Overruling Normalization

Overruling Normalization

As much as ensuring that a database is in 3NF will help guarantee reliability and viability, you won't fully normalize every database with which you work. Before undermining the proper methods, though, understand that doing so may have devastating long-term consequences.

The two primary reasons to overrule normalization are convenience and performance. Fewer tables are easier to manipulate and comprehend than more tables. Further, because of their more intricate nature, normalized databases will most likely be slower for updating, retrieving data from, and modifying. Normalization, in short, is a trade-off between data integrity/scalability and simplicity/speed. On the other hand, there are ways to improve your database's performance but few to remedy corrupted data that can result from poor design.

The previous page included an example where normalization is ignored: a message's post date and time is stored in one field. As mentioned, because MySQL is so good with dates, there are no dangers to this approach. Another situation where you would overrule normalization is a table that stored a person's preference for a certain setting, such as "receive notifications." If stored as just Y/N or Yes/No (instead of linking to an answers table), there would be many repeating values. But that is fine in this case, since those labels are stable values, not likely to change over time (i.e., it's unlikely that a third option will be invented, or that "Yes" will be renamed, forcing a mass update of half the records in the table).

Practice and experience will teach you how best to model your database, but do try to err on the side of abiding by the normal forms, particularly as you are still mastering the concept.