STUDIA INFORMATICA 2013
Volume 34 Number 2B (112)

Lech TUZINKIEWICZ
Politechnika Wrocławska, Instytut Informatyki
Paweł WIATER
CUBE.ITG, Dział rozwoju bankowości

ANALYSIS AND COMPARISON OF RELATIONAL AND NON-RELATIONAL DATABASES

Summary. This paper presents a method that allows comparison of the quality of different DBMS systems and implemented in these systems data models. The proposed working model is a subset of quality characteristics of quality models defined in ISO / IEC 25010: 2011 (Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQUARE) – System and software quality models). The whole is illustrated by an example of the application of the presented method.

Keywords: data model, quality model, DBMS

ANALIZA I PORÓWNANIE RELACYJNYCH I NIERELACYJNYCH BAZ DANYCH

Streszczenie. W artykule przedstawiono metodę umożliwiającą porównanie pod względem jakościowym różnych systemów DBMS i implementowanych przez nie modeli danych. Zaproponowany w pracy model jakości jest podzbiorem zbioru charakterystyk modeli jakości zdefiniowanych w normie ISO/IEC 25010: 2011 (Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models). Całość została zilustrowana przykładem zastosowania przedstawionej metody.

Słowa kluczowe: model danych, model jakości, SZBD

1. Introduction

Data and information is a valuable asset of any company. Therefore, having current data and at the same time the possibility of storing it in a secure manner is a very important aspect of creating and maintaining databases. An equally important element in process of database assessment is the issue of processing data by applications which support the implementation of business processes. Currently there is a discussion on various forums about the future of relational databases in the context of a growing number of support systems for non-relational data models and their advantages regarding large volumes of data (MongoDB, CouchDB, Casandra, etc.). These discussions are often spontaneous, and the expressed opinions are highly subjective and likely to be a result of positive or negative experience gained from using a well-known database management system, and thus assessed the data model.

Non-relational databases (no SQL) are represented by the class of database management systems which do not operate on the relational model of data using SQL. This type of databases is often recommended when there is a problem with a lot of data and their frequent increments, i.e. in the case of performance problems. In practice, non-relational systems are being increasingly used in other situations, dangerously forgetting the fact that there is no definition of the objects on the meta-level (no database schema). This applies to the consistency and integrity of data in databases. NoSQL systems do not guarantee the implementation of operations in accordance with ACID. Most frequently the atomicity of operations on the data is limited to single objects (both simple and complex), and the integrity and consistency of global data is transferred to the application level. The advantage of this solution is high processing performance and flexibility in defining the required data structures in the database depending on the used objects in the programming languages in which the software is created. This is supposed to contribute to faster software development. There is no need to create a database in a separate manufacturing cycle relative phase encoding (especially in the “agile” approach). Modeling data in NoSQL databases usually starts with defining a query-driven application (application-driven data modeling or application-specific access patterns). But it should also be taken into account that this is one of the aspects of database design. Another aspect is the type of operation performed on the data stored in the database. In the case of cross-cutting requirements for the creation of reports based on aggregations, performance may not be optimal in this type of databases. Thus, the problem of which data model and database management system we should choose in the context of system development remains unresolved. Taking into account these arguments, it seems reasonable to develop rules for the assessment of data and models and for the support of their management systems.

Choosing the right data model and database management system is a difficult task, because of the many factors that affect the manner and quality of the solutions to the problems. There are some data models which can be used to store and retrieve data in business activity. So it is important to consider both the structure and the flexibility of a given database system (DBMS and implemented data model) taking into account the available tools for creating, storing, updating, and retrieving data as well as their maintainability.

When choosing a database structure, the users are required to identify:

- What kind of information should be stored and processed in a database?
- What kind of operations will be performed on the data?

Knowing these factors in advance can help to determine which type of database will be most appropriate and effective in the context of considered software applications. However this raises the question of how and to what extent different DBMS and data models should be analyzed, compared and evaluated.

The paper proposes a quality model that can assess the comparison of various data models. It is important to identify the characteristics essential in deciding both which data model and database management system to choose, in terms of ongoing projects. The analysis was conducted in the context of relational and non-relational models. An example of performance quality model was presented to evaluate the quality of the sample database environments: Oracle (relational model), MongoDB (NoSQL) and GT.M (hierarchical model).

2. Taxonomy of non-relational database model

The term NoSQL does not identify a specific solution but rather the group of databases which are alternative to relational model. NoSQL is a database management system based on the non-relational model. Data is stored in a different structure to tables. There are a few specific database groups. It is difficult to analyze non-relational database models because they do not have a precise definition. There are many types of NoSQL data models. The most popular are shortly characterized below:

- **Column type database** stores data in columns. This solution increases the efficiency in finding specific column information and works well with big data warehouses and analytical system where a lot of data is aggregated. The same type of data in a column allows algorithms to compress data.

- **Key-Value type database** can be introduced as a table with two columns where the first one is a key and the second is a value for that key. Its simplicity allows for fast read and writes speed.
- **Document type database** which stores the form of a document. A document is a set of unordered pairs of key-value grouped into collections. In this model the stored data is the most important and not the data structure.
- **Graph type database** was introduced to store massive quantities of closely related data, e.g. social relations, public transport links, road maps or network topologies
- **XML type database** is the result of popularization of standards of XML language in network service communication and configuration files. XML language is a common language for all different types of NoSQL databases. This type stores XML data. A special query language Xquery and Xpath was introduced to find data in XML files [3], [6].

3. Comparison of relational and non-relational models

Comparison of data models and DBMSs begins with defining the evaluation criteria. Both relational and NoSQL databases differ in many aspects – among others data structure, data manipulation languages [11], [12]. Advantages and disadvantages of relational and NoSQL data models are presented in Table 1.

Table 1

General characteristics of comparing the two types of data models

	Relational	**Non-Relational**
Advantages	• Persistent Data • Data definition at meta-level • Concurrency • Integration constrains • Standards • Maturity • Popularity • Transactional processing	• Open– source • Satisfactory performance of big data • Scalable • Driven by need to run on clusters • Based on needs of 21st web demand • Schemaless – ease of creation • Diminished need for DBA • Performance
Disadvantages	• Impedance Mismatch • Scalability – costly	• Lack of maturity • Analytical and Business Intelligence • Schemaless – data integrity problem

Evaluation will be performed on three different types of databases systems (data models): relational (Oracle) [1], [11] hierarchical (GT.M) [4], and NoSQL (MongoDB) [5], [9].

GT.M is qualified as key-value NoSQL type database (from structure point). Internal characteristics like transactional process, security point resemblance to relational data model.

Detail characteristics of chosen models are presented in the Table 2. Introduced characteristics are used to evaluate these models.

Table 2

Detailed characteristics of compared data models

	Relational/Oracle	**Hierarchical/GT.M**	**NoSQL/MongoDB**
Rapid growth and unpredictable demand	Scalability at high cost end is generally reconsidered as problematic	Massively scalable	Great scalable, sharding mechanism great distributing and controlling mechanism
Cost	High cost: Heavy hardware requirements; Administration; License	Depends on Operating system – Open source; Administration	Open source
Ease of Maintenance	DBA essential	Limited DBA requirement	Limited DBA requirement
Ultra-high reliability and resilience	DBA essential	Ready to use solutions for specific problem domains (financial, healthcare)	Web applications
Structural Flexibility	Schemas are rigid	Dynamic model, easiness of modification data model	Schemaless dynamic model
Performance	Depends on Normalization level and necessity of joins	Strongly depends on problem solution	Excellent aggregated data problem with cross-section operation
Stability	Stable	Stable	Not proven stability
Development	Database design prior to use in application	Development on flight	Development on fight
Scalability	Up	Out	Out
Query Language	SQL	Build-in scripting language M	JavaScript
Data integrity	Provided by DBMS	DBMS + application	Application
Transactional processing	ACID	ACID	BASE
Unicode support	YES	YES	YES
Security	Advance Transparent, Authentication, Authorization, Roles and Permissions, Data Encryption (TDE)	Authentication, Authorization, Roles and Permissions , Data Encrypted	Restricted security
Indexing/Composite Key	YES	YES	Yes
Operating System	Cross-platform	Cross-platform	Cross-platform
Distribution/Replication	YES	YES	YES

4. Quality model

The quality of the database and used models are useful for specifying quality requirements, establishing measures, and performing quality evaluations. The characteristics of quality models make it possible to choose an appropriate set of attributes to compare different data models. Two proposed quality models have been defined based on the standard ISO/IEC 25010 [7]. The first one the Database Quality Model is presented in figure 1.

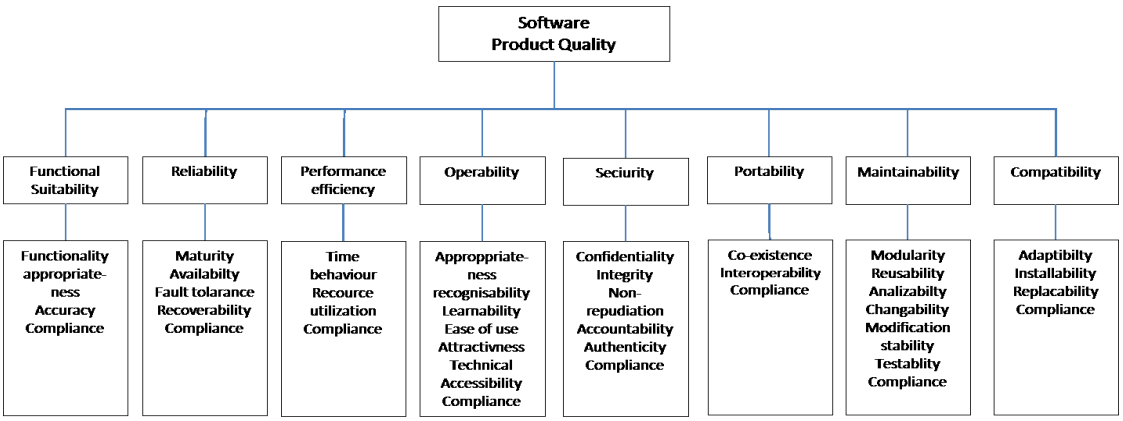

Fig. 1. Database Quality Model based on ISO/IEC 25010 [7]
Rys. 1. Model jakości bazy danych na podstawie ISO/IEC 25010 [7]

The second Database Quality in use model is presented in figure 2.

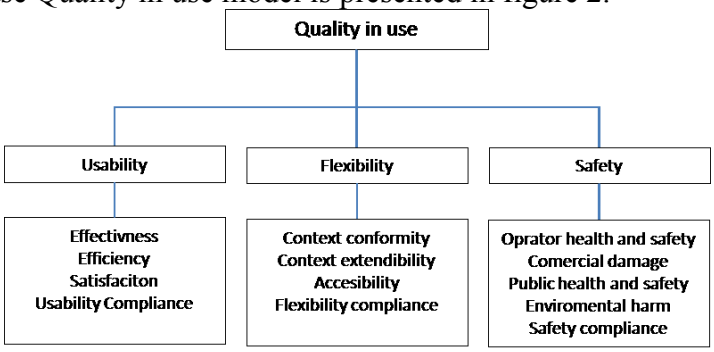

Fig. 2. Database Quality in use model based on ISO/IEC 25010 [7]
Rys. 2. Użytkowy model jakości bazy danych na podstawie ISO/IEC 25010 [7]

Introduced quality model allows to us create instance quality model that will be used in presented example to compare mentioned above database systems.

5. Example of quality comparison analysis of chosen data models

The chosen DBMS and data models implemented in these systems were evaluated based on the instance of the quality model (selected subset of characteristics). For this purpose

a conceptual "Demographics" data model has been defined (Fig. 3), which consists of a set of sample classes and relationships between them (one to one, one to many, many to many).

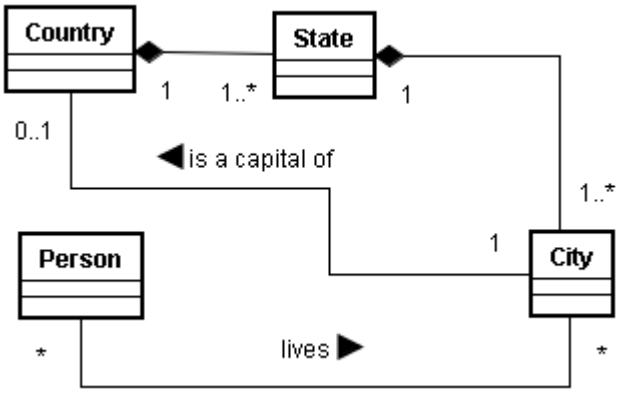

Fig. 3. Exemplary "Demographics" conceptual data model
Rys. 3. Wzorcowy, konceptualny model danych „Demograpics"

This model has been implemented in the three systems to be evaluated and populated with data order to measure the time response of queries (Table 3).

Table 3

Size of data samples used in the measurement processes

No.	Entity type	Sample 01	Sample 02	Sample 03	Sample 04	Sample 05
1.	Country	3	3	3	3	3
2.	State	30	30	30	30	30
3.	City	35	35	35	35	35
4.	Person	30000	100000	500000	1000000	5000000

Table 4

Measurement results of response time performed on the sample data

Command	Data Model	Execution time [s]				
		~30,000	**~100,000**	**~500,000**	**~1000000**	**~5000000**
Insert	**Oracle**	1.5	6.1	29.033	58.32	296.80
	GT.M	0.89	1.60	14.20	31.80	72.60
	MongoDB	0.88	1.90	16.30	30.00	30.00
Delete	**Oracle**	0.034	5.303	28.074	41.332	267.98
	GT.M	0.60	0.90	0.90	0.90	0.90
	MongoDB	0.98	1.20	8.30	17.80	21.70
Update	**Oracle**	0.19	0.18	0.36	0.46	0.57
	GT.M	0.05	0.05	0.05	0.05	0.05
	MongoDB	0.04	0.05	0.06	0.06	0.30
Sets of queries (average response time)	**Oracle**	0.4	0.095	1.405	2.43	3.584
	GT.M	0.20	0.562	1.225	1.55	1.875
	MongoDB	0.004	0.402	0.68	0.75	1.05

Measurements were performed on computer with the following parameters:

- Intel(R) Core(TM) i3 CPU, 2.53GHz processor clock, 8.00GB RAM hard drive,

- Oracle XE 10g,
- MongoDB 2.2,
- GT.M V6.0-000.

Measurement results of time responses for different queries (for the five samples of data) are presented in Table 4.

In the Table 5 is presented summary of time behavior. Evaluation is determined based on measurement results. Calculated final values were determined by counting the number of occurrences of red (low), black (medium), and green (high) elements in the Table 4.

Table 5

Measurement results of queries execution time for the test sample data

	Data model	Low	Medium	High	Final Evaluation
Evaluation	**Oracle**	**18**	0	2	Low
	GT.M	1	9	**10**	Medium
	MongoDB	0	**11**	9	Medium

Based on a survey of nine users with different roles in DBMS systems (developer, the database administrator, and analyst) the individual systems have been assessed in the context of a set of characteristics of the quality model. Characteristic of data model in context of quality characteristics of the database systems [4], [9], [11] are presented in Table 6.

Table 6

Instance of Quality Model – quality characteristics of data systems (data models)

Subcharacteristic	**Product**	**Description**
Reliability		
Maturity	GT.M	Has a long pedigree and outstanding track record, supporting large complex databases in demanding, real-world business environments for over 40 years.
	MongoDB	Is fairly new (on market since 2009). This system has not been challenged with many critical errors to its performance but there are few "open tickets" existing since 2010 and are still not resolved.
	Oracle	Proved itself by service to big robust companies.
Fault tolerance	GT.M	Replication mechanism is designed in such a way that a network failure between instances will not stop an application from being available-which is a limitation of techniques such as high availability clustering. There are mechanisms in place for edge cases like processing "on the flight" transactions and common cases like handling backlog of updates after recovery from a network failure.
	MongoDB	Data model has low fault tolerance [10].
	Oracle	Stability and mechanisms to support fault tolerance.
Recoverability	GT.M	Updates are written to journal files before being written to database files, and in the event of a system crash, database files can be recovered from journal files.

Table 6

Instance of Quality Model – quality characteristics of data systems (data models)

Subcharacteristic	**Product**	**Description**
	MongoDB	One strategy for recoverability is to have a backup. True backup point-in-time is only possible when all write activity from the application is stopped.
	Oracle	Database can be reconstructed easily from data files, control files and archive log files. The backup mechanisms that work at the physical level protect against damage at the file level, such as the accidental deletion of a data file or the failure of a disk drive.
Usability		
Understandability	GT.M	Complete and accessible documentation, support online.
	MongoDB	Complete documentation, online practical tutorial.
	Oracle	Good documentation, many tutorials, big online community, many tools helping to imagine and get better idea of how it works.
Learnability	GT.M	Once you change the way of thinking from relational database it is quite simple to operate in database.
	MongoDB	Once you change way of thinking from relational database it is quite simple to operate in database.
	Oracle	Depending on level of knowledge of relational data model. Is not a difficult to learn, learning to administer large Oracle databases can take years or longer.
Performance Efficiency		
Time Behavior	GT.M	It is capable of handling massive concurrent transactions.
	MongoDB	Great time with usage of many users, big volumes of data.
	Oracle	Depends on the complexity of queries (number joins/length of path/level of normalization).
Maintainability		
Reusability	GT.M	Components/parts of database files can be reused.
	MongoDB	Depends on quality of application.
	Oracle	Highly reusable system and its components (pre-define view, procedures).
Analyzability	GT.M	Time consuming.
	MongoDB	Difficult and depends on application code.
	Oracle	Provides methods and services helping to analyze data model.
Stability	GT.M	Stable.
	MongoDB	Not Stable.
	Oracle	Stable.
Testability	GT.M	Language allows generating data. Test cases can be invoked by scripts.
	MongoDB	Difficult. Existence of support tools like EmbedMongo for unit testing.
	Oracle	Tools and mechanisms to generate testing data, test case examples.

Table 6

Instance of Quality Model – quality characteristics of data systems (data models)

Subcharacteristic	Product	Description
Operability		
Learnability	GT.M	Most complicated to learn. Archaic operating language– Mumps.
	MongoDB	Easy, intuitive use of database through JavaScript. No constraints.
	Oracle	Quite difficult at initial state of usage. Easy to learn knowing SQL language. Supported by community of experts.
Ease of use	GT.M	Debugging, relational-object interface, easy access to routines allowing learn how to correctly of creating scripts and move in database.
	MongoDB	Supports modern development methodologies (Agile). Developers can easily develop software product using iteration and incremental methods.
	Oracle	Generation of templates, auto-hints, auto-completion, debugging.
Security		
Integrity	GT.M	Most database files have a UNIX file structure externally and a GT.M Database Structure (GDS) internally. Management of the GDS files by the GT.M run-time system assures high level integrity.
	MongoDB	There is low level access control in MongoDB. There are basically 3 types of users: admin, normal (read/write), and read only.
	Oracle	Provides high level of integrity. Possible configuration in file.

Evaluation of the considered systems for selected characteristics of quality model is presented in Table 7. The values in the table were determined based on the following principles:

- High – for at least 80% positive responses (votes),
- Medium – for positive responses in the range of 20% – 79%,
- Low – less than 20% of positive responses (votes).

In order to make a comparative assessment of the analyzed database systems the following evaluation function has been defined:

$$eF = k_1 * NoOfHigh + k_2 * NoOfMedium + k_3 * NoOfLow, \quad (1)$$

where k_1, k_2, k_3 – coefficients of quality range.

In the example the values of coefficients are set respectively for:

$k_1 = 0.6$, $k_2 = 0.3$, and $k_3 = 0.1$.

Table 7

Results of measurement of chosen characteristic in data models based on Table 7

Characteristic	**Sub characteristic**	**Supporting questions[13]**	**Oracle**	**GT.M**	**MongoDB**
Reliability	*Maturity*	The probability of executing faults in the software	High	High	Low
	Fault tolerance	Is the software capable of handling errors?	High	Medium	Medium
	Recoverability	Can the software resume working & restore lost data after failure	Medium	High	High
Usability	*Understandability*	Does the user comprehend how to use the system easily?	High	Medium	Low
	Learnability	Can the user learn to use the system easily?	Medium	Low	Medium
	Attractiveness (trust)	Does the client trust the product?	High	High	Low
Performance Efficiency	*Time Behavior*	Measurement of performance for data models	Low	Medium	Medium
Maintainability	*Stability*	Can the software continue functioning if changes are made?	High	High	Low
	Testability	Can the software be tested easily?	High	Medium	Low
	Reusability	The degree to which an asset can be used in more than one software system, or in building other assets	High	Medium	High
	Analysability	The ease with which the impact of an intended change on the rest of the software can be assessed. The software product can be diagnosed for deficiencies or causes of failures.	High	Low	Low
Operability	*Learnability*	The degree to which the product enables users to learn its application	High	Medium	High
	Ease of use	The degree to which users find the product easy to operate and control	Medium	Medium	Medium
Security	*Integrity*	The degree to which a system or component prevents unauthorized access to, or modification of, computer programs or data	High	High	Low

According to definition (1) has been calculated evaluation function values for the analyzed database systems:

$$eF_{Oracle} = 0.6 * 10 + 0.3 * 3 + 0.1 * 0 = 6.9 \quad (2)$$

$$eF_{GT.M} = 0.6 * 5 + 0.3 * 6 + 0.1 * 2 = 5.0 \quad (3)$$

$$eF_{MongoDB} = 0.6 * 1 + 0.3 * 4 + 0.1 * 7 = 2.5 \quad (4)$$

Results of the evaluation functions values (2), (3), (4) can be summarized as follows:

- Oracle received the highest score. This is due to its mature state and great support of Oracle Company,
- GT.M representing hierarchical data model received positive review. Its middle position is due to niche product nature,
- MongoDB has the worst score in the context of the quality model. Its immature state, problems with maintenance, lack of tools supporting data analysis/optimization adds up to overall score.

6. Conclusions

The problem of assessing various data models using quality models is, by virtue of editorial constraints, presented in an incomplete way and this material should be regarded as a presentation of research conducted by the authors.

The presented quality models make it possible to define a perspective of quality assessment of different database systems by choosing only the needed quality characteristics. The method of using quality models is illustrated on the example of a comparative assessment of three data models and database management systems that represent the "three epochs" in the area of databases. The hierarchical model, widely recognized as forgotten, is still used in dedicated problem domains and in confrontation with the relational data model has been evaluated positively. The trend towards non-relational databases is a result of changing requirements dictated by the collection and processing of large volumes of data. These kinds of data models are represented by a group of NoSQL databases. In our experiment MongoDB system was chosen to contrast with the relational model because of its high popularity. The evaluation indicates that this is an interesting, yet imperfect solution, due to the following flaws. **Non-counting B-Trees** – MongoDB uses non-counting B-trees as the underlying data structure to index data. **Uncompressed field names** – with every new record, database stores names. **Global write lock**_– a process-wide write lock. A write on collection X blocks a write on collection Y, despite MongoDB having no concept of transactions or join semantics. **Safe off by default** – means that we get a product with all protection systems off. **Offline table com-**

paction – The on-disk data size with MongoDB grows unbounded until you compact the database plus it has to be done while offline or on a secondary/slave server. **Second level of servers does not keep hot data in RAM** – The primary level does not relay queries to secondary servers, preventing secondary levels from maintaining hot data in memory [2], [8].

Running queries on all three data models shows that MongoDB is the fastest one. But there is a file storage problem for MongoDB. MongoDB uses "mmap" (memory mapping). During shutdown of the system a problem of data loss occurs. The size of data files on disk is also bigger than in case of Oracle database or GT.M database (which is the leader in smallest file size).

It is clear that relational data model is starting to have problems with the demands of users of most popular web services. Developers are forced to look into new solutions. In fact, NoSQL databases are fulfilling the requirements of speed and scalability but other important issues hang against this type of data model. Open-source products as MongoDB are created by a small group of people. Because of that, there is no proper documentation or specific information about the product, which is very important for developers to understand and properly implement a given model.

MongoDB and GT.M are not popular enough. Educational materials are not common accessible for these products. Popular software products are result among others of intensive promotion and good marketing. Important role place also software support (maintenance). GT.M is strongly supported by FIS (Fidelity National Information Services), where 90% of the products are used in banking and hospital areas.

GT.M offers a full suite of system administration capabilities, including functions such as online ("hot") backup. In fact, the GT.M hot backup directly creates a transaction-consistent snapshot of the database as of the start of the backup, without the need to "rollback" the backup from journal files [4]. With traditional relational databases, for example, there is a need to roll the backup back to the desired state using the journal files. GT.M has plug-in architecture for database encryption in order to protect data at rest.

According to the authors proposed model to evaluate the quality of the database system can be used in a process of choosing appropriate data model and database management system in the context of software product development.

BIBLIOGRAPHY

1. Greenwald R., Stackowiak R., Stern J.: Oracle Essentials: Oracle Database 11g. O'Reilly Media, Inc., 2008.
2. Forum Hacker News, http://news.ycombinator.com/item?id=3202081.

3. Fowler M., Sadalage P. J.: NoSQL Distilled A Brief Guide to the Emerging World of Polyglot Persistence. Addison–Wesley, 2012.
4. GT.M Documentation, http://www.fisglobal.com/products-technologyplatforms-gtm-userdocumentation.
5. Integration with MongoDB, http://java.dzone.com/articles/integration-testing-mongodb.
6. Introduction to NoSQL databases, http://www.slideshare.net/dstainer/introduction-to-nosql-databases.
7. ISO/IEC 25010:2011, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733.
8. Kiip Blog Engineering, http://blog.engineering.kiip.me/post/20988881092/a-year-with-mongodb.
9. MongoDB Documentation, http://docs.mongodb.org/manual/.
10. MongoDB-Fault-Tolerance-Broken, http://www.infoq.com/news/2013/02/.
11. Oracle Documentation, http://www.oracle.com/pls/db102/homepage.
12. SQL vs. NoSQL: Which Is Better? http://slashdot.org/topic/bi/sql-vs-nosql-which-is-better/.
13. IOS Standards, http://hufee.meraka.org.za/Hufeesite/staff/the-hufee-group/paula-kotze-1/publications/SACLA%202010%20paper%20Final.pdf.

Wpłynęło do Redakcji 16 stycznia 2013 r.

Omówienie

W artykule przedstawiono metodę umożliwiającą porównanie pod względem jakościowym różnych systemów DBMS i implementowanych przez nie modeli danych. Zaproponowany w pracy model jakości jest podzbiorem zbioru charakterystyk modeli jakości zdefiniowanych w normie ISO/IEC 25010: 2011 (Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models). Całość została zilustrowana przykładem zastosowania przedstawionej metody. Z racji rosnącej popularności systemów NoSQL, ocenie jakościowej zostały poddane modele danych z podziałem na modele relacyjne i inne, reprezentowane przez trzy DBMS – Oracle (relacyjna baza danych), GT.M (hierarchiczna baza danych) oraz MongoDB (baza NoSQL). Uzyskane rezultaty są interesującym przyczynkiem do kontynuacji prac zmierzających do określenia szczegółowych kryteriów oraz zaleceń dotyczących wyboru modeli baz danych w zależności od przeznaczenia i wymagań jakościowych potencjalnych użytkowników.

Addresses

Lech TUZINKIEWICZ: Politechnika Wrocławska, Instytut Informatyki,
ul. Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Polska, lech.tuzinkiewicz@pwr.wroc.pl.
Paweł WIATER: CUBE.ITG, Dział rozwoju bankowości, ul. Wołowska 6, 51-116 Wrocław, Polska, Pawel.Wiater@cubeitg.pl.