# UniBench: A Benchmark for Multi-Model Database Management Systems

Chao Zhang, Jiaheng Lu, Pengfei Xu, and Yuxing Chen
Department of Computer Science
University of Helsinki, Finland

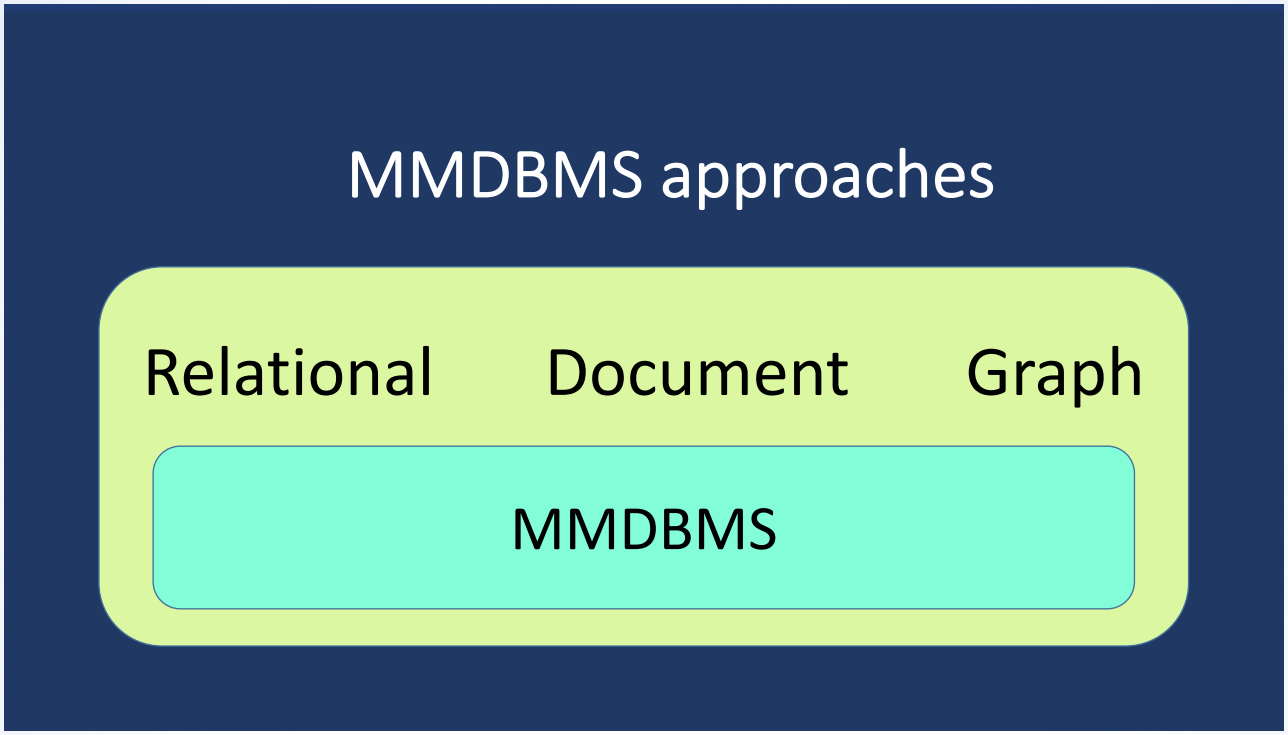
HELSINGIN YLIOPISTO

# Outline

- **Background & Motivation**
- Contributions of UniBench
- Data model and Data Generation
- Workloads
- Experimental Results
- Conclusion & Future work

HELSINGIN YLIOPISTO

# Multi-Model DBMSs




A unified platform to manage multi-model data

with flexibility, scalability and consistency.

# Multi-Model DBMSs

Table 1: Comparison of Multi-Model DBMSs

| System | Query Language | Primary Model | Secondary Model | Storage Strategy |
|---|---|---|---|---|
| **AgensGraph** | OpenCypher, SQL | Relational | Graph, JSON | One Engine |
| **ArangoDB** | AQL | JSON | Graph, Key-value | One Engine |
| **OrientDB** | SQL-like | Graph | JSON, Key-value | One Engine |
| **Marklogic** | Xpath | XML | JSON, RDF | One Engine |
| **Redis** | API | Key-value | Graph, JSON | One Engine |
| **NitrosBase** | SparQL, SQL | RDF | Graph, JSON, Key-value | One Engine |
| **Datastax** | CQL | Column | JSON, Graph | Multiple Engines |
| **DynamoDB** | API, SQL | - | JSON, Graph, Key-value | Multiple Engines |
| **CosmosDB** | API, SQL | - | **ALL** but XML | Multiple Engines |
| **Oracle 12c** | SQL-extension | Relational | **ALL** | **Both** |

HELSINGIN YLIOPISTO

# Motivation

- Crucial to investigate their pros and cons
- Previous benchmarks lack a comprehensive consideration for multi-model data.

HELSINGIN YLIOPISTO

# Outline

- Background & Motivation
- **Contributions of UniBench**
- Data model and Data Generation
- Workloads
- Experimental Results
- Conclusion & Future work

HELSINGIN YLIOPISTO

# Related work

- LDBC SNB : Choke-point design
- BigBench : Big data analytical queries
- TPC-C: ACID read-write transaction

HELSINGIN YLIOPISTO

# Contributions of UniBench

- Scope: Multi-model databases
- Underlying model: Social commerce
- Workload: Multi-model queries and transactions

HELSINGIN YLIOPISTO

# Outline

- Background & Motivation
- Contributions of UniBench
- **Data model and Data Generation**
- Workloads
- Experimental Results
- Conclusion & Future work

HELSINGIN YLIOPISTO

# Data model

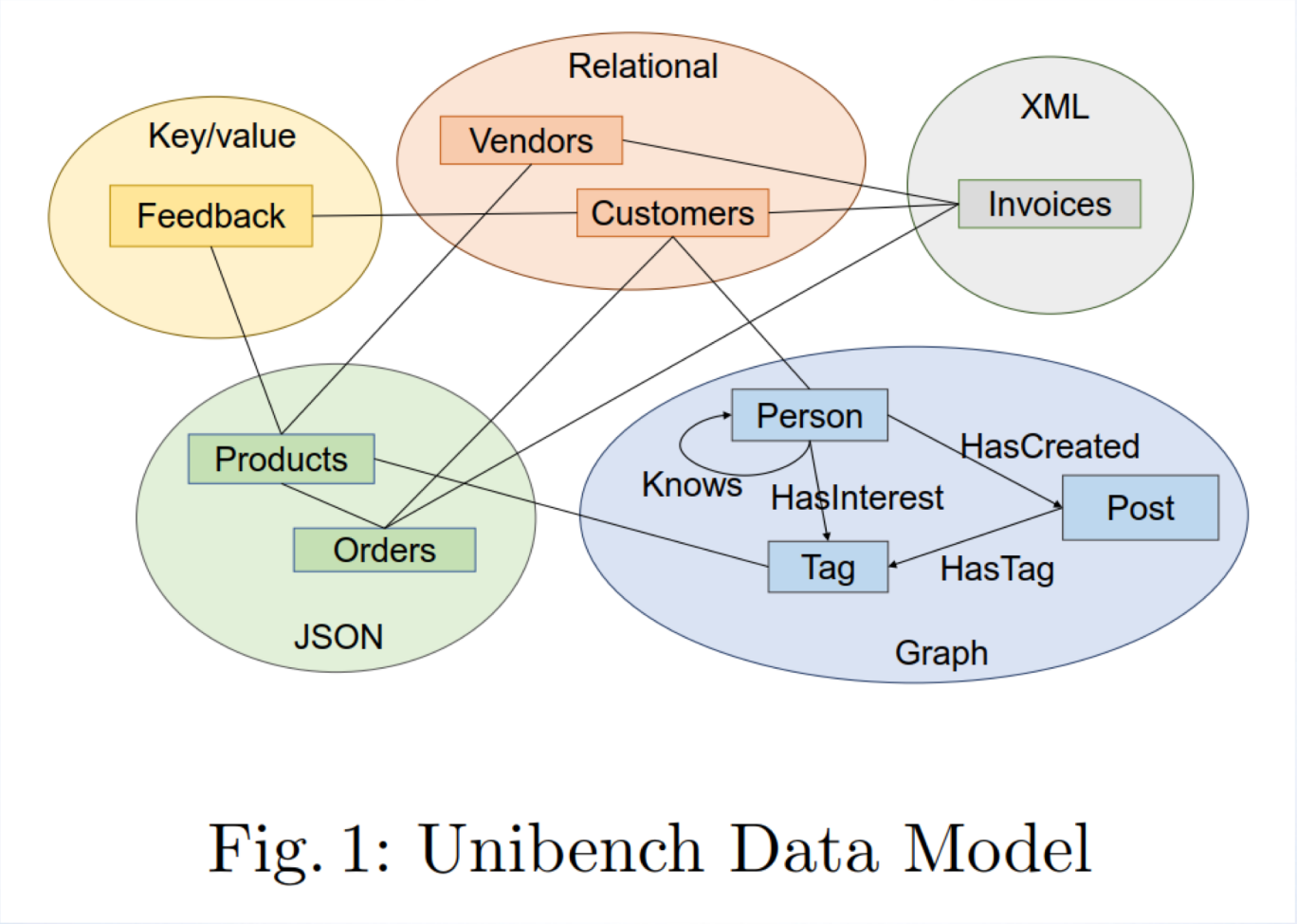



Fig. 1: Unibench Data Model

# Data Generation

## Three-phase framework

1. Purchase : interest-oriented transaction
2. Pro-purchase: friend-influenced transaction
3. Re-purchase: probabilistic transaction

## Scalable and Skew

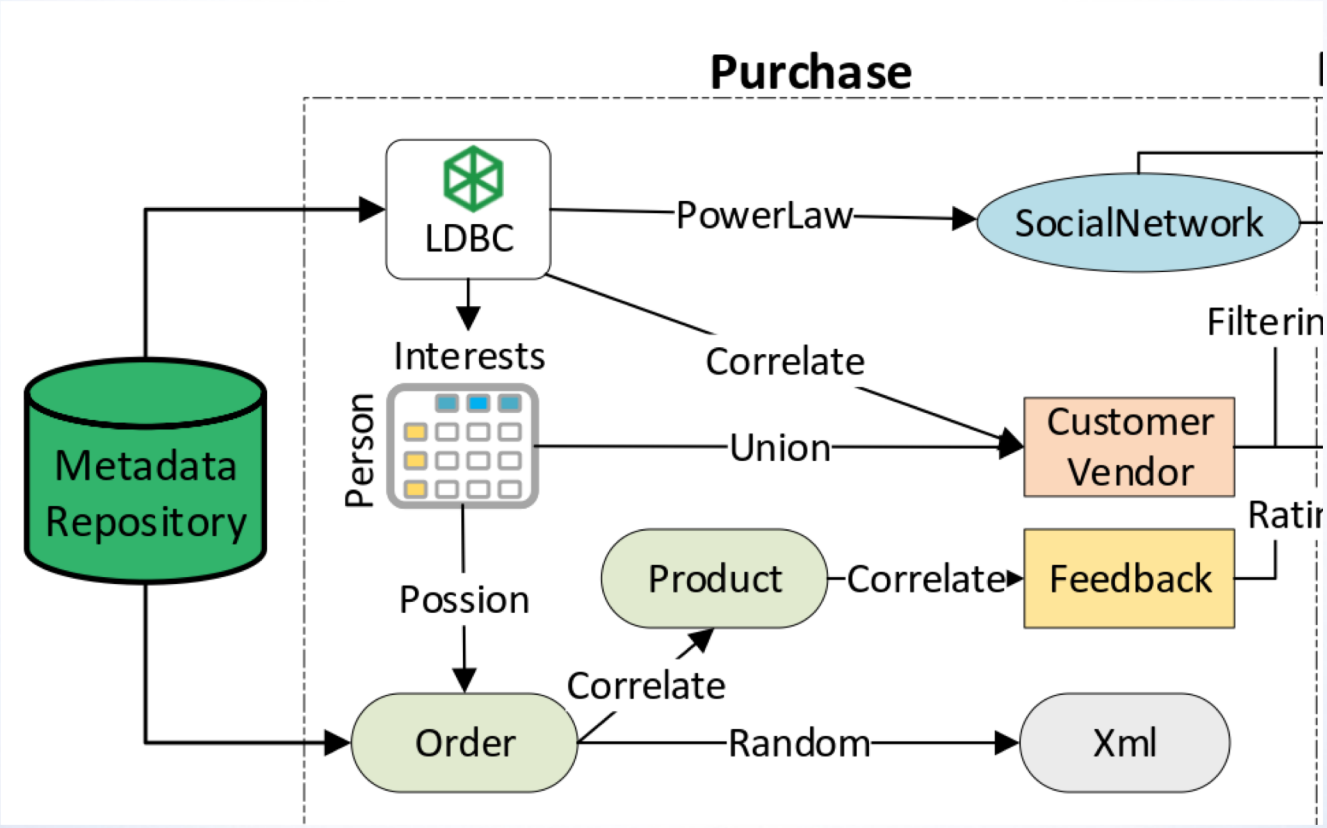
Implemented on Spark, linear with scale factor

Power-law distribution

HELSINGIN YLIOPISTO

# Three-phase framework

Purchase : interest-oriented transaction



# Three-phase framework

Pro-purchase: friend-influenced transaction

$$S_{ui} = \sum_k k \times Pr(R_{ui} = k | A = a_u) + E(R_{vi} : \forall v \in N(u))$$

HELSINGIN YLIOPISTO

# Three-phase framework

Re-purchase: probabilistic transaction

$$S_{ib}(CLVSC) = E(X^* \mid n^*, x', n, m, \alpha, \beta, \gamma, \delta) \times (E(M \mid p, q, v, m_x, x) + E(S \mid \bar{s}, \theta, \tau))$$

HELSINGIN YLIOPISTO

# Outline

- Background & Motivation
- Contributions  of UniBench
- Data model and Data Generation
- **Workloads**
- Experimental Results
- Conclusion & Future work

HELSINGIN YLIOPISTO

# Workload:

4 business layers including 10 queries, 2 transactions

Table 2: Characteristics of Workload

| Label | Business category | Technique dimension | Description |
|---|---|---|---|
| Q1 | Individual | Perform point query on a customer's all multi-model data. | For a given *customer*, find her **profile**, **orders**, **feedback**, and **posts**. |
| Q2 | Conversation | Join data from Relation, Graph, and JSON. | For a given *product*, find the **persons** who had bought it and posted on it. |
| Q3 | Conversation | Join data from Relation, Graph, and Key-value, filter structured and unstructured data. | For a given *product*, find **persons** who have commented and posted on it, and detect negative **sentiments** from them. |
| Q4 | Community | Aggregate and sort the JSON order, Perform the 3-hop graph traversal in the subgraph, return the intersection of two sets. | Find the top-2 **persons** who spend the highest amount of money in orders. Then for each person, traverse her knows-graph with 3-hop to find the friends, and finally return the common **friends** of these two persons. |
| Q5 | Community | Join data from Relation, Graph, and Key-value with two predicates, recursive path query for Graph, embedded array operation for JSON, and composited-key lookup for Key-value. | Given a start *customer* and a product *category*, find **persons** who are this customer's friends within 3-hop friendships in knows-graph, and they have bought products in the given category. Finally, return **feedback** with the 5-rating review of those bought products. |
| Q6 | Community | Perform the shortest path calculations between two nodes, find the correlated JSON orders of nodes in the path, aggregation on returned JSON orders. | Given *customer 1* and *customer 2*, find persons in the **shortest path** between them in the subgraph, and return the TOP 3 **best sellers** from all these persons' purchases. |
| Q7 | Commerce | Join data from Relation, JSON and Key-value, compare the aggregation results between two periods, identify the reviews with negative sentiment. | For the *products* of a given *vendor* with declining sales compare to the former quarter, analyze the **reviews** for these items to see if there are any negative sentiments. |
| Q8 | Commerce | Perform the embedded array filtering and aggregation on JSON order, aggregate the correlated graph data for each records. | For all the *products* of a given *category* during a given year, compute its **total sales amount**, and measure its **popularity** in the social media. |
| Q9 | Commerce | Perform the embedded array filtering, aggregation, and sorting on JSON order, then find the correlated graph data. | Find top-3 **companies** who have the largest amount of sales at one *country*, for each company, compare the number of the male and female customers, and return the most recent **posts** of them. |
| Q10 | Commerce | Perform the aggregation and sort on graph data, then find the correlated Key-value and JSON data. | Find the top-10 most active **persons** by aggregating the *posts* during the last year, then calculate their **RFM (Recency, Frequency, Monetary) value** in the same period, and return their recent **reviews** and tags of **interest** |
| T1 | New Order Transaction | Check the ACID properties and evaluate the efficiency on read-heavy multi-model transaction that involves JSON and XML. | (i) create and insert the **order**, (ii) update the quantity of involved **products**, (iii) insert the **invoice**. |
| T2 | Payment Transaction | Check the ACID properties and evaluate the efficiency on write-heavy multi-model transaction that involves Relation, JSON and XML. | (i) retrieve the unpaid **order**, (ii) update the balance of the **seller** and **buyer**, (iii) update the **order** status to paid, (iv) update the related **invoice**. |

# Business Cases

1. Individual level, Q1
2. Conversation level, Q2 and Q3
3. Community level, Q4, Q5, Q6
4. Commerce level, Q7, Q8, Q9, Q10

HELSINGIN YLIOPISTO

# Multi-model transaction

New Order.

(i) create and insert the order,
(ii) update the quantity of involved products,
(iii) insert the invoice.

Payment.

(i) retrieve the unpaid order,
(ii) update the balance of the seller and buyer,
(iii) update the order status to paid,
(iv) update the related invoice.

HELSINGIN YLIOPISTO

# Choke-point design

1. Choose the right join type and order
2. Perform complex aggregation
3. Ensure the consistency and efficiency

HELSINGIN YLIOPISTO

# Choke-point design

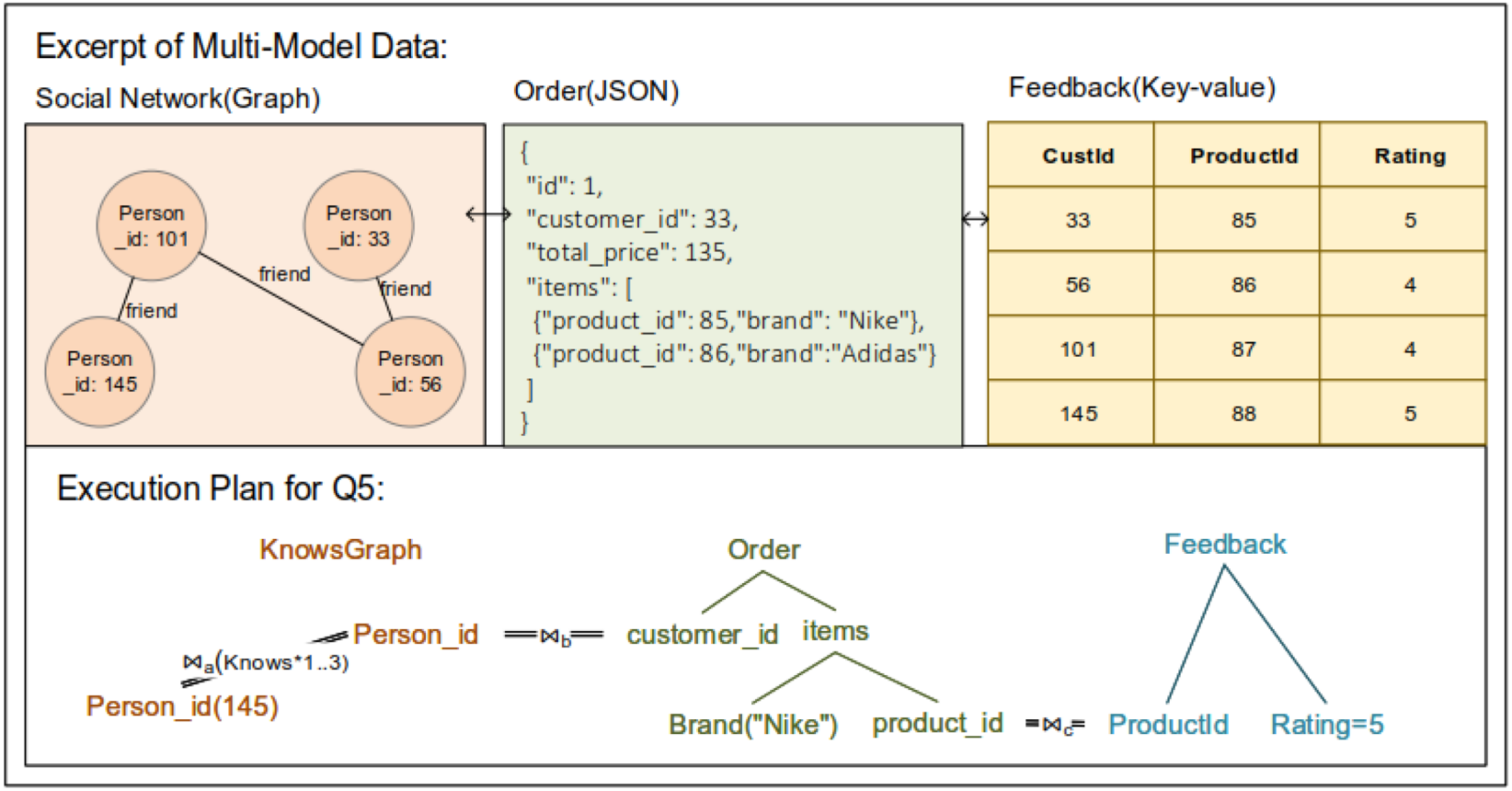



Fig. 3: Example of multi-model join

# Outline

- Background & Motivation
- Contributions  of UniBench
- Data model and Data Generation
- Workloads
- **Experimental Results**
- Conclusion & Future work

HELSINGIN YLIOPISTO

# Experimental Setup

Data generation: on a cluster of three machines, each with double 4-core Xeon-E5540 CPU, 32GB RAM, and 500GB HDD

Benchmarking: a 6-core Xeon-E5649 CPU, 100GB RAM, and 500GB HDD server. The client machine has a 4-core i5-4590 CPU with 16GB RAM. ArangoDB and OrientDB A Node.js driver.

HELSINGIN YLIOPISTO

# Experimental Results: Data generation

Table 3: Characteristics of datasets.

| SF | Generation Time(min) | Number ($\times 10^4$) & Size in Megabytes | | | | |
|---|---|---|---|---|---|---|
| | | Relational entries | Key-value pairs | JSON objects | XML objects | Nodes and Edges of Graph |
| 1 | 10 | 1.2 & 1.1 | 25.2& 233.7 | 25.2& 219.2 | 25.2& 326.5 | (123.1, 338.9) & 236.6 |
| 10 | 40 | 7.4 & 6.5 | 234.2 & 2313.1 | 234.2& 2189.8 | 234.2& 3568.6 | (969.3, 3208.3) & 2095.8 |
| 30 | 60 (3 nodes) | 18.3 & 15.8 | 636.8& 6367.8 | 636.8& 6184.9 | 636.8& 11771.31 | (2674.3, 10951.5) & 6191.5 |

Data generation is at scale for all five data models

HELSINGIN YLIOPISTO

# Experimental Results: Data importing

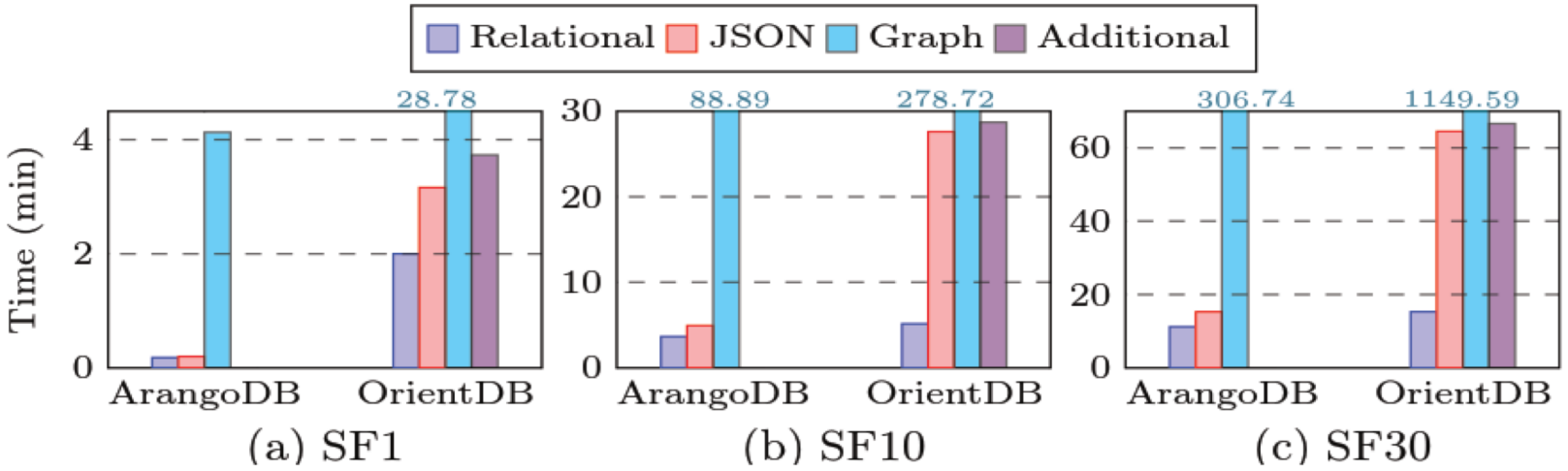



Fig. 4: Processing time for importing the multi-model datasets.

ArangoDB outperforms OrientDB in data importing

HELSINGIN YLIOPISTO

# Experimental Results: multi-model query

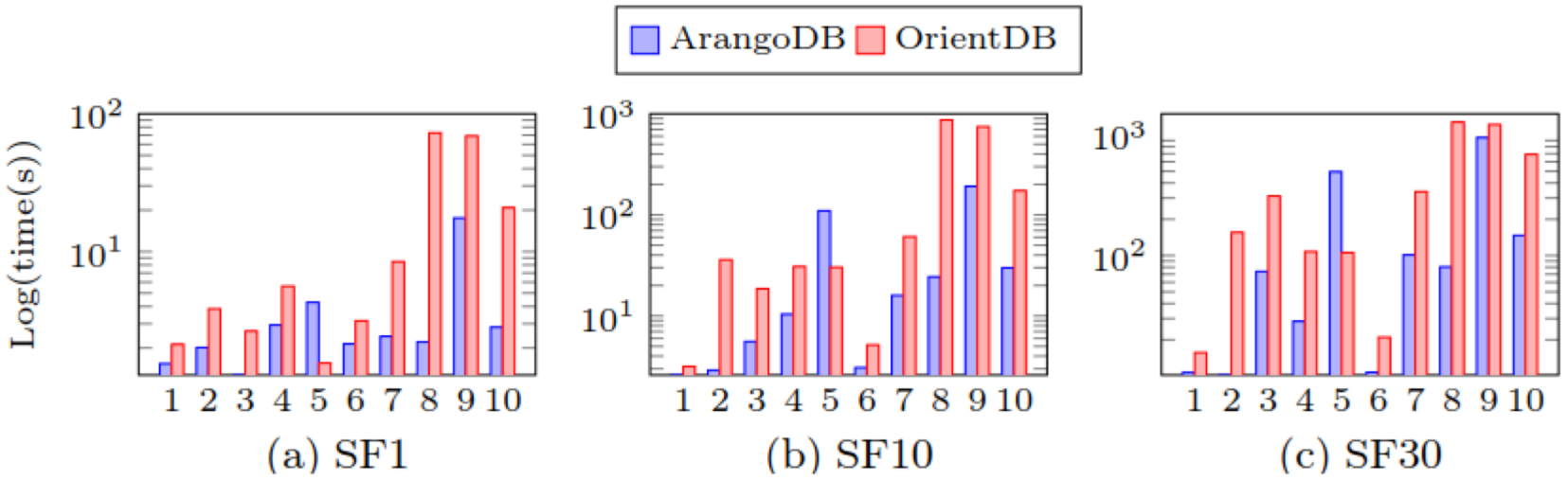



Fig. 5: Processing time on a logarithmic scale for queries, x-axis labels are query ids, i.e., Q1 to Q10.

ArangoDB outperforms OrientDB in most cases

HELSINGIN YLIOPISTO

# Experimental Results: multi-model transaction

Table 4: Throughput (transactions/second) of Multi-model transactions

| Database | Access method | Throughput for New Order | Throughput for Payment |
|---|---|---|---|
| **ArangoDB** | Asynchronous (Nodejs) | 230.6 | 738.5 |
| **OrientDB** | Synchronous (Java) | 138.3 | 22.9 |

ArangoDB excels in write-intensive transaction,
OrientDB excels in read-intensive transaction

HELSINGIN YLIOPISTO

# Take Aways

(i) MMDBs can ingest a variety of data into storage efficiently, such as table, json, graph.

(ii) MMDBs can support multi-model joins. e.g., Graph-JSON, JSON-relation.

(iii) MMDBs can support multi-model ACID-compliant transactions in the standlone mode.

HELSINGIN YLIOPISTO

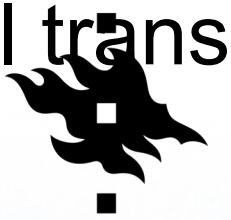
# Outline

- Background & Motivation
- Contributions  of UniBench
- Data model and Data Generation
- Workloads
- Expertimental Results
- **Conclusion & Future work**

HELSINGIN YLIOPISTO

# Conclusion

- UniBench: the first benchmark for multi-model databases
- Social commerce model
  - JSON, graph, xml, relational, key-value
- Worloads
  - 10 multi-model queries and
  - 2 multi-model transactions



HELSINGIN YLIOPISTO

# Future work

- Incorparate the flexibility for data generation
  - vary data schema and data model
- Evaluate performance in distributed clusters
- Novel Metrics

HELSINGIN YLIOPISTO

# Thanks for your attention!

# Question?

Link to UniBench: http://udbms.cs.helsinki.fi/bench/

Link to our contact: jiaheng.lu@helsinki.fi

HELSINGIN YLIOPISTO