Evaluation of Database Management Systems and Techniques Applied in Distributed Systems

Mary Jane C. Samonte
School of Information Technology,
Mapúa University,
Manila, Philippines
mjcsamonte@yahoo.com

Sasky A. Samonte
School of Information Technology,
Mapúa University,
Manila, Philippines
saskysamonte@gmail.com

Abstract

The technical solution used to improve and manage the storage and retrieval of data from databases is known as a database management system (DBMS). A transaction signifies a change in the database's data. It is an activity or sequence of operations performed by a single user or application software to read or change the database's contents. A transaction can fail before all the actions in the set are completed, which is a concern. This might occur because of a power outage, a system crash, or other factors while Concurrency Control in a Database Management System is a method of controlling many actions at the same time without their interfering. To resolve these issues, the various techniques are presented in this study with discussion of techniques and algorithms used. Concurrency control methods like locking, timestamps, and Optimistic Concurrency Control can be used to handle concurrent transactions and avoid conflicts. Studying a distributed database for relational (MySQL) and NoSQL (MongoDB), as well as the implementation and performance of ACID transactions in NoSQL, is another proposal (MongoDB).

Keywords
Transactional Log Analysis, Performance and Scalability, Concurrency Control

1. Introduction

Databases allow users to store, retrieve, and analyze data quickly and easily. For business and research, databases are essential. organizations and other fields where data has important role to play. Globally in business environment, organizations are to process more data than ever before. All the crucial business decisions are made on the basis on that available data and only well-designed data management policy will make data more reliable to help make better decisions Rana et al. (2018). Concurrency Control in a Database Management System is a method of controlling many actions at the same time without their interfering. It ensures that Database transactions are completed in a timely and accurate manner to deliver accurate results without compromising the database's data integrity. However, the system is saturated with conflicts due to frequent rollbacks, long time waiting and blocking, and a high rate of aborted transactions, resulting in deadlock. Additionally, the evolution of distributed systems applications accelerates, accompanied by the emergence of several other new challenges in need of resolution and the demand for technologies that regulate the distributed environment. MongoDB is a significant rival to MySQL, as well as a complement to it. It outperforms SQL in a variety of domains and circumstances when compared to MySQL.

1.1 Objectives

The Internet, particularly social media and instrument sensors are largely to blame for such massive data dissemination and complexity. The concept of adopting a relational model as a "one-size-fits-all" solution has been questioned as to whether it can meet these new difficulties, hence other data models known as NoSQL are being investigated Deari (2018). To resolve these issues, the various techniques are presented in this study. Different techniques of algorithms used by previous researchers are presented and appraised through a thorough literature study of various articles related to transaction management and concurrency control and this paper also reviewed the comparison between Relational and NoSQL database management systems released in the last five years.

2. Literature Review

As more and more big data-related application systems are developed, Database systems based on NoSQL are becoming increasingly popular but there are various drawbacks to NoSQL databases, like as SQL is not supported, which is an industry standard, and there are no reports or any extra capabilities, with the proposed model, the transactions of Key-Value NoSQL databases may be used in a lock-free and Multi-Version Concurrency Control free manner Li and Gu (2018). In distributed system, it is investigated that the improvement of employing cache-based query optimization to reduce response time is suitable option because it saves huge computational cost and time, especially introducing cache at each site instead of implementation of one network-based cache however, it is observed that currently cache is implemented only in homogeneous distributed databases Rana et al. (2018). The rise in the number of processes performed on a shared platform many issues arose as a result of the database, which necessitated the development of strategies to address them such as competing operations and deadlocks. If the system is aware of the risk of a stalemate in one of the processes, it has the ability to cancel any transaction that requests a new lock. This transaction's modifications have been reversed and rescheduled. All locks are released, and the other transactions are continued which means can avoid the conditions that lead to deadlocking Hamied (2018).

3. Methods

Google Scholar is the search engine used for researching a wide variety of sources such as research, journals, and articles. A set of actions are taken when seeking publications in the mentioned search engine: (1) Type a topic or a keyword; (2) Filter the publication date to only display results from 2017 to 2020; (3) Add the studies to the discovered studies column; (4) Filter the studies based on the title and use techniques and add to the filtered studies table.

3.1 Study Filtering

The selected and filtered studies are ACM = 5, Research Gate = 2, Springer = 1, MDPI = 1, Sholar Space = 1, Scientific Research = 1, InfoTech = 1, Digital Library = 1, Philippine Journal of Librarianship and Information Studies = 1, International Journal for Research in Applied Science and Engineering Technology =1, Cornell University = 1, Misurata University Publications = 1, Novelty Journals = 1, International Journal of Engineering Intelligent Systems, = 1 and Academia =1 with a total of twenty (20) studies. Each study of used proposed techniques and algorithm in different types and distributed database managements systems are included in Table 1, along with their popularity among comparable research based on the number of citations.

Table 1: Popularity of The Studies Chosen.

Title Of Study	Source Of Study	Technique/s Used	Number Of Citations
Ruppel and Lucia (2019) Transactional Concurrency Control for Intermittent, Energy-Harvesting Computing Systems.	ACM	Capybara Hardware Platform, GPIO Pin Powered Arduino, Bitcount (BC)	26
Mhawes and Ali Hassan (2020) A Top-Down Approach to Achieving Performance Predictability in Database Systems.	ACM	Tprofiler/Vprofiler Tool	24
Villega et al. (2020) A Business Intelligence Framework for Analyzing Educational Data.	MDPI	BI Framework, Unified Modeling Language, SIPOC Diagram, Kimball Methodology	8
Deari (2018) Analysis And Comparison of Document-Based Databases with SQL Relational		Insert Operation, Read	

Title Of Study	Source Of Study	Technique/s Used	Number Of Citations
Databases: Mongodb Vs Mysql.	Infotech	Operation, Update Operation, Delete Operation. All Operation Used Nodejs Scripts to Perform Testing	7
Sauer (2019) Modern Techniques for Transaction Oriented Database Recovery.	Digitallibrary	Instant Restart, Single-Pass Restore, Decoupled Persistence, Fineline Algorithm For Recovery and Isolation Exploiting Semantics (ARIES), Yahoo! Cloud Serving Benchmark (YCSB), Remote Flush Avoidance Python And Shell Scripts	6
Haubenschild et al. (2020) Rethinking Logging, Checkpoints, And Recovery for High-Performance Storage Engines.	ACM	Netty Framework, Algorithm 1 Graphnode.Class, Algorithm 2 Begin, Algorithm 3 Commit, Algorithm 4 Simplyconflicts, Algorithm 5 Simplyswapgraphnode	5
Lee and Fortes (2019) Improving Data-Analytics Performance Via Autonomic Control of Concurrency and Resource Units.	ACM	Transaction Log Analysis (TLA)	5
Fan et al. (2020) 2PC*: A Distributed Transaction Concurrency Control Protocol of Multi-Microservice Based on Cloud Computing Platform.	Springer	K-Means Clustering Algorithm, AES Algorithm	3
Fresnido and Barsaga (2019) Transaction Log Analysis of OPAC Searches in An Academic Library: Basis for OPAC Interface Improvement.	Philippine Journal of Librarianship And Information Studies	Hybrid Cloud Automation, Cloud Bursting Deployment Model And Wireguard Communication Protocol	2
Lokhande and Deshmukh (2019) A Novel Approach for Transaction Management in Heterogeneous Distributed Real-Time Replicated Database Systems.	International Journal for Research in Applied Science and Engineering Technology	Algorithm 1: Worker Thread, Algorithm 2: Log Manager Thread, Algorithm 3: Log Manager Recovery for Thread, Algorithm 4: Worker Recovery Thread, Algorithm 5: LV Compression for Log Records	2
Mansouri and Ali Babar (2020) The Impact of Distance on Performance and Scalability Of Distributed Database Systems In Hybrid Clouds.	Cornell University	Search Space, Search Strategy and Distributed Cost Model	2
Xia et al. (2020) Taurus: Lightweight Parallel			

Title Of Study	Source Of Study	Technique/s Used	Number Of Citations
Logging for In-Memory Database Management Systems (Extended Version).	ACM	CQL Statements to Create, Populate, And Select from A Table	2
Rana et al. (2018) Analysis of Query Optimization Components in Distributed Database.	Researchgate	Yahoo! Cloud Serving Benchmark (YCSB), Automated Testing Scripts (Unix Shell Based Bash)	1
Marungo (2018) A Primer on Nosql Databases for Enterprise Architects: The CAP Theorem and Transparent Data Access with Mongodb And Cassandra.	Sholarspace	Transaction Sorter Algorithm, Transaction Dispatcher Algorithm, Transaction Executor Algorithm, Transaction Rollback Method, Timeout Processing Method, Faults Detection Method	1
Pandey (2020) Performance Benchmarking and Comparison of Cloud-Based Databases Mongodb (Nosql) Vs Mysql (Relational) Using YCSB.	Researchgate	Banker's Algorithm	1
Li and Gu (2018) A Surfing Concurrence Transaction Model for Key-Value Nosql Databases.	Scientificresearch	Top-Down Method, Bottom-Up Method	1
Transaction Management and Concurrency Control. Hamied (2018)	Misurata University Publications	Matlab And C++ Programming Software	0
Mhawes and Ali Hassan (2020) Methods to Implement Homogeneous and Heterogeneous Distributed Database.	Noveltyjournals	Bank Application Program with A Timer Scheduler Using C# Programming	0
Peng et al. (2020) Intelligent Indexing Algorithm for The Association Rules of a Multi-Layer Distributed Database.	International Journal of Engineering Intelligent Systems		0
Christopher and Kabari (2020) Hybridized Concurrency Control Technique for Transaction Processing in Distributed Database System.	Academia		0

4. Data Collection

The first quantitative investigation is carried out by the researcher. MySQL and Postgres are two of the most common.the largest and most popular open-source and VoltDB a non-conventional database. This study carried out with a tool called TProfiler that, given the source code of a database system and programmer annotations indicating the start and end of a transaction, is able to identify the dominant sources of variance in transaction latency. The researcher throughput of 500 transactions per second was utilized across the board in terms of workloads and algorithms. Table 2 shows modifying each of TProfiler's recognized functions has an impact. The last three columns compare transactions from beginning to finish, before and after each adjustment, there are latencies. Huang et al. (2017).

In another research, the study compared the two database systems' performance (Relational MySQL and NoSQL), and chose YCSB as our performance testing tool. Its ability to perform workloads with a variety of settings. Researcher chose Workload A, B, and C. Researcher used the databases C and F and executed the workloads for both of them. Counts and operations range from 12500 to 50000, 100000 to 150000 to 200000. The experiment was carried out using the Amazon Web Services cloud platform. YCSB allows to test the performance of various workloads against the database. The various types of standard workloads are detailed in Table 3, with the workloads employed in this experiment highlighted in color green Pandey (2020).

The benefits of MongoDB may be demonstrated in the tests that were undertaken. Using the YCSB framework, it is proven that for each type of workload (A, C, and F), MongoDB outperformed MySQL in every parameter. Particularly in terms of MongoDB shines out in terms of latency and throughput, particularly in terms of the number of operations it can do. Several relevant works were examined, and practically all of them were found to be excellent have proven that NoSQL databases are a superior solution for massive cloud-based applications Pandey (2020).

5. Results and Discussion

5.1 Database Management System (DBMS)

Previous and future researchers continuously study optimizing the query response time, improving performance, and preventing transaction failures in different database management systems (DBMS) using a proposed method such as techniques and a new model approach. DBMS software serves as a bridge between the end-user and the database, handling data, the database engine, and the database schema at the same time to make data organization and manipulation easier. For effective data storage and manipulation, transactional databases are a mission-critical component of enterprise software. Table 4 lists the types of DBMS discussed in each of the papers included in this evaluation.

Table 4. Various Types of DBMS Conducted in Transaction and Concurrency Control Analysis.

DBMS Name	Database Model	Number of Papers	Reference Studies
MySQL	Relational	8	Huang et al. (2017), Marungo (2018), Deari (2018), Lokhande and Deshmukh (2019), Fan et al. (2020), Villega et al. (2020), Pandey (2020), Mansouri and Ali Babar (2020)
PostgreSQL	Relational	2	Huang et al. (2017), Deari (2018)
VoltDB	Relational	1	Huang et al. (2017)
Microsoft SQL	Relational	1	Christopher and Kabari (2020)
MongoDB	NoSQL	6	Marungo (2018), Deari (2018), Li and Gu (2018), Lokhande and Deshmukh (2019), Pandey (2020), Haubenschild et al. (2020)
Cassandra	NoSQL	2	Marungo (2018), Mansouri and Ali Babar (2020)
HBase	NoSQL	1	Li and Gu (2018)
Riak	NoSQL	1	Mansouri and Ali Babar (2020)
CouchDB	NoSQL	1	Mansouri and Ali Babar (2020)
Redis	NoSQL	1	Mansouri and Ali Babar (2020)
DynamoDB	NoSQL	1	Lokhande and Deshmukh (2019)

5.2 Distributed Database Management System

By fragmenting and replicating data, a distributed database is physically scattered over several locations. The capacity of the query optimizer to implement appropriate query processing plans is critical to the operation and performance of a Distributed Database. Relationships in dispersed databases are fragmented and/or duplicated to several sites, and all locations require data from each other. As a result, query response time is quite long. The techniques were used are Search Space, Search Strategy and Distributed Cost Model. At the end, it is evaluated if improving response time by utilizing cache-based query optimization is a viable alternative because it saves significant computational cost and time. It also discovered that caching is now used in homogenous distributed databases Rana et al. (2018).

5.2.1 Types of Distributed Database

Distributed databases are divided into homogeneous and heterogeneous databases, each with its own subcategories. Homogeneous Databases are distributed over many servers, as well as database management systems (DBMS), are all the same. This means that all dispersed sites have the same sort of database, both in terms of structure and database management systems while Heterogeneous Database are dispersed across several servers, where some sites have a certain type of database with a specific management system (DBMS) and structure, while other sites have various types of databases Mhawes and Ali Hassan (2020).

5.3 "Not SQL or" NoSQL Data Model

NoSQL is a set of concepts that enables for the quick and effective processing of large data volumes while focusing on performance, dependability, and agility. MongoDB is one of the examples of NoSQL data model, is a database management system discuss in this review to show the advantages in MySQL (Relational Data Model). There is no requirement to ensure referential integrity because each record is duplicated independently. ACID (Atomicity, Consistency, Isolation, and Durability) transactional properties are typically not provided by NoSQL, researcher proposed the CAP Theorem, BASE semantics, and ACID transactions in the context of business architecture the CAP theorem is introduced by the requirement to duplicate data. As a result, a key implication of the CAP Theorem is that combining redundancy and consistency leads to system instability. Instead, systems can duplicate records in a CP or AP fashion and control ACID relaxation using BASE semantics Marungo (2018). MongoDB (NoSQL) performs better than MySQL in all parameters for each kind of workload (A, C, and F). In terms of sharding, security, performance, and availability, MongoDB (NoSQL) outperforms MySQL and NoSQL databases are a better choice for big cloud-based applications Pandey (2020).

5.4 Relational Data Model

A relational database is a form of database that stores and allows access to data points that are related. MySQL is an example of Relational Database Management System discuss in this review to compare in MongoDB (NoSQL). MySQL's architecture works best with data that has organized and finite fields since MySQL can search and arrange it in several dimensions. However, this method will not work with unstructured data. MySQL is a popular choice for high-performance RDBMS since it preserves data integrity even during scaling and sharding Pandey (2020). If data integrity is a top priority, MySQL is the way to go because it supports ACID transactions. In comparison to MongoDB (NoSQL), which was founded on the BASE (essentially available, soft state, with eventual consistency) idea, MySQL has been deemed a big benefit in ensuring data integrity through ACID features. However, in MongoDB's most recent version, there is a significant shift toward support for ACID characteristics Deari (2018).

5.5 Transaction Management

A transaction is defined as a series of activities performed sequentially within a computer program and guided by the application user who initiates and ends the transaction. Transactions function as atomic units; it can either be completed successfully if the process has been completed, or they can fail owing to aborted processes for various reasons. Consistency and recovery unit in ACID refers to a set of qualities that all database transactions have in common Hamied (2018). Database systems are also very performance-sensitive, and as a result, a major portion of database research is focused to making databases as fast as possible. By eliminating components like logging and recovery, locking, and buffer management, this approach improves transactional performance by orders of magnitude. This study was a significant "wake-up call" that affected many of the systems built in the previous decade, the extreme approach of removing as much functionality as possible in the name of efficiency is likely to limiting for database systems in general. Subsequent research articles have gradually reintroduced features like as concurrency control, recovery, and buffer management Sauer (2019). Coati is the first system to correctly handle interrupt-driven currency in an intermittent system, with a familiar interface that includes synchronous computing activities and asynchronous interrupt-driven events, all of which are robust to intermittent operation. Coati supports two alternative serialization algorithms for events, tasks, and transactions, allowing sequences of tasks to be atomic with regard to events Ruppel and Lucia (2019).

5.5 Concurrency Control

Concurrency control is a feature of a (DBMS) that allows several users to access a database at the same time. It prohibits two users from simultaneously changing the same record and serializes transactions for backup and recovery. Concurrent transactions may produce conflicts, and simultaneous executions of transactions in a multi-user database system may have an impact on data integrity and consistency, resulting in lost changes, uncommitted data, and inconsistent retrievals. It may avoid this conflict which result in an inaccurate overall result, even though each transaction is valid when run separately by using concurrency management techniques that give a serialization order among competing transactions Hamied (2018). Concurrency control has been actively investigated for several years, with two-phase locking being used as a standard solution for transaction processing. However, the system is saturated with conflicts due to frequent rollbacks, long time waiting and blocking, and a high rate of aborted transactions, resulting in deadlock. A hybridized concurrency management strategy that combines two phase locking and timestamp ordering was proposed by the researcher Christopher and Kabari (2020). Conflicts are unavoidable when different microservices operate as participants and concurrently execute the same set of distributed transactions, and they have a significant impact on the performance of concurrent transactions. A unique concurrency control mechanism based on the 2PC in this part is presented to reduce the chance of conflict between concurrent transactions Fan et al. (2020).

5.5 Evaluated Proposed System

5.5.1 Distributed Database

The study determined how to implement distributed databases in a distributed environment with both homogeneous and heterogeneous data types, each with its own set of procedures for dealing with them Mhawes and Ali Hassan (2020). In other study reviewed researcher proposes an intelligent index technique for multilayer distributed database association rules based on fuzzy correlation fusion clustering analysis. The multi-tier distributed database association rule intelligent index is created by combining it with the distributed structure reorganization method of the multi-tier distributed database association rule storage nodes in order to achieve the optimal design of the multi-tier distributed database association rule intelligent index algorithm Peng et al. (2020). The proposed hybridized system combines basic two-phase locking with Thomas Write Rule, a recent approach for timestamp ordering proposed by the researcher Christopher and Kabari (2020).

5.5.2 Concurrence Transaction Model

The surfing concurrence transaction model proposed by the researcher to make possible allowing users to access data in the ACID (Atomicity, Consistency, Isolation and Durability) properties way by proposing an effective transaction model for key-value NoSQL databases including Redis. Previous researcher proposed model's architecture has six functional components, six queues, and six transaction objects that make up the surfing concurrency transaction model. Transaction client, transaction manager, transaction sorter, transaction dispatcher, transaction executor, and actors are the six functional components. The waiting queue, the waiting read queue, the waiting write queue, the waiting execution queue, the executing queue, and the finished execution queue are the six queues Li and Gu (2018).

5.5.3 Transaction Process

5.5.3.1 Transaction Initiator

An active participant in the process is the transaction initiator. It notifies the coordinator to execute commit or rollback actions on the transaction group in addition to initiating transaction group formation and executing the local transaction is the process of transaction initiator proposed by the research from the evaluated research Fan et al. (2020).

5.5.3.2 Transaction Actor

In distributed transaction processing, the transaction actor is a support role. Its primary responsibilities include adding the micro-business service's module to the distributed transaction group and performing local transactions according to the coordinator's instructions is another process of transaction for actor proposed by the researcher Fan et al. (2020).

6. Implication and Conclusion

Most distributed computer systems performed one or more interdependent or concurrent transactions. Concurrency control techniques such as Locking, Timestamps, and Optimistic Concurrency Control, and prevent conflict, can be used to manage concurrent transactions. The deadlock problem occurs in transactions when four requirements are met: mutual exclusion, hold and wait, no pre-emption, and circular wait Hamied (2018). The use of cache-based query optimization to increase response time is a good alternative since it saves a lot of money and effort, especially when implementing a cache at each site instead of a single network-based cache. It has been discovered that caching is now used in homogenous distributed databases. In terms of future research, the presented approach may be used to implement heterogeneous distributed databases Rana et al. (2018). In the comparison of Relational (MySQL) and NoSQL (MongoDB), MongoDB is a schema-less database with the ability to run in a distributed environment. It is ideally suited to instances when data is not precisely organized, and handling is not difficult. It may also be used in cloud services and can easily grow horizontally utilizing sharding Hamied (2018). Rather of competing, SQL and document-based databases are complementary. MongoDB is a schema-less database that may be used in a variety of applications a dispersed environment it is well positioned in situations when the data is not strictly structured and not complex to handle. MongoDB is a strong competitor/supplement to MySQL. Compared to MySQL, it outperforms SQL in various areas and scenarios. As a result, it becomes a popular database. Deari (2018). MongoDB performs better other databases in CRUD operations, especially when dealing with huge amounts of data and multiple users Pandey (2020). This study identified that several relevant studies were examined, and practically all of them concluded that NoSQL databases are a superior solution for big cloud-based applications.

7. Limitation and Future Research

Future improvements and applications currently, cache is enabled. Only homogeneous distributed databases are suitable for based query optimization. This technology may be used in the future. implemented for a variety of database types Rana et al. (2018). Future researchers also will be able to examine and improve the research's limitations in comparable investigations. Another recommendation is also study on a distributed database for relational (MySQL) and NoSQL (MongoDB), as well as the implementation and performance of ACID transaction in NoSQL (MongoDB).

References

Christopher G., and Kabari L. Hybridized Concurrency Control Technique for Transaction Processing in Distributed Database System. *Artificial Life and Robotics*. vol.9, pp. 118-127, 2020.

Deari R., Zenuni X., Ajdari J., Ismaili F., and Bujar Raufi, Analysis and Comparision of Document-Based Databases with Relational Databases: MongoDB vs MySQL. *International Conference on Information Technologies (InfoTech)*. pp. 1-4, 2018.

Fan P., Liu J., Yin., Wang H., Chen X., and Sun H. 2PC*: A Distributed Transaction Concurrency Control Protocol Of Multi-Microservice Based On Cloud Computing Platform. *Journal of Cloud Computing*. vol.9. 10.1186/s13677-020-00183-w. 2020.

Fresnido A., and Barsaga A. Transaction Log Analysis of OPAC Searches in An Academic Library: *Basis for OPAC Interface Improvement.* Vol.39, no. 39. 2019.

Hamied S., ARhoma A., and Alhaweiaj N. Transaction Management and Concurrency Control. *Scientific Journal of Faculty of Education, Misurata University-Libya*, vol.1, no.10. 2018.

Haubenschild M, Sauer C., Neumann T., and Leis V. Rethinking Logging, Checkpoints, and Recovery for High-Performance Storage Engines. *In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20).* Association for Computing Machinery, New York, NY, USA, pp. 877–892. DOI:https://doi.org/10.1145/3318464.3389716. 2022.

Huang J., Mozafari B., and Schoenebeck G., and Wenisch T., A Top-Down Approach to Achieving Performance Predictability in Database Systems. Pp. 745-758. 10.1145/3035918.3064016. 2017.

Lee G. and Fortes J. Improving Data-Analytics Performance Via Autonomic Control of Concurrency and Resource Units. *ACM Trans. Auton. Adapt. Syst*. vol.13, no.3, article 13. pp. 25.. 2019.

Li G., and Gu J. A Surfing Concurrence Transaction Model for Key-Value NoSQL Databases. *Journal of Software Engineering and Applications.* 11. 467-485. 10.4236/jsea.2018.1110028. 2018.

Mhawes A., and Hassan M. Methods to Implement Homogeneous and Heterogeneous Distributed Database. Available at: www.noveltyjournals.com. 2020.

Mansouri Y., and Ali Babar M. The Impact of Distance on Performance and Scalability of Distributed Database Systems in Hybrid Clouds. https://arxiv.org/abs/2007.15826. 2020.

Marungo, F. A Primer on NoSQL Databases for Enterprise Architects: The CAP Theorem and Transparent Data Access with MongoDB and Cassandra. 10.24251/HICSS.2018.583. 2018.

Pandey R. Performance Benchmarking and Comparison of Cloud-Based Databases MongoDB (NoSQL) Vs MySQL (Relational) using YCSB. 2020. 10.13140/RG.2.2.10789.32484.

Peng H., Yang S., Liu Q., Peng Q., Li Q., *Intelligent Indexing Algorithm for the Association Rules of a Multi-Layer Distributed Database, Engineering Intelligent Systems*, vol. 28 no. 4, pp. 229-240, 2020.

Rana I., Aslam S., Sarfraz M., and Shoaib, U. Analysis of Query Optimization Components in Distributed Database. *Indian Journal of Science and Technology*. 11. 1-10. 10.17485/ijst/2018/v11i18/122267. 2018.

Ruppel E. and Lucia B. Transactional Concurrency Control for Intermittent, Energy-Harvesting Computing Systems. *In Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2019).* Association for Computing Machinery, New York, NY, USA, 1085–1100. DOI:https://doi.org/10.1145/3314221.3314583. 2019.

Sauer, C. Modern Techniques For Transaction-Oriented Database Recovery. In: Grust, T., Naumann, F., Böhm, A., Lehner, W., Härder, T., Rahm, E., Heuer, A., Klettke, M. & Meyer, H. (Hrsg.), BTW 2019. *Gesellschaft für Informatik, Bonn*. pp. 487-496., 2019.

Villegas-Ch, W., Palacios-Pacheco, X., Luján-Mora, S. A Business Intelligence Framework for Analyzing Educational Data. *Sustainability*. vol.12, no.14, 5745, 2020.

Yu X., Yu X., Pavlo A., and Devadas S. Taurus: lightweight parallel logging for in-memory database management systems. Proc. VLDB Endow. vol.14, no.2, pp. 189–201. October 2022.

Biography

Mary Jane C. Samonte has a double bachelor's degree in computer education and information technology. She also has two post graduate degree; Information Technology and Computer Science. She finished her Doctor in IT with a study focusing in Deep Learning. She has a wide range of research interests that are centered around educational technologies, gamification, mobile and ubiquitous learning, digital game-based learning, artificial intelligence in education, e-health, assistive technology, natural language processing, green computing and data analytics-based studies.

Sasky A. Samonte, is a web application developer in iClick Media Pte. Ltd. In Singapore. He earned his Bachelor of Technology Major in Computer Engineering Technology at Technological University of the Philippines in Cavite branch with several years of experience in IT industry. He has an extensive experience in web and system development and programing, he drives the team to oversee the day today performance and efficiency in delivering an excellent system to provide an outstanding service to clients, he is a coordinator lead of the business. Furthermore, he was able to develop different systems for the business, schools, and governments. More recently he has been pursuing his master's degree in technology at Mapúa University of at Manila Philippines to further develop and enhance his skills in Technology and obtain a broad understanding of his field of specialization.