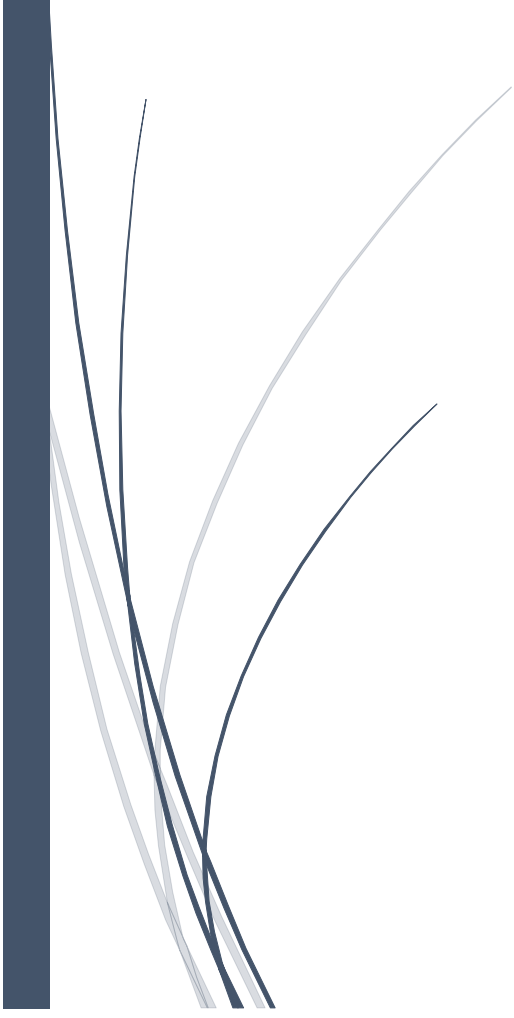
10/1/2020

Object-Relational Database Management Systems



WORKMAN, Antonio & ALKINS, Joshua
190907 & 190908

1. Table of Contents

2. Introduction

Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model invented by E. F. Codd in 1970. It is currently the dominant database technology in use today. It has provided the basis for research on the theory of data/relationship/constraint, numerous database design methodologies, the standard database access language called structured query language (SQL) and almost all modern commercial database management systems. Whereas, an Object-Oriented Database Management System (OODBMS) is a database management system which supports the modelling and creation of data as objects. It allows for the object-oriented programmers to enjoy the consistency of using one programming environment because the database is integrated with the programming language and uses the same representational model. The OODBMS continues to find new application areas, such as the World Wide Web. Until recently, the choice of DBMS seemed to be between the relational DBMS and the object-oriented DBMS. It was agreed upon that the traditional relational DBMS is not suited to advanced applications. As such, the best way to remedy this situation was to extend the relational model with the type of features the object-oriented programming offers. The Object-Relational Database Management System (ORDBMS) is a database management system that seeks to bridge the gap between the relational DBMS and object-oriented DBMS.

Michael Stonebraker is a computer scientist who specializes in database research. His research and products are central to many relational database systems. Stonebraker states the world of database systems and uses for these systems are broken into four quadrants as shown in Figure 1. The first quadrant, the lowerleft quadrant represents applications that have simple data and do not require any query capability such as word processing applications. These types of applications do not need a database system. The lower-right quadrant represents applications that have complex data and do not need query capabilities such as computeraided design applications. These applications are best suited using an object-oriented database system (OODBMS). The upper-left quadrant represents applications that have simple data and need query capabilities such as traditional banking applications. Finally, the upper-right quadrant represents applications that have complex data, but also need query capabilities. Stonebraker proposes that applications of this type can benefit most from using ORDMBSs. ORDBMS is similar to a relational database, except that it has an object-oriented database model. This system supports objects, classes and inheritance in database schemas and query language.

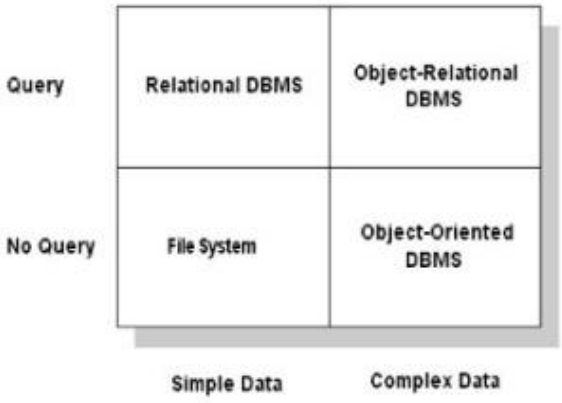


Figure 1: Stonebraker's Depiction of the Four-Quadrants

At this time the three leading database management systems are Oracle, Microsoft and IBM. They have extended their database system to support the SQL:2003 standard which includes object-relational features.

The focus of this paper is the study of the characteristics of object-relational databases, referring to its advantages, and the evaluation of the most popular object-relational database: Oracle.

3. The Basics of Object-Relational Database Composition

The relational database structure is used as the data storage framework for the Object-Relational database. The qualities of these tables are extended to accommodate the storage of various types of objects. The follow are types of tables used:

a) Object table: Only row type objects are stored.
b) Relational table: In the column, different types of data objects can be stored.

In order to allow for an easy exchange of data between object-relational and relational database structures, row type objects have identifiers OI. When When performing data retrieval from the table, data can be obtained as a single object and as a tuple in relational algebra.

A data storage structure for XML and graphic data has been developed by specializing object-relational database technology. By using XMLType, which is an object-relational database type developed for storing XML data, one can:

a) Develop a XML data table.
b) Include XMLType objects in. relational table's columns.

In Oracle the possibilities of relational databases are extended by object-relational technology. XML, relational and object tables are used. Object-relational databases offer a wide range of possibilities to create data association.

Object-relational database technology has mainly developed due to data processing methods. Composed methods allows for the extending of the programming possibilities of database server as well as decreasing the load of computer networks. They also include database languages in SQL queries.

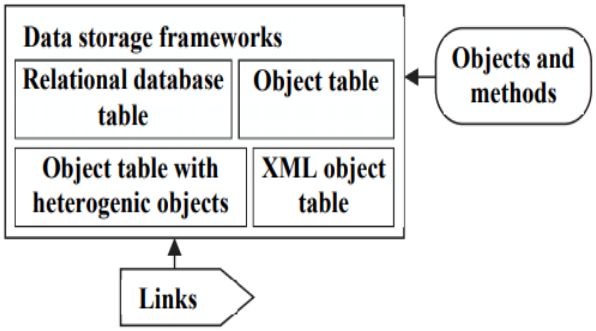


Figure 2: Creation of the Structure of Object-Relational Database

The figure above shows the creation off the structure of object-relational databases.

4. Key Features of the Object-Relational Model

There is no accepted standard for the object-relational data model. However, there is a commonly used model which is based on the elements supported by the recent SQL standards. These features violate many of the rules applied to relational databases.

OORDBMSs continue to support standard relational database functionality, such as backup and recovery, scalable connectivity queries, and more. The oracle database implements the object-type model as an extension of the relational model.

Most SQL RDBMS as it pertains to complex data creation, is based on the preliminary definition of the schema via the user-defined data type. A relational database should have no data structures other than tables. In contrast, the ORDBMS allow the definition of a datatypes, functions and operators to be done by the user. They allow for attributes to support a reference to a row in another table. The Object-Relational database can use the concept of primary and foreign-keys but it is not required to do so. A reference to a row may be used in its stead. This results in the functionality and performance of the to increase because the joins needed to follow data relationships are unnecessary. However, the downside to using rows as references is the integrity of the

relationships cannot be verified by the DBMS as referential integrity requires the use of both primary and foreign keys.

Relational databases are limited to one value within the intersection of a single column and row. In contrast an object-relational database can store multiple. The value types vary, as they can be an array of the same type of data, a row of data, an unordered collection of data of varying types, or an entire object.

Classes are implemented as user-defined data types. Inheritance of all user defined is supported to derive new sub-types or sub-classes which would therefore inherit all attributes and methods from the parent type. Multiple inheritance is not allowed. A UDT has a default accessor and mutator method as well as a default constructor, each of which can be overridden by the programmer. In the relational database UDTs may be used, however, in order for this to occur a custom data type must hold only single value.

UDTs have methods defined within them. Those methods may be overloaded. Polymorphism is also supported. However, relational databases have no concept of storing procedures with data.

Extensibility, or the concept of extensibility to be specific, is a principle innovation of the ORDBMS technology. While using SQL-92, a problem which frequently occurs when developing information systems is that in modelling complex data structures and implementing complex functions is difficult. A solution to this issue is for the DBMS vendor to build the DBMS engine in such a way that it can accept the addition of new, application-specific functionality.

Developers can specialize or extend many of the features of an ORDBMS:

- the data types
- OR-SQL expressions
- the aggregates, and so on.

In fact, it is useful to think of the core ORDBMS as being a kind of software backplane, which is a framework into which specialized software modules are embedded.

5. Why are Object-Relational Database Systems Needed?

The use of object-oriented programming languages has resulted in a set of new issues. According to William Cook, these issues "arise at the boundary between programming languages and databases". Object persistence introduces issues of impedance mismatch between programming languages and databases because of complex relationships and user defined data types. Complex relationships can be defined as many-to-many relationships and inheritance. Likewise, the definition of complex data is nested objects, multi-dimensional arrays, unstructured data (voice, video), data in non-first normal form, and user-defined data types . Complex data and relationships, such as inheritance, nested objects, and user-defined data types are properties of OOPLs. Therefore, the need to store complex relationships and data is in part a result of using OOPLs to develop applications. Before OOPLs, the relational model was adequate for storing application data.

6. Database Stored Procedures and Storage Management

The implementation of business processes by the creation of database procedures is done by almost all RDBMS. This allows for the movement of considerable portions of an information system's total functionality into the DBMS by the developer. In many situations, centralized CPU and memory requirements on a single machine can improve the system's overall throughput and simplify its management.

By implementing application objects within the server, it becomes possible, through the use of Java for example, though not always desirable, to push code implementing one of an application-level object's behaviors into the ORDBMS. The interface in the external program simply

passes the work back into the IDS engine. It is important to remember that without changing the code in anyway or recompiling it, with Java, the same logic can be deployed either within the ORDBMS or within an external program.

Conventionally the main purpose of a DBMS was to organize and centralize data storage. A single machine ran a DBMS program that would take blocks of that machine's disk space, control it and store data in them. With the use of RDBMS products, more sophisticated data structures and more efficient memory caching and data scanning techniques as well as the storage of large data objects were included.

However, only a portion of an organizations data can be held in one location due to data being distributed amongst many systems. This is a consequence of a subset of a variety of varying factors.

7. Object Relational Database Advantages and Uses

Object-Relational Database Management System (ORDBMS) is a database management system (DBMS) similar to a relational database, but with the addition of object-oriented database model functionality. Objects, classes, and inheritance are directly supported in the ORDBMS database schemas and in its query language. These databases are some of the most popular database systems for financial and telecommunications applications.

Object-relational databases gain the advantages of both relational databases and object-oriented databases, in that they can store and interpret arbitrary custom data type while maintaining the benefits of relational databases. In this way they bridged the gap between relational databases and object-oriented practices commonly used in the software industry. Treated as relation databases with support for objects at the cost of some complexity by ORDB companies.

To discuss the benefits of Object Relational Database Management Systems we must first discuss the context in which they arose. Therefore, we must first discuss Relational Database Management Systems and Object-Oriented Database Management Systems.

7.1 Relational Database Management Systems & SQL Database Management Systems

Relational Database Management Systems and SQL Database Management Systems have been the predominant Database Management Systems in the software industry for many years. These Database Management Systems offer a level of simplicity in storing data, with all information being placed into tables with their structure being predefined in the Database Schema. These Database Management Systems also provide comprehensive functionality for querying data. This is achieved thanks to data being stored with foreign keys that point to the data that it is related to, which is where this DBMS gets its name.

This DBMS provides many advantages:

- Simple Model
- Data Accuracy
- Easy to access data
- Data Integrity
- Flexibility
- Normalization
- High Security
- Modifiable

Despite these advantages there are a few problems present in the relational database model, namely that there is a mismatch/conflict between the method of storing data and a common coding practice in the form Object-Oriented Programing. Some systems developed using the Object-Oriented practice require the storage of complex datatypes/objects in a database, such as CAD (Computer Aided Design) software. These datatypes/objects commonly contain values consisting of

standard datatypes, methods, and often other nested objects. This complex data poses a challenge for the Relational Database Model as it is difficult to map this complex data to the rigid table system of the RDBMS. To resolve this Object Database Management Systems and Object-Oriented Database Management Systems were Developed.

7.2 Object Database Management Systems & Object-Oriented Database Management Systems

Object Database Management Systems and Object-Oriented Database Management Systems were developed specifically for storing arbitrary complex datatypes in the form of objects, with values (variables stored in object), including other nested objects, and methods. This database model gained many of the advantages of found in the object-oriented coding practice. These advantages include:

- Real World Modeling
- Reduced Maintenance
- Reliability and Flexibility
- Code Reusability
- Inheritance
- Polymorphism
- Encapsulation
- Complex Data Types
- Extensibility

In addition, this DBMSs often allow some methods to be run server-side in the database when querying data, improving application performance. However, these DBMSs, while extremely useful, are highly specialized and are not necessary for most data being stored. Furthermore, the cost of transitioning from a Relational Database Management System, used by most companies today, to an ODBMS or an OODBMS is often prohibitable expensive. Enter the Object Relational Database Management System.

7.3 Object Relational Database Management Systems

The Object Relational Database Management System attempted to bridge the gap between Relational Database Management Systems and Object-Oriented Database Management Systems. There is no standard method for achieving this goal and so each individual database system's implementation may differ. However, typically this is achieved by adding support for objects onto the relational model.

In this way the Object Relational Database Management System manages to gain the benefits of both the Relation Database Management System and the Object-Oriented Database Management System while circumventing the drawbacks of each, at the cost of some additional complexity.

8. ORDBMSs Vs. NoSQL DBMSs

There has been a new trend in the database field, in the form of a resurgence of the NoSQL database systems. NoSQL databases allow data to be stored and retrieve in a format other than the tabular format found in traditional relational databases. NoSQL databases are characterized by their lack of need for a fixed schema as shown in [1]. This allows these NoSQL databases to be well suited to handling more unstructured data.

Carlo Strozzi coined the term NoSQL in 1998 for the database he developed. It is generally accepted that NoSQL stands for "Not Only SQL", implying that it is to be used along with traditional SQL databases and not as a replacement or rejection of SQL database.

Nance et al. "NoSQL Vs RDBMS – Why there is room for both" [2], explored this point further and concludes that even with the rise in NoSQL, RDBMSs are not going anywhere any time soon as each database system is better suited to different situations based on their advantages and disadvantages.

NoSQL databases tend to be highly scalable, reliable and have a very simple data model and bare query language. NoSQL databases

typically do not adhere to ACID (atomicity, consistency, isolation, durability) restraints, which allows for exceptionally fast performance and a tendency to scale very well on commodity hardware. However due to NoSQL databases' lack of adherence to ACID restraints, issues can arise when applications require great precision [3]. NoSQL tend to have almost no support for security, Okaman et al. [4], with a vulnerability to SQL injection and DOS (denial of service) attacks. However, for these reasons NoSQL databases are better suited for modern web applications that support millions of concurrent users.

Many relational databases can now be considered multi-model as they have introduced from other database systems, including NoSQL database. This is similar the adoption of Object Relational database functionality by RDBMSs, as these relational databases can now provide a larger more comprehensive set of tools to their users. Additional users and organizations may have an easier time transitioning to NoSQL features if they are provided by a database service they are already using.

There are essentially four basic kinds of NoSQL databases. [5]

- Document databases
- Graph Stores
- Key-Value Store
- Columnar Store

DB-Engines [6], also list time-series and search-engines as two other notable types of database systems.

Of these Oracle has introduced support for key-value store, document, and columnar models in Oracle NoSQL, as well as spatial and graph features [7]. Oracle stating on their website, "Oracle database products offer customers cost-optimized and high-performance versions of Oracle Database, the world's leading converged, multi-model database management system, as well as in-memory, NoSQL and MySQL databases."

8.1 Document-Store

Document databases, as the name would suggest, store data in documents. These documents are often similar to JSON objects, containing pairs of fields and values. Document store databases are the most popular of NoSQL databases.[6] These databases have a variety of uses and is well suited for a general-purpose database according to MongoDB [8]. The main advantages of document store databases are their blindingly fast speeds and easily horizontal scalability, making them well suited to storing incredibly large amounts of data.

8.2 Graph Databases

These graph databases are used to store information about networks of data. These databases store data as nodes and edges, with nodes storing information on objects while edges store information on the relationships between objects. These databases excel in traversing relationships between objects and looking for patterns in them. Graph Databases are also becoming more popular as time goes on, one major contributor to this fact is they are particularly useful for social networks. Social networks store information on users and their social relationships (their social network), and are some of the most popular services on the internet.

8.3 Key-Value Store

Key-value stores are the simplest form of a NoSQL database. Key-Value stores use the associative array (also known as a map or dictionary) as their fundamental data model [5]. In this array the key is a unique identifier that can only appear once in the array. The value can be any kind of data that needs to be stored, from fundamental data types to Strings to JSON object etc.

8.4 Columnar

Column-Oriented databases store data in cells grouped in columns rather than in rows. These databases are optimized for

queries over large datasets. The data does exist in a kind of table similar to relational databases but in column-oriented databases the table is much more loosely defined. Columns can exist in column families that groups columns that store data that is often used together. These columns can also change from row to row allowing programmers to not be restricted by a predefined schema.

8.5 Performance

In a study in 2012 by Alexandru Boicea et al. [9], the performance of one of the most popular NoSQL databases, MongoDB, was compared to Oracle's relational database. They found that oracle's relational database would take exponentially more time to add, update and delete records, while mongoDB's time for adding data only increased form 1msec to 3msec for adding records and stayed consistently at 1msec for updating and deleting. See figures 1-3 for display of difference in times for common database operations found in the 2012 study.

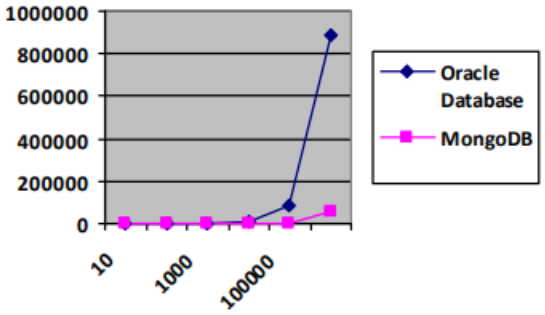


Figure 3: Inserting times(msec) [9].

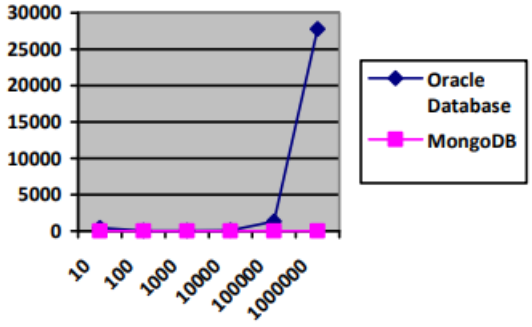


Figure 4: Updating times(msec) [9].

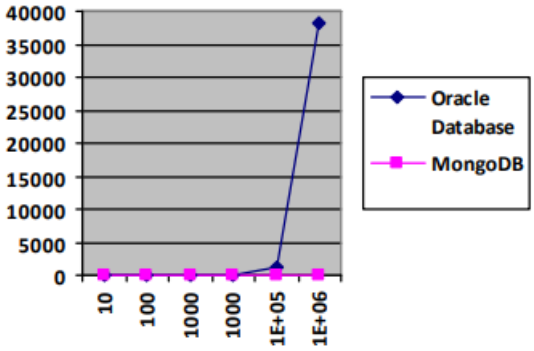


Figure 5: Deleting times(msec) [9].

Another paper published in 2014, Abramov a et al. [11] compared the performance of several NoSQL databases performing various operations using Yahoo! Cloud Serving Benchmark. In the study were databases several representing each of the four major types of NoSQL databases. The study included both MongoDB representing document-store and Oracle NoSQL using a Key-Value store model.
This study found that Oracle NoSQL, while it performed the worse out of all the databases being compared in loading, was comparable to MongoDB in all workloads tested and even had a significantly less total execution time for all operations. However, it should be noted that of key-value store databases Oracle NoSQL consistently performed the worse in terms of execution time. This could possibly be attributed to extra overhead as Oracle NoSQL offers a hybrid ACID/BASE model [12], though they claim this comes at no cost to performance.
These studies illustrate the advantages of NoSQL databases, as well as Oracle NoSQL's ability to provide NoSQL services in terms of performance.

9. Popularity of Object-Relational Databases and Oracle DB

The Relational Database Management System has been exceedingly popular in the software industry since its inception due to its simple and effective method for storing and managing data, as previously mentioned. It quickly became the industry

standard where it has remained to this day. According to db-engines [6], 6 of the 10 most popular database systems primarily use relational model, 4 of which are the highest ranked database systems.

The rankings presented by db-engines.com are based on a weighted scoring model using a variety of factors to generate its rankings. These factors include mentions on websites, google search trends, job offers, and mentions in professional network profiles.

Many systems previously relying on a RDBMS require or would benefit from the features offered by Object Database Management Systems, but in most cases transitioning to an ODBMS from a RDBMS can be prohibitively expensive. Object-Relational Database Management Systems offer a nice middle ground in this situation, allowing companies to transition relatively easily to this kind of DBMS that retain the structure and functionality of RDBMSs but also offer the additional Object-Oriented Database functionality that many companies/applications require.

In response to this, many of the most prominent RDBMS's now offer some form of support for objects to meet this demand. This in a way now makes the ORDBMS currently the most popular database management system, though this would only be due to these features being added to already popular Database Management Systems as the majority of systems that implement these DBMSs do not necessarily utilize these features.

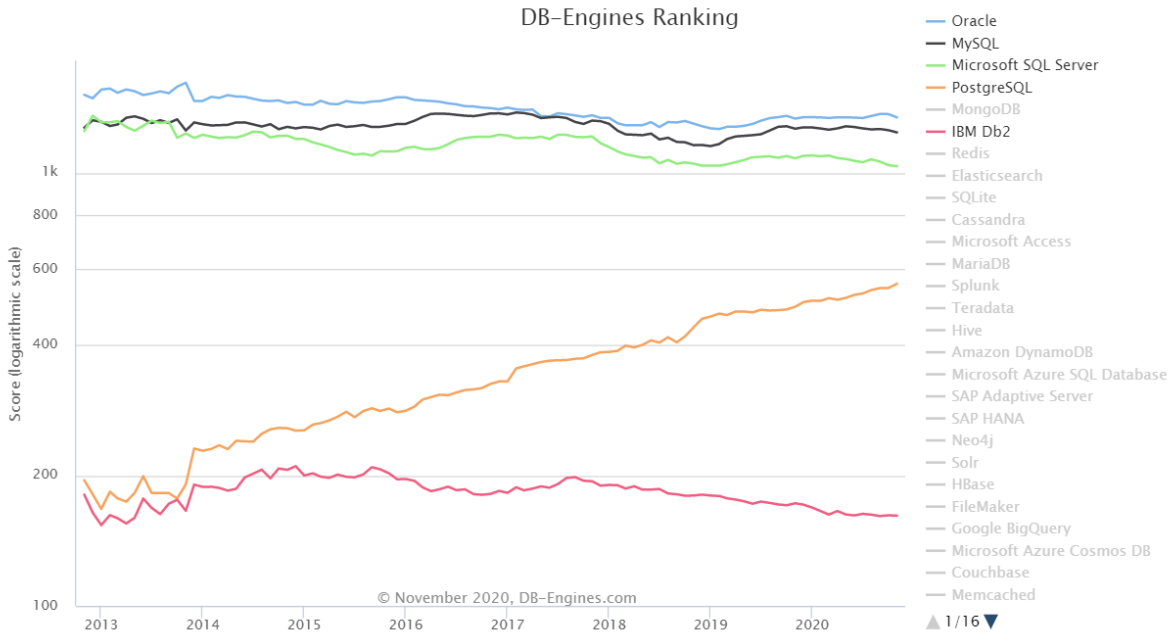


Figure 6: Popularity of Relational Database Systems[6].

Database systems like Oracle, Microsoft SQL Server, PostgreSQL, and IBM Db2 all offer support for objects. In fact, of the 5 highest ranked RDBMSs according to db-engine.com, MySQL is the only one to not offer any kind of direct support for objects. Similarly of those same 5 database systems all are listed as Multi-Model [6], except for MySQL again. These DBMSs have incorporated multiple models into their database systems including NoSQL models, as demonstrated by Oracle.

Of the RDBMSs Oracle has always been an incredibly popular DBMS since its creation, if not the most popular. According to db-engines.com, Oracle has consistently been the most popular RDBMS and DBMS since 2013, when the site began tracking the rankings of DB systems. See figure 6.

10. Conclusion

Relational Multi-Model database systems, stemming from Object-Relational database systems, will continue to be a main stay in the IT industry for the foreseeable future. The benefits of the relational model continue to be the best solution in many situations, and by incorporating features from multiple models including Object-Oriented and NoSQL these multi-model database systems can cover the needs of most applications, as well as provide organizations already using their database system a way to easily transition to new database models. This implies that NoSQL will be implemented alongside the relational model to solve modern issues that relational database management systems are not well suited for.

For these reasons relational multi-model database management systems will seemingly continue to be the best default option for many applications' database needs.

11. Bibliography

[1] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, Dawn Wilkins, "A Comparison of a Graph Database and a Relational Database," Proceedings of the 48th Annual Southeast Regional Conference, 2010, Oxford, MS, USA, April 15-17, 2010

[2] Nance, Cory; Losser, Travis; Iype, Reenu; and Harmon, Gary, "NOSQL VS RDBMS - WHY THERE IS ROOM FOR BOTH" (2013). SAIS 2013Proceedings. 27. http://aisel.aisnet.org/sais2013/27

[3] N. Leavitt, "Will NoSQL Databases Live Up to Their Promise?," in *Computer*, vol. 43, no. 2, pp. 12-14, Feb. 2010, doi: 10.1109/MC.2010.58.

[4] L. Okman, N. Gal-Oz, Y. Gonen, E. Gudes and J. Abramov, "Security Issues in NoSQL Databases," 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications, Changsha, 2011, pp. 541-547, doi: 10.1109/TrustCom.2011.70.

[5] Raj R. Parmar, Sudipta Roy, "MongoDB as an Efficient Graph Database: An Application of Document Oriented NOSQL Database" in "Data Intensive Computing Application for Big Data", 29 (2018) 331 – 358.

[6] DB-Engines, https://db-engines.com/en/ranking, accessed October 4th, 2020.

[7] Oracle, https://www.oracle.com/database/technologies/related/nosql.html, https://www.oracle.com/database/nosql-cloud.html, accessed October 10th, 2020

[8] MongoDB, https://www.mongodb.com/, accessed October 10th, 2020.

[9] A. Boicea, F. Radulescu and L. I. Agapin, "MongoDB vs Oracle -- Database Comparison," *2012 Third International Conference on Emerging Intelligent Data and Web Technologies*, Bucharest, 2012, pp. 330-335, doi: 10.1109/EIDWT.2012.32.

[10] A. Gupta, S. Tyagi, N. Panwar, S. Sachdeva and U. Saxena, "NoSQL databases: Critical analysis and comparison," *2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)*, Gurgaon, 2017, pp. 293-299, doi: 10.1109/IC3TSN.2017.8284494.

[11] V. Abramova, J. Bernardino, P. Furtado, "EXPERIMENTAL EVALUATION OF NOSQL DATABASES" in International Journal of Database Management Systems (IJDMS) Vol.6, No.3, June 2014.

[12] Oracle NoSQL Database (https://www.oracle.com/technetwork/database/nosqldb/learnmore/nosql-database-498041.pdf), June 2018.