Wisconsin Benchmark Data Generator: To JSON and Beyond

Shiva Jahangiri
University of California Irvine
shivaj@uci.edu

ACM Reference format:
Shiva Jahangiri. 2021. Wisconsin Benchmark Data Generator: To JSON and Beyond. In *Proceedings of 2021 ACM SIGMOD International Conference on Management of Data (SIGMOD'21), June 20-25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3448016.3450577

Problem and Motivation

Benchmarks have always been one of the greatest assets in evaluating a Data Management System (DBMS) performance and providing a standard way to compare different DBMSs from various angles. As DBMSs evolve over time, new benchmarks are created, and the older ones need to advance and adapt in order to continue to be a valid and capable comparison tool. Many of the standard benchmarks use synthetic data and as such may not be able to capture the complexity of real data and its relationships correctly. The Wisconsin Benchmark was one of the first and main benchmarking tools made four decades ago by Dewitt et al at the University of Wisconsin. One of the most powerful features of this benchmark is that its relations are designed so their structure and distribution of attributes is easy to understand and control. While the Wisconsin Benchmark was a very powerful and widely-used benchmark years ago, it is given less attention in current studies. We believe that the Wisconsin Benchmark and its carefully designed relations can be utilized and provide capabilities that are unique and useful.. In this paper, we present a flexible, easy-to-use and scalable JSON Data Generator implemented in java based on the Wisconsin Benchmark Data Generator description, with more advanced features to provide relations and attributes closer to real-world data. Attribute skewness and variable length records using different size distributions are some of these newly added features. It is a ready-to-use, parameterized, and scalable data generator tool that since its development has been used in several AsterixDB's [4] performance benchmarking and publications [13,14,17,18], and we believe that can be useful to many others. The source code and more information are provided at [1].

SIGMOD '21, June 20–25, 2021, Virtual Event, China.

ACM ISBN 978-1-4503-8343-1/21/06.
https://doi.org/10.1145/3448016.3450577

Attribute Name	Range	Order	Comment
unique1	0..(MAX-1)	random	unique, random order
unique2	0..(MAX-1)	random	unique, sequential
two	0..1	cyclic	(unique1 mod 2)
four	0..3	cyclic	(unique1 mod 4)
ten	0..9	cyclic	(unique1 mod 10)
twenty	0..19	cyclic	(unique1 mod 20
onePercent	0..99	cyclic	(unique1 mod 100)
tenPercent	0..9	cyclic	(unique1 mod 10)
twentyPercent	0..4	cyclic	(unique1 mod 5)
fiftyPercent	0..1	cyclic	(unique1 mod 2)
unique3	0..(MAX-1)	cyclic	unique1
evenOnePercent	0,2,4,...,198	cyclic	(onePercent * 2)
oddOnePercent	1,3,5,...199	cyclic	(onePercent * 2)+1
stringu1		random	candidate key
stringu2		cyclic	candidate key
string4		cyclic	

Figure 1: The original Wisconsin benchmark schema

Background and Related Work

In 1980s, the lack of a standard database benchmark initiated efforts by multiple researchers to design benchmarks for different purposes and applications. DeWitt et al introduced the Wisconsin Benchmark [6,9] at the completion of the DIRECT database machine; the benchmark was designed to test the performance of the major components of a relational database system while the semantics and statistics of its relations would be easy to understand. This Benchmark was initially a single-user micro-benchmark for measuring individual query performance, which was criticized mostly for being single-user only [9]. A multi-user version of this benchmark [6] which was developed a few years later did not attract as much as attraction as past, due to existence of other well-established competitors such as DebitCredit Benchmark, a multi-user OLTP benchmark led by Jim Gray [16]. While the Wisconsin Benchmark does not have as large an audience today as in the past, we believe that its simple but powerful design easy to understand and highly controllable relations and attributes are unique and can be beneficial in evaluating database systems from many angles. As such, we have built a Wisconsin-inspired JSON data generator in java with more add-ons and advances features to support more modern data features.

Currently, the TPC benchmark family is one of the well-known standard benchmark sets widely used in academia as well as industry. While TPC-H and TPC-DS are powerful benchmarks with complex workloads, their attributes are mostly uniformly distributed and independent of each other. [7] introduced join crossing correlation with skew in TPC-H benchmark and [8] introduces skew in TPC-H by grouping nations into high and low populated.

Approach and Novelty

One of the challenges of the benchmarks that use synthetic data is that they are then incapable of generating data that is realistic. On the other hand, benchmarks which use real data often contain data values that are not flexible and controllable which makes them harder to understand and less capable of providing a range of specific scenarios. To overcome these problems, we created a Wisconsin-inspired JSON Data Generator in Java which creates records with the same logic and attributes as the Wisconsin Data Generator. In addition to that, we added other features such as attribute distributions to provide attribute skewness which is a missing but very important and useful feature to have. A number of other features were added to support semi-structured in addition to structured databases; those are explained below.

JSON Records. Our data generator provides records in JSON format which is supported as the input format or even as the data model in many of modern database systems, especially those that manage semi-structured data. Apache AsterixDB [2] is an example that we have benchmarked using data generated by our data generator.

Nullable & Missing Attributes. One of the features that was missing from the original Wisconsin Data Generator was the capability to have nullable attributes and have a knob to control the distribution of null values in those attributes. Also, in a semi-structured world, a field may appear in some of the records but not in all. Some database management systems such as AsterixDB that manage semi-structured data, have the option to define a field as optional which may exist in some of the records and be missing in others. In our generator, for each attribute there are options for setting a field as nullable and/or missing. In addition to that, user can specify what percentage of the data they wish to be null and/or missing.

Variable Record Lengths. In the original Wisconsin Data Generator, strings are generated in a random or cyclic format. In the case of random, the string representation of the unique1 attribute is used as the prefix (which is unique as well) and enough 'x' characters will be padded to the string to reach the desired length. In the case of cyclic, string values are generated from the domain of four prefix in a cyclic format. While padding to the strings is a simple and useful way of reaching the required length of the string, it does not create variation in strings' length. In order to generate variable length strings, we added 5 more properties to the attribute definition. The first property is a percentage which based on the binomial distribution decides if the string should be long or short. The other four fields are used for specifying the minimum and maximum length of the long and short strings. The length of the long and short string will be chosen uniformly from these ranges. These knobs give us control on the distribution of short and long strings as well as their length ranges. We also support selecting the length of a string from a defined range using Zipf distribution.

Real-Word & HEX Strings. In addition to supporting the mentioned algorithms, we support strings that are generated by concatenating words from a list made of 10,000 real words. This approach helps with reducing the impact of data compression due to less repetitive characters. For this case, an average number of words to be included in the string is provided as an input, and algorithm concatenates as many as asked from the word list based on a specific distribution (Uniform, Normal, Gamma, and Zipf distributions are supported). Generating random HEX strings is another way of generating variable length strings with lower impact due to compression.

Attribute Skewness. One of the missing but important feature in a number of current benchmark's data generators is the ability to provide attribute skewness, yet data skew is unavoidable in real world. For our purpose, to benchmark the performance of AsterixDB's join algorithms under join attribute skewness, we applied normal distribution on integer attributes. Users can specify the standard deviation and the mean of their distribution to get the desired dataset.

Contributions & Future Work

The JSON Wisconsin Data Generator has been used in multiple studies and publications so far. [14] used this data generator to benchmark the performance of memory-intensive operators in AsterixDB. They generated variable sized and large records using random HEX string generator under Normal and Gamma distributions. [17,18] have used Json Wisconsin Data Generator to generate large volumes of highly controllable data to benchmark their large-scale analytical frameworks. The author in [13] used this data generator for re-evaluating a study [19] on various query plans for multi-join queries. The author is currently utilizing the advanced features of this data generator to benchmark the performance of Dynamic Hybrid Hash Join in AsterixDB under variable record sizes, skewed join attributes, and a combination of both. We will work on generating nested array-valued fields in one relation with regard to the values in other relations to provide data for benchmarking join queries on nested data.

ACKNOWLEDGMENTS

I would like to thank my PhD advisors Michael J. Carey and Johann-Christoph Freytag for their continued help and support of this work.

REFERENCES

[1] JSON Wisconsin Data Generator: Shiva Jahangiri. (2020, December 11). shivajah/JSON-Wisconsin-Data-Generator: First Release (Version v1.0.2). Zenodo. http://doi.org/10.5281/zenodo.4316003

[2] https://asterixdb.apache.org

[3] 1993. TPC Benchmark TMA: Standard Specification. In The Benchmark Handbook for Database and Transaction Systems (2nd Edition), Jim Gray (Ed.). Morgan Kaufmann.

[4] Sattam Alsubaiee, Yasser Altowim, Hotham Altwaijry, Alexander Behm,Vinayak R. Borkar, Yingyi Bu, Michael J. Carey, Inci Cetindil, MadhusudanCheelangi, Khurram Faraaz, Eugenia Gabrielova, Raman Grover, Zachary Heil-bron, Young-Seok Kim, Chen Li, Guangqiang Li, Ji Mahn Ok, Nicola Onose,Pouria Pirzadeh, Vassilis J. Tsotras, Rares Vernica, Jian Wen, and Till Westmann.2014. AsterixDB: A Scalable, Open Source

BDMS.CoRRabs/1407.0454(2014).arXiv:1407.0454 http://arxiv.org/abs/1407.0454

[5] Carrie Ballinger. 1993. TPC-D: Benchmarking for Decision Support. In The Benchmark Handbook for Database and Transaction Systems (2nd Edition), JimGray (Ed.). Morgan Kaufmann.

[6] Dina Bitton, David J. DeWitt, and Carolyn Turbyfill. 1983. Benchmarking Database Systems A Systematic Approach. In9th International Conference onVery Large Data Bases, October 31 - November 2, 1983, Florence, Italy, Proceedings, Mario Schkolnick and Costantino Thanos (Eds.). Morgan Kaufmann, 8–19.http://www.vldb.org/conf/1983/P008.PDF

[7] Peter A. Boncz, Angelos-Christos G. Anadiotis, and Steffen Kläbe. 2017. JCC-H: Adding Join Crossing Correlations with Skew to TPC-H. In Performance Evaluation and Benchmarking for the Analytics Era - 9th TPC Technology Conference,TPCTC 2017, Munich, Germany, August 28, 2017, Revised Selected Papers (Lecture Notes in Computer Science), Raghunath Nambiar and Meikel Poess (Eds.),Vol. 10661. Springer, 103–119. https://doi.org/10.1007/978-3-319-72401-0_8

[8] Alain Crolotte and Ahmad Ghazal. 2011. Introducing Skew into the TPC-H Benchmark. In Topics in Performance Evaluation, Measurement and Characterization - Third TPC Technology Conference, TPCTC 2011, Seattle, WA, USA, August29-September 3, 2011, Revised Selected Papers (Lecture Notes in Computer Science),Raghunath Othayoth Nambiar and Meikel Poess (Eds.), Vol. 7144. Springer, 137–145. https://doi.org/10.1007/978-3-642-32627-1_10

[9] David J. DeWitt. 1991. The Wisconsin Benchmark: Past, Present, and Future. In The Benchmark Handbook for Database and Transaction Systems (1st Edition), JimGray (Ed.). Morgan Kaufmann, 119–165.

[10] Jim Gray (Ed.). 1993.The Benchmark Handbook for Database and Transaction Systems (2nd Edition). Morgan Kaufmann.

[11] Jim Gray. 1993. Database and Transaction Processing Performance Handbook. In The Benchmark Handbook for Database and Transaction Systems (2nd Edition),Jim Gray (Ed.). Morgan Kaufmann.

[12] Rui Han, Lizy Kurian John, and Jianfeng Zhan. 2018. Benchmarking Big Data Systems: A Review.IEEE Trans. Serv. Comput.11, 3 (2018), 580–597. https://doi.org/10.1109/TSC.2017.2730882

[13] Shiva Jahangiri. 2020. Re-evaluating the Performance Trade-offs for Hash-Based Multi-Join Queries. In Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020, David Maier, Rachel Pottinger, An Hai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 2845–2847.https://doi.org/10.1145/3318464.3384406

[14] Taewoo Kim, Alexander Behm, Michael Blow, Vinayak R. Borkar, Yingyi Bu,Michael J. Carey, Murtadha Al Hubail, Shiva Jahangiri, Jianfeng Jia, Chen Li,Chen Luo, Ian Maxon, and Pouria Pirzadeh. 2020. Robust and efficient memory management in Apache AsterixDB.Softw. Pract. Exp.50, 7 (2020), 1114–1151.https://doi.org/10.1002/spe.2799

[15] Francois Raab. 1993. TPC-C - The Standard Benchmark for Online transaction Processing (OLTP). In The Benchmark Handbook for Database and Transaction Systems (2nd Edition), Jim Gray (Ed.). Morgan Kaufmann.

[16] Omri Serlin. 1993. The History of DebitCredit and the TPC. In The Benchmark Handbook for Database and Transaction Systems (2nd Edition), Jim Gray (Ed.).Morgan Kaufmann.

[17] Phanwadee Sinthong and Michael J. Carey. 2019. AFrame: Extending DataFramesfor Large-Scale Modern Data Analysis. In2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, December 9-12, 2019. IEEE, 359–371.https://doi.org/10.1109/BigData47090.2019.9006303

[18] Phanwadee Sinthong and Michael J. Carey. 2020. PolyFrame: A RetargetableQuery-based Approach to Scaling DataFrames (Extended Version).CoRRabs/2010.05529 (2020). arXiv:2010.05529 https://arxiv.org/abs/2010.05529

[19] Donovan A. Schneider and David J. DeWitt. (1990). Tradeoffs in Processing Complex Join Queries via Hashing in Multiprocessor Database Machines. In Proceedings of the 16th International Conference on Very Large Data Bases (VLDB '90). San Francisco, CA, USA, 469-480.