A study for performance comparison of different in-memory databases

Conference Paper · October 2013

DOI: 10.1109/ICAICT.2013.6722739

4 authors, including:

Sule Gunduz Oguducu
Istanbul Technical University
68 PUBLICATIONS **996** CITATIONS

SEE PROFILE

A study for Performance Comparison of Different In-Memory Databases

Şule Gündüz Öğüdücü*, Mehmetcan Gayberi†, Erhan Akpınar†, Hakan Kutluay†
*Istanbul Technical University
Computer Engineering
Istanbul, Turkey
{sgunduz}@itu.edu.tr
†Idea Teknoloji Çözümleri
Istanbul, Turkey
{mehmetcan.gayberi, erhan.akpinar, hakan.kutluay}@ideateknoloji.com.tr

***Abstract*—In-memory databases have been researched for over three decades now. Since a major trend in OLAP systems, especially in Business Intelligence applications, is right and real time computing, in-memory databases may be an efficient solution to provide such requirements. There are several in-memory database systems on the market today. In several studies, the performance of a traditional disk resident database and a selected in-memory database is compared. However, there is a lack of a comparative study that evaluates different in-memory databases in the context of analytical applications. In this empirical study we compared performance of different in-memory databases. For this purpose, we selected Timesten, Altibase, solidDb and SQLite. Then we evaluated them both on Linux and Windows systems in terms of query response time using queries that need to access different amounts of data and are of different level of complexity. Our results show that, Altibase product gives the best results with Linux operating systems and TimesTen product performs much better than other in-memory products under the same conditions.**

I. Introduction

Database systems are indispensable for all kind of today's organizations in order to store and process data in modern business applications. Reporting tools are also needed to make critical business decisions. However, creating effective reporting applications is a cumbersome and time consuming process due to the data distributed among multiple databases. With the increasing demands of complex business applications, the transaction processing part of databases, so-called Online Transaction Processing (OLTP), are designed and tuned to suit the various requirements of these applications. Since the operational databases of enterprises are finely tuned to support known OLTP workloads, executing complex queries against the operational databases to generate reports for decision support systems would result in unacceptable performance. Besides, decision support systems usually perform queries on historical data in order to understand trends and make prediction. These systems also use consolidated data to generate custom reports. The data warehouses support Online Analytical Processing (OLAP) systems which were developed to satisfy the requirement of enterprises to analyze their data in order to help them making business decisions. Today, most enterprises usually move operational data into data warehouses and then prepare for predefined reports in order to gain new analytical efficiencies.

A data warehouse contains data from operational systems, but is physically separate and has its own database which is synchronized on a regular basis with data from source databases. However, this system has a limitation in doing real-time analytics where the analytical queries are posed against a copy of the data in the OLAP system that may not include the latest transactions in the operational databases. Besides, to support complex analyses and visualization, the data in the data warehouses are modeled in special materialized data structures, called cubes. Such cubes are based on a fixed number of hierarchical dimensions. A cube can create only a particular set of reports. If a new type of report is needed, which requires other dimensions, then a new data cube has to be created or existing ones have to be extended which results in a growth of the amount of data stored in data warehouse. This causes both performance and maintenance problems. These systems allow organizations to generate predefined reports. Although todays business intelligence, or decision support, applications give companies a more flexible way, they still suffer from performance issues when dealing with large volumes of data. In memory databases may be a solution for these problems where no more pre-computed materialized data structures are needed to generate reports and the database is always up to date reflecting the current state of each business transaction.

Main memory or in-memory databases existed since the1980's [7] are databases that are designed to store all the data in main memory. Thanks to the absence of disk I/O operations, in-memory databases provide significant performance improvements by permitting fast query response times. This yields to an increase in the number of companies offering in-memory technology, such as Oracle Exadata X3[1] and SAP HANA[2]. Oracle Exadata X3 is an integrated platform with the database. SAP HANA also combines database and information processing capabilities in one system.

While in-memory database technology promises faster and more flexible data analysis when generating ad-hoc reports, different in-memory databases may result in different performance in terms of response time. Although there are several studies for testing the performance of an in-memory database [6], they have used different hardware platforms, different

[1] http://www.oracle.com/us/products/database/exadata/
[2] http://www.saphana.com/

CPUs, memory and disk drives. The experiments have been also performed on databases of different sizes. So, it is difficult to compare different in-memory databases because of such differences. In this paper, we focus on determining an in-memory database which provides the best performance for reporting applications.

To this end, we conducted experimental studies on different in-memory databases using queries in different complexity levels to generate reports. Four complexity levels were tested: high level of complexity refers to queries which consist of `OUTER JOIN`, `ORDER BY` and `GROUP BY` statements, medium level of complexity was achieved by queries that includes `OUTER JOIN` and `ORDER BY` statements, simple queries have only `SELECT` statements and simple joins that only make listing operations and basic queries don't include `OUTER JOIN` statements, these queries consist of simple joins, `GROUP BY` and `ORDER BY` statements. Basic queries are implemented according to queries which are frequently submitted by users.

We conducted extensive experiments to evaluate the performance of different in-memory databases running on Linux and Windows systems. Four in-memory database products have been evaluated in the experimental part of this study: Oracle TimesTen[3], SQLite[4], IBM solidDB[5], Altibase HDB[6] for Windows and XDB[7] for Linux. In the study, some criteria are structured to select in-memory database products to test. These criteria are configured towards selecting products which could deal with big data sets, continuously developed, commonly used in the sector and could be operated in both Windows and Linux operating systems. In order to compare different products, same hardware was used in the experiments. The experimental results show that generally in-memory database products execution times increase linearly while data size increases. While best results occur in simple complexity queries, medium and high level of complexity queries give worse results than simple level of complexity queries. As expected, basic queries perform better than medium and high level of complexity queries and worse than simple level of complexity queries in general. Most of in-memory database products perform better in Linux operating system than in Windows operating system. It may be the outcome of being able to set related kernel parameters in Linux operating system. However, there are more queries for which several in-memory databases cannot produce any result in a predefined period of time. In both operating system, Oracle TimesTen product looks better than other the products for each complexity level. On the other hand, Altibase XDB product also gives good results in Linux operating system. As a result, Oracle TimesTen product performs much better than other in-memory products under the same conditions.

The rest of the paper is organized as follows; a brief literature review in topics related to this paper is given in Section II. Section III describes the overall system for testing in detail. Section IV gives the results of the experiments and a discussion of these results. Finally, in Section V we conclude the paper and discuss the future work.

[3]http://www.oracle.com/us/products/database/overview/

[4]http://www.sqlite.org/

[5]http://www-01.ibm.com/software/data/soliddb/

[6]http://www.altibase.com/products/hdbtm-hybrid-dbms

[7]http://www.altibase.com/products/xdbtm-data-stream

II. Related Works

There have been several works on the performance comparison of a disk resident database and an in-memory database [10], but there has not been a comparative study of the performance of different in-memory databases. There is a study to show the potential advantages of in-memory databases over traditional relational disk resident databases in terms of speed using SQLite database as in-memory database [6]. It has been shown that the in-memory database results don't summon up on positive side and present varying values for speedup factor. Schaffner et al. [11] aimed to develop a model for characterizing the load on an in-memory column database containing multiple tenants with different request rates. The approach used to model database performance was experimental. Rather than characterizing all constituents of the system and their interactions, they extracted the model from observations of a running system. In that study, they showed how to use this model to predict whether a database instance will be able to meet a given response time goal.

The work of Pagano et al. [9] focuses on implementing a test environment with in-memory relational database management system and row-level data on the cloud. HyperSQL[8] has been used in the test environment. In that study, a basic model with local users and dossiers is used. It is stated that in-memory databases have advantages such as fast transactions, no translations, high reliability and multi-user concurrency. It has been also shown that in-memory database systems perform better than traditional database systems with solid state disks.

Krueger et al. [8] discussed data structures for in-memory database systems and evaluated their performances based on real customer data. Column based and row based in-memory databases and their performances are compared in that study. As a result, with the hardware development, column based storages are shining and in memory column storages is advantageous for enterprise applications with mixed workloads.

III. Test System Details

The purpose of the experiments conducted in this study is to compare the performance of different in-memory databases in terms of speed when creating reports from data that reside on databases. This objective was achieved with the set up of a proper test environment. All the experiments were performed on the same server that has 4 Intel(R) Xeon(R) CPU E7-4820 @ 2.00 Ghz processors, 240 GB Ram and 300 GB hard disk. Test scenarios have been performed in 64-bit Linux Server Gnome 2.28.2 and Windows Server 2008 R2. In-memory database product versions are: Oracle Timesten 11.2.2 x64, Altibase HDB 6.1.1.0.10 x64 for Windows, Altibase XDB 6.1.1 x64 for Linux, IBM solidDb 7.0.100 x64 and SQLite 3.7.17.

As stated before, four types of queries are used according to their complexity level. Each query complexity group has further divided into six different groups based on the number of tables involved in the query. The number of tables involved in a

[8]www.hsqldb.org

query is selected as 2, 4, 6, 8, 10 and 13 which is denoted with the same number after the letter used for the query type in the experimental tables. While simple level queries (denoted with an S in the experimental tables) uses the `SELECT` statement to identify a data set that contains a subset of data, queries in medium level of complexity (represented with M in the experimental tables) include one `ORDER BY` operation, one `OUTER JOIN` operation for $M2$ and $M4$ test queries and two `OUTER JOIN` operations for $M6$, $M8$, $M10$ and $M13$ queries. Queries in high level of complexity (termed C in the experimental tables) involve two `ORDER BY` operations, half of the tables involved in `GROUP BY` operations and two `OUTER JOIN` operations. Basic queries (denoted with an N in the experimental tables) perform only one `ORDER BY` operation for $N2$ and $N4$ queries, two `ORDER BY` operations for rest of queries and half of the number of tables involved `GROUP BY` operations.

Test database consists of eleven fixed-size tables; five of them have 100 and below row count, five tables have a row count ranging from $1,000$ to $3,000$ and one table has $200,000$ rows. The tests also performed on three different data sizes (obtained data for three, six and twelve months) to study the performance differences between the in-memory databases. For this, two tables are added that have different number of rows for each different data size. One of these tables includes $1,000,000$ rows for three months, $2,000,000$ rows for six months and $4,200,000$ rows for twelve months data. The other table consists of $5,500,000$ rows for three months, $12,000,000$ rows for six months and $24,000,000$ rows for twelve months. These two non-fixed tables are always used in all queries of group 6. The table with $200,000$ rows is used in only queries of group 13. Other tables are added to query groups progressively.

A standard index structure for in-memory databases has been prepared within all tests. While primary keys provide unique indexes, non-unique indexes are also used for the columns that exist in `WHERE` clauses. Same structure has been implemented for indexes to avoid causing a difference between in-memory databases.

Cache systems have been deactivated and disregarded for all in-memory database products. Despite cache systems accelerate execution time of queries, the main purpose has been measuring in-memory database performances under same conditions in the scope of study. Differences between products cache systems might cause deviations and it might prevent getting a reasonable result.

IV. Experimental Results

SQLite, IBM solidDB, Altibase HDB/XDB and TimesTen are served as in-memory databases in our tests. As stated on its web site, SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed SQL database engine in the world [4]. The source code for SQLite is in the public domain. When using ":memory" option provided for creating a database connection, SQLite database can then exist purely in memory. It is also a cross platform product both running on Linux and Windows plotforms. IBM solidDB is a complete relational in-memory database management system which combines the high performance of in-memory tables with the nearly unlimited capacity of disk-based tables [3]. The durability is provided by keeping two separate but synchronized copies of database all the times as well as permanent log file stored on disk. Altibase HDB is a hybrid database system similar to IBM solidDB. As pointed out by vendor, developing optimizations, special algorithms and data structures for in-memory databases provide a good performance [1].

Altibase XDB and Oracle TimesTen databases are directly developed towards in-memory system standards. TimesTen is an in-memory, relational database management system which also supports persistence and recoverability properties. It uses a technology which stores all data within the database in RAM at runtime and disks are used for only persistence and recovery [2]. Altibase XDB is also created with a pure in-memory database engine and it uses multi-process and multi-thread structure [5]. At this point it differentiates from TimesTen, as TimesTen does not support multi-processes. According to the vendor announcement, Altibase XDB is 2 to 3 times faster than the leading in-memory database management systems.

In the experiments, response time to the query is measured in seconds for each group of queries in each level. It is assumed that queries time out after 30 minutes of inactivity. A timeout is denoted with $-$ in the experimental tables.

To observe about differences in performance between different in-memory databases for `SELECT` statements when creating reports, we submitted simple level queries to each of the selected in-memory database and measure the response time for the queries. Table I summarizes these results obtained both in Windows and Linux platforms. As can be seen from the table, solidDB outperforms other in-memory databases in Windows platform whereas SQLite works better than the other ones in Linux operating system. Moreover, in Windows SQLite is as well performing as IBM solidDB. Generally, Altibase products HDB for Windows and XDB for Linux produce better execution times than Oracle TimesTen in both operating systems.

The results of the queries of the medium level of complexity are given in Table II. In most of the queries, IBM solidDB in-memory database does not respond within the timeout period. Despite SQLite can produce results for all queries in Windows, this database engine cannot execute most of the queries in Linux within the timeout period. Oracle TimesTen looks performing well in Windows when working on small data sets. With the increase of data size, performance of TimesTen decreases. Yet this performance can be admissible when compared to other in-memory products. Altibase HDB and XDB can response all queries at low execution times. It should be noted that HDB product on Windows has a lower response time for queries that need to access large tables. On the other hand, XDB product distinctly presents a perfect performance in Linux. It performs the best results for all queries in this type of queries.

Table III presents the results for basic queries. SQLite is the only product that can execute all queries in Windows. However, when the other products generate results for the queries, the execution performance of SQLite is not better than the other ones and most of its execution times are also long.

Table I. QUERY TIME FOR SIMPLE LEVEL QUERIES MEASURED IN SECONDS

		WINDOWS						LINUX					
	Database	S2	S4	S6	S8	S10	S13	S2	S4	S6	S8	S10	S13
3-Month Data	TimesTen	0,21	0,67	1,32	2,17	3,21	5,18	0,28	0,70	4,46	5,34	6,61	8,92
	SQLite	**0,00**	0,03	0,02	0,03	**0,03**	**0,05**	**0,01**	**0,02**	**0,02**	**0,17**	**0,17**	**0,17**
	SolidDb	**0,00**	**0,00**	**0,00**	**0,01**	0,04	0,84	0,11	0,33	0,57	0,58	0,78	1,96
	Altibase	0,08	0,13	0,16	0,19	0,20	0,25	0,60	1,15	3,26	4,36	5,39	6,85
6-Month Data	TimesTen	0,18	0,71	3,11	4,15	3,23	5,32	0,27	0,71	8,26	9,18	10,22	12,96
	SQLite	**0,00**	0,03	0,03	0,03	**0,03**	**0,03**	**0,01**	**0,15**	**0,16**	**0,16**	**0,16**	**0,17**
	SolidDb	**0,00**	**0,00**	**0,00**	**0,01**	0,05	0,75	0,12	0,34	0,55	0,56	0,80	1,73
	Altibase	0,08	0,09	0,14	0,16	0,19	0,23	1,25	1,81	5,20	6,29	5,90	8,64
12-Month Data	TimesTen	0,37	0,70	6,59	6,17	3,37	5,42	0,27	0,70	15,92	14,08	17,85	21,57
	SQLite	0,02	0,02	0,03	0,03	**0,03**	**0,05**	**0,01**	**0,15**	**0,16**	**0,16**	**0,16**	**0,17**
	SolidDb	**0,00**	**0,00**	**0,01**	**0,01**	0,05	0,81	0,12	0,34	0,49	0,62	0,83	2,62
	Altibase	0,08	0,14	0,16	0,17	0,22	0,25	2,58	3,19	8,38	9,20	10,67	12,53

Table II. QUERY TIME OF QUERIES FOR MEDIUM LEVEL OF COMPLEXITY MEASURED IN SECONDS

		WINDOWS						LINUX					
	Database	M2	M4	M6	M8	M10	M13	M2	M4	M6	M8	M10	M13
3-Month Data	TimesTen	**14,53**	63,06	**76,79**	**81,18**	**82,74**	**87,38**	22,46	70,77	64,40	133,56	72,31	74,55
	SQLite	57,30	185,75	329,00	406,05	464,81	502,36	50,01	-	-	-	-	-
	SolidDb	132,46	-	-	-	-	-	125,07	-	-	-	-	-
	Altibase	17,09	**45,20**	191,36	107,62	102,97	110,44	**4,68**	**19,09**	**25,54**	**35,33**	**45,75**	**39,16**
6-Month Data	TimesTen	**32,73**	143,57	**167,68**	**156,81**	263,00	**195,83**	59,76	163,50	145,56	302,13	165,42	163,36
	SQLite	121,09	367,69	692,52	852,98	983,08	1057,41	107,69	-	-	-	-	-
	SolidDb	297,99	-	-	-	-	-	292,38	-	-	-	-	-
	Altibase	40,86	**132,75**	287,49	203,48	**232,21**	254,01	**10,18**	**40,34**	**55,02**	**74,09**	**96,02**	**77,92**
12-Month Data	TimesTen	76,35	320,82	-	**336,00**	-	-	111,40	346,19	517,31	682,50	349,93	340,34
	SQLite	242,47	730,89	1372,63	1713,22	**1932,11**	-	209,12	-	-	-	-	-
	SolidDb	714,60	-	-	-	-	-	530,41	-	-	-	-	-
	Altibase	**70,10**	**226,33**	**493,42**	409,47	-	**520,21**	**21,45**	**88,54**	**113,07**	**151,81**	**197,26**	**161,32**

Table III. QUERY TIME FOR BASIC QUERIES MEASURED IN SECONDS

		WINDOWS						LINUX					
	Database	N2	N4	N6	N8	N10	N13	N2	N4	N6	N8	N10	N13
3-Month Data	TimesTen	16,79	47,56	**70,47**	**78,70**	**81,70**	**85,90**	15,01	47,28	87,66	126,29	**60,82**	**61,94**
	SQLite	63,64	194,69	298,47	336,09	379,36	434,38	77,32	-	-	-	-	-
	SolidDb	26,85	394,82	-	-	-	-	67,92	-	-	-	-	-
	Altibase	**7,67**	**37,80**	100,99	129,94	147,92	174,09	**8,48**	**36,55**	**69,35**	**93,16**	118,95	146,08
6-Month Data	TimesTen	21,01	88,27	**146,19**	**152,69**	**177,19**	**191,36**	32,78	107,36	193,61	280,97	**138,03**	**134,40**
	SQLite	142,63	418,48	631,75	713,94	812,78	922,03	158,30	-	-	-	-	-
	SolidDb	64,27	-	-	-	-	-	160,00	-	-	-	-	-
	Altibase	**16,55**	**83,58**	250,35	320,56	346,05	395,36	**19,54**	**81,92**	**154,51**	**201,81**	255,29	319,20
12-Month Data	TimesTen	**27,62**	168,64	-	-	-	-	72,83	234,51	436,45	677,27	**321,02**	**312,01**
	SQLite	288,52	841,19	**1271,50**	**1451,47**	**1618,59**	**1992,86**	317,50	-	-	-	-	-
	SolidDb	129,25	-	-	-	-	-	277,45	-	-	-	-	-
	Altibase	37,21	**87,90**	-	-	-	-	**40,66**	**167,79**	**327,04**	**435,69**	538,90	689,35

Table IV. QUERY TIME OF QUERIES FOR HIGH LEVEL OF COMPLEXITY MEASURED IN SECONDS

		WINDOWS						LINUX					
	Database	C2	C4	C6	C8	C10	C13	C2	C4	C6	C8	C10	C13
3-Month Data	TimesTen	**5,21**	**38,59**	**64,51**	**69,62**	**79,06**	**80,62**	13,64	**35,28**	**50,73**	127,64	168,02	**61,78**
	SQLite	64,06	-	319,17	368,67	376,88	482,36	75,33	-	-	-	-	-
	SolidDb	44,82	457,89	-	-	-	-	77,84	-	-	-	-	-
	Altibase	8,92	49,53	191,35	111,03	138,88	111,87	**8,61**	39,69	55,77	**68,28**	**76,79**	83,96
6-Month Data	TimesTen	**11,44**	**85,27**	**166,39**	**147,53**	**167,13**	**180,51**	31,32	109,57	**111,45**	281,38	369,79	**140,88**
	SQLite	144,19	-	672,86	791,14	811,55	1020,03	161,29	-	-	-	-	-
	SolidDb	196,14	-	-	-	-	-	175,15	-	-	-	-	-
	Altibase	19,14	111,78	309,66	271,34	325,85	268,85	**18,80**	**84,24**	125,60	**150,17**	**169,98**	184,30
12-Month Data	TimesTen	**27,34**	**180,00**	-	-	-	-	81,29	231,75	516,86	675,20	843,26	450,96
	SQLite	290,47	-	**1348,34**	**1598,86**	**1605,03**	-	325,46	-	-	-	-	-
	SolidDb	437,52	-	-	-	-	-	310,65	-	-	-	-	-
	Altibase	36,27	231,64	-	-	-	-	**39,57**	**182,52**	**276,70**	**319,73**	**358,25**	**389,38**

IBM solidDB can produce results only when the data to be accessed is small. Altibase HDB performs well in Windows unless data set is large. When data set gets larger, Altibase HDB cannot response some queries. While Oracle TimesTen looks performing best in Windows, Altibase XDB generally performs better in Linux. Still, TimesTen presents a reasonable performance.

The results of queries for high level of complexity are shown in Table IV. As can be seen in the table, IBM solidDB cannot response most of the queries. As a results of these experiments, we come to the conclusion that IBM solidDB is inadequate as an in-memory database with large data sets. Strangely, SQLite can execute most of queries which other databases cannot execute in Windows. However, in Linux SQLite cannot execute most of high complexity queries in contrast to Windows. Oracle TimesTen looks performing better than Altibase HDB on Windows platform in general. In Linux tests, Altibase XDB clearly shows a better performance against to Oracle TimesTen, especially when data sets get larger.

In general, SQLite and solidDB give good results only if queries are not complex. SolidDB is a hybrid database management system and it is not developed as a pure in-memory database. Thus, its performance has been behind of other products. The database engine developed in SQLite does not require a server or special configurations and contains all data in a single file with in-memory option. These properties might be considered as advantages, yet most of these properties become significant disadvantages when the data to be accessed to execute the query get larger. SQLite has given perfect results if data set is not large and query is not complex. However, as well as data set gets larger and queries get more complex, the performance of SQLite decreases dramatically. The results of these experiment were also confirming the negative effect of `GROUP BY`, `ORDER BY` and `FULL OUTER JOIN` statements on in-memory databases. TimesTen is clearly the best performing in Windows operating system. In Linux operating system, Altibase XDB product is separated from other in-memory databases with its perfect performance. While comparing the operating systems, in Linux there are fewer queries that could not be responded for all products. Moreover, in Linux performance tests, all products give better results on most of queries when compared to Windows tests. As a general result, Altibase XDB product gives the best results with Linux operating systems.

V. Conclusion

In-memory databases are offered as a new type of database management system where all the data are hold in main memory to accelerate data storing and retrieving processes. According to vendors, since in-memory databases do not use disk for data processing, they provide performance gains. For this, in-memory databases have been employed in performance-sensitive applications such as telecommunication, financial services and Business Intelligence. Analytical and financial planning applications, called OLAP systems, were developed to meet the needs of large enterprises to analyze their data. Traditionally, operational and analytical systems of enterprises are separated in order to provide adequate query response times both in operational and analytical part. However, this brings another challenge: the data in the operational part should be transferred into the analytical part. Depending on the steps of this transfer, there may be a delay ranging from several hours to days between data being entered into the operational system until being available for reporting. This delay has a particular effect on performance when applications need to do both operational processing and analytics. Thus, in-memory databases may be a solution to handle this problem.

Although in-memory technologies provide high performance, scalability and flexibility to Business Intelligence tools, a specific problem in in-memory databases is query optimization. Since the query costs in in-memory database systems depend on fuzzy factors such as CPU execution cost of a routine due to the lack of I/O as dominant cost factor, query optimization is much more difficult in these systems. There are several in-memory technologies aimed at analytical exploitation of data collected in the enterprises. However, they differ in query performing. In this paper, our goal was to conduct an empirical study to evaluate the performance of different in-memory database systems in terms of query response times when generating analytical reports. SQLite, IBM solidDB, Altibase HDB and TimesTen are served as in-memory databases in our tests both on Windows and Linux platforms. Queries of different level of complexities and databases of different sizes are prepared in order to compare in-memory databases in terms of speed. Our experimental results show that, Altibase product gives the best results with Linux operating systems and TimesTen product performs much better than other in-memory products under the same conditions.

An interesting future research direction may be to study an appropriate data structure for in-memory databases to further accelerate the process during data retrieval for generating different kinds of analytical reports.

References

[1] Altibase hdb: In-memory dbms with hybrid architecture. http://www.altibase.com/products/hdbtm-hybrid-dbms. Accessed: 2013-06-20.

[2] Extreme performance using oracle timesten in-memory database. http://www.oracle.com/technetwork/products/timesten/overview/wp-timesten-tech-132016.pdf. Accessed: 2013-06-20.

[3] Ibm soliddb version 7.0 getting started guide. ftp://public.dhe.ibm.com/software/data/soliddb/info/7.0/man/GettingStartedGuide.pdf. Accessed: 2013-06-20.

[4] Sqlite. http://www.sqlite.org/mostdeployed.html. Accessed: 2013-06-20.

[5] Altibase. *Altibase XDB 6.1.1 Release Note*, May 2013.

[6] S. Choinyambuu. In memory database: Performance evaluation based on query time. In *Seminar Database Systems*, pages 1–15, December 2010.

[7] H. Garcia-Molina and K. Salem. Main memory database systems: an overview. *Knowledge and Data Engineering, IEEE Transactions on*, 4(6):509–516, 1992.

[8] J. Krueger, M. Grund, M. Boissier, A. Zeier, and H. Plattner. Data structures for mixed workloads in in-memory databases. In *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference*, pages 394 – 399, Seoul, November - December 2010. ICCIT.

[9] F. Pagano and D. Pagano. Using in-memory encrypted databases on the cloud. In *Securing Services on the Cloud (IWSSC), 2011 1st International Workshop*, pages 30–37, Milan, September 2011. IWSSC.

[10] F. Raja, M. Rahgozar, N. Razavi, and M. Siadaty. A comparative study of main memory databases and disk-resident databases. 2006.

[11] J. Schaffner, B. Eckart, D. Jacobs, C. Schwarz, H. Plattner, and A. Zeier. Predicting in-memory database performance for automating cluster management tasks. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference*, pages 1264 – 1275, Hannover, April 2011. IEEE.