Towards Comprehending Energy Consumption of Database Management Systems - A Tool and Empirical Study

Hemasri Sai Lella*
Research in Intelligent Software & Human Analytics Lab
Indian Institute of Technology Tirupati, India
cs20b020@iittp.ac.in

Rajrupa Chattaraj*
Research in Intelligent Software & Human Analytics Lab
Indian Institute of Technology Tirupati, India
cs22s504@iittp.ac.in

Sridhar Chimalakonda†
Research in Intelligent Software & Human Analytics Lab
Indian Institute of Technology Tirupati, India
ch@iittp.ac.in

Kurra Manasa‡
Research in Intelligent Software & Human Analytics Lab
Indian Institute of Technology Tirupati, India
cs20b019@iittp.ac.in

ABSTRACT

In the dynamic landscape of contemporary data-driven technologies, software systems depend significantly on vast datasets and ongoing data center operations that utilize diverse database systems to facilitate computationally intensive tasks. The management of vast amounts of data also introduces challenges related to energy efficiency. With the growing concern over energy consumption in software systems, the selection of a *Green* database system for its energy efficiency becomes crucial. While various software components have been scrutinized for their energy consumption, there exists a gap in the software engineering literature concerning the energy efficiency of database management systems. To bridge this gap, we performed an empirical study to investigate the energy consumption of queries associated with popular database systems namely MySQL, PostgreSQL, MongoDB, and Couchbase. Our assessments, performed on three commonly used datasets, uncover substantial variations in the energy consumption of these database systems. The study suggests a potential need for optimizing energy usage in various database systems, enhancing developer awareness of the impact of running queries on energy consumption. This empowers them to make informed, sustainable choices, warranting further research in this area.

CCS CONCEPTS

• **Information systems → Database utilities and tools**.

KEYWORDS

Energy consumption, MySQL, MongoDB, PostgreSQL, Couchbase

ACM Reference Format:
Hemasri Sai Lella, Rajrupa Chattaraj, Sridhar Chimalakonda, and Kurra Manasa. 2024. Towards Comprehending Energy Consumption of Database Management Systems - A Tool and Empirical Study. In *28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024), June 18–21, 2024, Salerno, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3661167.3661174

*Both authors contributed equally to this research - Technical Work, Writing
†Contribution - Idea, Conceptualization, Supervision, Writing
‡Contribution - Technical Work

EASE 2024, June 18–21, 2024, Salerno, Italy

ACM ISBN 979-8-4007-1701-7/24/06
https://doi.org/10.1145/3661167.3661174

1 INTRODUCTION

Data plays a pivotal role in deriving insights, informed decision-making processes, and fueling advancements in various fields, emphasizing its fundamental importance in today's information-driven landscape. Data generation has surged at an unprecedented rate, with daily estimates reaching approximately 0.33 zettabytes (around 10^{12} GB)[1]. Effectively handling this vast amount of data poses a considerable challenge, leading to the establishment of numerous 24/7 operational data centers. However, the substantial energy consumption of these data centers is a growing concern [2]. In 2023, data centers are estimated to account for around 3% of the world's total electricity consumption[2]. As data centers continue to proliferate across various sectors and industries, it becomes imperative to scrutinize the heart of these centers—the databases—from an energy consumption standpoint [21]. To address this, Microsoft constructed the underwater data centers, leveraging the cold sea subsurface to mitigate the heat generated by these data centers, thereby achieving energy-efficiency[3].

As data continues to grow, effectively handling it through software like database systems becomes more crucial. Database systems are a fundamental component of the software industry, serving as repositories for storing and managing various types of data. They are utilized across various software development stages, from initial application development to deployment and maintenance[23]. A variety of popular database systems, including various SQL and NoSQL database systems are utilized globally for a wide range of tasks such as data storage and retrieval, web applications, and data security and compliance. SQL database systems are often preferred for applications with complex queries and structured data [42], while NoSQL database systems are chosen for their flexibility and scalability in handling diverse and large datasets[14].

Numerous studies [6, 10, 24] in the literature compare the performance and scalability of SQL and NoSQL database systems. Agarwal

[1]https://explodingtopics.com/blog/data-generated-per-day#
[2]https://datacentremagazine.com/articles/efficiency-to-loom-large-for-data-centre-industry-in-2023
[3]https://news.microsoft.com/source/features/sustainability/project-natick-underwater-datacenter/

et al. [1] demonstrated through a comparative study that NoSQL database systems perform better by an average factor of 10 than SQL database systems, highlighting the significant impact of database system choice on application performance. Considering this performance difference, it is crucial to assess the disparity in energy efficiency between SQL and NoSQL database systems. While NoSQL database systems are lightweight and scalable, Li and Manoharan [29] reported in a study that SQL database systems outperformed them in certain operations such as DELETE, key fetching, and READ operations. With growing concerns about sustainability and energy efficiency, it is essential to explore the energy consumption differences between SQL and NoSQL systems to optimize resource allocation in data center environments [35].

While the conventional emphasis in the realm of database systems has predominantly centered on achieving rapid processing and scalability [26, 29], a limited number of studies in the literature have delved into the energy-aware query optimization [19, 46]. Additionally, there has been significant oversight in quantifying the energy consumption associated with queries written in a specific database system constituting the majority of operations that a database performs. Moreover, although tools such as pyJoules[4], RJoules [12], jRAPL [30], and CodeCarbon[5] have been available to measure the energy consumption of specific programming languages, *there is currently no tool that can support in measuring the energy consumption of database queries.* In this regard, our initial step involves analyzing and quantifying the energy consumption linked to queries utilizing various database systems which might aid to select an efficient database system.

Through this paper, we introduce *DBJoules*, a tool designed to measure the energy consumption of database queries. Leveraging *DBJoules*, this paper conducts an in-depth empirical analysis of two SQL: MySQL, PostgreSQL and two NoSQL: MongoDB, and Couchbase database systems. The empirical evaluation focuses on the energy consumption of these database systems, particularly queries supporting CRUD operations and basic JOIN operations. *To the best of our knowledge this is the first empirical study comparing queries of various database systems, from an energy consumption perspective.*

The paper makes the following contributions.

- An open-source tool *DBJoules*[6], designed to measure the energy consumption of queries across four database systems, i.e., MySQL, PostgreSQL, MongoDB, and Couchbase. The tool measures the energy by gathering the information about CPU and RAM usage, by utilizing *psutil* package[7].
- An empirical study to compare queries associated with different database systems concerning two SQL and two NoSQL systems. For this we formulate the below research questions.
  - **RQ1:** Which type of database system, SQL or NoSQL, is more efficient in terms of energy consumption and run-time taken for the given queries?
  - **RQ2:** Which is the efficient database system among the SQL systems considered, MySQL or PostgreSQL, in terms of energy and run-time consumed?
  - **RQ3:** Which is the efficient database system among the NoSQL systems considered, MongoDB or Couchbase, in terms of energy and run-time consumed?
- An analysis of correlation between energy consumption and run-time in database queries.

Table 1: Details of Database systems and Connection Packages considered for the Empirical Study

Database System	version	Database Connection Package	version
MySQL	8.0.34	mysql-connector-python	8.0.32
PostgreSQL	15.3	psycopg2	2.9.6
MongoDB	5.0.12	pymongo	4.3.3
Couchbase	7.2.5325	couchbase	4.1.7

2 BACKGROUND

A Database Management System (DBMS) is a software system designed to facilitate the easy management of a database. It provides users with the capability to access and interact with the underlying data within the database. These interactions span from basic data queries to the definition of database schemas. The primary goal of a DBMS is to facilitate secure and concurrent interactions with databases, ensuring that users can work simultaneously without interfering with one another, while also upholding data integrity. This is often achieved through the implementation of ACID properties [15], although the exact consistency model may vary depending on the DBMS being used. Some NoSQL (non-relational) database systems may adopt less stringent transactional models compared to traditional SQL (relational) database systems, allowing for some degree of data inconsistency [27]. Relational and non-relational database systems are two major categories of database management systems utilized for storing, retrieving, and analyzing data.

Relational Database Systems (SQL): These database systems organize data into tables with rows and columns and unique identifying keys to establish connections between data in different tables. They are specifically designed for storing structured data, such as names, dates, and quantities, that can be standardized within a table. SQL, or Structured Query Language, is the predominant programming language for interacting with relational database systems. MySQL[8] and PostgreSQL[9] were chosen due to their extensive usage [16] and unique advantages in the relational database systems domain. MySQL is recognized for its performance and scalability, while PostgreSQL boasts advanced features such as support for complex data types and transactions [45].

Non-relational Database Systems (NoSQL): In contrast, non-relational database systems eschew the table-based storage model and instead adopt formats that suit the specific type of data they handle. They are tailored for managing unstructured data. Unlike relational databases, which adhere to a tabular structure, there exists a variety of non-relational database models. One common approach involves Key-Value stores, where data is assigned a unique identifier for retrieval and sorting purposes. MongoDB and Couchbase were chosen for their leading roles in the NoSQL landscape, each offering unique features to manage unstructured data effectively.

[4]https://pypi.org/project/pyJoules/
[5]https://codecarbon.io
[6]https://github.com/rishalab/dbms_energy_consumption/
[7]https://github.com/giampaolo/psutil
[8]https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html
[9]https://www.postgresql.org/docs/current/intro-whatis.html

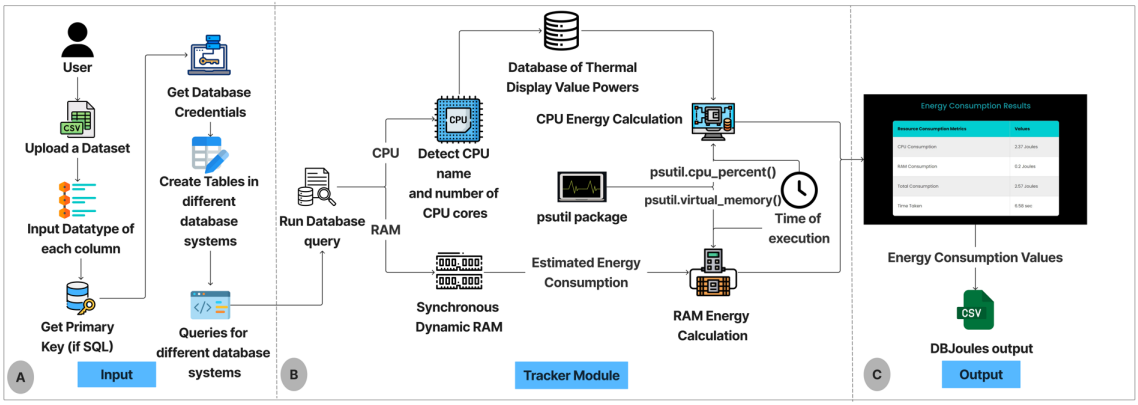

Figure 1: Workflow Diagram of *DBJoules*: A describes the input functionalities of *DBJoules*, B describes the internal workflow of *Tracker Module*, C describes the output derived from *DBJoules*.

MongoDB's[10][8] document-oriented approach suits flexible scalability, while Couchbase's[11] key-value model, coupled with robust caching, ensures high performance, making them both ideal for comparative analysis [29]. All selected database systems are open-source, and Table 1 presents the respective versions of database systems and connection packages utilized in this study.

3 *DBJOULES*

3.1 Overview

While tools exist for measuring the energy consumption of programming languages [12, 43] and machine learning models[12], there has been a notable absence of a tool specifically designed for measuring the energy consumption of database queries. *DBJoules* is the first tool to quantify the energy consumption associated with queries across various database systems. It supports four database systems chosen for their widespread popularity and extensive usage[13]. These include two SQL database systems, namely MySQL and PostgreSQL, and two NoSQL database systems, MongoDB and Couchbase. The tool currently supports fundamental queries: SELECT, INSERT, DELETE, UPDATE and JOIN. It offers a web-based user interface, facilitating tasks such as importing datasets, executing queries, and providing real-time energy consumption and runtime values. While many tools [12, 30] depend on Intel-RAPL, restricting them to Intel systems, *DBJoules* is compatible with both Intel and AMD processors.

DBJoules utilizes the *psutil* package at its core to measure the energy consumed by querying database systems by fetching the CPU utilization and memory allocation associated with database systems. *psutil* (process and system utilities)[14] is a python cross-platform library that retrieves information about running processes and system utilization by providing a set of system related functions which return statistics about system memory (RAM) usage and CPU utilization. It performs system monitoring, profiling and limiting process resources and management of running processes.

[10]https://www.mongodb.com/company
[11]https://docs.couchbase.com/home/index.html
[12]https://codecarbon.io
[13]https://survey.stackoverflow.co/2023/#section-most-popular-technologies-databases
[14]https://github.com/giampaolo/psutil

Multiple studies [9, 20] and software projects[15] have demonstrated the reliability of this library for measuring energy consumption of ML models and remote live forensics. *DBJoules*, developed in Python, uses various python packages for establishing connections with database systems such as *mysql-connector-python, psycopg2, pymongo* and *couchbase*, as detailed in Table 1.

3.2 WorkFlow

The workflow of *DBJoules*, illustrated in Figure 1, initiates with the user uploading a dataset in a `.csv` file. The tool establishes connections with database systems through the designated connection packages and identifies the columns present in the dataset. Users are prompted to specify data types of columns and select the primary key column if present. Database system credentials, including usernames, passwords, and database names, are provided by the user for each database system. Tables are then created in all four database systems (Figure 1A). Subsequently, users formulate input queries for all four database systems, and *DBJoules* computes the energy consumption using its *tracker module* (Figure 1B).

The tracker module initially identifies the CPU model and the number of processors on the host machine to determine thermal design power (TDP) values. These TDP values, indicating average power consumption for specific CPU models, are obtained from a dataset compiled by Budennyy et al. [9], covering 3279 distinct processors from Intel and AMD. In cases where the exact CPU model name cannot be found, the module uses pattern matching to identify the closest match. Otherwise, it assigns a constant value of 100, following the approach by Maevsky et al. [32]. The module retrieves system-wide CPU utilization as percentage through the `psutil.cpu_percent()`[16] function, dividing this value by the number of CPU cores to determine the average CPU utilization percentage. Using this information, the module calculates CPU energy consumption based on Equation 1.

$$E_{CPU} = TDP \times W_{CPU} \times t \quad (1)$$

where, E_{CPU} denotes the overall energy consumption of the CPU during query execution, W_{CPU} stands for the total loading of all processors, and t represents the CPU loading time. CPU load time

[15]https://github.com/google/grr
[16]https://psutil.readthedocs.io/

refers to the duration for which the CPU is actively processing tasks, measured from the start to the completion of a workload. Furthermore, the module calculates the energy consumption associated with RAM, recognizing the significant impact of data read or write activities on database operations. The power consumption of RAM is directly correlated with the power allocated by the presently running process. This is evaluated using `psutil.virtual_memory()`, and the RAM energy consumption is determined using the Equation 2 suggested by [32]:

$$E_{RAM} = 0.375 \times M_{RAM} \times t \quad (2)$$

where, E_{RAM} represents the RAM energy consumption, M_{RAM} denotes the allocated memory in GB as measured by psutil, and t is the loading time. The estimated specific energy consumption of DDR3 and DDR4 modules is 0.375 W/GB [32]. Finally, the tracker module outputs these computed values in a CSV file (Figure 1C).

3.3 Evaluation of *DBJoules*

To evaluate the effectiveness of *DBJoules*, we conducted an empirical experiment focusing on measuring energy consumption within the MySQL database system and validated our results against an established benchmark Intel-RAPL [25]. Utilizing the Countries States Cities dataset, a popular dataset sourced from Kaggle[17], the experiment was conducted on an Intel Xeon Gold 6226R CPU with specifications including a frequency of 2.90GHz, 16 cores, and 128 GB RAM. *DBJoules* is primarily developed for Windows systems. To empirically validate its findings under the same environment as Intel-RAPL, we developed Linux-compatible scripts following the methodology of *DBJoules* for MySQL database system. Ubuntu `20.04.6 (LTS)` served as the operating system for the setup. These scripts have been included in the replication package for accessibility. Energy consumption and run-time values are validated against Intel-RAPL findings and presented in Table 2. On average, a difference of 7.9% in energy consumption measurements was observed, potentially influenced by various factors such as noise and background processes, as discussed in Section 6.

In order to statistically test the difference between results obtained from Intel-RAPL and *DBJoules*, we incorporated Spearman's rank coefficient test [39]. The correlation values in Table 2 indicate a positive correlation for energy consumption and run-time values between *DBJoules* and Intel-RAPL, except for the energy consumption values of Join queries. Except for the p-value obtained for run-time of the DELETE query (0.009), all other p-values (≥ 0.05) indicate no statistically significant difference between the results obtained from Intel-RAPL and *DBJoules*. This validation reinforces the effectiveness and reliability of *DBJoules* in accurately assessing the energy consumption of database queries.

4 EXPERIMENTAL PROCEDURE

In this section, we provide details about the datasets, database queries, and the procedure followed for the empirical study.

4.1 Datasets

To compare the energy consumption of the database systems, we executed the same set of queries on standardized datasets. However, it is important to acknowledge that the energy consumption of database systems might fluctuate depending on the dataset size they process. A database system that is more energy-efficient for smaller datasets may become less efficient as the datasets grow in size. To address these variations, we conducted experiments on three distinct datasets of varying sizes. We selected widely-used datasets from the UCI Machine Learning Repository[18] and Kaggle[19], filtering them specifically for contexts relevant to database-related work. The datasets were converted to CSV format if not already present in this format. Below is a brief overview of the datasets.

Netflix Userbase Dataset: This dataset provides insights into a simulated Netflix userbase, offering details such as subscription type, monthly revenue, and activity. This synthetic dataset, referred to as D1 in this paper, contains 2500 data points with ten attributes and occupies a file size of 1,81,211 bytes.

Online Retail: The dataset encompasses transnational data spanning transactions from 01/12/2010 to 09/12/2011 for a UK-based online retail business specializing in unique all-occasion gifts. This dataset, referred to as D2, includes 541909 data points with eight attributes and occupies a file size of 4,61,33,248 bytes.

International Energy Statistics: The dataset, sourced from the United Nations Statistics Division, presents statistics on the production, trade, conversion, and final consumption; conventional and non-conventional; and new and renewable sources of energy. It consists of 1189482 data points with seven attributes and is referred to as D3 in this paper. It occupies a file size of 13,00,03,960 bytes.

4.2 Database Queries

Database queries are used to perform several operations on database systems, including read/write operations, visualization and statistical operations. In order to compare the energy consumption of these queries while working with the database systems, we have used four different database systems to perform the same set of queries on three different datasets. We examine a few frequently performed queries, i.e., SELECT, INSERT, UPDATE, DELETE and JOIN. These queries represent fundamental operations commonly used in real-world database applications, allowing us to assess various aspects such as data retrieval, modification, and relational operations, ensuring a thorough analysis of the databases' capabilities and efficiency across different scenarios [38]. Moreover, these fundamental queries serve as building blocks for more intricate and advanced query structures [44]. *DBJoules* accommodates various type of JOIN queries for MySQL, PostgreSQL, and Couchbase. However, for MongoDB, the tool exclusively supports the LEFT OUTER JOIN operation, as it is the only supported operation in MongoDB, achievable through *aggregate* and *lookup* operations [11]. Due to this reason, we have incorporated LEFT OUTER JOIN in our empirical study for all database systems. In instances where JOIN is mentioned, it signifies the utilization of LEFT OUTER JOIN.

4.3 Procedure

To address the research questions (RQs), we conducted an empirical analysis on the energy consumption and run-time efficiency of four database systems—MySQL, PostgreSQL, MongoDB, and Couchbase—while executing fundamental queries, namely SELECT, INSERT, DELETE, UPDATE, and JOIN. We employed *DBJoules*, as explained in Section 3, to quantify the energy consumption of these queries. The experiments were performed on two separate systems, whose configurations are detailed below:

[17] https://www.kaggle.com/datasets/tanweerulhaque/countries-states-cities-dataset

[18] https://archive.ics.uci.edu

[19] https://www.kaggle.com/datasets

Table 2: Evaluation of *DBJoules*: Mean values for energy consumption and run-time in Joules and Seconds, respectively

MySQL	Intel-RAPL				DBJoules				Total		Time	
Query	CPU	RAM	Total	Time	CPU	RAM	Total	Time	ρ	p-value	ρ	p-value
SELECT	3.491	0.321	3.812	0.041	3.12	0.006	3.482	0.051	0.309	0.385	0.491	0.150
INSERT	0.219	0.028	0.247	0.003	0.18	0.040	0.22	0.004	0.309	0.385	0.539	0.108
UPDATE	0.197	0.026	0.223	0.003	0.163	0.004	0.167	0.005	0.430	0.214	0.534	0.108
DELETE	0.200	0.026	0.226	0.003	0.150	0.04	0.190	0.004	0.224	0.533	0.770	**0.009**
JOIN	7.678	0.680	8.358	0.086	6.19	0.462	6.652	0.093	**-0.479**	0.161	0.067	0.855

Table 3: D1: Mean values for energy consumption and run-time of four database systems in Joules and Seconds, respectively

Netflix Userbase		System1						System2					
Database	Queries	CPU	RAM	Total	Time	ρ	p-value	CPU	RAM	Total	Time	ρ	p-value
MySQL	SELECT	2.189	0.116	2.305	**6.501**	0.128	0.725	0.512	0.113	**0.626**	3.812	0.183	0.613
	INSERT	1.775	0.108	**1.884**	**6.066**	0.650	0.042	0.37	0.113	**0.481**	3.72	-0.139	0.701
	UPDATE	1.524	0.123	**1.643**	**7.335**	0.340	0.336	0.451	0.121	0.571	4.013	0.559	0.093
	DELETE	1.937	0.116	**2.055**	7.132	0.092	0.801	0.429	0.116	0.545	3.841	0.553	0.097
	JOIN	4.031	0.402	4.434	13.248	0.430	0.214	2.241	0.126	2.366	**3.993**	0.171	0.637
PostgreSQL	SELECT	1.998	0.112	**2.11**	6.982	-0.164	0.650	0.561	0.119	0.68	3.889	0.480	0.160
	INSERT	1.959	0.111	2.069	7.004	-0.151	0.676	0.422	0.12	0.54	3.926	0.113	0.756
	UPDATE	2.132	0.12	2.254	7.475	0.079	0.828	0.472	0.12	0.591	4.013	-0.109	0.763
	DELETE	2.474	0.105	2.581	**6.788**	-0.903	0.00034	0.453	0.122	0.576	4.089	0.417	0.230
	JOIN	3.765	0.399	**4.165**	**13.139**	-0.140	0.700	2.104	0.13	**2.234**	4.05	0.018	0.960
MongoDB	SELECT	4.05	0.38	4.43	12.466	0.236	0.511	0.563	0.112	0.677	**3.677**	0.456	0.185
	INSERT	3.737	0.377	4.113	12.213	-0.080	0.827	0.505	0.118	0.628	3.809	0.128	0.725
	UPDATE	3.287	0.325	3.612	11.117	0.436	0.208	0.367	0.12	**0.488**	3.931	0.543	0.105
	DELETE	2.232	0.177	2.409	6.807	-0.948	3.00E-05	0.435	0.118	0.556	3.899	0.273	0.446
	JOIN	4.105	0.424	4.53	13.765	0.489	0.151	2.144	0.131	2.276	4.083	-0.139	0.701
Couchbase	SELECT	3.574	0.385	3.959	12.528	0.152	0.674	0.529	0.13	0.661	4.022	0.334	0.345
	INSERT	3.436	0.387	3.824	12.63	0.097	0.789	0.405	0.116	0.521	**3.649**	0.195	0.589
	UPDATE	3.771	0.382	4.152	12.41	-0.207	0.567	0.39	0.12	0.51	**3.784**	0.462	0.179
	DELETE	3.692	0.385	4.076	12.562	0.103	0.776	0.375	0.116	**0.492**	**3.696**	0.419	0.228
	JOIN	3.918	0.416	4.332	13.64	0.537	0.110	2.15	0.133	2.283	4.018	-0.304	0.393

Table 4: D2: Mean values for energy consumption and run-time of four database systems in Joules and Seconds, respectively

Online Retail		System1						System2					
Database	Queries	CPU	RAM	Total	Time	ρ	p-value	CPU	RAM	Total	Time	ρ	p-value
MySQL	SELECT	2.944	0.354	3.299	11.895	0.794	0.006	1.26	0.201	1.459	4.941	0.195	0.590
	INSERT	3.604	0.272	3.878	14.243	0.188	0.603	1.259	0.17	1.428	4.385	0.261	0.466
	UPDATE	4.234	0.285	4.518	16.137	0.644	0.044	1.637	0.215	1.851	5.238	0.395	0.258
	DELETE	3.899	0.248	4.146	14.260	0.334	0.345	1.611	0.203	1.816	5.032	0.711	0.021
	JOIN	4.640	0.589	5.233	16.119	0.505	0.137	1.360	0.158	1.517	4.518	-0.442	0.201
PostgreSQL	SELECT	3.067	0.318	3.386	11.006	0.745	0.013	1.241	0.167	1.405	4.409	0.354	0.316
	INSERT	3.519	0.27	**3.792**	14.111	0.042	0.907	1.161	0.157	1.317	4.281	0.779	0.008
	UPDATE	4.251	0.247	4.5	14.685	-0.006	0.987	1.22	0.156	1.376	4.352	0.444	0.199
	DELETE	3.796	0.219	4.014	13.214	0.418	0.229	1.043	0.153	1.194	4.11	-0.079	0.828
	JOIN	3.623	0.461	4.085	**13.633**	0.539	0.108	0.815	0.126	**0.939**	**3.945**	-0.249	0.487
MongoDB	SELECT	3.185	0.31	3.493	10.703	0.721	0.019	1.122	0.151	1.275	4.432	0.122	0.738
	INSERT	4.043	0.275	4.317	14.079	0.515	0.128	1.215	0.153	1.368	4.536	0.450	0.192
	UPDATE	4.128	0.258	4.387	14.954	0.648	0.043	1.193	0.146	1.337	4.134	0.665	0.036
	DELETE	4.153	0.219	4.371	12.944	0.703	0.023	1.037	0.14	**1.179**	4.081	0.354	0.316
	JOIN	4.980	0.805	5.784	16.737	0.818	0.004	1.234	0.218	1.452	4.577	0.426	0.220
Couchbase	SELECT	2.856	0.294	**3.153**	**10.332**	0.559	0.093	0.953	0.147	**1.101**	**4.324**	0.030	0.933
	INSERT	3.962	0.272	4.235	**13.911**	0.283	0.428	0.984	0.143	**1.129**	**4.129**	-0.158	0.663
	UPDATE	3.376	0.231	**3.607**	**13.849**	0.648	0.043	0.735	0.117	**0.851**	**3.668**	0.683	0.030
	DELETE	3.025	0.196	**3.22**	**11.890**	0.474	0.166	1.321	0.143	1.464	**4.030**	0.607	0.063
	JOIN	3.358	0.477	**3.534**	13.798	0.503	0.138	1.114	0.149	1.264	4.249	0.207	0.567

Table 5: D3: Mean values for energy consumption and run-time of four database systems in Joules and Seconds, respectively

International Energy Statistics		System1						System2					
Database	**Queries**	**CPU**	**RAM**	**Total**	**Time**	ρ	**p-value**	**CPU**	**RAM**	**Total**	**Time**	ρ	**p-value**
MySQL	SELECT	5.379	0.293	5.674	16.376	0.267	0.455	1.893	0.237	2.132	6.195	0.467	0.174
	INSERT	3.68	0.218	**3.899**	**13.315**	-0.236	0.511	1.121	0.161	1.283	4.791	0.018	0.960
	UPDATE	6.453	0.349	6.804	18.577	0.754	0.012	2.436	0.293	2.731	7.392	0.297	0.405
	DELETE	4.132	0.487	4.618	13.778	0.657	0.039	2.324	0.294	2.621	7.457	0.353	0.318
	JOIN	5.178	0.606	5.786	16.299	0.721	0.019	3.154	0.428	3.58	8.634	0.491	0.150
PostgreSQL	SELECT	3.819	0.23	4.048	13.643	0.485	0.156	1.28	0.163	**1.442**	**4.747**	-0.012	0.973
	INSERT	3.936	0.221	4.157	13.363	0.649	0.042	1.24	0.15	1.391	**4.503**	0.294	0.410
	UPDATE	5.13	0.271	5.401	15.319	0.669	0.035	1.466	0.192	1.658	5.348	0.224	0.533
	DELETE	3.573	0.334	3.908	10.905	0.770	0.009	1.322	0.169	1.492	4.966	-0.552	0.098
	JOIN	3.208	0.346	3.555	10.588	0.782	0.008	1.432	0.207	**1.638**	**4.965**	0.498	0.143
MongoDB	SELECT	3.73	0.195	**3.925**	**13.532**	0.321	0.365	1.426	0.152	1.578	5.454	-0.018	0.960
	INSERT	3.991	0.192	4.185	13.329	-0.109	0.763	1.139	0.106	**1.245**	4.862	-0.049	0.894
	UPDATE	3.862	0.19	4.054	13.164	-0.006	0.987	1.116	0.11	**1.226**	**4.955**	0.030	0.934
	DELETE	3.344	0.296	**3.639**	**9.973**	0.721	0.019	1.483	0.103	1.586	5.706	0.347	0.327
	JOIN	4.829	0.642	5.469	13.417	0.879	0.001	2.46	0.353	2.813	7.465	0.406	0.244
Couchbase	SELECT	3.98	0.217	4.199	13.85	0.723	0.018	1.339	0.138	1.477	4.981	-0.158	0.663
	INSERT	3.799	0.191	3.991	13.383	-0.541	0.106	1.393	0.14	1.532	5.051	-0.255	0.476
	UPDATE	3.04	0.29	**3.314**	**9.956**	0.794	0.006	1.504	0.175	1.682	5.46	0.091	0.803
	DELETE	3.768	0.187	3.956	13.313	0.419	0.228	1.132	0.136	**1.269**	**4.911**	-0.049	0.894
	JOIN	3.000	0.315	**3.315**	**9.697**	0.842	0.002	1.621	0.148	1.768	5.416	0.104	0.776

- **System1:** AMD(R) Ryzen 5 3500U CPU with a frequency of 2.10GHz, 4 cores, and 8GB RAM
- **System2:** Intel(R) Core(TM) i5-1135G7 CPU with a frequency of 2.40GHz, 4 cores, and 8GB RAM

Factors such as noise and the impact of background processes can affect energy consumption experiments in a software system. To mitigate potential influences from background processes, we manually terminated all concurrent background processes and executed a script to halt other concurrent tasks. Each query underwent ten executions on all four database systems, following the methodology outlined by Georgiou et al. [18] and Shanbhag et al. [41], to ensure the correctness in energy consumption measurements and the mean value of the energy measurements was recorded. To minimize the potential noise impact, we executed these queries in a randomly shuffled order instead of sequentially. After the execution of each query, we maintained the system in an idle state for 30 seconds to ensure stability and alleviate the impact of power tail states [7]. Energy readings obtained from the *DBJoules* tool were saved in a `DBJoules_output.csv` file after each iteration, including the execution time for each query and the corresponding energy consumption readings for CPU and RAM in Joules. The scripts used for the study, along with the detailed energy consumption values of the tasks for each trial, are accessible at the following URL: https://github.com/rishalab/dbms_energy_consumption/.

5 RESULTS

In this section, we present the results of our empirical study according to the research questions (RQs) guiding this study, as given in Section 1. The results of the empirical experiments are presented in Tables 3, 4, and 5 for datasets D1, D2, and D3, respectively. These tables showcase the average energy consumption and run-time values for the executed query operations.

5.1 Findings for RQ1:

Comparing SQL and NoSQL database systems for efficiency in terms of energy consumption and run-time

On analyzing the mean energy consumption and run-time values for D1, we found that SQL database systems are more energy and run-time efficient as compared to NoSQL database systems in case of System1. Notably, MySQL (SQL) shows 60.42% greater energy efficiency for UPDATE query and 51.97% greater run-time efficiency for INSERT query than Couchbase (NoSQL). Interestingly, on System2, SQL database systems show better energy efficiency for SELECT, INSERT and JOIN queries whereas NoSQL database systems exhibit greater energy efficiency for UPDATE and DELETE queries. NoSQL database systems are more runtime efficient than SQL databases, except for JOIN queries, as depicted in Table 3.

(1) PostgreSQL stands out as the most energy-efficient option for the JOIN query across all datasets for System2 (Intel).
(2) SQL database systems are more energy-efficient for D1, with MySQL outperforming MongoDB by 60.42% on System1 for the UPDATE query. However, as data size increases for D2 and D3, NoSQL databases demonstrate higher energy efficiency.
(3) NoSQL database systems show greater run-time efficiency for most of the operations observed.

In the context of D2, except for INSERT query, NoSQL database systems are more energy efficient in the case of System1, as given in Table 4. The only exception occurred: PostgreSQL (SQL) is 12.16% more energy efficient than MongoDB (NoSQL) for the INSERT query. Except for JOIN query, NoSQL database systems are more run-time efficient on system1. Similarly, on System 2, except for JOIN queries, NoSQL database systems demonstrate greater efficiency in both energy consumption and run-time. Notably, PostgreSQL (SQL) is 35.33% more energy efficient and 13.80% more run-time efficient than MongoDB (NoSQL) for JOIN query.

For D3, except for INSERT query, NoSQL database systems are both energy and run-time efficient than SQL database systems on System1. On System2, except for SELECT and JOIN, NoSQL database systems are more energy efficient. PostgreSQL is 32.36% and 54.24% more energy efficient for SELECT and JOIN queries, respectively. Except for SELECT and INSERT queries, NoSQL database systems are more run-time efficient. PostgreSQL (SQL) is 12.96% more run-time efficient for SELECT query than MongoDB (NoSQL) and 10.84% more efficient than Couchbase (NoSQL) for INSERT query (Table 5).

5.2 Findings for RQ2:

Comparing SQL database systems i.e, MySQL or PostgreSQL for efficiency in terms of energy consumption and run-time

On analyzing the mean energy consumption and run-time values of SQL Database systems i.e, MySQL and PostgreSQL for D1, we found that MySQL is energy efficient for INSERT, UPDATE and DELETE queries whereas PostgreSQL is energy efficient for SELECT and JOIN queries in the case of System1. For System2, except for JOIN query, MySQL is more energy efficient than PostgreSQL. MySQL is more run-time efficient for SELECT, INSERT and UPDATE queries, whereas PostgreSQL is more run-time efficient for DELETE and JOIN queries on System1. On System2, MySQL is more run-time efficient for all queries than PostgreSQL.

(1) MySQL demonstrates higher energy efficiency for the INSERT query, except for D2.
(2) For D1, MySQL surpasses PostgreSQL in both energy and run-time efficiency for most operations. However, for larger datasets D2 and D3, PostgreSQL becomes the preferred choice.

For D2, except for SELECT query (on System1), PostgreSQL is more energy efficient than MySQL for the rest of the queries. PostgreSQL is the most run-time efficient SQL database system for both System1 and System2. For D3, except for INSERT query, PostgreSQL is more energy efficient than MySQL on both the systems. MySQL is more efficient than PostgreSQL on both System1 and System2 for INSERT query. Except for INSERT query, PostgreSQL is more run-time efficient than MySQL on System1. MySQL is 0.35% more run-time efficient for INSERT query on System1, whereas PostgreSQL is more run-time efficient for all queries on System2.

5.3 Findings for RQ3:

Comparing NoSQL database systems i.e, MongoDB or Couchbase for efficiency in terms of energy consumption and run-time

On analyzing the mean energy consumption and run-time values of NoSQL Database systems i.e, MongoDB and Couchbase for D1, we found that, except for UPDATE and DELETE queries, Couchbase is more energy efficient than MongoDB for System1. For System2, except for UPDATE and JOIN queries, Couchbase is more energy efficient than MongoDB for System2. In the case of run-time for D1, except for JOIN query, MongoDB is more run-time efficient than Couchbase for System1. In the case of System2, except for SELECT query, Couchbase is more run-time efficient than MongoDB.

For D2, Couchbase is more energy efficient than MongoDB for all the queries on System1. For System2, except for the DELETE query (MongoDB is 19.46% more energy efficient on System2), Couchbase is more energy efficient than MongoDB. In the case of run-time, Couchbase is more run-time efficient than MongoDB for all the queries on both the systems.

For D3, except for SELECT and DELETE query, Couchbase is more energy efficient than MongoDB for System1. On System2, except for INSERT and UPDATE queries, Couchbase is again more energy efficient than MongoDB. MongoDB shows 27.11% more energy efficiency than couchbase for UPDATE query. Considering run-time efficiency, except for UPDATE and JOIN queries, MongoDB is more run-time efficient than Couchbase on System1. Couchbase is 24.36% and 27.7% more run-time efficient for UPDATE and JOIN query, respectively. On System2, except for INSERT and UPDATE queries, Couchbase is more run-time efficient than MongoDB.

(1) For D1, MongoDB is more run-time efficient for System1 (AMD), whereas Couchbase outperforms for System2 (Intel), except for the SELECT query.
(2) For D2, Couchbase excels in run-time efficiency across all operations on both systems.
(3) For D1 and System1, MongoDB shows 40.89% more energy efficiency than Couchbase for DELETE query.

In the replication package, we have documented standard deviation (SD) values for each query across the four examined database systems. The highest SD for energy consumption was observed in the MySQL-System1-D3 JOIN query (2.758), while the lowest was found in the MongoDB-System2-D2 DELETE query (0.007). Regarding run-time, the highest SD was recorded in the MongoDB-System1-D2 DELETE query (5.607), with the lowest observed in the MySQL-System2-D1 JOIN and PostgreSQL-System2-D1 DELETE queries (0.0). On average, energy consumption exhibited an SD of 0.473, while run-time showed an SD of 0.850. These values offer insights into the variability of energy usage and query execution times across database systems and workloads. The SD values signify the degree of deviation of individual iteration results from the mean value and could be attributed to several factors affecting energy measurements, as discussed in Section 6.

5.4 Correlation between Energy Consumption and Run-Time

As part of the data analysis, we explored the potential correlation between energy consumption and run-time in database queries, incorporating Spearman's rank coefficient [39]. The ρ values presented in Tables 3, 4, and 5 indicate a positive correlation between energy consumption and run-time for most operations (93 out of 120 cases). Notably, 15 cases show strong correlation ($0.70 \leq \rho \leq 0.89$) between energy and time, although with weak low probability values. For instance, for System1 and D3, MongoDB and Couchbase exhibit very strong relations with probability values (p-value=0.001 and 0.002, respectively). We found 21 very weakly correlated cases ($0.0 \leq \rho \leq 0.19$) with high probability values ranging from p-value= 0.509 to 0.960. Most of the sample results fall under weakly and moderately correlated scenarios. For a few cases of negative correlation, probability values are considerably high. For instance, the UPDATE operation in PostgreSQL (D2) and MongoDB (D3) for System1 shows a negative correlation with a very high probability (p-value=0.987). In conclusion, while there is some correlation between energy and run-time efficiency, it cannot be unequivocally asserted that a *faster* database is always the *greener* option.

6 THREATS TO VALIDITY

In this section, we address potential threats to the validity of our empirical study, categorized according to Wohlin et al.[49].

6.1 Construct Validity

In this empirical study on energy measurement, we utilized our tool *DBJoules*, leveraging the *psutil* package. While relying on a specific package may impact construct validity, we addressed this by open-sourcing *DBJoules* to ensure replicability. The values provided by the *psutil* package to *DBJoules*, encompassing system-wide CPU utilization and memory allocation, is susceptible to noise interference and concurrent background processes. Hence, users of this tool are encouraged to manually terminate any background processes. Additionally, the computation of CPU energy consumption relies on TDP values extracted from the dataset in Budennyy et al. [9], which are not real-time and may not precisely reflect the current state of the host machine. Consequently, there might be a margin of error in estimating CPU energy consumption. However, this bias remains consistent across all database systems, mitigating its impact on potential errors. It is important to note that we did not delve into query optimization [31] in this study; rather, we focused on fundamental queries, as discussed in Section 4.2.

6.2 Internal Validity

To address potential threats, including the influence of noise, voltage spikes, and daemons, in our energy consumption experiments, we implemented several measures. Tasks were repeated 10 times, and averages from the recorded values were used for evaluation, aligning with the methodology outlined by Shanbhag et al. [41]. To minimize the impact of unnoticed background processes, we manually shut down all background processes. Additionally, to mitigate the effect of any unnoticed background threads, we introduced random uniform shuffling during experiments. Power tail states [7], where certain system components draw higher power even after completing a process, were addressed by incorporating a 30-second idle period before commencing the next task. Due to hardware constraints, we utilized two systems—one with an Intel processor and another with an AMD processor—both configured with basic specifications. It is important to note that the outcomes may differ based on system configuration and hardware setups.

6.3 External Validity

The energy consumption measurement experiments utilized specific versions of MySQL, PostgreSQL, MongoDB, and Couchbase. As open-source libraries, these database systems frequently undergo optimization and enhancement. Considering different versions of these database systems potentially yielding varied results, comprehensive information about the versions of the libraries and database systems used in this study is provided in Table 1. This transparency facilitates replication of our findings and future comparisons.

The choice of datasets is an external factor that may influence the outcomes of our study. The characteristics of attributes, data types, and dataset sizes can impact the computational workload during query executions, especially considering the specific implementations of various database systems. To address this potential threat, we conducted our experiments using three datasets with differing sizes and attributes. Also, SQL database systems can only handle structured data, whereas NoSQL database systems can also support unstructured data. However, to enable a fair comparison in the empirical experiment, we have incorporated only structured data, as mentioned in Section 4.1.

6.4 Conclusion Validity

A potential threat to conclusion validity in our study could arise from the low statistical power of the tests used to address the research questions (RQs). To mitigate this, we systematically collected and analyzed the data. The study utilized a total of 3.6K data samples, considering combinations of operations, datasets, systems and reruns to address our RQs. Despite our efforts to address the threats, the findings presented in this exploratory study should be regarded as initial insights, laying the groundwork for future research on sustainable database systems in the field of software engineering.

7 DISCUSSION

Through this empirical analysis, we aimed to explore the energy consumption and run-time efficiency of database systems for fundamental operations. To quantify the energy consumption of database queries, we have proposed a tool, *DBJoules*. The existing version of *DBJoules* is limited to the Windows operating system (OS). However we plan to provide support for Linux and mac OS in future. While the tool presently accommodates fundamental queries, including SELECT, INSERT, UPDATE, DELETE, and JOIN, there are plans to incorporate more complex and embedded queries in future.

The results indicate that, in terms of run-time efficiency, NoSQL database systems outperform SQL database systems across most observed operations, except in the case of D1 and System1. Likewise, as we scale up the data, NoSQL database systems demonstrate more energy efficiency than SQL database systems for the majority of operations. The discrepancy in performance between SQL and NoSQL database systems may stem from their distinct optimization approaches. NoSQL database systems often integrate efficient indexing and caching mechanisms to mitigate disk I/O, enhancing query performance [33]. However, our findings from RQ1 suggest that SQL database systems outperform in INSERT and JOIN queries, possibly due to their utilization of optimized algorithms such as nested loop joins and hash joins [17]. Delving deeper into this aspect could provide further insights on these observations.

Results from RQ2 show that PostgreSQL demonstrates both better energy efficiency and run-time efficiency for most operations, particularly in the case of larger datasets (D2 and D3). This performance advantage can be attributed to PostgreSQL's advanced query optimization capabilities, including a sophisticated query planner and optimizer. Results from RQ3 reveal that Couchbase exhibits superior energy and run-time efficiency for most operations across both systems and all three datasets. However, in the case of D3, MongoDB demonstrates better run-time efficiency for most operations. An interesting observation is that a particular database system may be more efficient for certain queries in System1, while for other queries, it might be more efficient in System2. This suggests that the performance of a database system can be influenced by the OS, system configuration, memory settings, and other system parameters. However, further research is needed to make definitive claims in this regard.

Energy consumption measurements can play a crucial role in informing software development practices, particularly in database-intensive applications. By using *DBJoules* to quantify the energy

consumption associated with database queries, developers gain valuable insights into query optimization strategies by identifying resource-intensive queries and suggesting alternative indexing strategies to minimize energy consumption. Additionally, application design decisions, such as data partitioning, caching mechanisms, and asynchronous processing, can be tailored to optimize energy usage based on the observed energy profiles of different database operations. Furthermore, *DBJoules* measurements can influence infrastructure provisioning decisions by highlighting the energy implications of hardware configurations, virtualization technologies, and deployment architectures.

Through the empirical study conducted using *DBJoules*, we have demonstrated that the choice of database system can have a significant impact on energy efficiency. For example, in larger databases, NoSQL systems may exhibit superior energy efficiency owing to their caching mechanisms. Conversely, SQL systems may outperform in specific scenarios, such as PostgreSQL being energy efficient for JOIN operations, leveraging advanced indexing techniques and query optimization algorithms to reduce energy consumption. By incorporating energy efficiency considerations alongside performance and scalability requirements, developers can make informed decisions that strike a balance between computational needs and environmental sustainability.

8 RELATED WORK

Recently, the research community has been actively engaged in finding different approaches to mitigate the energy consumption of diverse software elements. This encompasses aspects such as programming languages and data structures. Couto et al. [13] define a ranking of energy efficiency in programming languages. Pereira et al. [37] conducted a study on the energy consumption of various Java Collection Framework (JFC) implementations. They proposed an energy optimization strategy for Java programs by analyzing the source code's calls to JFC methods. Shanbhag et al. [41] found that the choice of dataframe library during the data pre-processing stage could impact energy consumption by up to 202 times. Verdecchia et al. [48] revealed that modifying datasets alone could lead to a drastic reduction of up to 92.16% in the energy consumption of machine learning algorithms.

A suite of energy measurement tools such as EcoML [22], jRAPL [30], pyJoules [43], and RJoules [12] have emerged with the aim of assisting developers in building more energy-efficient software. Anthony et al. [3] developed a tool named Carbontracker to track the energy consumption and carbon footprint of training Deep Learning models. Similarly, Lannelongue et al. [28] developed a tool named Green Algorithms which allows users to estimate the carbon footprint from computation. The majority of existing tools rely on Intel-driven architectures, primarily due to their dependence on Intel-RAPL. Additionally, language-dependent tools such as pyJoules, jRAPL, and Codecarbon create a gap in quantifying energy consumption specifically for database queries.

A substantial body of literature exists on database management systems, exploring their characteristics, strengths, and limitations, particularly from a hardware perspective. For instance, Matallah et al. [36] conducted a comparative study between MySQL and MongoDB, while Bani et al. [5] discussed the impact of cloud patterns on energy consumption of database systems, utilizing the Power-API tool to track energy at the process level in cloud-based applications. Studies have compared SQL and NoSQL database systems, focusing on aspects such as performance [26], scalability [10, 14] and data handling [47]. However, existing literature assessing energy consumption values in database systems is limited. This study aims to address this gap by quantifying the energy consumption of querying database systems and conducting an empirical analysis of SQL and NoSQL database systems from an energy perspective.

9 CONCLUSION AND FUTURE WORK

In this paper, we have conducted an empirical study, to analyze the energy consumption of SQL versus NoSQL database management systems. In order to quantify the energy consumption of various database systems, we proposed a tool, *DBJoules*. The empirical study focused on evaluating two popular SQL database systems, MySQL and PostgreSQL, along with two NoSQL database systems, MongoDB and Couchbase. Fundamental queries such as SELECT, INSERT, UPDATE, DELETE, and JOIN were analyzed. The findings indicate that there is no single database system that excels in both energy efficiency and run-time efficiency across all operations. The choice of a database system should consider factors such as user requirements, data type, and dataset size. For example, PostgreSQL may be a suitable choice for energy-efficient JOIN operations.

In addition to comparing energy consumption among database systems, this study aims to raise awareness within the research community regarding the significance of energy efficiency in database systems. We believe this endeavor seeks to inspire further investigations and advancements in this field. The study provides a foundation for exploring the impact of various database systems on energy consumption, leading to potential optimizations and *green* database solutions. Understanding the energy consumption profiles of different database systems allows developers and practitioners to align their choices with specific energy efficiency requirements, contributing to sustainable software development practices.

The primary aim of this work is to initiate the investigation of energy consumption in database systems. While this study serves as an initial exploration, we plan to expand it in several ways. Our future endeavors involve incorporating more database systems and conducting the study on a larger scale with extensive datasets and high-configuration systems. In terms of future directions, it is crucial to consider and verify the energy consumption of GPU-accelerated [40] and cloud databases [4, 34]. Furthermore, we intend to incorporate an assessment of the energy consumption linked to disk read/write operations, recognizing its importance in database operations. This research work lays the foundation for future advancements in energy-efficient database systems. Potential research avenues include investigating the integration of *DBJoules* with emerging database architectures such as multi-tenancy, containerization, and edge computing. *DBJoules* measures database energy consumption using software tools, offering detailed insights and easy integration. However, hardware-based tools provide precise and direct measurements. Future research could explore this direction further.

ACKNOWLEDGMENTS

The authors are thankful to Ambati Pooja Sree and Guddeti Namitha Reddy for their involvement during the development of the

preliminary version of *DBJoules*, and to the anonymous reviewers for their constructive comments.

REFERENCES

[1] Sarthak Agarwal and KS Rajan. 2017. Analyzing the performance of NoSQL vs. SQL databases for Spatial and Aggregate queries. In *Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings*, Vol. 17. 4.

[2] Kazi Main Uddin Ahmed, Math HJ Bollen, and Manuel Alvarez. 2021. A review of data centers energy consumption and reliability modeling. *IEEE Access* 9 (2021), 152536–152563.

[3] Lasse F Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. 2020. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *arXiv preprint arXiv:2007.03051* (2020).

[4] Indu Arora and Anu Gupta. 2012. Cloud databases: a paradigm shift in databases. *International Journal of Computer Science Issues (IJCSI)* 9, 4 (2012), 77.

[5] Béchir Bani, Foutse Khomh, and Yann-Gaël Guéhéneuc. 2016. A study of the energy consumption of databases and cloud patterns. In *Service-Oriented Computing: 14th International Conference, ICSOC 2016, Banff, AB, Canada, October 10-13, 2016, Proceedings 14*. Springer, 606–614.

[6] Sneha Binani, Ajinkya Gutti, and Shivam Upadhyay. 2016. SQL vs. NoSQL vs. NewSQL-a comparative study. *database* 6, 1 (2016), 1–4.

[7] James Bornholt, Todd Mytkowicz, and Kathryn S McKinley. 2012. The model is not enough: Understanding energy consumption in mobile devices. In *2012 IEEE Hot Chips 24 Symposium (HCS)*. IEEE, 1–3.

[8] Shannon Bradshaw, Eoin Brazil, and Kristina Chodorow. 2019. *MongoDB: the definitive guide: powerful and scalable data storage*. O'Reilly Media.

[9] Semen Andreevich Budennyy, Vladimir Dmitrievich Lazarev, Nikita Nikolaevich Zakharenko, Aleksei N Korovin, OA Plosskaya, Denis Valer'evich Dimitrov, VS Akhripkin, IV Pavlov, Ivan Valer'evich Oseledets, Ivan Segundovich Barsola, et al. 2022. Eco2ai: carbon emissions tracking of machine learning models as the first step towards sustainable ai. In *Doklady Mathematics*, Vol. 106. Springer, S118–S128.

[10] Rick Cattell. 2011. Scalable SQL and NoSQL data stores. *Acm Sigmod Record* 39, 4 (2011), 12–27.

[11] Antonio Celesti, Maria Fazio, and Massimo Villari. 2019. A study on join operations in MongoDB preserving collections data models for future internet applications. *Future Internet* 11, 4 (2019), 83.

[12] Rajrupa Chattaraj and Sridhar Chimalakonda. 2023. RJoules: An Energy Measurement Tool for R. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2026–2029.

[13] Marco Couto, Rui Pereira, Francisco Ribeiro, Rui Rua, and João Saraiva. 2017. Towards a green ranking for programming languages. In *Proceedings of the 21st Brazilian Symposium on Programming Languages*. 1–8.

[14] Vitor Furlan de Oliveira, Marcosiris Amorim de Oliveira Pessoa, Fabrício Junqueira, and Paulo Eigi Miyagi. 2021. SQL and NoSQL Databases in the Context of Industry 4.0. *Machines* 10, 1 (2021), 20.

[15] Yuetang Deng, Phyllis Frankl, and Zhongqiang Chen. 2003. Testing database transaction concurrency. In *18th IEEE International Conference on Automated Software Engineering, 2003. Proceedings.* IEEE, 184–193.

[16] Paul DuBois. 2013. *MySQL*. Addison-Wesley.

[17] Avrilia Floratou, Umar Farooq Minhas, and Fatma Özcan. 2014. Sql-on-hadoop: Full circle back to shared-nothing database architectures. *Proceedings of the VLDB Endowment* 7, 12 (2014), 1295–1306.

[18] Stefanos Georgiou, Maria Kechagia, Tushar Sharma, Federica Sarro, and Ying Zou. 2022. Green ai: Do deep learning frameworks have different costs?. In *Proceedings of the 44th International Conference on Software Engineering*. 1082–1094.

[19] Binglei Guo, Jiong Yu, Bin Liao, Dexian Yang, and Liang Lu. 2017. A green framework for DBMS based on energy-aware query optimization and energy-efficient query processing. *Journal of Network and Computer Applications* 84 (2017), 118–130.

[20] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the systematic reporting of the energy and carbon footprints of machine learning. *The Journal of Machine Learning Research* 21, 1 (2020), 10039–10081.

[21] Ralph Hintemann and Simon Hinterholzer. 2019. Energy consumption of data centers worldwide. *Business, Computer Science (ICT4S)* (2019).

[22] Stefan Igescu, Giovanni Monea, and Vincenzo Pecorella. [n. d.]. EcoML: A tool to track and predict the carbon footprint of machine learning tasks. ([n. d.]).

[23] Ylber Januzaj, Jaumin Ajdari, and Besnik Selimi. 2015. DBMS as a Cloud service: Advantages and Disadvantages. *Procedia-Social and Behavioral Sciences* 195 (2015), 1851–1859.

[24] Mohammad A Kausar and Mohammad Nasar. 2021. SQL versus NoSQL databases to assess their appropriateness for big data application. *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science)* 14, 4 (2021), 1098–1108.

[25] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K Nurminen, and Zhonghong Ou. 2018. Rapl in action: Experiences in using rapl for power measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)* 3, 2 (2018), 1–26.

[26] Wisal Khan, Teerath Kumar, Cheng Zhang, Kislay Raj, Arunabha M Roy, and Bin Luo. 2023. SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A Systematic Literature Review. *Big Data and Cognitive Computing* 7, 2 (2023), 97.

[27] Douglas Kunda and Hazael Phiri. 2017. A comparative study of nosql and relational database. *Zambia ICT Journal* 1, 1 (2017), 1–4.

[28] Loïc Lannelongue, Jason Grealey, and Michael Inouye. 2021. Green algorithms: quantifying the carbon footprint of computation. *Advanced science* 8, 12 (2021), 2100707.

[29] Yishan Li and Sathiamoorthy Manoharan. 2013. A performance comparison of SQL and NoSQL databases. In *2013 IEEE Pacific Rim conference on communications, computers and signal processing (PACRIM)*. IEEE, 15–19.

[30] Kenan Liu, Gustavo Pinto, and Yu David Liu. 2015. Data-oriented characterization of application-level energy optimization. In *Fundamental Approaches to Software Engineering: 18th International Conference, FASE 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings 18*. Springer, 316–331.

[31] Guy Lohman. 2014. Is query optimization a "solved" problem. In *Proc. Workshop on Database Query Optimization*, Vol. 13. Oregon Graduate Center Comp. Sci. Tech. Rep, 10.

[32] DA Maevsky, EJ Maevskaya, and ED Stetsuyk. 2017. Evaluating the RAM energy consumption at the stage of software development. *Green IT Engineering: Concepts, Models, Complex Systems Architectures* (2017), 101–121.

[33] Divya Mahajan, Cody Blakeney, and Ziliang Zong. 2019. Improving the energy efficiency of relational and NoSQL databases via query optimizations. *Sustainable Computing: Informatics and Systems* 22 (2019), 120–133.

[34] Phan Thi Anh Mai, Jukka K Nurminen, and Mario Di Francesco. 2014. Cloud databases for internet-of-things data. In *2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*. IEEE, 117–124.

[35] Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith, and Jonathan Koomey. 2020. Recalibrating global data center energy-use estimates. *Science* 367, 6481 (2020), 984–986.

[36] Houcine Matallah, Ghalem Belalem, and Karim Bouamrane. 2021. Comparative study between the MySQL relational database and the MongoDB NoSQL database. *International Journal of Software Science and Computational Intelligence (IJSSCI)* 13, 3 (2021), 38–63.

[37] Rui Pereira, Marco Couto, João Saraiva, Jácome Cunha, and João Paulo Fernandes. 2016. The influence of the java collection framework on overall energy consumption. In *Proceedings of the 5th International Workshop on Green and Sustainable Software*. 15–21.

[38] Wittawat Puangsaijai and Sutheera Puntheeranurak. 2017. A comparative study of relational database and key-value database for big data applications. In *2017 International Electrical Engineering Congress (iEECON)*. IEEE, 1–4.

[39] Philip Sedgwick. 2014. Spearman's rank correlation coefficient. *Bmj* 349 (2014).

[40] Anil Shanbhag, Samuel Madden, and Xiangyao Yu. 2020. A study of the fundamental performance characteristics of GPUs and CPUs for database analytics. In *Proceedings of the 2020 ACM SIGMOD international conference on Management of data*. 1617–1632.

[41] Shriram Shanbhag and Sridhar Chimalakonda. 2023. An Exploratory Study on Energy Consumption of Dataframe Processing Libraries. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE, 284–295.

[42] Vatika Sharma and Meenu Dave. 2012. Sql and nosql databases. *International Journal of Advanced Research in Computer Science and Software Engineering* 2, 8 (2012).

[43] Spirals Research Group. 2021. pyJoules: A Python library to capture the energy consumption of code snippets. *University of Lille and Inria* (2021). https://github.com/powerapi-ng/pyJoules

[44] Chanchai Supaartagorn. 2011. PHP Framework for database management based on MVC pattern. *AIRCC's International Journal of Computer Science and Information Technology* (2011), 251–258.

[45] Wojciech Truskowski, Rafał Klewek, and Maria Skublewska-Paszkowska. 2020. Comparison of MySQL, MSSQL, PostgreSQL, Oracle databases performance, including virtualization. *Journal of Computer Sciences Institute* 16 (2020), 279–284.

[46] Yi-Cheng Tu, Xiaorui Wang, Bo Zeng, and Zichen Xu. 2014. A system for energy-efficient data management. *ACM SIGMOD Record* 43, 1 (2014), 21–26.

[47] Sitalakshmi Venkatraman, Kiran Fahd, Samuel Kaspi, and Ramanathan Venkatraman. 2016. SQL versus NoSQL movement with big data analytics. *International Journal of Information Technology and Computer Science* 8, 12 (2016), 59–66.

[48] Roberto Verdecchia, Luís Cruz, June Sallou, Michelle Lin, James Wickenden, and Estelle Hotellier. 2022. Data-centric green ai an exploratory empirical study. In *2022 International Conference on ICT for Sustainability (ICT4S)*. IEEE, 35–45.

[49] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.