Comparing the Performance of Object and Object Relational Database Systems on Objects of Varying Complexity

Reza Kalantari♣, Christopher H. Bryant

School of Computing, Science and Engineering, Newton Building,
University of Salford, Salford, Greater Manchester, M5 4WT, UK
me@reza-kalantari.com, C.H.Bryant@salford.ac.uk

Abstract. This is the first published work to compare the performance of object and object relational database systems based on the object's complexity. The findings of this research show that the performance of object and object relational database systems are related to the complexity of the object in use. Object relational databases have better performance compared to object databases for fundamental database operations, with the exception of insert operations, on objects with low and medium complexity. For objects with high complexity, the object relational databases have better performance for update and delete operations.

1 Introduction

When object oriented programming languages such as Java, C++ and Smalltalk became popular in the 1980s, application developers found a mismatch between their applications' needs and Relational Database Management Systems (RDBMSs). The mismatch led to the invention of Object Database Management Systems (ODBMSs). In fact, ODBMSs are an extension of object oriented programming into the world of databases and they benefit from using object programming languages. Despite the fact that ODBMSs are very suitable for some specific applications, developers encountered major problems when using them in place of RDBMSs such as a lack of a universal standard, complex query optimization and poor support for large scale business information systems. These drawbacks made developers generate another type of database system, namely Object Relational Database Management Systems (ORDBMSs). The main objective of ORDBMSs was to achieve the benefits of both the relational and the object models and, in fact, ORDBMSs combine the features of RDBMSs with the best ideas of ODBMSs. ORDBMSs store data in tables but the main difference between ORDBMSs and RDBMSs is that ORDBMSs have object-oriented features. The standard programming language for ORDBMSs is OR-SQL which is also known as SQL3. Many well known database vendors such as IBM and

♣ To whom correspondence should be addressed.

Oracle have released the object relational version of their database management systems [2].

The success or failure of an application directly depends on the performance of the database system in use. Therefore performance is a vital factor for the selection of database systems in real-time applications. A variety of different ideas about the performance of ODBMSs and ORDBMSs have been published. While [8] states ODBMSs are known to be rich in functionality but poor in performance, [7] believe that the performance of object databases is far better than hybrid ORDBMSs. The contrast between these findings motivated the research described in this paper which determines which one of object and object relational database management systems is better in terms of performance for fundamental database operations such as Insert, Update, Lookup and Delete. This paper presents the results of a fair comparison of the performance of ODBMSs and ORDBMSs by means of an object oriented application and it takes the object's complexity into account.

Section 2 describes related work. In Section 3, we describe the performance criteria for this work and justify why benchmarks are not used. Section 4 presents the environment of the case study, the results of our evaluations and an analysis of the results. In Section 5, we briefly summarize the main contributions of this paper and identify the need for further research in this area.

2 Related Work

Over the last two decades, when ODBMSs were still rather new, there were a variety of studies to assess the performance of this kind of DBMS. For example, [9] compared the performance of various commercial ODBMSs. More recently, [11] compared the performance of ODBMSs and Object Relational Mapping (ORM) tools.

In the study by Van zyl et al. [11], Db4o represents the ODBMS and Hibernate represents the ORM tool. Both of these are popular open source products. Hibernate is an ORM tool that stores and retrieves in-memory objects to and from a RDBMS. Hibernate can be used with any RDBMS but in their research it was used with Postgres for persisting objects. The OO7 benchmark was used to compare the speed of execution of a suite of typical persistent-related operations in both candidates. For good documentation of OO7 benchmark, see [4]. Van zyl et al. [11] decided to use Java objects for their research study because they believed that "most of the large persistence mechanism providers provide persistence for Java objects". As a result of this decision, they had to re-implement OO7 in Java because the OO7 benchmark had been developed in C++ for Versant. Db4o can be run as an embedded DBMS, as a local server in the same virtual machine or as a remote server; for their research Db4o was run as an embedded DBMS. Both of Db4o and Hibernate were to persist the in–memory Java objects generated by the OO7 benchmark. Van zyl et al. [11] concluded that Db4o's overall performance is better than that of Hibernate. They propose that the overhead of object-relational translation causes ORM-based implementations to be consistently slower than staying in object form with an ODBMS. The study by Van zyl et al. [11] is similar to the one described in this paper in the sense that both compare the performance of Db4o with a hybrid database solution, on an artificial

dataset. However our study is different to the one by Van zyl et al. [11] because they used OO7 benchmark for performance evaluation while we use an object oriented application.

Hohenstein et al. [9] performed an application-specific comparison of the three best known commercial ODBMSs. The goal of their evaluation was to create a realistic test for ODBMSs, allowing for a fair and precise comparison of performance. The researchers took as their starting point an existing warehouse application running on a relational DBMS. The application was a large software system that maintains automatic warehouses. For simplicity and to reduce the effort, they restricted the application to only one procedure, namely storing materials. The researchers also compared the ODBMSs with the original, real-world relational system; however they believed that this comparison is vague because the times for the RDBMS were measured while concurrent processes may influence locking and elapsed times. In the study by Hohenstein et al. [9] the ODBMSs remain anonymous and they are introduced as ODBMS1, 2 and 3. Each ODBMS has been tuned heavily according to its specific architecture. Their experiments measure the times for the whole application's test rather than for simple database operations. The test consists of placing 860 containers with articles in the warehouse and specific functions such as queries.

Hohenstein et al. [9] concluded that traversals of relationships are much faster in the page server ODBMSs than in related SQL queries. Since ODBMSs do update operations in the primary memory and update in the server and disk is postponed to the commit, this results in slower update operations by ODBMSs compared to RDBMS. The complex search is also very fast in ODBMSs. The study by Hohenstein et al. [9] is similar to our work in that it evaluates the performance of DBMSs by means of a concrete object-oriented application. However they use a real dataset for their experiments while we use an artificial dataset. We justify our use of an artificial dataset in Section 4.2.

3 Performance Measurement

In this work we aim to evaluate and compare the efficiency of Db4o and Informix DBMSs for performing four fundamental database operations: insert, update, look up and delete. The efficiency of these operations in any database system is a vital factor of performance. For measuring performance, we use Response time. Response time measures the performance of an individual transaction or query. Response time is typically treated as the elapsed time from the moment that a query's execution starts until the time that the execution finishes successfully.

One approach used by research studies aiming to evaluate the performance of database systems is benchmarking. A lot of standard benchmarks have been published in the literature. Benchmarks are general applications that reduce the effort required to implement and perform performance tests. For example, the OO1 benchmark [6] models a graph of interconnected nodes in which each node is related with three other nodes [5]. Other benchmarks such as HyperModel [1] and OO7 [3] model more complex schemas; they take into account inheritance hierarchies and various forms of

relationships between nodes. Nevertheless, these benchmarks are compact, general and do not meet the requirements of all performance tests. In reality, applications interface database systems and use them to store and retrieve data. Also applications perform access and make additional demands of DBMSs that standard benchmarks do not cover at all; therefore performance of database systems should be evaluated by means of applications. In addition, a benchmark that meets the requirements of our research could not be found.

4 Case Study

4.1 Database Products for this Case Study

Db4o[1] is an open source pure ODBMS that enables Java and .NET developers to store and retrieve any application object; eliminating the need to predefine or maintain a separate, rigid data model. Db4o's programming can be integrated in the application code; therefore database access is largely transparent, which is one of the main objectives of ODBMSs [10]. *Informix Dynamic Server*[2] is a well-known commercial ORDBMS that completely supports the object relational specifications. Informix provides an application programming interface for C, C++, Java and .NET.

4.2 Dataset

Datasets have an important role in experimental studies which evaluate the performance of databases. Data is the core of a database system and it affects the database's performance. This means that a performance test on a specific database system with two different datasets may result in different conclusions. One of the common approaches in experimental studies is to use a dataset which is already in the public domain. For this work an online dataset that fits in the designed database could not be found. The other common approach is to create an artificial dataset by randomly generating data of the required form. This is the approach we use. Objects are populated with random data when they are instantiated.

4.3 Object Oriented Database Schema

This section describes the design of the object oriented database schema used in the experiments. Since the aim of the research is to compare the performance of two database systems which are both object oriented, three objects of varying complexity were designed. Project, Staff and Department objects represent objects with low, medium and high complexity respectively. As Fig. 1 shows, Project, Staff and Department objects consist of different attributes. For simplicity the objects have no method. Project object is an object with low complexity because its attributes are of

[1] Db4o 6.4 .Net 2.0

[2] Informix Dynamic Server v11.50

basic data types such as String, Integer and Date that have low complexity from database management point of view. The Staff object is more complex. It includes the Project attribute that is an ordered list of the projects that the employee took part in. Each of the elements in this list is a Project object. Also it includes the address attribute which is a user defined data type. Address consists of four attributes of type String which hold the employee's address.

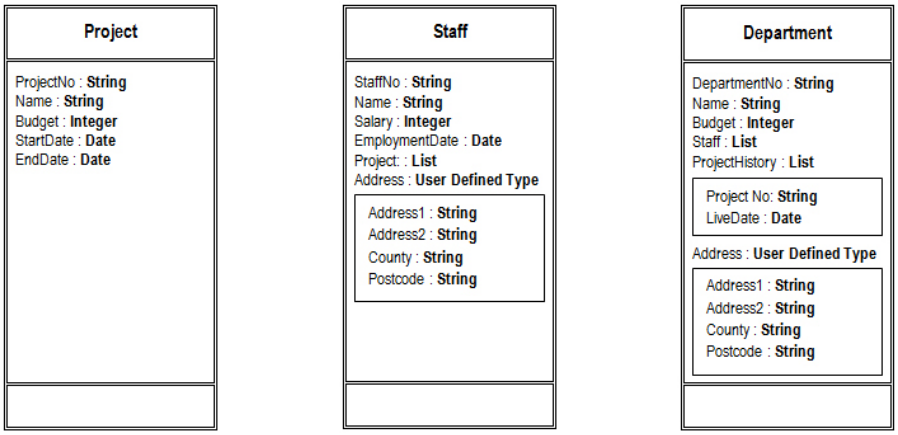

Fig. 1. The schema of Project, Staff and Department objects

The Department object includes the Staff attribute which is an ordered list of Staff who work for the department. Each element in this list is a Staff object. Another attribute of Department object is the ProjectHistory. This attribute is an ordered list of a user defined data type that holds the previous projects and their live date. The Department object also includes the Address attribute which is a user defined data type that holds the Department's address. Therefore, Department object is considered as an object with high complexity.

4.4 Object Oriented Test Application

The Object Oriented Test Application (OOTA) is a .NET object oriented application that has been developed to perform the performance tests against both Db4o and Informix DBMSs. OOTA implements the Project, Staff and Department objects which represent the objects with low, medium and high complexity respectively. OOTA also implements four test functions for each object to perform the performance tests against the database systems for the object. Each test function performs a specific performance test against the DBMSs.

4.5 Methodology

To perform the empirical experiments, the object oriented database schema has been implemented with both Db4o and Informix DBMSs. OOTA was developed to interface both Db4o and Informix DBMSs and performs the performance tests against

them. OOTA performs the performance tests through test functions. OOTA obtains the response time by measuring the time before the functions' call and after the functions' execution. All experiments have been repeated five times and the mean of response times is reported in the results. The standard deviation was less than 3% in every experiment.

In all empirical experiments, the performance of both Db4o and Informix databases has been evaluated for six different quantities of objects. The six different quantities are 1000, 5000, 10000, 20000, 50000 and 100000. These quantities represent a variety of small to large databases. For each experiment, each test function has been called six times against both Db4o and Informix DBMSs for the six different quantities of each object. The experiments allow a fair comparison of Db4o and Informix DBMSs because:-

- The same hardware and operating system was used in all the experiments.
- The same database model (i.e., the same objects) has been implemented with both Db4o and Informix.
- The same performance test application, (i.e., the OOTA) is used to perform the tests against both Db4o and Informix.
- The same performance tests have been performed against both Db4o and Informix.
- The mechanism for creating new objects within OOTA is the same for both Db4o and Informix.
- The object's data that OOTA generates in the object construction process is completely random and the mechanism is the same for both Db4o and Informix.
- The most optimized function's code has been developed within OOTA for both Db4o and Informix DBMSs according to the database vendors' release notes and tutorials.
- The interface creation time for Db4o and the connection time for Informix have been excluded from the response time.

4.6 Object Insertion

The aim of this experiment is to determine whether Informix or Db4o has a better performance for the insert operation. The response times for insert operations in this experiment includes the object's creation time. The results of the Object Insertion experiment for inserting different quantities of objects with low, medium and high complexity into both of Db4o and Informix DBMSs are shown in Fig. 2.

As Fig. 2 shows, for objects with low complexity, although both database systems' response times are very close until 5,000 objects, Db4o performed the insert operations in less time compared to Informix throughout the experiment. The more objects in the insert operation, the bigger the difference between their response times. Another point is that Informix has the same performance during the experiment but Db4o's performance is slightly variable and is best when inserting 5000 to 10000 objects.

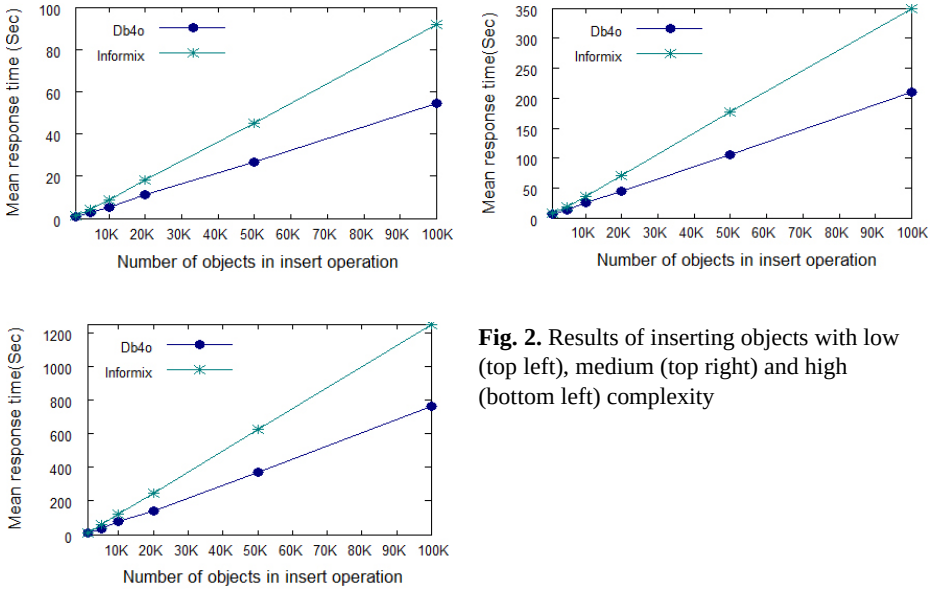

Fig. 2. Results of inserting objects with low (top left), medium (top right) and high (bottom left) complexity

According to results for object with medium complexity, Db4o's performance is better throughout the experiment. The performance of Informix is nearly the same as Db4o for inserting less than 5000 objects but after this point the Informix's performance decreases. Db4o has a constant performance for inserting more than 10000 objects. Surprisingly, Informix's performance is not the same during the experiment for all the number of objects; it performed better for quantities between 1,000 and 5,000.

Db4o has inserted the high complexity object in less time than Informix for every quantity. Similar to the medium complexity experiment, Db4o's performance is constant for inserting more than 10,000 objects. The Informix's performance is worse than that of Db4o and it is constant throughout the experiment.

4.7 Object Modification

The aim of this experiment is to determine whether Informix or Db4o has a better performance for the update operation. In each update operation one object is modified. The results of the Object Modification experiment for updating objects with low, medium and high complexity with both Db4o and Informix DBMSs while they hold different quantities of these objects are shown in Fig. 3.

According to Fig. 3, the performance of Informix is far better than that of Db4o for updating objects with low complexity. With the exception of 5000 to 10000 objects, the response times of both Db4o and Informix increase as the number of objects in the databases increases. Informix's performance is more consistent compared to Db4o during the experiment.

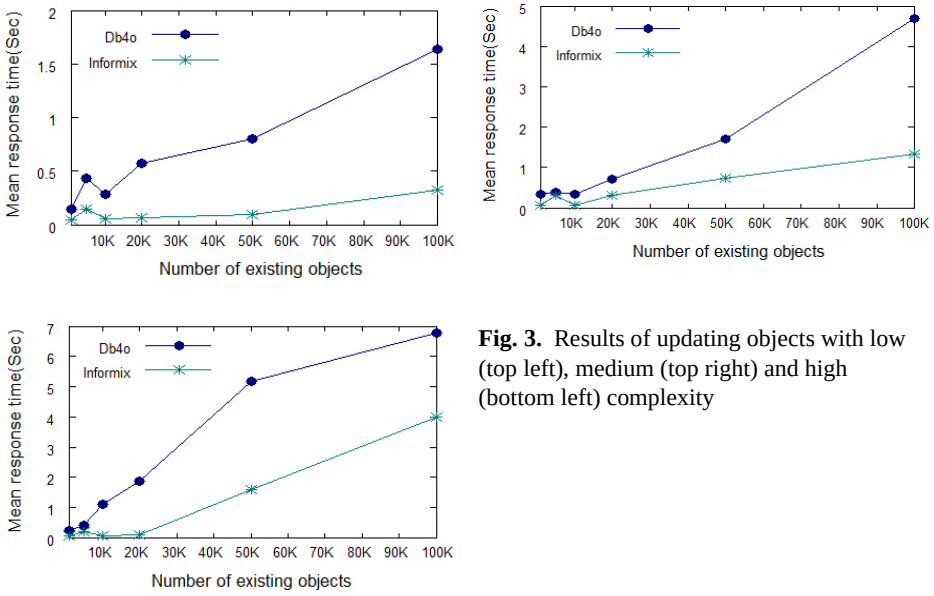

Fig. 3. Results of updating objects with low (top left), medium (top right) and high (bottom left) complexity

As Fig. 3 shows, Informix's response time is less than that of Db4o for updating objects with medium complexity. Informix has a better performance while less than 5,000 objects exist. The two DBMSs have nearly the same performance while 5,000 objects exist but, after this point, again the Informix has a better performance. Informix's performance is consistent while more than 20,000 objects exist. After 50,000 objects, the difference between their performances becomes considerable.

The results for objects with high complexity shows that Informix performs the update operation faster than Db4o because the response time of Informix is less than Db4o's throughout the experiment. The response times are very close while less than 5,000 objects exist in the databases. After this point, the difference between their performances becomes considerable.

4.8 Object Lookup

The aim of this experiment is to determine whether Informix or Db4o has a better performance for the lookup operation. In this experiment just one project object was looked up as a result of the look up query. For Staff and Department objects when different quantities of these objects exist in the database, different number of these objects were looked up but the number of returned objects for Db4o and Informix is nearly the same. The results of the Object Lookup experiment for objects with low, medium and high complexity with both Db4o and Informix DBMSs while they hold different quantities of these objects are shown in Fig. 4.

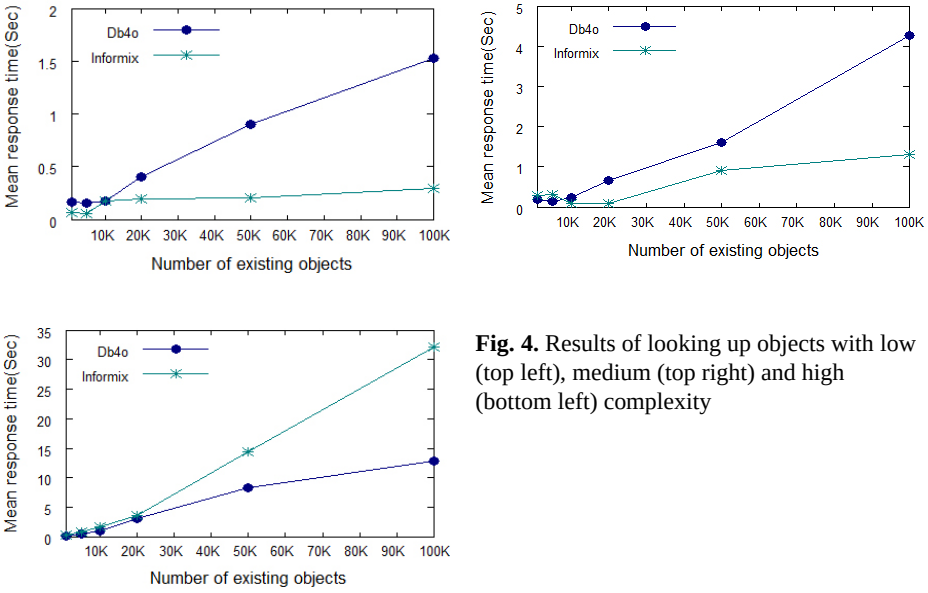

Fig. 4. Results of looking up objects with low (top left), medium (top right) and high (bottom left) complexity

As the results for object with low complexity show, the response time of Informix is less than that of Db4o while less than 10,000 objects exist in the DBMSs. The two DBMSs have the same performance at 10,000. For the rest of the experiment, increasing the number of objects increases Db4o's response time significantly. As Fig. 4 shows, the Informix's performance is more consistent than that of Db4o in looking up object with low complexity.

The results for object with medium complexity show that Db4o's response time is less than that of Informix while less than 8,000 objects exist in the databases. As the number of objects increases, Db4o's response time increases. After 8,000 objects Db4o's performance is worse than Informix's. Informix's performance is the best while between 10,000 and 20,000 objects exist. The results show that the performance of Informix is better than Db4o for looking up object with medium complexity.

As Fig. 4 shows, Db4o's response times are less than those of Informix for looking up objects with high complexity. There is only a tiny difference between the response times of the two DBMSs while less than 20,000 objects exist. For the rest of experiment, as the number of objects increases, the performance of Db4o becomes better compared to Informix. Overall, the results show that the Db4o is better than Informix for looking up object with high complexity.

4.9 Object Deletion

The aim of this experiment is to determine whether Informix or Db4o has a better performance for the delete operation. In this experiment one project object has been deleted as result of delete operation. For Staff and Department objects when different quantities of these objects exist in the database, different number of objects has been deleted but the number of deleted objects for Db4o and Informix is nearly the same.

The results of the Object Deletion experiment for objects with low, medium and high complexity with both Db4o and Informix DBMSs while they hold different quantities of these objects are shown in Fig. 5.

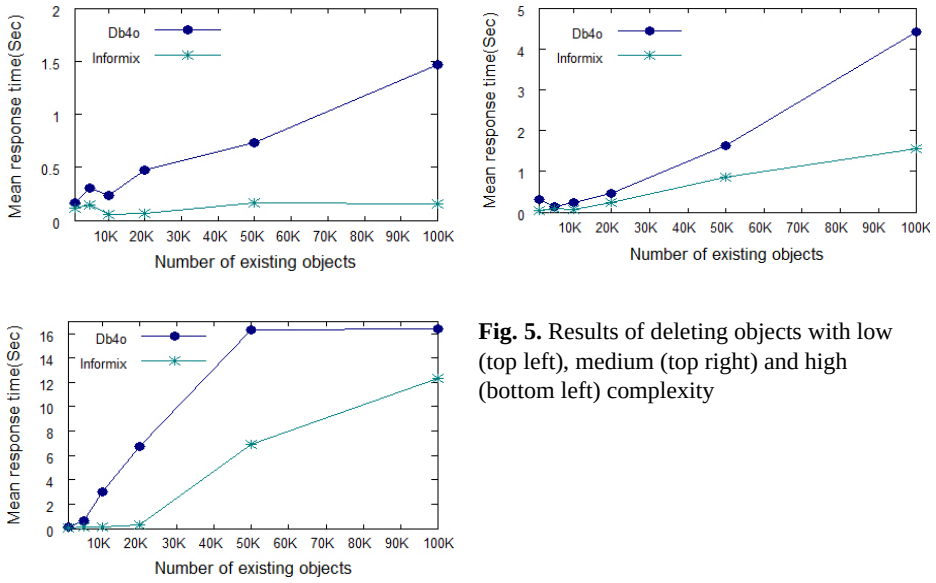

Fig. 5. Results of deleting objects with low (top left), medium (top right) and high (bottom left) complexity

As Fig. 5 shows, Informix's response time for deleting an object with low complexity is less than that of Db4o. Both DBMSs have similar response times up to 10,000 objects. After this point, the difference in their performance increases as the number of objects increases. Informix's performance is more constant compared to Db4o's for delete operation on objects with low complexity.

The results for medium complexity (see Fig. 5 top right) show that the performance of Informix is better than that of Db4o. Db4o's response time for deleting an object while 5,000 objects exist is less than when 1,000 objects exist. Db4o's response time starts increasing when more than 5,000 objects exist but the corresponding number of objects for Informix is 10,000.

As Fig. 5 shows, throughout the experiment, Informix's response time is less than that of Db4o for deleting objects with high complexity. Informix's response time is very low while less than 20,000 objects exist but after this point its response time increases considerably. Although Db4o's response time is low while less than 5,000 objects exist, its response time increases after this point until 50,000 objects; after which point its response time remains the same for the rest of the experiment.

The results of all empirical experiments are summarized in Table 1. For each experiment it shows which DBMS is better in terms of performance according to the complexity of object. For example, it shows that Db4o has better performance for inserting objects with low complexity; Informix is better in terms of performance for updating objects with medium complexity and so on.

Table 1. Summary of the empirical experiments' results

Empirical experiment	Object complexity		
	Low	Medium	High
Object Insertion	Db4o	Db4o	Db4o
Object Modification	Informix	Informix	Informix
Object lookup	Informix	Informix	Db4o
Object Deletion	Informix	Informix	Informix

According to Table 1, Informix has better performance for modifying, looking up and deleting objects with low complexity. Db4o is just better than Informix in terms of performance for inserting this kind of object. The results for fundamental database operations on objects with medium complexity are the same as objects with low complexity. For objects with high complexity, the results are different; Db4o has a better performance than that of Informix for inserting and looking up objects with high complexity while Informix has a better performance for modifying and deleting this kind of objects.

5 Conclusions and Future Work

The findings from this work suggest that the complexity of the object in use affects the performance of object and object relational DBMSs. The performance of the object relational DBMS is better than the object DBMS for fundamental database operations with the exception of insert operations for objects with low and medium complexity. Increasing the level of object's complexity affects the performance of object relational DBMS. For objects with high complexity, in addition to insert operation, the object DBMS has better performance for the look up operation compared to the object relational DBMS.

The findings suggest that system developers should consider the following factors when selecting a DBMS for persisting objects: 1) The complexity of the object in use; 2) The database operations that the system will perform most frequently. For example, if a system uses objects that are mainly highly complex and it performs a lot of look up operations then this research suggests that an ODBMS is more efficient than ORDBMS as a mechanism for persisting objects.

Due to limited time, this work focused on the performance of object and object relational DBMSs for fundamental database operations such as Insert, Update, look up and Delete. The following could be the subject to further studies.

First of all, the performance analysis of ODBMSs and ORDBMSs for fundamental database operations on objects that have behaviour (methods). In reality objects have behaviour and adding methods to objects may impact the performance of DBMSs. Also the performance analysis on Binary Large OBjects (BLOBs) and Character Large OBjects (CLOBs) could be evaluated and compared for these two database technologies. With the rise in popularity of image, audio and multimedia

databases, further research is required to determine which one of ODBMSs and ORDBMSs have better performance for database operations on these kinds of objects.

Secondly, in addition to the fundamental database operations, the performance of object and object relational DBMSs for complex queries involving two or more objects could be evaluated and compared. Today's systems are more complex than before and further research is required to determine which one of ODBMSs and ORDBMSs have better performance for complex queries.

Finally, other object and object relational database systems could be taken into account in the comparison. Evaluating the performance of other database products, would make the results more precise and realistic.

6 References

1. Anderson, T. L., Berre, A. J., Mallison, M., Porter, H. H., and Schneider, B.: The HyperModel Benchmark. In: 2nd Int. Conf. on Extending Database Technology (1990)
2. Brown, P.: Introduction to Object-Relational Database Development. Prentice Hall, USA (2001)
3. Carey, D. J., DeWitt, D., and Naughton, J. F.: The OO7 Benchmark. In: Proceeding of the ACM SIGMOD Int. Conf. on Management of Data, pp. 12—21. Washington DC (1993)
4. Carey, M. J., Dewitt, D. J., Naughton, J. F.: The OO7 Benchmark. CS Tech Report, University of Wisconsin-Madison, April (1993b)
5. Carey, M. J., Dewitt, D. J., Naughton, J. F.: A Status Report on the OO7 OODBMS Benchmarking Effort. In: Proceedings of the Ninth Annual Conf. on Object-Oriented Programming Systems, Language, and Applications, pp. 414—426. ACM Press, New York (1994)
6. Cattell, R.G.G., Skeen, J.: Object Operations Benchmark. ACM Transactions on Database Systems, Vol. 17, No. 1, 1-31 (1992)
7. Chaudhri, Akmal B., McCann, Julie A., and Osmon, P:. A Performance Study of Object Database Management Systems. Theory and Practice of Object Systems, Vol. 5(4), 263-279 (1999)
8. Gorla N.: An Object-Oriented Database Design for Improved Performance. Journal of Data & Knowledge Engineering. 37 117—138 (2001)
9. Hohenstein, U., Volkmar, P., Rainer, H.: Evaluating the Performance of Object-Oriented Database Systems by Means of a Concrete Application. Theory and Practice of Object Systems, Vol. 5(4), 249--261 (1999)
10. Ritchie, C.: ed. Database Principles and Design. Cengage Learning EMEA, London (2008)
11. Van zyl, P., Derrick, G., Kourie and Boake, A.: Comparing the Performance of Object Databases and ORM tools. Proceeding of South African Institute for Computer Scientists and Information Technologists, pp. 1-11 (2006)