# **Lesson 7: Discussion**

4 4 unread replies. 4 4 replies.

**Purpose:**

Review the importance of normalization and denormalization in database design and in the overall software development process.

**Tasks:**

1. How can we apply in practice concepts in the normalization and denormalization process? Is normalization important or is it just an academic concept? Please give one pro and con and support your position with concrete examples.
2. Please, respond to the initial post and respond to at least two of your classmates. The initial discussion question (DQ) response message is due by 11:59 PM on Thursday, and at least two (2) responses to other peers are due by 11:59 PM EST on Sunday.

Based on the information introduced into the lesson on normalization and denormalization in a database, they are both important in practical applications, not just for academic purposes. The process of applying normalization and denormalization forces the database designer to consider every detail of the database design. You (the designer) have to balance perfect data integrity with performance. This is where it is important to consider the pros and cons of normalization and denormalization.

A pro to normalization is that, as mentioned above, it maintains high data integrity. This means there is minimal data redundancy which leads to less chances for there to be inaccuracies. A con for normalization is that you increase the complexity of your database. You will have to introduce new tables to maintain the normalized state of the design. This can make database searches much less performant.

A pro to denormalization is opposite to the con of normalization. You can centralize your data into fewer tables. This means that querying the database is much more efficient. Of course, the pro also comes with the con of data redundancy. The pros and cons of normalization and denormalization are inversely proportional.

As an example, let’s look at a database that tracks customers and their orders at a restaurant. In a normalized database, you would need tables for Customers, Orders, Products, and Order Details. Let’s say that Order contains attributes order\_id, customer\_id, and total\_amount, while Order Details contains attributes, order\_detail\_id, order\_id, product\_id, and quantity. If you wanted to run a query to get all of the data for an order you would have to join the Order and Order Details tables and joins are generally an expensive operation on databases. It is also easy to maintain this data. If you denormalize the database, you can merge the Order and Order Details tables to be order\_id, customer\_id, total\_amount, product\_id, and quantity. This would result in a more performant query, but it becomes more likely that there are data inaccuracies and it is harder to maintain the data. Of course, these are small tables, but the concept can be applied to much larger datasets.

References:

* The modules
* https://www.geeksforgeeks.org/advantages-and-disadvantages-of-normalization/