CSC 315 Project Database Model Document
Alexander Benasutti, Raymond Chow, Sam Haseloff, Alexander Viola
GitHub: https://github.com/TCNJ-degoodj/stage-iv-group-11
Group 11

Boyce-Codd Normal Form (BCNF)

Normal Form Requirements:

- 1NF
  - Single valued attributes
- 2NF
  - 1NF
  - No partial dependency
    - If the primary key returns (duplicated) information that could be in another relation
- 3NF
  - 2NF
  - No transitive dependency
- BCNF
  - 3NF
  - For every dependency $A \rightarrow B$, A is a superkey

Every table satisfies the above requirements. Therefore, all of the tables in the schema are in BCNF.

Users:

- int id: primary key (identity)
- varchar name
- varchar email (unique)
- varchar password_hash
- int user_type = 0 (user)

There are no multi-valued attributes (1NF), there are no partial dependencies (2NF), there are no transitive dependencies (3NF), and A (id) is a superkey for every non-prime attribute (BCNF). Here, the *email* column is not used as a prime attribute; the unique constraint is only for relational integrity.

Audio:

- int id: primary key (identity)
- varchar name
- (long string type) description (nullable)
- date audio_date (nullable)
- date upload_date

There are no multi-valued attributes (1NF), there are no partial dependencies (2NF), there are no transitive dependencies (3NF), and A (id) is a superkey for every non-prime attribute (BCNF).

Transcripts:

- int id: primary key (identity)
- int audio_id (Audio: id)
- (long text) text
- text revision_comment (nullable)
- bool is_latest = true

There are no multi-valued attributes (1NF), there are no partial dependencies (2NF), there are no transitive dependencies (3NF), and A (id) is a superkey for every non-prime attribute (BCNF). While *audio_id* references another table, it does not act as part of the primary key as the primary key is the identity column *id*.

TagCategory:

- int id: primary key (identity)
- varchar name (unique)

There are no multi-valued attributes (1NF), there are no partial dependencies (2NF), there are no transitive dependencies (3NF), and A (id) is a superkey for every non-prime attribute (BCNF). The column *name* is unique for integrity, but mostly visual integrity. For example, there shouldn't be two categories named "Location" showing up at the search sidebar/panel.

Tags:

- int id: primary key (identity)
- int category_id (TagCategory: id)
- varchar name

There are no multi-valued attributes (1NF), there are no partial dependencies (2NF), there are no transitive dependencies (3NF), and A (id) is a superkey for every non-prime attribute (BCNF). The column *category_id* is not part of a composite key; our primary key is the *id* identity column. The column *name* is unique for integrity purposes, but mostly visual integrity.

AudioTag (+ unique constraint with both columns):

- int audio_id (Audio: id)
- int tag_id (Tags: id)

There are no multi-valued attributes (1NF), there are no partial dependencies (2NF), there are no transitive dependencies (3NF), and there are no non-prime attributes (BCNF).

UserEditRequest:

- int id: primary key (identity)
- int user_id (Users: id)

- int transcript_id (Transcripts: id)
- (long text) text
- varchar edit_comment (nullable)
- bool request_approved = false
- timestamp create_time

There are no multi-valued attributes (1NF), there are no partial dependencies (2NF), there are no transitive dependencies (3NF), and A (id) is a superkey for every non-prime attribute (BCNF). There can be multiple edit requests by the same user for a single transcript. Therefore, each request is uniquely identified by the *id* identity column and not a composite key of *user_id* + *transcript_id*.

AuditLogs:

- int id: primary key (identity)
- int user_id (Users: id)
- varchar type (type of log: e.g. approval of edit request, ban user(?), etc.)
- jsonb data
- timestamp create_time

There are no multi-valued attributes (1NF), there are no partial dependencies (2NF), there are no transitive dependencies (3NF), and A (id) is a superkey for every non-prime attribute (BCNF). While *user_id* references another table, it does not act as part of the primary key as the primary key is the identity column *id*. The *data* column can contain arbitrary data, though it will only ever contain single objects.

Required Views/Virtual Tables

Transaction Requirements = What need to be the possible user actions that query the database?

- User needs to be able to select filters and generate audio recording results
- Users need to be able to create and submit an edit request to a transcript
- Administrators need to be able to view transcript edit requests and approve or deny them

Data Requirements = Data involved in/queried upon user action

- Filter Listing
    - Tag id
    - Tag category_id
    - Tag name

Example Query: A user goes to select certain filters within the Trentoniana oral history archives in order to properly search for what audio file/transcript they are looking for. This view will return the name of the tags with their categories for use by the server.

- Results
    - Audio id
    - Audio name
    - Audio upload_date
    - Transcript text AS transcript_text

Example Query: A user makes their selection of filters on the Trentoniana oral history archives and queries the server database for what results these apply to. This view will return the names of the audio files and their upload date, in addition to their associated transcripts if available.

- Pending User Submitted Transcript Edit Requests
    - Audio name
    - Transcript id
    - Transcript text AS old_text
    - User_Edit_Request text AS new_text
    - User_Edit_Request user_id
    - User_Edit_Request id AS edit_id
    - User_Edit_Request edit_comment
    - User_Edit_Request create_time

Example Query: A user sees some typo or incorrect information within an audio file's associated transcript. In an effort to make a correction, they open a transcript edit request. This view will return the name of the audio file whose transcript is being edited, in addition to the associated text which is being edited. The user can also add an edit comment to leave a reason for their edit.

- Administrator Transcript Edit Request Approval
    - Audio name
    - Transcript id
    - User_Edit_Request edit_comment
    - Transcript text AS old_text
    - User_Edit_Request text AS new_text
    - User_Edit_Request id AS edit_id
    - User_Edit_Request edit_comment
    - User_Edit_Request request_approved AS approval_status

Example Query: An administrator sees that there is an edit request to a transcript which is awaiting approval. This view will return the transcript being edited in addition to the edited text of said transcript. They can view the edit comment left by the user, and also approve or deny the edit request, leaving a revision comment if they choose to approve it.

SQL Queries for Views

- **Filter Listing:**

```
SELECT tags.id AS tag_id, tag_category.name AS category_name,
tags.name AS tag_name FROM tags JOIN tag_categoryON
tag_category.id = tags.category_id;
```

- **Transcript Requests:**

```
SELECT
      user_edit_request.id AS request_id,
      user_edit_request.user_id,
      transcripts.id AS transcript_id,
      audio.name AS audio_name,
      transcripts.text AS old_text,
      user_edit_request.text AS new_text,
      user_edit_request.edit_comment AS edit_comment,
      user_edit_request.create_time AS request_time
FROM user_edit_request
INNER JOIN transcripts ON user_edit_request.transcript_id =
transcripts.id
INNER JOIN audio ON audio.id = transcripts.audio_id
WHERE user_edit_request.request_approved IS NULL;
```

- **Admin Transcript Request View:**

```
SELECT
      user_edit_request.id AS request_id,
      user_edit_request.user_id,
      transcripts.id AS transcript_id,
      audio.name AS audio_name,
      transcripts.text AS old_text,
      user_edit_request.text AS new_text,
      user_edit_request.edit_comment AS edit_comment,
      user_edit_request.create_time AS request_time,
      user_edit_request.request_approved AS approval_status
FROM user_edit_request
INNER JOIN transcripts ON user_edit_request.transcript_id =
transcripts.id
INNER JOIN audio ON audio.id = transcripts.audio_id
WHERE user_edit_request.request_approved IS NULL;
```

- **Results:**

```
SELECT audio.id, audio.name, audio.upload_date, transcripts.text
AS transcript_text FROM audio
JOIN transcripts ON transcripts.audio_id = audio.id
WHERE transcripts.is_latest = TRUE;
```