Data Generation for Application-Specific Benchmarking

Y.C. Tay
National University of Singapore
dcstayyc@nus.edu.sg

ABSTRACT
The Transaction Processing Council (TPC) has played a pivotal role in the database industry’s growth over the last twenty-five years. However, its handful of domain-specific benchmarks are increasingly irrelevant to the multitude of data-centric applications, and its top-down process is slow.

This mismatch calls for a paradigm shift to a bottom-up community effort to develop tools for application-specific benchmarking. Such a development program would center around techniques for synthetically scaling (up or down) an empirical dataset. This engineering effort in turn requires the development of a database theory on attribute value correlation.

1. INTRODUCTION
A database management system for an enterprise or web service is a complicated collection of software and hardware. Its complexity and its importance require that a storage service is a complicated collection of software and hardware. Its complexity and its importance require that a storage service is a complicated collection of software and hardware.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Articles from this volume were invited to present their results at The 37th International Conference on Very Large Data Bases, August 29th - September 3rd 2011, Seattle, Washington.

Proceedings of the VLDB Endowment, Vol. 4, No. 12
Copyright 2011 VLDB Endowment 2150-8097/11/08... $ 10.00.

This research was supported in part by MOE Grant No. R-252-000-394-112.
1http://www.tpc.org/

not covered by the benchmarks. This situation can only get worse, as the proliferation of new data-centric applications far outpaces the approval of new TPC benchmarks [18].

Hence, there is an urgent need for a paradigm shift: Instead of continuing with TPC’s top-down approach to domain-specific benchmark design by committee consensus, the database community should collaborate in a bottom-up program to develop tools for application-specific benchmarking.

The pivotal role played by the TPC benchmarks in the last two decades suggests that such a program will play a similar role in database research and development for many years to come.

2. THE DATASET SCALING PROBLEM
The central problem for the above program lies in the scaling of an empirical dataset. We state this issue as the Dataset Scaling Problem:

Given a dataset $D$ and a scale factor $s$, generate a synthetic dataset $\tilde{D}$ that is similar to $D$ but $s$ times its size.

One can define “$s$ times its size” in various ways (number of records or bytes, etc.), and numerical precision is unnecessary — if $s = 3$, it would not matter if the generated $\tilde{D}$ were actually $3.14$ times $D$’s size (however defined).

Rather, the issue is “similarity”; e.g. if $D$ is a set of tables, then $\tilde{D}$ must reflect relationships among the columns and rows of $D$. Depending on the application, one could define similarity in terms of statistical correlation, graph properties (e.g. if $D$ represents a social network), etc. If similarity is defined by query results, then that raises the question of how queries are to be factored into the generation of $\tilde{D}$.

3. MOTIVATION FOR $S > 1$, $S = 1$, $S < 1$
There are various possibilities for why one might want to synthetically scale up ($s > 1$) an empirical dataset. Some web applications have user populations that grow at breakneck speed (one recent example being Animoto2), so a small but fast-growing service may need to test the scalability of their hardware and software architecture with larger versions of their datasets.

Another example is where an enterprise supplies a vendor with only a sample of its dataset (e.g. the entire dataset is too large for easy transfer, as is the case with uploading into

2http://animoto.com
D
deficit copies. Note, however, that some information leakage
This scarcity may be alleviated by a tool for making syn-
by uploading a synthetic copy.
(i.e. their crown jewels), an enterprise can reduce their risk
(leveraging on its elasticity, and before investing in a partic-
ferent system configurations or implementations in the cloud

dataset, the dataset owner may be motivated by privacy

to randomly pick 1000 buyers; e.g. we may need to add their
suppliers’ other buyers, and this recursive adding can grow
the sample to an indeterminate size.

For example, if a dataset contains 2000000 buyers, and we
would like to downsize it (choice (a) in Fig. 1), this may violate
in
\
\{a, b, c, d\}
\]
\end{figure}

\begin{figure}
\centering
\includegraphics[width=\columnwidth]{figure3}
\caption{Users \(x\) and \(y\) comment on each other’s photographs. Such interactions induce inter-column and inter-row correlations in the tables above.}
\end{figure}

\section{4. Attribute Value Correlation Problem}

To see why scaling a dataset is nontrivial, consider the toy
relational \(D\) in Fig. 1. An obvious possibility for \(s = 2\) is to
scale \(D\) to \(\tilde{D}\) by making a copy. However, if the copy uses
new values not in \(D\) (choice (a) in Fig. 1), this may violate
attribute constraints like value range or number of distinct
values; moreover, if \(D\) were a social network, such copying
will create two social networks that are disconnected from
each other. On the other hand, if the copy uses only values
in \(D\) (choice (b) in Fig. 1), the result of a join query may
scale by a wrong factor.

Besides, copying does not work for \(s < 1\).

For a better understanding of the issues, suppose \(D\) is a
relational dataset (from a service like Flickr\(^{3}\)) with four ta-
bles \texttt{User}, \texttt{Photo}, \texttt{Comment} and \texttt{Tag} that records photographs

3http://www.flickr.com
The Attribute Value Correlation Problem for Social Networks (AVC\textsubscript{SN})

Suppose a database $D$ records data from a social network. How do the social interactions affect the correlation among attribute values in $D$?

For example, two persons with the same age, or use “similar” tags, may be more likely to comment on each other’s photographs; the Facebook\footnote{http://www.facebook.com/} wall for a user $x$ may receive many comments on $x$’s birthday; etc. What correlation do such social interactions induce in the database?

Dataset scaling aside, attribute value correlation is of independent interest since it is relevant to query optimization, materialized views, index design and storage organization (clustering, partitioning, sharding, etc.) --- see Sec. 5.

The mushrooming of online social networks is as unstoppable as the spread of the Web, and many businesses and services are angling to leverage on them. These networks first appeared several years ago and, going forward, more and more datasets will be generated by them.

We were therefore surprised that, when we encountered the attribute value correlation issue and looked into the literature, we found no theory to guide us in scaling datasets that are generated by social networks. This is why we highlight the AVC\textsubscript{SN} problem here --- we believe it points to a rich, new area for database researchers reminiscent of dependency theory.

5. BACKGROUND

The TPC approach is being adopted by a new generation of benchmarks \cite{1, 6}. However, Seltzer et al. \cite{16} have observed how standard benchmarks can be irrelevant for particular applications, and argued for application-specific benchmarking. For database systems, this alternative approach must start with application-specific datasets.

The TPC way of generating a completely synthetic dataset can be traced back to the Wisconsin benchmark \cite{7}. Its designers had considered the possibility of using real data to construct a benchmark for database systems. However, they decided against that for three reasons: (i) They would need the dataset to be large, so that it reflects their underlying probability distributions; these days, this is not an issue for many databases, some of which are huge. (ii) Query design is easier if the data is synthetic, so one can adjust the table sizes, join selectivity, etc.; this is also not an issue for application-specific benchmarking, since the application would already have a set of queries on hand. (iii) Empirical datasets are hard to scale. This third reason remains true; however, 28 years have passed, and a relook at the problem is long overdue.

So far, the use of empirical data in dataset generation is very limited. For example, MUDD \cite{17} only extracts names and addresses from a real dataset; TEXTURE \cite{9} extracts word distribution, document lengths, etc. from “seed” documents and use them to independently generate synthetic documents (like how TPC generates tuples); and Duan et al. samples from a given RDF dataset (i.e. $s < 1$ and no synthetic table generation) \cite{8}.

Similarly, the data generating tool by Houkjær et al. \cite{12} only uses cardinalities and value distributions extracted from real data and, other than referential integrity, does not replicate their correlation. This is also current practice in the industry; e.g. Teradata and SQL Server both use only column statistics (maximum, mode, number of rows and distinct values, etc.) for data generation. IBM’s Optim\footnote{http://www-01.ibm.com/software/data/data-management/optim-solutions/} and HP’s Desensitizer \cite{5} are focused on data extraction and obfuscation, not synthetic data generation.

Bruno and Chaudhuri’s Data Generation Language \cite{4} can specify value distributions and generate data tuples, while Hoag and Thompson’s Synthetic Data Description Language \cite{11} has a construct for specifying foreign keys, but data generation by both languages do not replicate correlation between foreign keys, nor between rows, etc.

A couple of tools use the queries to guide data generation: Binnig et al.’s reverse query processing \cite{2} uses query results to generate a smallest dataset to test the application, whereas QAGen uses a given query plan with size constraints to generate a corresponding dataset \cite{3}, without requiring similarity to real data. Thus, neither tool addresses the Dataset Scaling Problem.

Even so, queries can help the discovery of inter-attribute correlations. CORDS \cite{13} is a tool that uses the application queries to select columns whose correlations are important for query optimization; it also uses the correlation to generate synthetic data, but this purely valued-based generation (e.g. humidity and temperature) cannot replicate the entity-based correlation (e.g. gardeners and flowers) described in Sec. 4. CORADD \cite{14} is another tool that discovers attribute correlations that are important to the queries, and use them to design materialized views and indexes.

We see CORDS and CORADD as early signs of a growing interest in the Attribute Value Correlation problem.
Progress in understanding this problem would help database research on social networks. This is because much of the activity in these networks may have little to do with explicitly declared lists of friends and contacts, etc.; rather, they are social interactions (e.g., writing on Facebook walls [19]) that are implicitly captured by values in several columns. Online social networks are major users of data-centric systems, and a better understanding of such data is necessary if one is to extract value from these systems. This is why we highlight the AVC\textsubscript{SN} problem here.

Incidentally, current techniques for growing social network graphs by adding one node at a time (e.g., the Forest Fire Model [15]) are too slow and will likely accumulate significant inaccuracies (consider, say, $s=2$ in Fig. 4).

6. CONCLUSION

We believe that the TPC benchmarking paradigm cannot sufficiently cover, in timely fashion, the diverse applications in the ballooning number of data-centric systems. We therefore propose here a paradigm shift to a collaborative program to develop tools and techniques for application-specific benchmarking.

Our contribution to such a community effort is UpSizeR\textsuperscript{6}, an open-source software that addresses the Dataset Scaling Problem for relational databases. UpSizeR is currently a first-cut tool — while it replicates correlation among key values, similar replication involving non-key values is rudimentary, since that is application-specific.

The current UpSizeR's generation of key values may be adequate for classical datasets (in finance, retail, etc.), but not for social networks. It has a swap technique for replicating the correlation in Fig. 3, but the technique is not sufficiently general for the graph injection in Fig. 4. The latter calls for a database-theoretic understanding of AVC\textsubscript{SN}.

We have highlighted two issues in our program proposal on application-specific benchmarking: (1) the Dataset Scaling Problem, for its fundamental importance to such a program; and (2) AVC\textsubscript{SN}, for its growing relevance to web-generated datasets. However, there are other related issues.

For example, applications grow with datasets; an insert transaction may generate more tuples, where the values inserted follow the correlation in the database. For a scalability study to exercise the indexes, locks, etc., the applications must also be scaled to match the dataset.

Similarly, the query log will also scale with a database. In fact, for Internet services, much of the value may lie in their click logs. One particular difficulty in log scaling lies in the correlation among the clicks. For example, a log records an interleaving of multiple click streams, so the data returned by one click is correlated with those for concurrent clicks in other streams, and probabilistically determines the next click in its own stream.

We therefore see a program to develop application-specific benchmarking as not only of considerable commercial interest, but also a rich trove of challenging problems for database research.

7. ACKNOWLEDGMENTS

Many thanks to Bing Tian Dai, Daniel T. Wang and Eldora Y. Sun for their work on UpSizeR.

\footnote{http://www.comp.nus.edu.sg/~upsizer}

8. REFERENCES