

2018


# Synthetic Order Data Generator for Picking Data

Mohammadnaser Ansari  
*Auburn University*, ansarim@auburn.edu

Behnam Rasoolian  
*Auburn University*, bzx0014@auburn.edu

Jeffrey S. Smith  
*Auburn University*, jsmith@auburn.edu

Follow this and additional works at: [https://digitalcommons.georgiasouthern.edu/pmhr\\_2018](https://digitalcommons.georgiasouthern.edu/pmhr_2018)

 Part of the [Industrial Engineering Commons](#), [Operational Research Commons](#), and the [Operations and Supply Chain Management Commons](#)

---

## Recommended Citation

Ansari, Mohammadnaser; Rasoolian, Behnam; and Smith, Jeffrey S., "Synthetic Order Data Generator for Picking Data" (2018). *15th IMHRC Proceedings (Savannah, Georgia. USA – 2018)*. 15.  
[https://digitalcommons.georgiasouthern.edu/pmhr\\_2018/15](https://digitalcommons.georgiasouthern.edu/pmhr_2018/15)

This research paper is brought to you for free and open access by the Progress in Material Handling Research at Digital Commons@Georgia Southern. It has been accepted for inclusion in 15th IMHRC Proceedings (Savannah, Georgia. USA – 2018) by an authorized administrator of Digital Commons@Georgia Southern. For more information, please contact [digitalcommons@georgiasouthern.edu](mailto:digitalcommons@georgiasouthern.edu).

# Synthetic Order Data Generator for Picking Data

Mohammadnaser Ansari\*, Behnam Rasoolian†, and Jeffrey S. Smith‡  
Industrial & Systems Engineering Department, Auburn University  
Auburn, AL 36849

Email: \*ansarim@auburn.edu, †bzs0014@auburn.edu, ‡jsmith@auburn.edu

**Abstract**—Sample data are in high demand for testing and benchmarking purposes. Like many other fields, warehousing and specifically order picking process are not exempt from the need for sample data. Sample data are used in order picking processes as a way of testing new methodologies such as new routing and new storage allocation approaches. Unfortunately, access to real order picking data is limited because of confidentiality and privacy issues which make it difficult to obtain practical results from the new methodologies. On the other hand, order data follows a highly complex and correlated structure that cannot be easily extracted and replicated. We propose a two-part synthetic data generator that extracts and mimics the general fabric of a set of real data and produces a conceptually unlimited number of orders with any number of SKUs while keeping the structure largely intact. Such data can fill the gap of missing data in order picking process benchmarking.

**Index Terms**—Data Generator, Association Rule, Correlation

## I. INTRODUCTION

We are in the world of data overload. Every purchase, every shopping decision, and every shipping process is recorded and kept on data servers all over the world. However, with much of this data, highly restrictive privacy, information security, and proprietary data regulations are in place. Our focus is on order picking data and, considering these regulations, even with the huge amount of order data being recorded and stored, they are not easily accessible by researchers outside of the firms in which they were generated. On the other hand, the best approach to test and validate any warehousing theory or order picking method is through empirical testing with real data. The best way to test new methods in warehouse operations is to implement it in real life scenarios. However, these types of trial and error and field testing are highly expensive and time-consuming. That is when simulation modeling and analysis come into play. Researchers simulate a warehouse in which they can test their new suggested approaches and compare it to benchmarks and other methods which are already in use. But like any other scientific experiment, an evaluation of performance needs to be tested in different scenarios and the data used to replicate real life environment should be real data or at least a close replica of it. The data should have the following characteristics:

- *Looks like real data*: The order data should either be real customer purchase history or a close replica (*synthetically generated data*) that maintains the essential features and structure of the real data. By using synthetic data, we can be sure of the similarity of simulated environment to real

life scenarios but we do not need to worry about privacy, security, or confidentiality.

- *Expandable*: Different simulation models are designed for different situations and warehouses. One simulation might have the goal of modeling a warehouse with 1000 SKUs while another is trying to model an enormous 1,000,000 SKU warehouse. The data in use should be expandable to be able to incorporate all possible modeling scenarios.
- *Modifiable*: Researchers should be able to modify the data to their need. For example if a researcher's goal is to analyze the performance of a warehouse in SKU overload (constriction), s/he should be able to manipulate the data to represent  $X$  percent more (or fewer) SKUs.
- *Standard*: The data in use should be standard among all its replications and should be reproducible. This helps to have a fair comparison ground when testing a new method in warehouse operations modeling.

By developing a methodology (and corresponding computer code) to *create* picking order datasets that meet these requirements, we can solve the big problem of test data availability. Such data can be specifically helpful to researchers in warehouse order picking processes to assess and benchmark new routing, location assignment problem (LAP), and other warehouse operations-related methods. Once developed, the methodologies and associated code will be freely available and disseminated as part of WODS [1].

In this paper, we suggest a Synthetic Data Generation approach to replicate the general and correlation structure a real order dataset. In our method, by abstracting the real data into a higher level we will be able to mimic the real dataset without presenting any real data. This method consists of two separate steps. In the first step we try to create arbitrary number of orders based on a limited number of real data orders (Section III-A). In the second step, we try to create any number of SKUs based on a set of real order data with limited number of SKUs (Section III-B).

## II. LITERATURE REVIEW

With more and more software development, the need for reliable data to test and benchmark them arises. Of course, real data is the best testing option, but privacy and unavailability of real data limit the access to them which forces the researchers to use synthetic data and synthetic data generators. Synthetic data are generated to give access to data that cannot

be obtained by direct measurements [2]. There is no standard Synthetic Data Generation method since the generators purpose-built the data and different fields of research use various methods which are either unique to that field or at least not applicable to all areas.

For example, consider SIR model, a simulation model vastly in use in the area of epidemiology which analyzes the spread of a disease via direct contact of people [3]. In these models, the population data and relations among the subjects are created by synthetic data generators. CROWD can be named as one of the tools that can be used to generate population samples. CROWD creates population model based on census data and places them on a model city which itself is based on the urban planning data to represent the relations [4]. Based on the characteristics that the population needs to have, probabilistic data generation based on the census data related to that such as age, height, income, education, etc. will be used.

In the field of application development, the final applications should be tested with a set of data that can be a representative of real data [5]. An example can be putting a web application to stress and load test by simulating multiple user access at the same time. Generating virtual visits to the website mimicking the steps that an actual user might take when visiting the website is a reasonable method used for web application and website testing.

In the field of database and server development, enormous amounts of data are required to put the system in a realistic load test. Tests include creating complex queries to load and stress test the databases to check its responsiveness and limitations. Running multiple queries and inserting thousands of entries that have been generated for testing purposes into tables are a few testing approaches that the development team takes. [6], [7] developed data generator specifically designed for this purpose.

Data mining tools are also needed to be tested before implementation to be sure of their performance. Testing a data mining software limits by assigning an enormous sample data to it to be mined can be named as one of data generators uses in this field. Feeding large text files to word mining software or health records to be mined to find the relation among diseases need sources of data that are not always easy to find. [8]–[10] developed data generators to have access to large sums of data to be able to put various data mining tools to stress and load test as well as performance evaluation by comparing the results of mining on their manually generated data to the expectations.

As it was mentioned, the need for data generators to test and benchmarking is not limited to one field. Supply chain and warehousing are not excluded from this group. Datasets are required to test the methods and approaches in solving warehouse operations related problems. Although there is a vast number of data generator out there, since they are created for other purposes than order data generators, they cannot be used in this field. SIR model population generators are specifically designed to create objects (peoples for example) and how they are connected in the network or in other words,

a graph by using a probabilistic approach from the sample data or metadata. The data generators for database testing mimic table relations mostly by random sampling from a probability function that has been generated based on some real sample data or its metadata. Based on the literature that has been reviewed, data generations are custom built for the purpose that they are supposed to be used in and there is no clear method or approach in neither how they are developed and how they are tested. The ones that do mention the approach they took in generating the data are using simple probabilistic approach based on sampling from a probability distribution which itself is extracted from a sample of real data.

There are some generators that can be considered as an option for generating order data. One of which is the highly versatile synthetic data generator software, KNIME®. KNIME® is an open source data analytics, reporting, and integration platform that can also generate sample data [11]. However, the data generated by KNIME® are not mimicking all aspects of the real data and are created only based on the probability of the products appearing in the real data sample. The dataset that is needed in the field of warehousing, should represent order data that comes to the warehouse and needs to be fulfilled. However, in order to be able to generate such data, we first need to understand the structure of real data.

Order data represent a set information related to each order that usually includes but is not limited to customer information, date, etc. Each order includes a set of items or SKUs that are asked by the customer. In Figure 1 an ER diagram representation of order data is illustrated [1].

The items asked by the customers can be correlated and that is where market basket analysis comes in. Market basket analysis scrutinizes the products customers tend to buy together [12]. Apriori and DHP are the most common techniques in market basket analysis or association rule mining. In Apriori method, the higher level rules are filtered out if not every member of it is significant based on lower level analysis [13]. DHP follows a similar approach to Apriori, however the employment of hash-tables in this method significantly reduces the database size [14].

Knowing such information about the order data can be highly beneficial in cross selling the products and promoting correlated products together [12]. For example consider sugar, tea, and coffee in a grocery store. Knowing that sugar and coffee are positively correlated tells us that if we decrease the price of coffee, not only the sale of the coffee will increase, but also of the sugar. On the other hand, reducing the price of coffee will decrease the sale of tea which is negatively correlated with coffee [12]. Mining for such patterns, correlations, and associations among products in order data is the goal of *association rule mining*. Considering the enormous size of the real data that sometimes can be obtained by researchers, the extraction of these rules should be highly efficient. [15] introduced new approach in mining for association rules while [16] defined new method in testing the significance of rules to facilitate the process of rule extraction. [17] introduced the use of market basket analysis in cross-sales potential and

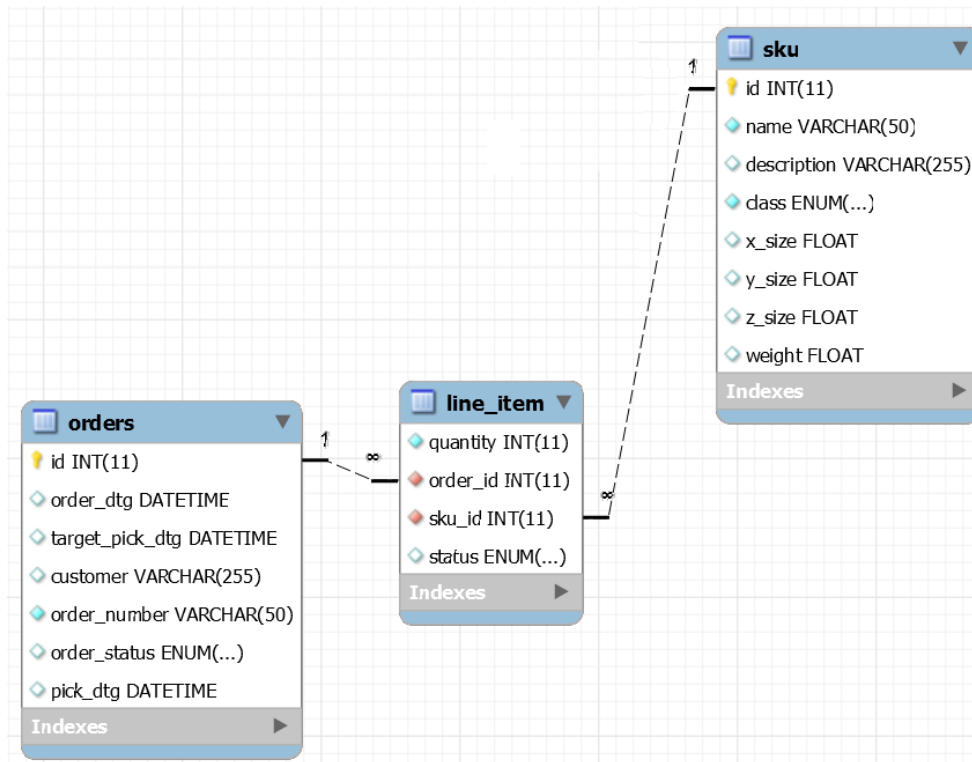


Fig. 1. An ER diagram of order data [1]

product selection decisions. Analyzing the market basket in multiple store scenarios is something that goes above the normal association rule mining approaches which has been done by [18].

Keeping these facts in mind, generating synthetic data for warehouse operations should also incorporate the correlation of products in mind. However, our research indicates that there is no data generator in the field of warehouse data and modeling that can be an acceptable replication for the real data.

### III. METHODS

A real set of order data has some unique characteristics that need to be replicated by the order generator to create “similar” data also fits into the WODS framework [1]:

- 1) The *Orders*, *SKUs*, and *Items* need to follow the entity-relationship diagram illustrated in Figure 1;
- 2) The probability of appearance for any SKU should follow the corresponding probability from real dataset; and
- 3) The correlation structure between the SKUs should also match the real data (including multi-SKU and “SKU Family” correlations as described below).

Generating a dataset while keeping the first and second points valid is not hard, and it can be done using a few lines of computer code. However, identifying the correlation structure between the SKUs and replicating this structure while

generating a set of synthetic data is not nearly as easy and is the main goal of this research.

The datasets of interest that are needed for our simulation model of warehouse operations include the list of orders and the items that comprise each order. Each order consists of multiple items which represent the SKUs that the customer asked for and their respective quantities (see Figure 2). Now when it comes to generating a replica of the real data, there are two aspects that need to be addressed:

| Order Number | Product (SKU) | Quantity |
|--------------|---------------|----------|
| 10150        | softdrink     | 1        |
| 10150        | fruitveg      | 5        |
| 10236        | frozenmeal    | 4        |
| 10236        | beer          | 3        |
|              | ...           |          |
| 10360        | fish          | 3        |
| 10360        | cannedveg     | 4        |
| 10360        | beer          | 1        |
| 10360        | frozenmeal    | 1        |
| 10451        | confectionery | 2        |
| 10451        | frozenmeal    | 6        |
| 10451        | beer          | 12       |
| 10451        | cannedveg     | 2        |

Fig. 2. A snippet of sample order data

- The generation method should be able to create an arbitrary number of orders. By having this characteristic, the resulting datasets can be the correct size to simulate

the warehouse operations of over an arbitrary length time window; and

- The generation method should be able to create orders based on an arbitrary number of SKUs. Considering that warehouse models can represent various warehouse sizes and designs, each model potentially needs a different number of SKUs to represent the generated order datasets.

Having both characteristics in the data generator gives the flexibility of data manipulation to the researchers to create the data and update it based on their need.

For solving each problem, a different approach has been used. Section III-A will describe the method used to create an arbitrary number of orders and Section III-B will illustrate the method used to generate data based on any number of SKUs. By incorporating both techniques in creating the dataset, we will achieve a *Synthetic Dataset Generator* that can generate any number of orders based on any number of SKUs that can be big enough to overcome any experimentation need.

#### A. Creating arbitrary orders

In this part of the paper, we intend to generate synthetic data of arbitrary length based on a real data of limited size. This can be done by using a concept called *association rules*. An association rule is a rule that specifies the probability of co-occurrence of items in a collection. The process of mining for such rules using an existing dataset is called *Association Rule Mining* or *Market Basket Analysis*. To explain the association rule mining process, a short explanation of related terms needs to be provided:

- *Item*: Each line of order and in our case, each asked SKU of an order;
- *Item-set*: Is a set of items or SKUs;
- *Rule Support*: The relative frequency of orders that contain specific SKU  $X$  and SKU  $Y$ .  $Support(X \rightarrow Y) = \frac{Support(X + Y)}{Support(X + Y)}$  the probability of SKU  $X$  and  $Y$  appearing in the same order where  $X$  and  $Y$  are two distinct SKUs;
- *Rule Confidence*: The conditional probability of SKU  $Y$  appearing in an order if SKU  $X$  is already in it.  $Confidence(X \rightarrow Y) = \frac{Support(X+Y)}{Support(X)}$  where  $X$  and  $Y$  are two distinct SKUs.

Each rule defines the correlation between two or more products with a calculated support and confidence. It is worth mentioning that the rules are not limited to two-way correlations. Three-way correlation such as  $X, Y$  and  $Z$  are also considered and are calculated as follows:

$Support(X, Y \rightarrow Z) = \frac{Support(X, Y + Z)}{Support(X, Y)}$  the probability of SKU  $X, Y$ , and  $Z$  appearing in the same order where  $X, Y$ , and  $Z$  are three distinct SKUs.

$Confidence(X, Y \rightarrow Z) = \frac{Support(X, Y + Z)}{Support(X, Y)}$  where  $X, Y$ , and  $Z$  are three distinct SKUs.

The same concept can also be applied to all possible combination of SKUs.

Considering the fact that the number of SKUs in the real data can be large, calculating and keeping a record of every

possible correlation is often impractical. Furthermore, most correlations are of no value to the synthetic data generation goal (i.e., value 0 or close to 0). That is why association rule extraction algorithms first try to extract item-sets with significant supports and then analyze each subset of these significant item-sets to pick rules with significant confidences. The significance factor is defined by user and depending on the scope of the research and computational power can be adjusted. For example consider a set of order consisting of products  $X, Y$ , and  $Z$ . If the support of the  $X-Y$  duo does not pass the significance factor, it is obvious that  $X-Y-Z$  will not be significant.

These association rules can be derived from real data or can be manually defined. Manual definition allows us to impose specific correlation structures to augment those from the real data. Predictive Model Markup Language (PMML), A standard XML vocabulary developed by the *Data Mining Group* to report results of a data mining analysis [19], [20] is used to store the association rules.

Our algorithm receives a PMML file as an input. This file can be the result of a mining process on real data, a set of manually generated rules, or a combination of both. Our purpose is to generate synthetic data of very large scale from a smaller real-world data or in case real data is not present at all, generate it by using a set of already known rules.

Then the new data needs to be generated in a way that the relative relationships between items is maintained. Because some of these item-sets are highly correlated to each other, adjusting one item-set regarding its support will change supports of other item-sets. Our algorithm provides an efficient random insertion of these item-sets so that their significance is ensured and also their support value is kept within an acceptable range of their original support. Figure 3 gives a flowchart of the processes in order to generate arbitrary number of orders.

We tried our algorithm on a real dataset in a quick test. The real dataset has 998 orders consisting of 11 different product types with arbitrary size of lines per order. We extracted the association rules from the real dataset (Figure 4 shows the extracted association rules based on minimum support of 0.1) and managed to create a synthetic data of size 100,000 (Figure 5) while keeping all the significant association rules (and certainly not introducing new ones). In the synthetic data generated via the algorithm, the support of each item-set is also within an acceptable range of its original ones. Figure 6 illustrates the association rules extracted from the generated dataset which can be compared to Figure 4. Later on, a more thorough testing of the method is required to evaluate the performance of the generated dataset under different scenarios and compare the results with the real data.

#### B. Creating any number of SKUs

In this section of the paper, the goal is to generate synthetic data by using a set of association rules, but instead of keeping the number of SKUs the same as the ones in the set of association rules, we want to be able to add SKUs to the list of SKUs that we have while keeping the rules not the

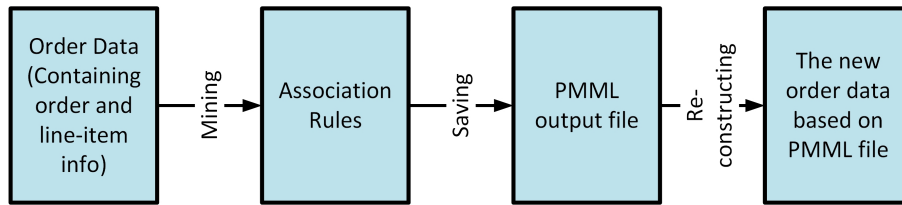


Fig. 3. Flowchart of Phase I in developing arbitrary number of orders

| Original Rules                      |
|-------------------------------------|
| [1];(0.19509594882729211 183)       |
| [2];(0.31769722814498935 298)       |
| [3];(0.31982942430703626 300)       |
| [4];(0.31236673773987206 293)       |
| [5];(0.31023454157782515 291)       |
| [6];(0.3230277185501066 303)        |
| [7];(0.2931769722814499 275)        |
| [8];(0.19402985074626866 182)       |
| [9];(0.3049040511727079 286)        |
| [10];(0.21641791044776118 203)      |
| [11];(0.1886993603411514 177)       |
| [7, 9];(0.1535181236673774 144)     |
| [6, 9];(0.10341151385927505 97)     |
| [4, 6];(0.17803837953091683 167)    |
| [3, 6];(0.18443496801705758 173)    |
| [2, 5];(0.1535181236673774 144)     |
| [3, 4];(0.1812366737739872 170)     |
| [3, 4, 6];(0.15565031982942432 146) |

Fig. 4. Sample association rules extracted from the real dataset

| Generated Order Data |
|----------------------|
| 2 11                 |
| 3 7 9                |
| 2 5 7 10             |
| 8                    |
| 3                    |
| 2 5 11               |
| 3                    |
| 2 5 9 11             |
| 5 7 11               |
| 4                    |
| 2 5 8                |
| 2 4                  |
| 1 5 8 10             |
| 1 2 6 7              |
| 1 3 4 6 7 9          |

Fig. 5. Sample generated order data represented by product ID (SKU)

same (since we are generating new SKUs, new rules will be added) but structurally similar. In Section III-A, a set of association rules is fed to the function (the association rules can be manually entered or being extracted from a sample set of orders). Then the function would be able to generate a conceptually infinite number of orders while keeping the association rules intact (or least make sure that they fluctuate within reasonable boundaries). Although doing so will give us a possibly infinite number of orders, the number of SKUs appearing in the orders follow the same pattern as the real dataset from which the association rules are extracted. This means that if we start with a sample of orders consisting of  $X$  number of SKUs, the output of Phase I will still consists of  $X$  number of SKUs. In this section, we propose a new method by which we take a step further and generate new rules while keeping the general relation of the SKUs the same.

For achieving this purpose, we decided to abstract the rules to a level that while keeps most of its properties, can be replicated. In doing so, we need to define *families*. Families are set of items that while being close in behavior, are not typically ordered together — two brands of the same product would be an example. Different brands in the same product type can practically be interchanged while the chance of them

appear together in the same order is slim (considering they both satisfy the same need in the customer basket). Consider Figure 7 in which level 4 represents the unique SKUs that are being mined for association rules. Level 3 and Level 2 represent the families that the unique SKUs belong to. Now if the rules are mined for the families, or in other words based on level 3 and level 2, the rules are an illustration of correlation among families.

For example, we know that people usually buy milk and cereal together, the milk can a store or any of a number of other brands. While they both follow the same behavior in orders, there is a negligible probability that they are being ordered together. We decided to use this fact as a basis for abstracting the rules. While the two types of milk belong to the same family, they are two individual SKUs ( $Milk_1$  and  $Milk_2$ ) which means that in the set of rules, we have both  $\{Milk_1, Cereal\}$  and  $\{Milk_2, Cereal\}$ . After categorizing them as families, we can collapse these two rules into one  $\{Milk, Cereal\}$ . Now when generating the data based on this rule, we are just making an order of  $Milk$  and  $Cereal$ , while  $Milk$  is later replaced by one of its members ( $Milk_1$  or  $Milk_2$ ) based on their relative popularity (either extracted from the sample dataset or specified by the user). The process

| Generated Rules |                            |
|-----------------|----------------------------|
| [1];            | (0.19431524547803616 1880) |
| [2];            | (0.3170025839793282 3067)  |
| [3];            | (0.32031007751937984 3099) |
| [4];            | (0.31328165374677003 3031) |
| [5];            | (0.31111111111111111 3010) |
| [6];            | (0.39080103359173124 3781) |
| [7];            | (0.29209302325581393 2826) |
| [8];            | (0.19410852713178295 1878) |
| [9];            | (0.3054263565891473 2955)  |
| [10];           | (0.2158139534883721 2088)  |
| [11];           | (0.18873385012919897 1826) |
| [7, 9];         | (0.1690956072351421 1636)  |
| [6, 9];         | (0.12454780361757106 1205) |
| [4, 6];         | (0.17870801033591732 1729) |
| [3, 6];         | (0.18470284237726098 1787) |
| [2, 5];         | (0.1537984496124031 1488)  |
| [3, 4];         | (0.182015503875969 1761)   |
| [3, 4, 6];      | (0.15979328165374676 1546) |

Fig. 6. Sample association rules extracted from the generated dataset

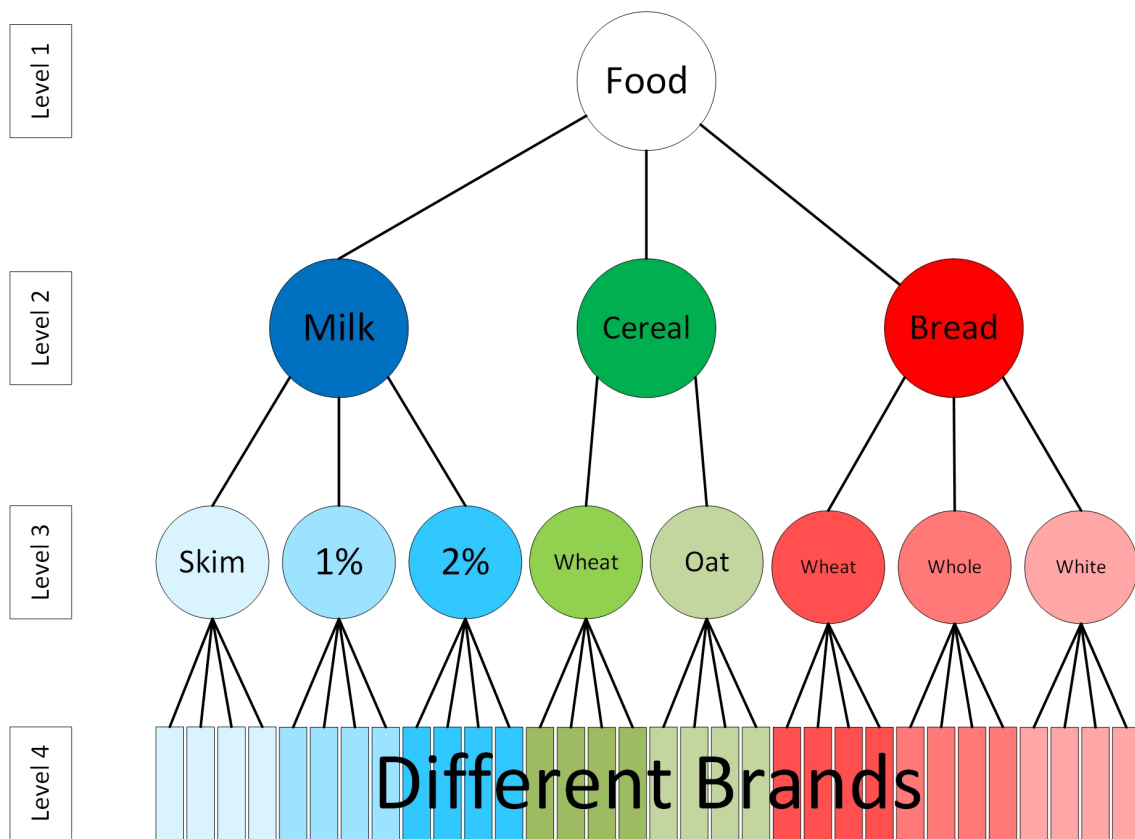


Fig. 7. A simple hierarchy representation of food products in a warehouse

of replacing a family with its members in the set of association rules will be based on probability distribution of the products

in the family. The columns in Figure 8 illustrate a sample probability distribution function (PDF). By generation a cumulative

distribution function (CDF) (the line in Figure 8) and random sampling, each family can be replaced by its members based on their probability of appearance in the orders.

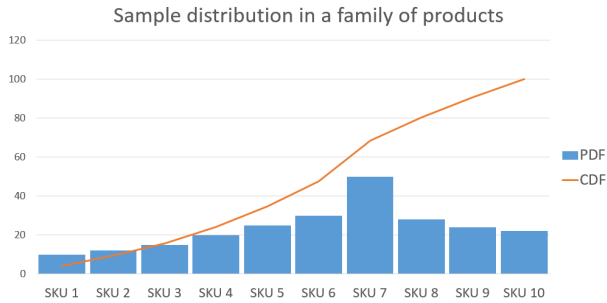


Fig. 8. A representation of product probability distribution function in Phase II

The products will be assigned to families based on product name, description, and most importantly, barcode. Word mining, word processing, and clustering will be used to assign products to their respective families. Families will be formed in a way that they will not have any overlap which will result in each product being a member of only one family. This can give us the ability to add members to families while keeping the rules intact. For doing that, a new member will be added to the family based on that families overall popularity compared to other families, and will be assigned a random relative popularity based on the popularity of other members of the family. Then when an order is generated based on the  $\{Milk, Cereal\}$  rule, the *Milk* can be replaced with both older members of that family or the new member. The same happens on the other side of the rule (*Cereal*). If the number of members in each family are high enough, the discrete probabilities illustrated in Figure 8 will be replaced with more continuous version. This continuous PDF can follow any of the known distribution including Normal, Lognormal, etc. (Figure 9)

Although in this case the SKU level rules will not remain the same, the goal is to keep the abstract of the rules (a.k.a. family rules) intact, and that is how we test the generated data to check whether they do, in fact follow the real data pattern. Figure 10 represents a flowchart of the overall process of generating order data. The boxed colored in blue are replica of phase I while the boxed colored in orange illustrate the processes of Phase II.

#### IV. FUTURE WORK

As it was mentioned in Section III-A, we were able to extract association rules with a specified significance level from a real dataset and later generate a conceptually unlimited number of orders based on this set of association rules. In the generated data, the number of SKUs and the pattern that they appear in the orders follows the same pattern as that of the real data. By providing Section III-B we are trying to solve this problem. We expect the results to be a synthetic Dataset Generator in which the user can specify the number of orders

and number of SKUs. Since this data generator mimics the real data and creates the orders and SKUs based on an actual data sample, characteristics such as correlation and popularity of the products will remain intact while the size of the data can be changed which will be highly helpful in modeling and testing order picking processes. Moreover, by incorporating manually created association rules and/or editing extracted rules, the user can change the characteristics of the generated data to suit the needs of the experiment. The method described in Sections III-A is already developed and coded, but further testing, and scenario replication is needed to ensure its performance with different data structures.

The method that we are going to use in Section III-B is current under development and testing. Once completed, the method needs to be tested both individually and also in cooperation with the process in Section III-A to ensure its viability and determine whether it, in fact, does keep the general structure of the data intact. The general method can also be expanded to be used not only in generating order picking data, but any kind of market basket analysis or correlated data generation.

#### REFERENCES

- [1] M. Ansari and J. S. Smith, "Warehouse operations data structure (wods): A data structure developed for warehouse operations modeling," *Computers & Industrial Engineering*, vol. 112, pp. 11–19, 2017.
- [2] S. P. Parker, "McGraw-hill dictionary of scientific and technical terms," *New York: McGraw-Hill, 1989, 4th ed., edited by Parker, Sybil P.*, 1989.
- [3] T. Tassier, "Sir model of epidemics," *Annual report*, 2005.
- [4] A. Skvortsov, R. Connell, P. Dawson, and R. Gailis, "Epidemic modelling: Validation of agent-based simulation by using simple mathematical models," in *MODSIM 2007 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand*, 2007, pp. 657–662.
- [5] D. García and M. Millán, "A prototype of synthetic data generator," in *Computing Congress (CCC), 2011 6th Colombian*. IEEE, 2011, pp. 1–6.
- [6] Y. Tay, "Data generation for application-specific benchmarking," *VLDB, Challenges and Visions*, 2011.
- [7] M. Frank, M. Poess, and T. Rabl, "Efficient update data generation for dbms benchmarks," in *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*. ACM, 2012, pp. 169–180.
- [8] D. Jeske, D. Gokhale, and L. Ye, "Generating synthetic data from marginal fitting for testing the efficacy of data-mining tools," *International journal of production research*, vol. 44, no. 14, pp. 2711–2730, 2006.
- [9] J. Eno and C. W. Thompson, "Generating synthetic data to match data mining patterns," *IEEE Internet Computing*, vol. 12, no. 3, 2008.
- [10] D. R. Jeske, P. J. Lin, C. Rendon, R. Xiao, and B. Samadi, "Synthetic data generation capabilities for testing data mining tools," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*. IEEE, 2006, pp. 1–6.
- [11] M. R. Berthold, N. Cebon, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel, "Knime-the konstanz information miner: version 2.0 and beyond," *ACM SIGKDD explorations Newsletter*, vol. 11, no. 1, pp. 26–31, 2009.
- [12] R. C. Blattberg, B.-D. Kim, and S. A. Neslin, "Why database marketing?" in *Database Marketing*. Springer, 2008, pp. 13–46.
- [13] M.-S. Chen, J. Han, and P. S. Yu, "Data mining: an overview from a database perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 866–883, 1996.
- [14] J. S. Park, M.-S. Chen, and P. S. Yu, *An effective hash-based algorithm for mining association rules*. ACM, 1995, vol. 24, no. 2.
- [15] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, "Dynamic itemset counting and implication rules for market basket data," in *ACM SIGMOD Record*, vol. 26, no. 2. ACM, 1997, pp. 255–264.



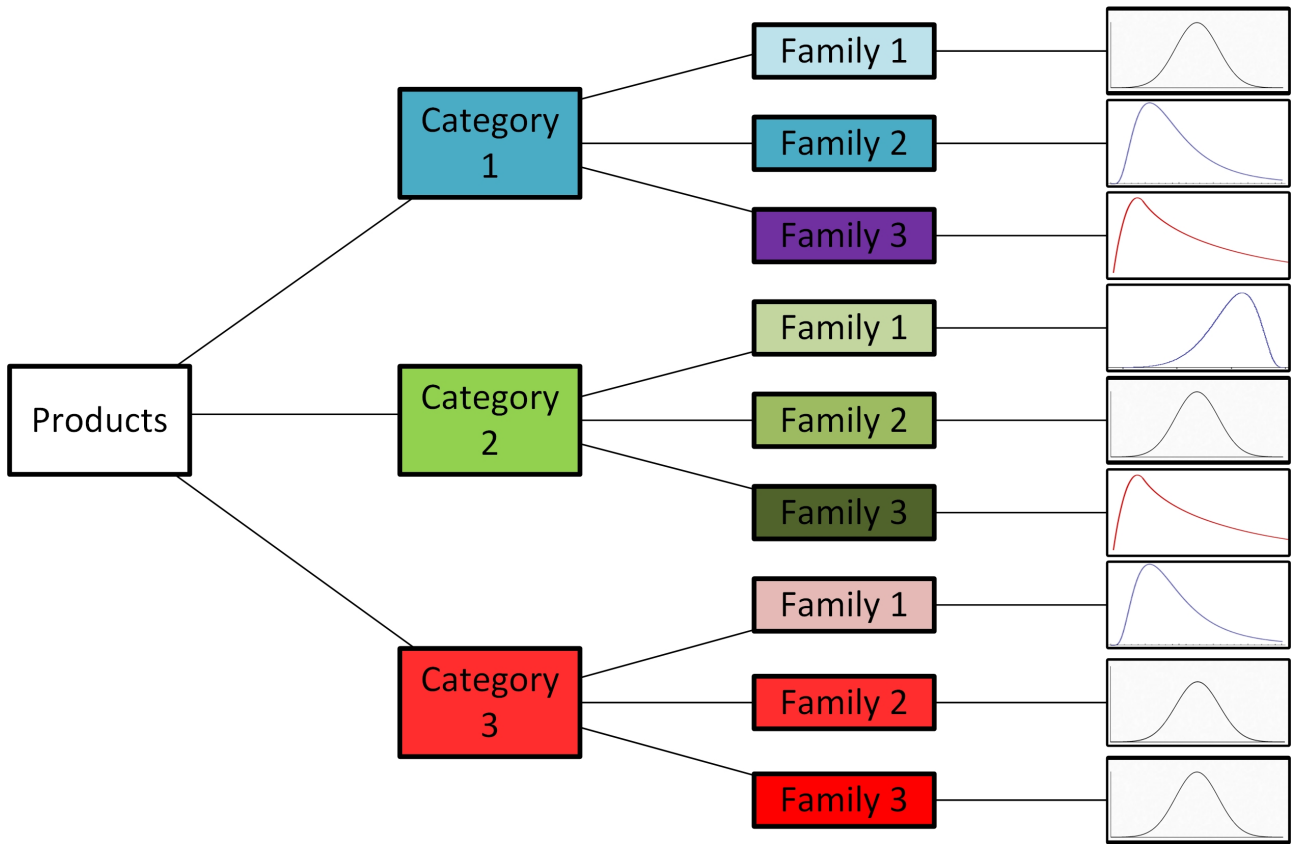


Fig. 9. A representation of product distribution in Phase II

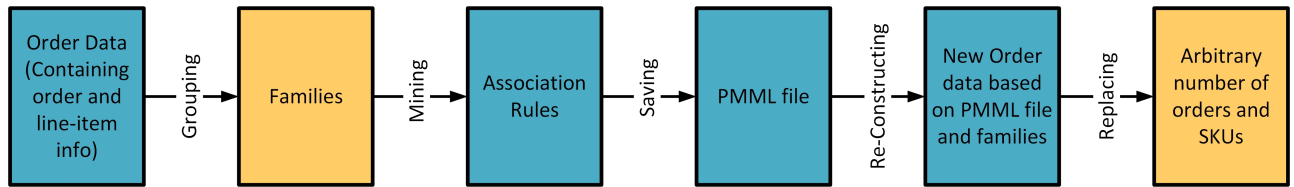


Fig. 10. Flowchart of Phase II in developing arbitrary number of SKUS

- [16] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations," in *Acm Sigmod Record*, vol. 26, no. 2. ACM, 1997, pp. 265–276.
- [17] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets, "Using association rules for product assortment decisions: A case study," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1999, pp. 254–260.
- [18] Y.-L. Chen, K. Tang, R.-J. Shen, and Y.-H. Hu, "Market basket analysis in a multiple store environment," *Decision support systems*, vol. 40, no. 2, pp. 339–354, 2005.
- [19] A. Guazzelli, M. Zeller, W.-C. Lin, G. Williams *et al.*, "Pmml: An open standard for sharing models," *The R Journal*, vol. 1, no. 1, pp. 60–65, 2009.
- [20] R. Pechter, "What's pmml and what's new in pmml 4.0?" *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 19–25, 2009.