Carolyne Holmes, Gabrielle Curcio, Xuanyi Zhao
April 1st, 2021
CSC 315
Stage IV:

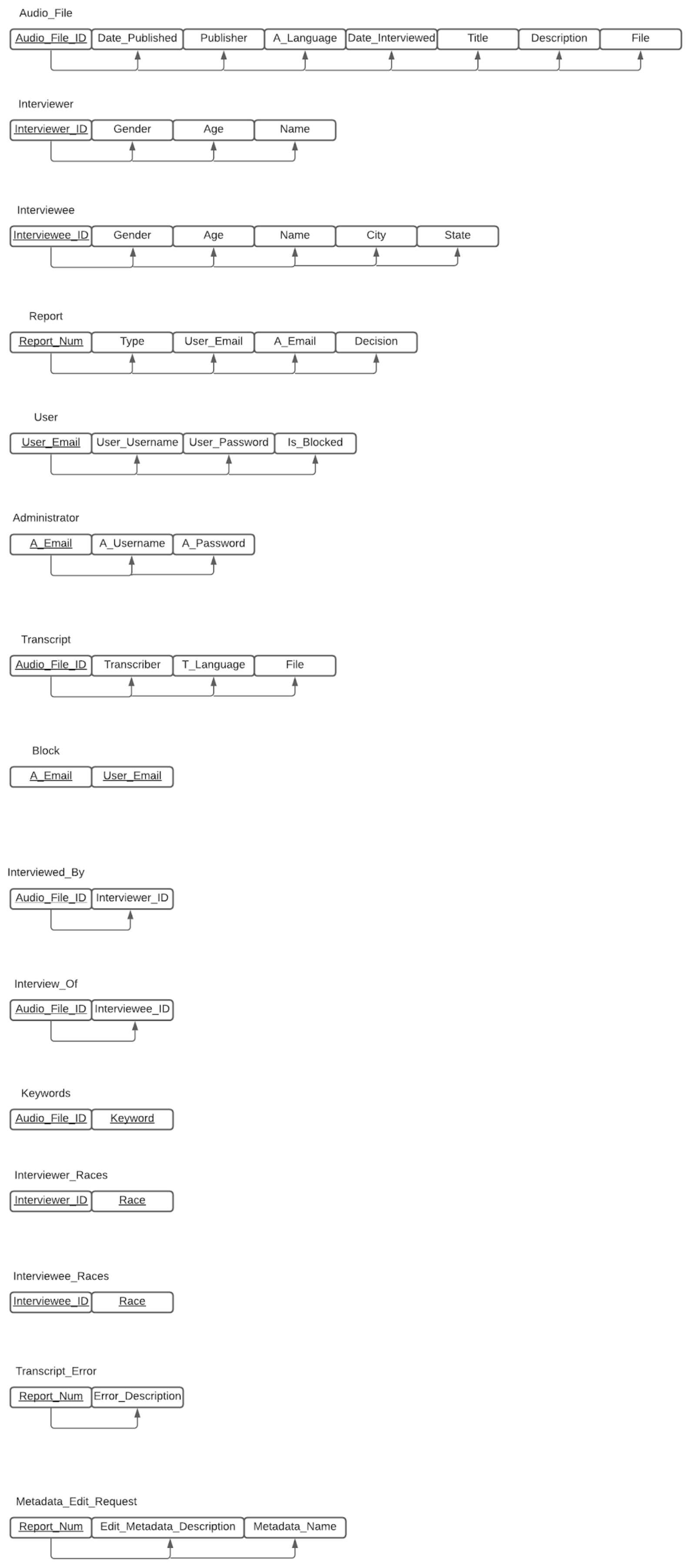
1. Review the Database Model document with stakeholders, and update the model as needed.

   We slightly edited our diagrams to remove some redundancy (like title was in transcript and audio file and the browse relation wasn't really needed)
2. Demonstrate that all the relations in the relational schema are normalized to Boyce–Codd normal form (BCNF).
   a. For each table, specify whether it is in BCNF or not, and explain why.
      i. **Audio_file:** BCNF
         1. Audio_File_ID -> Date_Published
         2. Audio_File_ID -> Publisher
         3. Audio_File_ID -> A_Language
         4. Audio_File_ID -> Date_Interviewed
         5. Audio_File_ID -> Title
         6. Audio_File_ID -> Description
         7. Audio_File_ID -> File
      ii. **Interviewer:** BCNF
         1. Interviewer_ID -> Gender
         2. Interviewer_ID -> Age
         3. Interviewer_ID -> Name
      iii. **Interviewee:** BCNF
         1. Interviewee_ID -> Gender
         2. Interviewee_ID -> Age
         3. Interviewee_ID -> Name
         4. Interviewee_ID -> City
         5. Interviewee_ID -> State
            a. We consider City and State to not be functionally dependent since a city name can belong to multiple states and a state can have multiple city values, so neither can predict a unique value for the other
      iv. **Report:** BCNF
         1. Report_Num -> Type
         2. Report_Num -> User_Email
         3. Report_Num -> A_Email
         4. Report_Num -> Decision
      v. **User:** BCNF
         1. User_Email -> User_Username
         2. User_Email -> User_Password
         3. User_Email -> Is_Blocked
      vi. **Administrator:** BCNF
         1. A_Email -> A_Username

        2. A_Email -> A_Password

   vii. **Transcript:** BCNF
        1. Audio_File_ID ->Transcriber
        2. Audio_File_ID ->T_Language
        3. Audio_File_ID ->File
   viii. **Block:** BCNF
        1. {A_Email,U_Email}
   ix. **Interviewed_by**: BCNF
        1. Audio_File_ID->Interviewer_ID
   x. **Interview_of:** BCNF
        1. Audio_File_ID ->Interviewee_ID
   xi. **Keywords:** BCNF
        1. {Audio_File_ID, Keyword}
   xii. **Interviewer_Races:** BCNF
        1. {Interviewer_ID, Race}
   xiii. **Interviewee_Races:** BCNF
        1. {Interviewee_ID, Race}
   xiv. **Transcript_Error:** BCNF
        1. Report_Num -> Error_description
   xv. **Metadata_Edit_Request:** BCNF
        1. Report_num -> Edit_metadata_description
        2. Report_num -> Metadata_name

b. For each table that is not in BCNF, show the complete process that normalizes it to BCNF.
   i. All tables were in BCNF, so not changes were made (besides those from question #1).


Audio_File
Audio_File_ID
Date_Published
Publisher
A_Language
Date_Interviewed
Title
Description
File
Interviewer
Interviewer_ID
Gender
Age
Name
Interviewee
Interviewee_ID
Gender
Age
Name
City
State
Report
Report_Num
Type
User_Email
A_Email
Decision
User
User_Email
User_Username
User_Password
Is_Blocked
Administrator
A_Email
A_Username
A_Password
Transcript
Audio_File_ID
Transcriber
T_Language
File
Block
A_Email
User_Email
Interviewed_By
Audio_File_ID
Interviewer_ID
Interview_Of
Audio_File_ID
Interviewee_ID
Keywords
Audio_File_ID
Keyword
Interviewer_Races
Interviewer_ID
Race
Interviewee_Races
Interviewee_ID
Race
Transcript_Error
Report_Num
Error_Description
Metadata_Edit_Request
Report_Num
Edit_Metadata_Description
Metadata_Name

3. Define the different views (virtual tables) required. For each view list the data and transaction requirements. Give a few examples of queries, in English, to illustrate.

   Note- all final search results will be displayed in date_published order (newest to oldest)

   a. Keyword Search View:
      i. **Data** = keywords entered by user
      ii. **Requirement** = list audio file table information for each audio file that contains (or pattern matches) the keywords entered as display the audio file title, date published, and some lines of file description.
      iii. Select from the keywords table, the audio file IDs of files that have the keyword (do this for each keyword entered sans words like 'the' or 'a' and order by best match). Take the union of the results of each keyword search (from the temporary view created) to get all relevant audio file IDs (or select via pattern matching where the keywords are OR'd together), then join the union with the audio files table to get the information of all audio files with those keywords. We may also do something along the lines of finding files that contain more of the keywords than others (like a file that has 'blue whale' and a file that has 'blue' so for the search 'big blue whale', the first file will appear first) by finding the intersection of the results of the keyword searches.
      iv. **Example** = select audio_ID where keywords = 'blue' OR 'whale' OR 'big' from keywords
   b. Interviewer Search View:
      i. **Data** = interviewer name from user search
      ii. **Requirements** = display all audio files that have an interviewer with the name that was searched
      iii. Select from the interviewer table all interviewer IDs that have a name matching that which was searched and create a temporary view of these results. Then join these interviewer IDs with the interview_by table to get the audio file IDs of all files by that interviewer and store in a view. Then join the audio IDs with the audio file table to get the information of all the relevant files, similarly to the previous table.
      iv. **Example** = select interviewer_ID where name = 'Cindy Smith' from interviewer; select audio_ID where interviewer_ID = searched_IDs from interview_by

          Select audio information from audio files where interviewer is Cindy Smith
   c. Interviewee Search View:
      i. **Data** = interviewee name searched by user
      ii. **Requirements** = display all audio files that have an interviewee with the name that was searched
      iii. Select from the interviewee table all interviewee IDs that have a name matching that which was searched and create a temporary view of these results. Then join these interviewee IDs with the interview_of table to get the audio file IDs of all files of that interviewee and store in a view. Then

join the audio IDs with the audio file table to get the information of all the relevant files, similarly to the previous table.

   iv. **Example** = select interviewee_ID where name = ‘Bob Ross’ from interviewee; select audio_ID where interviewee_ID = searched_IDs from interview_of
   Select audio information from audio files where interviewee is Bob Ross

d. Race Search View:
   i. **Data** = race searched by user
   ii. **Requirements** = display all audio files information (the same information displayed as in the other views) with interviewee race that matches what was searched; the race options are listed like checkboxes to prevent someone entering a race not found in the database
   iii. Select the interviewee IDs that have a race matching what was searched from the interviewee_race table and save the results as a view. Then use a join (or optionally a nested select) of the interviewee IDs with the interview_of table to get the audio file information of these interviewees and store the audio IDs in a view. Then use a join of these audio IDs and the audio file table to get the information of the relevant audio files.
   iv. **Example** = select interviewee_ID where race = ‘Native American’ from interviewee_race; select audio_ID where interviewee_ID = searched_IDS from interview_of
   Select audio information from audio files where interviewee is Native American

e. City/State Search View:
   i. **Data** = city and/or state selected from a checkbox list by the user
   ii. **Requirements** = display only the information about audio files in which the person being interviewed is from the searched city and/or state
   iii. In the case of the city or state only search, select the interviewee ID from the interviewee tabel when the city or state matches what was searched. If both a city and state are searched, select the interviewee ID when city and state both match what was searched. Store this as a view and perform a join with the interview_of table to get the audio_IDs that apply to the given search city/state (this could also be done with the IN operation, it depends on what is most efficient). Store these results in a view and perform a join of the view with the audio file table to get the information about the relevant audio files.
   iv. **Example** = select interviewee_ID where city = ‘Trenton’ AND state = ‘NJ’ from interviewee; select audio_ID where interviewee_ID = searched_IDs from interview_of
   Select audio information from audio files where interviewee is from Trenton, NJ

f. Reports View:
   i. **Data** = administrator decided to click a button to display reports and selected which type of report they would like to view; optionally, may allow

administrator to view reports from a searched user (by username) or only users who have not been blocked, but we haven't decided yet.

   ii. **Requirements** = display the reports, the username and email of the reporter, and part of the description of the report.
   iii. Select from the reports table all reports that match the selected type and store the report numbers in a view. Use this view in a join, IN operation, etc. with the table that matched the report type (either metadata edit or transcript error) and then project the desired information from the report so that the administrator can see it.
   iv. **Example** = select report_num where type = 'edit' from report; join the reports view with metadata_edit table; project user_email, user_username, description, and decision
   Select report information from report where type is edit

g. Blocked Users View:
   i. **Data** = administrator clicks a button to view a list of blocked users, no other data is entered
   ii. **Requirements** = show the username, user email, and administrator email or username who blocked the user
   iii. Select from the list of users the user emails of those who have is_Blocked set to true and store it as a view. Join this view with the blocked table to get the administrator email of the one who blocked them. Project the user information and administrator information for the administrator who wanted to view the blocked users.
   iv. **Example** = select user_email where is_blocked = 'true' from users; select admin_email, user_email, user_username where user_email = (IN) blocked_emails from block table
   Select block information from user where block is true

h. Delete Report:
   i. **Data** = administrator selects a report number to delete
   ii. **Requirements** = report number must exist in report table and the report should be deleted from both the report and specific report type tables
   iii. Select the report type for the chosen report number and get the corresponding specific report record (from either metadata_edit_request or transcript_error table) and delete the specific report. Then delete the tuple in the report table. To maintain referential integrity, cascade will have to be used to change invalid foreign keys after the delete to null.
   iv. **Example** = delete report #123 from metadata_edit_request; delete report #123 from report table
   Delete report #123 from report table

4. Design a complete set of SQL queries to satisfy the transaction requirements identified in the previous stages, using the relational schema and views defined in tasks 2 and 3 above.
   a. SELECT
      i. SELECT audio_file_id

```
      FROM KEYWORDS
      WHERE search = keyword;
      SELECT title, date_published, audio_file_id, description
      FROM AUDIO_FILE
      WHERE audio_file_id IN files;
ii.   SELECT interviewer_id
      FROM INTERVIEWER
      WHERE search_name = name;
      SELECT audio_file_id
      FROM INTERVIEW_BY
      WHERE interviewer_id IN interviewers;
      SELECT title, date_published, audio_file_id, description
      FROM AUDIO_FILE
      WHERE audio_file_id IN files;
iii.  SELECT interviewee_id
      FROM INTERVIEWEE
      WHERE search_name = name;
      SELECT audio_file_id
      FROM INTERVIEW_OF
      WHERE interviewee_id IN interviewees;
      SELECT title, date_published, audio_file_id, description
      FROM AUDIO_FILE
      WHERE audio_file_id IN files;
iv.   SELECT interviewee_id
      FROM INTERVIEWEE_RACE
      WHERE searched_race = race;
      SELECT audio_file_id
      FROM INTERVIEW_OF
      WHERE interviewee_id IN interviewees;
      SELECT title, date_published, audio_file_id, description
      FROM AUDIO_FILE
      WHERE audio_file_id IN files;
v.    SELECT interviewee_id
      FROM INTERVIEWEE
      WHERE searched_city = city AND searched_state = state;
      SELECT audio_file_id
      FROM INTERVIEW_OF
      WHERE interviewee_id IN interviewees;
      SELECT title, date_published, audio_file_id, description
      FROM AUDIO_FILE
      WHERE audio_file_id IN files;
vi.   SELECT report_num
      FROM REPORT
      WHERE searched_type = type;
```

SELECT report_num, user_email, user_username, description, decision
FROM (reports NATURAL JOIN METADATA_EDIT_REQUEST);

vii. SELECT user_email, user_username
FROM (USER NATURAL JOIN BLOCK);
SELECT user_email, user_username, a_email, a_username
WHERE user_email IN blocked_users;

viii. SELECT title, date_published, audio_file_id, description
FROM AUDIO_FILE;

b. INSERT
   i. INSERT INTO AUDIO_FILE
   VALUES (date_published, publisher, a_language, date_interviewed, title, description, file);
   ii. INSERT INTO INTERVIEWER
   VALUES (gender, age, name);
   iii. INSERT INTO INTERVIEWEE
   VALUES (gender, age, name, city, state);
   iv. INSERT INTO REPORT
   VALUES (type, user_email, a_email, decision);
   v. INSERT INTO USER
   VALUES (user_email, user_username, user_password, FALSE);
   vi. INSERT INTO TRANSCRIPT
   VALUES (audio_file_id, transcriber, t_language, file);
   vii. INSERT INTO KEYWORDS
   VALUES (audio_file_id, keyword);
   viii. INSERT INTO INTERVIEWER_RACES
   VALUES (interviewer_id, race);
   ix. INSERT INTO INTERVIEWEE_RACES
   VALUES (interviewee_id, race);
   x. INSERT INTO TRANSCRIPT_ERROR
   VALUES (report_num, error_description);
   xi. INSERT INTO METADATA_EDIT_REQUEST
   VALUES (report_num, metadata_edit_description, metadata_name);

c. DELETE
   i. DELETE FROM AUDIO_FILE
   WHERE audio_file_id = id_to_del;
   ii. DELETE FROM INTERVIEWER
   WHERE interviewer_id = int_id_to_del;
   iii. DELETE FROM INTERVIEWEE
   WHERE interviewee_id = int_id_to_del;
   iv. DELETE FROM REPORT
   WHERE report_num = rnum_to_del;
   v. DELETE FROM USER
   WHERE user_email = u_email_to_del;
   vi. DELETE FROM TRANSCRIPT

WHERE audio_file_id = id_to_del;

vii. DELETE FROM KEYWORDS
WHERE audio_file_id = id_to_del;

viii. DELETE FROM INTERVIEWER_RACES
WHERE interviewer_id = int_id_to_del;

ix. DELETE FROM INTERVIEWEE_RACES
WHERE interviewee_id = int_id_to_del;

x. DELETE FROM TRANSCRIPT_ERROR
WHERE report_num = rnum_to_del;

xi. DELETE FROM METADATA_EDIT_REQUEST
WHERE report_num = rnum_to_del;

d. UPDATE

i. UPDATE AUDIO_FILE
SET (title = new_title, description = new_descrip)
WHERE audio_file_id = desired_id;

ii. UPDATE INTERVIEWER
SET (name = new_name, gender = new_gender)
WHERE interviewer_id = desired_id;

iii. UPDATE INTERVIEWEE
SET (name = new_name, gender = new_gender)
WHERE interviewee_id = desired_id;

iv. UPDATE USER
SET (user_username = new_username, user_password = new_password, is_blocked = new_blocked_status)
WHERE user_email = desired_email;

v. UPDATE ADMINISTRATOR
SET (a_username = new_username, a_password = new_password)
WHERE a_email = desired_email;

vi. UPDATE TRANSCRIPT
SET (transcriber = new_transcriber, t_language = new_language)
WHERE audio_file_id = desired_id;

vii. UPDATE KEYWORDS
SET (keyword = new_keyword)
WHERE audio_file_id = desired_id;

viii. UPDATE INTERVIEWER_RACES
SET (race = new_race)
WHERE interviewer_id = desired_id;

ix. UPDATE INTERVIEWEE_RACES
SET (race = new_race)
WHERE interviewee_id = desired_id;