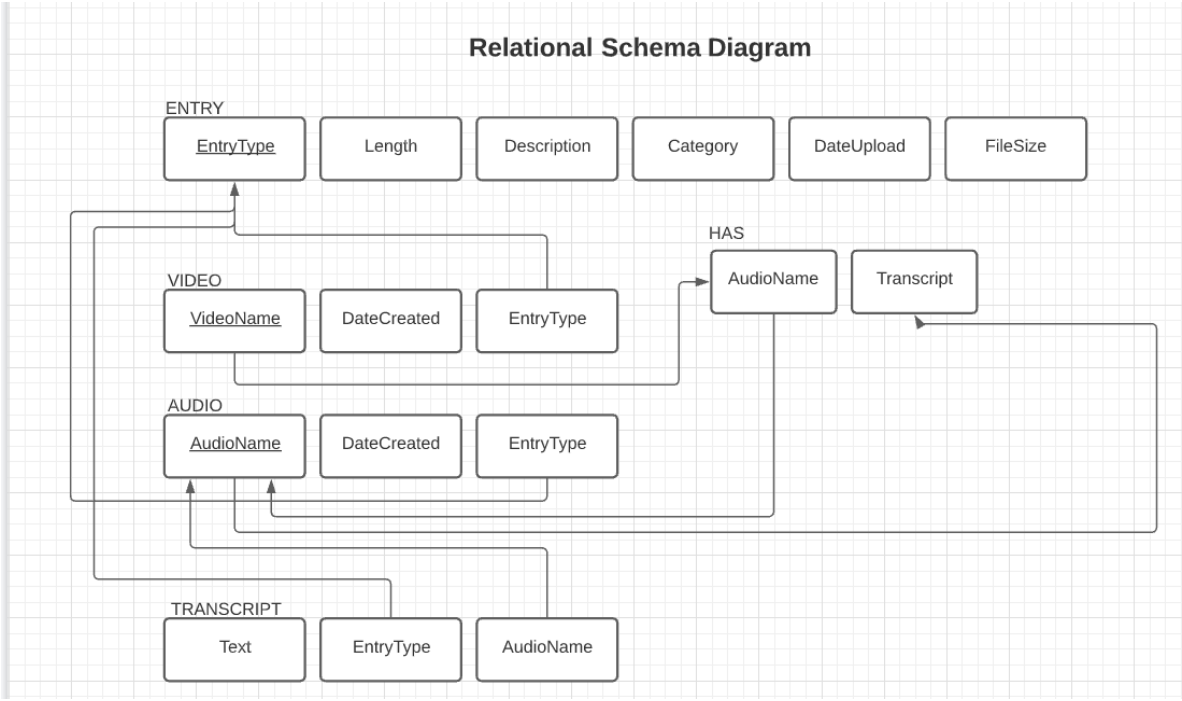
*Stage IV - Elaboration: Design*

**Kiera Gill, Johnny O'Brien, Roman Rychkov, Sydney Blanchard**

---

**Relational Schema:**




**Normalizing relations to BCNF**

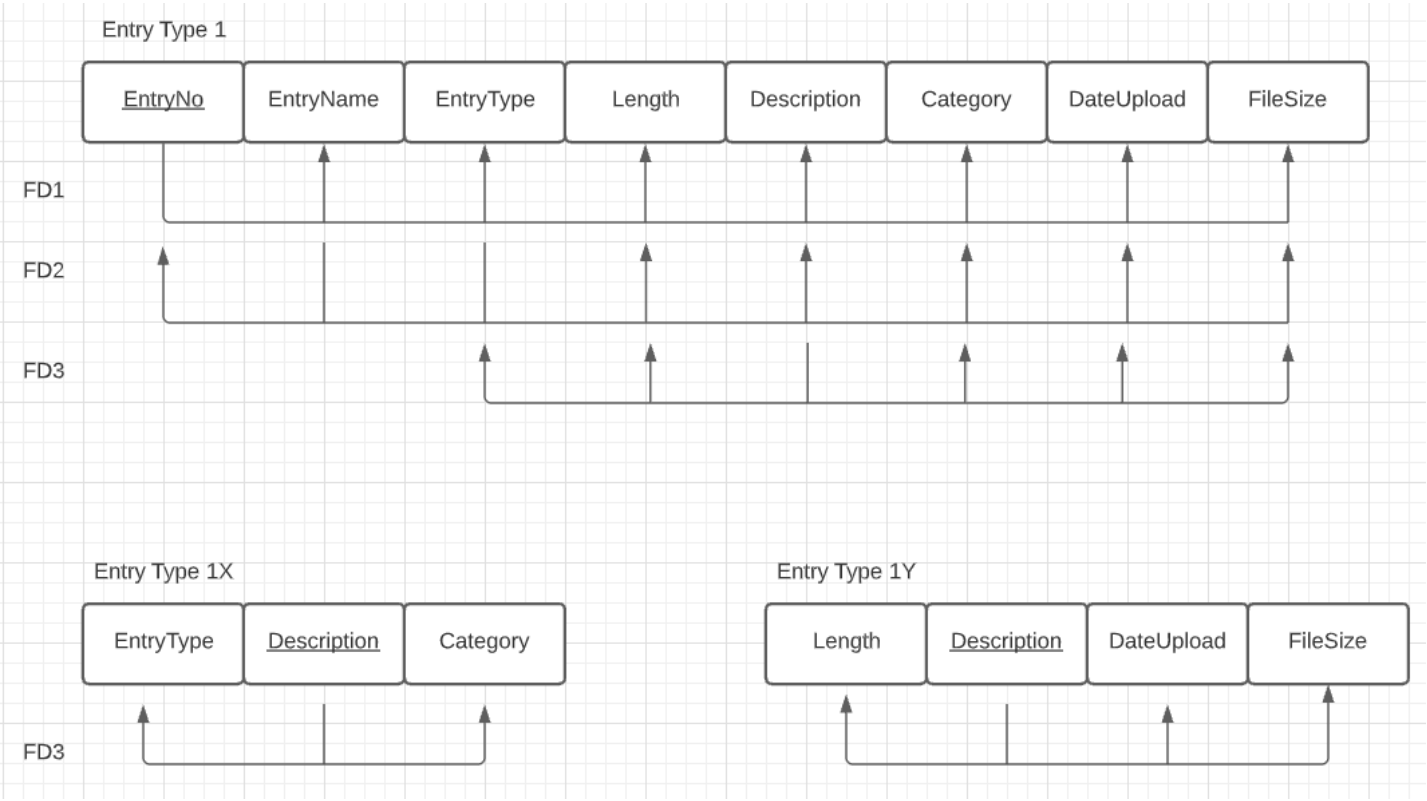
1.

| EntryType | Length | Description | Category | DateUpload | FileSize |
|---|---|---|---|---|---|

   a. This entry is not in BCNF. We have modified the table entry to put it in BCNF, as shown below.

| EntryNo | EntryName | EntryType | Length | Description | Category | DateUpload | FileSize |
|---|---|---|---|---|---|---|---|

Fixed below




We normalized the Entry Type table by breaking down the individual functional dependencies of the attributes. FD1 shows how EntryNo points to the access of all other attributes of the entry. FD2 shows how EntryName and EntryType points to the access of all other attributes of the entry as well. Finally, in FD3, the Description points to the EntryType, Length, Category, DateUpload, and FileSize, which will all be somewhere in the Description. We decided to break down this Functional dependency into two groups in order to normalize with a specific category in mind. The Entry Type 1X list of attributes will be the “letters” descriptions, where the descriptions have words containing each attribute. The Entry Type 1Y has attributes that will be “number” descriptions, where the descriptions have numbers containing each attribute. By decomposing, we were able to achieve 3NF normalization of our relational diagram and make it functionally determine other non-key attributes.

Similarly, we believe that video, audio, and transcript are already in the BCNF form because they cannot be further broken down. All three of them rely on the Entry Type to determine other distinguishing features. This is because each different entry type has the same additional attributes, including EntryName, length, description, category, DateUploaded and FileSize. Essentially, the thing that distinguishes between all of the sources in the database is the entry type.

**Defining different views**

1. *Views and roles*
    a. User
        i. A user can search for entries and access entries in the database
    b. Admin / Super User
        i. An admin can post, search, delete, and modify the entries
2. *Transactions that the views will have*
    a. Admin only transactions:
        i. Upload Entry
        ii. Delete Entry
        iii. Modify Entry
        iv. **Virtual Tables Required**:
            1. entry(EntryNo, EntryName, EntryType, DateUpload, Description), audio(AudioName), video(VideoName), transcript(Text)
    b. General user transactions:
        i. Pause/Play the Audio
        ii. Pause/Play the Video
        iii. Read Transcript
        iv. Search Entry by Name
        v. Search Entry by Date
        vi. Search Entry by Type
        vii. Search by Author
        viii. Filter by Date
        ix. Filter by File Size
        **x. Virtual Tables Required:**
            1. entry(Name, DateUpload, Author, Description, Length)

**Designing SQL queries to satisfy transaction requirements**

CREATE TABLE entries (entryNo SERIAL PRIMARY KEY, entryName text, length interval, description text, category text, dateUpload date, fileSize varchar)

1. *PSQL commands for an admin/superuser*
    a. INSERT INTO entries VALUES ( ‘Big Speech’, ‘00:02:15’, ‘A big speech given by an important person’, ‘Audio’, ‘12-04-1886’, ‘125 MB’);
        i. Inserting a 125mb file called “Big Speech” into the entry database.
    b. DELETE FROM entries WHERE (entryNo = ‘35’)
        i. Deleting an entry whose entry number is ‘35’.
    c. UPDATE entries SET dateUpload = ‘1-16-1948’ WHERE entryNo = ‘35’
        i. Modifying an entry whose number is ‘35’.
2. *PSQL commands for general user*

a. SELECT audioFile FROM Audio WHERE audioTitle = title;
    i. Retrieving an audio file to be played.
b. SELECT videoFile FROM Video WHERE videoTitle = title;
    i. Retrieving a video file to be played.
c. SELECT text FROM Transcript WHERE audioName = title;
    i. Retrieving a transcript to be read.
d. CREATE VIEW searchName AS SELECT * FROM entries WHERE entryName = name;
    i. Creating a view for the user, who searched for the entry by name.
e. CREATE VIEW searchDate AS SELECT * FROM entries WHERE uploadDate = date;
    i. Creating a view for the user, who searched by date.
f. CREATE VIEW searchType AS SELECT * FROM entries WHERE fileType = type;
    i. Creating a view for the user, who searched by file type.
g. CREATE VIEW searchAuthor AS SELECT * FROM entries WHERE fileAuthor = author;
    i. Creating a view for the user, who searched by author.
h. SELECT dateUpload FROM **the current view** ORDER BY dateUpload ASC;
    i. Displaying the results by date order.
i. SELECT fileSize FROM **the current view** ORDER BY fileSize ASC;
    i. Displaying the results by file size.