Spring 4-2001

Swim Search: An Online Sports Management Information Retrieval System

Kevin Erich Heinrich
University of Tennessee-Knoxville

Appendix D - **UNIVERSITY HONORS PROGRAM SENIOR PROJECT - APPROVAL**

Name: Kevin Heinrich

College: Arts & Sciences Department: Computer Science

Faculty Mentor: Michael W. Berry

PROJECT TITLE: SwimSearch: An Online Sports Management Information Retrieval System

I have reviewed this completed senior honors thesis with this student and certify that it is a project commensurate with honors level undergraduate research in this field.

Signed: [illegible], Faculty Mentor

Date: 4/30/01

Comments (Optional):

SwimSearch:
An Online Sports Management Information Retrieval System

by Kevin Heinrich

Senior Honors Project

May 2, 2001

Abstract

Information is vital to success and excitement in sports. Often success in sports is heavily dependent upon the information a team has about its opponents. Analyzing an opponent's abilities and preparing to best an opponent contributes greatly to the excitement of watching many sporting events by adding to the anticipation of that event. Especially when an opponent's abilities are comparable to those of another, the better-informed team often has a considerable advantage over the other. Thus, competitors and spectators alike seek information to further enhance the excitement of competition.

The objective of this project is to further the information available to those interested in swimming. In particular, this project will focus on swimming in the Greater Knoxville Area Interclub Swimming Association (GKAISA). Information is most easily accessible through the Internet; thus, this project utilizes the Internet to disseminate GKAISA swimming information. Through the use of scripts written in Perl and PHP, CGI (Common Gateway Interface), and both a MySQL and a PostgreSQL (Structured Query Language) database, this project will present information in a need-based manner—only specific, needed information will be returned to the user to avoid *pushing* excessive material to the user and detracting from the user's original information retrieval goal. This project is more than just an exercise in data mining/retrieval in that some query options require further processing of retrieved data before they are returned to the user—functions such as predicting future results and customizing views by team will be querying options that are useful to meet many of the needs of the swimming world.

Needs

Approximately two thousand swimmers belonging to roughly thirty teams divided into five or six leagues compete every summer as a part of GKAISA. GKAISA is a competitive league, and the top teams are constantly looking for an advantage over other teams whether by scouting, keeping previous years' lineups, etc. Coaches are always trying to gain a competitive edge over each other, since winning meets is often a big morale boost to all of their team's swimmers. Coaches that can guess opposing teams' lineups have an invaluable edge over teams that enter a meet blindly. Many parents, too, often want to know how well their child will do against each team and how likely it is that their team will beat the other in a dual meet competition. Swimmers also would like to know whom they have to swim against at each meet so they will know what to expect and how much they should train, rest, etc. Thus, nearly every person who has an interest in a swimmer has a distinct information need about the opposing teams.

Test Data

The two major meets of GKAISA swimming are the Smokey Mountain Invitational (SMI) and City Meet. The City Meet competition is split into preliminaries and finals, each of which I considered a separate meet. The initial data given to me included eleven years' worth of results. I obtained the 1999 SMI and 2000 City Meet Preliminaries and Finals from the Web. The 2000 SMI was given to me directly through

Hy-Tek Meet Manager, the swimming software used to run the swim meets. All other results were exported from Hy-Tek into a text file. The results given to me were both the SMI and City Meet Preliminaries and Finals for years 1999 and 2000; I was only able to obtain results from the City Meet Finals for years 1990 to 1998. Hy-Tek Meet Manager was DOS-based prior to 1999, so the results prior to 1999 are in a different format from the files from 1999 and 2000. Since some of the results were copied from the Web, those results are also in another different format. After parsing and eliminating duplicate swimmers, I found that there were approximately four thousand swimmers with slightly greater than 24,000 swims representing thirty-five different teams.

Project Goals

The goal I have for this project is to meet the needs of the swimming community and to further my knowledge of database management systems and scripting languages. To satisfy the competitive nature of the coaches' needs, I created a view of results that is grouped by event so that each swimmer is matched up against every other within constraints. To satisfy the personal nature of the swimmers' and parents' needs, I created a view of results that is grouped by swimmer which shows all times on record for each swimmer.

Initial Attempt

Initially, I only had possession of the 1999 and 2000 results that I stored in a MySQL database. I used Perl scripts to normalize the results into a standard, semicolon-delimited format. Once formatted, I used the Perl Database Interface module (DBI) to read the results into the database [1]. At that point, my database consisted of a swimmers table and a table for each of the meets. The swimmer table only contained an integer identifier as a primary key as well as each swimmer's first and last name. Each meet table contained the appropriate swimmer's identifier as well as the team, age, time, and event. I used a hash based on the swimmer's name, sex, and team to ensure that an identifier represented a unique swimmer. Due to the nature of the results files, most all the fields were text fields. Searching through my first database was done either by first or last name only. Many problems existed with my database structure as well as the integrity of my data.

As my knowledge of database design increased, I redefined my tables into the third normal form. I changed the swimmers table to contain an identifier, first and last name, team, gender, and birth year. The meet tables were reduced to contain only a swimmer's identifier, the stroke, distance, and time. Since the results files only contain each swimmer's age, I calculated each swimmer's birth year as the difference between the meet year and the age of each swimmer listed. Since the age for each swimmer for the swim season is based on his or her age as of May 31st of that year, the birth year should be consistent for each swimmer across meets. Searching could be done over any of the fields in either the meets tables or the swimmers table. All Web interfaces were done with the Perl DBI and CGI.

Revisions

Once a PostgreSQL database was available through the Computer Science Department [2], two classmates assisted me with working on this project. Justin Giles primarily converted all the Perl scripts I wrote into PHP scripts [3], while Patrick Lynn developed a better interface (see figures 1 & 2). Meanwhile, I was able to obtain the results from 1990 to 1998.

With the increased amount of data, swimmers that swam for different teams over the years became more of a concern. As a result, I made several more tables. I created the teams table which contained a team's unique code as well as their name. Along with the teams table came the team_lookup table that matches a team's code with the spelling of the team's name within a given year. Thus, different spellings for the same team could be mapped to the same unique code. For example, the Maryville-Alcoa Flying Dolphins had several different spellings throughout the years to accommodate length constraints on the team name. These spellings include MAFD, MaryvilleAlcoa Flying Dolphins, Maryville-Alcoa Flying Dolphin, etc.

To accommodate the swimmers who swam for different teams over different years, I migrated the team field from the swimmers table to a new table named swimmer_team. In this table, I matched each swimmer's identifier with a team code and a year. Since swimmers are not allowed to switch teams within a year, this method should be sufficient. To accommodate the growing number of meets, I created a meets table that contained the table names of the meets as well as the name of the meets themselves and the year in which they were swum.

One alternative to having a separate table for each meet is to have one large table named results. The drawbacks against this, however, are many. If I were to combine all results into one large table, every query of the database would be over all swims for all years. On the other hand, my method, while making the SQL query more complex [4], has the potential to save many comparisons by only querying over selected meets.

After the initial pass through the results with my Perl scripts to load all the swimmers data into the database, I found two main problems. First, many swimmers had their names spelled differently over the years, causing them to be indexed as two separate people. Second, birth years for the same swimmer did not match up. For example, some swimmers were listed as 17 years of age during both 1999 and 2000, causing them to have corresponding birth years of 1982 and 1983. Also, from one year to the next, a swimmer's listed name may change from Matt to Matthew. To remedy this, several automatic procedures were considered, but in the end manual verification could not be avoided.

Prior to manually combining many swimmers into one record, there were over 4,800 records. After manual passes through the database, that number was reduced to 4,000. In other words, with respect to the swimmers, approximately seventeen percent of the data given to me was bad data. Upon inspection of the types of misspellings and birth year mismatches that caused a swimmer to be multiply indexed, I have concluded that any type of automatic correction method would have produced as many errors as it would have fixed. Once the records were properly merged, I made one final pass through the database to fix any obvious name misspellings.

Storing the database in its new, properly normalized form also caused queries to become much more complex. To extract data from the database, the meets table must first be queried over to determine which meet tables to query. After that information has been determined, a union is done over all the appropriate meet tables. Obviously, queries over all meets will take a much longer time to process. However, our interface discourages use of broad queries over all years by placing the "all years" option last in the drop-down menu.

One of the most useful functions of our portal, SwimSearch, is the automatic age-up of each swimmer. This is useful for coaches trying to determine what lineup to expect from each team and is easy to implement with a slight modification to the query sent to the database.

Another function I found both interesting and somewhat useful is the "predict future results" function. This function predicts each swimmer's results for next year for each event that he or she swam in this year. The predictions are based on the average improvement for every swimmer who swam that event in the previous year. Thus, the prediction is equivalent to saying "if your swimmer improves as much as every other swimmer did last year on average, then he or she will swim this." For a more in depth explanation on this function, visit the help section of our portal. A link to our portal can be found off of http://www.thextus.com.

Future Extensions

This portal, in its current state, meets most of the needs of the swimming community. However, there are always more extensions possible. One obvious extension other than obtaining more results is to make the prediction function better. That is, predict future results based on previous performances of an individual swimmer rather than the aggregate performance the age that the swimmer is.

Another modification I plan to make soon is to change the birth year field into a birth date using PostgreSQL's Date data type. Adding a Web-accessible administration page would make this extension much more feasible by allowing coaches to enter in their swimmers' birth dates online. Also, since this project has sparked much interest, I may be able to obtain swimmers' birth dates from GKAISA records. If I can obtain birth dates, then birth year discrepancies mentioned before would no longer exist.

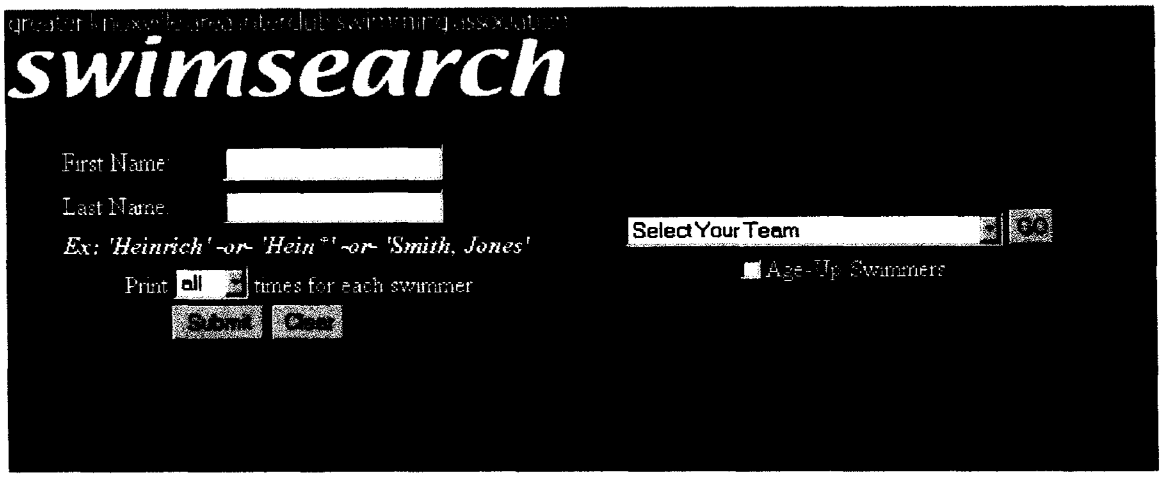
Unfortunately, I did not learn too much about SQL until much of the project was already implemented and the data already entered into the database. In the future, I plan to change the time field in each of the meet tables to a Time data type, while I allow DQ and NS swims to be indexed under some pre-determined, unobtainable time. To find DQ and NS swims, I could then use a CASE statement within the SQL.

Some people are interested in relay finishes; however, the usefulness of relay finishes and times is miniscule compared to the usefulness of individual swims. One possible extension would be to add a best relay finder. That is, within a given year (and other constraints), find the best relay based on the results in the database. Such a function, if implemented correctly, would enable coaches and swimmers alike to be able to completely size up their opponent before swimming them.

Acknowledgements

I would like to thank Dale Montgomery for providing me with the results from 1990 to 1998, Keith Lambert for his constant feedback of my portal, Dr. Michael Berry for his encouragement and advice, and Dr. Peiling Wang for her advice with SQL statements. I would also like to express my appreciation to Justin Giles for writing most of the PHP scripts for the portal and to Patrick Lynn for designing the layout and JavaScripts used by the portal.

Figures


greater knoxville area interclub swimming association

swimsearch

First Name:

Last Name:

Ex: 'Heinrich' -or- 'Hein' -or- 'Smith, Jones'*

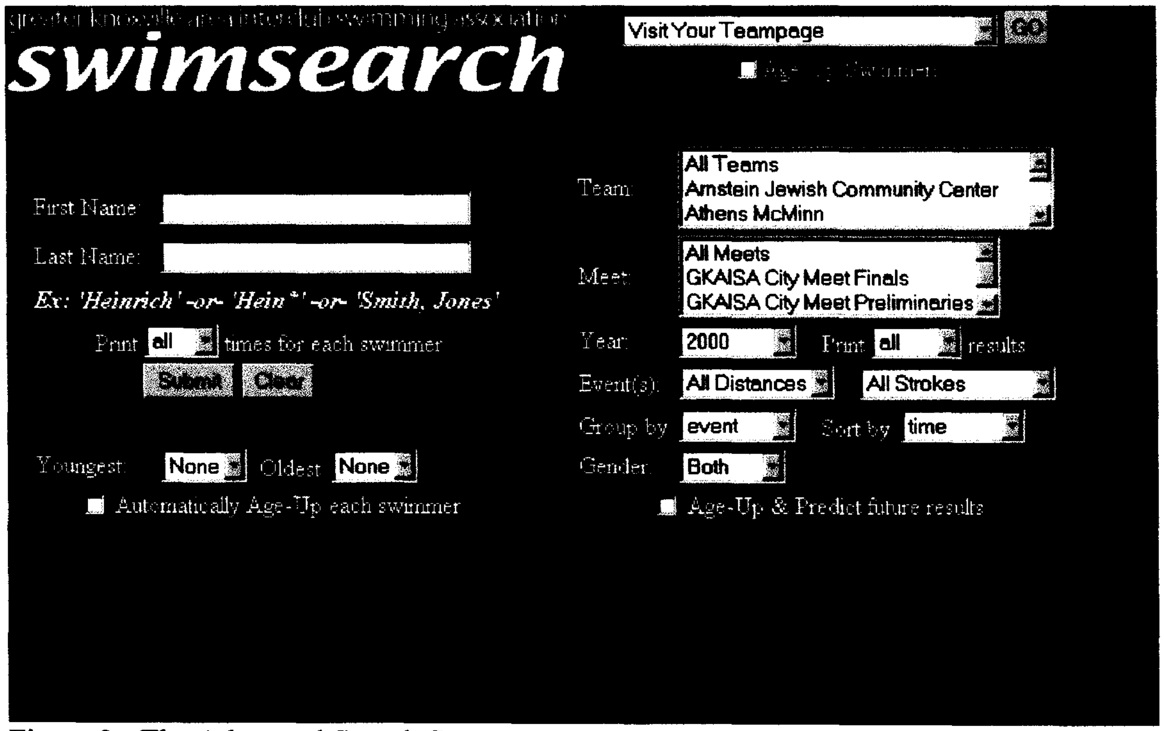
Print all times for each swimmer

Submit Clear

Select Your Team GO

Age-Up Swimmers

Figure 1: The Basic Search Screen.


greater knoxville area interclub swimming association

swimsearch

Visit Your Teampage GO

Age-Up Swimmers

First Name:

Last Name:

Ex: 'Heinrich' -or- 'Hein' -or- 'Smith, Jones'*

Print all times for each swimmer

Submit Clear

Youngest: None Oldest: None

Automatically Age-Up each swimmer

Team: All Teams / Arnstein Jewish Community Center / Athens McMinn

Meet: All Meets / GKAISA City Meet Finals / GKAISA City Meet Preliminaries

Year: 2000 Print all results

Event(s): All Distances All Strokes

Group by event Sort by time

Gender: Both

Age-Up & Predict future results

Figure 2: The Advanced Search Screen.

References

[1] Alligator Descartes and Tim Bunce, *Programming the Perl DBI*, Cambridge: O'Reilly & Associates, 2000.

[2] The PostgreSQL Global Development Group, *PostgreSQL 7.1 Reference Manual*, [online] 2001, http://postgresql.readysetnet.com/users-lounge/docs/7.1/reference/ (Accessed: 3 April 2001).

[3] Anne-Scott Whitmire, *Creating Database-Enabled Web Pages Using PostgreSQL and PHP*, [online] 2001, http://yoho.cs.utk.edu/~cs494/ (Accessed: 2 April 2001).

[4] James R. Groff and Paul N. Weinberg, *SQL: The Complete Reference*, Berkeley: Osborne/McGraw-Hill, 1999.