Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2022.015488
Article

Tech Science Press

Evaluation of Natural Language Software Interfaces to Databases

Fiaz Majeed[1], Muhammad Shoaib[2], Monagi H. Alkinani[3], Wazir Zada Khan[4], Shahzada Khurram[5], Akber Abid Gardezi[6] and Muhammad Shafiq[7,*]

[1]Department of Software Engineering, University of Gujrat, Gujrat 50700, Pakistan
[2]Computer Science and Engineering Department, University of Engineering and Technology Lahore, Pakistan
[3]College of Computer and Engineering, Department of CS & AI, University of Jeddah, Saudi Arabia
[4]Department of Computer Science, Capital University of Science and Technology, Islamabad, Pakistan
[5]Faculty of Computing, The Islamia University of Bahawalpur, Bahawalpur 63100, Pakistan
[6]Department of Computer Science, COMSATS University Islamabad, Islamabad, Pakistan
[7]Department of Information and Communication Engineering, Yeungnam University, Gyeongsan, 38541, Korea
*Corresponding Author: Muhammad Shafiq. Email: shafiq@ynu.ac.kr
Received: 24 November 2020; Accepted: 28 July 2021

Abstract: Relational databases are still important in modern times due to their many advantages, such as ease of interaction, simplicity and data integrity. In this regard, structured query language (SQL) and technical knowledge about database schemas are the basic building blocks for retrieving information from relational databases. Generally, non-expert users cannot skillfully write technical queries on the target database. To this end, many database natural language interfaces (NLIDB) have been developed to greatly facilitate users. However, each system provides a different interface for new users, so beginners can use different interactive modes to enter keyword-based queries. For users, migrating from one NLIDB to another is a difficult problem. To solve this problem, NLIDB needs to be standardized so that novice users can easily enter queries in their native language. This paper proposes a standardized interface for NLIDB, which considers the three quality parameters of interface, query comparison and performance. Based on these parameters, we developed a standardized interface called the keyword-based frequent search engine (FKBSE). The DBLP data set has been evaluated experimentally. The results show the effectiveness of FKBSE compared with parallel systems in terms of quality parameters.

Keywords: NLIDB; FKBSE; SQL; standardization; benchmark

1 Introduction

Structured Query Language (SQL) is famously used to retrieve information from relational databases. Usually, novice users are not familiar with SQL, so they lack the technical knowledge to deal with data in the database schema. The natural language interface of the database (NLIDB) is used to formulate natural language queries related to user inquiries to store and retrieve information from the targeted database. First, the user inputs a query to NLIDB in natural language, and then converts it to SQL. Then

retrieve the required information from the database and display it on the interface. Different interfaces have appeared in the literature, including menu-based, formal query language and graphical interface, but because the user's query ability is limited, they have the risk of interface format changes [1–3].

In this article, we considered four popular NLIDBs, including BANKS, Object Summary (OS), Scalable Keyword Search on Big Data Streams (SKSLDS) and SPARK to investigate user behavior and develop one for novice learning Interface. In banks, the backward expansion search algorithm is equipped with heuristics to calculate the results incrementally [4]. The affinity measurement relationship between pattern graphs and graphs originated from the OS ranking function technology. In the existing literature, the Affinity Ranking Correctness (ARC) is measured, and the ranking paradigm is evaluated in [5]. In addition, a keyword-based interface for data flow is provided in SKSLDS [6,7]. Finally, SPARK uses a sorting method based on traditional information retrieval (IR) sorting technology. It provides a method to search for top-k keyword queries of related tuples in relational databases [8].

NLIDB is designed to help novice users. However, the difference in NLIDB with different parameters makes it difficult for novice users to enter queries and search results. In order to solve this problem, a standardized interface based on Frequent Keyword Search Engine (FKBSE) is proposed. The interface is based on three parameters, namely interface, query comparison and performance. The DBLP data set has been evaluated experimentally. The results show the effectiveness of FKBSE compared with parallel systems. We studied the proposed FKBSE from the aspects of interface, query comparison, and performance.

The rest of this article is organized as follows. Section 2 provides a detailed literature survey based on relevant parameters. Section 3 introduces the proposed solution. Section 4 demonstrates the experimental evaluation using the parallel system. Finally, Section 5 summarizes the conclusions and future directions.

2 Related Works

The literature review is organized for the parameters used for comparative analysis in this work.

2.1 Interface

In the literature, there are text, visualization, and voice-based interfaces. The Data Warehouse Search (SODA) system is one of the text-based interfaces that enables business users to generate answers to *ad hoc* queries from the data warehouse. It uses graph pattern matching method to convert keywords into SQL queries. Initially, SODA used ontology and pattern-based graphs to map keywords to pattern elements to explore data [9]. The SQL aggregation using keywords (SQAK) system simplifies query input by setting up SQL aggregation queries of simple keywords by using a measure based on edit distance. A complete software development cycle has been adopted, and sampling-based search algorithms have been used [10,11].

The visualization-based interface is an extension of the text-based interface to facilitate non-technical users [12,13]. Among them, the DataTone system was demonstrated to provide a combination of natural language query and visualization. It solves the ambiguity of natural language processing by providing visualization. Here, the user can select the appropriate option from the list of available visualizations. The system has been evaluated on six data sets, but these limitations are limited in the sense that each data set contains only one table. In order to evaluate the efficiency and accuracy, the data tones should be analyzed on the benchmark data set [14,15].

We can also find research trends in NLIDB related to voice-based interfaces. EchoQuery is a similar system, it first accepts voice input from the user. Later, it will optimize the given query and convert it to SQL. The system was evaluated on two data sets including MIMIC and star mode benchmark tests. This system only works in relational databases [16]. In addition, the Articulate system obtains queries from users in the form of voice and converts them into visual content. It proposes a VisQL specification

language to request the construction of diagrams. In order to find relevant charts, a classification technique was chosen. Providing follow-up queries and lack of metadata support are the limitations of this work [13].

2.2 Query Comparison

The NLIDB system shall provide relevant results while meeting the query comparison parameters. Many systems intend to provide relevant top k results for user input queries. However, the EASE system models structured unstructured and semi-structured data as graphics. It generates the top k answers for a large amount of heterogeneous data [17]. In addition, the SPARK system establishes a ranking method based on traditional IR ranking technology. The global pipeline algorithm has been used for optimization. For the sorting function, a skyline scan is established, and the block pipeline algorithm performs query processing [8]. The OS ranking paradigm is given, in which the retrieved tuples are regarded as children of the tree. For the retrieval of significant results, the affinity of the relationship was estimated, and the level of the retrieved OS was measured [5]. In addition, the Kite solution also provides queries on heterogeneous databases. It links records from multiple databases based on foreign keys to display query results [18].

We can find algorithms for keyword search in data graphs in [19,20]. It has two components, including an engine and a ranker. The results show that the algorithm eliminates redundancy and sorts related answers, similar to search engines. In the figure, the scoring function is suitable for ranking related top structures. Among them, mode-based and modeless methods have been used. In [21,22], the algorithm is suitable for backward search, dynamic programming and enumeration methods. In the literature [23], a ranking strategy based on relevance is proposed, and the algorithm generates an answer graph with the support of breadth first search (BFS). The NaLIR system conducts input query through user interaction to accurately understand the user's intention. The system decodes its internal understanding through parsing and database context, where the explanation is presented to the user to choose the correct understanding. Accuracy is achieved through interactive sessions between NaLIR and users.

2.3 Performance

Performance evaluation is an important step in any system [24]. Research groups have established many NLIDB systems, but few people evaluate their performance [25]. The full grid algorithm is one of the systems that uses Naïve algorithm to communicate with SQL and uses sparse algorithm to improve query processing. Single-pipeline and global pipeline algorithms form queries, while hybrid algorithms make decisions for a given query [26]. DISCOVER uses candidate network generators and plan generators to discover information from relational databases. Greedy algorithm, optimal algorithm and naive algorithm execution plan execution and evaluation [27].

BANKS-II's two-way extended search algorithm generates answers to keyword queries based on graphs. Bidirectional expansion search can be evaluated by using sparse and backward expansion algorithms [27]. In addition, DBXplorer uses hashing and compression technology, which is a compression of the symbol table, which compresses the symbol table for the common keywords in the database column. The two-tier index model contains content related to keywords and node mapping in the top-level block index. The index tells whether the node has reached the specific key and the shortest distance. Using BFS can improve query performance [28].

The CNEvalDynamic algorithm has provided m-keyword support for relational data streams. In this work, the problem of seeking continuous and large data streams is solved. It limits the intermediate joins to speed up query processing. Only from the perspective of CPU [7,29] were evaluated. In addition, the PRECISE system is implemented on the basis of the theoretical framework. It claims to answer every user question correctly. According to experiments, the system can accurately answer 80% of queries. The system rarely answers these queries, so it is required to be rewritten [30].

The Quest system contributes by enhancing machine learning technology to check the database. It proposes two methods, namely forward and backward methods. In the forward approach, query keywords are mapped to schema elements. Use backward methods to rank connection paths using machine learning models. The system has been evaluated on IMDB, DBLP and Mondial databases [31]. In addition, the metadata method realizes the mapping of keywords to queries by calculating the weights of inherited nodes. In [32], a threshold algorithm was proposed to reduce the response time and frequency of results. In [33], the database and IR technology were merged into one system.

3 Proposed Solution

For the standardized design of NLIDB tools, a parameter-based architecture has been proposed. As shown in Fig. 1, the parameters include interface, query comparison and performance. This work is aimed at non-technical users who do not have the technical knowledge of the tool. The interface parameter suggests eight functions to standardize the tool design for these users. According to our observations, the diversity of interfaces hinders getting the most benefit from tools. A single standard interface for each tool can increase the scope of analysis for users. The query comparison parameters solve the general requirements when writing user queries in the form of text-based, navigation-based or voice-based. Performance parameters use a threshold to evaluate the throughput of the tool.

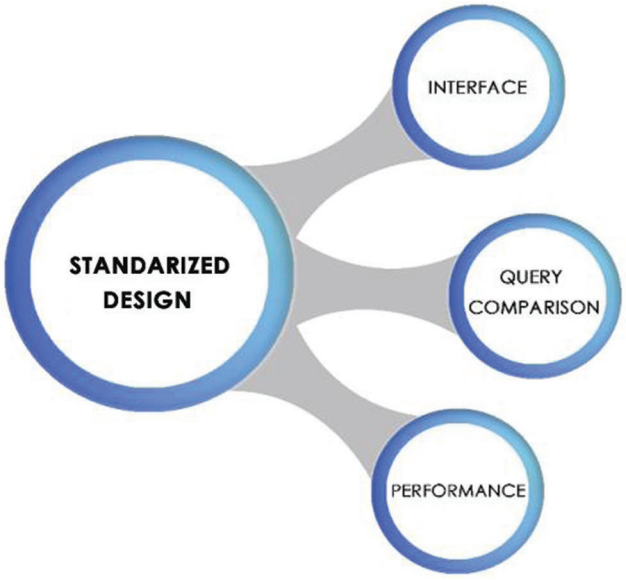


Figure 1: Proposed parameters-based architecture

3.1 Benchmark Parameters

3.1.1 Interface

End users and systems use interfaces as the basic source of communication. If the interface is properly designed, the user can easily interact with the system. The information retrieved from the database is displayed on the interface. So far, different interfaces have appeared in the literature, including menu-based, formal query languages and graphical interfaces, but these interfaces limit users' query capabilities by displaying changing interface formats. In Tab. 1, we have defined interface functions that enhance the usability of the system.

Table 1: Parameters of the interface

S#	Interface parameters
1	Ease of query input
2	User friendly
3	Ease for data selection
4	Details are visible
5	Database details available
6	Ease of result understanding
7	Interface interactivity
8	Result format

3.1.2 Query Comparison

The NLIDB system can be evaluated according to the query comparison parameters, and it can be measured according to the following functions.

The number and complexity measurement of keyword support: This function refers to how many query keyword searches the system can perform at the same time and how to solve the function of keyword complexity. Formally, the keyword support S_k can be measured as given in the Eq. (1). The result is represented by R, which contains all the different search records for all keywords K (input query). Therefore, each unique record r_i is a concatenation of at least one query keyword k_i, such as,

$$S_k = \sum_{i=1}^{R} (\text{Distinct}(k_i, r_i)) | \forall\, k_i \subseteq K \tag{1}$$

The system should establish a connection path for a large number of keywords behind the scenes. As the complexity of this process increases, accuracy and performance become critical.

Relevancy and Accuracy: The related results include all keywords specified in the user's query are regarded as accurate results. However, the retrieved results include the entire record displayed for the query. No input query keywords or a small number of records that are rated as irrelevant query keywords. Therefore, the correlation average r_a can be calculated by dividing the related records by the total retrieved records, as shown in Eq. (2). The total search record is represented by R, where k1 ⊓ k2 …. ⊓ kn defines related records as,

$$r_a = \frac{r_i}{R} \tag{2}$$

Dynamicity and Query Independence: Dynamic and query independence functions show the flexibility and comfort of query input in NLIDB. Query reconstruction supports dynamics, and query reconstruction can be used to obtain more accurate results according to user expectations. It practices spelling correction and alternative term selection techniques, ultimately making the system automated and user-friendly. The system that provides the basis for the dynamics becomes efficient and effective, and also provides relevant records. According to the independence of the query, the system cannot bind the user to the database schema used to enter the query. In contrast, SQL restricts the database schema to be notified to users to retrieve results, while a keyword-based system makes users independent.

3.1.3 Performance

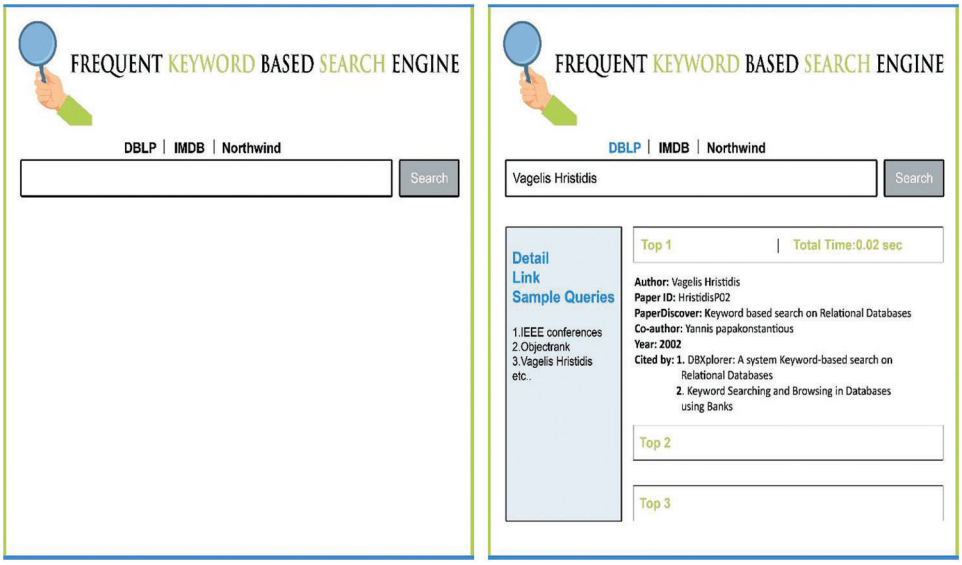
Performance is a measure of response time obtained by the system to display the complete results of a user's query. The response time is calculated after each query is executed on the interface of each system. It is the sum of processing time, ranking time and result display time. Research groups have established many NLIDB systems, but few people evaluate their performance.

3.2 Standard Design Based on Parameters

The advantages and disadvantages of the system can be determined through comparative analysis. Few systems expose flaws in their interface, while other systems perform poorly and constrain users during query input. A system that presents a lot of details and irrelevant results can affect performance. However, due to incorrect alignment and grouping, the results of some systems are not easy to understand. Too many details without any specific arrangements will irritate users [34]. Few systems do not respond to valid queries. In order to increase the usability and understandability of the system, the results must be reasonably visualized. Overcoming the defects of the interface can improve the usability of the system.

As shown in Fig. 2, we have designed an interface called FKBSE, which captures the limitations of the existing system. According to the survey results, the following are reasonable characteristics of NLIDB:

- Simple interface design
- System availability and understandability should be high
- Important ingredients should be visible
- Easy for new users
- Support novice users to understand
- Data selection should be easy
- Display response time
- Display the total number of records retrieved
- Display results in ordered groups
- Result ranking organization



(a) **(b)**

Figure 2: Proposed FKBSE interface (a) FKBSE interface (b) DBLP result format on FKBSE

3.3 User Experience

It is very important to evaluate the interface from the user's perspective. Fig. 3 shows the FKBSE interface, which includes the eight functions given in Tab. 1. As NLIDB's interface design has ambiguities at multiple levels, the FKBSE user interface is designed to make query input easy to interpret the results. The interface is intended to have all the functions defined for standardization. Let us describe the interface with an example scenario. For example, if the user (Hannah) wants to search for related research papers by the author. He wrote the query "Vagelis Hristidis" (see Fig. 3 (1): Easy query input). He can choose the sample query suggested in the current input query (see Fig. 3 (2): User Friendly). When Hannan presses the search button, the results will be retrieved from the specified database. Now, he can choose to view any number of highest-ranked records, such as the top 3 records (see Fig. 3 (3): easy selection of data). The results can be easily interpreted, such as Top 1, total time and total records (see Fig. 3 (4): details are visible). The user can select any available database to view the results (see Fig. 3 (5): available database selection). In addition, Hannan can easily understand the results, which are marked with titles such as Authors, PaperID, etc. (see Fig. 3 (6): Easy to understand results). In addition, the design is interactive for the user, for example, he can expand the results, set many records, etc. (see Fig. 3 (7): Interface Interactivity). Finally, the result is displayed in a simple format (see Fig. 3 (8): result format).

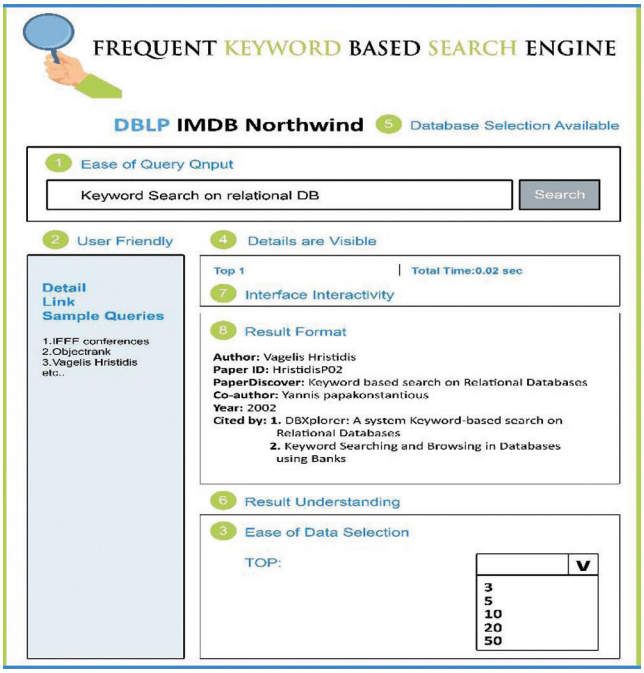


Figure 3: Scenario representing user experience on the interface

The query comparison parameter is evaluated by relevancy and accuracy. Fig. 4 shows the FKBSE query comparison rules. We follow the scenario of a specific user who can write any number of keywords, such as "keyword-based search on relational databases and multidimensional databases" in order to explain the comparison of queries similar to this interface. Relative to the input query retrieval results, and the achieved relevance is 92%. In addition, FKBSE can be used by non-technical users who have zero knowledge of database architecture and structure.

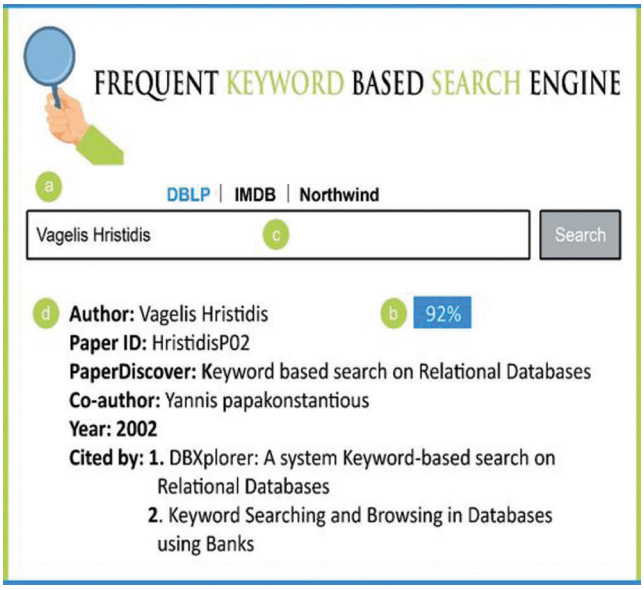


Figure 4: Characteristics of the FKBSE: FKBSE supports any number of keywords with 92% relevancy; No schema knowledge required for writing query; Current FKBSE version does not provide dynamicity

4 Experimental Evaluation

Here, we evaluate the performance of our method through in-depth experiments. We initially discussed parallel systems and data sets. Finally, the experimental analysis and results are given.

4.1 Parallel Systems

We studied four parallel systems. In this regard, SPARK (see Fig. 5a) is a well-known and effective keyword-based NLIDB system. During operation, SPARK is convenient for users to query easily. The interface of the system is user-friendly for novices. On the other hand, OS NLIDB (see Fig. 5b) is a novel keyword search paradigm for relational databases. It produces results in the form of OSs that characterize the retrieved associated tuples for user queries. When executed on the DBLP and Northwind datasets, it is not portable. Sometimes, because the OS allows a lot of details to be displayed, it may irritate the user. However, novice users may have trouble entering queries because it limits the user's perception of the patterns used to determine keywords to formulate queries. However, the SKSLDS system (see Fig. 5c) is easier than the above system. It supports a larger number of keywords. In addition, user independence is important in this system. Therefore, it unconditionally restricts users from entering queries without any restrictions. The results are displayed in a tree format, which can improve user comprehensibility. Users can smoothly understand the database architecture. BANKS-I and BANKS-II (see Fig. 5d) are well-known keyword search systems for relational databases. BANKS-I implements a backward expansion scheme, while BANKS-II uses a two-way expansion mechanism. Both help to enter up to two keywords and display the top 10 results for any query. According to requirements, users can explore further results. The results are easy to understand, and users can easily enter queries without knowing the architecture. These NLIDB also provide sample queries for novice users.

4.2 Dataset

We have used the benchmark data set DBLP to evaluate the parameters on the selected system. A query set based on DBLP has been developed for comparative analysis of FKBSE with these systems. The specifications of the data set are given in Tab. 2.

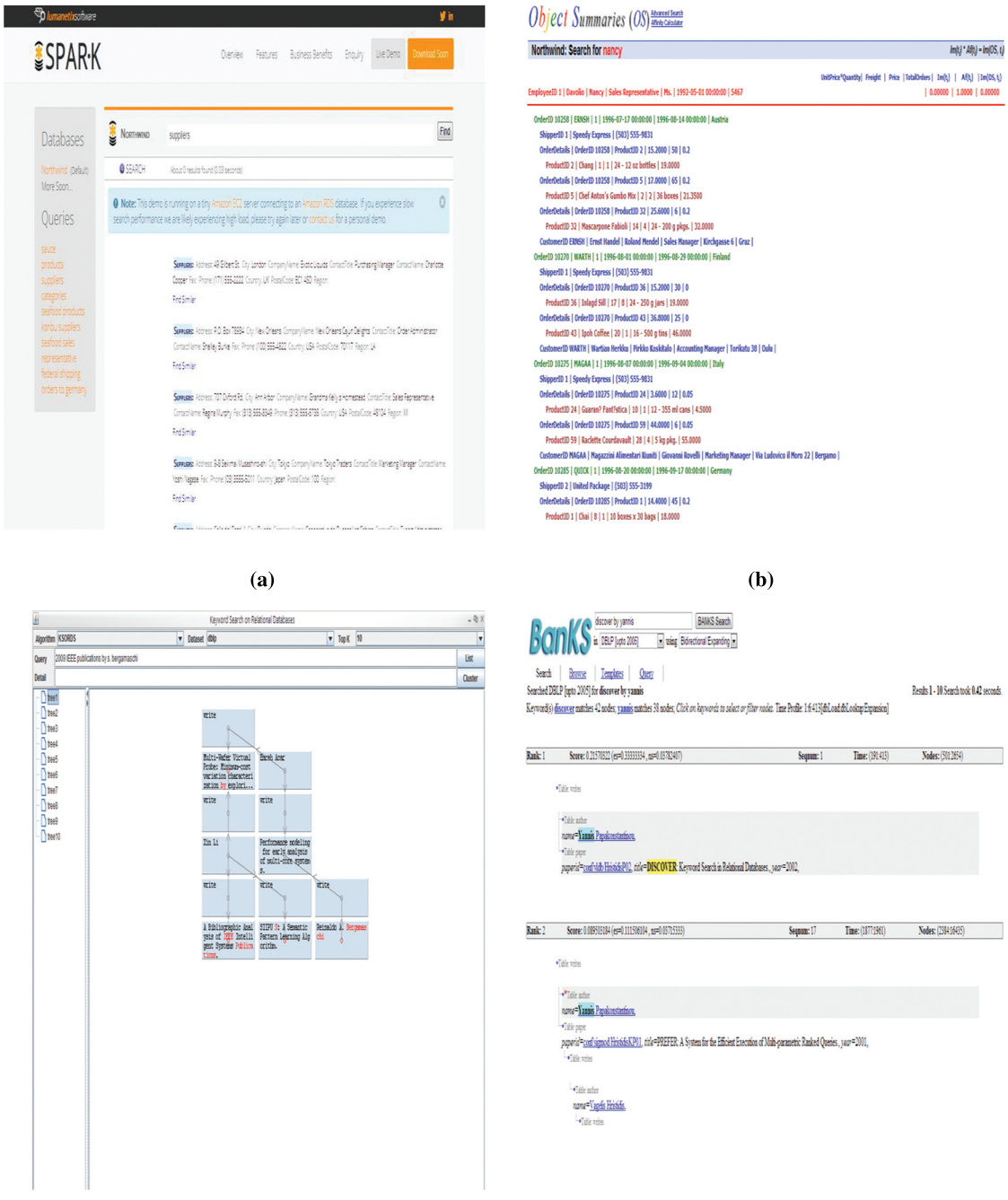


Figure 5: Interfaces of the parallel systems (a) SPARK (b) Object summaries (c) SKSLDS (d) BANKS

Table 2: Statistics of data set from DBLP

Queries built	Articles
100	2.5 million

The sample in the query set is displayed in the Tab. 3. These queries are used to evaluate the parameters of the system. From 2002 to 2016, we conducted some experiments on the DBLP dataset. The data set has a computer science bibliography website at the University of Trier, Germany. The bibliography contains more than 2.5 million articles, many of which are linked to the homepages of computer scientists.

Table 3: A sample from the query set

S#	Queries
1	Discover by yannis
2	IEEE publications
3	S.Chakrabarti
4	University of California
5	OLAP Data cubes
6	Peer-to-Peer network
7	Spatial data mining
8	Multidimensional databases
9	VLDB conference
10	Objectrank
11	Vagelis Hristidis

4.3 Results Discussion

We conducted an on-site survey to test interface parameters using data set queries. Prepare a questionnaire to obtain the ranking score of the selected system relative to the interface parameters. In our field survey, 150 users who understand the common interface voluntarily participated. In Fig. 6, the results show that, compared to other systems, the FKBSE interface ranks higher in the eight functions (given in Tab. 1) (with the highest score of 39.66). The y-axis shows the average score (%), and the x-axis shows NLIDB. Few systems are better for specific functions, and those systems are not suitable for other functions. From this perspective, BANKS-I and BANKS-II are easier to use. The SPARK interface is more interactive and provides a click environment. Based on the analysis of all competing systems, FKBSE has all the best features available. Therefore, FKBSE received the highest score in the user evaluation. Users explored SPARK, SKSLDS, OS, BANKS-1 and BANKS-II, so they found all the positive features in a standardized form in FKBSE. In order to achieve 100% user satisfaction, the interface must be simpler and user-friendly.

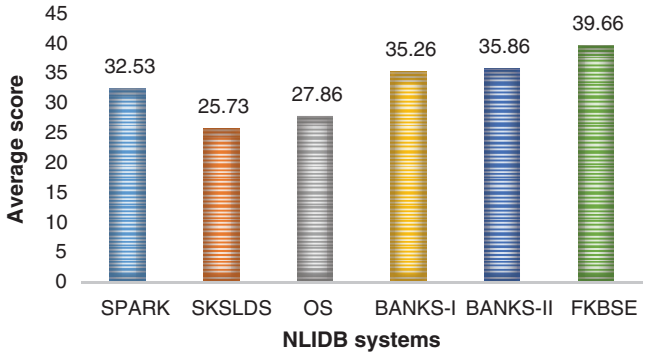


Figure 6: Interface features

In the query comparison parameters, FKBSE supports searching for 5 keywords that are higher than the parallel system. SPARK and SKSLDS support up to four keywords relative to this parameter, while OS, BANKS-1 and BANKS-II limit two keywords. The results are depicted in Fig. 7. The number 1 is supported, but the 0 is not supported. According to the results, FKBSE not only standardizes NLIDB, but also contains the greatest advantages that exist in any system. Therefore, it supports 5 keywords, which is the most in the existing system. The retrieved results place those documents in the highest position, and the document keeps all keywords in adjacent positions. This research puts forward this concept through relevance and accuracy. The accuracy of the comparison result is measured according to the level of the relevant result. The enumeration of dependencies is specified in Tab. 4. The following abbreviations are used as follows:

- RET = Retrieved records
- R = Relevant from retrieved records
- IR = Irrelevant from retrieved records

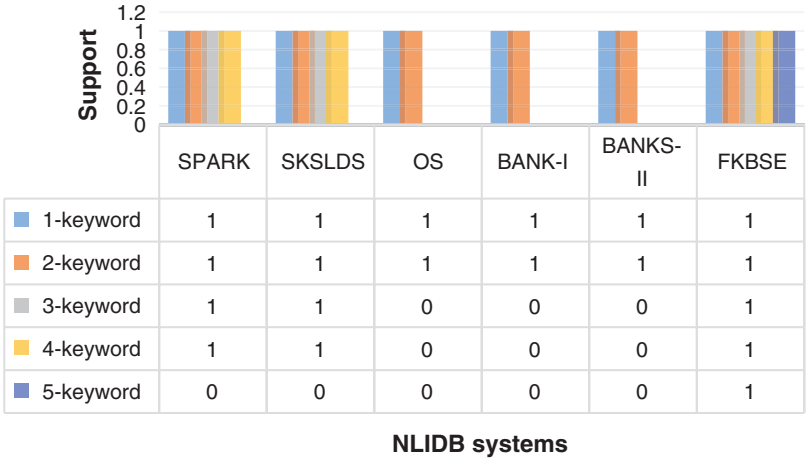


Figure 7: Capability of systems for keywords support

Table 4: Relevancy statistics

	SPARK			SKSLDS			OS			BANKS-I			BANKS-II			FKBSE		
Q#	RET	R	IR	RET	R	IR	RET	R	IR	RET	R	IR	RET	R	IR	RET	R	IR
1	5	5	0	1000	976	24	2	2	0	350	350	0	960	960	0	982	982	0
2	2	2	0	1000	797	203	0	0	0	400	400	0	40	40	0	1000	1000	0
3	4	3	1	1000	954	46	0	0	0	40	40	0	3	2	1	748	748	0
4	2	1	1	1000	897	103	0	0	0	260	260	0	80	80	0	150	150	0
5	3	3	0	1000	899	101	1	1	0	20	20	0	340	340	0	402	401	1
6	13	12	1	1000	783	217	11	9	0	375	207	168	140	56	84	140	120	20
7	18	17	1	1000	877	123	13	13	3	40	40	0	320	320	0	999	991	8
8	23	22	1	1000	863	137	19	19	0	350	350	0	140	140	0	1000	1000	0
9	29	29	0	1000	798	202	25	25	0	80	80	0	60	60	0	809	809	0
10	4	4	0	3	3	0	1	1	0	0	0	0	0	0	0	10	10	0
11	4	4	0	1	1	0	1	1	0	1	1	0	1	1	0	4	4	0

For example, for query 1 on SKSLDS, RET = 1000 records have been retrieved. These RET related records are R = 976, while IR = 24 records are irrelevant. The average correlation is 97.6. The results are shown in Fig. 8. According to the results, FKBSE has significant correlation with BANKS-II, BANKS-I, SKSLDS and OS, respectively. FKBSE has a correlation of approximately 100%, while BANKS-II provides a correlation of 80%. Therefore, FKBSE provides more than 20% relevant results.

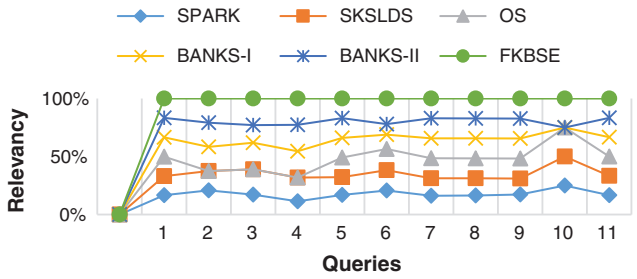


Figure 8: Comparison for retrieval of the relevant results

Fig. 9 shows that all systems except OS support query independence. In the OS system, users must understand the architecture to understand the results generated by user queries. In addition, only the SPARK system supports dynamics. It uses existing knowledge to automatically perform query refactoring. Currently, the query reconstruction function is not provided in FKBSE. The query reconstruction function will be implemented in FKBSE, which is part of future work. Fig. 10 summarizes a comparison of parallel systems used to evaluate performance. The results show that FKBSE takes less time to display the results. It takes 1.13 s on average. BANKS-II, BANKS-I, SKLDS, OS and SPARK consume an average of 2.64, 3.19, 4.23, 4.34 and 4.4 s respectively. Therefore, the efficiency of FKBSE is 43% lower than that of BANKS-II, and the time of BANKS-II is 2 min less. The increased dynamics in the system will significantly improve the performance of FKBSE. In this way, several optimization plans will be constructed from which the most effective plan can be executed.

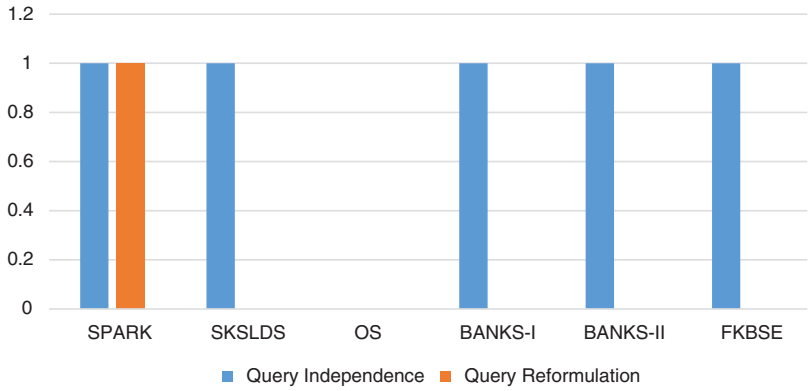


Figure 9: Dynamicity and query independence support

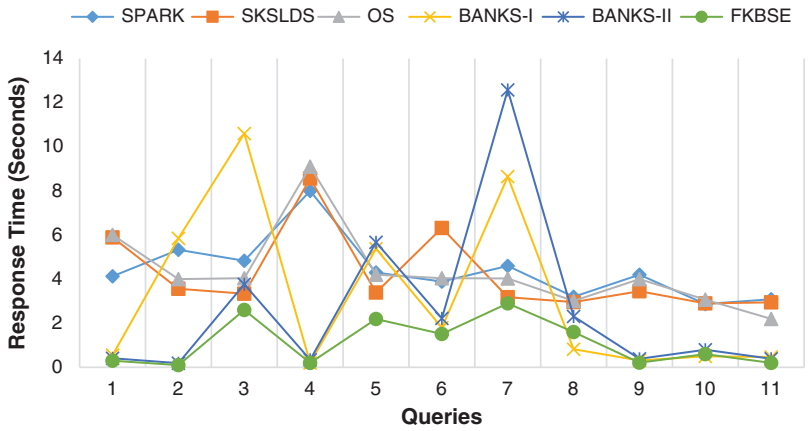


Figure 10: Performance probe of competitive systems

5 Conclusions

This paper proposes a standardized interface FKBSE. It has been evaluated using parallel systems from different parameters including interface, query comparison and performance. According to the interface parameters, our proposed FKBSE has the highest average score of 39.66. FKBSE provides the function of searching for five keywords at the same time based on query comparison parameters. On the other hand, SPARK and SKSLDS can only support up to four keywords. The relevance of FKBSE search results is higher than that of BANKS-II, which obtains relevant results in the second position. Based on performance parameters, our results verify that FKBSE takes less time to display results than other systems. For portable computers equipped with FKBSE, surveys are essential. It will be developed as the only system that can be deployed with any relational database and meets all requirements.

Acknowledgement: The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number MoE-IF-20-01.

Funding Statement: This works was supported by the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia through the project number MoE-IF-20-01.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. I. Androutsopoulos, G. D. Ritchie and P. Thanisch, "Natural language interfaces to databases—An introduction," *Journal of Natural Language Engineering*, vol. 1, no. 709, pp. 50–81, 1995.

[2] I. Mathkour, B. Hassan, S. Shahzad and Al-Wakeel, "Software risk management and avoidance strategy," in *Proc. ICMLC*, Singapore, pp. 477–481, 2011.

[3] C. Baik, H. V. Jagadish and Y. Li, "Bridging the semantic gap with SQL query logs in natural language interfaces to databases," in *Proc. ICDE*, Macao, China, pp. 374–385, 2019.

[4] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti and S. Sudarshan, "Keyword searching and browsing in databases using BANKS," in *Proc. ICDE*, San Jose, CA, USA, pp. 431–440, 2002.

[5] G. J. Fakas, "Automated generation of object summaries from relational databases: A novel keyword searching paradigm," in *Proc. ICDE*, Cancun, Mexico, pp. 564–567, 2008.

[6] Y. Xu, "Scalable top-k keyword search in relational databases," *Cluster Computing*, vol. 22, no. 1, pp. 731–747, 2019.

[7] L. Qin, J. X. Yu and L. Chang, "Scalable keyword search on large data streams," *The VLDB Journal*, vol. 20, no. 1, pp. 35–57, 2011.

[8] X. Lin, "SPARK: Top-k keyword query in relational databases," in *Proc. SIGMOD*, Vienna, Austria, pp. 115–126, 2007.

[9] L. Blunschi, C. Jossen, D. Kossmann, M. Mori and K. Stockinger, "SODA: Generating SQL for business users," *Proceedings of the VLDB Endowment*, vol. 5, no. 10, pp. 932–943, 2012.

[10] S. Tata and G. M. Lohman, "SQAK: Doing more with keywords," in *Proc. SIGMOD*, Vancouver, Canada, pp. 889, 2008.

[11] B. Shahzad, Y. Al-Ohali and A. Abdullah, "Trivial model for mitigation of risks in software development life cycle," *International Journal of Physical Sciences*, vol. 6, no. 8, pp. 2072–2082, 2011.

[12] F. Siwei, K. Xiong, X. Ge, Y. Wu, S. Tang *et al.,* "Quda: Natural language queries for visual data analytics," arXiv preprint arXiv: 2005.03257, pp. 1–17, 2020.

[13] Y. Sun, J. Leigh, A. Johnson and S. Lee, "Articulate: A semi-automated model for translating natural language queries into meaningful visualizations," in *Proc. SG*, Banff, Canada, pp. 184–195, 2010.

[14] T. Gao, M. Dontcheva, E. Adar, Z. Liu and K. Karahalios, "Datatone: Managing ambiguity in natural language interfaces for data visualization," in *Proc. UIST*, Charlotte, NC, USA, pp. 489–500, 2015.

[15] A. S. rinivasan and J. Stasko, "How to ask what to say?: Strategies for evaluating natural language interfaces for data visualization," *IEEE Computer Graphics and Applications*, vol. 40, no. 4, pp. 96–103, 2020.

[16] G. Lyons, V. Tran, C. Binnig, U. Cetintemel and T. Kraska, "Making the case for query-by-voice with EchoQuery," in *SIGMOD*, San Francisco, USA, pp. 2129–2132, 2016.

[17] G. Li, B. C. Ooi, J. Feng, J. Wang and L. Zhou, "EASE: An effective 3-in-1 keyword search method for unstructured, semi-structured and structured data," in *Proc. SIGMOD*, Vancouver, Canada, pp. 903–914, 2008.

[18] M. Sayyadian, H. LeKhac, A. Doan and L. Gravano, "Efficient keyword search across heterogeneous relational databases," in *Proc. ICDE*, Istanbul, Turkey, pp. 346–355, 2007.

[19] M. I. Lali, R. I. Mustafa, K. Saleem, M. S. Nawaz, T. Zia *et al.,* "Finding healthcare issues with search engine queries and social network data," *International Journal on Semantic Web and Information Systems*, vol. 13, no. 1, pp. 48–62, 2017.

[20] K. Golenberg, K. Golenberg, B. Kimelfeld, B. Kimelfeld, Y. Sagiv *et al.,* "Keyword proximity search in complex data graphs," in *Proc. SIGMOD*, Vancouver, Canada, pp. 927–940, 2008.

[21] N. Hormozi, "Disambiguation and result expansion in keyword search over relational databases," in *Proc. ICDE*, Macao, China, pp. 2101–2105, 2019.

[22] J. X. Yu, L. Qin and L. Chang, "Keyword search in relational databases: A survey," *IEEE Data Engineering Bulletin*, vol. 33, no. 1, pp. 67–78, 2010.

[23] F. Liu, C. Yu, W. Meng and A. Chowdhury, "Effective keyword search in relational databases," in *SIGMOD*, Chicago, IL, USA, pp. 563–574, 2006.

[24] J. Al-Muhtadi, B. Shahzad, K. Saleem, W. Jameel and M. A. Orgun, "Cybersecurity and privacy issues for socially integrated mobile healthcare applications operating in a multi-cloud environment," *Health Informatics Journal*, vol. 25, no. 2, pp. 315–329, 2019.

[25] F. Li and H. V. Jagadish, "Constructing an interactive natural language interface for relational databases," *Proceedings of the VLDB Endowment*, vol. 8, no. 1, pp. 73–84, 2014.

[26] S. Diego, L. Gravano and S. Diego, "Efficient IR-style keyword search over relational databases," in *Proc. VLDB*, Berlin, Germany, pp. 850–861, 2003.

[27] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai *et al.,* "Bidirectional expansion for keyword search on graph databases," in *Proc. VLDB*, Trondheim, Norway, pp. 505–516, 2005.

[28] H. He, H. Wang, J. Yang and P. S. Yu, "BLINKS: Ranked keyword searches on graphs," in *SIGMOD*, Vienna, Austria, pp. 305–316, 2007.

[29] A. Markowetz, "Keyword search on relational data streams," in *Proc. SIGMOD*, Vienna, Austria, pp. 605–616, 2007.

[30] A.-M. Popescu, O. Etzioni and H. Kautz, "Towards a theory of natural language interfaces to databases," in *Proc. IUI*, Miami, Florida, USA, pp. 149–157, 2003.

[31] S. Bergamaschi, F. Guerra, M. Interlandi, R. Trillo-Lado and Y. Velegrakis, "Quest: A keyword search system for relational data based on semantic and machine learning techniques," *Proceedings of the VLDB Endowment*, vol. 6, no. 12, pp. 1222–1225, 2013.

[32] S. Bergamaschi, E. Domnori, F. Guerra, R. Trillo Lado and Y. Velegrakis, "Keyword search over relational databases: A metadata approach," in *Proc. SIGMOD*, Athens, Greece, pp. 565–576, 2011.

[33] L. Qin, J. X. Yu and L. Chang, "Keyword search in databases: The power of RDBMS," in *Proc. SIGMOD*, Providence, Rhode Island, USA, pp. 681–693, 2009.

[34] M. Shafiq, M. Ahmad and J.-G. Choi, "Public system usability analysis for the valuation of cognitive burden and interface standardization: A case study of cross-ATM design," *Journal of Organizational Computing and Electronic Commerce*, vol. 27, no. 2, pp. 162–196, 2017.