NTNU

Department of Computer Science

TDT4145 - Data Modeling, Databases and Database Management Systems

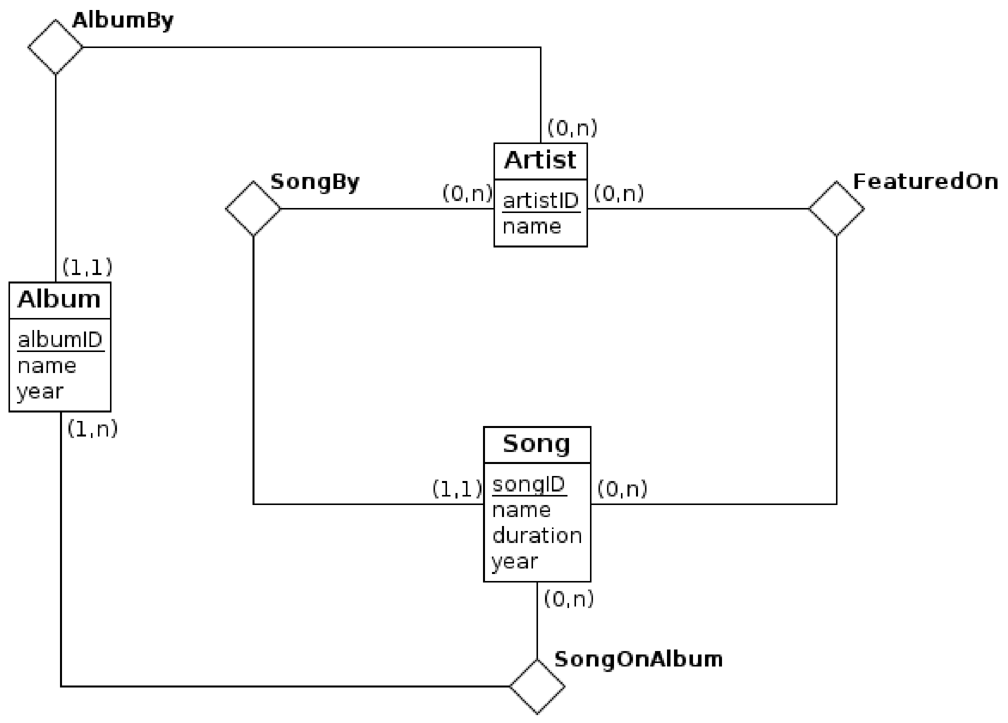
Exercise 3

The learning outcome of this exercise is to:

- Learn to write queries in SQL and relational algebra.
- Be able to evaluate and improve relational database schemas based on principles from normalization.
- Be able to evaluate and find functional dependencies in a table with data.

Task 1: SELECT Queries in SQL

For this task you'll be making queries against a pre-existing SQLite database.
The database in question is a simple music database, consisting of artists, albums and songs, as well as some basic information about said entities, such as the names of all of them, the year of release and the duration of a song (in seconds).
It also includes information about which artists are **featured** on a song, i.e. the artists who contribute but are not the main artist. An example of this would be the song "Baby" by Justin Bieber featuring Ludacris; Bieber is the main artist, whereas Ludacris contributes a guest verse, which makes him featured.
The database is constructed from the following ER diagram:



As evident from the diagram, the database has a few restrictions for the sake of simplicity:

- A song can and must have only one main artist
- An album can and must be by only one artist

As a result, the database contains the following tables (underlined text indicates the table's primary key):

- **artist(**artistID, name)
- **album**(albumID, name, year, artistID)
- **song**(songID, name, duration, year, artistID)
  - artistID is a foreign key referencing artist.artistID
- **featuredOn**(artistID, songID)
  - artistID and songID are foreign keys referencing artist.artistID and song.songID respectively
- **songOnAlbum**(songID, albumID)
  - songID and albumID are foreign keys referencing song.songID and album.albumID respectively

The database is provided in the file **musikk.db**. Open the file in the way you prefer (using the sqlite3 command, DB Browser, etc.) and start querying! Your answers should include both your query and the resulting output.

a) Write a query which returns the songID, title, duration, year and artistID for all songs in the database.
b) Write a query which returns the name and year of all albums released before 2017.
c) Write a query which returns the name and year of all songs released between 2018 and 2020 (inclusive), ordered by year.
d) Write a query which returns the artist name and song name for all artists that have featured on a song (i.e. they are not the main artist), ordered by the artist name and song name.
e) Write a query which returns the song name, album name and song year for all songs by Ariana Grande, ordered by year, album name and song name.
f) Write a query which, for all songs where Ty Dolla Sign is either the main artist or is featured, returns the name of the main artist and the song name, ordered by the artist name and song name.
g) Write a query which returns the artist name and song name for all song names that include the sequence of characters "the".
h) Write a query which returns the artist name and number of features for the artist in the database with the highest number of features (songs with said artist as the main artist do not count).
(HINT: a HAVING clause might be beneficial to use here.)

Task 2: More Queries in Relational Algebra

Using the same database as in task 1, construct queries in relational algebra which…

a) Finds the artist names and song names for all songs in the database that do not feature on an album. The resulting relation should be renamed songsWithoutAlbums(artistName, songName).
b) Finds all artists who have featured on a song from this decade whose artist name starts with B, as well as artists who have released a song in the oughts. The query should return the artist name and the song name.
c) For all artists, returns the artist name and the number of songs by them in the database, ordered descendingly.

Task 3: Introduction to Database Normalization

Consider the following table with data:

filmID	name	year	directorID	directorName	directorBirthYear
1	PlayTime	1967	1	Jacques Táti	1908
2	Mon Oncle	1953	1	Jacques Táti	1908
3	Spring Breakers	2012	2	Harmony Korine	1973
4	Monsieur Hulot’s Holiday	1953	1	Jacques Táti	1908
5	Trafic	1971	1	Jacques Táti	1908
6	The Watermelon Woman	1996	3	Cheryl Dunye	1966

a) The table above is an example of quite terrible database design. Identical information is stored in several tuples, a phenomenon known as data redundancy, which indicates that the database could probably benefit from having a different structure. The above design also leads to problems when inserting, updating or deleting tuples. If we were to delete a record for a film and the database contains no other films by the same director, we would lose all information about said director.
If we discover that Jacques Tati’s birth year is in fact 1907 and that the a in his surname is spelled without an accent, how many cells do we need to update?

b) Can you propose an alternative design which results in fewer cells needing to be updated in the case of correcting Tati’s birth year and name? (Hint: In database normalization, solutions take the form of splitting a table into multiple subtables. In the design above we are storing information about films and directors in the same table.)

You can assume that directorName and directorBirthYear will have the same values for identical values of directorID. In other words, we have a *functional dependency*: directorID -> directorName, directorBirthyYear.

Task 4: Functional Dependencies, Keys and Closures

a) Consider the following table. Which of the following statements must be wrong? Justify your answers.

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_1	d_2
a_2	b_2	c_2	d_2
a_3	b_4	c_3	d_3
a_3	b_3	c_3	d_1
a_4	b_3	c_4	d_2

1) A -> A
2) A -> B
3) A -> C
4) AB -> C
5) C -> D
6) D -> C
7) ABCD is a superkey for the table
8) ABC is a superkey for the table
9) D is a candidate key for the table
10) ABD is a candidate key for the table

b) X+ is called the closure of X, and is the set of all attributes functionally dependent on X, given a set of functional dependencies F.

Given the table R = {A, B, C, D} and F = {D -> A, B-> D, ABD -> C}.
Find D+, BC+, AB+, BD+. How many candidate keys does R have?