Hash algorithms are data structures that allow for a quick retrieval of data, regardless of the data size. They are more efficient than sequential search methods, such as brute force. They work based on the value of the key stored in the array/primary storage index, and a calculation that is used to place it in a particular location. This is the fundamental reason hashing functions are so fast and appeal to most developer’s designs. Depending on how the hashing function is used, there are possible drawbacks though. The most common example is when the mapping of two keys to the same storage position occurs, otherwise known as a ‘collision.’ This issue can be addressed through using collision algorithms, that redirect the conflicting placement to another new available location. Having to correct a collision translates to a longer processing time, not a desired effect. Smaller data sets that address unique situations can implement a perfect hashed data structure, but overall that same structure design is less likely to be able to be applied perfectly to another situation. Two more aspects of hashing structures are that they can use the main operations, insert, fetch, delete, and update. They also should implement preprocessing, to minimize collisions, convert non-numeric references into numeric, and implement pseudo keys to address any text or non-negative variables that exist.