Data Modeling - Project 2, Stage 1 - Spring 2014

Sawyer Jager, Rees Klintworth, Derek Nordgren, Brad Steiner

Data Representation

We chose to represent the data as a set of key-value stores within a text file, where each line in the text file represents a basket containing multiple items. Each line in the file lists the IDs for the items in the basket, separated by spaces. The value, the list of items IDs, can be accessed by simply reading in the correct line of the input text file. This text file was created by simply reading in each transaction and creating a basket (line of text) for that transaction. Since the code provided to us to access the transaction files was written in C, we used C to create the input file, and passed that file to our Java-implemented solver.

If we were to expand the project to search for other criteria, similar to the first project, we would implement the following representational data structures:

Location	**Customer**	**Basket**	**Item**	**ItemPurchase**
stateName	customerID	basketID	itemID	basketID
locationID	locationID	customerID		itemID
		dayOfWeek		customerReview

Each Customer has a related ID and location. Each Basket, which represents multiple items, has a purchase date and the ID of the purchasing customer. The ItemPurchase table will be used to keep track of the relationship between Baskets and the Items they contain, as well as the respective customer reviews for each Item.

For this stage of the project, we decided to represent the data in the simplest possible form so as to minimize the complexity of implementation. This simpler form also keeps the running time to a minimum.

Processing Time

Our input file (generated in C) takes 0.6 seconds to generate.
Processing to find triple sets (done in Java) averaged 18.4 seconds.

Complexity Analysis

The complexity was determined to be $O(T*I^3)$, where ‘I’ is the total number of items, and ‘T’ is the number of transactions. In the average case, the complexity will be much lower due to the Apriori algorithm filtering itemsets with less than three occurrences. Additionally, ‘I^3’ represents the number of combinations of ‘I’ elements, depending on the iteration of the algorithm. If our algorithm does three passes, to find the minimum support of three, our algorithm will need to

make (I 3) [I choose 3] combinations in the worst case. In our given assignment, the I^3 portion was represented by 1000 combinations of 1, 49229 combinations of 2, and 171 combinations of 3.