Denormalization is an implementable strategy to improve the read performance of a database. In database programming, the four basic operations are create, read, update, and destroy. Denormalization sacrifices write performance (create, update) in order to improve reading performance. Denormalization does this via storing redundant data in tables. This sacrifice of data storage space allows for increased read speeds by avoiding unnecessary join operations and can improve performance and scalability depending on how it’s implemented. Join operations can become increasingly slow based on the amount of relations being joined. In order to avoid this increase in time-complexity, denormalized databases will store the same information in multiple tables in order to bypass these join operations. Something that’s important to note is the fact that denormalization is not the inverse of normalization. Denormalization is a strategy that is directly applicable to normalized databases. The implementation of denormalization falls under two separate categories, DBMS support and DBA implementation. DBMS support is an integration of denormalization that allows the database management system itself to manage redundant information and improve query response times. In this scenario, the DBMS is responsible for making sure that redundant copies of data are kept consistent throughout the database. The other type, DBA implementation, requires the programmer to denormalize the data design themeself. In this scenario, it is the responsibility of the programmer to ensure redundant data does not become inconsistent. One way of managing this is via constraints. Constraints allow us to specify how redundant data should be synchronized across the database.