Introduction to Database Systems
Homework 4

Björn Thór Jónsson

Fall 2021

Do as many of the following exercises as you have time for. Focus on the exercises you feel will benefit your preparation the most.

Note: Most of the topics of Questions 1 and 2 will be covered in weeks 11 and 12. So, if you struggle with these questions before then, that is normal!

Note: Question 5 is the SQL part of an exam—only 20%, however—from an older version of the course. As you will note, these queries are significantly easier than some of the queries you can expect in the final exam. Please peruse the more recent exams for some more realistic and challenging queries, including division queries.

1 Hardware and DBMS Design

Question 1.A Select the correct statements below:

(a) SSDs are especially well suited to improve performance of retrieving large result sets using unclustered indexes.

(b) Before a transaction can be committed, all the disk pages it has updated must be written to secondary storage to ensure durability.

(c) In a main memory system, it is not necessary to include the "old" values when logging changes made by transactions to secondary storage.

(d) As discussed in a lecture, Google claims to have built a distributed system that offers both consistency and availability in the face of network partitions; this proves that the CAP theorem is wrong.

Question 1.B Reflect upon why ACID transactions are rarely used in distributed systems.

2 Data Systems for Analytics

Question 2.A Select the correct statements below:

(a) Today's key-value stores implement all the functionality required to support *all* needs of analytics applications.

(b) In big data, "velocity" means that it is necessary to react quickly to the large amounts of data being added to the system.

(c) The information that multimedia content (image, video and audio content) is inherently very unstructured in nature.

(d) Big data analytics applications cannot be implemented on any system unless that systems supports ACID transactions.

Question 2.B Consider a scenario where you have 1PB of raw data files that have just been produced as a result of a scientific experiment. You are trying to make scientific discoveries using this data, using complex computations, and also determine the experiments you would like to perform in the future. You have a cluster of machines in your lab (100ish nodes), which have 128GB main-memory and 16 cores each. Explain (with convincing arguments) what type of data management/processing system you would you pick for this scenario.

3 Normalisation

Consider the SQL script, and the associated description, in Section 2 of Homework 3. In Homework 3, you normalised the Rentals relation. Now, you are asked to normalise the Projects relation. For the Projects relation, take the following steps:

1. Find all the FDs in the relations, given the constraints and assumptions from Homework 3.

2. Decompose the relation until each sub-relation is in BCNF, or in 3NF but not BCNF, while preserving all non-redundant FDs. Write down the results schema description in a simple Relation(columns) format.

3. Write the detailed SQL commands to create the resulting tables (with primary keys and foreign keys) and populate them, by extracting the relevant data from the original relations.

4. Select the correct normal form for the decomposed schema.

Please note that the details of these steps, as well as constraints of the relation, are found in the description of Homework 3. Please refer to the description there!

4 Godly strife (by Johan von Tangen Sivertsen)

Zeus is tired of all the infighting among the gods. He has brought you to mount Olympus to help deal with the problem. To avoid conflicts Zeus wants a simple database that tracks all the promises that the gods have made to humans. That way a god can quickly check for conflicts before making a new promise. In case conflicts arise anyway the database should track the various sacrifices made by humans to gods. The time, place and what was sacrificed should all be recorded. Sacrifices are divisible into three categories, flesh(cooked,livestock), wine and valuables(gold,gemstones,etc.). Each category has different attributes that describe them but they all have value. It should be possible to sum up the total value of sacrifices on a pr. human pr. god level to help resolve any conflicting promises.

Further, some humans are appointed priests, a priest is always promised protection by his associated deity and if a sacrifice is conducted in a ceremony presided over by a priest, it is twice as valuable as normally. Any priest serves a single deity. Finally Poseidon has requested that the database also tracks if a human attacks any sea-monster or cyclops and in that case a hecatomb should be deducted from their total sacrifice value for all gods.

1. Draw an ER-diagram that supports the requirements.
2. Write the DDL for a database according to your design.

Remember that Zeus will likely strike you down with lightning if he is not pleased with your work!

5 SQL

In this part you will work with a music database. To start working with the database, import/run `HW4.sql` found in LearnIT using the PostgreSQL DBMS on your laptop. The database has the following relations:

```
Artists(ArtistId, Artist, ArtistImageUrl)
Songs(SongId, Title, ArtistId, Duration, IsExplicit, ImageUrl, ReleaseDate)
Genres(GenreId, Genre)
Albums(AlbumId, Album, AlbumImageUrl, AlbumReleaseDate)
AlbumArtists(AlbumId, ArtistId)
AlbumGenres(AlbumId, GenreId)
AlbumSongs(AlbumId, SongId)
SongGenres(SongId, GenreId)
```

Primary and foreign key attributes have names that end with `Id`. The meaning of other attributes should be self-explanatory. The first four relations have their first attribute as primary key. In the last four relations there is a composite primary key consisting of both

attributes, and each attribute separately is a foreign key reference to one of the first four relations. Secondary indexes exist on `AlbumArtists(ArtistId)`, `AlbumGenres(GenreId)`, `AlbumSongs(SongId)`, `SongGenres(GenreId)`, and `Songs(ArtistId)`.

You will need to work with the `ReleaseDate` and `Duration` attributes, which have the type `date` and `time`, respectively. The expression `interval '1 minute'` can be used to generate a duration value to compare to, the expression `extract(year from ReleaseDate)` can be used to get a year from a date, and `extract(epoch from Duration)` can be used to get the total number of seconds in an interval.[1]

Answer each of the following questions using a single SQL query on the examination database. Enter the result of each query into the quiz on LearnIT. As before, queries should adhere to the detailed guidelines given in Homework 1.

(a) In the database, 372 songs have a duration of at most 1 minute. How many songs have a duration of over 1 hour?

(b) What is the total duration, in seconds, of all songs in the database?

(c) The database contains just 5 songs released in 1953. What is the *largest* number of songs released in a single year?

Note: This is a very simple query. Try answering instead which year had the largest number of songs. Observe how much harder this query is!

(d) The database contains 12 albums by the artist `Queen`. How many albums by the artist `Tom Waits` are in the database?

(e) The database contains 187 different albums with a genre whose name starts with `Ele` (for example, some of these have the genre `Electronica`). How many different albums have a genre whose name starts with `Alt`?

(f) For how many songs does there exist another different song in the database with the same title?

Note: In MySQL, an index on `Songs(Title, SongId)` *was needed for performance, since it only supports nested loops joins. Which join method is used by PostgreSQL o evaluate this query? Does creating the index above change that?*

(g) The average number of `albumId`s per `genreId` in `albumGenres` is 26.5246. An album can have multiple genres. What is the average number of `genreId`s per `albumId`?

(h) An album can have multiple genres. There are 1215 albums in the database that do *not* have the genre `Rock`. How many albums do *not* have the genre `HipHop`?

[1] For more details, see: `https://www.postgresql.org/docs/current/functions-datetime.html`.