COMMISSIONED BY

EDB Postgres AI

MCKNIGHT CONSULTING GROUP

DATA STREAMING PERFORMANCE

Self-Hosted Database Transactional & NoSQL Performance Testing

EDB Postgres AI
Oracle Database
Microsoft SQL Server
MongoDB MySQL

RELATED RESEARCH

- Data Stream Processing Ease of Use and TCO
- Benchmark of data pipeline performance & cost

William McKnight

Jake Dolezal

Prepared by

MCKNIGHT CONSULTING GROUP

McKnight Consulting Group
www.mcknightcg.com

Contents

Executive Summary

A self-hosted database provides control, security, cost-effectiveness, flexibility and compliance. It allows organizations to customize and optimize performance, ensuring data resides securely within their infrastructure.

Performance in these databases is critical. By enabling fast transaction processing, low latency, high throughput and scalability, always-on database performance profoundly impacts the user experience, business operations and data integrity. Slow performance frustrates users, hindering productivity and satisfaction, while transaction processing delays affect revenue and competitiveness. Moreover, poor performance increases the risk of data corruption and loss.

Scalability is also compromised by inefficient performance, limiting growth and causing system overload. Furthermore, performance vulnerabilities create security risks, providing entry points for cyber threats. Resource utilization is also affected, wasting computing resources.

High performance ensures seamless transactions, efficient data processing and reliable operations. It maintains organizational credibility, prevents maintenance complexities and safeguards against security threats. Effective performance is essential. Organizations that prioritize performance benefit as a result.

By investing in the most fit database, including consideration of performance, companies can ensure their applications are running smoothly and securely, ultimately leading to better overall business outcomes.

EnterpriseDB (EDB) provides a data and AI platform that enables organizations to harness the full power of Postgres for transactional, analytical, and AI workloads across any cloud, anywhere. EDB empowers enterprises to control risk, manage costs and scale efficiently for a data and AI led world. EDB's data-driven solutions enable customers to modernize legacy systems and break data silos while leveraging enterprise-grade open source technologies. EDB delivers the confidence of up to 99.999% high availability with mission critical capabilities built in such as security, compliance controls, and observability.

We tested the unified EDB Postgres® AI sovereign data and artificial intelligence platform against popular options like Oracle and SQL Server to assess the performance and cost differences, as well as the potential value of exceeding enterprise agreements for a self-hosted transactional database.

We used TPC-C, which is a widely recognized benchmark for measuring database performance.

The results showed that EDB Postgres AI outperformed both Oracle and SQL Server in terms of speed and cost-effectiveness, making it a top choice for businesses looking to optimize their database operations. This is well worth it for a critical application that requires high performance and cost efficiency.

The TPC-C benchmark results show that EDB Postgres AI, Oracle, and SQL Server perform well under varying concurrency levels. EDB consistently outperforms Oracle and SQL Server, making it suitable for demanding enterprise environments. Peak performances are reached at 32 virtual users, with EDB Postgres AI and Oracle maintaining performance up to 128 virtual users.

EDB Postgres AI's cost-effectiveness is noteworthy, with a mere $0.21 per unit, compared to Oracle's $1.58 and SQL Server's $1.26. In a 3-year comparison, EDB Postgres AI ($205,605) offers substantial cost savings compared to Oracle ($1,333,765) and SQL Server ($957,768). EDB Postgres AI's superior database performance makes it ideal for mixed enterprise workloads, where efficient resource utilization and concurrency management are crucial for maximizing high transaction throughput, as well as handling large volumes of semi-structured data.

We also tested EDB Postgres AI against MongoDB and MySQL to evaluate the performance, cost differences, and potential benefits of going beyond enterprise agreements for a self-hosted NoSQL database. EDB Postgres AI offers better NoSQL performance compared to MongoDB and MySQL, particularly as data volume increases.

These tests highlight EDB Postgres AI's compelling value proposition, making it a prime choice for enterprises seeking a single database as a high-performance transactional and NoSQL database without breaking the bank.

Comparing EDB Postgres AI, Oracle, SQL Server, MongoDB, and MySQL Database Performance

EDB Postgres AI

EDB Postgres AI is a unified data platform for transactional, analytical, and new AI workloads powered by an enhanced Postgres engine. It offers enterprise-class features, scalability and reliability. EDB Postgres AI supports various platforms and programming languages. Its security features include Transparent Data Encryption (TDE) that significantly enhances security for data management systems, as well as row-level security and multi-factor authentication. EDB Postgres AI can be deployed as self-managed software in the cloud or on-premises, a managed cloud service, or as a hardware-integrated solution. EDB Postgres AI provides "single pane of glass" management and observability, AI-driven assistance, and database migration tooling.

Oracle

Oracle Database is a robust, scalable relational database management system. It provides advanced features for data warehousing, analytics and enterprise applications. Oracle's security features include encryption, access control and auditing. Its high availability and disaster recovery capabilities ensure minimal downtime. Oracle supports various platforms and programming languages.

SQL Server

Microsoft SQL Server is a comprehensive relational database management system. It offers robust security, scalability and performance. SQL Server supports advanced analytics, machine learning and data visualization. Its features include encryption, access control and auditing. High availability, disaster recovery and backup options ensure data protection. SQL Server integrates seamlessly with Microsoft tools and platforms.

MongoDB

MongoDB is a NoSQL document-based database designed for high scalability and flexibility. It stores data in JSON-like documents, allowing for dynamic schema design and efficient querying. MongoDB supports horizontal scaling, high-performance data retrieval, and flexible data modeling, making it a widely-used database management system for modern web and mobile applications.

MySQL

MySQL is a relational database management system that stores data in a structured format using tables, rows, and columns. It supports standard SQL and offers features like indexing, caching, and transactions to ensure data consistency and integrity. MySQL is widely used for web applications, supporting a large number of concurrent connections and queries, making it a reliable choice for managing and analyzing large datasets.

Platform Summary

For the transactional performance study, we tested the following releases of the platforms:

Vendor	EDB Postgres AI	Oracle Database	Microsoft SQL Server
Tier	EDB Postgres Advanced Server	Enterprise Edition	Enterprise 2022
Version	17.2	21c*	CU15 (16.0.4145.4)
OS	RHEL 9	Oracle Linux 7.9	RHEL 9

*A newer version of Oracle (23c) is generally available; however, at the time of this study, it was not offered as a download from Oracle.

For the NoSQL performance study, we tested the following releases of the platforms:

Vendor	EDB Postgres AI	MongoDB	MySQL
Tier	EDB Postgres Advanced Server	Enterprise Edition	Enterprise Edition
Version	17.2	8.0.3	8.0.38
OS	RHEL 9	RHEL 9	RHEL 9

Setup

This section analyzes the methods we used in our database transactional and NoSQL testing.

AWS EC2 Instances

To perform our testing, we used Amazon Web Services EC2 instances. We chose the following EC2 instance types for their high memory and attached NVMe storage:

Instance Type	i4i.8xlarge
Processors	32 vCPU Intel Xeon Scalable Ice Lake Processors
RAM	256 GB
Disks	Root on EBS gp3 + Data on 2x 3,750GB NVMe SSD Mirrored Disks (RAID1)

For each EC2 instance, we installed the database software and HammerDB as specified in the previous section. See the **Appendix** for all configuration settings used.

Transactional Workload: HammerDB TPC-C

The workload and data used in the transactional test were a workload derived from the well-recognized industry standard benchmark TPC-C. From tpc.org[1]:

> "TPC-C is an on-line transaction processing (OLTP) benchmark. TPC-C involves a mix of five concurrent transactions of different types and complexity either executed on-line or queued for deferred execution. The database consists of nine types of tables with a wide range of record and population sizes. TPC-C is measured in transactions per minute (tpmC). While the benchmark portrays the activity of a wholesale supplier, TPC-C is not limited to the activity of any particular business segment, but, rather represents any industry that must manage, sell, or distribute a product or service."

However, this is NOT an official TPC-C, and the results are not comparable with other published TPC-C results. To perform a TPC-C-like workload, standard practice involves using a driver that is already built. There are several TPC-C-like drivers available for free. We used HammerDB 4.11[2]. HammerDB is a widely used and accepted implementation of the TPC-C test.

[1] More can be learned about the TPC-C benchmark at http://www.tpc.org/tpcc/.

[2] For more information about or to download HammerDB, visit https://www.hammerdb.com/.

Transactional Test Measurements

Our benchmark collected two measurements.

- **NOPM (New Orders Per Minute)** – measures the throughput of new order transactions that a system can handle per minute, reflecting its transaction processing capability.
- **Price-Performance** – measures the cost-effectiveness of a system by calculating the total price per transaction per minute (NOPM), indicating how efficiently the system performs transactions relative to its price.

NOPM and Price-Performance are pivotal measurements for evaluating self-hosted transactional databases. NOPM assesses transaction throughput, scalability and real-world applicability by measuring the database's ability to process transactions efficiently, handle concurrent requests and simulate real-world workloads. Price-Performance evaluates cost-effectiveness, comparing performance to costs, enabling value assessments across databases and optimizing resource allocation.

These measurements are chosen for their relevance to transactional use cases, facilitating comparability across databases and informing optimization, scaling and cost management decisions. By considering NOPM and Price-Performance, users gain comprehensive insights into database performance, scalability and cost-effectiveness, guiding informed decisions for optimal database selection and configuration.

NoSQL Workload: PG NoSQL Benchmark

The workload and data used in the NoSQL benchmark were a workload derived from the PG NoSQL Benchmark[3]. The original benchmark was only designed for PostgreSQL and MongoDB. We modified the benchmark to add a MySQL suite of functions based on its query syntax. EDB was already fully compatible with the PostgreSQL component of the benchmark, so no modifications for EDB were needed.

Our test performed the following tasks to compare EDB, MongoDB, and MySQL:

- Generated large sets of JSON documents with the amount of data ranging from 5,000,000 documents (13GB) to 100,000,000 documents (266GB)
- Loaded the data into each database using EDB's SQL COPY, mongoimport, and MySQL's LOAD DATA LOCAL INFILE commands

[3] More can be learned about the PG NoSQL Benchmark at https://github.com/EnterpriseDB/pg_nosql_benchmark.

- Loaded the same data using each platform’s INSERT command
- Executed 4 SELECT queries on each platform

NoSQL Test Measurements

Our benchmark collected three measurements.

- **Bulk load elapsed time** – measures the total time elapsed to load all the JSON via bulk load
- **INSERT elapsed time** – measures the total time elapsed to INSERT all the JSON via bulk load
- **Average SELECT time** – measures the average time to run each SELECT query

These measurements are chosen for their relevance to NoSQL use cases, facilitating comparability across databases and informing optimization, scaling and performance decisions.

Results

This section analyzes the results of our transactional and NoSQL workload scenarios for the platforms we tested.

Transactional NOPM (New Orders Per Minute)

The following chart shows the NOPM of the three databases at various levels of virtual (concurrent) users, up to 320. A typical concurrency load for an enterprise workload similar to TPC-C can vary but average ranges are probably in the 25-75 range.

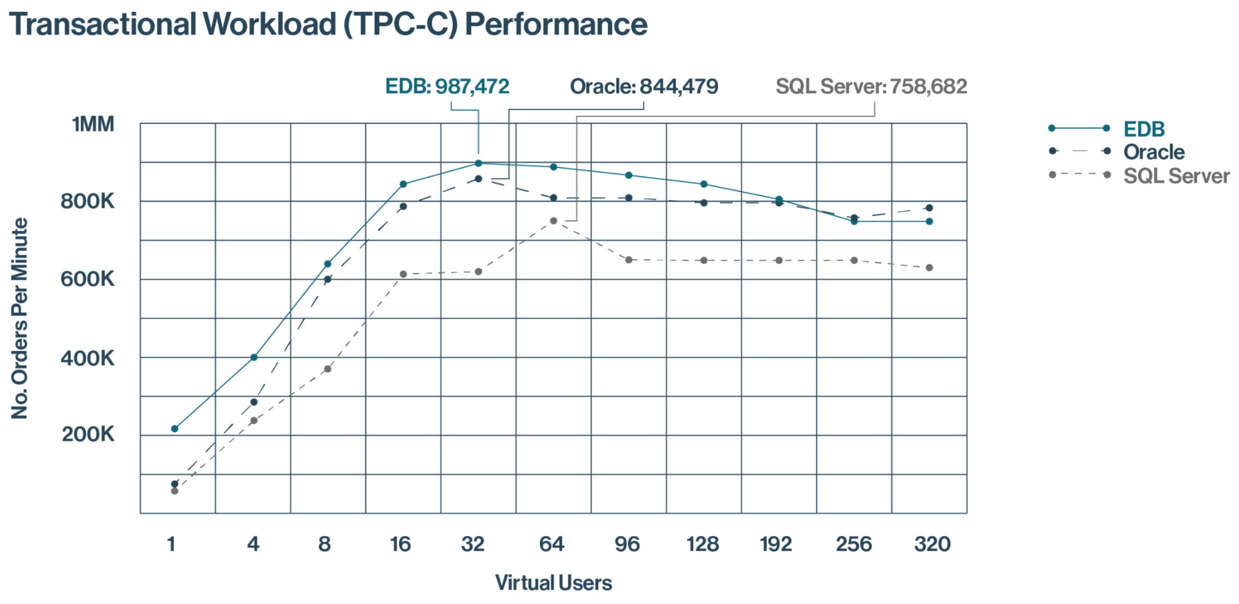

Figure 1: New Orders Per Minute by Number of Virtual (Concurrent) Users

The TPC-C benchmark results offer valuable insights into EDB, Oracle and SQL Server performance under varying concurrency levels. Notably, all databases exhibit scaling improvements up to 64 virtual users, indicating efficient resource utilization. However, beyond this point, NOPM growth slows or declines, signaling concurrency limits.

EDB consistently outperforms Oracle and SQL Server, showcasing its suitability for demanding enterprise environments. Peak performances were reached between 32 and 64 virtual users: EDB (987,472 NOPM), Oracle (844,479 NOPM) and SQL Server (758,682 NOPM).

These findings have significant implications. EDB's superior performance makes it ideal for high-concurrency enterprise workloads. Efficient resource utilization and concurrency management are crucial for maximizing NOPM. Understanding scalability limits informs capacity planning and infrastructure investments. Database selection should be informed by performance benchmarking.

Price-Performance

EDB Postgres AI took the top spot for maximum New Orders Per Minute (NOPM), processing an impressive 987,472 transactions. Oracle closely followed with 844,479 NOPM, while SQL Server trailed at 758,682 NOPM.

However, the real surprise came with price-performance comparisons. EDB's cost-effectiveness shone, boasting a mere $0.21 per unit. Oracle's $1.58 and SQL Server's $1.26 rendered them 7 and 6 times more expensive, respectively.

This stark pricing disparity underscores EDB's compelling value proposition. EDB Postgres AI's unparalleled performance, coupled with unmatched affordability, makes it a prime choice for enterprises seeking high-performance transactional databases without breaking the bank.

NoSQL Performance

The following charts show the NoSQL performance of the three databases at various levels of data scale up to 266GB of JSON Data.

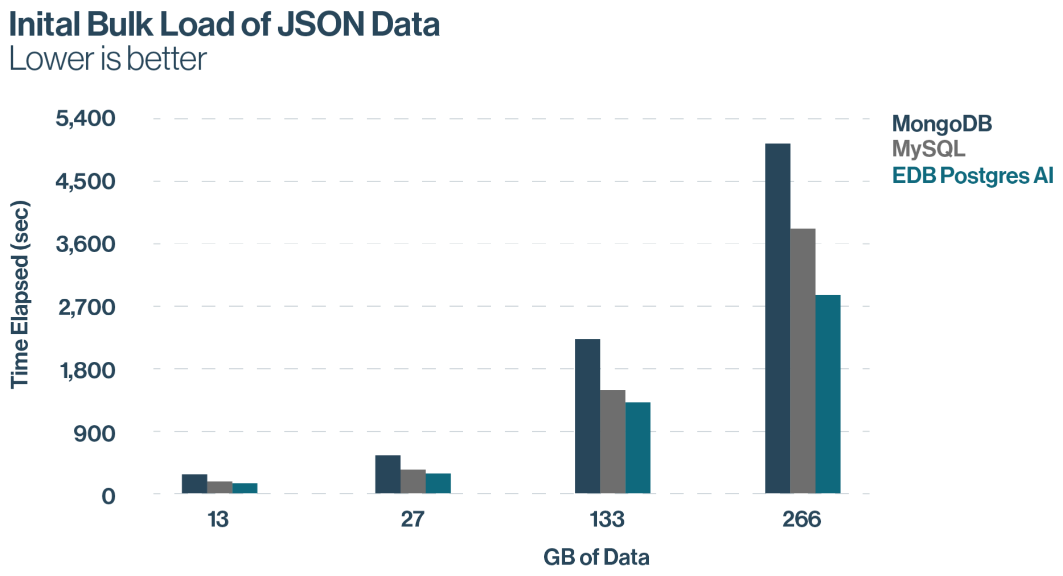

Figure 3: Total Elapsed Time of Initial Bulk Load of JSON Data

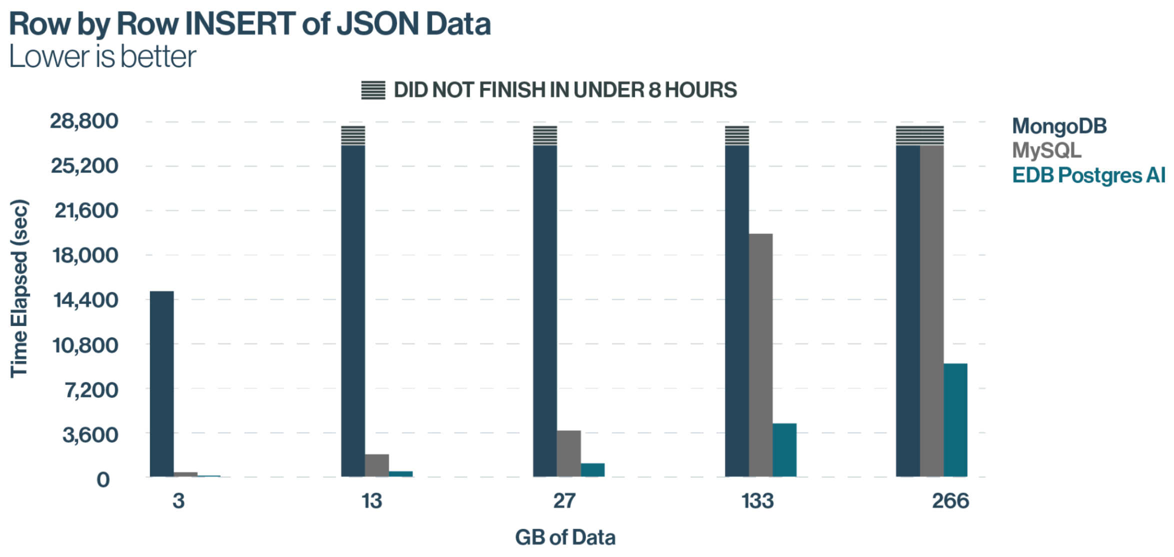

Figure 4: Total Elapsed Time of Row-by-Row INSERT of JSON Data

NOTE: After 8 hours, MongoDB did not finish the row-by-row INSERTs for the 13GB-266GB test runs. MySQL did not finish loading 266GB row-by-row within 8 hours. Those tests ended after 8 hours and were marked as incomplete.

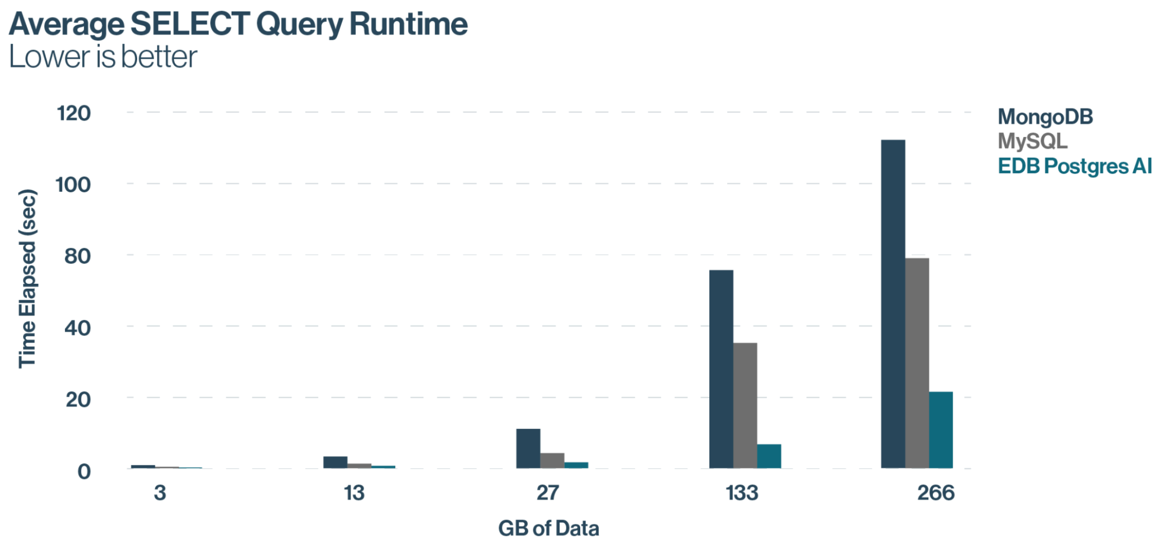

Figure 5: Average Elapsed Time of SELECT queries of JSON Data

The PG NoSQL Benchmark results offered insight into raw EDB, MongoDB, and MySQL NoSQL performance with varying amounts of data.

- For bulk loads, EDB was 34% faster than MongoDB and 63% faster than MySQL.
- For row-by-row INSERTs, EDB was over 4 times faster than MySQL and well over 150 times faster than MongoDB.
- For average SELECT query times, EDB was nearly 3 times faster than MySQL and over 5 times faster than MongoDB.

Key takeaways

1. EDB Postgres AI excelled in transactional performance.
2. Oracle and SQL Server trailed in performance.
3. EDB Postgres AI offered superior price-performance.
4. Oracle and SQL Server were significantly pricier.
5. EDB Postgres AI outperformed MongoDB and MySQL even as data volumes increased.

For enterprises demanding high-performance, cost-effective transactional NoSQL databases, EDB Postgres AI stands out as the optimal solution.

Total Cost

We compared the total cost for the transactional competitors.

A cost comparison of EDB Postgres AI, Oracle and SQL Server on AWS reveals significant pricing disparities. Instance costs were held identical at $24,055/year. Software licensing costs differ substantially: EDB's per-core licensing ($2,780), Oracle's perpetual license, processor-based ($47,500) and SQL Server's 2-core pack ($15,123).

Support costs also vary: EDB's is included, Oracle's ($10,450) and SQL Server's ($3,327). First year costs range from $68,535 (EDB), $951,255 (Oracle) and $319,256 (SQL Server). The three-year projections are $205,605 (EDB), $1,333,765 (Oracle) and $957,768 (SQL Server).

The implications are clear. **EDB Postgres AI's per-core licensing offers substantial savings, making it a cost-effective choice.** Oracle's processor-based licensing is costly upfront, while SQL Server's 2-core pack falls in between. Support costs significantly impact total expenses.

Enterprises must carefully evaluate licensing models, factor support costs and consider long-term commitments to optimize database expenditures.

	EDB	ORACLE	SQL Server
Edition	Enterprise	Enterprise	Enterprise
Cloud	AWS	AWS	AWS
Region	US East 1	US East 1	US East 1
Instance Type	i4i.8xlarge	i4i.8xlarge	i4i.8xlarge
vCPU	32	32	32
Instance $/hour	$2.746	$2.746	$2.746
Instance $/year	**$24,055**	**$24,055**	**$24,055**
Software UoM	Per Core	Processors	2 Core Pack
Software Units	16	16	16
Software Period	Annual	Perpetual	Annual
Software $/unit	$2,780	$47,500	$15,123
Support $/unit	included	$10,450	$3,327
Software & Support Year 1	**$44,480**	**$927,200**	**$295,201**
Software & Support Year 2+	**$44,480**	**$927,200**	**$295,201**
3 Year Cost	**$205,605**	**$1,333,765**	**$957,768**

Conclusion

EDB Postgres AI offers cost-effective per-core licensing, compared to SQL Server's two-core pack and Oracle's expensive processor-based licensing. It performs better in demanding business settings, with 32 virtual users achieving peak performance. EDB Postgres AI processed the highest New Orders Per Minute (NOPM) total, with each unit costing $0.21, compared to Oracle's $1.58 and SQL Server's $1.26. Support charges significantly impact total expenses.

EDB Postgres AI offers enterprises significant benefits through its cost-effective per-core licensing, superior performance and lower support charges. EDB Postgres AI's scalability, reliability and streamlined operations enhance financial efficiency, reducing Total Cost of Ownership. Its competitive pricing ($0.21/unit vs. Oracle's $1.58 and SQL Server's $1.26) and peak performance with 32 virtual users make it ideal for demanding business settings. EDB Postgres AI is vastly less expensive than Oracle and SQL Server over three years.

EDB Postgres AI offers superior NoSQL performance compared to MongoDB and MySQL, particularly as data volume increases, with a focus on how JSON data formats impact performance and scalability.

By choosing EDB Postgres AI, enterprises boost return on investment, gain budget flexibility and drive strategic growth through innovation and digital transformation, ensuring a forward-thinking technology infrastructure, all in a single database.

Appendix: Test Configuration

We tested each platform under the following configurations and conditions:

EDB Postgres AI

For the disk configuration, we moved **pgdata** and **pg_wal** to the mirrored NVMe disks.
In the **edb-as-17.service** configuration file, we added:

```
After=tuned.service
```

In **postgresql.conf**, we changed:

```
max_connections = 1000
shared_buffers = 64GB
effective_cache_size = 128GB
work_mem = 8MB
maintenance_work_mem = 64GB
autovacuum_work_mem = 1GB
effective_io_concurrency = 200
maintenance_io_concurrency = 200
max_worker_processes = 128
max_parallel_maintenance_workers = 64
synchronous_commit = on
wal_level = replica
wal_buffers = 512MB
checkpoint_timeout = 1d
checkpoint_completion_target = 0.9
max_wal_size = 1TB
min_wal_size = 64GB
random_page_cost = 1.0
vacuum_cost_limit = 8000
autovacuum = off
autovacuum_freeze_max_age = 800000000
autovacuum_multixact_freeze_max_age = 800000000
max_locks_per_transaction = 512
edb_dynatune = 0
```

In **HammerDB**, we used the following build parameters:

```
diset tpcc pg_count_ware 1000
diset tpcc pg_num_vu 32
diset tpcc pg_oracompat false
diset tpcc pg_storedprocs false
diset tpcc pg_partition true
```

Also in **HammerDB**, we used the following run parameters:

```
diset tpcc pg_allwarehouse true
diset tpcc pg_oracompat false
diset tpcc pg_storedprocs false
diset tpcc pg_vacuum true
```

Oracle

For the disk configuration, we moved **/u02/oradata** to the mirrored NVMe disks.

As **sysdba**, we ran:

```
alter system set aq_tm_processes=0;
alter system set commit_logging='BATCH';
alter system set commit_wait='NOWAIT';
alter system set db_block_checking=FALSE;
alter system set db_block_checksum=FALSE;
alter system set db_cache_advice=off;
alter system set db_cache_size=100G;
alter system set db_unrecoverable_scn_tracking=FALSE;
alter system set ddl_lock_timeout=30;
alter system set deferred_segment_creation=FALSE;
alter system set log_checkpoint_interval=0;
alter system set log_checkpoint_timeout=0;
alter system set log_checkpoints_to_alert=TRUE;
alter system set open_cursors=2400;
alter system set optimizer_capture_sql_plan_baselines=FALSE;
alter system set optimizer_dynamic_sampling=4;
alter system set parallel_degree_policy='AUTO';
alter system set query_rewrite_enabled=FALSE;
alter system set shared_pool_size=32G;
alter system set trace_enabled=FALSE;
alter system set undo_retention=30;
alter tablespace TEMP add tempfile '/u02/oradata/CDB1/tpcc/temp02.dbf' SIZE 10G
AUTOEXTEND ON NEXT 1G MAXSIZE 32767M;
alter tablespace SYSTEM add datafile '/u02/oradata/CDB1/tpcc/system02.dbf' SIZE
1G AUTOEXTEND ON;
alter tablespace SYSAUX add datafile '/u02/oradata/CDB1/tpcc/sysaux02.dbf' SIZE
1G AUTOEXTEND ON;
alter tablespace USERS add datafile '/u02/oradata/CDB1/tpcc/users02.dbf' SIZE
10240M AUTOEXTEND ON NEXT 1G MAXSIZE 32767M;
alter tablespace USERS add datafile '/u02/oradata/CDB1/tpcc/users03.dbf' SIZE
10240M AUTOEXTEND ON NEXT 1G MAXSIZE 32767M;
alter tablespace USERS add datafile '/u02/oradata/CDB1/tpcc/users04.dbf' SIZE
10240M AUTOEXTEND ON NEXT 1G MAXSIZE 32767M;
```

As **root**, we made the following OS changes:

```
sudo sysctl vm.dirty_ratio=80
sudo sysctl vm.dirty_background_ratio=3
sudo sysctl vm.swappiness=1
sudo sysctl vm.dirty_expire_centisecs=500
sudo sysctl vm.dirty_writeback_centisecs=100
sudo sysctl fs.aio-max-nr=1048576
sudo sysctl fs.file-max=6815744
sudo sysctl kernel.shmmni=4096
sudo sysctl kernel.shmall=1073741824
sudo sysctl kernel.shmmax=4398046511104
sudo sysctl kernel.panic_on_oops=1
```

In **HammerDB**, we used the following build parameters:

```
diset tpcc count_ware 1000
diset tpcc partition true
diset tpcc hash_clusters true
```

Also in **HammerDB**, we used the following run parameters:

```
diset tpcc pg_allwarehouse true
```

SQL Server

For the disk configuration, we moved the **data, log, and temp files** to the mirrored NVMe disks.

In **sqlcmd**, we ran:

```
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE WITH OVERRIDE;
GO
EXEC sp_configure 'max degree of parallelism', 1;
GO
RECONFIGURE WITH OVERRIDE;
GO
```

As **root**, we made the following OS and SQL Server changes:

```
sudo sysctl kernel.sched_min_granularity_ns=10000000
sudo sysctl kernel.sched_wakeup_granularity_ns=15000000
sudo sysctl vm.dirty_ratio=40
sudo sysctl vm.dirty_background_ratio=10
sudo sysctl vm.swappiness=10
sudo sysctl -w vm.max_map_count=262144
sudo sysctl -w kernel.numa_balancing=0
sudo /opt/mssql/bin/mssql-conf set filelocation.defaultdatadir /data
sudo /opt/mssql/bin/mssql-conf set filelocation.defaultlogdir /log
sudo /opt/mssql/bin/mssql-conf set network.tlsprotocols 1.2
sudo /opt/mssql/bin/mssql-conf traceflag 3979 on
sudo /opt/mssql/bin/mssql-conf set control.writethrough  1
sudo /opt/mssql/bin/mssql-conf set control.alternatewritethrough  0
sudo /opt/mssql/bin/mssql-conf set memory.memorylimitmb 235900
```

In **HammerDB**, we used the following build parameters:

```
diset tpcc mssqls_count_ware 1000
```

Also in **HammerDB**, we used the following run parameters:

```
diset tpcc mssqls_allwarehouse true
```

About EnterpriseDB

Nearly 1,500 customers worldwide have chosen EDB software, services, and support.

Enterprises and governments around the globe trust EDB to harness the full power of Postgres. With unmatched expertise, EDB ensures high availability, reliability, security, 24x7 global support and advanced professional services, both on premises and in the cloud, to help control risk, manage costs and scale efficiently. As a leading contributor to the growing PostgreSQL community, EDB is committed to driving technology innovation.

As your organization grows, you need strong strategies for the technologies you bet on. Postgres delivers superior technology, powered by a thriving, vibrant and fast-growing community. To accelerate your Postgres journey, and to run Postgres as your enterprise database standard, you need the unified EDB Postgres AI platform.

EDB is the heartbeat of Postgres, with hundreds of technologists and developers, and more open source contributions to Postgre than any other company. EDB drives innovation, with enterprise-class software and services that enable businesses and governments globally to harness the full power of Postgres, the world's leading database.

About McKnight Consulting Group

Information Management is all about enabling an organization to have data in the best place to succeed to meet company goals. Mature data practices can integrate an entire organization across all core functions. Proper integration of that data facilitates the flow of information throughout the organization which allows for better decisions – made faster and with fewer errors. In short, well-done data can yield a better run company flush with real-time information... and with less costs.

However, before those benefits can be realized, a company must go through the business transformation of an implementation and systems integration. For many that have been involved in those types of projects in the past – data warehousing, master data, big data, analytics - the path toward a successful implementation and integration can seem never-ending at times and almost unachievable. Not so with McKnight Consulting Group (MCG) as your integration partner, because MCG has successfully implemented data solutions for our clients for over a decade. We understand the critical importance of setting clear, realistic expectations up front and ensuring that time-to-value is achieved quickly.

MCG has helped over 100 clients with analytics, big data, master data management and “all data” strategies and implementations across a variety of industries and worldwide locations. MCG offers flexible implementation methodologies that will fit the deployment model of your choice. The best methodologies, the best talent in the industry and a leadership team committed to client success makes MCG the right choice to help lead your project.

MCG, led by industry leader William McKnight, has deep data experience in a variety of industries that will enable your business to incorporate best practices while implementing leading technology. See www.mcknightcg.com.

Disclaimer

McKnight Consulting Group (MCG) runs all its tests to strict ethical standards. The results of the report are the objective and unbiased results of the application of queries to the simulations described in the report. The report clearly defines the selected criteria and process used to establish the field test. The report also clearly states the data set sizes, the platforms, the methods, etc. that were used. The reader is left to determine for themselves how to qualify the information for their individual needs. The report does not make any claims regarding third-party certification and presents the objective results received from the application of the process to the criteria as described in the report. The report strictly measures performance and does not purport to evaluate other factors that potential customers may find relevant when making a purchase decision. This is a sponsored report. The client chose its configuration, while MCG chose the test, configured the database and testing application, and ran the tests. MCG also chose the most compatible configurations for the other tested platforms. Choosing compatible configurations is subject to judgment. The information necessary to replicate this test is included. Readers are encouraged to compile their own representative configuration and test it for themselves.