COL362 Project Milestone 2

Arush Utkarsh (2020CS10584) Rajeev Gupta (2019CS51018)
Vibhanshu Gupta (2019PH10665)

April 2023

1 Introduction

This submission describes the features that we have planned to implement as of now, the queries required to run these features, indexes created to enhance the performance and the general time performance of the queries. Note that the performance has been tested on the local machine because we were getting some error creating indexes on the portal because of owner restrictions. Performance varies slightly depending on which value is searched for, but has a maximum deviation of around 5 percent.

2 Updates in Relations from Milestone 1 Submission

We have 2 major tables in our database, the songs table with 13853 relations and the composition table with 135803 relations. Previously we decided that all the detailed information of the song will be in the composition table but since we do not have detailed information for all the songs, we are deciding to put an extra feature where users can add any detail to the song and it will be saved in the songs table. Now songs table has all the songs about which we have all the information which is 13k and the composition table has all the other songs we have 135k. We will give the option of adding pre-decided features about the song like tempo key etc.

3 Features

- **'Search for Artist'**: The user searches for a particular artist, in order to get options of songs among which the user intends to choose a song to learn.

  Query:

  ```
  \set artist '\'Johnny Cash\''
  ```

```
    CREATE INDEX Composition2 ON Composition(song,artist);
    CREATE INDEX Songs2 ON Songs(name,artist_names);
    SELECT song FROM
     (SELECT * FROM Composition WHERE artist = :artist) AS t1
   LEFT JOIN
     (SELECT * FROM Songs WHERE E'\''||:artist||E'\'' = ANY(artist_names)) AS t2
   ON t1.song = t2.name;
```

Time:
Without Index: 45 ms
With Index: 13 ms

- **'Search for Song'**: The user searches for a song and gets the matching artists (since multiple artists may have songs of the same title) among which user can choose one song.

  Query:

```
    \set song '\'Smells Like Teen Spirit\''
    CREATE INDEX Composition1 ON Composition(song,artist);
    CREATE INDEX Songs1 ON Songs(name,artist_names);
    SELECT artist, artist_names FROM
     (SELECT * FROM Composition WHERE song = :song) AS t1
   LEFT JOIN
     (SELECT * FROM Songs WHERE name = :song) AS t2
   ON E'\''||t1.artist||E'\'' = ANY(t2.artist_names);
```

  Time:
  Without Index: 50ms
  With Index: 2 ms

- **'Finalize a Song/Search for Artist and Song'**:The user enters both artist and song, or alternatively, chooses one of the results generated from a search to view the details about this particular song.

  Query:

```
    \set song '\'Black Hole Sun\''
    \set artist '\'Soundgarden\''

    1.
    CREATE INDEX Composition3 ON Composition(song,artist);
    CREATE INDEX Songs3 ON Songs(name,artist_names);
    CREATE INDEX Artist1 ON Artist(name);
    SELECT * FROM
    (SELECT * FROM
            (SELECT * FROM Composition WHERE artist = :artist AND song = :song)
                   AS t1
        LEFT JOIN
            (SELECT * FROM Songs WHERE name = :song AND E'\''||:artist||E'\'' =
                                                         ANY(artist_names)) AS t2
        ON 1=1
    ) AS t3
  JOIN
    (SELECT * FROM Artist Where name = :artist) AS t4
  ON 1=1;
```

```
 2.
 CREATE INDEX Composition4 ON Composition(song,artist);
 CREATE INDEX Chords1 ON Chords(name);
 SELECT name, barre, position FROM
  Chords
   JOIN
  (SELECT chordlist FROM Composition WHERE song = :song AND artist = :artist)
            AS t1
   ON E'\''||Chords.name||E'\'' = ANY(t1.chordlist);
```

Time:
Query 1:
Without Index: 130 ms
With Index: 2 ms
Query 2:
Without Index: 40 ms
With Index: 2 ms

- **'Search for Genre'**:The user searches for a genre, and is returned all the artists who are associated with that genre. The user can then choose one of these artists and go through their songs to choose which one to learn.
  Query:

  ```
  \set genre '\'blues\''
  CREATE INDEX Artist2 ON Artist(genres_and_tags);
  SELECT name FROM Artist WHERE genres_and_tags LIKE '%'||:genre||'%';
  ```

  Time:
  Without Index: 80 ms
  With Index: 80 ms
  There is no speedup here because the indexes don't help with like operator unless the string starting is provided.

- **'Search for Award Category'**:The user chooses an award category, with the intention of learning only those songs which have won an award in the chosen category.
  Query:

  ```
     \set category '\'Video of the Year\''
     CREATE INDEX Awards1 ON Awards(category,nominee);
     CREATE INDEX Composition5 ON Composition(artist);
     SELECT t1.artist, song FROM
      (SELECT * FROM Awards WHERE category = :category) AS t1
    JOIN
      Composition
    ON t1.nominee = Composition.song AND t1.artist = Composition.artist;
  ```

  Time:
  Without Index: 70 ms
  With Index: 7 ms

- **'Inspire Me'**:The user doesn't have a particular song in mind to learn today. So, they want us to choose a random song for them.
  Query:

  ```
     SELECT artist, song FROM
      Composition
    ORDER BY RANDOM()
    LIMIT 1;
  ```

  Time: 100 ms (No use of an index here)

- **Complexity**: For different setting we need to change x, y and p variables in query:

  ```
  CREATE INDEX tempo_index_songs ON songs(name, tempo);
  CREATE INDEX song_hordlist_index_comp ON composition(song, chordlist);
  CREATE INDEX chords_index ON chords(name, barre);

  select name
  from songs
  where tempo between x and y
  union
  select song
  from (
          select song, unnest(chordlist) as chordlist
          from composition
  ```

```
        where array_length(chordlist, 1) between x and y
) as t1
inner join (
        select *
        from chords
        where barre= p
) as t2
on t1.chordlist like '%' || t2.name || '%';-- here chordlist is a string, unnest used.

DROP INDEX tempo_index_songs;
DROP INDEX song_hordlist_index_comp;
DROP INDEX chords_index;
```

Time:
Without index: 1977.319
With index: 1409.209 ms

- **Moods**

```
CREATE INDEX comp_index ON composition(song, tags, chordlist);

-- Single query for different moods with arr as our variable
-- arr: string of genres that define that mood
-- Example: '%rap%', '%club%', '%metal%', '%edm%'  => for Energy
-- or  '%club%', '%samba%', '%funk%', '%house%', '%disco%' => for Fun




SELECT DISTINCT song
FROM (
SELECT song, unnest(chordlist) as chordlist FROM composition
WHERE tags LIKE ANY(ARRAY[ '%club%', '%samba%', '%funk%', '%house%', '%disco%' ])
) as t1
INNER JOIN chords
ON t1.chordlist LIKE '%'|| chords.name || '%';

DROP INDEX comp_index;
-- ( for arr: '%club%', '%samba%', '%funk%', '%house%', '%disco%' )
```

Query time without index table: 1989.921 ms Query time with index table:
1389.845 ms