# COEN 280 HW 2

Justin Li

**Task 1: Normal Forms [25pts]**

Given the relation R and the following functional dependencies, answer the following questions.

R(A, B, C, D, E)

All attributes contain only atomic values.

FD1: A → BC

FD2: CD → E

FD3: B → D

FD4: A → E

[5 pts] Compute A+, the attribute closure of attribute A.

A+:

a) A → BC  b) CD → E  c) B → D  d) A → E  (given)
e) A → B  f) A → C  (a and decomposition)
g) A → D  (e, c, and transitivity)
h) A → ABCDE  (e, f, g, d, and union)

[5 pts] List the candidate keys of R.

A

[5 pts] What's the highest normal form that R satisfies and why?

2NF, there are no partial dependencies, but it violates 3NF because there is an implied transitive functional dependency A → D because of A → B and B → D.

[5 pts] If R is not already at least in 3NF, then normalize R into 3NF and show the resulting relation(s) and their candidate keys. Your decomposition should be both join-lossless and dependency-preserving. If R is already in 3NF, just list the candidate keys of R.

| | |
|---|---|
| R0 (A, B, C) | key A |
| R1 (B, D) | key B |
| R2 (C, D, E) | key C,D |

[5 pts] Is your answer to (d) also in BCNF? Why or why not?

Yes, all functional dependencies are key constraints

**Task 2: Normal Forms [25 pts]**

Given the relation R and the following functional dependencies, answer the following questions.

R(A, B, C, D, E, F, G)

All attributes contain only atomic values.

FD1: $AB \rightarrow C$

FD2: $BC \rightarrow E$

FD3: $B \rightarrow D$

FD4: $F \rightarrow G$

[5 pts] Compute A+, the attribute closure of attribute A.
A+:

a) $A \rightarrow A$ (trivial)

[5 pts] List the candidate keys of R.

AB+:

a) $AB \rightarrow C$ b) $BC \rightarrow E$ c) $B \rightarrow D$ d) $F \rightarrow G$ (given)
e) $AB \rightarrow A$ f) $AB \rightarrow B$ (e and d and reflexivity)
g) $AB \rightarrow D$ (g, c, and transitivity)
h) $AB \rightarrow E$ (f, a, b, and transitivity)
i) $AB \rightarrow ABCDE$ (e, f, a, g, h, and union)

now missing only G… which we can only obtain from F
Therefore, candidate key = ABF

[5 pts] What's the highest normal form that R satisfies and why?

1NF since we have a partial dependency of $B \rightarrow D$, which violates 2NF

[5 pts] If R is not already at least in 3NF, then normalize R into 3NF and show the resulting relation(s) and their candidate keys. Your decomposition should be both join-lossless and dependency-preserving. If R is already in 3NF, just list the candidate keys of R.

| | |
|---|---|
| R0 (B, D) | key B |
| R1 (F, G) | key F |
| R2 (A, B, C) | key A,B |
| R3 (B, C, E) | key B,C |
| R4 (A, B, F) | key A, B, F |

[5 pts] Is your answer to (d) also in BCNF? Why or why not?

Yes, all functional dependencies are key constraints

**SQL (Point 50)**

Make a copy of colabLinks to an external site. (don't edit in-place), and review the cells that show you how to write and execute SQL in colab.

Add a new cell and write the following SQL queries in your copy of the colab and test them.

Copy them to your assignment solution doc for submission. You don't need to share your colab in the solution.

1. Find artists that have created songs that belong to Pop genres, but not Rock genre.
```
qry_pop_not_rock = """
-- Find artists that have created songs that belong to Pop genres,
but not Rock genre.
SELECT DISTINCT Songs.artist
FROM Songs
WHERE Songs.genre = "Pop" and Songs.genre != "Rock";"""
runSql("Pop not Rock", qry_pop_not_rock);
```

2. List of unique genres in the Songs table.
```
qry_unique_genres = """
-- List of unique genres in the Songs table
SELECT DISTINCT genre
FROM Songs
WHERE genre IS NOT NULL;"""
runSql('Unique genres in the Songs table', qry_genres)
```

3. Find number of songs in each genre.
```
qry_num_songs = """
SELECT genre, COUNT(Songs.title) AS count
FROM Songs
WHERE genre IS NOT NULL
GROUP BY genre;"""
runSql("Number of songs in each genre", qry_num_songs)
```

4. Find number of songs by each artist in a genre (i.e. count songs by genre-artist).
```
qry_songs_genre = """
SELECT genre, Songs.artist, COUNT(*) AS count
FROM Songs
WHERE genre IS NOT NULL
GROUP BY genre, Songs.artist;"""
runSql("Number of songs by each artist in a genre", qry_songs_genre)
```

5. Find number of songs that have "hip hop" in their genre.
```
qry_hip_hop = """
SELECT COUNT(*) AS count
FROM Songs
WHERE genre = "hip hop"
GROUP BY genre;"""
runSql("Number of songs that have 'hip hop' in genre", qry_hip_hop)
```

6. Find popular songs based on the number of times a long is listened to. List results by num of listens decreasing.
```
qry_popular_songs = """
SELECT Songs.title, COUNT(*) AS listened
FROM Songs, Listens
WHERE Songs.song_id = Listens.song_id
GROUP BY Songs.title
ORDER BY listened DESC;"""
runSql('Popular songs by listens', qry_popular_songs)
```

7. Find Avg rating for all songs by Beatles.
```
qry_avg_beatles = """
SELECT AVG(Listens.rating) AS avg_rating
FROM Songs, Listens
WHERE Listens.song_id = Songs.song_id and Songs.artist =
'Beatles';"""
runSql('Average Beatles listens', qry_avg_beatles)
```

8. Update the query #7 to exclude songs that don't have a rating in the table in your query.
```
qry_avg_beatles = """
SELECT AVG(Listens.rating) AS avg_rating
FROM Songs, Listens
WHERE Listens.song_id = Songs.song_id and Songs.artist = 'Beatles'
and
Listens.rating IS NOT NULL;"""
runSql('Average Beatles listens', qry_avg_beatles)
```

9. Find number of users who have given avg song rating > 4.
```
qry_avg_over_4 = """
SELECT Users.name, AVG(Listens.rating) AS avg_rating
FROM Users, Listens
WHERE Users.user_id = Listens.user_id and Listens.rating IS NOT NULL
```

```
GROUP BY Users.name
HAVING AVG(Listens.rating) > 4;"""
runSql('Number of users who have given avg song rating > 4',
qry_avg_over_4)
```

10. Find Avg song ratings for Taylor Swift songs that have been listened to by at least one person.

```
qry_avg_taylor = """
SELECT Songs.title, AVG(Listens.rating) AS avg_rating
FROM Songs, Listens
Where Songs.song_id = Listens.song_id and Songs.artist = 'Taylor
Swift'
GROUP BY Songs.title;"""
runSql('Avg song ratings for Taylor Swift songs', qry_avg_taylor)
```