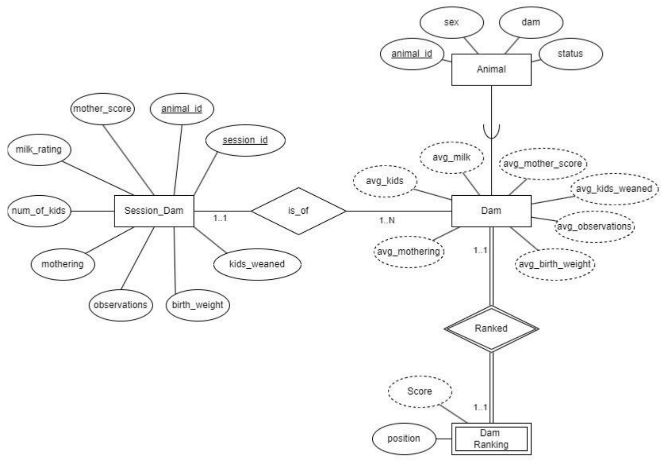
Group 10: Goat Ranking System

Daniel Andrusiewicz, Aidan Stoner, Rooby Dartiny

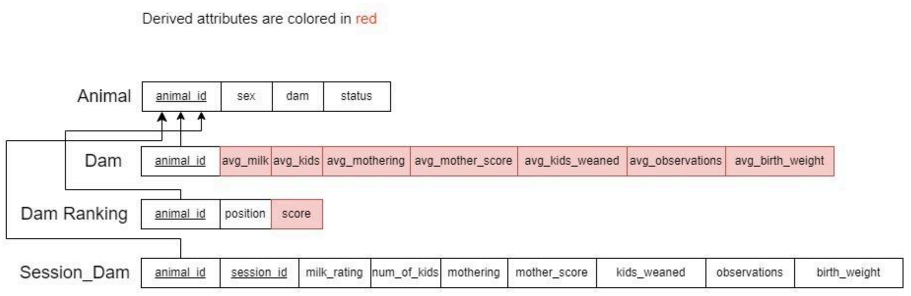
Phase I - Phase IIb:

These phases were simply for pitching the idea of our goat ranking system, outlined the points-based system, along with which criteria the points will be determined by. These phases don't have any technical details, so for the brevity of the report, we will just say that we outlined our plan and decided to proceed with the ranking system.

Phase III: Database Model



In our initial database model, we were going to have the statistics for each dam (such as avg_kids, avg_milk, avg_mothering, etc..) to be derived attributes, computed dynamically. However, in our final implementation we decided against this pattern, and instead opted to use a view-based approach. In our new design, we would have just the session_dam statistics, GROUP BY the animal ID, and simply select the AVG(x) of each variable, instead of having to compute it beforehand. This technically would be slower, since it would have to be computed each time the database was queried, but massively simplified the work that had to be done on our end. In the end, we decided that actually finishing the project with a technically (but not visibly) slower solution that could be further improved later on was better than not finishing the project at all.



Phase IV: Elaboration & Database Design

This phase was focused on database normalization of our previously designed schema. By this point, we had moved to SELECT-based averaging instead of using derived attributes:

The "Dam_Ranking" relation that we initially had was removed, instead we will be doing the ranking through SQL queries (like SELECT returning the score for the given query), and that information will not be saved in the database.

We also initially stated that we were going to not use any views in our final product:

For our project, we will not require any views to be able to retrieve the data needed to rank the dams.

In the end, we ended up using a view to automatically SELECT all dams (females) whose status was “Alive”. This was done to avoid ranking dams that were either sold or already dead, though this information might still have been useful, the initial goal was to rank the dams who were currently on the farm. This view ended up helping us out, since it reduced the complexity of the final SQL statements that ended up getting used in the actual project.

Phase Va - Vb: Implementation and Construction

The actual construction of the webpage was done using the Flask template website provided to us. We also decided to modify the website to have the results be displayed on the same page, instead of navigating to a second page to display results, as that would improve the user experience. Special care was taken to ensure that the checkboxes for the different ranking criteria had their state saved between webpage submissions, and that the results table was clearly visible against the background image.

Conclusion

We think that our project’s end result worked really well. The database is fast to respond, the UI is easy to understand, and the instructions to run the website are listed in the project's README. While we don't think that it's all too practical for our actual stakeholder to regularly use the website, mostly because it would take some programming experience to be able to do, it's a great starting point for another group that might be interested in taking up the project and improving upon it. There is potential for a speed increase by using derived attributes, an even better UI design, or using the farm's master database so that there would be no need to manually update the website's database using the dataextraction python program. The last point we feel is the most important to fix if this project wants to be legitimately used by the stakeholder, as it's unrealistic to require the stakeholder to update the database themselves every time they want to use the website.