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 redesign, a scale out of machines, a new business application, etc., be adequately tested before deployment. Such testing needs to use a dataset of an appropriate size.

One possibility is to use a TPC[1] benchmark for such tests. TPC datasets can be scaled to desired sizes, and are also domain-specific: TPC-C for online transaction processing, TPC-H for decision support, etc. Vendors have used these benchmarks to improve and compare their products, and researchers have used them to test and compare their algorithms and prototypes. The TPC benchmarks have thus played an important role in the growth of the database industry and the progress of database research.

However, while there is a tremendous variety of database applications, there are only a few TPC benchmarks. It follows that a TPC benchmark may not be equally relevant to two different applications within its domain; furthermore, at any moment, there are numerous applications that are 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 $\mathcal{D}$ and a scale factor s, generate a synthetic dataset $\widetilde{\mathcal{D}}$ that is similar to $\mathcal{D}$ 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 $\widetilde{\mathcal{D}}$ were actually 3.14 times $\mathcal{D}$'s size (however defined).

Rather, the issue is "similarity"; e.g. if $\mathcal{D}$ is a set of tables, then $\widetilde{\mathcal{D}}$ must reflect relationships among the columns and rows of $\mathcal{D}$. Depending on the application, one could define similarity in terms of statistical correlation, graph properties (e.g. if $\mathcal{D}$ 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 $\widetilde{\mathcal{D}}$.

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 Animoto[2]), 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

*This research was supported in part by MOE Grant No. R-252-000-394-112.

[1] `http://www.tpc.org/`

[2] `http://animoto.com`

Proceedings of the VLDB Endowment, Vol. 4, No. 12

(a) R'

A	B
a_1	1
a_2	1
a_3	2
a'_1	3
a'_2	3
a'_3	4

(a) T'

B	C
1	c_1
2	c_2
2	c_3
3	c'_1
4	c'_2
4	c'_3

R

A	B
a_1	1
a_2	1
a_3	2

T

B	C
1	c_1
2	c_2
2	c_3

(b) R'

A	B
a_1	1
a_2	1
a_3	2
a'_1	1
a'_2	1
a'_3	2

(b) T'

B	C
1	c_1
2	c_2
2	c_3
1	c'_1
2	c'_2
2	c'_3

Figure 1: $\mathcal{D} = \{R, T\}$, $\widetilde{\mathcal{D}} = \{R', T'\}$, $s = 2$. Naive copying does not work: For (a), creating new values may violate constraints on B (e.g. value range or number of distinct values); for (b), without creating new values, the scale up in join sizes may be wrong.

a cloud), and the vendor needs to scale up the sample to an appropriate size.

Taking a small sample of a large dataset is itself nontrivial. For example, if a dataset contains 2000000 buyers, and we want to extract a sample with 1000 buyers, it does not suffice 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.

An enterprise may want to downsize its dataset, not just for a vendor, but for itself. For example, rather than debug a new application by running it on a production dataset, one may want to downsize it ($s < 1$) to get a small synthetic copy for testing.

In providing a vendor with just a small sample of its dataset, the dataset owner may be motivated by privacy or proprietary considerations. A tool to make a synthetic copy ($s = 1$) of a dataset can hence be viewed as a form of anonymization.

Such anonymization can be useful for, say, exploring different system configurations or implementations in the cloud (leveraging on its elasticity, and before investing in a particular configuration). Rather than expose their real dataset (i.e. their crown jewels), an enterprise can reduce their risk by uploading a synthetic copy.

One reason for the popularity of TPC benchmarks among academic researchers is the dearth of real application data. This scarcity may be alleviated by a tool for making synthetic copies. Note, however, that some information leakage is inevitable since $\widetilde{\mathcal{D}}$ is, after all, similar to $\mathcal{D}$.

4. ATTRIBUTE VALUE CORRELATION PROBLEM

To see why scaling a dataset is nontrivial, consider the toy relational $\mathcal{D}$ in Fig. 1. An obvious possibility for $s = 2$ is to scale $\mathcal{D}$ to $\widetilde{\mathcal{D}}$ by making a copy. However, if the copy uses new values not in $\mathcal{D}$ (choice (a) in Fig. 1), this may violate attribute constraints like value range or number of distinct values; moreover, if $\mathcal{D}$ 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 $\mathcal{D}$ (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 $\mathcal{D}$ is a relational dataset (from a service like Flickr[3]) with four tables `User`, `Photo`, `Comment` and `Tag` that records photographs uploaded by, commented upon and tagged by a community of users. The foreign key constraints are as indicated by the schema graph in Fig. 2. These constraints must hold in $\widetilde{\mathcal{D}}$, and we can view such referential integrity as a form of correlation between attributes.

Classical dependency theory is a form of attribute correlation, and there is recent work on discovering, say, conditional functional dependencies [10]. However, there are many other forms of inter-attribute correlation. For example, the non-key columns for age and gender in the table `User` may be correlated; a user is more likely to comment on her own photographs, so there is correlation between foreign keys `CUid` in table `Comment` and `PUid` in table `Photo` for `CPid`=`Pid`; a gardener is more likely to comment on the photograph of a flower, so the foreign keys `CUid` and `CPid` in `Comment` are correlated; etc.

Aside from inter-column correlations, there are also inter-row correlations. For instance, the dimensions for different photographs uploaded by a user may be similar, and the tags used by a gardener may be recognizably different from those used by a bird watcher.

Some correlations are both inter-column and inter-row. For example, two friends x and y may comment on each other's photographs, so the attribute values in `Photo` and `Comment` have the correlation shown in Fig. 3. This illustrates a correlation that is induced by a social interaction.

In general, how can we scale a dataset $\mathcal{D}$ that is generated by a social network $\mathcal{G}$? Intuitively, we would need to also construct a synthetic social network $\widetilde{\mathcal{G}}$ when generating $\widetilde{\mathcal{D}}$. The graph $\mathcal{G}$ of edges between users may not be explicitly stored in $\mathcal{D}$, but one can extract $\mathcal{G}$ from $\mathcal{D}$ by some join expression (like in the case of Fig. 3).

Scaling $\mathcal{G}$ to $\widetilde{\mathcal{G}}$ is nontrivial: How should the number of edges scale — by s, s^2, or some other factor? How about

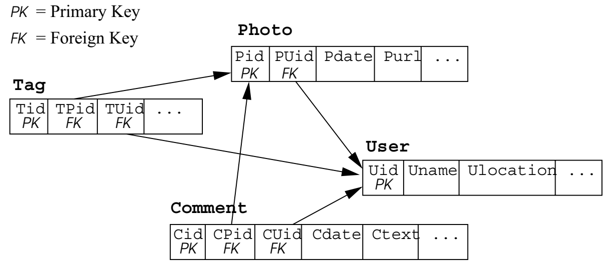

Figure 2: A small schema graph for a photograph database $\mathcal{D}$. `Photo` records the owners (`PUid`) who uploaded the photographs, `Comment` records the comments on photographs (`CPid`) and their authors (`CUid`), and `Tag` records the tags on photographs (`TPid`) and the users who specified the tags (`TUid`). `User` records these owners, authors and taggers.

Photo

Pid	PUid	...
Px	*x*	
Py	*y*	

Comment

Cid	CPid	CUid	...
	Px	*y*	
	Py	*x*	

Figure 3: Users x and y comment on each other's photographs. Such interactions induce inter-column and inter-row correlations in the tables above.

[3]http://www.flickr.com

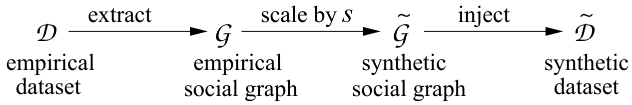

Figure 4: The social graph $\mathcal{G}$ can be extracted from $\mathcal{D}$ with some join expression and $\mathcal{G}$ can be scaled to $\widetilde{\mathcal{G}}$ by some graph-theoretic algorithm; but how to create the dataset $\widetilde{\mathcal{D}}$ for $\widetilde{\mathcal{G}}$?

the number of triangles (a friend of a friend is likely to be a friend)? What about the path lengths (6 degrees of separation)? These issues are graph-theoretic.

From the database perspective, the real difficulty comes *after* extracting $\mathcal{G}$ from $\mathcal{D}$ and scaling $\mathcal{G}$ to $\widetilde{\mathcal{G}}$: namely, how to "inject" $\widetilde{\mathcal{G}}$ into $\widetilde{\mathcal{D}}$ (see Fig. 4). For example, in Fig. 3, the same four attributes reflect correlation
(i) from $\mathcal{G}$ induced by x and y commenting on each other's photographs,
(ii) between gardeners and photographs of flowers and
(iii) between commentator and owner;
in what order should the scaling algorithm replicate these correlations? How about correlation between these attributes and other columns?

We see that, while the extraction of $\mathcal{G}$ from $\mathcal{D}$ can be done with a straightforward join query, the injection of $\widetilde{\mathcal{G}}$ into $\widetilde{\mathcal{D}}$ requires an understanding of how attribute values are correlated in social network data. We state this issue as **The Attribute Value Correlation Problem for Social Networks (AVC$_{\mathbf{SN}}$)**

> *Suppose a database $\mathcal{D}$ records data from a social network. How do the social interactions affect the correlation among attribute values in $\mathcal{D}$?*

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[4] 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$_{\mathrm{SN}}$ 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 [1, 6]. However, Seltzer et al. [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 [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 [17] only extracts names and addresses from a real dataset; TEXTURE [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) [8].

Similarly, the data generating tool by Houkjær et al. [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[5] and HP's Desensitizer [5] are focused on data extraction and obfuscation, not synthetic data generation.

Bruno and Chaudhuri's Data Generation Language [4] can specify value distributions and generate data tuples, while Hoag and Thompson's Synthetic Data Description Language [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 [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 [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 [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 [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.

[4] `http://www.facebook.com/`

[5] `http://www-01.ibm.com/software/data/data-management/optim-solutions/`

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_{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[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_{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_{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.

[6] `http://www.comp.nus.edu.sg/~upsizer`

8. REFERENCES

[1] C. Binnig, D. Kossmann, T. Kraska, and S. Loesing. How is the weather tomorrow?: Towards a benchmark for the cloud. In *DBTest*, pages 1–6, 2009.

[2] C. Binnig, D. Kossmann, and E. Lo. Reverse query processing. In *ICDE*, pages 506–515, 2007.

[3] C. Binnig, D. Kossmann, E. Lo, and M. T. Özsu. QAGen: Generating query-aware test databases. In *SIGMOD*, pages 341–352, 2007.

[4] N. Bruno and S. Chaudhuri. Flexible database generators. In *VLDB*, pages 1097–1107, 2005.

[5] M. Castellanos, B. Zhang, I. Jimenez, P. Ruiz, M. Durazo, U. Dayal, and L. Jow. Data desensitization of customer data for use in optimizer performance experiments. In *ICDE*, pages 1081–1092, 2010.

[6] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with YCSB. In *SoCC*, pages 143–154, 2010.

[7] D. J. DeWitt. The Wisconsin Benchmark: Past, present, and future. In *The Benchmark Handbook*. Morgan Kaufmann Publishers Inc., 1993.

[8] S. Duan, A. Kementsietsidis, K. Srinivas, and O. Udrea. Apples and oranges: A comparison of RDF benchmarks and real RDF datasets. In *SIGMOD*, 2011.

[9] V. Ercegovac, D. J. DeWitt, and R. Ramakrishnan. The TEXTURE benchmark: Measuring performance of text queries on a relational DBMS. In *VLDB*, pages 313–324, 2005.

[10] W. Fan, F. Geerts, L. V. S. Lakshmanan, and M. Xiong. Discovering conditional functional dependencies. In *ICDE*, pages 1231–1234, 2009.

[11] J. E. Hoag and C. W. Thompson. A parallel general-purpose synthetic data generator. *SIGMOD Rec.*, 36(1):19–24, 2007.

[12] K. Houkjær, K. Torp, and R. Wind. Simple and realistic data generation. In *VLDB*, pages 1243–1246, 2006.

[13] I. F. Ilyas, V. Markl, P. Haas, P. Brown, and A. Aboulnaga. CORDS: Automatic discovery of correlations and soft functional dependencies. In *SIGMOD*, pages 647–658, 2004.

[14] H. Kimura, G. Huo, A. Rasin, S. Madden, and S. Zdonik. CORADD: Correlation aware database designer for materialized views and indexes. In *Proc. VLDB Endow.*, volume 3, pages 1103–1113, 2010.

[15] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1, March 2007.

[16] M. Seltzer, D. Krinsky, K. Smith, and X. Zhang. The case for application-specific benchmarking. In *HOTOS*, pages 102–109, 1999.

[17] J. M. Stephens and M. Poess. MUDD: A multi-dimensional data generator. In *WOSP*, pages 104–109, 2004.

[18] M. Stonebraker. A new direction for TPC? In *TPCTC*, pages 11–17, 2009.

[19] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao. User interactions in social networks and their implications. In *Eurosys*, pages 205–218, 2009.