CPSC 304 Project Cover Page

Milestone #: 4

Date: November 27, 2024

Group Number: 71

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Andrew Xie	23613136	x3s5u	adxie12@gmail.com
Alice Sin	16582144	g2z0b	sin.alicee@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

DESCRIPTION

Our application is used to manage information on record labels and its operation. The application allows users to add and remove record labels, add artists, and add and update contracts. Users can also make several queries about the songs and albums that have been released by different artists.

DIFFERENCES IN SCHEMA

We made several changes to our schema:

1. ON UPDATE CASCADE statements were taken out because the Oracle database does not support update statements.
2. We modified WritesContract1 and WritesContract2 because the type attribute (contract type) in WritesContract1 alone cannot be a primary key – there can be multiple artists with the same type of contract across different labels. Instead, we modified it so that the primary key is now type and stageName, as each artist cannot have two of the same types of contracts.
   In WritesContract2, we added (type, StageName) as UNIQUE, since those two together must be unique.

SQL QUERIES

INSERT
appService.js: line 326

DELETE
appService.js: line 202

UPDATE
appService.js: line 419

SELECTION

appService.js: line 162

PROJECTION
appService.js: line 100

JOIN
appService.js: line 131

Aggregation with GROUP BY
appService.js: line 443

```
SELECT stageName, professionalName, MAX(numTracks)
FROM Album
WHERE numTracks <15
GROUP BY stageName, professionalName
ORDER BY ;
```

For each album by an artist-producer duo, find the maximum number of tracks on an album under 15 tracks. Displays the artist name, producer name, and number of tracks.

Aggregation with HAVING
appService.js: line 462

```
SELECT stageName, COUNT(*)
FROM Album
WHERE numTracks > 9
GROUP BY stageName
HAVING COUNT(*) > 1;
```

This query returns the names of the artists and number of albums they have with over 9 tracks, if they have more than one album with over 9 tracks. Displays the artist name and the number of albums with over 9 tracks.

Nested Aggregation with GROUP BY
appService.js: line 479

```
SELECT professionalName, AVG(numTracks)
FROM Album
```

```
GROUP BY professionalName
HAVING AVG(numTracks) >= (
   SELECT AVG(numTracks)
   FROM Album
);
```

This query returns the names of producers and the average number of tracks per album by each producer, where the producer's average number of tracks per album is more than the average number of tracks across all albums in the database. Displays the producer's name and their average number of tracks per album.

Division
appService.js: line 498

```
SELECT DISTINCT A.stageName
FROM Album A
WHERE NOT EXISTS (
   SELECT S.genre
   FROM Song S
   WHERE NOT EXISTS (
      SELECT *
      FROM Album A2, Song S2
      WHERE A2.stageName = A.stageName AND S2.genre = S.genre AND A2.UPC = S2.UPC
   )
);
```

This query returns all artists that have at least one song of each genre (across all songs in the database) on a single album. Displays the name of the artist.