North Dakota State University

Crickets Database

Databases Final Project

URL: <http://students.cs.ndsu.nodak.edu/~aabeyer/Crickets/>

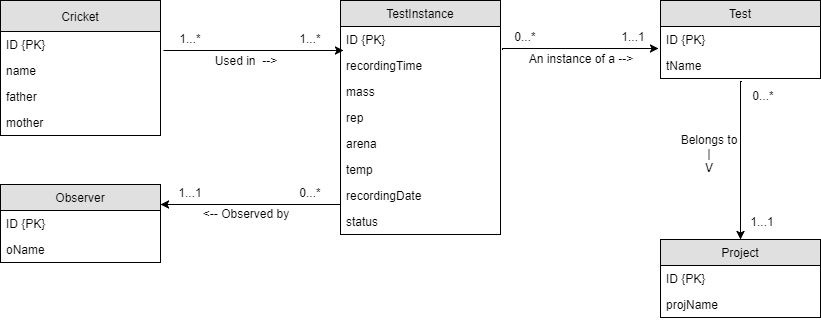
Aaron Beyer

Tyler Tjosaas

Goals

* Create web-based database system to organize data from various cricket experiments.
* Combine data from different experiments.
* Allow uploading/download of csv files.

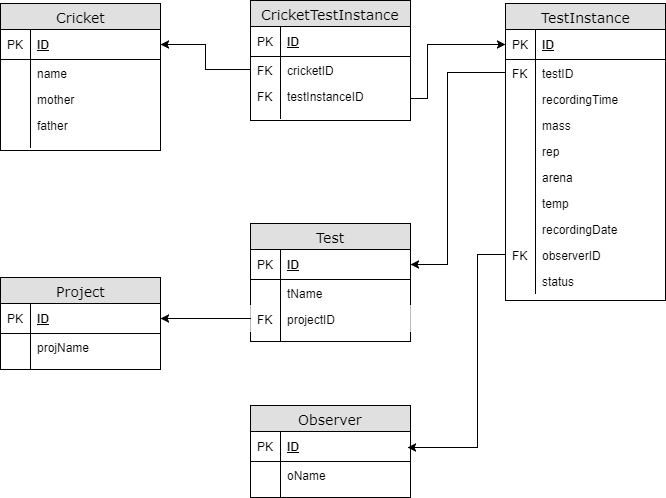
Entity-Relationship Model



Functional Specifications

* Used by
  + This web-based database will be used by the team led by Professor Ned Dochtermann of the NDSU Biology Department carrying out cricket experiments. In practice, the database will be used by anyone managing the experiment data on the project.
* Tasks
  + Uploading Data
    - The data in the database will all come from csv files holding data from various experiments. Uploading data will be one of the main tasks to be carried out; this will be done by using the “Upload Data” page. The user chooses a csv file and specifies the project and test it belongs to.
  + Downloading Data
    - Once data has been uploaded to the database, the data from all the tests can be downloaded into a single csv file from the “Download Data” page. The user selects criteria to filter the data or just downloads all the existing data.

Schema (Relational Model)



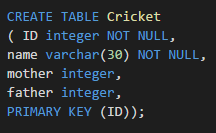
Walk-through (of each page)

* Upload Data
  + Upload data by choosing a csv file, project, and test, followed by clicking the submit button. The csv file will be selected from local files on the computer and will have to follow a specific format for the first ten columns; the format is shown in an image at the top of the page.
  + There is validation on the page that checks the csv file and makes sure it follows the correct format. It also will not allow any other type of file to be uploaded. The project and test fields also have validation, requiring them to have a value selected.
* Download Data
  + Download data by selecting criteria for a filter, clicking the submit button, and then clicking the “Export As CSV” button. After clicking submit, a preview of the data will be shown at the bottom of the page; the exported csv will contain the same data shown in the preview.
  + There is validation on the page for the criteria section. For every option besides “All Data”, the value field must not be blank and only allows letters and white space. SQL injection is prevented from the validation.
* View Data
  + View data in any of the database tables by selecting a table and clicking the submit button. There is also an “Export As CSV” button available if the user desires to have the table data in a csv file.
  + There is validation on the page that requires a table to be selected.
* Delete Data
  + Delete data from the database by selecting a delete option and clicking the submit button. The two options are “Delete Test Data Only” and “Delete All Data”.
    - “Delete Test Data Only” removes all the rows from the TestInstance table. Since the download page joins on the TestInstance table, there will be no test data to download from the database. This option is helpful when clearing all test data is desired, but the user wants the crickets, observers, tests, and projects to remain in the database.
    - “Delete All Data” removes all the rows from every table. This option is used when the user wants to completely clear everything from the database.
  + There is validation on the page that requires a delete option to be selected.

Sample Queries

Queries were written for creating tables, uploading data, downloading data, viewing table data, and deleting data. The data is stored in an Oracle database.

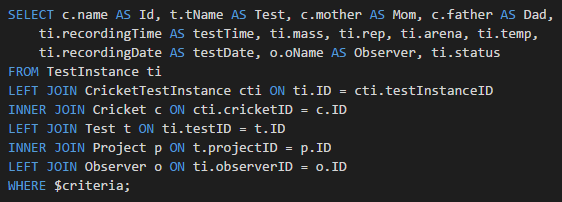
* Creating tables: The create table queries are all very similar, with different names and fields. For example, here is the query for creating the Cricket table.



* Uploading data: When the user uploads a csv file, it goes through php code that inserts rows into each table (if necessary). All insert queries follow the same format, using php variables for the values to be inserted. For example, here is the query for inserting a row into the Project table.



* Downloading data: The query for the final csv download is the same for all filters just with the WHERE at the end determined by the criteria for the filter.



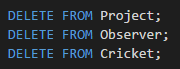
* Viewing table data: The query for viewing table data is just a simple SELECT \* query from the table chosen by the user.



* Deleting data: The queries for deleting data are very simple. They simple delete all rows from specific tables, determined by the delete option the user chose. For example, here is the delete query for “Delete Test Data Only”



Since all foreign keys in the database contain “ON DELETE CASCADE”, the tables with foreign keys do not need to be included in the delete queries. Thus, here are the queries for “Delete All Data”



Satisfied Functional Specifications

The “Upload Data” and “Download Data” pages satisfy all functional specifications for the database. Uploading data is done easily and covers all different tests if they follow the specified format. Also, downloading data is simple with the optional filter and the “Export As CSV” button.

The database follows the 3-tier system. The tables are located on the NDSU cs Oracle database, the php and html is hosted on NDSU’s “students.cs.ndsu.nodak.edu/~aabeyer”, and the client is wherever the user is accessing the URL from.

Besides adding CSS to the pages and working on the visuals, the only addition to the database that could be useful is adding more fields to the final downloaded csv. This would consist of adding fields specific to different tests to each row, resulting in null values for rows belonging to different tests. To implement such a change, professor Dochtermann and his team would need to sit down and discuss the important fields of each individual test and select the fields to be added. This list was requested while making the database but has not yet been provided; thus, the common fields from each test are currently the only fields used in the final csv.

Cost-benefits of Design

There were few compromises made on our design; we were able to create the database professor Dochtermann desired without many problems. The only compromises/choices we made had to do with the Cricket table and the TestInstance table.

Originally, the Cricket table was going to have “mother” and “father” fields as foreign keys pointing to the Cricket table itself, as the parents are crickets as well. However, we found that the only way of identifying an individual cricket was by using the “ID” field in each test csv. Since the parents are not always in tests themselves, there would be a lot of inconsistencies in the table. Thus, deriving “mother” and “father” fields (no longer being foreign keys) from the “ID” field itself seemed like the best option.

For example, an ID taken from a test’s data: AD\_CS2\_134\_132\_23

* + 134 identifies the mother
  + 132 identifies the father

With this method, the mother and father can still be analyzed, but they do not need to be entered into the table as crickets themselves. It is clear that the 134 and 132 could not be entered into the table as new crickets, as the whole ID is what is used to identify a new cricket.

The other decision made was only using the shared fields amongst all the tests in the TestInstance table. The reason behind this decision is explained in the previous “Satisfied Functional Specifications” section.