[COSC-257] ER Schema and Relational Schema
Authors: Hanaa Charania, Rachel Lin, Nayeon Shin, and Hewan Worku
Date: October 17, 2023

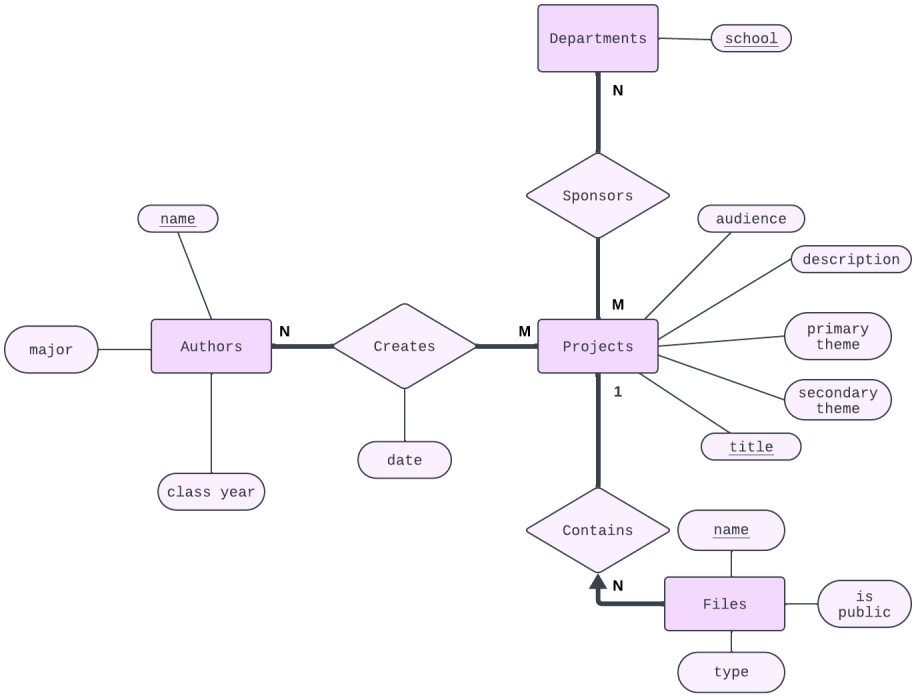
Project Revision:

On October 6th, we requested our client, Sheila Jaswal, to share her data and discuss its organization. Sheila was interested in adding features such as keyword search and template presentations for students with outstanding final projects. However, the data she provided was disorganized, with multiple folders having varying names, and spreadsheets linked to other files, slides, and documents. On top of that, the documents containing links to other folders were only accessible through those specific documents. Some folders were outside any group and locked from our view. We later reviewed the data and sent her a list of questions about the shared folders, but she has not responded yet.

While we await her response, we need to revise our initial project plans. Approximately, only 60% of the data is organized, limiting our current database scope. A substantial amount of data is stored in PDFs with non-clickable links. Our database entities will remain the same, but we won't be able to include as many materials and projects as originally planned because a significant subset of this data lacks representation in spreadsheets and the descriptive names are required for efficient database searches. This data is inaccessible without further "cleaning." Therefore, we can only work with data organized into specific folders within the shared drive. Concentrating on the already organized data will allow us to focus on developing the database and ensure satisfactory user accessibility, proper data formatting, and sufficient front-end features.

Furthermore, we need to host the website on the Amherst server instead of AWS to maintain consistency with the existing STEM website, which is currently hosted on the Amherst server. We also require more raw data from Sheila to determine our metadata needs. With this metadata, we can better identify data trends that can be reflected in the attributes of our ER diagram.

Final ER Diagram:



Relational Schema:

```
CREATE TABLE Authors (
     name CHAR(30),  # author name
     major CHAR(30),
     class_year INTEGER,
     PRIMARY KEY (name)
);

CREATE TABLE Projects (
     title CHAR(100),
     description CHAR(500),
     primary_theme CHAR(50),
     secondary_theme CHAR(50),
     audience CHAR(40)
     PRIMARY KEY (title)
);
```

```
CREATE TABLE Creates (
     name CHAR(30),    # author name
     title CHAR(100),  # project title
     date CHAR(10),
     PRIMARY KEY(name, title),
     FOREIGN KEY (name)
          REFERENCES Authors,
     FOREIGN KEY (title)
          REFERENCES Projects
);

CREATE TABLE Departments (
     school CHAR(100),
     PRIMARY KEY (school)
);

CREATE TABLE Sponsors (
     school CHAR(100),
     title CHAR(100),
     PRIMARY KEY(school, title),
     FOREIGN KEY (school)
          REFERENCES Departments,
     FOREIGN KEY (title)
          REFERENCES Projects
);

CREATE TABLE Files (
     title CHAR(100) NOT NULL,
     name CHAR(100),  # file name
     is_public BOOLEAN,
     type CHAR(30),
     PRIMARY KEY (name),
     FOREIGN KEY (title)
          REFERENCES Projects,
          ON DELETE NO ACTION
);
```

Role Assignments:

- ➔ Data Scraping: all
- ➔ Data Population and Management: Hewan and Hanaa
  - ◆ CRUD
- ➔ User Authentication & Permissions: Rachel
- ➔ Keyword and Query Search: Nayeon
- ➔ User Interfaces (Front-End Development): all

Description of Software Installed:

- ➔ Visual Studio Code IDE: free source code editor with multiple language extensions and features for efficient software development
- ➔ Node.js: JavaScript runtime environment that allows developers to run JavaScript code on the server, rather than solely on web browsers
- ➔ React: JavaScript library for building user interfaces, with a focus on component-based development
- ➔ Django: Python framework designed for handling the server-side logic of a web application and facilitating database interactions
- ➔ PostgreSQL: object-relational database management system with extensibility and support for read-write operations, large datasets, and complex queries

Open Questions:

1. How can we implement the feature of letting the users query into our database system?
2. How do we configure the different levels of user permissions? Specifically, how do we only allow Sheila Jaswal to add, modify, or delete files while ensuring that regular Amherst students and faculty not involved with HSTEM management do not possess these privileges?
3. Where will we store our data? Will we store it in a cloud-based solution, and if so, which cloud service will we opt for?
4. What types of data will our database hold? Are we planning to store the links to the files on Google Drive?
5. For data with identical content and attributes but varying permissions, how will we structure our database to maintain organization and consistency?
6. How do we extract or scrap and integrate data from different folders, files, or links into our database?
7. Lastly, how should we manage and organize data that is confidential and that we do not have permission to access?