Unit 7

Database Design (Reflection)

Normalisation

This module addressed the three stages of normalising data model structures, named as first normal form (1NF), second normal form (2NF) and third normal form (3NF). Each step of normalisation is aimed at **reducing dependencies** such that a single set of attributes (row of one or more columns in relational databases) is **uniquely** identifiable. Normalisation is considered a bottom-up approach whose aim is to reduce redundancy and data inconsistencies (

My experience over many years causes me to normalise data whenever I notice *groups of related information* are bunched together into a single table; I reach for my normalisation hat and begin splitting data sets into logically relatable groups. For this reason, I classify 1NF and 2NF as the simplest, since they are concerned with reducing duplicate values; while 3NF is concerned with further reducing left over collection of attributes into respective tables; I shall say something akin to reducing to attributes (1NF, 2NF) and reduce to objects (3NF).

Third Normal Form

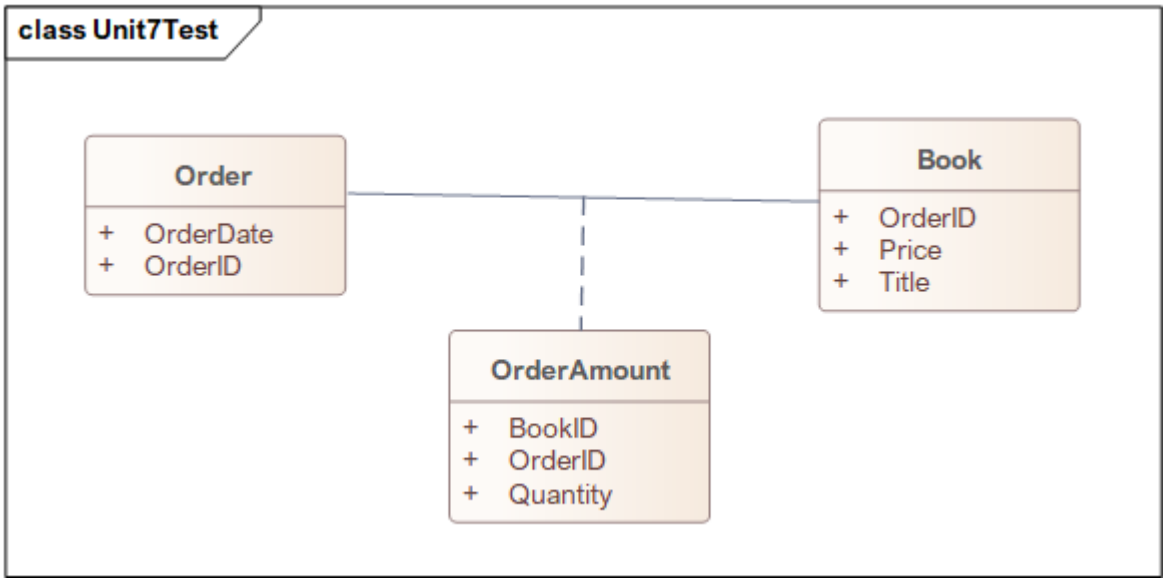
As I read through the material provided at Wilkinson (2020), I found rather interesting the relation between third normal form (removing of partial dependencies) and **UML association classes** in scenarios where there exists many-to-many relationships in the data set being normalised. For example,

OrderID	OrderDate	CustomerID	LastName	Address	BookID	Title	Price	Quantity

Here, normalising this table according to Wilinson (2020), results in this particular table

OrderID	BookID	Quantity

Which, to me is similar in modelling concept to the following UML class diagram where OrderAmount represents a UML association class:



Though, in a real-world system, I would not duplicate the BookID and OrderID attributes on the Order and Book classes, since the association class already shows them.

Interestingly, I agree wholeheartedly with Wilkinson (2020) that composite keys are seldomly used. I find this to be a true statement in reality because developers and DB designers often substitute a row's identity to a *single* attribute that is (mostly) an auto-incrementing integer value or a GUID value. I think this is driven by the fact that composite keys are a burden for software developers because *more* information is required to lookup a given row of attributes (much like the UML association class, OrderAmount, shown above).

Boyce Codd Normal Form, Fourth Normal Form & Fifth Normal Form

These normalisation forms were raised, and I am aware they exist. However, in practice of my career, I have never encountered software development teams that go beyond 3NF when normalising their data models.

Normalisation in Software vs Data

Data normalisation is a systematic approach to reducing dependencies, and ties in conceptually with the concept of “refactoring” in software development. In particular, the various software development patterns aimed at delivering better, quality information systems. One set of pattern I consider particularly relevant for comparison are the SOLID principles (Single responsibility; Open Closed principle; Liskov Substitution; Interface Segregation; Dependency Inversion) that are the cornerstone of any good software developer’s toolkit. I home in on the **Single Responsibility** (SR) principle for software development. SR attempts to ensure that each function, class and interface all do one collective thing and do it well. From this principle comes the concept of **Cohesion** which is the degree to which elements inside a **module belong together** (Thaler, ND)**.** These two software concepts match up perfectly with data normalisation because the end-goal of normalisation is to build data sets that are highly “cohesive”, whose collection of attributes present a “single responsibility”.

Normalisation and NoSQL

Data normalisation is a *must-have* when persisting data within relational models. Yet, when persisting data in NoSQL models, I find the requirement for normalisation is not as high given that NoSQL databases persist data as either documents (JSON or XML), key-value pairs or as a graph of data. However, I will still advocate the concept of normalisation in data simply because it forces me to consider data as a single set of information, related to other sets of information.

References

Kumar, K. and Azad, S.K. (2017). Database normalization design pattern. *4th IEEE Uttar Pradesh Section international Conference on Electrical, Computer and Electronics (UPCON)*: 318-322. IEEE.

Thaler, J. (ND), Software Engineering Cohesion, Coupling and SOLID Principles.

Bibliography

Kennedy, D. (2000). Database Design and the Reality of Normalisation. In 13th Annual NACCQ Conference, http://www. in-site. co. nz/misc_links/papers/kennedy167. pdf (Google, 2012.05. 05)

Wilkinson, V (2020) Design the Logical Model of Your Relational Database. Available from https://openclassrooms.com/en/courses/5671741-design-the-logical-model-of-your-relational-database/6260611-normalize-to-the-first-second-and-third-normal-forms [Accessed on 5 Jul. 2021]