search page of the database. If logging in as a swimmer, their personal information will be displayed, and they can search for their own times and compare their times to time standards in the database. They have the option to view all the time standards in the database. Furthermore, they can edit their information inside the database by opening a new page. There are three options to choose from on this page: add, remove, and update. After a best new time, swimmers can add that race and time into the database, along with the suit they used to achieve that time (to track suit information). Users can also add themselves to more teams if they compete for several different teams. Users can also delete the best time if they do not want to have that event tracked, and they can update their best times, personal information, and information with the team they are a part of. When logging in as a coach, the same basic functionality will be given to the coach, except that they can also search for any swimmer's times, and they can remove swimmers from teams.

Back-End Discussion

The back end of the database is implemented using Microsoft SQL Server. All the information for swimmers, events, times, races, equipment, and more is stored with tables in the database. Data is taken from the database and strung together to provide a seamless and fluid experience for users. More details of the back-end functionality will be given later in the database design section.

C. Key Challenges

Challenge: The database was designed initially without taking account user logins and having different features for admins and coaches.

Solution: A new table needed to be added to the account for user login information and each person in the database was given a username field to tie it to. An admin account was created by giving it a special login and having certain fields be hidden from users if the admin login was used.

Analysis: Adding this new information to the person required a lot of revision to stored procedures and tables. It was a tedious but not difficult fix. For the admin issue, this was not the most elegant solution to solve the problem by forcing the admin account to have a unique ID and to track each this ID throughout the program to hide features, but it was a very simple and easy fix. However, overall, it was effective for the needs of the project and is sufficient. Given more time, a better solution probably could have potentially been figured out.

Challenge: Getting a datatype to track swimmer times in the standard format was very difficult.

Solution: Initially, the time unit from SQL was used. However, with the current version of SQL used, milliseconds could not be tracked in this time unit. Millisecond tracking is essential to swimming times, so this was unacceptable. After several different iterations and failed experiments, the best way to solve the issue of storing the times and presenting them in a ‘1:57.99’ format was to convert the time given to the GUI into one a decimal data type of seconds with milliseconds as the decimal value. This value could then be converted back into a readable format when searched upon in the database.

Analysis: This once again was not the cleanest solution, but after lots of online research it seemed like the only solution that would work. The conversion was coded manually in Java, and it allowed the user to add in times in this normal format. However, times that do not follow this format end up breaking the program because of this. Using a different version of SQL where time stores milliseconds would have been nice, but overall, we did the best we could.

D. Database Design

Security Measures

The database is only accessed by the java application using a user account with restricted permissions in order to reduce any risk to the SQL server. The java application alters the database via callable and prepared statements only. This removes the risk of SQL injection attacks as only the desired inputs can be passed into the database as parameters to stored procedures, functions, etc. All stored procedures are also designed to check the input parameters for basic false inputs, such as trying to add a duplicate tuple or inputting null values for an input that cannot be null. In order to protect the data that can be accessed and changed by users, different permissions are given to different users. Users' private information is hidden from other users and information about users can only be added by that user or their coach. This helps to prevent any user from editing information that does not pertain to them or creating false information in the database.

Integrity Constraints

Many of the integrity constraints that exist in the system are referential or entity integrity constraints defined on the tables. These are demonstrated on the Relational Schema in part B of the Appendix and are described in more detail in the security and integrity analysis document [4]. Many "Not Null" constraints are also placed on many attributes of the tables in the schema as well. Other constraints added on the system include:

- A person's sex must be either 'M' or 'F'
- A person's date of birth (DOB) must be set before the current date
- Event Unit must be either 'LCM', 'SCY', 'SCM'
- Event Stroke must be either 'Butterfly', 'Backstroke', 'Breaststroke', 'Freestyle', 'IM'
- Event Distance must be > 0 and <= 1650
- A time standard level must be either 'NCAA D3 A', 'Futures', 'NCAA D3 B'
- The male time and female time for a time standard must be < 2000.00 seconds
- The time in a competes in entry must also be < 2000.0 seconds
- A team name must be unique
- A person's hours per week for being part of one team must be between 0 and 168 inclusive.

Stored Procedures

The following stored procedures are used by the database:

- AddToDatabase:
    - Adds a full set of information associated with a person to the appropriate databases. Takes the following parameters: FName, LName, Sex, DOB, Height, Weight, Experience, Style, TeamName, Group, HoursPerWeek, Username, PasswordSalt, PasswordHash, AdminPass
    - Returns a status code if the operation fails, or 0 if it succeeds.
- AddUser:
    - Adds a full set of information associated with a person specifically to the AddUser table. Takes the following parameters: FName, LName, Sex, DOB, Height, Weight, Experience, Style, TeamName, Group, HoursPerWeek, Username, PasswordSalt, PasswordHash
    - Returns a status code if the operation fails, or 0 if it succeeds.
- delete_coach:
    - Deletes someone's coach status from the database. Takes the person's ID.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Delete_CompetesIn:
    - Deletes a person's participation in an event. Takes the person's ID and the event's distance, unit, and stroke.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Delete_equipment:
    - Deletes a piece of equipment from the database. Takes the equipment's model.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Delete_partof:
    - Deletes a person's association with a team. Takes the person's ID and the team's name.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Delete_person:
    - Deletes a person from the database. Takes the person's ID.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Delete_swimmer:
    - Makes a person no longer a swimmer. Takes the person's ID.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Delete_team:
    - Deletes a team from the database. Takes the team's name.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Delete_TimeStandard:
    - Deletes a time standard from the database. Takes the distance, unit, stroke, and level.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- GetPersonID:
    - Finds the PID of a person with the given name. Takes the person's first and last name and uses an output parameter to report the resulting PID.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- GetPID:

    - Finds the PID of an account with the given username. Takes the username and uses an output parameter to report the resulting PID.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Insert_coach:
    - Makes a person in the database a coach. Takes the person's ID and their coaching experience and style.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Insert_CompetesIn:
    - Registers a person as competing in an event. Takes the person's ID, the distance, stroke and unit of the event, the model of equipment used, and the achieved time.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Insert_equipment:
    - Adds a piece of equipment to the database. Takes the equipment's model, brand, and type.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Insert_event:
    - Adds an event to the database. Takes the event's distance, stroke, and unit.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Insert_partof:
    - Adds a person to a particular team. Takes the person's ID, the name of the team, the group they compete in, and the hours per week they practice.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Insert_person:
    - Adds a basic person to the database. Takes their first and last name, sex, and date of birth.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Insert_swimmer:
    - Makes a person a swimmer. Takes the person's ID, height, and weight.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Insert_team:
    - Adds a team to the database. Takes the team's name and region.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Insert_TimeStandard:
    - Adds a time standard to the database. Takes the distance, stroke, and unit the standard applies to, as well as the level and expected time for male and female swimmers.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- IsCoach:
    - Determines whether a given person is a coach. Takes the person's ID.
    - Returns 1 for coaches, 2 for non-coaches, and 3 if no ID is supplied.
- Register:
    - Adds a user account to the database. Takes a username, password hash, salt, and ID to associate this account with.

    - Returns a status code if the operation fails, or 0 if it succeeds.
- Update_coach:
    - Updates a coach's information. Takes the coach's ID, new experience, and new style.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Update_CompetesIn:
    - Updates a swimmer's participation in an event. Takes the ID of the swimmer as well as the distance, stroke, and unit of the event in question, along with the new time achieved and model of swimsuit used.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Update_equipment:
    - Updates the information of a given model of equipment. Takes the model to be updated, as well as the new brand and type to be assigned.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Update_partof:
    - Updates a person's participation information on a team. Takes the person's ID and their team ID, as well as the new group and hours per week to be assigned.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Update_person:
    - Updates a person's information. Takes the person's ID, as well as the new name, sex, and date of birth to be applied.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Update_swimmer:
    - Updates a person's swimmer information. Takes the swimmer's ID, as well as the new height and weight to be assigned.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Update_team:
    - Updates the region a team is located in. Takes the name of the team to be modified and the new region to be assigned.
    - Returns a status code if the operation fails, or 0 if it succeeds.
- Update_TimeStandard:
    - Updates the details of a time standard. Takes the distance, stroke, unit, and level of the standard to be modified, as well as the new male and female times to be used.
    - Returns a status code if the operation fails, or 0 if it succeeds.

Views

The swimming information database contains one view, **viewTimeStandards.** This view is simply meant to display all the information about the time standards, specifically the event information it correlates to, the level of the time standard, and the male and female time for the time standard. The time standard table does not hold the event information, but usually, they are

presented together so this view helps to perform that functionality by grouping the event information with its corresponding time standards.

Indexes

Name_Search is an index placed on the combination of first name and last name of the person table. This index will be useful in many of the searches that are performed over our database. The database administrator and coaches search for swimmers' times and information using the first and last name of that swimmer. The getID method takes the first name and last name input for a database search and returns the ID for that person. Since this method is called many times over application execution, this index should make that method run more efficiently.

Triggers

There is one trigger in place on the database named noDuplicatePeople on the person table. This trigger takes place after an insert in the person table and checks to see whether or not that person exists. If the person does exist, it will roll back the insert transaction and the duplicate person will not be added.

E. Design Analysis

Strengths

- There are security measures throughout the whole application and in the SQL backend. Passwords are encrypted and certain users have less access to the data than others. Accessing the data is always done through prepared and callable statements and never uses query concatenation.
- The referential, entity, and other domain constraints are active on the database.
- Users never see or use the IDs for any of the data they will interact with. Queries, however, use the primary key IDs to perform fast searches and operations.
- The application is fairly intuitive for swimmers to use, and the displayed information is easy to understand.

Weaknesses

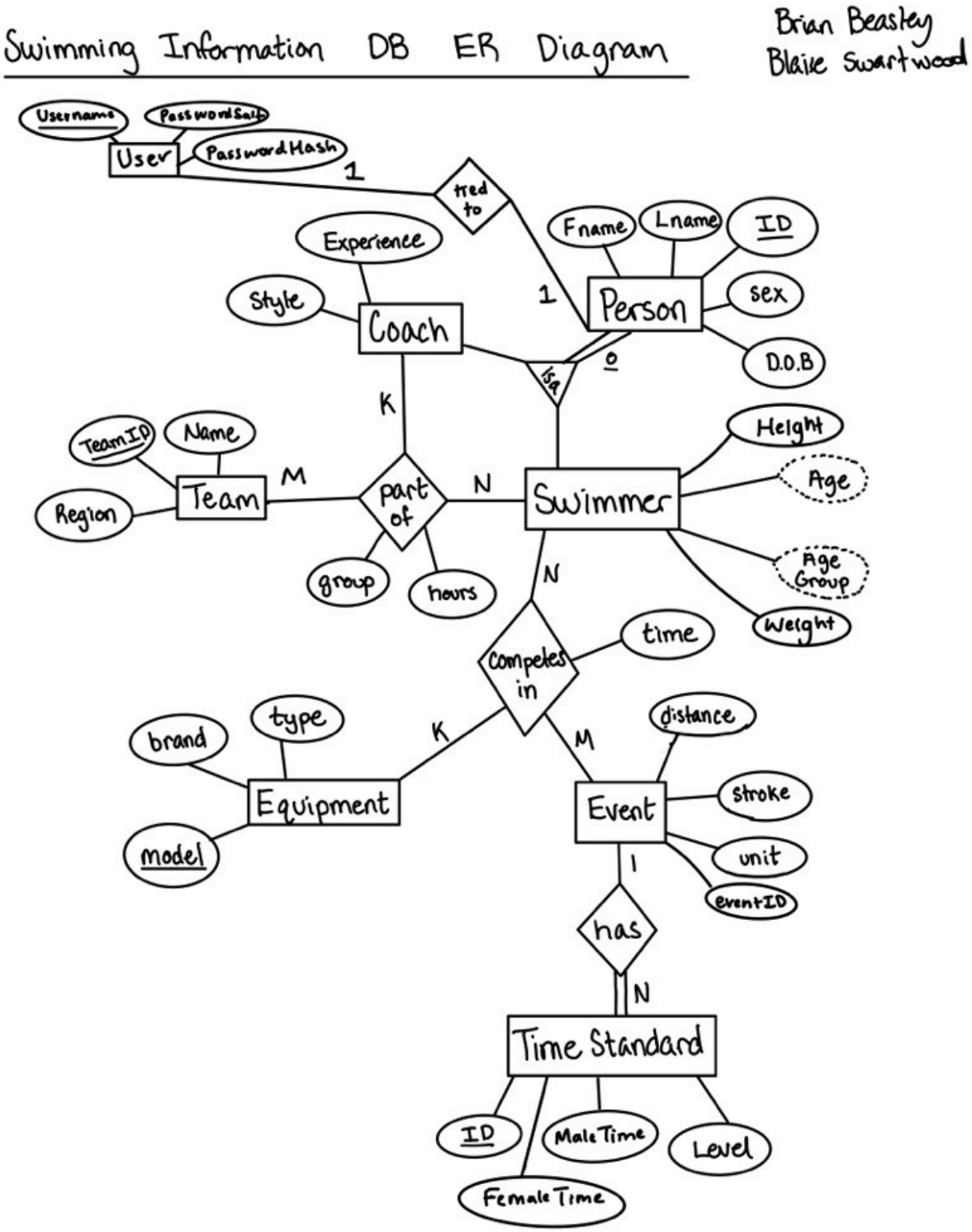
- Users are restricted in their access in order to maintain data integrity. Users can only accesss their own information and change their own best times – they cannot read or change any other swimmer's times. This reduces the flexibility the user has over their control of the

program. While this measure may be necessary, some users may view this as a weakness and wish to have more control.

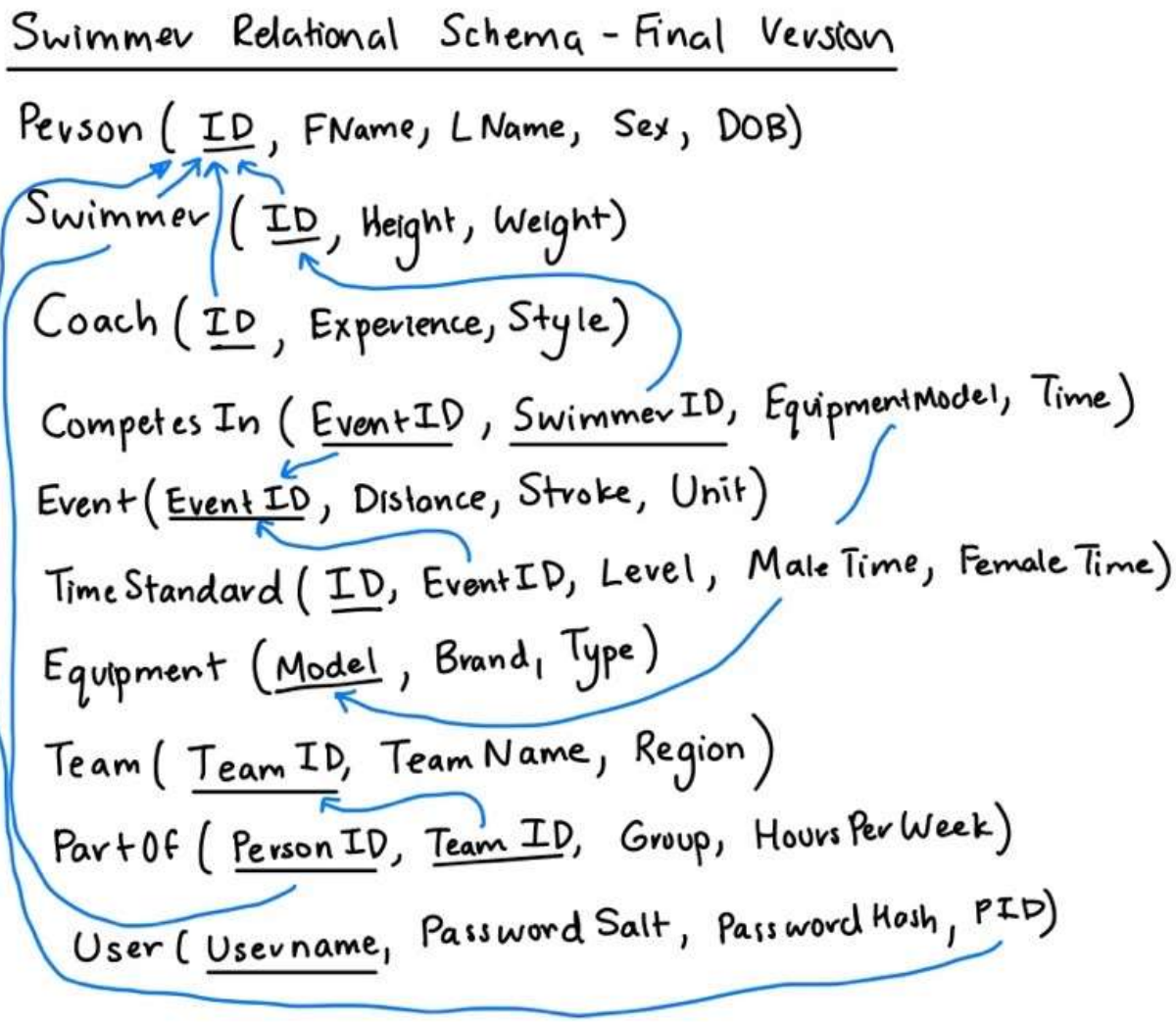
- There is no way to get a search result for all swimmers or view multiple swimmers' times at once. This is not a feature described in the problem statement, but one that may be desired by future users and would have been implemented given more time.
- Users have the option to be both a swimmer and a coach. If a user is both a swimmer and a coach, the coach's view will take priority. That user will see their coach information and have all the access a coach will have. Users may want the ability to log in as either a swimmer or a coach depending on the data they wish to view, but this feature has not been implemented yet.
- If a user wants to update a team, all the teams show up to choose from, not just the team they are a part of. This inconsistency is because we were originally getting teams from a textbook and not a drop-down menu. In the future, given more time, the user would only be able to select times and teams to update that they already have.

Appendix

A: ER Diagram



B: Relational Schema



C: Explanation of ER Diagram

The database revolves around its users which are tied to the people who will be using the database. These people are split into two distinct types: swimmers and coaches. Both swimmers and coaches can be part of a team and can track their own individual stats, such as coaching style/experience for coaches and height/weight for swimmers. Swimmers are also part of an age group, which can be determined using their derived age. Each of these tables and relationships described above are depicted in the top part of the diagram. Swimmers then have the ability to keep track of their best times swum for specific events. In order to do this, the basic event information (the distance swam, the stroke swam, and the unit measurements of the pool) must be provided. When documenting their best time, swimmers also have the option to include the tech suit they wore for that swim. Therefore, the information about the suits such as make, model, and equipment type (suit) are stored in the equipment table. Finally, each event can have multiple time standards associated with it. A time standard is very similar to a qualifying time for swimmers and swimmers often compare their best times to these time standards. Therefore, for each event we must include the time standards associated with that event at each level for both male and female swimmers, documented in the time standard table at the bottom of the diagram.

Index

Glossary

Entity-Relationship (ER) Diagram – representation of a database through depictions

Technical Racing (tech) Suit – a swimsuit meant for competitive racing, which usually cost anywhere from $200-500 and can only be worn a few times before needing to be replaced

Event Units – A pool can either be set up in long course meters (LCM), short course meters (SCM), or short course yards (SCY). Each makes for a different event by differing the pool length and style. For example, one lap in a LCM pool is 50 meters long, one lap in a SCM pool is 25 meters long, and one lap in a SCY pool 25 yards long.

References

[1] USA Swimming Time Search: Individual Times Search (usaswimming.org)

[2] SwimCloud Time Search Information: Swimcloud

[3] Information on tech suits and why it is hard to distinguish between which tech suit is better than the other without data: 21 Best Tech Suits for Swimming- The 2023 Expert Review (swimcompetitive.com)

[4] S3G3 Security and Integrity Analysis document