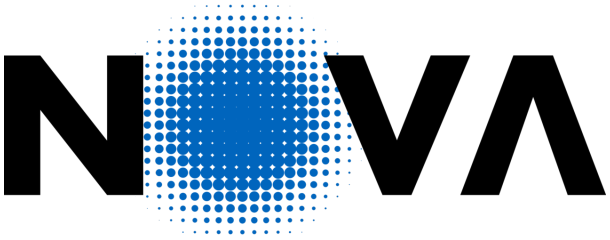

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

NOVA UNIVERSITY OF LISBON

MSC IN COMPUTER SCIENCE

PERFORMANCE COMPARISON BETWEEN DBMSS UNDER TPROC-C WORKLOADS

José Costa (62637)
Rodrigo Albuquerque (70294)
Rodrigo Silva (70567)

DATABASES SYSTEMS

JUNE 4, 2025

Contents

1 Introduction

This project aims to benchmark and compare the performance of different database systems using the TPC-C workload, a standard for evaluating OLTP (Online Transaction Processing) environments.

Automated scripts are used to run identical tests across multiple databases, each configured with similar settings to ensure a fair comparison.

Key metrics such as Transactions Per Minute (TPM) are measured under varying levels of concurrent users.

The results help identify the strengths and limitations of each database in handling transactional workloads.

In this test, the databases used were PostgreSQL, MySQL and MariaDB.

In total, we ran 54 tests:

- 4 tests scaling the number of virtual users (2 4 8 12) and warehouses (VU*5) on all PCs and databases (48 tests in total);
- 1 test with the number of virtual users set to the same as the number of threads in that PC, warehouses set to VU*5, allwarehouse = true on all databases on just one PC (3 test in total);
- 1 test with the number of virtual users set to the same as the number of threads in that PC, warehouses set to VU*5, with default config on all databases on just one PC (3 test in total).

2 Overview of HammerDB

HammerDB is a free, open-source tool for benchmarking the performance of relational databases [1].

It supports popular databases like Oracle, SQL Server, PostgreSQL, MySQL, and more. HammerDB uses industry-standard workloads such as TPROC-C and TPROC-H to simulate real-world database activity.

It offers both a graphical interface and command-line options, making it suitable for developers, DBAs, and system administrators to test, compare, and tune database performance.

In some cases we used HammerDB in docker containers to run the tests, which allows for easy setup and isolation of the testing environment.

In another case, we used the Windows version of HammerDB to run the tests on a Windows machine.

2.1 Overview of TPROC-C

TPROC-C is a benchmark designed to evaluate the performance of database management systems (DBMS) using a transactional workload. It simulates a typical online transaction processing (OLTP) environment, focusing on operations like inserts, updates, and deletes across multiple tables.

2.2 TPROC-C vs TPROC-H

TPROC-H is a benchmark designed for data warehousing and analytical workloads, while TPROC-C is focused on transactional processing. TPROC-H emphasizes complex queries and large data sets, whereas TPROC-C simulates real-time transactions with a focus on insert, update, and delete operations.

3 Problem & DBMS Summary

4 Benchmark Description

5 Methodology

5.1 Hardware and Software Setup

PC	1	2	3	4
OS	Windows 11	Windows 11	Linux (Unraid)	MacOS Sequoia
CPU	Intel i7-13700H	AMD Ryzen 5 3600	Intel i3-10100F	Apple M1
Cores	14 (6P 8E)	6	4	8
Threads	20	12	8	8
RAM	16GB	16GB	32GB	16GB
Disk	SSD M.2 NVMe	SSD M.2 NVMe	SSD M.2 NVMe	SSD M.2 NVMe
Read	3500 MB/s	2500 MB/s	3500 MB/s	3400 MB/s
Write	2700 MB/s	2100 MB/s	3300 MB/s	2800 MB/s

Table 1: Hardware used in the benchmarks

In PCs **1**, **3**, and **4**, we used HammerDB and all the databases in docker containers to run the tests, which allows for easy setup and isolation of the testing environment.

We used docker compose for this setup, which allowed us to easily run the same tests on different machines with the same configuration.

In PC **2**, we used everything installed on the host machine.

5.2 Database Setup

For MySQL and MariaDB, this was the configuration used:

```
[mysqld]

# BASIC SETTINGS
port = 3308
bind-address = 0.0.0.0
max_connections = 1000
max_connect_errors = 1000000
wait_timeout = 28800
interactive_timeout = 28800
connect_timeout = 10
back_log = 1500

# CHARACTER SET & COLLATION
character-set-server = utf8mb4
collation-server = utf8mb4_unicode_ci

# STORAGE AND INNODB ENGINE
innodb_buffer_pool_size = 4G
innodb_buffer_pool_instances = 4
innodb_log_file_size = 512M
innodb_log_buffer_size = 64M
innodb_file_per_table = 1
innodb_flush_log_at_trx_commit = 2
innodb_flush_method = O_DIRECT
innodb_io_capacity = 2000
innodb_io_capacity_max = 4000
innodb_read_io_threads = 8
innodb_write_io_threads = 8
innodb_purge_threads = 4
innodb_doublewrite = 1
innodb_autoinc_lock_mode = 2
innodb_stats_persistent = 1
innodb_lru_scan_depth = 2048
innodb_adaptive_flushing = 1
innodb_adaptive_hash_index = 0
innodb_change_buffering = none

# TEMPORARY TABLE & BUFFERS
tmp_table_size = 256M
max_heap_table_size = 256M
sort_buffer_size = 1M
join_buffer_size = 1M
read_buffer_size = 512K
```

```
read_rnd_buffer_size = 2M

# LOGGING
slow_query_log = 1
long_query_time = 2
slow_query_log_file = /var/lib/mysql/mysql-slow.log
general_log = 0

# BYNARY LOGGING
skip-log-bin
sync_binlog = 0

# SECURITY & COMPATIBILITY
local_infile = 0
sql_mode = STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION
transaction_isolation = REPEATABLE-READ

# PERFORMANCE
performance_schema = OFF
max_prepared_stmt_count = 12800
table_open_cache = 2000
table_open_cache_instances = 4
open_files_limit = 65535
thread_cache_size = 50
thread_stack = 256K

# MONITORING
innodb_monitor_enable = '%'
```

For PostgreSQL, this was the configuration used:

```
# FILE LOCATIONS
data_directory = '/var/lib/postgresql/data' # Important: match Docker volume
hba_file = '/var/lib/postgresql/data/pg_hba.conf'
ident_file = '/var/lib/postgresql/data/pg_ident.conf'

# CONNECTIONS AND AUTHENTICATION
listen_addresses = '*' # Allow external connections (Docker host network)
port = 5432 # Default PostgreSQL port
max_connections = 100

# RESOURCE USAGE
shared_buffers = 512MB # Adjust depending on host memory (e.g., 25% of RAM)
work_mem = 64MB # Suitable for OLTP like TPC-C
maintenance_work_mem = 256MB
effective_cache_size = 2GB # Depends on total system RAM

# WRITE-AHEAD LOG
wal_level = replica
synchronous_commit = off # Can improve write performance (acceptable for benchmarks)
checkpoint_timeout = 15min
checkpoint_completion_target = 0.9
max_wal_size = 2GB
min_wal_size = 512MB

# LOGGING
logging_collector = on
log_directory = 'log'
log_filename = 'postgresql.log'
log_statement = 'none'
log_min_duration_statement = 1000 # Log queries slower than 1s
```

6 Results

7 Discussion

8 Conclusions

Bibliography

[1] Wikipedia contributors. *HammerDB — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-May-2025]. 2025. URL: https://en.wikipedia.org/w/index.php?title=HammerDB&oldid=1275860580.