An Approach to Development of Interactive Adaptive Software Tool to Support Data Analysis Activity

Dmytro Orlovskyi, Andrii Kopp and Ivan Bilous

National Technical University "Kharkiv Polytechnic Institute", Kyrpychova str. 2, Kharkiv, 61001, Ukraine

Abstract

In the recent decades, databases have been used in any field of human activity to keep valuable data about ongoing processes. Large amounts of data stored in enterprise databases are useless without having a specialized software tool for data discovery or querying. Most business users that make data-driven decisions usually do not have special training and experience in database querying using special formal languages. Existing solutions are based on "query wizards" and database query forms that require knowledge of a database schema and inconvenient for users without special training. Proposed approach is based on the content-based filtering of already executed queries by usage frequency and similarity criteria in order to suggest relevant queries that may be re-used. It is a baseline of the interactive adaptive system for data analysis, which design and development is outlined in this study. Software prototype was demonstrated and its usage was discussed. Conclusions were made and future research was formulated.

Keywords 1

Database Query, Adaptive Query Interface, Data Analysis, Interactive Software Tool

1. Introduction

Nowadays enterprise databases contain extremely large collections of transactional and aggregated analytical data records. Such data volumes contain descriptions of hundreds or even thousands of data entities, described by multiple various attributes of different data types. Without having interactive and flexible querying tools, such data collections are basically useless, since no data could be obtained for analytical processing, visualization, and decision making. Existing querying languages, such as SQL (Structured Query Language), LINQ (Language Integrated Query), DAX (Data Analysis eXpressions), or XQuery, need special training and experience in software engineering, database design, and database administration. Besides querying languages, it is required to know the database schema structure to apply these query languages. Such data preparation tasks may distract data analysts from their primary activities, which include interpretation and communication of data analysis results (e.g. using simple spreadsheets or data visualizations based on descriptive, or predictive models [1]) to stakeholders. Also improper usage of database querying languages and lack of database schema knowledge, may mislead data analysts and block problems solving.

Modern database management systems are accompanied with so-called "query wizards" intended to simplify data querying and do not require knowledge of SQL or other data querying languages. These software components are considered as user-friendly tools that could be used by non-technical users to build analytical reports and glean insights. On practice most data analysis problems require integration of enterprise databases or other corporate data sources with a specialized software tool or service. There are Business Intelligence (BI) tools focused on visual reports design (e.g. Power BI or QlikView) [2], data science methods and models provided by a standalone software tool (e.g. RStudio or Jupyter) [3] or libraries and frameworks (e.g. Pandas or Matplotlib) [4]. The first step for data

CMIS-2021: The Fourth International Workshop on Computer Modeling and Intelligent Systems, April 27, 2021, Zaporizhzhia, Ukraine
EMAIL: orlovskyi.dm@gmail.com (D. Orlovskyi); kopp93@gmail.com (A. Kopp); ivanbilous2000@gmail.com (I. Bilous)
ORCID: 0000-0002-8261-2988 (D. Orlovskyi); 0000-0002-3189-5623 (A. Kopp); 0000-0001-8110-786X (I. Bilous)


visualization, or even for simple data presentation using spreadsheets, is data querying. Since query languages require skills and experience to be applied, and existing "query wizards" are limited in their functionality and mostly are not understandable for end users, the problem of development of a software tool for data querying becomes relevant. This software tool should provide interactive adaptive user interface, which can be used by such users as: data analysts (if they for some reason do not have special training in database management systems), business analysts, top managers, or board members.

Therefore, this study proposes an approach to development of interactive adaptive software tool to support data analysis activity that is considered as the research objective. Research subject includes the approach to development of interactive adaptive software tool. This study aims to decrease complexity of data analysis process by supporting related activities with the interactive adaptive software tool.

This paper consists of six sections. First section introduces research problem and is already outlined. Second section demonstrates literature review and problem relevance. In third section a formal problem statement is given. Fourth section includes description of a recommending procedure, elicited software requirements, a database structure, and a system design. Software prototype usage and its discussion is outlined in fifth section. Sixth section includes conclusions and a future research statement.

2. Literature Review

Analysis of the state-of-the-art has demonstrated that considered research topic is already covered by several studies. Tang et al. in [5] proposed a database query form (DQF) interface used to generate query forms based on captured user's preferences. In these DQF preferences are gathered to rank query form components in order to assist users in making decisions. Query form generation process outlined in [5] is iterative and is guided by users: among proposed lists of components users may choose desired ones in order to include into the query form. At each iteration users can submit their forms and adjust DQF until they are satisfied with the obtained response [5].

The term DQF introduced in [5] then appeared in multiple research studies. Authors of [6] propose a DQF system based on user interactions captured in order to adapt questions submitted through query forms. Proposed procedure is also iterative, which means that form could be dynamically refilled till the user satisfies with query results [6]. Paper [7] proposes a survey on DQF approach and reviews its core concepts: query form interfaces, interface components ranking metrics, and estimation of ranking score. The DQF survey also surveyed in [8], where four functional modules of DQF software systems are considered: query form enrichment module, query execution module, customizable query form, and query recommendation module. It is also concluded in all of the considered survey papers that dynamic database querying approach results in higher success rate and easier usage in compare to static database querying approach [6, 7, 8].

Minor contributions to DQF proposed in [5] were made by authors of [9] and [10], while substantial contribution was made by Dagade and Bhonsle, who proposed search optimization for dynamic query form approach or SODQF, which is based on keywords in addition to user feedbacks, as the idea for further development [11]. The keywords based DQF is proposed in [12], which authors also consider queries as items for the collaborative filtering approach in order to provide recommendations. There are also many papers devoted to DQF study, such as [13, 14, 15, 16, 17, 18, 19]. All of this papers are based on previously developed concept of DQF, while proposing some adjustments and improvements to the core idea. For example, in [13] the focus is made on elaboration of differences in DQF application to SQL and NoSQL databases. In [14] authors propose to develop methods to capture user preferences to replace direct feedbacks. Authors of [15] propose an alternative to DQF called auto query forms (AQF) that take queries written in human understandable language and translate them into SQL queries using natural language processing (NLP) methods. Performance measurement of DQF approach was made in paper [16], authors of [17] elaborated the algorithm for DQF generation, and query form formalization was made in [18].

While previously mentioned papers considered mostly SQL baseline and relational databases, in study [19] authors also consider DQF with iterative search and query enhancements by users

feedbacks and responses, but with the use of document-oriented NoSQL database management systems based on JSON (JavaScript Object Notation) structures like MongoDB [19]. Usage of DQF approach for NoSQL databases also considered in research [20], where obtained results of performance comparison between SQL based and NoSQL database management systems demonstrate prevalence of relational databases in terms of querying performance.

Fig. 1 demonstrates rapid increase of DQF research topic popularity after it was mentioned for the first time in 2013 by authors of [5]. However, after 5 years of presumably insufficient results or lack of practical implementations, it does not seem really popular (see Fig. 1). Publications statistics outlined in Fig. 1 was received from Google Scholar as the most comprehensive index of research papers.

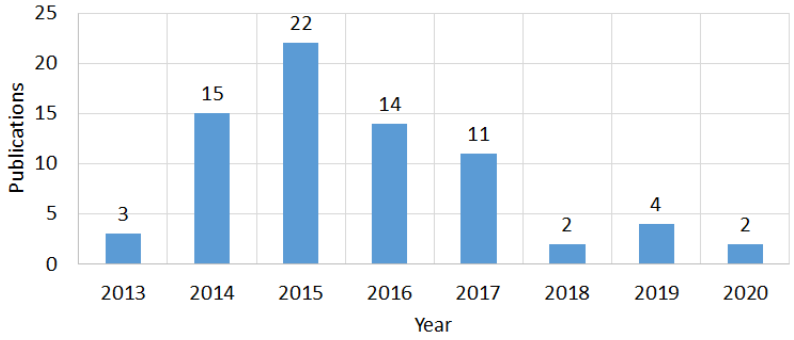

Figure 1: Research interest to DQF

The problems of DQF technique mentioned in two other survey and review papers [21, 22] related to ranking of suggested queries or form components, high complexity of formal querying languages for end users, and inconvenience when working with large database schemas. Considering such problems remaining after years of research and unfair lack of interest for such promising, by our opinion, research topic, the DQF approach should be elaborated in this study in order to develop interactive adaptive software tool to support data analysis activity.

3. Formal Problem Statement

The interactive adaptive software system for data analysis is supposed to be utilized by two types of users: administrator and analyst. Administrator should be responsible for data sources connections and configuration, analysts profiles creation, and management of their permissions. Analyst can run queries to data sources with possibility of further visualization of obtained results. Administrator should be able to use the same functionalities that analyst can use.

When using this system, administrator should configure a workspace for analysts. The configuration process starts with analysis of business requirements received from managers or other stakeholders who are responsible for data-driven decision making. Then, administrator should add required data sources using their properties. Usually these properties are server name, database name, login and password. At the next stage administrator should prepare queries (e.g. using stored procedures on the server side) and provide additional information on their parameters. Then administrator creates user profiles for analysts and grants permissions: which data sources certain analysts could access and which queries they could execute. Querying tools are essential for the proposed system. That is why prepared queries, presented as stored procedures, should be described with the metadata sufficient for their displaying as graphical user interface (GUI) forms.

Using provided GUI analyst could query connected data sources without knowledge of SQL or other database languages. Using different interface components user could provide parameters for attributes demonstration, records filtering, and other stored procedure features. When preparing queries analysts may use instructions of stakeholders and use required data source, query, and corresponding parameters. Formed using GUI query then executed and received data are displayed on the screed ready for further processing and visualization. The system of adaptive interfaces includes recommending techniques for analytical queries. System should observe user activity and capture

statistics of queries execution. Using this statistics analysts may receive recommendations of queries the most similar to queries executed before. Recommendations should be formulated for each data source (e.g. stored procedure). Structural model of such system is demonstrated on the IDEF0 diagram below (Fig. 2).

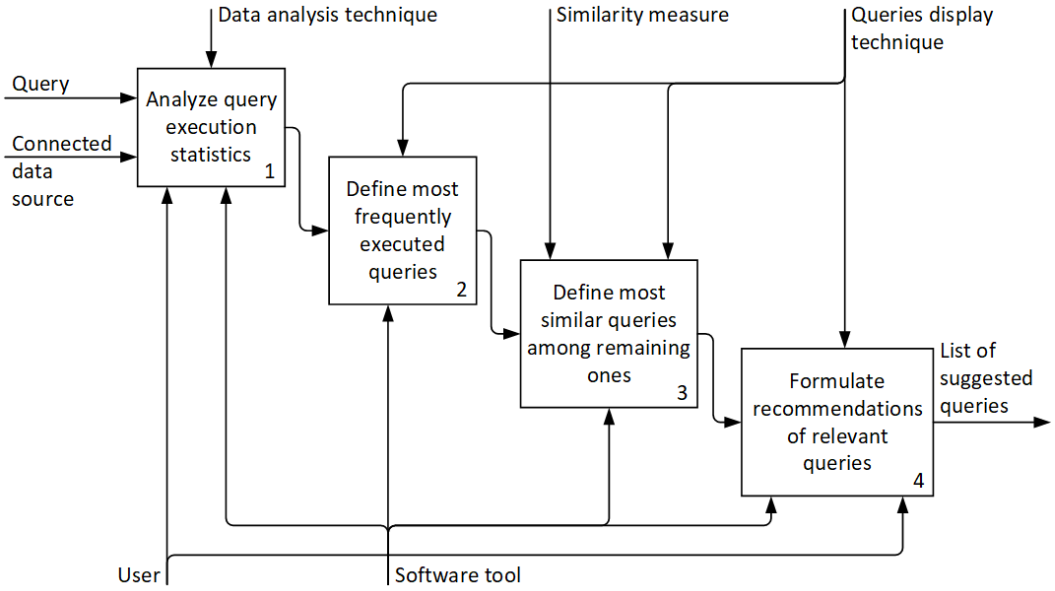

Figure 2: Structural model of the analytical queries recommending system

Recommending algorithm should analyze query execution statistics and define frequencies of query execution in order to detect relevant queries. Three most frequently used queries should be compared to remaining queries in order to detect two most similar queries for each of the most frequently used ones. Therefore, there will be proposed nine most relevant queries for data analytics. Such number of recommended queries (9 items) originates from the statement that human perception allows to take into account 7±2 objects at once. Recommending mechanisms provide interactive querying system that may adapt to user's preferences. Formulated suggestions are optional and not mandatory to follow.

4. Proposed Approach

4.1. Recommending Procedure

Proposed approach is based on the content-based filtering approach. Users get recommendations of those queries, which are similar to the queries executed by them before. The underlying idea considers capturing statistics of queries execution by users and therefore it becomes possible to formulate the list of frequently executed queries for each user by each data source.

Let us briefly describe proposed recommending procedure:

- At first, three most frequent queries should be chosen.
- Then for each of the most frequent queries, selected at first step, should be chosen two most similar to them queries within corresponding data sources. Similarity between two queries could be calculated using analysis of their SQL code.
- As the result, the list of nine recommended queries should be formulated.

Most of recommending systems that support content-based filtering approach use keyword matching or vector space model (VSM) with simple TF-IDF weighting [23].

VSM is the algebraic representation of document collection using vectors that belong to the vector space common for the whole document collection. TF-IDF is the statistical indicator used to evaluate importance of terms (words) within the context of a document that belongs to the document collection (corpus). Weight (importance) of the term is directly proportional to the frequency of such term use in

a document and inversely proportional to the frequency of such term use in remaining documents of the collection [23].

The usage of the considered recommending procedure is demonstrated in Fig. 3.

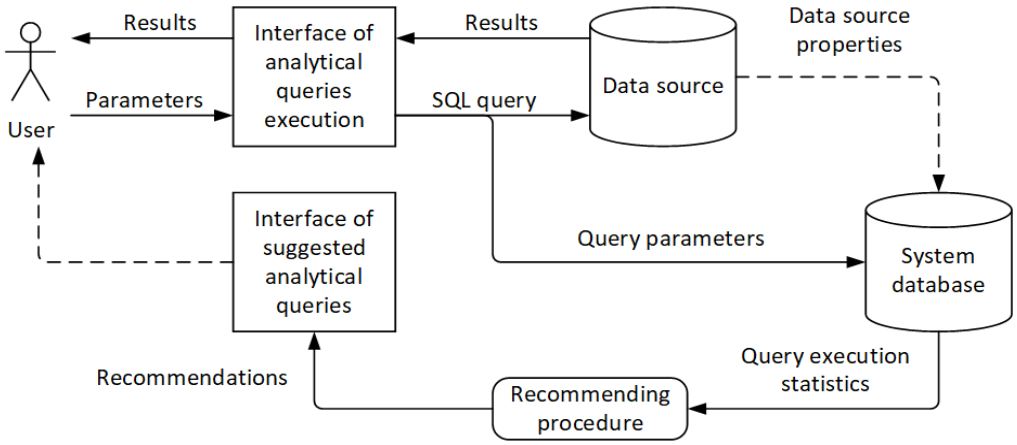

Figure 3: Usage of the recommending procedure

According to the proposed recommending procedure, SQL statements that correspond to queries are considered as documents. Document collection includes the set of all SQL statements accessible to the user within a corresponding data source.

Therefore, formal description of the proposed recommending procedure is based on the pair:

$$\langle D, T \rangle, \tag{1}$$

where:

- $D = \{d_1, d_2, \ldots, d_m\}$ is the collection of SQL statements (documents), d_j, $j = \overline{1,m}$ is the SQL statement (document) represented as the vector of length n, and m is the number of SQL statements (documents) within the collection [23];
- $T = \{t_1, t_2, \ldots, t_n\}$ is the general vocabulary of the document collection, which contains n terms (pairs of parameter names and values of SQL statements that call stored procedures) [23].

Each document (SQL statement) d_j, $j = \overline{1,m}$ of the collection defined in (1) could be represented as the vector of length n:

$$d_j = \{w_{1j}, w_{2j}, \ldots, w_{nj}\}, \tag{2}$$

where w_{kj} is the weight of the term (represented by the pair of parameter name and value of the SQL statement that calls a stored procedure) t_k, $k = \overline{1,n}$ in the document d_j, $j = \overline{1,m}$.

Documents representation using VSM allows weighting terms and detecting similarity of document vectors.

TF-IDF weighting approach is based on the following empirical assumptions [23, 24]:

- Rare terms are not less relevant than frequent terms (so called IDF assumption).
- Multiple occurrences of a term in the document make it less relevant than single occurrence of a term in the document (so called TF assumption).
- Documents of large size do not have advantage over small documents (so called normalization assumption).

Hence, terms that occur in one document, but rarely occur in remaining documents, are more often relevant to the document's topic. Also normalization of result vectors of weights allow avoiding large documents' advantage. These assumptions are demonstrated by the TF-IDF function [24]:

$$\text{TF}-\text{IDF}(t_k, d_j) = \frac{n_j}{\sum_{l=1}^{n} n_l} \cdot \log \frac{m}{n_k}, \tag{3}$$

where:

- m is the number of documents (SQL statements) within the collection;
- n_k is the number of documents within the collection of SQL statements, in which the term t_k, $k = \overline{1,n}$ occurs at least once;
- n_j is the number of times the term t_k, $k = \overline{1,n}$ occurs in the document d_j, $j = \overline{1,m}$ (2);
- n_l is the number of times each term t_l, $l = \overline{1,n}$ occurs in the document d_j, $j = \overline{1,m}$ (2).

In order to obtain weights within the interval $[0,1]$ and documents represented using vectors of the same length, TF-IDF measures obtained using (3) should be normalized [23, 24, 25]:

$$w_{kj} = \text{TF}-\text{IDF}(t_k, d_j) \cdot \left(\sqrt{\sum_{s=1}^{|T|} \text{TF}-\text{IDF}(t_k, d_j)^2} \right)^{-1}, k = \overline{1,n}, j = \overline{1,m}. \quad (4)$$

Cosine similarity measure could be used to calculate similarity between two vectors of normalized TF-IDF measures (4) [23]:

$$sim(d_i, d_j) = \sum_{k=1}^{n} (w_{ki} \cdot w_{kj}) \cdot \left(\sqrt{\sum_{k=1}^{n} w_{ki}^2} \cdot \sqrt{\sum_{k=1}^{n} w_{kj}^2} \right)^{-1}, i \neq j, i = \overline{1,m}, j = \overline{1,m}. \quad (5)$$

Besides cosine similarity, there may be used other measures, such as Jaccard similarity [26] or even adjusted cosine-based similarity measures, such as soft cosine similarity [27].

Using equation (5), the most frequently used queries should be compared to remaining queries that belong to the collection of SQL statements D. For three of the most frequently used queries, remaining queries with the highest similarity scores are then selected in order to provide recommendations within respective data source. In this study we consider data sources as stored procedures also known as executable routines hosted on a database server. Unlike related papers that consider both SQL and NoSQL databases, in this study we focus on relational database management systems (DBMS) only, since among the most widely used and popular DBMS at are still SQL-based systems, such as Oracle, MySQL, Microsoft SQL Server, or PostgreSQL [28]. User-defined stored procedures are created by DBA for encapsulation, security, and performance reasons. On practice stored procedures implement parameterized queries on the server side to provide business logic for various heterogeneous clients. According to the proposed approach, stored procedures should be used as data sources for adaptive interactive querying tools. For each of these data sources (stored procedures) the set of its calls exists. The SQL query to call a stored procedure includes its name and the list or parameters. Sample SQL query that calls the stored procedure to obtain the list of students by the year of admission and country of origin is shown below:

```
EXEC GetStudents @Year = '2020', @Country = 'Turkey';
```
(6)

For example, such stored procedure is called ten times with different parameters. Table 1 shows data gathered using these calls: documents collection $D = \{d_1, d_2, \ldots, d_{10}\}$ of ten SQL queries used to call the stored procedure with different parameters (used as terms in pairs with their names).

Table 1
Sample documents collection

Document	Year	Country	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
1	2018	Morocco	1	0	0	0	1	0	0	0
2	2019	Turkey	0	1	0	0	0	1	0	0
3	2019	Turkey	0	1	0	0	0	1	0	0
4	2018	Turkey	0	1	0	0	1	0	0	0
5	2020	Algeria	0	0	1	0	0	0	1	0
6	2017	Turkey	0	1	0	0	0	0	0	1
7	2017	Morocco	1	0	0	0	0	0	0	1
8	2020	Pakistan	0	0	0	1	0	0	1	0
9	2020	Turkey	0	1	0	0	0	0	1	0
10	2018	Morocco	1	0	0	0	1	0	0	0

Therefore, e.g. documents d_1 and d_4 could be represented using the following vectors:

$$d_1 = \{0.71,0,0,0,0.71,0,0,0\}, d_4 = \{0,0.5,0,0,0.87,0,0,0\}. \quad (7)$$

These vectors were obtained using equations (3) and (4), while similarity between d_1 and d_4 could be calculated using (5) applied to the vectors of document weights (7), $sim(d_1, d_4) = 0.61$.

There is also another document exists, which similarity is greater than zero when compare it to the d_1 document, it is d_7:

$$d_1 = \{0.71,0,0,0,0.71,0,0,0\}, d_7 = \{0.6,0,0,0,0,0,0,0.8\}. \quad (8)$$

Similarity between d_1 and d_7 calculated using (5) applied to the vectors of document weights (8) is $sim(d_1, d_7) = 0.42$. Therefore, there are two queries that should be recommended as relevant to (6):

```
EXEC GetStudents @Year = '2018', @Country = 'Turkey';   (9)
EXEC GetStudents @Year = '2017', @Country = 'Morocco';
```

Obtained queries (9) are similar to (6) by one of the year or country parameter values.

The workflow of recommending procedure is demonstrated in Fig. 4.

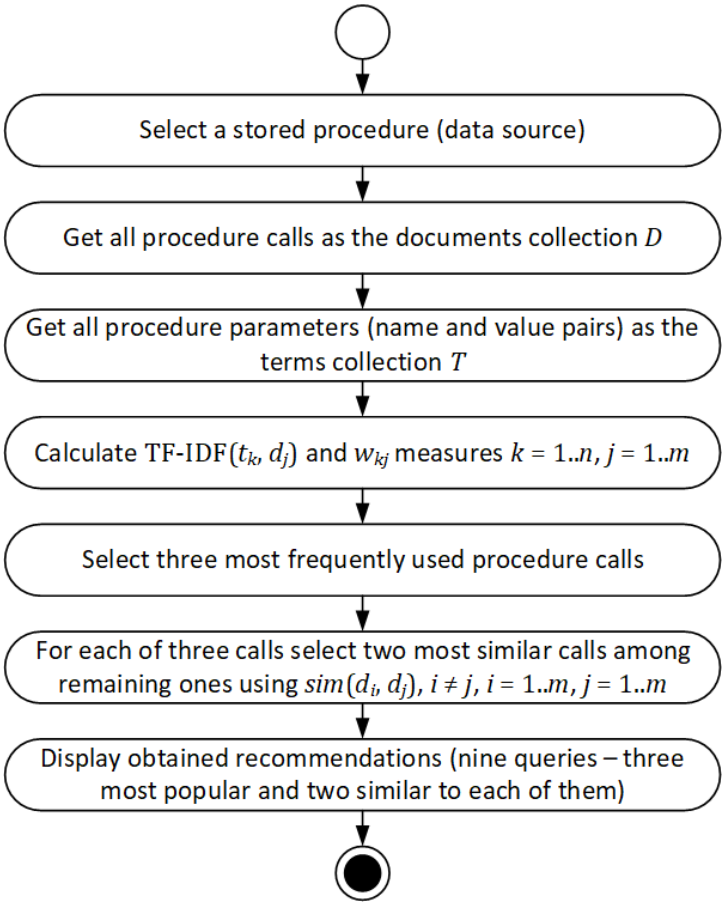

Figure 4: Workflow of the recommending procedure

Demonstrated workflow is the baseline of the interactive adaptive data analytics system proposed in this paper. Software requirements, database schema, and system design are outlined in following sub-section. Prototype demonstration and discussion are given in section 5.

4.2. Software Requirements

There are two kinds of users that may interact with the system of adaptive interactive interfaces:

- Database administrators (DBA).
- Data analysts (DA).

Preliminary domain analysis has led to the functional requirements (FR) for the system under design demonstrated in Table 2.

Table 2
Elicited functional requirements

User	Requirement	Details
DBA	FR1	Manages data source connections (connects, disconnects, and modifies connection properties)
	FR2	Manages analytical queries and corresponding stored procedures (creates new and modifies existing ones)
	FR3	Manages DA profiles (creates, modifies, and disables) and profile privileges for data access
DBA, DA	FR4	Runs available analytical queries within granted data sources
	FR5	Searches over, visualizes, or exports received datasets to Microsoft Excel spreadsheets
	FR6	Receives recommendations and suggestions for analytical queries
	FR7	Searches over available data sources, analytical queries, and visualization tools

As it is outlined in Table 2 above, DBA should be able to access the same functionality accessible to DA. Functional requirements are demonstrated on SysML requirements diagram [29] below (Fig. 5), which clarifies user roles, granted permissions, and responsibility areas.

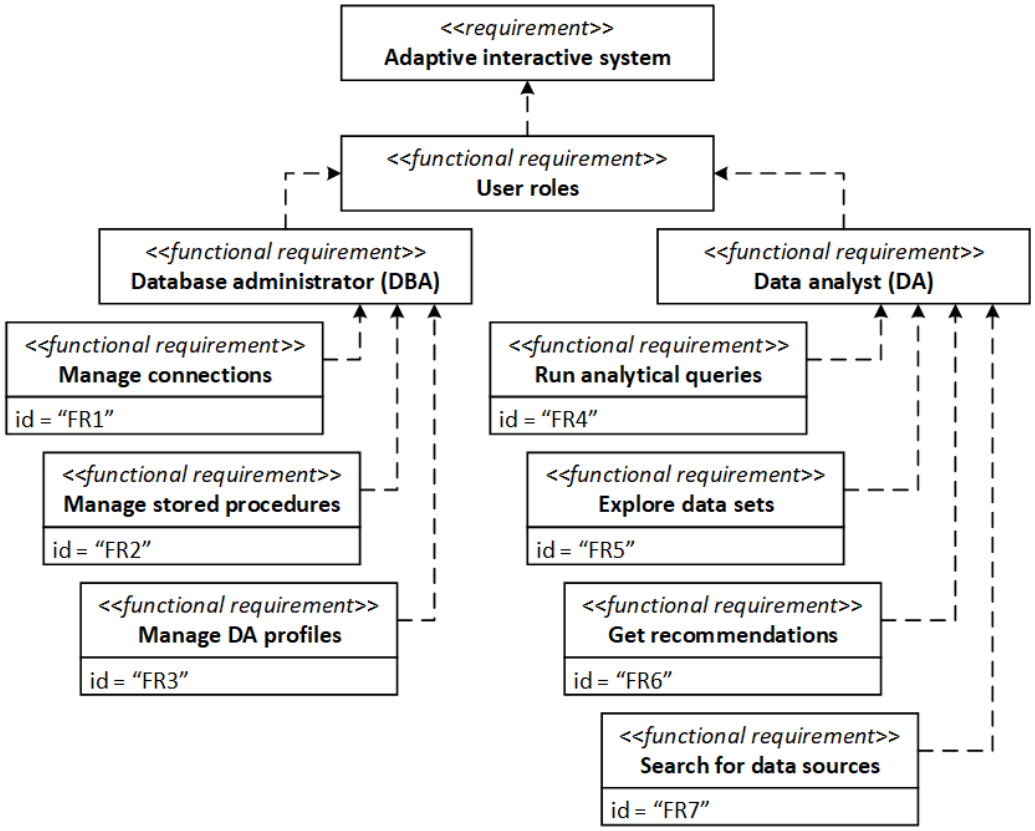

Figure 5: Functional requirements diagram

Also there were formulated non-functional requirements for the system of adaptive interactive tool for data analysis activities. Elicited non-functional requirements (NFR) include performance, usability, security, and reliability requirements (Table 3).

Table 3
Elicited non-functional requirements

Attribute	Requirement	Details
Performance	NFR1	System should respond in a reasonable time even for complex queries and large data sets
Usability	NFR2	Graphical user interface should be friendly and convenient for end users without knowledge of SQL and database schema
Security	NFR3	User's actions should be restricted according to their privileges and responsibility areas (DBA or DA)
	NFR4	Users' accounts should be disabled if suspicious activity detected
Reliability	NFR5	System should be resistant to errors and user's actions should not damage connected data sources
	NFR6	States of user's workspace should be periodically saved for recovery purposes

For each of NFR outlined in Table 3 should be given specific formal or even numerical measures to verify future software system. Non-functional requirements are demonstrated on SysML requirements diagram [29] below (Fig. 6), which clarifies measurable quality attributes.

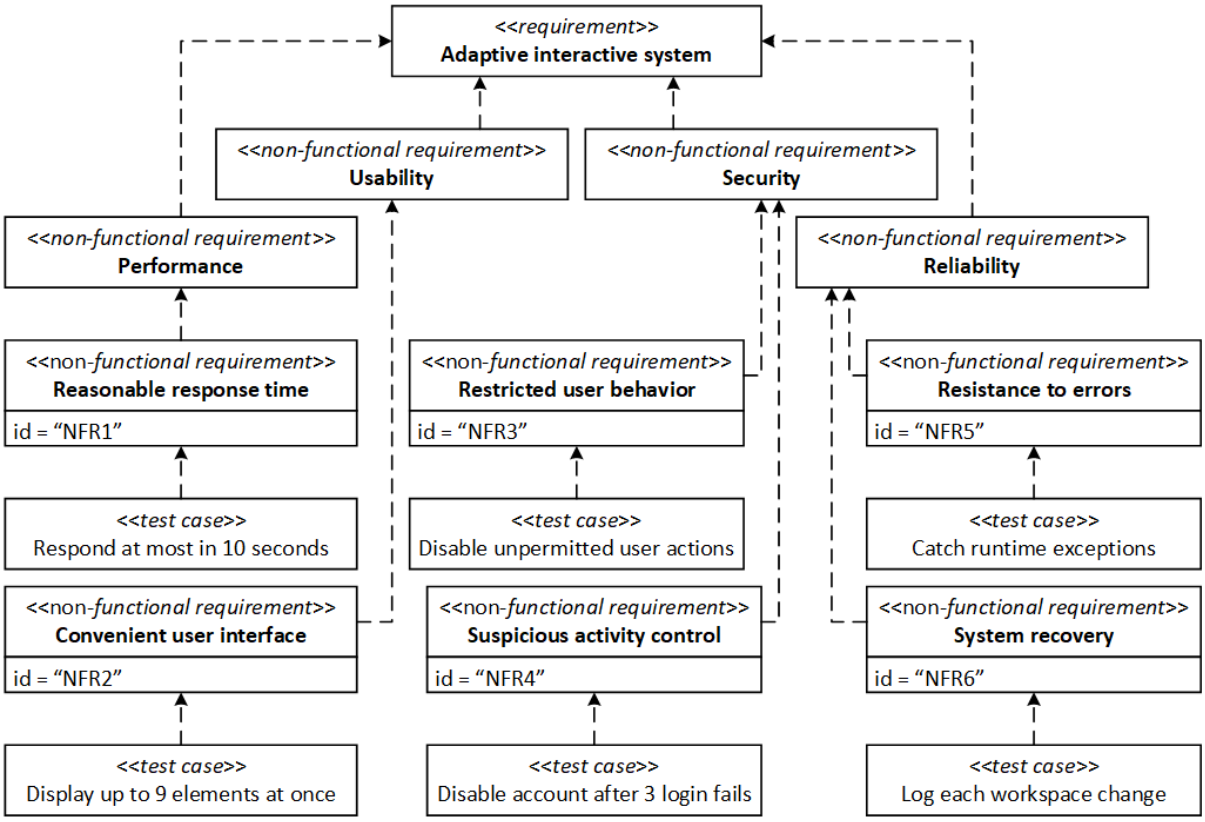

Figure 6: Non-functional requirements diagram

Recommending system should have its own database to store user profiles and privileges, connected data sources, and configured analytical queries.

4.3. Database Structure

Besides storing user profiles, data sources, and queries, the database should contain captured usage statistics to utilize it for recommending purposes.

The data model, which describes database tables and relationships, is shown in Fig. 7.

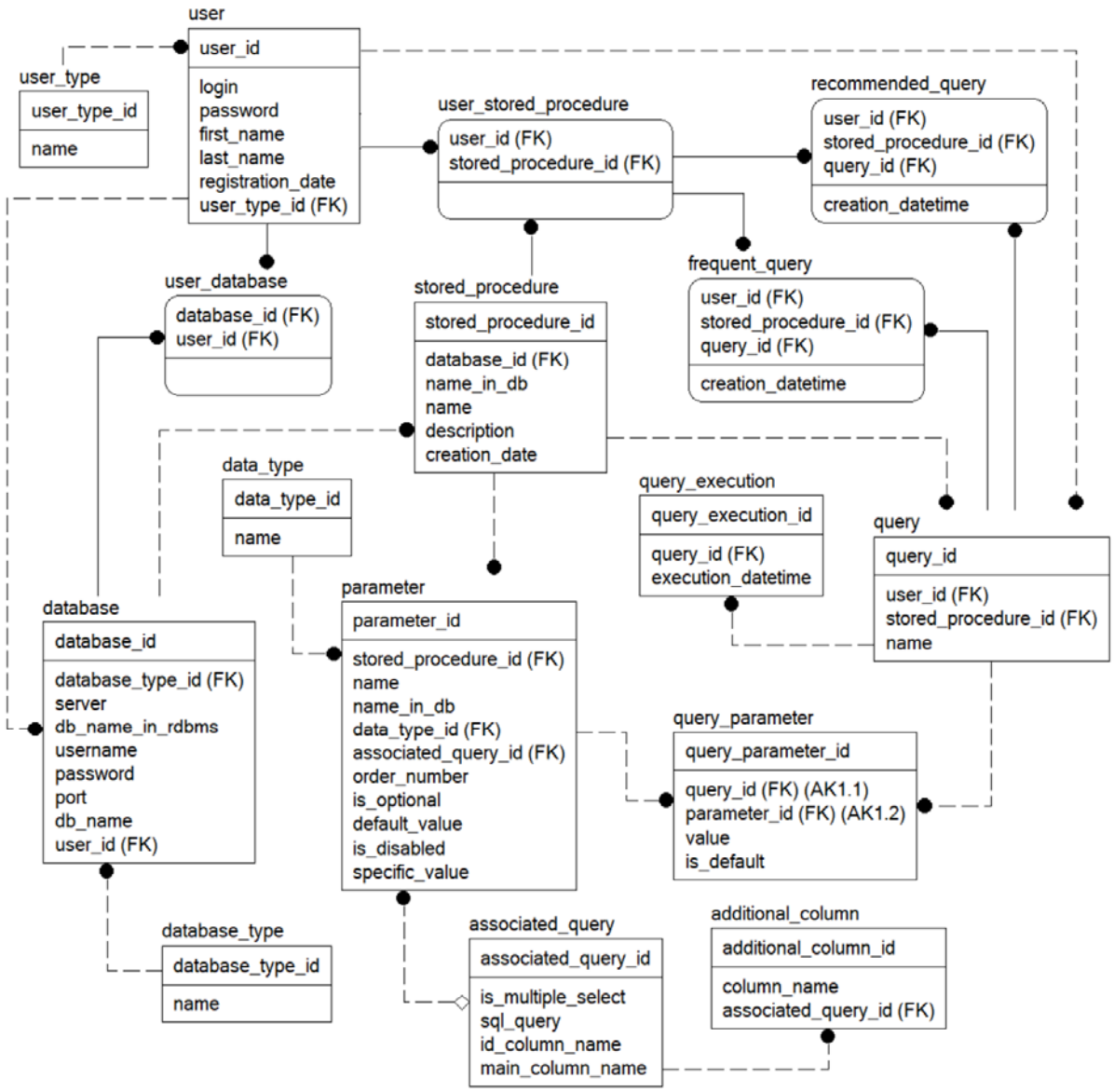

Figure 7: Database structure

Demonstrated data model includes entities for generic user, as well as for administrators and analysts with their own attributes. Administrators attach databases of different types assigned to analysts, stored procedures, and given recommendations. Stored procedures are described by parameter names and data types. Attributes of datasets produced by stored procedures are also considered. Stored procedures are assigned to analysts according to permissions configured by administrators. Each query run is stored in order to capture usage statistics for recommending purposes.

4.4. System Design

Let us start system design description with the dynamic models based on CMMN (Case Management Model and Notation) graphical diagrams [30]. When user selects desired data source from the dropdown list with the search feature, respective stored procedures appear in the left side of application's window, while recommended analytical queries appear in the right side of the window. Administrators can access all attached databases and stored procedures used as data sources for

analytical queries, while analysts can access only databases and stored procedures according to their privileges. Both lists could be filtered by name of a database or stored procedure respectively. After certain query is executed, a modal window is displayed to provide parameters of a stored procedure (to filter produced dataset by rows) and choose desired attributes (to filter produced dataset by columns). After query is executed, result set of records is demonstrated. Recommended queries already have relevant parameters and attributes and do not need to be additionally configured.

Case management notation is the novel technique for description of ad-hoc processes [30], including ad-hoc analytical data querying. The CMMN model that describes query execution using the interactive adaptive software system for data analysis is shown in Fig. 8.

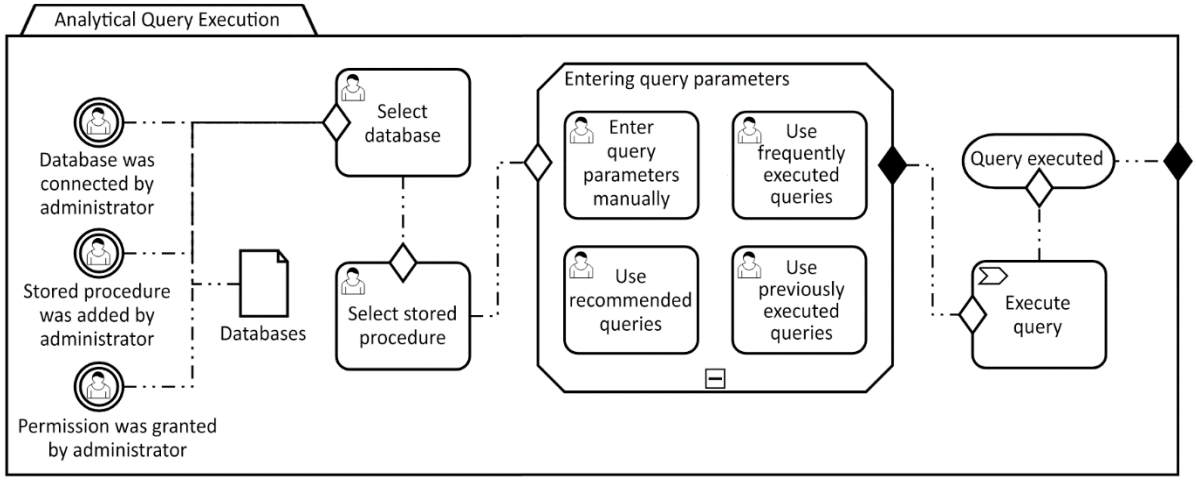

Figure 8: Case management model of the interactive adaptive system for data analytics

User interface structure of analytical querying could be described using the IFML (Interaction Flow Modeling Language) [31] diagram, which describes user interaction, interface structure, and behavior of the software system. The IFML model that describes analytical querying GUI structure and behavior is displayed in Fig. 9.

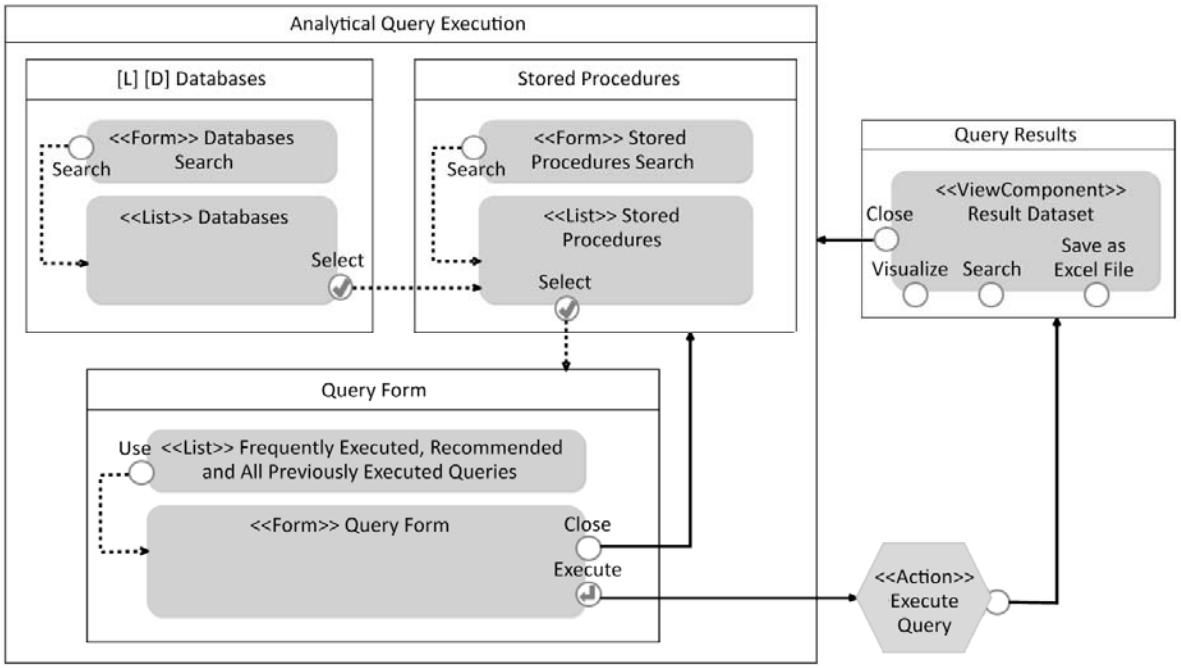

Figure 9: Interaction flow model of the interactive adaptive system for data analytics

Obtained result dataset allows search over, export to Microsoft Excel, or visualization using graphs and charts that fit considered data.

5. Results and Discussion

5.1. Software Architecture

The software prototype was implemented according to design principles of ad-hoc analytical queries support (Fig. 8) and user interaction behavior (Fig. 9) outlined in previous section. The software design is demonstrated in Fig. 10 using components deployment diagram.

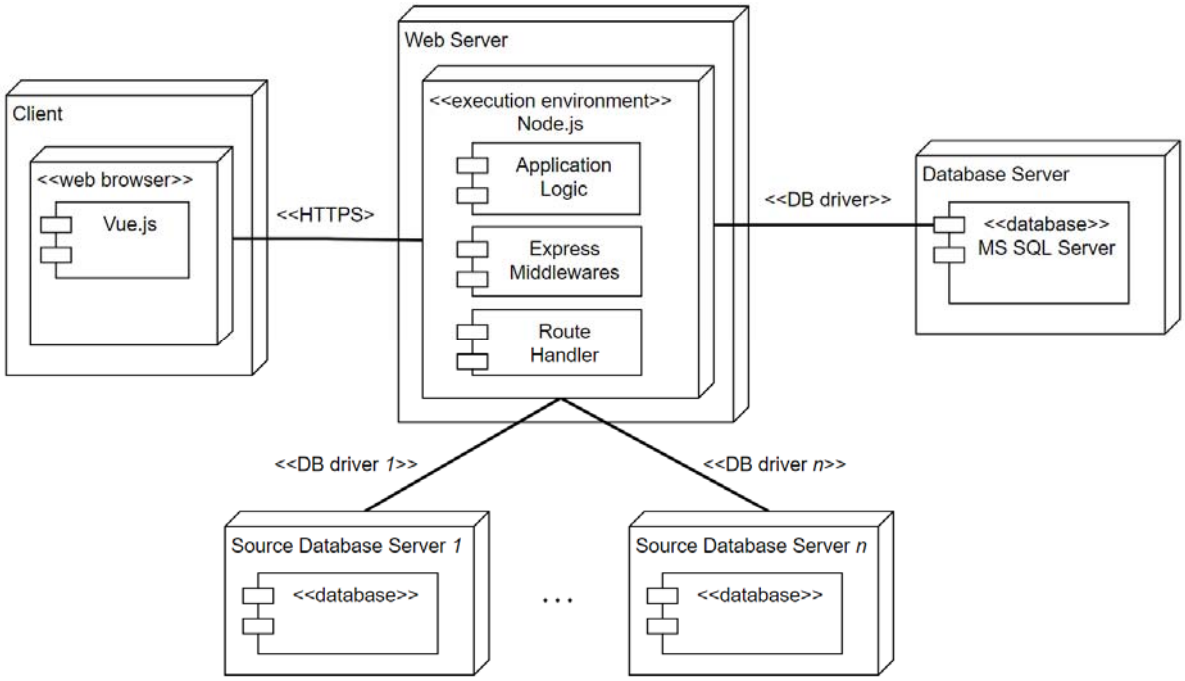

Figure 10: Software architecture model of the interactive adaptive system for data analytics

As it is shown in Fig. 10, the software prototype should be implemented using the three-tier client-server architecture consisting of the following nodes and components:

- Client tier containing only the web-browser as a "thin" client that provides user interface built using the Vue.js JavaScript framework.
- Application server tier containing Node.js back-end application that implements business logic and handles HTTP (HyperText Transfer Protocol) requests, and responses to interact with the client side. Application server uses database drivers to interact with database servers.
- Database server containing Microsoft SQL Server database that stores user profiles and granted permissions, data source connection properties, analytical queries, and usage statistics.

At this moment developed software prototype supports connection to Microsoft SQL Server as the source database server, while it is planned to extend its integration capabilities to support other widely used relational DBMS, such as MySQL or PostgreSQL [28].

5.2. Software Prototype Demonstration

The software prototype currently is under development, however, its generic functionality is already implemented. Fig. 11 demonstrates querying interface that works with connected database servers and stored procedures hosted on these database servers. The user interface is developed according to ad-hoc data querying principles and information flows outlined in CMMN (Fig. 8) and IFML (Fig. 9) models respectively. Generic GUI areas are:

- Database connection properties.
- Stored procedures displayed for each of attached databases.
- Recommended queries displayed for each of stored procedures.
- Querying form that could be filled by users manually for a certain stored procedure or could be filled automatically based on suggested relevant queries.

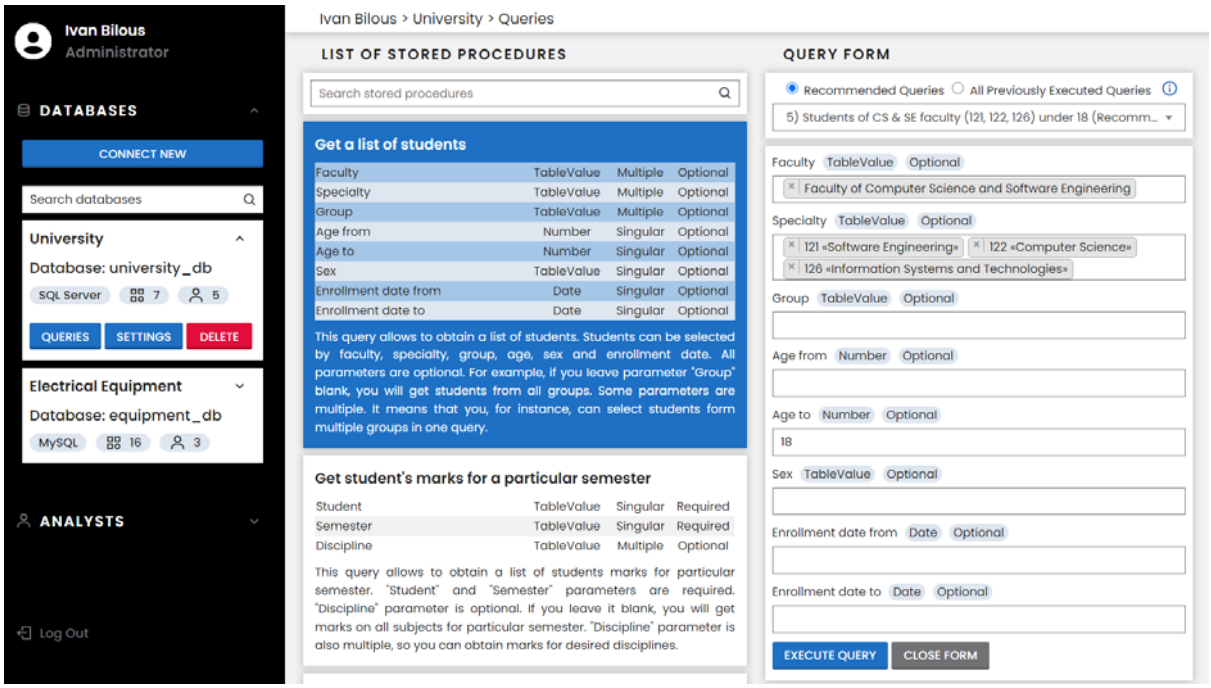

Figure 11: Interactive adaptive querying interface of the software prototype

As it is shown in Fig. 11 above, users first of all should select a database to work with. If necessary, new database connection could be created (this feature shall be used only by DBA and will be invisible for users). It is planned to allow using multiple relational DBMS in the same environment (e.g. MySQL, Microsoft SQL Server, Oracle, and PostgreSQL as the most popular and widely used database systems). However, connected database systems should be of client-server architecture, since using of file-server databases, such as Microsoft Access or SQLite, is not planned, since they usually do not support stored procedures and used as embedded or small ledgers. Database connection properties could be configured by DBA or even removed if necessary. Also DBA can manage stored procedures and users assigned to each of database connection profiles.

When a certain database connection selected, the list of stored procedures available for this database is displayed. Besides the meaningful name that hides a system name of the stored procedure, the list of parameters is outlined for each of stored procedures together with its brief textual description. Each of demonstrated parameters is accompanied with the respective data type, multiplicity, and optionality.

For each of stored procedures users can get recommended queries suggested using recommending procedure introduced in section 4.1 (Fig. 4). The querying form is available for manual input, however, when user selects specific item from the list of recommended queries, the form is filled with respective values of filtering parameters. Such feature simplifies significantly repetitive access to certain data sets with minimum efforts required from the user (when no changes are necessary and recommended query could be executed as-is, or when one or two filtering values should be changed for "what-if" analysis or other ad-hoc purposes). The list of recommended queries includes three most popular ones, ordered by their usage frequencies, and then six queries – two most similar for each of three most popular ones, ordered by their similarity values. However, users may expand the list of suggested queries in order to access all of the relevant queries (beyond the threshold of two most similar queries for each of three most frequently used queries) ordered by their similarity values (to

three most popular queries). In order to distinguish obtained recommendations, queries should be identified by their names given by users when these queries are executed for the first time.

6. Conclusion and Future Work

In this paper we have proposed an approach to development of interactive adaptive software tool for data analysis based on the content-based filtering approach. Outlined recommending procedure captures usage statistics of database queries and suggests the most relevant ones among previously used queries based on usage frequency and similarity criteria. According to the proposed recommending procedure, the system works with SQL queries used to access attached data sources (we use stored procedures or routines as data sources, since they are representing a good practice of database schema securing and encapsulation for usage by multiple applications and clients). These SQL queries that call data sources are treated as documents, while pairs of parameter names and values are used as terms for content-based filtering. Suggested queries as the most relevant to a particular stored procedure then could be used as given with pre-defined parameter values or partially changed parameter values in order to reuse queries executed before, or run some ad-hoc inquires respectively. The software tool that should be developed according to the proposed approach currently is implemented as a prototype with limited functionality. However, it allows to connect Microsoft SQL Server databases to access provided data sources, execute SQL queries, and receive recommendations. In future the software should be completed to fulfill all of the elicited requirements, such as user roles and permissions, and connection to multiple DBMS servers. It is also planned to elaborate recommending procedure by considering advanced filtering techniques, similarity measures, and thresholds in order to provide suggestions adjustable by users.

7. References

[1] SHRM Survey Findings, Jobs of the Future: Data Analysis Skills, 2016. URL: https://www.shrm.org/hr-today/trends-and-forecasting/research-and-surveys/Documents/Data-Analysis-Skills.pdf.

[2] SelectHub, Competitive report: Tableau vs. QlikView vs. Power BI, 2019. URL: https://www.smetricinsights.com/wp-content/uploads/2019/05/Tableau-VS-QlikView-VS-Power-BI-2019-Update.pdf.

[3] J. W. Nicklas et al., Supporting distributed, interactive Jupyter and RStudio in a scheduled HPC environment with Spark using Open OnDemand, in: Proceedings of the Practice and Experience on Advanced Research Computing, PEARC'18, 2018, pp. 1–8. doi: 10.1145/3219104.3219149.

[4] F. Nelli, Python Data Analytics: With Pandas, NumPy, and Matplotlib, Apress, Berkeley, CA, 2018. doi:10.1007/978-1-4842-3913-1.

[5] L. Tang et al., Dynamic query forms for database queries, IEEE transactions on knowledge and data engineering 9(26) (2013) 2166–2178. doi:10.1109/TKDE.2013.62.

[6] P. P. Nikam, A Review on Dynamic Query Forms for Database Queries, International Journal of Computer Science and Information Technologies 5(6) (2014) 8079–8081.

[7] V. Jadhav, A. Priyadarshi, A Survey on Database Queries by using Dynamic Query Forms, International Journal of Science and Research 3(11) (2014) 1157–1159.

[8] S. S. Baravkar, A. Gupta, A Survey on Dynamic Query Forms for Database Queries, International Journal of Science and Research 3(6) (2017) 1566–1568.

[9] M. M. Jisha, M. A. Jacob, Dynamic Query Forms for Database Queries, International Journal of Engineering Research & Technology 5(7) (2016) 217–223.

[10] K. L. Jaiswal, S. S. Joshi, Dynamic Query Form Generation, International Journal of Emerging Technologies in Engineering Research 5(4) (2017) 95–97.

[11] P. Dagade, M. Bhonsle, Espionage on Search Optimization using Dynamic Query Form, International Journal of Engineering Research and General Science 2(6) (2014) 380–386.

[12] G. Patle, R. Uikey, E. Meshram, Ranked Based Dynamic Query Forms for Database Queries, International Journal of Scientific Research in Computer Science, Engineering and Information Technology 5(2) (2019) 1209–1212.

[13] A. V. S. Kumar, O. Devakiran, Dynamic Query Forms for Ad-Hoc Queries on Databases, International Journal of Scientific Engineering and Technology Research 3(36) (2014) 7176–7179.
[14] G. H. K. Reddy, A. Bhattacharjya, S. S. Rawat, S. Reddy, Multiple Methods for Detection of User Priority of Dynamic Query Forms on Databases, International Journal of Scientific Engineering and Technology Research 4(31) (2015) 6005-6008.
[15] R. B. Sangore, P. P. Bhavsar, M. S. Patil, T. M. Chaudhari, An Alternative For Database Queries:Auto Query Forms, International Journal of Advanced Research in Computer and Communication Engineering 4(3) (2015) 94–96. doi:10.17148/IJARCCE.2015.4322
[16] G. Radhakrishnan, S. S. Babu, Creation of Dynamic Query Forms and Ranking of its Components based on User's Preference, International Journal of Innovative Research in Computer and Communication Engineering 3(8) (2015) 7326–7332. doi:10.15680/IJIRCCE.2015.0308028
[17] G. S. G. Pavan Kumar, S. U. Maheswara Rao, Generating Efficiency and Robustness Dynamic Query Forms for Advanced Database Queries, International Journal of Research in Information Technology 4(9) (2016) 42–51.
[18] K. Srinivasarao, G. R. Bharathi, Automated Creation of a Database Queries by using Dynamic Query Forms, International Journal of Scientific Engineering and Technology Research 5(4) (2016) 0609–0612.
[19] K. Ozarkar, R. Rajani, Optimization Technique for Efficient Dynamic Query Forms with NoSQL, International Journal of Science and Research 3(11) (2014) 2041–2044.
[20] D. Pagaro et al., Dynamic Query Forms for Non-Relational Database, International Journal of Advanced Engineering, Management and Science 2(6) (2016) 552–555.
[21] P. S. Revathy, A survey paper on dynamic query forms for database queries, International Journal of Advance Research in Computer Science and Management Studies 2(9) (2014) 65–69.
[22] J. M. Jambukar, M. B. Vaidya, A Review paper on Construction of Query Forms, International Journal of Computer Science and Information Technologies 6(3) (2015) 2426–2428.
[23] F. Ricci, L. Rokach, B. Shapira, B. P. Kantor, Recommender Systems Handbook, Springer US, 2011. doi:10.1007/978-0-387-85820-3
[24] R. Manjula, A. Chilambuchelvan, Content Based Filtering Techniques in Recommendation System using user preferences, International Journal of Innovations in Engineering and Technology 7(4) (2016) 149–154.
[25] J. M. Pazzani, D. Billsus, Content-Based Recommendation Systems, in: Brusilovsky P., Kobsa A., Nejdl W. (Eds.), The Adaptive Web. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2007, pp. 325–341. doi:10.1007/978-3-540-72079-9_10
[26] S. Gupta, Overview of Text Similarity Metrics in Python, 2018. URL: https://towardsdatascience.com/overview-of-text-similarity-metrics-3397c4601f50.
[27] P. Sitikhu, K. Pahi, P. Thapa, S. Shaya, A Comparison of Semantic Similarity Methods for Maximum Human Interpretability, 2019. URL: https://arxiv.org/pdf/1910.09129.pdf.
[28] DB-Engines Ranking. URL: https://db-engines.com/en/ranking.
[29] SysML FAQ: What is a Requirement Diagram and how is it used? URL: https://sysml.org/sysml-faq/what-is-requirement-diagram.html.
[30] Case Management Model and Notation. URL: https://www.omg.org/cmmn/.
[31] Scope – IFML: The Interaction Flow Modeling Language. URL: https://www.ifml.org/scope/.