Design and Analysis of a Relational Database for Behavioral Experiments Data Processing

https://doi.org/10.3991/ijoe.v14i02.7988

Radoslava Kraleva(✉), Velin Kralev, Nina Sinyagina
South-West University, Blagoevgrad, Bulgaria
rady_kraleva@swu.bg

Petia Koprinkova-Hristova, Nadejda Bocheva
Bulgarian Academy of Sciences, Sofia, Bulgaria

Abstract—This paper presents the results of a comparative analysis between different approaches to experimental data storage and processing. Several studies related to the problem and some methods for solving it have been discussed. Different types of databases, ways of using them and the areas of their application are analyzed. For the purposes of the study, a relational database for storing and analyzing specific data from behavioral experiments was designed. The methodology and conditions for conducting the experiments are described. Three different indicators are analyzed, respectively: memory required to store the data, time to load the data from an external file into computer memory and iteration time across all records through one cycle. The obtained results show that for storing a large number of records (in the order of tens of millions rows) either dynamic arrays (stored on external media in binary file format), or an approach based on a local or remote database management system can be used. Regarding the data loading time, the fastest approach was the one that uses dynamic arrays. It outperforms significantly the approaches based on a local or remote database. The obtained results show that the dynamic arrays and the local data sets approaches iterated much faster across all data records than the remote database approach.

Keywords—experimental data; database; relational database; relational database schema design; normal forms; integrity; database rules;

1 Introduction

Conducting experiments and drawing inferences from the accumulated evidence is a must in every scientific study. Nowadays the rapid increase in variety and dimensionality of the collected experimental data sets reveals the problems of their convenient storage, access, sharing, processing and analyzing. In order to provide access to all the collected information and to support its further processing, it is necessary to choose a proper format for data preprocessing and storing. The most commonly used approaches are: first, to store the data in the format it was collected, or second, to reformat the data importing it to a pre-designed local or remote database changing its

format in order to facilitate its further processing. In the second case, access to such database will be done through a database management system (DBMS).

The study is focused on behavioral experiments performed with the participation of human subjects. Stimuli (translating random dot kinematograms) were presented on a monitor screen in a circular aperture. The task of the subject was to indicate the perceived motion direction (left or right) of each presented stimulus by a saccade (very fast jump from one eye position to another) from the current eye position (that is the center of the monitor) towards one of two dots presented close to the left or to the right border of the monitor and to press a mouse button to start the next trial. The aim of the experiment was to collect data about human motion perception and eye movements to stimuli with different characteristics in order to assess reactions of subjects from different age groups. The data collected during the experiments are very specific and they are spread into several files with different formats as follows:

1.1 Files from the eye tracker device

The eye movements of all subjects during the experiment were recorded by a specialized hardware – Jazz novo eyetracking system (Ober Consulting Sp. Z o.o.) and its specialized software. All recordings from all the sensors of the device for one session per person were collected with 1 KHz frequency and the information is stored in files with a specific internal format. These include:

a) Calibration information
b) Records of horizontal and vertical eye positions in degrees of visual angle;
c) Screen sensor signal for presence (value above 300)/absence (value below 300) of a stimulus on the monitor;
d) Microphone signal recording sounds during the experiment (it is aimed at detection of mouse clicks)
e) Information about the tested subjects (name or code) and the type of the experimental trail for each particular record are kept in the metadata of the corresponding file.

1.2 Files with information about generated stimuli and their characteristics.

Each stimulus is composed of some predefined number of frames containing constant number of dots. Each frame lasted for a specified time period. The stimuli were generated in advance but the order of their presentation was random. The coordinates and colors of dots of every frame from a given stimulus are kept in a file. Three experiments were performed that differ within the color of the dots: all white, all black or half of them white and the rest - black. All the information about the parameters of the generated stimuli is kept in another file with the corresponding format. It contains information about:

a) **Coherence of a given stimulus.** The stimuli differ in the proportion of dots moving in a common direction. The parameter specifying this proportion is called co-

herence. For example, when all dots (100%) moved in the same direction, it is said that the stimulus has 100% coherence; if half of the dots moved in a common direction, the coherence of the stimulus was 50% and if all dot movements were random, the coherence is 0%. In the conducted experiments the coherence of the stimuli varied from 0 to 14%. In addition, a stimulus with coherence of 100% was used. Each coherence level was presented 10 times to the tested subjects using 10 different samples for each experimental condition.

b) **Global direction and velocity of dots movement for each stimulus.** In the present experiments the coherently moving dots translated either to the left or to the right.
c) **Life time of dots in the given experiment.** Each dot of a stimulus had a limited lifetime defined by the number of frames it exists (for the current experiment it was three frames). After the end of its lifetime every dot is randomly re-positioned. In every frame one third of the dots were randomly re-positioned.
d) **Number of frames and their lasting time for the current experiment.** Each stimulus had 100 frames of 50 dots. Each frame lasted 33.3ms.

1.3 Other files

Information about order of stimulus presentation during each experimental session, the main stimulus characteristics and the subject's response to each stimulus were kept in an additional file.

All the information has to be pre-processed in a format that allows the users to extract the needed portions of data from different files keeping them synchronized in time. For this aim a relational database structure is proposed.

2 Literature Review

Various aspects related to the use of remote servers for data storage and processing are discussed in several scientific studies. An effective method for authenticating and gaining access to remote servers storing objects and experimental data is discussed in [1]. In [2] a multiuser architecture based on a remote server, enabling the calculation and analysis of experimental data (with the possibility of easy scaling) is presented. Other studies related to the database design process are also discussed in a number of research papers. An effective approach for assessing the integrity of a database is presented in [3]. This approach detects inconsistencies between database logs and the physical state of the data. Similar techniques are presented in [4] and [5]; specialized programs for the recovery of deleted data have been developed for their application, too. A model for designing a distributed database based on integral linear programming is proposed in [6]. Multiple algorithms for processing replications and executing queries for updating data in tables have been reviewed. Other approaches to detecting and analyzing errors during the conceptual data modeling are presented in [7] and [8]. The results show that these approaches can be used successfully in the design of databases. A number of problems related to the determination of functional

dependencies between the data are analyzed in [9]. For the solution to these problems parallel algorithms that are performed for polynomial time are presented. The results obtained are good and persistent when tested with different experimental structures. A performance evaluation model for executing queries is presented in [10]. It can also evaluate the defined rules in a database.

An automated way of designing databases using the entity-relationship approach (ER) is presented in [11]. In addition, a software product has been developed in which this feature is implemented. The results show that the formal specification of the database scheme (based on the ER model) can be transformed into a relational or class model. Also, a Structured Query Language (SQL) code can be generated to create the database and its constraints. The ER model has also been used successfully in designing a relational database for bibliometric information [12]. The usefulness of the design scheme is experimentally proven by analyzing the execution of pre-generated SQL queries. Designing other databases using different approaches and for different application areas is presented in [13] and [14].

In other studies, different approaches are discussed regarding the optimization of the execution of queries in terms of productivity. An algorithm based on the merging of trees is presented in [15]. This algorithm improves the performance of insertion and deletion of database entries. A number of successful experiments with different tree indexes have been done. Another good optimization approach based on index tables is presented in [16]. It enables generating effective SQL queries. A method for improving semantic transformation, editing and executing queries is presented in [17]. Furthermore, a framework providing interoperability with the database through a specialized application programming interface has been developed. After performing a series of experiments on different data sets and different configurations, the results obtained were analyzed. Several heuristic algorithms for optimizing and improving query performance are presented in [18]. In this context, a modification of a genetic algorithm has been proposed and a comparison of different query optimizers has been made. The results show that stochastic approaches have an advantage in terms of execution time.

In other studies various methods of extracting, filtering, and searching for information have been analyzed. For example, a hybrid data filtering approach (by records and columns) is proposed in [19]. The results show that this approach achieves an improvement in performance in terms of execution time for queries. A recursive technique for retrieving information from heterogeneous classifiers (applied to different tables) is presented in [20]. The proposed algorithm has been tested with three different types of data sets; the results show that the data classification time is significantly reduced. Another approach to searching for information is presented in [21]. In this keyword-based approach information is searched in a relational database. Initially, the corresponding query is formulated using the user-defined keywords. Then the data organization (in the relational database scheme) is analyzed. Finally, these two processes are combined. The proposed methodology has been tested experimentally and its effectiveness has been proven.

Another widely discussed aspect of processing and storing information is data security. An analysis of a set of data encryption methods is made in [22]. It has been

found out that the existing methods provide a high level of security but have a significant impact on the system performance. This requires changes to the application layer of the system where the business logic of the application is concentrated. The case where high performance is provided, but there are weaknesses in the system security (and the database, respectively), is also analyzed. In relation to this, a new type of architecture is proposed and a new module is added to the database management system. It allows encryption of each individual value stored in the database. This architecture makes it possible to achieve a high level of data security, while increasing the application layer performance as well. The results show that this type of architecture outperforms the other types in terms of efficiency, security and performance. Another method for enhancing the security of information is proposed in [23]. The goal is to reduce the risk of unauthorized access to data by providing an additional layer of protection. The efficiency of the proposed architecture was evaluated on a set of data containing 1, 000, 000 records. The results obtained show that with the increase in the number of records the time to obtain information (via the Select structure) increases significantly. In contrast, the runtime of Insert, Update, and Delete constructions increases more slowly. An approach to accessing the most commonly used data in a database is proposed in [24]. This approach uses a pessimistic technique to lock certain records in the database. The middle layer tracks all current transactions and co-operates with the other layers of simultaneous transaction execution. The results show that data integrity is maintained while increasing the system performance with easy scaling.

The studies that have been analyzed discuss various aspects of the use of the relational data model. This model was proposed by Edgar F. Codd in 1970 [25]. Currently it is the most widely used model having a serous theoretical fondation. In this model, the data are grouped into relations, called also entities, that are often perceived as tables. Each relation is made up of attributes (columns) and tuples (rows). Each tuple contains a set of interconnected atomic data that describes the properties of an object or an event from the real world. Each tuple in a given relation is uniquely characterized by one attribute (or a combination of several attributes) called a primary key. The primary key uniquely determines the corresponding tuple. The values of the primary key in a single relation are unique. The relationship between two relations is accomplished by a common attribute, which in the first relation represents the primary key, and in the second - foreign key. The number of value quotes in the second (detailed) relation (i.e., the relation in which the foreign key is), determines also the type of relationship between the two relations, for instance: one-to-one, one-to-many and many-to-many. The latter type of relationship is realized through an associative relation in which the primary keys of the relationship are encountered [26].

To sum up, the relational data model provides several advantages over other models: data integrity is provided at the field, relation and database level; logical and physical independence of the data from applications that access them; guaranteed consistency and accuracy of data; an easy and convenient way to retrieve data through queries. On the basis of the above mentioned advantages we selected the relational data model in the present study.

3 Design of the relational database

The schema of the designed relational database entitled “Modeling of Voluntary Saccadic Eye Movements during Decision Making” (Mvsemdm) is presented in Fig. 1.

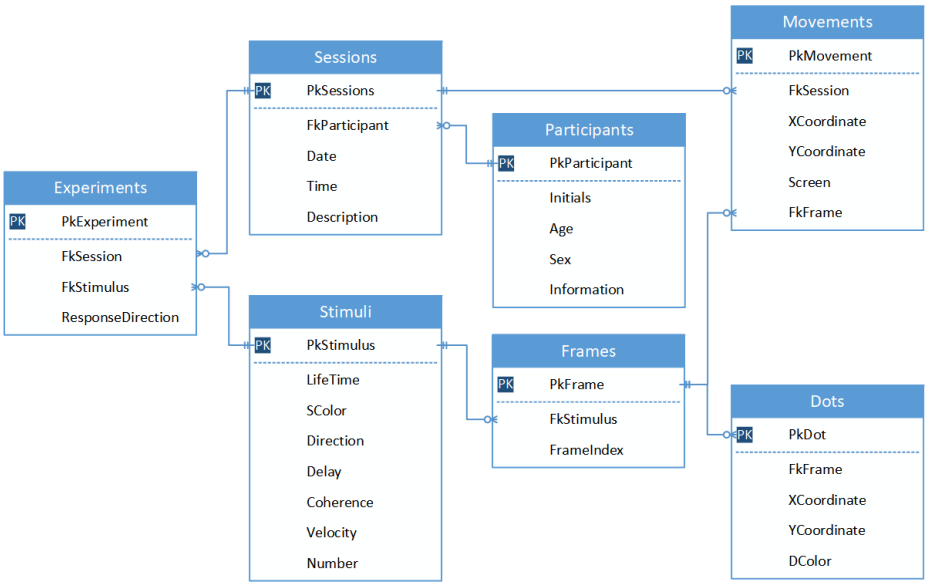

Fig. 1. Structure of the relational database "Mvsemdm"

Each box on the figure above contains one table called here relation of the database. The seven relations designed are:

- **"Participants":** This relation stores information about the participants in the experiments. The characteristics of these participants are their initials, their age and their sex. If it is necessary to store some additional information that is related to the data, it can be added to the field "Information". This field is a "long text" type and cannot be used to search, filter, and sort data in the table. The information in this field is used by the expert who analyzes the data.
- "**Stimuli**": This relation stores information about the global stimulus characteristics, which are: the lifetime of dots (the LifeTime field) and their color (the SColor field); the actual direction of the stimulus (the Direction field); the presentation duration of each frame (in time steps) (the Delay field); the coherence level (the Coherence field); the dots velocity in pixels (the Velocity field), and the number of presented frames per stimulus (the Number field).
- "**Frames**": This relation stores information about a given frame whose index is in the field “FrameIndex”. Through the "FkStimulus" field a many-to-one relationship is established between the "Frames" and "Stimuli" tables. This relationship makes it possible to determine the affiliation of a frame to the respective stimulus.

- "**Dots**":This relation stores information about all dots. The coordinates of each dot are stored in the "XCoordinate" and "YCoordinate" fields, and the corresponding dot color in the "DColor" field. Through the "FkFrame" field a many-to-one relationship is established between the "Dots" and "Frames" tables. This relationship makes it possible to determine the affiliation of a dot to the respective frame.
- "**Sessions**": This relation stores information about the experimental sessions. For this purpose, the Date and Time fields respectively store the date and time of the experiment. The field "FkParticipant" stores the participant code. This field implements a many-to-one relationship between the "Sessions" and "Participants" tables.
- "**Movements**": This relation stores information about the eye movements of the participant during an experimental session. The information is stored in the "XCoordinate" and "YCoordinate" fields and represents the coordinates of the projection of the eye view on the screen. The field "Screen" keeps the value of screen sensor that determines whether at that moment a stimulus is displayed on the screen or there is no stimulus. A many-to-one relationship between the tables "Movements" and "Sessions" was realized through the field "FkSession".
- "**Experiments**": This relation stores information about the experiments conducted by the participants. The participant's response to a particular stimulus is stored in the field "ResponseDirection". This is done through a chain of relationships between primary and foreign keys as follows: PkParticipant - FkParticipant (between "Participants" and "Sessions" tables); PkSession - FkSesson (between "Sessions" and "Experiments" tables); FkStimulus - PkStimulus (between "Experiments" and "Stimuli" tables).

The "FkFrame" and "PkFrame" fields in the "Movements" and "Frames" tables are used to synchronize the recorded coordinates of eye movements and the frames.

The most important relation in the represented relational database is the "Experiments". A many-to-many relationship between the "Sessions" and "Stimuli" tables is realized through the "FkSession" and "FkStimulus" fields because, on the one hand, many experiments can be conducted by one participant, and on the other hand, every stimulus can be shown to many participants. As it was discussed earlier, this type of relations is realized through a third relation, that has foreign keys that correspond to the primary keys of these two relations. In this particular case these are the FkSession and FkStimulus fields from the "Experiments" relation, which is associative. The SQL code for creating the "Experiments" table is presented in Fig. 2.

Analysis of the defined relations showed that all of them have a primary key, do not have multi-valued attributes, and all the data that will be stored are atomic. Therefore, all relations fulfill the requirements of the first normal form (1NF). Furthermore, there are no partial and transitive functional dependencies between non-key attributes and the corresponding key attribute in any of the relations. Therefore, all relations fulfill the requirements of both the second normal form (2NF) and the third normal form (3NF), as this is true for the entire relational database "Mvsemdm".

```
create table [Experiments](
  [PkExperiment] [int] not null,
  [FkSession] [int] not null,
  [FkStimulus] [int] not null,
  [ResponseDirection] [smallint] not null,
  constraint [PK_Experiments] primary key clustered
  ([PkExperiment] asc) with (
     pad_index = off, statistics_norecompute = off,
     ignore_dup_key = off, allow_row_locks = on,
   allow_page_locks = on) on [primary]) on [primary] go
alter table [Experiments] with check add constraint
  [FK_Experiments_Sessions] FOREIGN KEY([FkSession])
  references [Sessions] ([PkSession]) go
alter table [Experiments] check constraint
  [FK_Experiments_Sessions] go
alter table [Experiments] with check add constraint
  [FK_Experiments_Stimulus] foreign key ([FkStimulus])
  references [Stimulus] ([PkStimulus]) go
alter table [Experiments] check constraint
  [FK_Experiments_Stimulus] go
```

Fig. 2. The SQL code for creating the table "Experiments"

4 Experimental results

4.1 Methodology of the experiments

The experiment data is stored in several ways. The aim is to analyze three different indicators, respectively: memory needed to store the data, time to load the data from an external file into a computer's memory, and time for iteration through all records.

The investigated ways of organizing and storing the data were as follows:

1. Record-type structures represented by dynamic arrays in memory and stored in external files in binary format;
2. Client datasets that store the data in external files in a specific file format (used by the client dataset);
3. xml-based datasets that store data in external xml files;
4. A local database management system (MS Access in this case), which stores the data in external files of a structure and format defined by the DBMS;
5. A remote database management system (MS SQL Server in this case), that stores the data in external files with a specific file format.

The experimental conditions were as follow: a computer with 64–bit OS Windows 10 Pro, x64–based processor and hardware configuration: Processor: Intel (R) Core (TM) i7–6700HQ CPU at 2.60 GHz; RAM: 8GB DDR3, and remote server with 64-

bit OS Windows Server 2012R2, x64-based processor and hardware configuration: Processor: Intel(R) Xeon(R) E5645 CPU at 2.40GHz 2.39GHz; RAM: 16GB DDR3.

Since the relation "Movements" contains the biggest amount of data in the database considered here, it was used further for analysis of the three selected indicators of the five data organization approaches.

4.2 Required memory for storing data

The file sizes needed to store the table "Movements" using the investigated five data organization methods depending on the number of records to be kept are presented in Table 1. It was observed that the largest files were generated by the XML-based approach. The sizes of these files are six and a half times larger than the sizes of the files that store information from dynamic arrays. The maximum number of rows (stored in XML-based files) that the testing application was able to save was 4 million records.

A comparison of the file sizes generated by the considered four (out of five) data organization approaches for table "Movements" containing 10 and 80 millions of records respectively is shown in Fig. 3.

Table 1. File size (in KB) of the stored table "Movements" for different number of records using five data organization approaches.

Records	Dynamic Arrays	Client DataSets	XML DataSets	Local Database	Remote Database
1 000 000	13 672	35 157	86 377	35 860	33 792
2 000 000	27 344	70 313	174 924	51 324	64 512
3 000 000	41 016	105 469	263 480	76 696	95 232
4 000 000	**54 688**	**140 626**	**352 021**	**102 060**	**125 952**
5 000 000	68 360	175 782	Insufficient memory for this operation	127 436	156 672
6 000 000	82 032	210 938		152 860	187 392
7 000 000	95 704	246 094		178 200	218 112
8 000 000	109 376	281 251		203 564	247 808
9 000 000	123 047	316 407		228 928	278 528
10 000 000	**136 719**	**351 563**		**254 272**	**309 248**
20 000 000	273 438	Insufficient memory for this operation		507 968	616 448
30 000 000	410 157			561 412	922 624
40 000 000	546 876			1 015 272	1 228 800
50 000 000	683 594			1 268 936	1 536 000
60 000 000	820 313			1 522 628	1 842 176
70 000 000	957 032			1 776 432	2 149 376
80 000 000	**1 093 751**			**2 030 104**	**2 455 552**
90 000 000	1 230 469			Cannot open database.	2 720 030
100 000 000	1 367 188				3 068 928
110 000 000	1 503 907				3 417 826
120 000 000	1 640 626				3 681 280

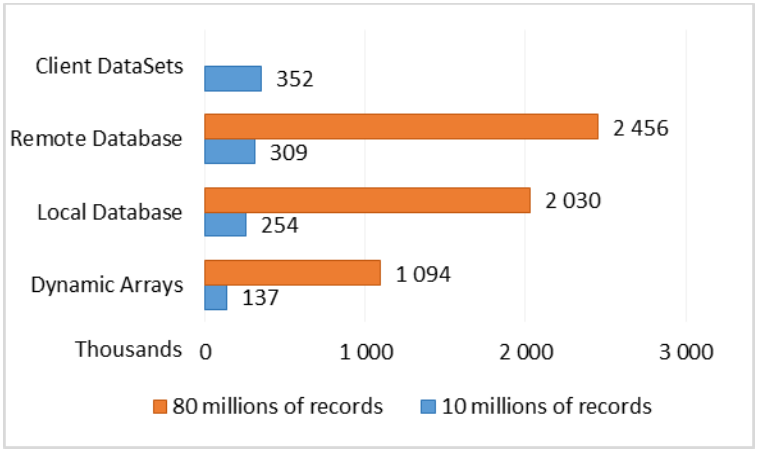

Fig. 3. Comparison of file size (in MB) to store 10 and 80 milions of records for the different data organization approaches.

The obtained results are as follows: increasing the number of records in the "Movements" table, increases the size of the file in which it is stored; the Client and XML DataSets approaches failed to save more that 10 and 4 millions of records respectively while dynamic arrays and database (local or remote) approaches were able to manage much larger data sets. The dynamic arrays approach outperforms other approaches since for the biggest number of records it needed half of the memory to store information in external files in comparison with them.

4.3 Time to load the data from an external file

The loading times of the files containing table "Movements" and stored by the investigated five data storage approaches were measured. In order to accurately measure the time, ten runs of the test program were performed. The average time values in dependence on the number of the stored records are shown in Table 2.

It was observed that the information from XML-based datasets has the slowest loading speed. This can be explained by the fact that the XML-generated file sizes were the largest in comparison with other data organization approaches. Moreover, the self-descriptive syntax of the XML language requires to use two markers (opening and closing) to describe each value.

Loading times of files containing 10 and 80 millions of records respectively and generated by the four (out of five) data storage approaches were compared in Fig. 4.

Linear increase of the records number increases linearly the file loading time in all cases. Again the files generated using dynamic arrays and having a large number of records (in the order of tens of millions rows) were loaded faster than the other four types of files. Moreover, the time required to load dynamic arrays data files is very small in comparison with the files generated by local or remote database approaches.

Table 2. Average data loading time (in milliseconds) for table "Movements" for different number of records using five data organization approaches.

Records	Dynamic Arrays	Client DataSets	XML DataSets	Local Database	Remote Database
1 000 000	0	328	2 516	1 172	1 797
2 000 000	15	656	5 047	2 360	3 422
3 000 000	31	1 078	7 531	3 485	5 266
4 000 000	**46**	**1 407**	**10 063**	**4 547**	**6 968**
5 000 000	47	1 765	Insufficient memory for this operation	5 813	8 594
6 000 000	47	2 093		7 015	10 453
7 000 000	63	2 422		7 687	12 485
8 000 000	78	2 797		9 375	14 219
9 000 000	78	3 110		10 325	16 219
10 000 000	**93**	**3 625**		**11 171**	**16 828**
20 000 000	187	Insufficient memory for this operation		23 062	32 045
30 000 000	250			34 078	50 438
40 000 000	344			52 469	67 704
50 000 000	437			57 625	86 625
60 000 000	516			69 672	109 141
70 000 000	594			85 329	119 016
80 000 000	**672**			**95 047**	**157 482**
90 000 000	750			Cannot open database.	194 230
100 000 000	844				236 244
110 000 000	922				282 168
120 000 000	985				321 357

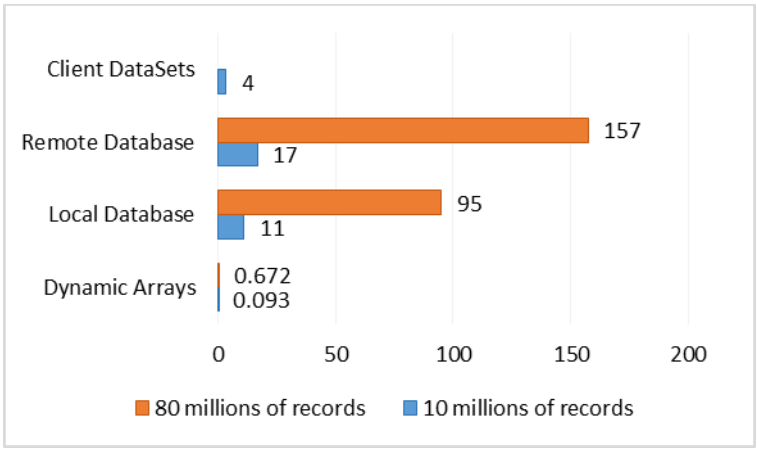

Fig. 4. Comparison of loading time (in seconds) for 10 and 80 millions of records for the different data organization approaches

4.4 Time to iterate through all the records

In order to investigate this indicator all file records were pre-loaded into the computer memory. Ten runs of iterations through all the records were performed and the average time for a single loop was determined. Table 3 shows the results of the time required to loop through all the records in the "Movements" table for the five investigated data storage approaches depending on the number of records.

It was observed that the dynamic arrays and the local data sets neeeded the least time to iterate in comparison with an approach using a database. In the case of a remote database server, the loop time is greater because the data are transferred from the server to the client application over the Internet (or over an internal corporate network). However, when retrieving fewer data (for instance up to 1 million records), loading data time and loop time are comparable with those of local database approach.

Table 3. Average time to iterate through all data records (in milliseconds) for table "Movements" for different number of records using five data organization approaches.

Records	Dynamic Arrays	Client DataSets	XML DataSets	Local Database	Remote Database
1 000 000	16	484	484	2 245	6 844
2 000 000	16	954	969	4 417	13 766
3 000 000	16	1 422	1 437	6 682	20 797
4 000 000	**32**	**1 875**	**1 906**	**8 963**	**27 671**
5 000 000	32	2 375	Insufficient memory for this operation	11 245	33 781
6 000 000	32	2 859		13 453	40 625
7 000 000	32	3 328		15 724	48 656
8 000 000	47	3 812		17 922	55 187
9 000 000	47	4 281		19 828	61 750
10 000 000	**62**	**4 765**		**22 224**	**65 765**
20 000 000	94	Insufficient memory for this operation		44 344	135 484
30 000 000	125			66 542	199 703
40 000 000	171			100 469	275 750
50 000 000	203			119 297	340 047
60 000 000	250			135 094	455 609
70 000 000	297			165 662	468 047
80 000 000	**359**			**198 844**	**571 395**
90 000 000	406			Cannot open database.	687 159
100 000 000	438				821 842
110 000 000	531				932 156
120 000 000	563				1 042 802

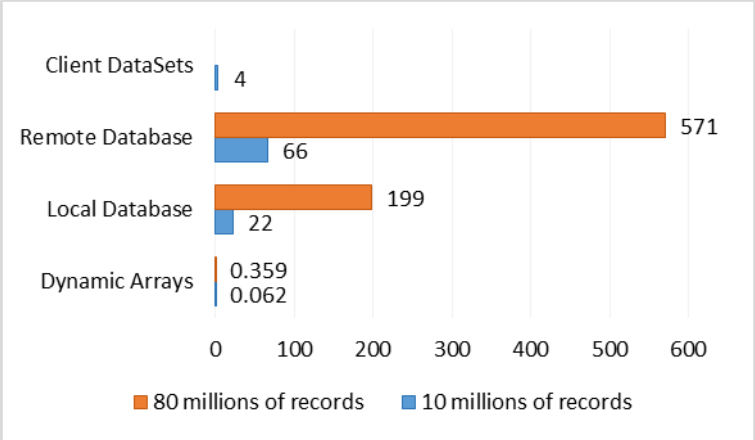

Fig. 5. Comparison of time needed to iterate over all records (in seconds) for 10 and 80 millions of records for the different data organization approaches

For more precise analysis, additional experiments are required, but they are not the subject of this study. We will note that in practice a smaller sets of data are usually used without retrieving all records from a table each time, that leads to processing a smaller part of the data (in the client application). Additionally, the DBMS approach provides the ability to store the most records. Only with this approach it is possible to store more than 120 million records.

From the chart on Fig. 5, it is evident that the linear increase of the records number increases linearly the time for iteration through all the records. Dynamic arrays and client data sets are iterated faster than data sets based on a local or remote database. The difference in performance is hundreds of times, and in some cases even more. For instance, for 80 millions of records, iteration of a data set communicating with a remote database server (via a data provider) takes about 1,500 times more time than if a dynamic array is used.

5 Summary and conslusions

Based on the analysis of the existing approaches for data storage and processing different types of data and the ways of using them as well as the areas of their application were discussed in details. Taking into account the specifics of the accumulated data set from behavioral experiments, a corresponding relational database structure needed for its sharing and analysis was designed. Five different approaches to data storage and processing were tested using that particular database. Three different indicators were analyzed, respectively: memory required to store the data, time to load the data from an external file into computer's memory and iteration time across all records through one loop. The obtained results showed that for storing a large number of records (in the order of tens of millions of rows), either dynamic arrays (stored on external media in binary file format), or an approaches based on a local or

remote database management systems are the proper choice. Regarding the data loading time, the approach using dynamic arrays outperforms significantly the other four approaches. With respect to the iteration time across all records, the obtained results showed that the dynamic arrays and the local data sets were iterated much faster than the data formats generated by database approaches.

6 Acknowledgment

The reported work is a part of and was supported by project № DN02/3 "Modelling of voluntary saccadic eye movements during decision making" funded by the Bulgarian Science Fund.

7 References

[1] Wang, N., Chen, X., Song, G., Parsaei, H. (2015). An experiment scheduler and federated authentication solution for remote laboratory access. International Journal of Online Engineering, 11(3): 20-26. https://doi.org/10.3991/ijoe.v11i3.4554

[2] Limpraptono, F. Y., Ratna, A. A. P., Sudibyo, H. (2013). New architecture of remote laboratories multiuser based on embedded web server. International Journal of Online Engineering, 9(6): 4-11. https://doi.org/10.3991/ijoe.v9i6.2886

[3] Wagner, J., Rasin, A., Glavic, B., Heart, K., Furst, J., Bressan, L., Grier, J. (2017). Carving database storage to detect and trace security breaches. Digital Investigation, 22: 127-136. https://doi.org/10.1016/j.diin.2017.06.006

[4] Kim, J., Park, A., Lee, S. (2016). Recovery method of deleted records and tables from ESE database. Digital Investigation, 18: 118-124. https://doi.org/10.1016/j.diin.2016.04.003

[5] Wagner, J., Rasin, A., Grier, J. (2016). Database image content explorer: Carving data that does not officially exist. Digital Investigation, 18: 97-107. https://doi.org/10.1016/j.diin.2016.04.015

[6] Tosun, U. (2014). Distributed Database Design: A Case Study. Procedia Computer Science, 37: 447-450. https://doi.org/10.1016/j.procs.2014.08.067

[7] Currim, S., Ram, S., Durcikova, A., Currim, F. (2014). Using a knowledge learning framework to predict errors in database design. Information Systems, 40: 11-31. https://doi.org/10.1016/j.is.2013.08.001

[8] Nicolaos, P., Katerina, T. (2015). Simple-talking database development: Let the end-user design a relational schema by using simple words. Computers in Human Behavior, 48: 273-289. https://doi.org/10.1016/j.chb.2015.02.002

[9] Gottlob, G., Pichler, R., Wei, F. (2010). Tractable database design and datalog abduction through bounded treewidth. Information Systems, 35(3): 278-298. https://doi.org/10.1016/j.is.2009.09.003

[10] Osman, R., Awan, I., Woodward, M. E. (2011). QuePED: Revisiting queueing networks for the performance evaluation of database designs. Simulation Modelling Practice and Theory, 19(1): 251-270. https://doi.org/10.1016/j.simpat.2010.06.010

[11] Dimitrieski, V., Celikovic, M., Aleksic, S., Ristic, S., Alargt, A., Lukovic, I. (2015). Concepts and evaluation of the extended entity-relationship approach to database design in a multi-paradigm information system modeling tool. Computer Languages, Systems & Structures, 44(C): 299-318. https://doi.org/10.1016/j.cl.2015.08.011

[12] Mallig, N. (2010). A relational database for bibliometric analysis. Journal of Informetrics, 4(4): 564-580. https://doi.org/10.1016/j.joi.2010.06.007
[13] Kralev, V., Kraleva, R. (2017). Approaches to Designing Relational Databases. 7th International Conference "Modern Trends in Science", FMNS-2017, Blageovgrad, Bulgaria. p 110.
[14] Kraleva, R. (2011). Design and development a children's speech database. 4th International Scientific Conference "Mathematics and Natural Sciences", FMNS-2011, Blageovgrad, Bulgaria, 2: 41-48.
[15] Suei, P-L., Lee, V. C. S., Lo, S-W., Kuo, T-W. (2013). An efficient B+-tree design for main-memory database systems with strong access locality. Information Sciences, 232: 325-345. https://doi.org/10.1016/j.ins.2012.12.018
[16] Suei, P-L., Lu, Y-F., Liao, R-J., Lo, S-W. (2012). A signature-based Grid index design for main-memory RFID database applications. Journal of Systems and Software, 85(5): 1205-1212. https://doi.org/10.1016/j.jss.2012.01.026
[17] Vavliakis, K. N., Symeonidis, A. L., Karagiannis, G. T., Mitkas, P. A. (2011). An integrated framework for enhancing the semantic transformation, editing and querying of relational databases. Expert Systems with Applications, 38(4): 3844-3856. https://doi.org/10.1016/j.eswa.2010.09.045
[18] Sharma, M., Singh, G., Singh, R. (2016). Design and analysis of stochastic DSS query optimizers in a distributed database system. Egyptian Informatics Journal, 17(2): 161-173. https://doi.org/10.1016/j.eij.2015.10.003
[19] Afify, G. M., Bastawissy, A. E., Hegazy, O. M. (2015). A hybrid filtering approach for storage optimization in main-memory cloud database. Egyptian Informatics Journal, 16(3): 329-337. https://doi.org/10.1016/j.eij.2015.06.007
[20] Manjunath, G., Murty, M. N., Sitaram, D. (2013). Combining heterogeneous classifiers for relational databases. Pattern Recognition, 46(1): 317-324. https://doi.org/10.1016/j.patcog.2012.06.015
[21] Bergamaschi, S., Guerra, F., Interlandi, M., Trillo-Lado, R., Velegrakis, Y. (2016). Combining user and database perspective for solving keyword queries over relational databases. Information Systems, 55: 1-19. https://doi.org/10.1016/j.is.2015.07.005
[22] Shmueli, E., Vaisenberg, R., Gudes, E., Elovici, Y. (2014). Implementing a database encryption solution, design and implementation issues. Computers & Security, 44: 33-50. https://doi.org/10.1016/j.cose.2014.03.011
[23] Trivedi, D., Zavarsky, P., Butakov, S. (2016). Enhancing Relational Database Security by Metadata Segregation. Procedia Computer Science, 94: 453-458. https://doi.org/10.1016/j.procs.2016.08.070
[24] Lotfy, A. E., Saleh, A. I., El-Ghareeb, H. A., Ali, H. A. (2016). A middle layer solution to support ACID properties for NoSQL databases. Journal of King Saud University - Computer and Information Sciences, 28(1): 133-145. https://doi.org/10.1016/j.jksuci.2015.05.003
[25] Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, 13(6): 377-387. https://doi.org/10.1145/362384.362685
[26] Date, C. J. (2012). Database Design and Relational Theory (Theory in Practice). O'Reilly Media.

8 Authors

Radoslava Kraleva defended her PhD Thesis "Acoustic-Phonetic Modeling for Children's Speech Recognition in Bulgarian" in 2014. Most of her publications are focused on the study of children and their interaction with computer technologies, including recognition of children's speech in Bulgarian. Currently, she is working on her post-doctoral research related to the design and development of interactive mobile applications for children and the study of human-computer interface for young children. Now she is a chief assistant professor at the Departement of Informatics, South-West University in Bulgaria. She has developed several academic courses and curricula.

Velin Kralev obtained a PhD degree in Informatics from the South-West University, Blagoevgrad, Bulgaria in 2010. Since 2004 he has been a chief assistant professor at the South-West University, Blagoevgrad, Bulgaria. His research interests include modeling, synthesis and analysis of various algorithms, development of client-server and multi-tier computing information systems, through the use of component-oriented software technology, database design, and object-oriented programming.

Nina Sinyagina has received her PhD degree in St. Petersburg Technical University in 1972. From 1998 she was Director of a research group in the Institute for parallel processing of information, Bulgarian Academy of Sciences. Since 2004 until now she has been a professor at the South-West University, Blagoevgrad. Her main research areas are: Pattern recognition, Artificial Intelligence, Computer architectures, Database, Computer Security, Operational Systems, and Computer Networks.

Petia Koprinkova-Hristova received MSc degree in Biotechnics from the Technical University - Sofia in 1989 and PhD degree on Process Automation from Bulgarian Academy of Sciences in 2001. Since 2003 she has been an Associate Professor at the Institute of Control and System Research and from January 2012 - at the Institute of Information and Communication Technologies, Bulgarian Academy of Sciences. Her main research interests are in the field of Intelligent Systems using mainly neural networks, fuzzy and neuro-fuzzy approaches. Currently she is a member of the European Neural Network Society (ENNS) executive committee and member of the Union of Automatics and Informatics in Bulgaria.

Nadejda Bocheva is an Associate Professor at Institute of Neurobiology (former Institute of Physiology), Bulgarian Academy of Sciences. She received a PhDegree in biology in 1986 from the Institute of Physiology, Bulgarian Academy of Sciences. In 2006 she became an Associate Professor in psychophysiology. At present she is head of Department of Sensory Neurobiology at the Institute of Neurobiology. Her research interests are in human visual information processing, spatial vision, motion perception, visual recovery of 3D shape, cognitive neuroscience and aging. She is a Fulbright fellow and was awarded a Fogarthy international collaborative Award in 2002.She is a member of the American Psychological Association and of the Sofia section of the Bulgarian Physiological Society.

Article submitted 15 November 2017. Resubmitted 30 December 2017. Final acceptance 05 February 2018. Final version published as submitted by the authors.