Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

A novel approach to database selection using feedforward neural networks

Nidal A. Al-Dmour [1,*], Hani Al-Zoubi [1], Ghazi Al Naymat [2], Hanan Hussain [3]

[1]*Department of Computer Engineering, College of Engineering, Mutah University, Karak, Jordan*
[2]*Artificial Intelligence Research Center (AIRC), College of Engineering and IT, Ajman University, Ajman, United Arab Emirates*
[3]*College of Engineering and IT, University of Dubai, Dubai, United Arab Emirates*

ARTICLE INFO

Article history:
Received 11 May 2024
Received in revised form
12 September 2025
Accepted 1 October 2025

Keywords:
Database selection
Feature selection
Neural network
Dataset generation
Prediction accuracy

ABSTRACT

Selecting an appropriate database is a common challenge for professionals, including web developers and machine learning engineers. Choosing the most suitable database for an application is important for maximizing its performance. However, because many features of different databases overlap, manually predicting the best database is difficult and prone to errors. To address this issue, a new approach is proposed using a Feedforward Neural Network (FFNN) for database selection. This method involves four steps: feature selection, dataset generation, neural network modeling, and database prediction. In the feature selection step, important features of seven major relational databases—MySQL, MS SQL Server, Oracle, IBM DB2, PostgreSQL, SQLite, and Microsoft Access—are gathered through web searches. These features are used to create a ground truth table. During dataset generation, 2,400 combinations of 75 features are generated, and labels for each instance are calculated using a weighted average method. The neural network modeling step involves selecting an optimal feedforward neural network based on its parameters and performance. The network is then trained using the Levenberg-Marquardt backpropagation algorithm. In testing, user input is provided (a selected set of features is fed into the pre-trained network), and the system predicts the best database with a mean squared error (MSE) of 5.16E-14.

1. Introduction

Database types (or models) refer to the structures used to organize and manage data within a database management system (DBMS). Over the past decades, various types of databases have been developed, starting from early systems such as the Information Management System to modern platforms including Oracle, IBM DB2, PostgreSQL, MySQL, SQLite, and Microsoft Access (Abbasi et al., 2024). Based on their usage, databases can be classified into several categories: centralized, distributed, personal, end-user, commercial, NoSQL (including graph, key-value, document, and column databases), operational, relational, cloud-based, and object-oriented databases (Mouhiha and Mabrouk, 2025). While each of these databases serves a specific purpose, this paper focuses on relational databases and their prediction using selected relevant features.

In a traditional row store, each entity is stored in a separate row (Kanade and Gopal, 2013). In contrast, a column store arranges data by storing the attributes of tuples contiguously (Abadi et al., 2008; El-Helw et al., 2011). Thus, row-oriented databases organize information by records, keeping all the data related to a record close together in memory. Popular examples of relational databases include MySQL, SQL Server, PostgreSQL, IBM DB2, Microsoft Access, SQLite, MariaDB, Informix, and Azure SQL.

An example can demonstrate how read and write operations are performed in a relational database. Consider a table named Friends that includes three records: Person A from City X aged 27, Person B from City Y aged 30, and Person C from City Z aged 33. In a row-oriented database, these records are stored on disk sequentially as: Person A, City X, 27; Person B, City Y, 30; Person C, City Z, 33. When a new record is inserted, such as Person D from City W, aged 35, the entry is appended to the end of the sequence.

Relational databases allow fast write operations because new data can be easily appended at the end.

* Corresponding Author.
Email Address: nidal75@yahoo.com (N. A. Al-Dmour)
https://doi.org/10.21833/ijaas.2025.10.017
Corresponding author's ORCID profile:
https://orcid.org/0000-0002-2898-3905

However, read operations tend to be slower. For instance, calculating a value such as the sum of ages requires loading all records into memory. Thus, each type of database has its own advantages and limitations, along with distinct features.

This paper proposes an automated approach for selecting a suitable database based on database characteristics and user requirements. Section 2 reviews related studies on the selection of row-oriented and column-oriented databases through experimental comparisons, as well as approaches based on feature selection. Section 3 describes the proposed methodology, including feature selection, dataset generation, neural network modeling, and database prediction. Section 4 presents the experimental results used to assess the performance of the proposed approach. Finally, Section 5 summarizes the main findings and highlights potential directions for future research.

2. Literature survey

Research on predicting suitable databases based on user input using machine learning techniques remains limited. However, several studies have investigated the prediction or selection of row-oriented and column-oriented databases through experimental analysis under different conditions.

Bousalem et al. (2019) compared a relational database (MySQL) with a NoSQL database (HBase) in terms of runtime and latency using the YCSB framework. They conducted three tests: (1) loading data, (2) running workloads while increasing the number of records with a fixed 10,000 operations, and (3) running workloads while increasing the number of operations with one million records. Their results showed that MySQL had consistently higher runtime during data loading, while HBase outperformed MySQL in most cases. For latency, MySQL was faster in read operations, whereas HBase was more efficient in write operations. In workload runtime, HBase performed better overall, except in read-only workloads.

Yassien and Desouky (2016) carried out a similar benchmark study using YCSB on MySQL, MongoDB, and HBase. They varied the operation count and thread count and applied the Pearson Correlation Test to examine relationships between workload parameters and performance metrics such as throughput, latency, and runtime. Their findings showed that HBase had the highest read latency but performed well in updates, runtime, and throughput, though it was negatively affected by large operation counts. MySQL, by contrast, had the lowest read latency under heavy read loads but exhibited the highest update and write latency, as well as the longest runtime during dataset loading and generation.

Salunke and Ouda (2024) focused on selecting suitable databases for machine learning projects, considering factors such as data volume, scalability, and support resources. They recommended using traditional databases for datasets up to 1 TB and shifting to solutions like Amazon Redshift or Google Big Query for larger datasets.

Earlier, Jarke and Vassiliou (1985) proposed a methodology for selecting database query languages for different user classes. Their approach was based on an interpretation model of database query languages, considering two key aspects: (1) programming language and database theory, and (2) human factors engineering.

3. Methodology

The proposed method consists of the following four main steps. Fig. 1 outlines the pictorial representation of the methodology.

- Feature selection
- Dataset generation
- Neural network modeling
- Database prediction

3.1. Feature selection

Prominent features of the database are searched from the web and literature (Kanade and Gopal, 2013; Kim, 2014; Okman et al., 2011; Kepner et al., 2016). Thirteen categories of features consisting of 75 sub-features are selected based on the review. Selected categories and their features are shown in Table 1.

Thirteen main features were identified for evaluation. These include the type of license, whether open source (freeware) or closed source (proprietary), the level of memory requirements (high or low), operating system compatibility, and fundamental functionalities such as ACID properties, referential integrity, transaction management, record-level locking (RLL), multi-version concurrency control (MVCC), Unicode support, type inference, compression, and scalability. Further categories include interface and schema features such as tables, views, indexing types, and supported data types, as well as database capabilities including union, intersect, except, inner joins, outer joins, nested selects, and merge joins. Partitioning methods such as range, hash, combined range-hash, list, and expression-based techniques were also considered. Access control features were included as well, such as native network encryption, brute-force protection, enterprise directory compatibility, password complexity rules, patch management, unprivileged execution, auditing, resource limits, separation of duties (RBAC), security certifications, and attribute-based access control (ABAC). In addition, other features such as crash recovery, big data handling, data warehousing, and cloud compatibility were taken into account.

Let F represent an instance of a database defined by a set of 75 selected features, expressed as:

$$F = (f(1), f(2), f(3), \ldots f(75))$$

where,

$$f(i) = \left\{ \begin{matrix} 1\,;\ if\ user\ selects\ feature\ f \\ 0; Otherwise \end{matrix} \right\}$$

For prediction purposes, seven commonly used databases were selected, namely MySQL, MS SQL Server, Oracle, IBM DB2, PostgreSQL, SQLite, and Microsoft Access. Ground-truth values were established for these databases based on their most recent versions, without including add-ons, extensions, or external routines. A complete list of the ground-truth values corresponding to the 75 features of the seven databases is provided in Appendix A.

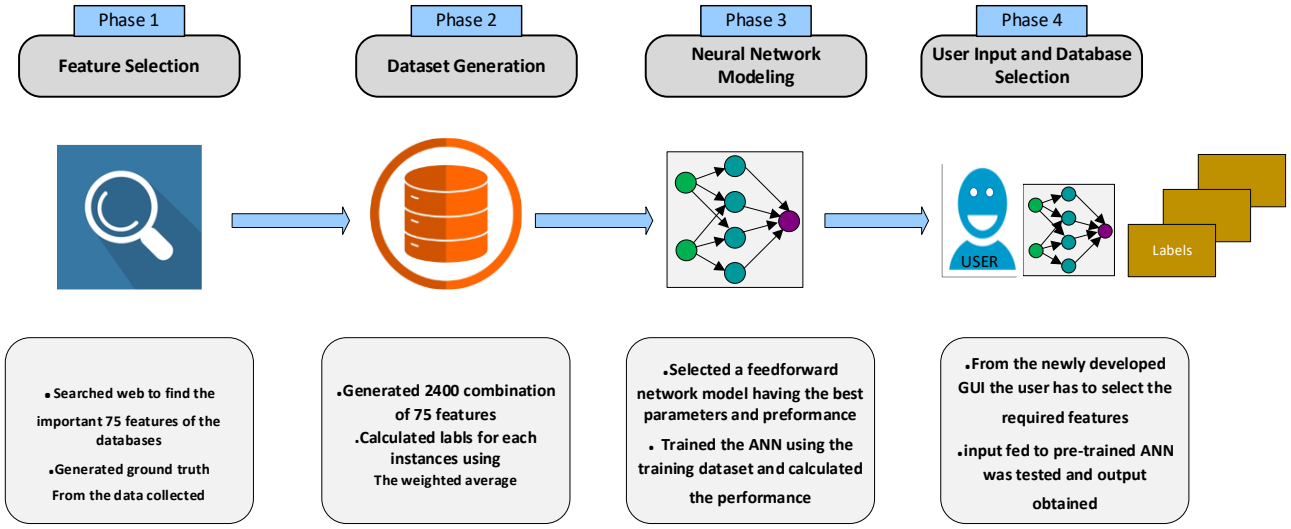

Fig. 1: Proposed methodology stepwise

Table 1: Prominent features and sub-features of databases

Category	Features
Type of the license	Open source, closed source
Memory requirements	High, low
OS support	Windows, macOS, Linux, BSD, Unix, AmigaOS, z/OS, Android, Open VMs
Basic features	ACID properties, referential integrity, transactions, locking (RLL), MVCC, Unicode, type inference, compression, scalability
Interface	API, GUI, SQL
Tables and views	Temporary tables, materialized views
Indexes	B-/B+ tree, R-/R+ tree, hash expression, partial, reverse, bitmap, full-text, spatial, FOT, duplicate index prevention
Data types	Integer, floating point, decimal, string
DB capabilities	Union, intersect, except, inner joins, outer joins, nested selects, merge joins, blobs and clobs, common table expressions, windowing functions, parallel query, system-versioned tables
Partitioning	Range, hash, combined range-hash, list, expression
Objects	Data domain, cursor, trigger, function, procedure, external routine
Access controls	Native network encryption, brute-force protection, enterprise directory compatibility, password complexity rules, patch management, run unprivileged, audit, resource limit, separation of duties (RBAC), security certification, attribute-based access control (ABAC)
Other features	Crash recovery, big data handling, data warehousing, cloud compatibility

3.2. Dataset generation

Different combinations of features were randomly selected and assigned values of either 0 or 1. After comparing the selected features with those in the ground truth data, the databases that matched more closely received higher scores, while those with fewer matches received lower scores. The databases with the highest scores were labeled with a larger percentage, and those with the lowest scores were labeled with a smaller percentage. In this way, 2400 instances were generated, each with its corresponding label, to train the neural network.

Let Scoredb(j) represent the final score obtained for the j-th database. For this calculation, Ft(i) for the j-th database takes a value of 1 or 0 depending on the ground truth data. If the i-th feature of the j-th database equals 1, then Ft(i) is set to 1; otherwise, it is set to 0. The score was calculated for all seven databases, and the corresponding Percentagedb was also determined. These percentage values were then assigned as labels for each database. In total, [2400 × 7] labels were generated corresponding to the [2400 × 75] feature instances. The equations used for calculating scores and percentages are as follows:

$$Scoredb(j) = \sum_{i=1}^{75} ft(i)\ where\ j = 1{:}7$$

$$ft(i) = \left\{ \begin{matrix} 1\,;\ if\ ground\ truth\ \ value\ of\ (f(i), db(j)) = 1 \\ 0; Otherwise \end{matrix} \right\}$$

$$Label = Percentagedb(j) = \frac{Scoredb(j)}{\sum_{j=1}^{7} Scoredb(j)} * 100$$

The training dataset S consists of features and labels. F(1) denotes the set of 75 features for the first instance, and Label(1) represents the corresponding set of seven values assigned to the databases. If n represents the total size of the training dataset, then the structure of S is expressed as:

$$S = \left[\left(F(1), Label(1)\right), \left(F(2), Label(2)\right), \ldots \left(F(n), Label(n)\right) \right]$$

3.3. Neural network modeling

The Feedforward Neural Network (FFNN) was employed to train the generated dataset. The proposed FFNN is composed of four layers. The first is the input layer, which contains 75 neurons, each corresponding to one of the feature vectors. The second and third layers are two identical hidden layers, each consisting of eight neurons and using the hyperbolic tangent sigmoid activation function (tansig). Each layer receives connections from the preceding one. The fourth and final layer is the output layer, which contains seven neurons corresponding to the database labels. This layer uses a purely linear activation function (purelin), and its outputs are used for database prediction. Variants of the FFNN include fitting networks, pattern recognition networks, and cascade forward networks.

Two main functions were applied in the FFNN. The first is the training function, which uses Levenberg-Marquardt backpropagation (trainlm) (Levenberg, 1944; Marquardt, 1963). This algorithm updates the weights and biases in the network. Although it requires more memory than other methods, it is the fastest approach for training feedforward neural networks. Because the algorithm relies on the Jacobian matrix, the performance measure must be either mean squared error (MSE) or the sum of squared errors (SSE). Training terminates when one of the following conditions is satisfied: the maximum number of training epochs reaches 1000, the performance goal approaches zero, the number of validation failures reaches six, the minimum performance gradient falls below 0.0000001, or the maximum mu value reaches 10.

The second is the transfer function, which employs the hyperbolic tangent sigmoid function (tansig) in the hidden layers and the linear function (purelin) in the output layer, as illustrated in Fig. 2. Fig. 3 illustrates the structure of the proposed FFNN (75, 8, 8, 7), which consists of 75 input neurons, two hidden layers with 8 neurons each, and 7 output neurons.

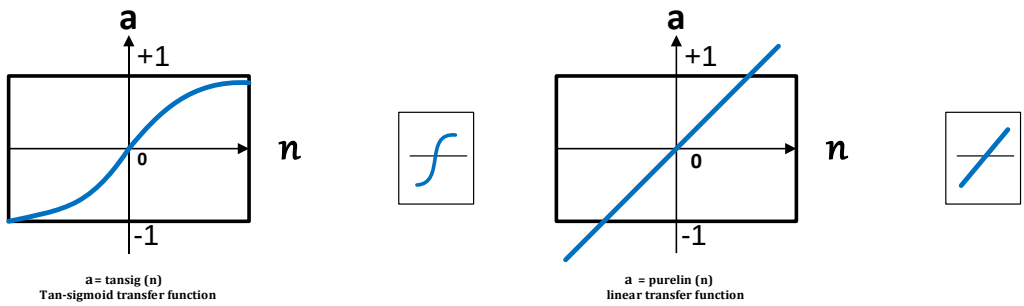

Fig. 2: Graphical representation of the transfer functions used

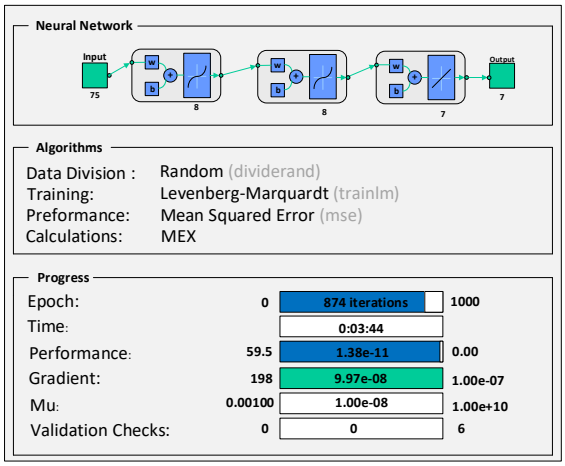

Fig. 3: Proposed FFNN (75, 8, 8, 7)

3.4. Database prediction

The neural network is trained with the given training dataset and labeled with a minimum error function. The final stage of this methodology is the testing part of the modeled neural network using user input. The methodology shown in Fig. 4 summarizes both the training and testing phases of the FFNN. Testing involves the prediction of the most relevant database based on user input. The user inputs are extracted and fed to the FFNN to test. Once the testing is over, the network labels each database with a particular score. Based on the relevance, each database receives a score from high to low scores as shown in Table 2.

4. Result analysis

Thirty-five experiments were conducted by changing the number of hidden layers to as follows:

one, two, and three. Additionally, the number of hidden neurons was shifted to 8 and 10 for all the varying hidden layer sizes.

4.1. Performance measurement: Mean squared error

The performance of the feedforward neural network (FFNN) was assessed using the mean squared error (MSE). As a widely recognized metric in neural network evaluation, MSE quantifies the average squared difference between predicted and actual outputs. The following equation presents the calculation of this performance measure:

$$MSE = \frac{1}{n}\sum_{i}^{n}(target(i) - out(i))$$

where, n is the number of output nodes, target is the desired neural network output, and out is the actual neural network output. MSE obtained for different experiments is given in Appendix B and Table 3.

4.2. Training functions

Four different training functions were evaluated, namely Levenberg-Marquardt backpropagation (trainlm), resilient backpropagation, Bayesian regularization (Burden and Winkler, 2008), and scaled conjugate gradient backpropagation (Møller, 1993). Among these, the trainlm algorithm was found to perform better than the others in terms of mean squared error (MSE) and elapsed time. The best results obtained for the proposed FFNN (75, 8, 8, 7) using different training functions and ten-fold cross-validation are presented in Table 4.

Table 2: The score achieved by each database

Database	Score predicted by ANN	Score predicted by average method
Oracle DB	18.2373	19.3548
MySQL	19.9824	17.7419
MS SQL Server	16.1902	16.1290
PostgreSQL	21.2966	20.1613
IBM DB2	15.2503	14.5161
MS Access	4.1646	2.4194
SQLite	4.8860	9.6774

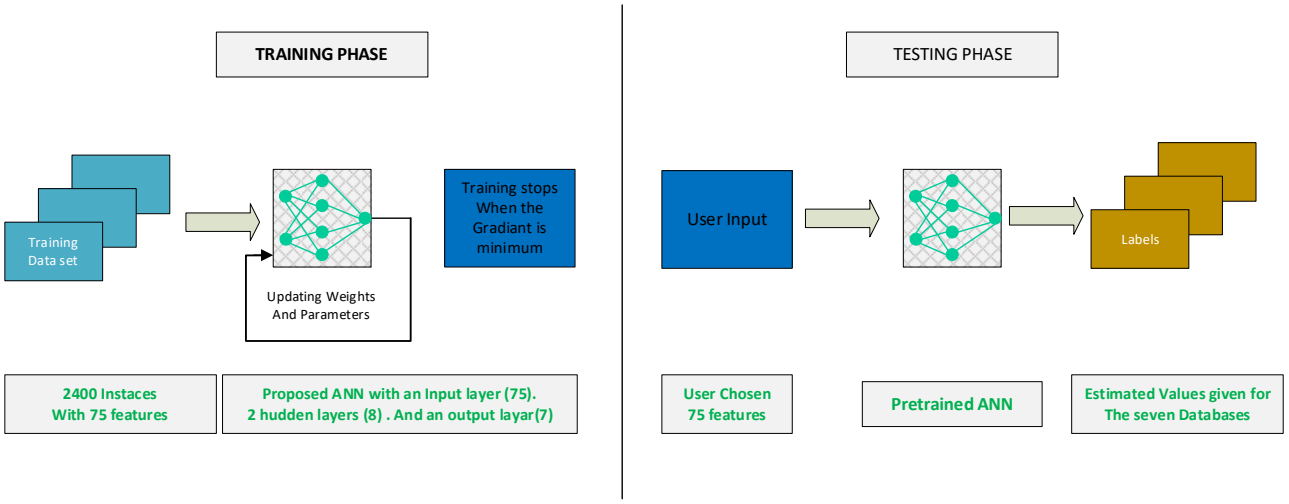

Fig. 4: Training and testing phases of the FFNN

Table 3: Experimental results of FFNN (75, 8, 8, 7) using different training functions

Training function	Elapsed time	MSE	Epoch	Exit condition
trainlm	0:53 minutes	5.16E-14	153	Minimum gradient reached
trainbr	0:12 minutes	1.01E-13	69	Minimum gradient reached
trainrp	0:02 minutes	8.13E-02	1000	Max epoch reached
trainscg	0:03 minutes	9.78E-03	1000	Max epoch reached

Table 4: Experimental results of FFNN (75, 8, 8, 7) using different cross-validation methods

Cross-validation method	Elapsed time	MSE	Epoch	Exit condition
Holdout, 75%-15%-10%	0:14 minutes	2.68E-14	47	Minimum gradient reached
10-fold cross-validation	0:53 minutes	5.16E-14	153	Minimum gradient reached
LOOCV	5:44 minutes	8.61E-14	1000	Max epoch reached

4.3. Cross-validation techniques

The evaluation employed three validation methods: the Holdout method (Arlot and Celisse, 2010), ten-fold cross-validation (McLachlan et al., 2005), and leave-one-out cross-validation (LOOCV) (Vanwinckelen and Blockeel, 2012). Both exhaustive validation approaches, such as LOOCV, and non-exhaustive approaches, such as the Holdout method and ten-fold cross-validation, were applied. Table 3 presents the experiments conducted on the proposed FFNN (75, 8, 8, 7) using the trainlm algorithm. Among these methods, ten-fold cross-validation was selected as the most suitable, as it

achieved better results than the others in terms of both elapsed time and mean squared error (MSE).

5. Conclusion

Database prediction based on user-selected features using a feedforward neural network has been discussed. The proposed methodology consists of four main stages: feature selection, dataset generation, neural network modeling, and database prediction. Feature selection involves identifying the key characteristics of row-oriented databases and generating ground-truth values for the seven selected databases, namely MySQL, MS SQL Server, Oracle, IBM DB2, PostgreSQL, SQLite, and Microsoft Access. Dataset generation is the second stage, which produces 2400 instances of different feature combinations along with their labels derived from scores and percentages. Neural network modeling forms the third stage and focuses on selecting the optimal parameters of the network to achieve the best performance. The final stage, database prediction, uses user inputs as features, which are fed into the trained FFNN to predict the most suitable database based on the network output.

A limitation of the current approach is the lack of access to raw data. Future work will aim to address this by applying other popular machine learning methods, such as support vector machines and random forests, and by incorporating a larger number of features and databases..

Appendix A. Ground Truth of databases and corresponding features

In Table A1, the value of $((f(i), db(j)) = 1$ indicates that the feature *'i'* is present in the database *'j'*. zero otherwise. Features underlined are common to all the selected databases.

Table A1: Ground truth data

Features	Databases						
	Oracle	Mysql	MS Sql Server	Postgre SQL	IBM DB2	Ms Access	SQLite
License							
Open source	0	1	0	1	0	0	1
Closed source	1	1	1	0	1	1	0
Memory req							
High	1	0	1	1	0	0	0
Low	0	1	0	0	0	0	1
OS support							
Windows	1	1	1	1	1	1	1
macOS	1	1	0	1	1	0	1
Linux	1	1	1	1	0	0	0
BSD	0	1	0	1	0	0	1
UNIX	1	1	0	1	1	0	1
AmigaOS	0	1	0	1	0	0	1
z/OS	1	1	0	0	1	0	0
iOS	0	0	0	0	1	0	1
Android	0	1	0	1	0	0	1
OpenVMS	1	0	0	0	0	0	0
Basic features							
ACID	1	1	1	1	1	1	1
Referential Integrity	1	1	1	1	1	1	1
Transactions	1	1	1	1	1	1	1
Locking (RLL)	1	1	1	1	1	0	0
Multiversion concurrency control	1	1	1	1	1	1	0
Unicode	1	1	1	1	1	1	1
Type inference	1	1	1	0	1	1	1
Compression	1	1	1	0	1	1	0
Scalability	1	1	1	1	0	0	0
Interface							
API	1	0	0	1	0	0	1
GUI	1	1	1	1	1	1	
SQL	1	1	1	1	1	1	1
Tables and views							
Temporary table	1	1	1	1	1	0	1
Materialized views	1	0	0	1	1	0	0
Indexes							
B-/B+ tree	1	1	1	1	1	1	1
R-/R+ tree	1	1	1	1	1	0	1
Hash expression	1	1	1	1	1	0	1
Partial	1	0	1	1	1	0	1
Reverse	1	0	1	1	1	0	1
Bitmap	1	0	1	1	1	0	0
Full-text		1	0	1	0	0	0
Spatial	1	1	1	1	1	0	1
FOT	1	1	1	1	0	0	1
Duplicate index prevention	1	0	0	0	0	0	0
Datatypes							
Integer	1	1	1	1	1	1	1
Floating point	1	1	1	1	1	1	1
Decimal	1	1	1	1	1		
String	1	1	1	1	1	1	1
Binary	1	1	1	1	1	1	1
Date/Time	1	1	1	1	1	0	0
Boolean	1	1	1	1	1	0	0
DB capabilities							
Union	1	1	1	1	1	1	1
Intersect	1	0	1	1	0	0	1

Except	1	0	1	1	0	0	1
Inner joins	1	1	1	1	1	1	1
Outer joins	1	1	1	1	1	1	1
Inner selects	1	1	1	1	1	1	1
Merge joins	1	0	1	1	0	0	0
Blobs and clobs	1	1	1	1	1	1	1
Common expression table	1	1	1	1	0	0	1
Windowing functions	1	0	1	1	0	0	1
Parallel query	1	0	1	1	0	0	1
System-versioned tables	1	0	1	1	0	0	
Aggregation queries	1	1	1	1	1	1	1
Partitioning							
Range	1	1	0	1	1	0	0
Hash	1	1	0	1	1	0	0
Composite (R+H)	1	1	0	1	1	0	0
List	1	1	0	1	1	0	0
Expression	1	1	0	1	1	0	0
Objects							
Data domain	1	0	1	0	1	1	0
Cursor	1	1	1	1	1	0	0
Trigger	1	1	1	1	1	0	1
Function	1	1	1	1	1	0	0
Procedure	1	1	1	1	1	1	0
External routine	1	1	1	1	1	1	1
Access controls							
Native network encryption	1	1	1	1	1	0	0
Brute-force protection	1	0	1	1	0	0	0
Enterprise directory compatibility	1	1	1	1	1	0	0
Password complexity rules	1	1	1	0	0	0	0
Patch access		1	1	1	0	0	0
Run unprivileged	1	1	1	1	1	1	1
Audit	1	1	1	1	1	0	1
Resource limit	1	1	1	1	1	0	1
Separation of duties (RBAC)	1	1	1	1	1	0	0
Security certification	1	1	1	1	1	0	0
Other features							
Crash recovery	1	0	1	1	1	0	0
Big data handling	1	0	0	1	0	0	0
Data warehousing	1	0	0	1	0	0	0
Cloud compatible	1	1	1	1	1	1	1

Appendix B. Experiments

Thirty-five experiments conducted as a part of neural network modeling are summarized in Table B1. Different parameters of neural networks were considered including the number of hidden layers, number of hidden neurons, different training functions and transfer functions, cross validation methods. After running the neural network its elapsed time and mean squared errors were examined carefully. The best set of parameters are chosen from the results obtained from the experiments.

Table B1: Experiments done as a part of neural network modeling

a	b	c	d	e	f	g	h	i
1	10	Trainbr	logsig, logsig	Holdout, 75%-15%-10%	2:46 minutes	1.17E-10	706	Max MU reached
1	10	Trainrp	logsig, logsig	Holdout, 75%-15%-10%	0:04 minutes	0.0382	1000	Max epoch reached
1	10	Trainlm	logsig, logsig	Holdout, 75%-15%-10%	4:04 minutes	2.30E-04	1000	Max epoch reached
1	10	Trainlm	tansig,purlin	Holdout, 75%-15%-10%	0:05 minuutes	7.71E-14	27	Minimum gradient reached
1	10	Trainscg	logsig, logsig	Holdout, 75%-15%-10%	0:03 minutes	0.037	1000	Max epoch reached
1	10	Trainlm	tansig,purlin	k fold cross validation, k=10	0:15 Minutes	9.44E-14	56	Minimum gradient reached
1	8	Trainrp	tansig,purlin	k fold cross validation, k=10	0:02 minutes	1.18E-01	1000	Max epoch reached
1	8	Trainlm	tansig,purlin	k fold cross validation, k=10	1:02 minutes	3.20E-14	211	Minimum gradient reached
1	8	Trainlm	tansig,purlin	LOOCV	1:52 Minutes	1.72E-14	332	Minimum gradient reached
1	8	Trainlm	logsig,purlin	LOOCV	0:57 minutes	1.50E-14	53	Minimum gradient reached
2	10--10	tarinlm	logsig ,logsig, softmax	Holdout, 75%-15%-10%	6:03 minutes	14.032	1000	Max epoch reached
2	10--10	trainbr	logsig ,logsig, softmax	Holdout, 75%-15%-10%	6:24 minutes	14.1	1000	Max epoch reached
2	10--10	tarinlm	logsig, logsig,logsig	Holdout, 75%-15%-10%	0:22 minutes	0.055701	58	Max MU reached
2	10--10	trainbr	logsig, logsig,logsig	Holdout, 75%-15%-10%	1:23 minutes	5.44E-10	337	Max MU reached
2	8--8	tarinlm	tansig,tansig,purlin	Holdout, 75%-15%-10%	0:14 minutes	2.68E-14	47	Minimum gradient reached
2	10--10	trainbr	tansig,tansig,purlin	k fold cross validation, k=10	0:56 Minutes	9.98E-14	153	Minimum gradient reached
2	8--8	tarinlm	tansig,tansig,purlin	k fold cross validation, k=10	0:53 Minutes	5.16E-14	153	Minimum gradient reached
2	8--8	trainbr	tansig,tansig,purlin	k fold cross validation, k=10	0:12 minutes	1.01E-13	69	Minimum gradient reached
2	8--8	trainrp	tansig,tansig,purlin	k fold cross validation,	0:02 minutes	8.13E-02	100	Max epoch reached

2	8--8	Trainscg	tansig,tansig,purlin	k=10 k fold cross validation, k=10	0:03 Minutes	9.78E-03	0 1000	Max epoch reached
2	10--10	trainrp	tansig,tansig,purlin	k fold cross validation, k=10	0:03 Minutes	2.43E-02	1000	Max epoch reached
2	8--8	tarinlm	tansig,tansig,purlin	LOOCV	5:44 minutes	8.61E-14	1000	Max epoch reached
2	8--8	tarinlm	logsig,logsig,purlin	LOOCV	0:19 minutes	1.07E-14	106	Minimum gradient reached
2	8--8	tarinlm	tansig,tansig,softmax	k fold cross validation, k=10	0:09 minutes	4.97E+00	52	Validation Stop
2	10--10	tarinlm	tansig,tansig,softmax	Holdout, 75%-15%-10%	6:06 minutes	4.69	1000	Max epoch reached
3	10--10--10	Trainlm	logsig, logsig,logsig,logsig	Holdout, 75%-15%-10%	0:0051minutes	15.2417	17	Max MU reached
3	10--10--10	Trainlm	tansig,tansig, tansig,purlin	Holdout, 75%-15%-10%	0:33 minutes	1.73E-14	70	Minimum gradient reached
3	10--10--10	Trainlm	tansig,tansig, tansig,purlin	k fold cross validation, k=10	1:02 Minutes	2.69E-13	144	Minimum gradient reached
3	10--10--10	trainrp	tansig,tansig, tansig,purlin	k fold cross validation, k=10	0:03 Minutes	0.0772	1000	Max epoch reached
3	8--8--8	Trainlm	tansig,tansig, tansig,purlin	LOOCV	8:36 minutes	1.07E-08	1000	Max epoch reached
3	8--8--8	trainrp	logsig, logsig,logsig,purlin	LOOCV	1:02 minutes	0.0471	1000	Max epoch reached
3	10--10--10	Trainlm	tansig,tansig, tansig,logsig	Holdout, 75%-15%-10%	0:35 Minutes	5.21	47	Validation Stopped
3	10--10--10	Trainlm	logsig, logsig,logsig,logsig	Holdout, 75%-15%-10%	4:10 minutes	7.89	509	Validation Stopped
3	10--10--10	Trainlm	purelin,purelin,purelin,tansig	Holdout, 75%-15%-10%	9:27 minutes	4.60E-10	1000	Max epoch reached
3	8--8--8	Trainscg	tansig,tansig, tansig,purlin	k fold cross validation, k=10	0:04 Minutes	0.0596	1000	Max epoch reached

a: #Hidden layers; b: #Hidden neurons; c: Training function; d: Activation functions; e: Cross validation methods; f: Elapsed time; g: MSE; h: epoch; i: Exit condition

List of abbreviations

ABAC	Attribute-based access control
ACID	ACID properties
ANN	Artificial neural network
API	Application programming interface
DBMS	Database management system
FFNN	Feedforward neural network
FOT	Feature-oriented typing
GUI	Graphical user interface
LOOCV	Leave-one-out cross-validation
MSE	Mean squared error
MVCC	Multiversion concurrency control
NoSQL	NoSQL database
RBAC	Role-based access control
RLL	Record-level locking
SQL	Structured query language
SSE	Sum of squared errors
YCSB	Yahoo! cloud serving benchmark

Compliance with ethical standards

Conflict of interest

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

Abadi DJ, Madden SR, and Hachem N (2008). Column-stores vs. row-stores: How different are they really? In the Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, Association for Computing Machinery, Vancouver, Canada: 967-980. https://doi.org/10.1145/1376616.1376712

Abbasi M, Bernardo MV, Váz P, Silva J, and Martins P (2024). Adaptive and scalable database management with machine learning integration: A PostgreSQL case study. Information, 15(9): 574. https://doi.org/10.3390/info15090574

Arlot S and Celisse A (2010). A survey of cross-validation procedures for model selection. Statistics Surveys, 4: 40–79. https://doi.org/10.1214/09-SS054

Bousalem Z, Guabassi IE, and Cherti I (2019). Relational databases versus HBase: An experimental evaluation. Advances in Science, Technology and Engineering Systems Journal, 4(2): 395-401. https://doi.org/10.25046/aj040249

Burden F and Winkler D (2008). Bayesian regularization of neural networks. In: Livingstone DJ (Eds.), Artificial neural networks: Methods and applications: 23-42. Humana Press, Totowa, USA. https://doi.org/10.1007/978-1-60327-101-1_3 **PMid:19065804**

El-Helw A, Ross KA, Bhattacharjee B, Lang CA, and Mihaila GA (2011). Column-oriented query processing for row stores. In the Proceedings of the ACM 14th International Workshop on Data Warehousing and OLAP, Association for Computing Machinery, Glasgow, UK: 67-74. https://doi.org/10.1145/2064676.2064689

Jarke M and Vassiliou Y (1985). A framework for choosing a database query language. ACM Computing Surveys, 17(3): 313-340. https://doi.org/10.1145/5505.5506

Kanade AS and Gopal A (2013). Choosing right database system: Row or column-store. In the International Conference on Information Communication and Embedded Systems, IEEE, Chennai, India: 16-20. https://doi.org/10.1109/ICICES.2013.6508217

Kepner J, Gadepally V, Hutchison D, Jananthan H, Mattson T, Samsi S, and Reuther A (2016). Associative array model of SQL, NoSQL, and NewSQL databases. In the IEEE High Performance Extreme Computing Conference, IEEE, Waltham, USA: 1-9. https://doi.org/10.1109/HPEC.2016.7761647

Kim W (2014). Web data stores (aka NoSQL databases): A data model and data management perspective. International Journal of Web and Grid Services, 10(1): 100-110. https://doi.org/10.1504/IJWGS.2014.058774

Levenberg K (1944). A method for the solution of certain non-linear problems in least squares. Quarterly of Applied Mathematics, 2(2): 164-168. https://doi.org/10.1090/qam/10666

Marquardt DW (1963). An algorithm for least-squares estimation of nonlinear parameters. Journal of the Society for Industrial

and Applied Mathematics, 11(2): 431-441. https://doi.org/10.1137/0111030

McLachlan GJ, Do KA, and Ambroise C (2005). Analyzing microarray gene expression data. John Wiley and Sons, Hoboken, USA. https://doi.org/10.1002/047172842X

Møller MF (1993). A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks, 6(4): 525-533. https://doi.org/10.1016/S0893-6080(05)80056-5

Mouhiha M and Mabrouk A (2025). NoSQL data warehouse optimizing models: A comparative study of column-oriented approaches. Big Data Research, 40: 100523. https://doi.org/10.1016/j.bdr.2025.100523

Okman L, Gal-Oz N, Gonen Y, Gudes E, and Abramov J (2011). Security issues in NoSQL databases. In the 10th International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, Changsha, China: 541-547. https://doi.org/10.1109/TrustCom.2011.70

Salunke SV and Ouda A (2024). A performance benchmark for the PostgreSQL and MySQL databases. Future Internet, 16(10): 382. https://doi.org/10.3390/fi16100382

Vanwinckelen G and Blockeel H (2012). On estimating model accuracy with repeated cross-validation. In the Proceedings of the 21st Belgian-Dutch Conference on Machine Learning, Ghent, Belgium: 39-44.

Yassien AW and Desouky AF (2016). RDBMS, NoSQL, Hadoop: A performance-based empirical analysis. In the 2nd Africa and Middle East Conference on Software Engineering, Association for Computing Machinery, Cairo, Egypt: 52-59. https://doi.org/10.1145/2944165.2944174