29506321: SQL Programming & Creative Writing—Responses

Task 1

Normalization is one of the important processes while designing databases. Normalization is actually a technique for reviewing the database design and it includes a set of mathematical rules.

Answer the following questions using specific examples:

- *Does normalization always lead to a good design? Why or why not?*
- *What kind of issues, problems are possible in the normalization process?*

Normalization usually leads to a good design—but, depending on how often a database is going to be accessed, it may not be necessarily so. Because normalization reduces redundancies—and makes table structures to represent only a single subject—it may lead to increased table joins when a routine seeks data that is spread across multiple tables (Rob & Coronel, 2009). Take the case where you want to store the details of a `user` relation, for instance. It may contain attributes like `id`, `firstName`, `lastName`, `addressLine1`, `addressLine2`, and `zipCode`. If a given `user` happens to have several addresses; maybe one for home, another for work, and another for holiday home; the tables will contain redundant data as the same `user` identifiers are used severally to distinguish the disparate address details. In such a case, normalization would remove the need for the `user` relation having that many columns. It attains that by suggesting the creation of a new relation, `address`, that contains unique address details with only a foreign key to relate it to a given `user`.

On the other hand, excessive normalization leads to its own issues too. When the original `user` relation is broken down into too many other relations, for example, it increases the chances that the database system would take noticeable time to fulfill query demands. So, if the `user` relation is broken down into distinct `name`, `addressLine`, and `zipCode` relations, for instance—the table structure starts to contain too many dependencies based on foreign keys. Moreover, this outcome could make the use of indexing superfluous (Rob & Coronel, 2009).

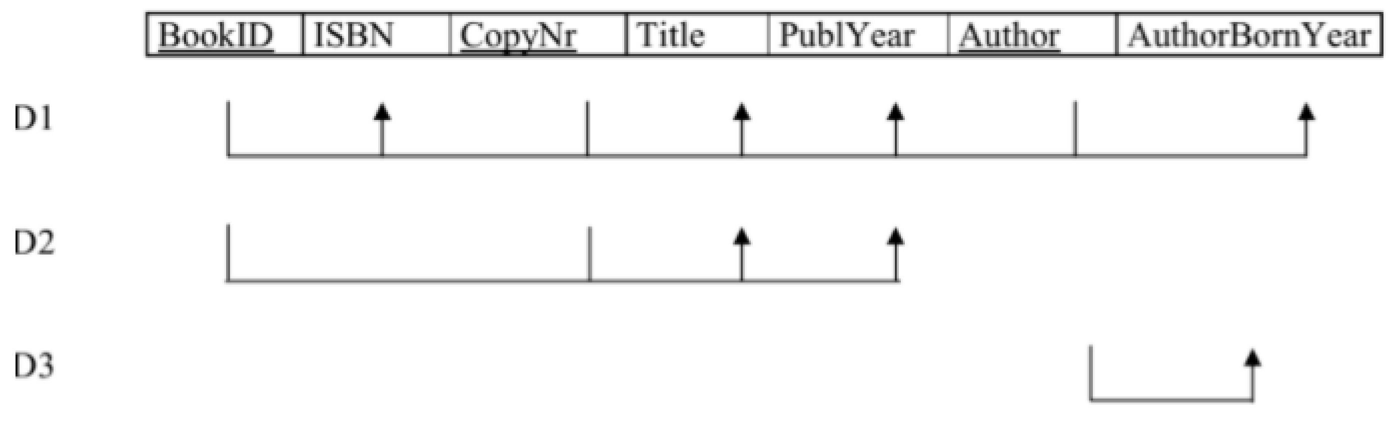
Reference

Rob, P., & Coronel, C. (2009). *Database systems: Design, implementation, and management.* Thomson/Course Technology.

Task 2

Consider a relation named as `BOOK` that contains data about the books in a library. `BOOK` relation was initially created with the attributes `BookID` (an id that library assigns), `ISBN`, `CopyNr` (used to differentiate copies of the same book), `Title`, `PublYear`, `Author`, and `AuthorBornYear`.

- Primary key: `BookID`, `CopyNr` and `Author`
- Candidate key: `ISBN`, `CopyNr` and `Author`
- Dependency 1 (D1): `ISBN`, `Title`, `PublYear`, `AuthorBornYear` were dependent on `BookID`, `CopyNr` and `Author`
- Dependency 2 (D2): `Title` and `PublYear` were dependent on `BookID`, `CopyNr`
- Dependency 3 (D3): `AuthorBornYear` was dependent on `Author`



- *First determine which normal form (1NF, 2NF, 3NF or BCNF) the above relation is, and why. Then, if necessary, convert the above relation to the highest normal form (BCNF). Write any assumptions that you make.*

The specified `BOOK` relation is a 1NF relation. The primary reason for this observation is the fact that each of the other attributes are dependent on the primary key, `BookID`. This means that for a given `BookID`, the other attributes will be unique for the `BookID` attribute (Elmasri & Navathe, 2016).

Still, the `BOOK` relation contains less desirable dependencies. For instance, the `Title`, `PublYear`, and `AuthorBornYear` only offer partial dependency to the composite primary comprising of the `BookID`, `CopyNr`, and `Author`. This means that the `BOOK` relation is a candidate for normalization into the 2NF.

The first step of converting the `BOOK` relation into 2NF involves decomposing the composite primary key into separate relations:

1. `BOOK (BookID, ISBN, CopyNr, Title, PublYear)`; where the primary keys are: `BookID` and `ISBN`
2. `AUTHOR (AuthorID, AuthorBornYear)`; where the new primary key is `AuthorID`

Here, the assumption is that in the two new relations, all the non-prime attributes are dependent on the entire primary key.

Still, there exists a transitive dependency between the `ISBN` and `PublYear` attributes on the `CopyNr` attribute. This means that the latest `BOOK` relation can be normalized further into 3NF as:

1. `BOOK (BookID, Title)`
2. `BOOK_COPY (CopyNr, ISBN, PublYear)`
3. `AUTHOR (AuthorID, AuthorBornYear)`

- *Using Introduction-Body-Conclusion format:*
  - *Describe what you did (This does not mean that you copy and paste from what you have posted or the assignments you have prepared). You need to describe what you did and how you did it,*
  - *what you learned,*
  - *your weekly activities,*
  - *in what ways are you able to apply the ideas and concepts gained, and finally,*
  - *describe one important thing that you are thinking about in relation to the activity.*

The `BOOK` relation contained many nested functional dependencies are transitive dependencies. The task was to, therefore, create new relations such that the only dependencies that remained were those that pointed to the primary key only. Hence, the normalization steps required the identification of candidate primary keys first. Then, it necessitated the creation of new relations that contained attributes which were all dependent on the assigned primary key. Finally, the task demanded that the relations were further extracted into new relations that confined any transitive dependencies in extra separate relations.

The entire process taught me how to assess functional dependencies with a keen eye. Whereas the approach seemed cumbersome and even tiresome at first, I imagined what would happen if unnecessary relations were inserted into a given table. Then, if later that table was the target of huge data updates, the design would lead to previously unseen redundancies that would lead to inefficiencies in read times. As a result, I attempted to generate dummy data for the relations that I created out of this exercise to see whether the normalization I undertook was effective as I had forecast. True enough, I realized that some attributes that had previously seemed unrelated started to cause repeating sets of data. The `ISBN` and `CopyNr` attributes, for instance, proved tricky when I tried to determine their relation. Ultimately, after examining how the dummy data presented itself, I concluded that the `ISBN` attribute was actually the one that was dependent on the `CopyNr` of a `BOOK` and not the other way round.

In the end, the important thing that I thought about while carrying on the task was whether what I was doing could have an impact on large sets of data. That thought nagged me me to the extent that I decided to try it out with some actual data sets.

Reference

Elmasri, R., & Navathe, S. (2016). *Fundamentals of Database Systems*. Pearson.