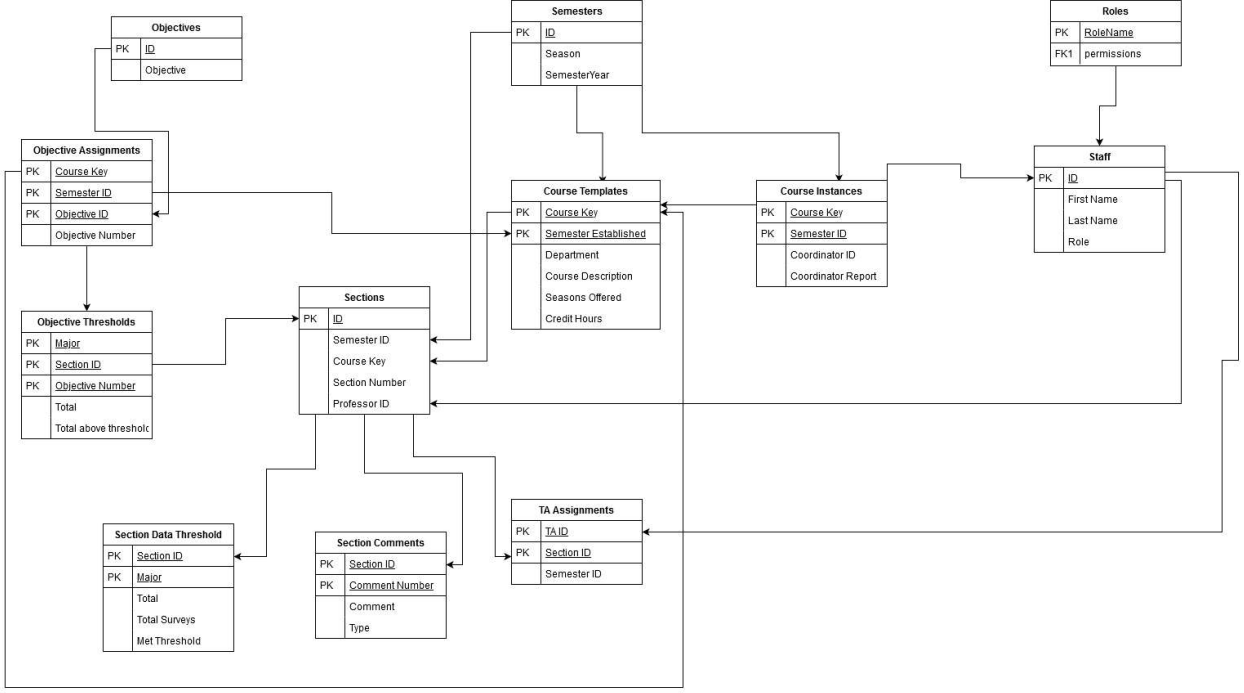


Authors: Rory Spralls and Basil Holloway

Database

Documentation

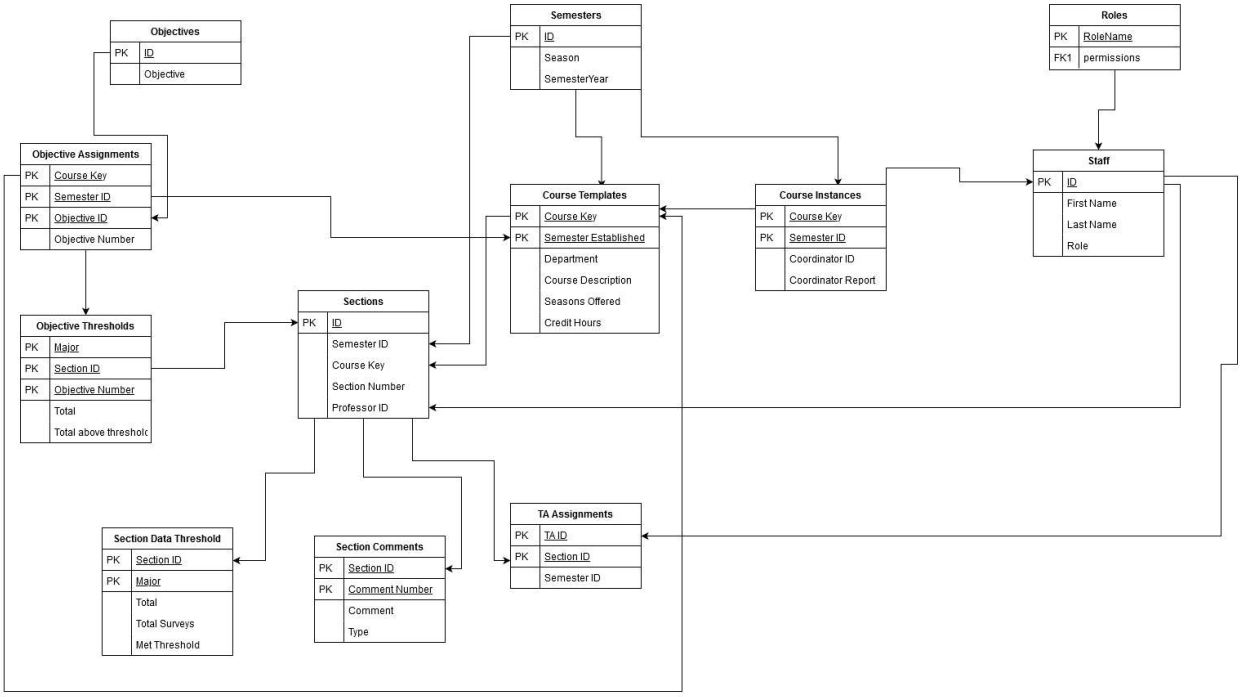
Database:

1. **The tables:**
    a. **Course Templates:** General data on courses like semester established and course description.
    b. **Course Instances:** More specific data like the current coordinator of the course and their comments on the course
    c. **Sections:** Very specific data like the professor, their comments on the course, and the number of students in the course.
    d. **Section Comments:** specific comments from students and TAs about a section of a course.
    e. **Objectives:** List of objectives, their descriptions, and which major they most closely follow
    f. **Objective Thresholds:** Logs how specific majors did in *specific portions* of the course
    g. **Section Thresholds:** Logs how specific majors did in the ***entire*** course
    h. **Semesters:** Mainly used to speed up queries because comparisons made between semesters is not very common
    i. **Staff:** Holds staff ids, and their roles. Includes TAs
    j. **Roles:** Handles roles and their permissions like what pages of the system they may view

k. **Assignment Tables:** These are sort of junctions that allow many-to-many connections in the database

More information on the tables can be gained by reading through the MS Access database design document, located in Backend\AbetData.accdb.

2. Their Relationships:



This figure shows a rudimentary picture of the relationships. Remember that PK, or *primary key,* means that in any given entry on any given table, there will *never* be a duplicate set of primary keys. IE: section comments can not have more than one comment that refers to the section 1 of CSCE 4110 and has the comment number 3.

3. **The API: At the end of this document will be resources that contributed to the construction of this portion of the back end.**

    a. **General Information**: The API acts as a sort of bridge between the front-end UI and the back-end database using Microsoft's ASP NET server framework. It was written and designed in a way that would enable it to be able to change the database that it referred to as easily as possible, hence the strange combination of a traditional ASP NET API and an API built for communicating with MS Access, which requires a different approach.

    b. **AbetController.cs:** This file handles the requests from the front-end and determines the actions required by the API to fulfill requests from the front-end. Has not been built to support MS Access yet, and will require integration with OdbcController.cs. *This portion of the code does not relate very closely to MS Access and can therefore be mostly ignored until edits are needed for it to work with MS Access*

        b.i. getRaw(string table): returns the raw DbSet. This is used internally, mainly to save on a few typestrokes.

        b.ii. getTable(string table): returns a DbSet as a list to be read asynchronously. This is sent to the user at the front-end.

        b.iii. createEntry(string table, object entry): creates and entry in the table and returns success or failure

        b.iv. editEntry(string table, string id, object entry): edits an entry in the table. Returns no content on success.

        b.v. deleteEntry(string table, string id): deletes an entry in the table

    c. **OdbcController.cs:** This file handles communication between the API and MS Access. There is little code within because the file is incomplete, as we ran out of time to fully build it, as we only learned of MS Access's shortcomings after constructing most of the API.

    d. **AbetContext.cs:** This simple file just puts everything that is within ABETmodels.cs into a set of DbSet list objects. These are used to manipulate the data in a way that is closer to SQL. Has little to do with MS Access, unfortunately.

    e. **ABETmodels.cs:** This file closely resembles the structure of the MS Access database in terms of what data is stored within each entry of the database, but it is very loosely related as it is the base of a traditional approach to database APIs.

4. **Learning resources used in the construction of the API**

    a. **Tradition API construction:** Can be used to understand the theory behind the structure of the API. Will help in altering the APIs code in the event of changes being necessary.

    b. **MS Access API construction:** Can be used to understand the theory behind the structure of the MS Access portions of the API.

    c. **UNT Free Microsoft Office Subscription:** Needed to use MS Access in a capacity that will allow you to make any serious changes to database structure.

    d. basilfholloway@gmail.com: the email of the main API and database programmer. Please contact him if you have any questions. Will be most helpful on weekends.