Generation of Non-Symmetric Synthetic Datasets

Thomas W. Pensyl
Department of Computer Science, North Carolina State University
Raleigh, NC 27695-8206, USA
twpensyl@ncsu.edu

Rong Huang
Department of Operations Research, North Carolina State University
Raleigh, NC 27695-8206, USA
rhuang@ncsu.edu

October 24, 2011

1 Introduction

We may define procedures for generating synthetic databases, for the purpose of testing database management systems. In this report, we will examine two such procedures and gauge the usefulness of the resulting datasets. The rest of this report is organized as follows. In Section 2, we analyze the type II non-symmetric dataset and show how to pick its parameters correctly. In Section 3 we define and analyze a variation, the type III non-symmetric dataset. In Section 4 we define a metric for how useful a dataset is for testing view selection algorithms. Finally, in Section 5 we present some experiment results and observations about Type II and Type III non-symmetric datasets.

2 Type II non-symmetric synthetic dataset

2.1 Definition

In [2] Chirkova, Fathi, and Huang defined the Type II non-symmetric synthetic dataset. They gave the following algorithm for generation. D is a Type I symmetric dataset having K attributes $a_1, ..., a_K$. Each attribute a_i has m_i possible values. S_L is the desired size of the Type II dataset. $p_1, ..., p_k$ is a set of input probabilities used as shown below. (See the original paper for more details and examples.)

Step 1. Input S_L and $D(K; m_1, ..., m_K)$. Choose $p_1, ..., p_K$ such that $S_L \geq \prod_{i=1}^{K} m_i p_i^{m_i}$. Set $k = 1$.

Step 2. Mark each row in the current table as 'selected' with probability $1 - p_k$. (If a row is marked, it will be eliminated.)

Step 3. For every row that is marked, also mark any rows which differ only in attribute a_k.

Step 4. Eliminate all the marked rows in the table. If $k = K$, output the table as D', otherwise $k = k + 1$ and go to step 2.

After generating some datasets with this algorithm, we find the bound given above on the final size of the dataset, S_L, is extremely loose, to the extent that the probabilities $p_1, ..., p_k$ must be chosen by trial and error in order to obtain a dataset of the desired size. This works for small datasets, but would be impractical when generating a large dataset, which could take hours for each attempt. Thus it is desirable to have a closer estimate of the final size of a dataset, as well as a way to generate values of p_i which produce a dataset of the desired size.

2.2 Expected size of dataset

Let g_k denote a group of rows which share all attributes in common except for a_k. Then during the kth iteration of row removal, the existing rows in any g_k either all survive or are all removed. Thus, the removal or survival of these rows are completely dependent events. The removals of rows which are not in the same g_k are not directly dependent events, although they may be indirectly dependent. However, we will generally treat them as independent events to simplify the analysis below.

Claim 1 *Let q_k be the probability that a row survives the kth iteration of row removal. Then let Q_k be the cumulative probability that a row survives all iterations up to and including the kth iteration, so that $Q_0 = 1$ and $Q_k = \prod_{i=1}^{k} q_k$. Then for $k > 0$,*

$$q_k = p_k((1 - Q_{k-1}) + Q_{k-1}p_k)^{m_k - 1} \ . \tag{1}$$

Proof. Consider the first round of elimination. Each row starts in a group of m_1 rows which are identical except for a_1. The odds of a row surviving are the odds of all m_1 rows in its group remaining unmarked (each with probability p_1). This agrees with the above definition of $q_1 = p_1((1 - 1) + p_1)^{m_1 - 1} = p_1^{m_1}$.

Now consider the kth round of elimination. For a row to survive, it must first itself remain unmarked (probability p_k). Additionally, each of the $m_k - 1$ rows in the row's group must either remain unmarked or have been removed in a previous round. Treat the previous

survival or removal of each row within the group as independent events with probability Q_{k-1} or $1 - Q_{k-1}$, respectively. This yields the above expression for q_k. ■

We have then that q_k is the expected fraction of rows which survive iteration k. The expected number of remaining rows in the final table is

$$\bar{S} = S_0 \prod_{i=1}^{K} q_i = S_0 Q_K = \prod_{i=1}^{K} m_i q_i \tag{2}$$

where S_0 is the size of the initial, full table. Our experiments confirm this to be an accurate prediction, despite the aforementioned simplifications about independence. We can easily choose $q_1, ..., q_K$ to satisfy this equation. However, we still need to know the inputl values of $p_1, ..., p_k$ which produce them. This form is not, in general, analytically solvable for p_k, motivating the following approximation.

Lemma 1 *For* $0 < x \leq 1$ *and* $0 \leq y \leq 1$

$$1 - y + yx \geq x^y \ .$$

Proof. Expand x^y into its Taylor series about $x = 1$. Taking the first two terms, we observe the remaining terms are all negative.

$$x^y = \sum_{n=0}^{\infty} \left(\prod_{i=0}^{n-1} (y - i) \right) \frac{1^{y-n}}{n!} (x-1)^n$$

$$x^y = 1 + y(x-1) + \sum_{n=2}^{\infty} y(x-1) \left(\prod_{i=1}^{n-1} (y-i)(x-1) \right) \frac{1}{n!} \leq 1 + y(x-1)$$

$$x^y \leq 1 - y + xy$$

■

From the above lemma we can see,

$$q_k = p_k (1 - Q_{k-1} + Q_{k-1} p_k)^{m_k - 1} \geq p_k (p_k^{Q_{k-1}})^{m_k - 1} = p_k^{1 + Q_{k-1}(m_k - 1)} \ . \tag{3}$$

In practice this bound appears to be very tight for most cases, and can itself be used as a good approximation. Solving for p_k, we get a close approximation,

$$p_k \approx q_k^{\frac{1}{1 + Q_{k-1}(m_k - 1)}} \ . \tag{4}$$

Thus, we modify the procedure as follows:

Step 1. Input S_L and $D(K; m_1, ..., m_K)$. Choose $q_1, ..., q_K$ such that $S_L = \prod_{i=1}^{K} m_i q_i$. Set $k = 1$.

Step 2. Set $p_k = q_k^{\frac{1}{1+Q_{k-1}(m_k-1)}}$. Then mark each row in the current table as 'selected' with probability $1 - p_k$. (If a row is marked, it will be eliminated.)

Step 3. For every row that is marked, also mark any rows which differ only in attribute a_k.

Step 4. Eliminate all the marked rows in the table. If $k = K$, output the table as D', otherwise $k = k + 1$ and go to step 2.

2.3 Memory requirement

To accelerate the procedure, we performed it on a simulated database in memory. A straightforward implementation of the elimination procedure required that we initially start with all possible rows in memory. This made the space complexity $O(S_0)$. Our strategy to minimize memory usage was to use a bitmap, where each bit address mapped to a potential row in the table, and the bit value stored whether the row was present or not. This scheme only required one bit per row. However, this could still quickly become an unfeasible amount of memory. For example, if we attempted to generate a table with similar size and attributes as the 7-attribute TCP-H dataset considered in Section 5, there would be over $6x10^{16}$ potential rows. This would have required over 7 petabytes of memory, even though the final table would only have around 300,000 rows.

However, if we examine the procedure closely we find that we need not start with all rows in memory. The key observation is that, during the ith iteration, rows which differ in any of $a_{i+1}, ..., a_K$ are completely independent of each other. If we sort the rows into groups which have the same values of $a_{i+1}, .., a_K$, we can perform iteration i on one group at a time, without regard to any rows outside that group. Figure 1 shows the minimal independent groups of a dataset at each iteration, organized as a tree of dependencies.

From this relationship we can see that the iterations do not need to be run strictly in order; we just need all previous iterations to have been run on the children of a group. Thus we can use a postorder traversal of the tree to perform the row removals. For example, we could run iterations 1 and 2 on the left half of the above rows before iteration 1 was run on the right half. But in order to run iteration 3, we must first run iterations 1 and 2 on the right half.

We take advantage of this with the following procedure: Allocate one hash table for each of the K iterations. Visit each node of the tree, using a postorder traversal. For each node:

1. Clear the corresponding table for this iteration.

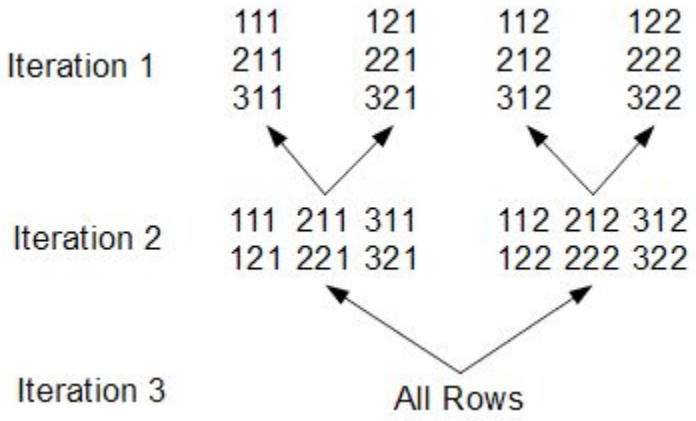

Figure 1: Each group is independent of other groups in the same level.

2. If it is a leaf (Iteration 1), populate the table with all potential rows.
3. Run the corresponding removal iteration on the table.
4. Add the remaining rows into the next iteration's table. If this is the root (last iteration), then the table is the final table.

The memory required is the sum of the size of each of the K hash tables. At the start of iteration i, we have $\prod_{j=1}^{i} m_j$ potential rows in a group, with a expected survival rate of $\prod_{j=1}^{i-1} q_j$ up to that point. Let the size of the hash tables be some constant C times the expected number of initial rows in each table. Then the memory M required is

$$M = \sum_{i=1}^{K} \frac{C}{q_i} \prod_{j=1}^{i} q_j m_j$$

The 7-attribute TCP-H-like dataset would require around 2 gigabytes of memory, with appropriate choice of probabilities. Consider the case that $m_1 = ... = m_K = m$ and $q_1 = ... = q_K = q$. Then

$$M = \sum_{i=1}^{K} \frac{C}{q} (qm)^i = \frac{C}{q} * \frac{(qm)^{K+1} - 1}{qm - 1}$$

$$M \in O\left(\frac{(qm)^K}{q}\right) = O\left(\frac{\bar{S}}{q}\right)$$

where $\bar{S}$ is the expected size of the final dataset. This is a much more feasible memory requirement. Additionally, using hash tables would preserve the constant-time access of the bitmap, so the time complexity should be similar or better, since we can avoid traversing a huge, sparse bitmap. Also, this method has inherent parallelism which could be exploited for multithreading.

2.4 Further Analysis

Claim 2 *Let $V_{\hat{i}}$ be the view consisting of all attributes except a_i. Then the expected size of $V_{\hat{i}}$ is given by*

$$|V_{\hat{i}}| = \left((1 - Q_{i-1} + Q_{i-1}p_i)^{m_i} - \left(1 - Q_{i-1} + Q_{i-1}p_i(1 - \frac{Q_n}{Q_i})\right)^{m_i} \right) \prod_{j \neq i} m_j \qquad (5)$$

Proof. Again, let g_k denote a group of rows which share all attributes in common except for a_k. Then each entry in the view $V_{\hat{k}}$ corresponds to a nonempty group in the final table. Thus we can say the expected size of the view $V_{\hat{k}}$ is

$$|V_{\hat{k}}| = |V_{\hat{k}}|_{max} \cdot P(|g_k| > 0) \ . \qquad (6)$$

The maximum view size is just the product of the number of possible values of each attribute in the view (in this case, all except a_i), so that $|V_{\hat{k}}| = \prod_{j \neq k} m_j$. We derive the probability of a nonempty group as follows.

Consider the variation that during the ith iteration, we skip Step 3, and only remove the rows directly marked, instead of the entire group. Let $\chi_i(g_i)$ be the number of rows that have been marked and removed from a particular group during the ith iteration, and let g_i' be the group of remaining rows. This makes the survival of the rows in g_i' independent from one another during iteration i. Assume as before that their survivals during the other iterations are also independent events. Now define the generating function

$$\Theta_i(x, y) \equiv \sum_{j,k} P\big(\ |g_i'| = k \ \cap \ \chi_i(g_i) = j \ \big) x^k y^j \ . \qquad (7)$$

This means, for example, that the coefficient of x^6y^2 in $\Theta_i(x, y)$ is the probability that for any given group g_i, two rows have been marked and removed during the ith iteration, and - if we leave the rest of the group intact - six rows remain in the group.

For the purposes of construction, let $\Theta_i^n(x, y)$ be the value of the generating function after the nth iteration. Consider the group after the $(i-1)$th iteration. At this point each row has survived with chance Q_{i-1} or been removed with chance $1 - Q_{i-1}$. (There is no y because we have not gone through the ith iteration yet.) Thus,

$$\Theta_i^{i-1}(x, y) = (1 - Q_{i-1} + Q_{i-1}x)^{m_i} \ .$$

For the next iteration, a row survives with probability p_i, or is marked and removed with probability $1 - p_i$. This only occurs if the row has survived to this point, so we substitute the corresponding expression for x:

$$\Theta_i^i(x, y) = \big(1 - Q_{i-1} + Q_{i-1}((1 - p_i)y + p_i x)\big)^{m_i} \ .$$

Each row which has survived this far either survives the remaining iterations with probability $\prod_{k=i+1}^{K} q_k = \frac{Q_K}{Q_i}$, or does not survive with probability $1 - \frac{Q_K}{Q_i}$. Again, substituting the corresponding expression for x we obtain the final generating function:

$$\Theta_i^k(x,y) = \Theta_i(x,y) = \Big(1 - Q_{i-1} + Q_{i-1}\big((1-p_i)y + p_i(1 - \frac{Q_K}{Q_i} + \frac{Q_K}{Q_i}x)\big)\Big)^{m_i} \ . \quad (8)$$

Now, to relate g_i' to g_i, we consider any groups with marked rows as being empty, since they should have been removed. Thus the probability of a nonempty group g_i is the sum of the coefficients of all terms with an x-component (nonempty) but no y-component (none marked).

$$\begin{aligned} P(|g_i| > 0) &= \sum_{k>0} P(|g_i| = k) \\ &= \sum_{k>0} P\big(\ |g_i'| = k \ \cap \ \chi_i(g_i) = 0\ \big) \\ &= \left(\sum_{k\geq 0} P\big(\ |g_i'| = k \ \cap \ \chi_i(g_i) = 0\ \big)\right) - P\big(\ |g_i'| = 0 \ \cap \ \chi_i(g_i) = 0\ \big) \\ &= \Theta_i(1,0) - \Theta_i(0,0) \end{aligned}$$

Evaluating the above expression with (8) and substituting into (6) yields the premise. ■

3 Type III non-symmetric synthetic dataset

3.1 Definition

The main motivation for the elimination procedure for the Type II dataset is to produce a change in the dataset's view sizes. However, after conducting this elimination, we observe that it only has a significant impact on views with many attributes. The views with less attributes are left untouched or barely modified in size.

To address this we define a second row elimination method by modifying the previous method. Again, start with the type I non-symmetric synthetic dataset $D(K; m_1, ..., m_K)$. We then conduct a round of row elimination for each of the $\binom{K}{2}$ unordered pairs of distinct attributes (a_i, a_j). For each round, we first remove each row with some probability p. For each row removed, we then also remove any other rows which have the same values for both a_i and a_j. This will ensure a change in the the size of views with only two attributes, in addition to a change in the size of the larger views.

3.2 Expected size of dataset

Both the Type II and III procedures involve a number of removal iterations, in each which rows are removed according to certain groups. Thus our results about the expected size of the dataset are analogous to those for the Type II dataset.

Claim 3 *Let q_k be the probability that a row survives the kth iteration of row removal. Then let Q_k be the cumulative probability that a row survives all iterations up to and including the kth iteration, so that $Q_0 = 1$ and $Q_k = \prod_{i=1}^{k} q_k$. Also let the initial size of the K-attribute dataset be S_0. If a_i and a_j are the pair of attributes selected for iteration $k > 0$,*

$$q_k = p_k(1 - Q_{k-1} + Q_{k-1}p_k)^{(\frac{S_0}{m_i m_j} - 1)}. \tag{9}$$

Proof. The size of the groups during the kth iteration is $\frac{S_0}{m_i m_j}$. The result then follows from the same reasoning as in Claim 1. ■

In the same way as before, we use an approximation to solve for p in the above equation.

$$p_k \approx q_k^{(1 + Q_{k-1}(\frac{S_0}{m_i m_j} - 1))^{-1}} \tag{10}$$

We thus define the following procedure for generating a Type III dataset:

Step 1. Input S_L and $D(K; m_1, ..., m_K)$. Choose an ordering of all ordered pairs of unique attributes (a_i, a_j) such that $1 \leq i < j \leq K$. Choose corresponding $q_1, ..., q_{\binom{K}{2}}$ such that $S_L = \prod_{k=1}^{K} m_i \cdot \prod_{k=1}^{\binom{K}{2}} q_k$. Set $h = 1$.

Step 2. If (a_i, a_j) is the hth pair, set

$$p_h = q_h^{(1 + Q_{h-1}(\frac{S_0}{m_i m_j} - 1))^{-1}}.$$

where $Q_n = \prod_{k=1}^{n} q_k$ and $Q_0 = 1$. Mark each row in the current table as 'selected' with probability $1 - p_h$. (If a row is marked, it will be eliminated.)

Step 3. For each row that is marked, also mark any other rows which share the same value of both a_i and a_j.

Step 4. Eliminate all the marked rows in the table. If $h = \binom{K}{2}$, output the table as D', otherwise $h = h + 1$ and go to step 2.

Note that, unfortunately, the memory-efficient implementation described for Type II does not work for the Type III procedure.

4 A metric for usefulness

Our goal in making these definitions is to obtain datasets which are useful in testing algorithms for the view selection problem. The view selection problem, in brief, consists of finding the most efficient set of views to materialize for answering a given set of queries, given a size constraint on the set of views. [1]

For example, with unlimited space, a set of queries could be answered by simply storing the views which most directly correspond to each query. Given space constraints, however, one must consider storing larger views which can answer more than one query.

This problem may vary in difficulty depending on the dataset. If a view is very close in size to its descendents, then it will almost always be selected instead of its descendents. On the other hand, if a view is very large compared to its descendents, it will never be selected. Either case is trivial and uninteresting. For a dataset to be useful in testing view selection algorithms, it must contain a significant number of views whose size is somewhere in the middle. We use the following metric to quantify the "usefulness" of a view.

Let $\mathcal{C}(V)$ be the set of direct descendents, or children, of view V (views obtained from V by considering one less attribute). Then we define $\tau(V)$ as the ratio of the view's size to the total size of all its direct children:

$$\tau(V) \equiv \frac{|V|}{\sum_{c \in \mathcal{C}(V)} |c|}.$$

Observe that if a view V can be used to answer a set of queries, $\mathcal{C}(V)$ can also be used to answer the same queries. Thus, in order for a view to be useful, it must be smaller than the total size of its direct children. This yields an upper requirement for $\tau(V)$:

$$|V| < \sum_{c \in \mathcal{C}(V)} |c|$$

$$\frac{|V|}{\sum_{c \in \mathcal{C}(V)} |c|} = \tau(V) < 1 \ .$$

On the other hand, we want the direct children of V to be significantly smaller than $|V|$. Let $n(V)$ be the number of attributes in view $|V|$. Then for some constant $b \geq 1$, we require that the view's average child-to-parent size ratio be less than $1/b$:

$$\frac{1}{n(V)} \sum_{c \in \mathcal{C}(V)} \frac{|c|}{|V|} < \frac{1}{b}$$

$$\frac{|V|}{\sum_{c \in \mathcal{C}(V)} |c|} = \tau(V) > \frac{b}{n(V)} \ .$$

Let $\mathcal{W}(D)$ be the set of all possible views with two or more attributes over dataset D. We measure the usefulness of the entire dataset by the fraction of views in $\mathcal{W}(D)$ which satisfy

the above bounds:

$$\omega(D,b) \equiv \frac{1}{|\mathcal{W}(D)|}\left|\left\{V \in \mathcal{W}(D) \;\middle|\; \frac{b}{n(V)} < \tau(V) < 1\right\}\right|.$$

5 Experimental results

We used the metric defined in the previous section to gauge the usefulness of Type II and III datasets generated using various parameters. The value used for b is somewhat arbitrary, but we used $b = 2$ in all the following results to obtain a consistent comparison. S/S_0 is the ratio of the dataset size to the original dataset's size. It is the fraction of rows which remain after elimination.

After running tests we notice several trends. For a given final size, the metric can be increased by starting with a larger original dataset, and removing more rows (Table 2). Another large factor in the number of useful views is the presence (or absence) of small-valued attributes, such as attributes containing only 2 or 3 possible values (Table 3). Such attributes tend to greatly increase the number of useful views.

Type III produces a dataset with more 'useful' views for testing. However, it also takes more time to generate, and a memory-efficient method is not yet clear. Thus, for larger datasets, Type II may be preferred. Also, although we have a good estimate of the expected size, it has a very high variance, which can make it difficult to obtain a dataset of a particular

Type	# Values per Attribute	Rows	S/S_0	$\omega(D,2)$
II	50 20 10 8 5 5 4 3 2	109977	.0023	.378
II	50 20 10 8 5 5 4 3 2	536934	.0112	.386
III	50 20 10 8 5 5 4 3 2	59624	.0012	.620
III	50 20 10 8 5 5 4 3 2	244550	.0051	.728

Table 1: 9-attribute datasets of various sizes

Type	# Values per Attribute	Rows	S/S_0	$\omega(D,2)$
II	7 7 7 7 7 7 7	301850	.3665	.039
II	10 10 10 10 10 10 10	300730	.0301	.055
II	15 15 15 15 15 15 15	304233	.0018	.165
III	7 7 7 7 7 7 7	296957	.3606	.063
III	10 10 10 10 10 10 10	306216	.0306	.110
III	15 15 15 15 15 15 15	279474	.0016	.228

Table 2: Starting with different sized datasets.

Type	# Values per Attribute	Rows	S/S_0	$\omega(D,2)$
TCPH	75000 10000 2524 500 11 3 2	299814		.158
II	80 40 30 25 2 2 2	290820	.0151	.465
II	80 40 20 10 5 3 2	301065	.0157	.213
II	80 20 10 9 5 5 5	305696	.0170	.087
III	80 40 30 25 2 2 2	291331	.0152	.402
III	80 40 20 10 5 3 2	290119	.0151	.496
III	80 20 10 9 5 5 5	309093	.0172	.299

Table 3: TCPH vs. Type II and III datasets, and the effect of small-valued attributes.

size. Overall, both types can produce a better dataset than the TCPH dataset we used for comparison, and both appear useful for testing database management systems.

References

[1] Chirkova, R. Halevy, A. Y., Suciu, D.: A formal perspective on the view selection problem. VLDB J. 11(3), pp. 216-237. (2002)

[2] Huang, R., Chirkova, R., Fathi, Y. (2011). Synthetic Datasets. Retrieved from North Carolina State University website: ftp://ftp.ncsu.edu/pub/unity/lockers/ftp/csc_anon/tech/2011/TR-2011-10.pdf