Thomas Orth, Matthew Van Soelen, Justin Pabon, Kathleen Burke

CSC 315-01 and LNG 370/HON 270 Collaborative Project Stage IV

Design

Part 2: Table Examination for BCNF compliance

Users Table:

This table is in BCNF due to the following. Take the following superkeys that are available in User:

- *Email*
- *User_id*
- *Password_hash*

These are the only superkeys in the table as they cannot have duplicates and if any of the attributes are deleted from the key, it is no longer a valid key. That leaves the only non-prime attribute columns to *LFlag*. Note the following Dependencies:

- *Email* → *User_id*
- *Email* → *Password_hash*
- *Email* → *LFlag*
- *User_id* → *Email*
- *User_id* → *Password_hash*
- *User_id* → *LFlag*
- *Password_hash* → *Email*
- *Password_hash* → *User_id*
- *Password_hash* → *LFlag*

There are no columns that are Functional Dependencies of LFlag. Therefore, this table is in BCNF.

Transcripts Table:

This table was originally not in BCNF as it had violated a 1NF constraint of no multivalued attributes. Upon review, we had determined that keywords were essentially a multivalued attribute. We split it from the transcripts table into a separate “Keywords” table and connect it to the Transcripts table via the “Describes” relationship.

This table is now in BCNF due to the following. Take the following superkeys that are available in User:

- *transcript_id* (Primary Key)

- *text_file_path* (candidate key)
- *audio_file_path* (candidate key)
- *text* (candidate key)
- *title* (candidate key)
- *summary* (candidate key)

These are the only superkeys in the table as they cannot have duplicates and if any of the attributes are deleted from the key, it is no longer a valid key. There are no non-prime attributes in this table. That is because each transcript is considered unique for these attributes and therefore, there cannot be duplicate values in the tuples. This means that this table only consists of candidate keys/prime attributes. The table/relation is therefore in BCNF.

Locations Table:

This table is in BCNF due to the following. Take the following superkeys that are available in User:

- *location_id* (Primary key)
- *street_name* (candidate key)

These are the only superkeys in the table as they cannot have duplicates and if any of the attributes are deleted from the key, it is no longer a valid key. Note the following Dependencies:

- *location_id* → *street_name*
- *street_name* → *location_id*

The only FD that exists is between superkeys. Therefore, it is in BCNF.

Participants Table:

This table is in BCNF due to the following. Take the following superkeys that are available in Participants:

- *p_id* (primary key)
- *name* (candidate key)

These are the only superkeys in the table as they cannot have duplicates and if any of the attributes are deleted from the key, it is no longer a valid key. Note the following Dependencies:

- *p_id* → *name*
- *name* → *p_id*

The only FD that exists is between superkeys. Therefore, it is in BCNF.

Keywords Table:

This table is in BCNF due to the following. Take the following superkeys that are available in Keywords:

- *k_id* (Primary key)
- *keyword* (candidate key)

These are the only superkeys in the table as they cannot have duplicates and if any of the attributes are deleted from the key, it is no longer a valid key. Note the following Dependencies:

- *k_id* → key*word*
- *keyword* → *k_id*

The only FD exists between superkeys. Therefore, this table is in BCNF.

Mentions Table:

This table is in BCNF. It consists solely of its primary key, (*transcript_id, location_id*), for which it is only functional dependent on and if you remove any of the attributes in that key, it would not be a key. Therefore, it is a super key. Since the table only has a superkey, it is in BCNF.

Bookmarks Table:

This table is in BCNF. It consists solely of its primary key, (*transcript_id, user_id*), for which it is only functional dependent on and if you remove any of the attributes in that key, it would not be a key. Therefore, it is a super key. Since the table only has a superkey, it is in BCNF.

Participates Table:

This table is in BCNF. It consists solely of its primary key, (*transcript_id, p_id*), for which it is only functional dependent on and if you remove any of the attributes in that key, it would not be a key. Therefore, it is a super key. Since the table only has a superkey, it is in BCNF.

Describes Table:

This table is in BCNF. It consists solely of its primary key, (*transcript_id, k_id*), for which it is only functional dependent on and if you remove any of the attributes in that key, it would not be a key. Therefore, it is a super key. Since the table only has a superkey, it is in BCNF.

Part 3: Views

We would need three views:

- Participant_transcript_view
    - This is a joining of Participants and Transcripts tables based on the Participates relation.
    - Queries:
        - Joining with the Location_transcript_view and Keyword_transcript_view to have a full joined database (Full_Transcript)
- User_transcript_view
    - This is a joining of transcript and user tables based on the bookmarks relation.
    - Queries:
        - Select all entries that match a given *user_id*

            - Filter on the above selected entries based on partial matching of *title*, *summary* or other transcript attributes
    - Location_transcript_view
        - This view is a joining of the Locations and Transcripts tables.
        - Queries:
            - Joining with the Participant_transcript_view and Keyword_transcript_view to have a full joined database (Full_Transcripts)
    - Keyword_transcript_view
        - This view is a joining of the Keywords and Transcripts tables
        - Queries:
            - Only the aforementioned joining
    - Full_Transcript_view
        - This view is a joining of Participant_transcript_view and Location_transcript_view
        - Queries:
            - Filtering on partial or full matches of *title, summary, keywords, tex_content, name (participant), word (keyword)* or *street_name*
            - Examples:
                - Find all transcript information whose title partially matches “trenton area”
                - Find all transcripts whose participant’s names partially match “joel”
                - Find all transcripts whose mentioned locations partially match “main street”
                - Find all transcripts who has the keyword “urban” that describes the transcript
                - Find all transcripts whose text content contains “welcome to today’s meeting”

Part 4: Queries

CREATE TABLE and CREATE VIEW queries

```
CREATE TABLE transcripts (
  transcript_id SERIAL PRIMARY KEY,
  title text UNIQUE,
  summary text UNIQUE,
  audio_file_path text UNIQUE,
  text_file_path text UNIQUE,
  text_content text UNIQUE
);

CREATE TABLE users (
  user_id SERIAL PRIMARY KEY,
  email text UNIQUE,
  password_hash text UNIQUE,
```

```
  lflag int
);

CREATE TABLE participants (
  p_id SERIAL PRIMARY KEY,
  name text UNIQUE
);

CREATE TABLE locations (
  location_id SERIAL PRIMARY KEY,
  street_name text UNIQUE
);

CREATE TABLE keywords (
  k_id SERIAL PRIMARY KEY,
  keyword text UNIQUE
);

CREATE TABLE bookmarks (
  user_id int REFERENCES users,
  transcript_id int REFERENCES transcripts,
  PRIMARY KEY (user_id, transcript_id)
);


CREATE TABLE participates (
  p_id int REFERENCES participants,
  transcript_id int REFERENCES transcripts,
  PRIMARY KEY (p_id, transcript_id)
);

CREATE TABLE mentions (
  location_id int REFERENCES locations,
  transcript_id int REFERENCES transcripts,
  PRIMARY KEY (location_id, transcript_id)
);

CREATE TABLE describes (
  k_id int REFERENCES keywords,
  transcript_id int REFERENCES transcripts,
```

```
  PRIMARY KEY (k_id, transcript_id)
);

CREATE VIEW participant_transcript_view AS
  SELECT * FROM (SELECT participants.*, transcripts.* FROM participates
            LEFT OUTER JOIN participants ON participants.p_id = participates.p_id
            LEFT OUTER JOIN transcripts ON transcripts.transcript_id =
participates.transcript_id) as PTV;

CREATE VIEW location_transcript_view AS
  SELECT * FROM (SELECT locations.*, transcripts.* FROM mentions
            LEFT OUTER JOIN locations ON locations.location_id = mentions.location_id
            LEFT OUTER JOIN transcripts ON transcripts.transcript_id =
mentions.transcript_id) as LTV;

CREATE VIEW user_transcript_view AS
  SELECT * FROM (SELECT users.*, transcripts.* FROM bookmarks
            LEFT OUTER JOIN users ON users.user_id = bookmarks.user_id
            LEFT OUTER JOIN transcripts ON transcripts.transcript_id =
bookmarks.transcript_id) as UTV;

CREATE VIEW keyword_transcript_view AS
  SELECT * FROM (SELECT keywords.*, transcripts.* FROM describes
            LEFT OUTER JOIN keywords ON keywords.k_id = describes.k_id
            LEFT OUTER JOIN transcripts ON transcripts.transcript_id =
describes.transcript_id) as UTV;

 CREATE VIEW full_transcript_view AS
  SELECT * FROM (SELECT keyword_transcript_view.keyword,
keyword_transcript_view.k_id, location_transcript_view.location_id,
location_transcript_view.street_name, participant_transcript_view.* FROM
participant_transcript_view
            LEFT OUTER JOIN keyword_transcript_view ON
keyword_transcript_view.transcript_id = participant_transcript_view.transcript_id
            LEFT OUTER JOIN location_transcript_view ON
location_transcript_view.transcript_id = participant_transcript_view.transcript_id) as FT;
```

INSERT Examples. NOTE: Variables are placeholders. They can be replaced with any acceptable values based on the schema.

INSERT INTO transcripts
VALUES (title, summary, audio_path, text_path, text);

INSERT INTO participants
VALUES (name);

INSERT INTO keywords
VALUES (keyword);

INSERT INTO locations
VALUES (street_name);

INSERT INTO participates
VALUES (participant_id, transcript_id);

INSERT INTO mentions
VALUES (location_id,transcript_id);

INSERT INTO describes
VALUES (k_id,transcript_id);

INSERT INTO bookmarks
VALUES (user_id, transcript_id);

User Filtering for id (needed to localize data per user)

SELECT user_id FROM users WHERE email = email AND password_hash = password_hash;

Transcript Filtering (search is a placeholder for the search string done by the user)

SELECT * FROM full_transcript_view WHERE title ILIKE '%search%';

SELECT * FROM full_transcript_view WHERE text_content ILIKE '%search%';

SELECT * FROM full_transcript_view WHERE summary ILIKE '%search%';

SELECT * FROM full_transcript_view WHERE keyword = search;

SELECT * FROM full_transcript_view WHERE street_name ILIKE '%search%';

SELECT * FROM full_transcript_view WHERE name ILIKE ‘%search%’;

Bookmarks Filtering (search is a placeholder for the search string done by the user and id is the currently signed in user)

SELECT * FROM user_transcript_view WHERE user_id = id AND title ILIKE ‘%search%’;

SELECT * FROM full_transcript_view WHERE user_id = id AND text_content ILIKE ‘%search%’;

SELECT * FROM full_transcript_view WHERE user_id = id AND summary ILIKE ‘%search%’;

SELECT * FROM full_transcript_view WHERE user_id = id AND keyword = search;

SELECT * FROM full_transcript_view WHERE user_id = id AND street_name ILIKE ‘%search%’;

SELECT * FROM full_transcript_view WHERE user_id = id AND name ILIKE ‘%search%’;

Delete entries (let id be a possible value for the id columns)

DELETE FROM transcripts where transcript_id = id;

DELETE FROM locations where location_id = id;

DELETE FROM keywords where k_id = id;

DELETE FROM participants where p_id = id;

DELETE FROM participates where p_id = id;

DELETE FROM participates where transcript_id = id;

DELETE FROM describes where k_id = id;

DELETE FROM describes where transcript_id = id;

DELETE FROM mentions where location_id = id;

DELETE FROM mentions where transcript_id = id;

DELETE FROM bookmarks where transcript_id = id;

Update Transcript (Strings are arbitrary. Id is a possible transcript id)

Update Transcript
SET title='title',
SET text_content='lorem ipsum',
SET summary='summary'
WHERE transcript = id;