Handling Large Data

Instead of saving word to memory addWord function will write words to database. Database will consist of one table having two columns: sorted_chars (String) and anagrams (Set). "sorted_chars" column should be indexed in order to improve lookup time in database. "sorted_chars" column contains words with their characters sorted. Words with their characters sorted will be passed to the database to check if the sorted word exists in sorted_chars column. It will then return corresponding set of anagrams. The new word will be added to the returned set of anagrams. If no value is returned, then a new hashSet will be created and written back to database. Once the entire input file has been processed and database is populated with sorted_chars and anagrams. We then efficiently retrieve data and output it in console. The problem could be that it can get very slow if we try to read the entire database and print it out. To prevent huge amount of loading time, in the final SELECT query, use LIMIT and OFFSET to select up to 1000 rows at a time and display them.

Through this approach, we will be able to store all keys and anagrams in our database and print them out efficiently. We shouldn't be concerned about denormalization of database since it involves only single table. We can also consider using NoSQL database since we have only single table which can be converted into key value pairs in NoSQL database. Relational database is not required. NoSQL database also gives us flexibility to scale. If number of read or write requests increases, then we can add more servers to cater those requests.