MMSBench-Net: Scenario-Based Evaluation of Multi-Model Database Systems

David Lengweiler[1], Marco Vogt[1] and Heiko Schuldt[1]

[1]*University of Basel, Department of Mathematics and Computer Science, Spiegelgasse 1, 4051 Basel, Switzerland*

Abstract

Multi-model database systems have gained increasing popularity due to their efficient management of diverse types of data and support for complex queries. They offer a unified approach for managing data in various formats, including structured, semi-structured, and unstructured data. However, benchmarking the performance of such systems is a challenging task, given their complexity, mainly due to their support for multiple data models. While significant research exists for benchmarking single-model databases, a comprehensive approach for evaluating multi-model databases is still in an early stage. To address this challenge, we propose MMSBench-Net, a benchmark for evaluating multi-model database systems that support structured relational, semi-structured document, and graph data models. MMSBench-Net enables comparative analysis of database systems and demonstrates how different workloads can reveal the strengths and weaknesses of multi-model database systems. To demonstrate the effectiveness of the benchmark, we compare the performance of two database systems: Polypheny and SurrealDB. Our work is a first step towards a comprehensive evaluation methodology for multi-model database systems.

Keywords

Database benchmark, Polystore, Multi-model database

1. INTRODUCTION

The field of data management has experienced a significant transformation in recent years. While relational databases continue to dominate the market, more specialized systems have emerged. Two data models that have gained substantial popularity are the graph and the document model. These data models allow data to be represented and queried unknown from the relational model [1]. However, these new data models are by no means an evolution of the relational model. As a result, use cases that could be modeled optimally with the relational model might only be modeled poorly with the graph or the document model. As a result, database management systems supporting multiple data models have gained popularity. These multi-model database systems allow applications to manage their data in a way that best suits the specific domains, but also introduce greater complexity. While there are well-established benchmarks like TPC-C [2], TPC-H [3] and YCSB [4] for single-model databases, the set of benchmarks targeting multi-model databases is very limited. Existing benchmarks for multi-model databases often focus on specific data models, which restricts the range of systems that can be evaluated. Moreover, these benchmarks typically involve complex scenarios that lack fine-grained workload adjustments, limiting their usefulness for detailed evaluations and only allowing for broad comparisons.

This paper makes two contributions: Firstly, we propose a benchmark called MMSBench-Net that is tailored to benchmarking multi-model database systems. It is based on a real-world scenario that deals with relational, document and graph data. Secondly, we demonstrate the utility of our benchmark by comparing the performance of two multi-model database systems, Polypheny[1] and SurrealDB[2] and discuss the results.

The remainder of this paper is structured as follows: In Section 2, we introduce the MMSBench-Net, discuss the underlying scenario and present the data, and workload that is being generated. In Section 3, we then briefly introduce the two multi-model database systems subject to the benchmark evaluation presented in this paper. Section 4 then presents and discusses the obtained results. The paper concludes with an overview of related work in Section 5, an outlook towards future work in Section 6 and a conclusion in Section 7.

2. BENCHMARK

To evaluate the performance of multi-model database systems, we propose MMSBench-Net, a benchmark that assesses their ability to manage structured relational, semi-structured document, and graph data models. MMSBench-Net is designed to evaluate the efficiency and versatility of multi-model database systems under different

34th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken)

david.lengweiler@unibas.ch (D. Lengweiler); marco.vogt@unibas.ch (M. Vogt); heiko.schuldt@unibas.ch (H. Schuldt)

0009-0004-0588-8210 (D. Lengweiler); 0000-0002-2674-2219 (M. Vogt); 0000-0001-9865-6371 (H. Schuldt)

[1]https://polypheny.com/

[2]https://surrealdb.com/

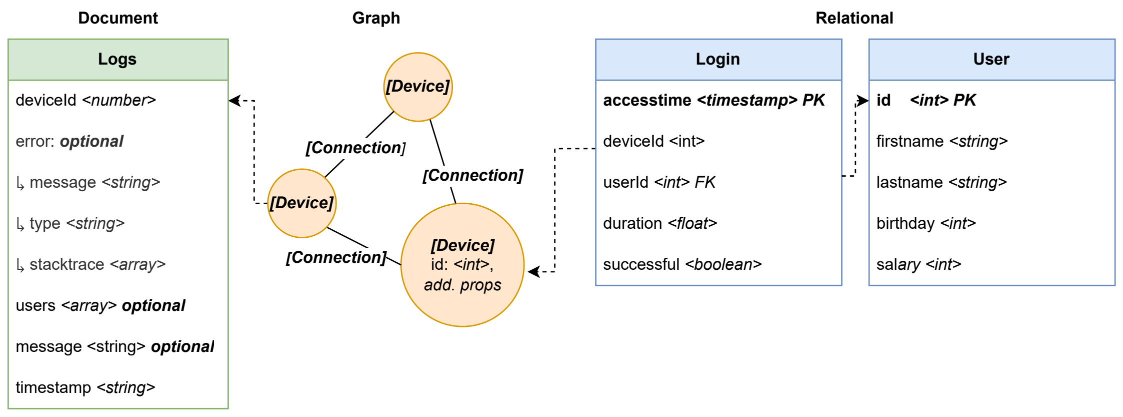

Figure 1: Multi-Model Schema of MMSBench-Net

workloads. The “-Net” suffix refers to the first scenario introduced in this paper. We plan to add more scenarios (and thus suffixes) in the future, leading to a complete suite.

The MMSBench-Net benchmark consists of a set of queries that reflect real-world use cases across the three data models. These queries are designed to evaluate various aspects of multi-model database systems, including their ability to handle complex data structures, support complex queries, and efficiently execute transactions.

2.1. Scenario

MMSBench-Net is inspired by a real-world scenario of a company’s network monitoring application. Network monitoring plays a vital role in identifying and addressing potential issues, threats and vulnerabilities in the network infrastructure, ensuring smooth operations and preventing data breaches or downtime. The monitoring application continuously collects all kinds of information about the network, including logged-in devices, usage statistics and log messages, resulting in huge amounts of heterogeneous data.

The network monitoring application modeled by MMSBench-Net maintains data in three data models, (i) a graph part, modeling the topology of the network, (ii) a document part, which consists of semi-structured logs produces by the devices and (iii) a relational part, which holds basic information about the users and recorded data about their access patterns. The complete schema is depicted in Figure 1.

The topology of the network is saved as a graph, where each device in the network is represented as a node, and a network connection between two devices is modeled as an edge. For both, the devices modeled as nodes and the connection modeled as edges, additional information is being stored, such as, the “manufacturer” of a device, its “purchase year” and other relevant information.

```
deviceId: 3,
timstamp: "2017-07-23:14-03",
error: {
    message: "Out of Memory",
    type: "Application Error",
    stacktrace: ["Error on start of..."]
},
user:{
    id: 34,
    status: "logged in"
},
users: [34, 45]
```

Figure 2: Example Status Log Showing an Error

In irregular intervals, each device produces a semi-structured log-entry containing information about its current state. An example of such a log can be seen in Figure 2. These logs might contain error information, indicating problems with the device. All log entries include the properties *deviceId*, *timestamp* and *users*. However, additional properties with varying levels of nesting are randomly generated for each log entry.

An important information for monitoring a network is which person is currently associated with which devices. For this scenario, we assume a rather simple user database represented as a relational table containing information on the employee. Furthermore, there is also a table for recording successful and failed login attempts and for accounting the usage of devices. Hence, this scenario necessitates the database system to deal with heterogeneous read and write workloads.

2.2. Schema and Data Generation

To generate the schema and to populate it with realistic, but artificially created data, MMSBench-Net starts with building a simulation. This simulation includes the graph representing the network that is being monitored, as well as the users interacting with it. The nodes in the graph represent *devices* (e.g., computers, mobile phones, switches, and routers). The edges between these nodes represent network *connections* between these devices. The simulation utilizes the defined topology to generate meaningful workloads. By making changes to this topology, it becomes possible to adjust the distribution of available targets for the queries. This enables to easily align with specific requirements and desired focus of a workload.

The process of generating this simulated network consists of multiple steps:

User Generation First, a configurable number of users is being generated.

Generation of Devices For each type of device (e.g., switches, computers), a random number (within a configurable range) of devices is being generated.

Device Properties and Logs Generation For each device, a random set of properties is being generated. Furthermore, a set of login, as well as status logs is added as well.

Generation of Connections According to the layout of the network, multiple pairs of devices are selected and connections between them are created.

Connection Properties Generation In contrast to the devices, connections do not have status logs, but they also create multiple dynamic properties.

After the generation of the network is done, it is used as a template to create the workload. Each workload consists of queries in a query language supported by the system under evaluation.

A distinction is made between the three data models. First, the graph data is handled as already seen in Figure 1. For this, each *device* is represented as a node and each *connection* is translated to an edge which connects them. The small set of dynamic properties is inserted directly as part of these nodes and edges (if properties are not supported by the data model, these are handled as if they are unstructured data). Then all generated device logs (an example can be seen in Figure 2), are translated to document queries. Each entity of type device translates its nested status logs into multiple document queries, each containing a timestamp and the ID of the device.

Finally, all login records are collected from the devices and together with the user data itself are translated into relational queries. The collected queries are then sequentially executed on the database systems.

2.3. Workload Generation

A workload consists of a collection of randomly chosen queries according to a configurable distribution. Since the order in which queries are executed can impact the performance of a database system (e.g., due to concurrency effects and locking), the implementation needs to make sure that the workload is identical for all systems under evaluation (e.g., by using the same seed). MMS-Bench-Netuses a variety of queries to build its workloads:

Read Device or Connection Selects a device or connection and retrieves it partially or fully. One of the static parameters is chosen for this.

Read Log Selects a device and reads all or parts of its logs. Filters as well as projections of underlying keys are chosen from the target device.

Remove Device Selects a device and deletes it, also all connections to this device are deleted as well. Logs are deleted as well, information on log-in attempts are kept.

Remove Connection Randomly selects a connection between two network devices and deletes it.

Add Device Adds a device to the network. Generate new connections to existing devices.

Remove Logs Randomly selects a device and deletes some of its logs.

Add Logs Creates a random log message and adds it to existing devices or connections.

Add User Creates a new user. All attributes are randomly generated.

Remove User Randomly selects a user who is being deleted.

Change User Randomly selects a user and adjusts an attribute.

Besides simple queries, there are also more complex retrieval operations which can be chosen, their frequency is also configurable.

Connectivity Checks *"Find all similar connected devices"* or *"Find connected device of specific type"*

Error Analysis *"Identify the top 10 most common errors"* or *"Calculate the percentage of errors caused by each user"*

Login Activity *"Successful logins by user and month"* or *"Average duration of successful logins by user and hour of the day"*

Firstly, the actions are selected and implemented on the simulated network, while concurrently being captured and converted into queries for the evaluated systems. Once the simulation concludes, the gathered queries are distributed across a configurable number of available threads and executed on the evaluated system. The execution time for each query is measured individually and recorded for subsequent analysis. This facilitates a comprehensive analysis of various aspects of the database systems. Each iteration of this workload generation and execution process is referred to as a *cycle*; in an evaluation, multiple cycles can be chained together to construct more extensive workloads.

3. EVALUATED SYSTEMS

To showcase the capabilities of MMSBench-Net, two multi-model databases have been chosen to be evaluated: Polypheny and SurrealDB. These two systems have been selected since they follow completely opposite approaches for implementing multiple data models beneath one facade. While Polypheny maintains the individual models independently, SurrealDB follows a more monolithic approach by combining all data models in one unified model.

3.1. Polypheny

Polypheny [5, 6] is a PolyDBMS [7], which is a multi-model database system built according to the architecture principle of a polystore and supporting multiple query languages. Data can be represented according to the relational, the document and the labeled-property graph data models. Polypheny utilizes multiple highly optimized database systems like HypherSQL[3], MongoDB, Neo4j, and PostgreSQL as storage and execution engines. To achieve competitive performance, Polypheny utilizes these underlying data stores to push down queries. Queries not supported by the underlying data store are executed within Polypheny itself. Polypheny also provides support for transactions with ACID guarantees.

3.2. SurrealDB

SurrealDB is a multi-model database management system that provides traditional database guarantees, such as ACID transactions, persistent data storage, and fine-grained data access control. Its primary objective is to provide fast performance while adhering to these guarantees. It also supports unstructured data and basic graph functionality, which makes it a suitable choice for this comparison. SurrealDB was designed with the goal of reducing the number of joins required for retrieval queries. It accomplishes this objective by utilizing a graph structure that allows a tuple to any other tuple. SurrealQL, a SQL-like query language, is the primary means of interacting with the system, which can be accessed through either a REST or a web socket interface.

[3]https://hsqldb.org/

4. EVALUATION

Our evaluation uses Chronos [8], an 'evaluation-as-a-service' framework which allows to easily execute different system evaluations and configurations in parallel. To achieve this, it manages a collection of nodes, which are used to execute these different evaluation configurations. The evaluation machines used for obtaining the results presented in this paper are equipped with an Intel Xeon X5650 24-core CPU with 24 GiB of RAM. All machines run Ubuntu 22.04 LTS (with kernel version 5.15.0-37) and the same patch level. As Java runtime environment, we use OpenJDK version 17.0.3. The presented numbers are the median over three runs.

Each run uses either a SurrealDB instance in a Docker[4] container, deployed from scratch and configured to use a persistent on-file configuration, or a fresh Polypheny instance. The Polypheny instance uses a MongoDB[5] store for the document data, a Neo4j[6] store for the graph data, and a PostgreSQL[7] store for the relational data. Each of these stores is deployed by Polypheny using Docker containers, this requires less setup than bare-metal deployments and achieves similar performance [9]. Both Polypheny and SurrealDB have indexes on their primary keys. We provide a reference implementation of the benchmark, including all configurations and the raw results[8].

As a first overview comparison, the default configuration of the benchmark, simulating a network with 10 users and around 65 devices, is being used. All scaling parameters are configured to only allow for a slight growth of the network. The different runtimes after multiple cycles of workloads can be seen in Figure 3.

With such a small network and thus a low number of queries, SurrealDB manages to execute the workloads faster than Polypheny, even when the amount of queries increases. If one observes the results grouped by the query model, Polypheny is faster than SurrealDB for the relational queries, this can be seen in Figure 4.

[4]https://www.docker.com/
[5]https://www.mongodb.com/
[6]https://neo4j.com/
[7]https://www.postgresql.org/
[8]https://download-dbis.dmi.unibas.ch/paper/GvDB23.zip

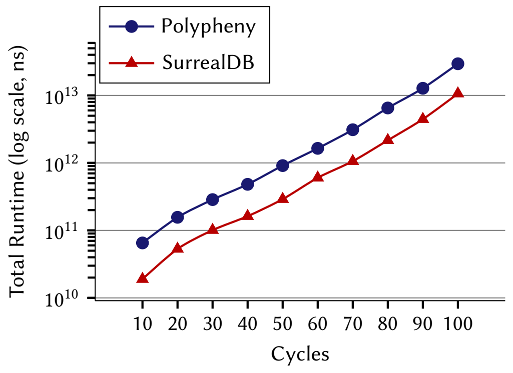

Figure 3: Runtime with Increasing Number of Cycles

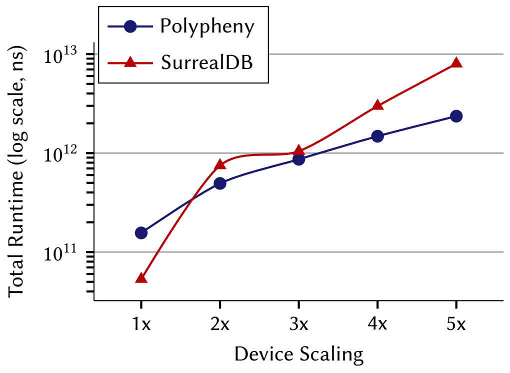

Figure 5: Runtime with Increasing Number of Devices

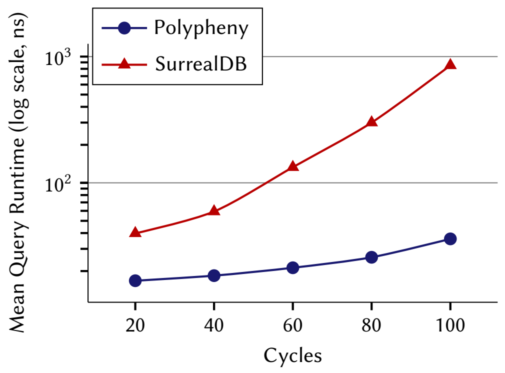

Figure 4: Mean Relational Query Runtime with Increasing Number of Cycles

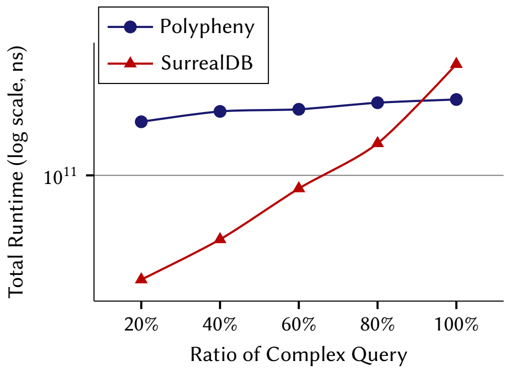

Figure 6: Runtime for Increasing Ratio of Complex Queries

However, in most real-world scenarios, the network starts with a significantly higher number than the 10 users used by default. Thus, the number of users is being adjusted, leading to a higher number of user logins and therefore more relational workload. With an increasing ratio of relational workload, Polypheny is able to perform similar to SurrealDB. This behavior is similar if the number of devices in the network is increased. While this does not increase the ratio of the relational workload, compared to the other data models, it still results in better overall performance of Polypheny, which is depicted in Figure 5. Figure 6 depicts a comparison of different ratios of complex queries in the workload.

The results obtained from the evaluation of the two quite different systems confirms the concepts of the MMSBench-Net benchmark, in particular that it is agnostic to the concrete database under evaluation and has a wide applicability for the evaluation of single- and multi-model database systems in realistic settings.

5. RELATED WORK

One of the first prominent benchmarks for evaluating database management systems was the Wisconsin [10] benchmark, introduced in 1983. The space of multi-model database evaluation, in contrast, has a rather short history. One of the first ones being BigBench [11], introduced by TPC as TPCx-BB. BigBench uses a schema, which combines structured, semi-structured and unstructured data. But beside TPC, there has been an increase in work, which provides similar benchmarks to the one proposed in this paper. In [12], a benchmark using key-value, column, document and graph data is used to compare ArangoDB[9] and OrientDB[10] against a combination of single-model databases, using a proposed synthetic generated benchmark. They were able to show, that depending on the scenario, multi-model databases can be faster than configurations combining multiple single-model database systems. UniBench [13] targets the same data models as MMSBench-Net, but also considers key-value and XML

[9]https://www.arangodb.com/
[10]https://orientdb.org/

data. It puts great effort in modeling an as realistic as possible social-commerce scenario. M2Bench [14] relies heavily on existing benchmark datasets and extends the used data models of UniBench by introducing the array model into its evaluation.

6. FUTURE WORK

Our goal for MMSBench-Net is to extend it into a benchmarking suite that offers various real-world usage scenarios for multi-model data management. However, there are some limitations with MMSBench-Net that we need to address in the future.

First, the minimal set of queries that we have chosen for evaluation may not be representative of all possible ways to query multi-model systems. In future evaluations, we should include a more diverse set of queries to reflect the range of possibilities when querying these systems. This would provide more comprehensive results and strengthen the obtained conclusions.

Second, the current composition of workloads is too broad and general to allow for nuanced comparisons of multi-model systems. We need to create more fine-grained workloads that focus on specific aspects of data models to capture the subtle differences between these systems.

In addition to the limitations of the benchmark, our evaluation only compared two systems, leaving a lot of unexplored territory. Future evaluations should include additional systems such as ArangoDB and OrientDB to gain more insights into their performance. Although not all multi-model databases support the same data models, it is possible to use parts of unsupported data models or substitute them with other models to expand the range of systems that can be evaluated.

Lastly, we should consider evaluating configurations that use a combination of multiple single-model databases to facilitate interesting comparisons. By addressing these limitations, we can develop a more comprehensive and nuanced benchmarking suite that offers a more accurate evaluation of multi-model systems.

7. CONCLUSION

In this paper, we introduced the MMSBench-Net, a new benchmarked superficially tailored to benchmark multi-model database systems that is based on the scenario of a network monitoring application. Our evaluation of Polypheny and SurrealDB demonstrates the effectiveness and applicability of the proposed benchmark.

Our research represents an important first step towards establishing a comprehensive evaluation methodology for multi-model database systems. The proposed benchmark allows for a fair comparison of different systems, and our results provide insights into the performance of Polypheny and SurrealDB under different workloads. Ultimately, this benchmark will guide the development and evaluation of novel multi-model database systems.

Acknowledgments

This work was partly supported by the SNSF (“Polypheny-DDI: A Flexible Polystore-based Distributed Data Infrastructure”, grant no. 200020_213121). The authors would like to thank R. Arnold, R. Gasser, S. Heller, L. Sauter, F. Spiess and A. Mbilinyi for their valuable feedback.

References

[1] E. F. Codd, A relational model of data for large shared data banks, Communications of the ACM 13 (1970) 377–387. doi:10/dwxst4.

[2] T. P. P. Council, TPC benchmark c revision 5.11, 2010. URL: https://tpc.org/tpc_documents_current_versions/pdf/tpc-c_v5.11.0.pdf.

[3] T. P. P. Council, TPC benchmark h standard revision 3.0.1, 2022. URL: https://tpc.org/tpc_documents_current_versions/pdf/tpc-h_v3.0.1.pdf.

[4] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, R. Sears, Benchmarking cloud serving systems with YCSB, in: Proc. SoCC'10, ACM Press, 2010, pp. 143–154. doi:10/cxjrfd.

[5] M. Vogt, Adaptive Management of Multimodel Data and Heterogeneous Workloads, Ph.D. thesis, University of Basel, 2022. doi:10/j44k.

[6] M. Vogt, N. Hansen, J. Schönholz, D. Lengweiler, I. Geissmann, S. Philipp, A. Stiemer, H. Schuldt, Polypheny-DB: Towards bridging the gap between polystores and HTAP systems, in: Proc. Poly'21, LNCS, Springer, 2020, pp. 25–36. doi:10/gnxv2h.

[7] M. Vogt, D. Lengweiler, I. Geissmann, N. Hansen, M. Hennemann, C. Mendelin, S. Philipp, H. Schuldt, Polystore systems and DBMSs: Love marriage or marriage of convenience?, in: Proc. Poly'21, volume 12921 of *LNCS*, Springer, 2021, pp. 65–69. doi:10/gn8qvm.

[8] M. Vogt, A. Stiemer, S. Coray, H. Schuldt, Chronos: The swiss army knife for database evaluations, in: Proc. EDBT'20, OpenProceedings.org, 2020, pp. 583–586. doi:10/g8w5.

[9] W. Felter, A. Ferreira, R. Rajamony, J. Rubio, An updated performance comparison of virtual machines and Linux containers, in: Proc. ISPASS'15, 2015, pp. 171–172. doi:10/gfvg6d.

[10] H. Boral, D. J. DeWitt, A methodology for database

system performance evaluation, in: Proc. SIGMOD'84, ACM, 1984, pp. 176–185. doi:10/fk5fbn.

[11] C. Baru, M. Bhandarkar, C. Curino, et al., Discussion of BigBench: A Proposed Industry Standard Performance Benchmark for Big Data, in: Performance Characterization and Benchmarking. Traditional to Big Data, Springer, 2015, pp. 44–63. doi:10/j44q.

[12] F. R. Oliveira, L. del Val Cura, Performance Evaluation of NoSQL Multi-Model Data Stores in Polyglot Persistence Applications, in: Proc. IDEAS'16, ACM, 2016, pp. 230–235. doi:10/j44n.

[13] C. Zhang, J. Lu, P. Xu, Y. Chen, UniBench: A Benchmark for Multi-model Database Management Systems, in: Performance Evaluation and Benchmarking for the Era of Artificial Intelligence, Springer, 2019, pp. 7–23. doi:10/j44m.

[14] B. Kim, K. Koo, U. Enkhbat, S. Kim, J. Kim, B. Moon, M2Bench: A Database Benchmark for Multi-Model Analytic Workloads, Proceedings of the VLDB Endowment 16 (2022) 747–759. doi:10/j44p.